

Fujitsu Enterprise Postgres 16 for Kubernetes

User's Guide

Linux

J2UL-UG16-02ENZ0(00)
September 2024

Preface

Purpose of this document

This document describes system configuration, design, installation, setup, and operational procedures of the Fujitsu Enterprise Postgres for Kubernetes.

Intended readers

This document is intended for people who are:

- Considering installing Fujitsu Enterprise Postgres for Kubernetes
- Using Fujitsu Enterprise Postgres for Kubernetes for the first time
- Wanting to learn about the concept of Fujitsu Enterprise Postgres for Kubernetes
- Wanting to see a functional overview of Fujitsu Enterprise Postgres for Kubernetes

Readers of this document are also assumed to have general knowledge of:

- Linux
- Kubernetes
- Containers
- Operators

Structure of this document

This document is structured as follows:

[Chapter 1 System Requirements](#)

Describes the system requirements.

[Chapter 2 Overview of Operator Design](#)

Describes an overview of the operator design.

[Chapter 3 Operator Installation](#)

Describes the installation of the FEP operator.

[Chapter 4 Deployment Container](#)

Describes container deployment.

[Chapter 5 Post-Deployment Operations](#)

Describes the operation after deploying the container.

[Chapter 6 Maintenance Operations](#)

Describes the maintenance operation after deploying the container.

[Chapter 7 Abnormality](#)

Describes the actions to take when an error occurs in the database or an application.

[Appendix A Quantitative Values and Limitations](#)

Describes the quantitative values and limitations.

[Appendix B Adding Custom Annotations to FEPCluster Pods using Operator](#)

Describes instructions for adding custom annotations to a FEPCluster pod.

[Appendix C Utilize Shared Storage](#)

Describes how to build a FEPCluster when using shared storage.

Appendix C Utilize Shared Storage

Describes how to build a FEPCluster when using shared storage.

Appendix D Key Management System Available for Transparent Data Encryption

Describes the key management system available for transparent data encryption.

Abbreviations

The following abbreviations are used in this manual:

Full Name	Abbreviations
Fujitsu Enterprise Postgres for Kubernetes	FEP
Fujitsu Enterprise Postgres	
Vertical Clustered Index	VCI
Transparent Data Encryption	TDE
Point in time recovery	PITR
Custom Resource	CR
Custom Resource Definition	CRD
Persistent Volume	PV
Persistent Volume Claim	PVC
Universal Base Image	UBI
OpenShift Container Platform	OCP
Mutual TLS	MTLS

Abbreviations of manual titles

The following abbreviations are used in this manual as manual titles:

Full Manual Title	Abbreviations
Fujitsu Enterprise Postgres for Kubernetes Release Notes	Release Notes
Fujitsu Enterprise Postgres for Kubernetes Overview	Overview
Fujitsu Enterprise Postgres for Kubernetes User's Guide	User's Guide
Fujitsu Enterprise Postgres for Kubernetes Reference	Reference

Trademarks

- Linux is a registered trademark or trademark of Mr. Linus Torvalds in the U.S. and other countries.
- Red Hat and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.
- S/390 is a registered trademark of International Business Machines Corporation in the United States or other countries or both.

Other product and company names mentioned in this manual are the trademarks or registered trademarks of their respective owners.

Export restrictions

If this document is to be exported or provided overseas, confirm legal requirements for the Foreign Exchange and Foreign Trade Act as well as other laws and regulations, including U.S. Export Administration Regulations, and follow the required procedures.

Issue date and version

Edition 2.0: September 2024 Edition 1.0: June 2024

Copyright

Copyright 2021-2024 Fujitsu Limited

Contents

Chapter 1 System Requirements.....	1
1.1 Components Embedded.....	1
1.2 CPU.....	1
1.3 Supported Platform.....	1
1.4 Collaboration Tool.....	2
Chapter 2 Overview of Operator Design.....	3
2.1 Design Task.....	3
2.2 System Configuration Design.....	3
2.2.1 Server Configuration.....	3
2.2.2 User Account.....	5
2.2.3 Basic Information of the Container.....	5
2.3 Design Perspective for Each Feature.....	10
2.3.1 Deployment.....	11
2.3.2 High Availability.....	11
2.3.3 Configurable Volume per Cluster.....	11
2.3.3.1 Disk Space Management.....	13
2.3.3.1.1 Increasing Disk Space.....	13
2.3.3.1.2 Reducing Disk Usage.....	14
2.3.3.2 Configuring PVC Auto Expansion.....	14
2.3.4 Deploying Pgpool-II and Connect to FEPCluster from Operator.....	15
2.3.5 Scheduling Backup from Operator.....	15
2.3.5.1 Important Setting Items.....	16
2.3.5.2 Parameters that cannot be Set.....	16
2.3.5.3 Restricted Parameters.....	19
2.3.5.4 About Sections in the Config File.....	19
2.3.6 Perform PITR and Latest Backup Restore from Operator.....	19
2.3.7 FEP Unique Feature Enabled by Default.....	20
2.3.8 Monitoring & Alert (FEPExporter).....	20
2.3.8.1 FEPExporter Custom Resource.....	21
2.3.8.2 Change to FEPCluster CR - metrics user.....	21
2.3.8.3 FEPExporter CR auto-create for FEPCluster.....	21
2.3.9 Scaling Replicas.....	21
2.3.9.1 Change to FEPCluster CR - auto scale out.....	22
2.3.10 Disaster Recovery.....	22
2.3.11 Transparent Data Encryption Using a Key Management System.....	23
2.3.12 Database Role Management.....	23
2.3.12.1 Creating Roles Related to Database Operation.....	24
2.3.12.1.1 Quarantine SUPERUSER.....	24
2.3.12.1.2 Database Administrator Role.....	24
2.3.12.1.3 Confidential Administrator Role.....	24
2.3.12.2 Expiration Management of Database Roles with Login Privileges.....	25
2.3.12.2.1 Policy-based Password Operation.....	25
2.3.12.2.2 Password Operation with the VALID UNTIL Clause.....	26
2.3.13 User Synchronization Using ldap2pg.....	27
2.3.13.1 How to Update FEPCluster CR for ldap2pg.....	27
2.3.14 Fixed Statistics.....	32
Chapter 3 Operator Installation.....	34
3.1 Using the OperatorHub.....	34
3.1.1 Pre-requisite.....	34
3.1.2 Deploying Operator.....	35
3.1.3 Upgrading Operators.....	37
3.2 Using the Helm Chart.....	37
3.2.1 Deploying Operator.....	37
3.2.2 Upgrading Operators.....	37

3.3 Using the Rancher UI.....	38
3.3.1 Pre-requisite.....	38
3.3.2 Register Helm Chart Repository.....	39
3.3.3 Deploying Operator.....	41
3.3.4 Upgrading Operators.....	42
3.4 Implement Collaborative Monitoring Tools.....	43
3.4.1 Implement GAP Stack.....	43
3.4.2 Implement Elastic Cloud on Kubernetes.....	43
3.4.2.1 Deploy ECK Operator.....	43
3.4.2.2 Deploy Elasticsearch Cluster.....	45
3.4.2.3 Deploy Enterprise Search.....	46
3.4.2.4 Deploy Kibana.....	46
3.4.2.5 Expose Kibana using OpenShift Route.....	47
3.4.2.6 Login to Kibana.....	49
3.5 Implement Client.....	50
Chapter 4 Deployment Container.....	51
4.1 Deploying FEPCluster using Operator.....	51
4.2 Deploy a Highly Available FEPCluster.....	56
4.3 Deploying FEPExporter.....	57
4.4 FEPExporter in Standalone Mode.....	59
4.5 Deploying FEPClusters with Cloud-based Secret Management.....	61
4.5.1 Installing Secret Store CSI Driver Using Helm Charts.....	62
4.5.2 Installing and Configuring Azure Provider for Secret Store CSI Driver.....	62
4.5.2.1 Install Azure Provider drivers using helm chart.....	62
4.5.2.2 Create Secret to Access Azure Key vault.....	62
4.5.2.3 Store Secret in Azure Key Vault.....	63
4.5.2.4 Store Certificate in Azure Key Vault.....	63
4.5.3 Installing and Configuring AWS Provider for Secret Store CSI Driver.....	64
4.5.3.1 Install AWS Provider drivers using helm chart.....	64
4.5.3.2 Setup EKS cluster along with service account with necessary IAM roles and permission to access Secret Manager.....	65
4.5.3.3 Store Secret in AWS Secrets Manager.....	65
4.5.3.4 Store Cert in AWS Secrets Manager.....	65
4.5.4 Installing GCP Provider for Secret Store CSI Driver.....	65
4.5.4.1 Install GCP Provider drivers using Kubernetes.....	65
4.5.4.2 Configure GCP secret manager and IAM.....	66
4.5.4.3 Create Secret to access GCP Secret manager.....	66
4.5.4.4 Store secret in GCP Secret manager.....	66
4.5.4.5 Store Cert in GCP Secret manager.....	66
4.5.5 Installing HashiCorp Vault Provider for Secret Store CSI Driver.....	66
4.5.5.1 Install HashiCorp Provider drivers using helm chart.....	66
4.5.5.2 Configure Kubernetes Authentication for HashiCorp Vault.....	67
4.5.5.3 Store Secret in HashiCorp Vault.....	67
4.5.5.4 Store Cert in HashiCorp Vault.....	67
4.5.5.5 Create policy and role to access the secrets from HashiCorp Vault.....	67
4.5.6 Configuring FEPCluster to use Provider for Secret Store Driver.....	67
4.5.6.1 Azure Provider for Secret Store CSI Driver.....	67
4.5.6.2 AWS Provider for Secret Store CSI Driver.....	68
4.5.6.3 GCP Provider for Secret Store CSI Driver.....	69
4.5.6.4 HashiCorp Vault Provider for Secret Store CSI Driver.....	70
4.6 Deploying a customized FEP server container image.....	71
4.6.1 Requirements.....	71
4.6.2 Build custom FEP image with extension.....	71
4.6.3 Adding SQLite Foreign Data Wrapper to FEP Server Container.....	71
4.6.4 Create FEP Cluster with custom image.....	72
4.7 Configuration FEP to Perform MTLS.....	73
4.7.1 When Using an Automatically Generated Certificate.....	73

4.7.1.1 How to Create a Certificate.....	73
4.7.1.2 How to Create a Client Certificate.....	74
4.7.2 When Using Your Own Certificate.....	74
4.7.2.1 Manual Certificate Management.....	75
4.7.2.2 Automatic Certificate Management.....	78
4.7.2.3 Deploy FEPCluster with MTLS support.....	83
4.7.2.4 Configurable Parameters.....	90
4.8 Replication Slots.....	92
4.8.1 Setting Up Logical Replication using MTLS.....	92
4.9 FEP Logging.....	95
4.9.1 FEPLoggging Configuration.....	95
4.9.1.1 FEPLoggging Custom Resources - spec.....	95
4.9.1.1.1 Define fepLogging image.....	97
4.9.1.1.2 Define fepLogging mcSpec.....	98
4.9.1.1.3 Define fepLogging restartRequired.....	98
4.9.1.1.4 Define fepLogging scrapeInterval and scrapeTimeout.....	98
4.9.1.1.5 Define fepLogging elastic.....	98
4.9.1.1.6 Define authSecret for elastic.....	99
4.9.1.1.7 Define fepLogging TLS.....	99
4.9.1.1.8 Define Prometheus TLS.....	99
4.9.2 FEPCluster Configuration.....	100
4.9.2.1 FEP Custom Resources - spec.fep.remoteLogging.....	100
4.9.2.1.1 Define remoteLogging enable and fluentdName.....	101
4.9.2.1.2 Define remoteLogging tls.....	101
4.9.2.1.3 Define remoteLogging image.....	102
4.9.3 FEPLoggging Operations.....	102
4.9.3.1 Log Forwarding to Elasticsearch.....	102
4.9.3.2 Log severity based Alarms/Metrics.....	102
4.9.3.3 Forwarding auditlog to Elasticsearch.....	103
4.9.4 Limitations.....	103
4.10 Configuring pgBadger.....	104
4.10.1 FEP Custom Resources - spec.fep.pgBadger.....	104
4.10.2 Define pgBadger Schedules.....	104
4.10.3 Define pgBadger Options.....	104
4.10.4 Define Endpoint for Uploading Report.....	104
4.10.5 Uploaded File on Web Server.....	107
4.11 Automating Audit Log Operations.....	107
4.11.1 Simplifies Parameter Setting.....	108
4.11.2 Alerting.....	108
4.11.3 Store in Cloud Storage.....	109
4.12 Transparent Data Encryption Using a Key Management System.....	109
4.12.1 Registration of Authentication Information.....	110
4.12.1.1 When Using a KMIP Server.....	110
4.12.1.2 When Using AWS Key Management Service.....	110
4.12.1.3 When using Azure Key Management Service.....	110
4.12.2 Configuring FEPCluster Custom Resources.....	111
4.12.2.1 Define spec.fepChildCrVal.customPgParams.....	111
4.12.2.2 Define spec.fepChildCrVal.sysTde.....	111
4.13 Disaster Recovery in Hot Standby Configuration.....	113
4.13.1 Continuous Recovery Method.....	113
4.13.2 Streaming Replication Method.....	113
4.13.3 Defining a Hot Standby Configuration.....	114
4.13.3.1 Defining a Continuous Recovery Method.....	114
4.13.3.2 Defining a Streaming Replication Method.....	114
4.13.3.3 Defining FEPCluster Custom Resources.....	115
4.14 Enabling Client Authentication with scram-sha-256 Authentication.....	115
4.14.1 Enabling Client Authentication Using scram-sha-256 Authentication in the FEP Server Container.....	115

4.14.1.1 Define spec.fepChildCrVal.customPgParams.....	116
4.14.1.2 Define spec.fepChildCrVal.customPgHba.....	116
4.14.2 Enabling Client Authentication Using scram-sha-256 Authentication in the FEPPgpool2 Container.....	116
4.14.2.1 Creating the Resources Required to Enable scram-sha-256 Authentication.....	116
4.14.2.2 Editing FEPPgpool2 Custom Resources.....	117
4.14.3 Enabling Client Authentication Using scram-sha-256 Authentication on Existing FEP Server and FEPPgpool2 Containers.....	118
4.15 Backing Up Statistics.....	118
Chapter 5 Post-Deployment Operations.....	120
5.1 How to Connect to a FEP Cluster.....	120
5.2 Configuration Change.....	121
5.3 FEPCluster Resource Change.....	122
5.3.1 Changing CPU and Memory Allocation Resources.....	122
5.3.2 Resizing PVCs.....	122
5.4 FEPPGPool2 Configuration Change.....	122
5.5 Scheduling Backup from Operator.....	125
5.6 Configure MTLS Setting.....	126
5.6.1 Certification Rotation.....	126
5.7 Monitoring.....	126
5.7.1 Monitoring FEP Operator and Operands.....	127
5.7.2 Monitoring FEP Server.....	127
5.7.2.1 Architecture.....	127
5.7.2.2 Default Server Metrics Monitoring	128
5.7.2.3 Default Alerts.....	130
5.7.2.4 Graphical user interface.....	131
5.7.3 Monitoring FEP Backup.....	132
5.7.3.1 pgbackrest_info_backup view.....	132
5.7.4 Monitoring FEP PGPool2.....	133
5.7.4.1 pgpool2_stat_load_balance view.....	133
5.7.4.2 pgpool2_stat_conn_pool view.....	133
5.7.4.3 pgpool2_stat_sql_command view.....	134
5.8 Event Notification.....	134
5.8.1 Events raised	134
5.8.2 Events that Occur when Custom Resources are Updated.....	135
5.8.3 Viewing the Custom Events.....	135
5.9 Scaling Replicas.....	136
5.9.1 Automatic Scale Out.....	136
5.9.2 Manual Scale In/Out.....	136
5.10 Backing Up to Object Storage.....	136
5.10.1 Pre-creation of Resources.....	136
5.10.1.1 Storing CA Files (Root Certificates).....	136
5.10.1.2 Storing Repository Key.....	137
5.10.2 Defining a FEPCluster Custom Resource.....	137
5.11 Disaster Recovery.....	138
5.11.1 Disaster Recovery by Backup/Restore Method.....	138
5.11.1.1 Disaster Recovery Prerequisites.....	138
5.11.1.2 Performing Disaster Recovery.....	139
5.11.1.2.1 Pre-creation of Resources.....	139
5.11.1.2.2 Defining a FEPCluster Custom Resource.....	139
5.11.2 Disaster Recovery with Continuous Recovery Method.....	141
5.11.2.1 Disaster Recovery Prerequisites.....	141
5.11.2.2 Performing Disaster Recovery.....	141
5.11.3 Disaster Recovery Using Velero.....	141
5.11.3.1 Disaster Recovery Prerequisites.....	141
5.11.3.2 Performing Disaster Recovery.....	142
5.11.3.2.1 Configuring FEPCluster Custom Resources.....	142
5.11.3.2.2 Velero Backup.....	143

5.11.3.2.3 Database Backup.....	144
5.11.3.2.4 Velero Restore.....	144
5.11.3.2.5 Return from Disaster Recovery Environment to Production Environment.....	145
5.11.4 Disaster Recovery with Streaming Replication Method.....	145
5.11.4.1 Disaster Recovery Prerequisites.....	145
5.11.4.2 Performing Disaster Recovery.....	145
5.11.5 Parameter Change in Disaster Recovery Environment.....	145
5.12 Operation of Transparent Data Encryption Using Key Management System.....	145
5.12.1 Updating Custom Resource Parameters.....	145
5.12.2 Update Credentials.....	146
5.12.3 Encrypting a Tablespace.....	146
5.12.4 Backup/Restore.....	146
5.12.5 Changing Key Management System Definitions.....	147
5.13 Confidentiality Management Feature.....	147
5.13.1 Enabling Confidentiality Management Feature.....	147
5.13.2 Monitoring Confidentiality Management Feature.....	148
5.14 Operation of Fixed Statistics.....	148
5.14.1 Preparing for Object Storage Connections.....	148
5.14.1.1 Using S3.....	148
5.14.1.2 Using AzureBlob.....	148
5.14.1.3 Using Google Cloud Storage.....	148
5.14.2 Storing validation environment statistics in object storage.....	148
5.14.3 Schedule for Fixed Statistics.....	149
5.14.4 Using Local Storage.....	149
5.14.4.1 Exporting Statistics Files to FEPCluster in a Validation Environment.....	149
5.14.4.2 Deploying Statistics Files to FEPCluster in a Production Environment.....	150
5.14.4.3 Specify File Names for FEPCluster Custom Resources.....	150
Chapter 6 Maintenance Operations.....	151
6.1 Minor Version Upgrade.....	151
6.2 Cluster Master Switchover.....	151
6.3 Perform PITR and the Latest Backup Restore from Operator.....	151
6.3.1 Setting Item.....	152
6.3.2 After Restore.....	152
6.4 Major Version Upgrade.....	152
6.4.1 Pre-work on the Data Source FEP Cluster.....	152
6.4.2 Operator Upgrade.....	153
6.4.2.1 Uninstalling the Old Operator.....	153
6.4.2.2 Installing a New Version of the Operator.....	153
6.4.3 Major Version Upgrade of FEP.....	153
6.4.3.1 Creating a New FEPCluster CR.....	153
6.4.3.2 Verifying FEP Major Upgrade Complete.....	156
6.4.4 Updating Each Custom Resource.....	156
6.4.4.1 Removing a FEPClusterCR for a Data Source.....	156
6.4.4.2 FEPPgpool2.....	156
6.4.4.3 FEPExporter Built in Standalone Mode.....	157
6.5 Assigned Resources for Operator Containers.....	157
6.5.1 How to Change Assigned Resources.....	157
6.5.1.1 When installing using OperatorHub.....	157
6.5.1.2 When installing using Helm Chart or RancherUI.....	158
6.6 Using SUPERUSER Privilege.....	158
6.6.1 CREATE EXTENSION.....	158
6.6.2 Change Password of SUPERUSER.....	159
6.6.3 Using SUPERUSER.....	159
Chapter 7 Abnormality.....	160
7.1 Handling of Data Abnormalities.....	160
7.2 Handling when the Capacity of the Data Storage Destination or Transaction Log Storage Destination is Insufficient.....	160

7.3 What to do when the Capacity of the Backup Data Storage Area is Insufficient.....	160
7.4 Handling Access Abnormalities When Instance Shutdown Fails.....	160
7.5 Collection of Failure Investigation Information.....	160
7.6 Log Collection Tool.....	161
7.6.1 Logs Collected.....	162
7.6.2 Location and Dependencies.....	162
7.6.3 User Interface.....	163
7.6.3.1 Command-Line Syntax.....	163
7.6.4 Log Collection Directory.....	164
7.6.4.1 Command-Line Specification.....	164
7.6.4.2 Directory Structure.....	164
Appendix A Quantitative Values and Limitations.....	165
A.1 Quantitative Values.....	165
A.2 Limitations.....	165
Appendix B Adding Custom Annotations to FEPCluster Pods using Operator.....	166
Appendix C Utilize Shared Storage.....	168
C.1 Creating a StorageClass.....	168
C.2 Creating a PersistentVolume.....	168
C.3 Creating FEPCluster.....	169
Appendix D Key Management System Available for Transparent Data Encryption.....	170
D.1 KMIP Server.....	170
D.2 AWS Key Management Service.....	170
D.2.1 Available Services.....	170
D.2.2 Available AWS KMS Keys.....	170
D.2.3 Required Privileges.....	170
D.2.4 Key ID.....	170
D.3 Azure Key Management Service.....	170
D.3.1 Available Services.....	170
D.3.2 Available Keys.....	171
D.3.3 Available Algorithms.....	171
D.3.4 Key Operation.....	171
D.3.5 Key ID.....	171
D.3.6 Sign In.....	171

Chapter 1 System Requirements

This chapter describes the system requirements.

1.1 Components Embedded

The FEP Server container embeds following components. However it is understood that these components are bound to be upgraded in the maintenance phase.

No	Component	Version	Description
1	Red Hat UBI minimal	9	Meant to provide base OS image for the container
2	Fujitsu Enterprise Postgres Server	16.4	To provide server capabilities
3	Patroni	3.3.1	To provide HA capabilities and other management to the Cluster

1.2 CPU

It should be noted that it provides supports to both the following CPU Architectures to meet the scope of work.

No	CPU architecture
1	x86
2	s390x
3	ppc64le

1.3 Supported Platform

It supports running on the following platforms.

No	Platform	Version
1	OpenShift Container Platform	4.13, 4.14, 4.15
2	Rancher Kubernetes Engine (on Linux hosts) (*1)	1.4.0+
3	Vmware Tanzu Kubernetes Grid (*1)	1.6+
4	Full Managed Kubernetes Service - Azure Kubernetes Service - Amazon Elastic Kubernetes Service - Alibaba Cloud Container Service for Kubernetes - Google Kubernetes Engine - IBM Cloud Kubernetes Service	1.26, 1.27, 1.28

*1: Kubernetes 1.26 - 1.28

Supports storage supported by OpenShift or Kubernetes (AKS, EKS, RKE, ACK, GKE, IKS and TKG).

However, you need shared storage, like NFS, or object storage for backup and archive WAL volumes. Object storage supports Amazon Simple Storage Service, Azure Blob Storage, and Google Cloud Storage.

1.4 Collaboration Tool

Supports integration with the following tools.

No	Tool	Version	How to obtain
1	Prometheus	- OpenShift The version installed OpenShift - Kubernetes - Prometheus v2.36.2 and later - AlertManager v0.24.0 and later - Rancher The version provided by Rancher Monitoring Chart	- OpenShift Preinstalled with OpenShift - Kubernetes prometheus-operator (v0.61.1 and later) https://github.com/prometheus-operator/prometheus-operator/prometheus-operator - Rancher Using the Rancher Monitoring Chart
2	AlertManager		
3	Grafana	- OpenShift and Kubernetes Grafana v7.5.17 and later - Rancher The version provided by Rancher Monitoring Chart	- OpenShift Provided by OperatorHub - Kubernetes grafana-operator (v4.7.1 and later) https://github.com/grafana-operator/grafana-operator - Rancher Using the Rancher Monitoring Chart
4	Helm	3.10.0 and later	- Kubernetes only Helm Web Site https://helm.sh/docs/intro/install/
5	Rancher	v2.7 and later	Rancher Web Site https://rancher.com/
6	Prometheus Adapter	- OpenShift and Kubernetes Confirmed the operation with v0.9.1 and later - Rancher The version provided by Rancher Monitoring Chart	- OpenShift and Kubernetes Prometheus Adapter https://github.com/kubernetes-sigs/prometheus-adapter - Rancher Using the Rancher Monitoring Chart
7	Elastic Search	8.5.2 and later	- OpenShift Provided by OperatorHub - Kubernetes https://github.com/elastic/helm-charts/tree/main/elasticsearch
8	Velero	v1.11 and later	Refer to the documentation published below. https://velero.io/docs/main/basic-install/

Chapter 2 Overview of Operator Design

This chapter describes an overview of the operator design.

2.1 Design Task

Installation/operation using an operator and necessity of design are shown below.

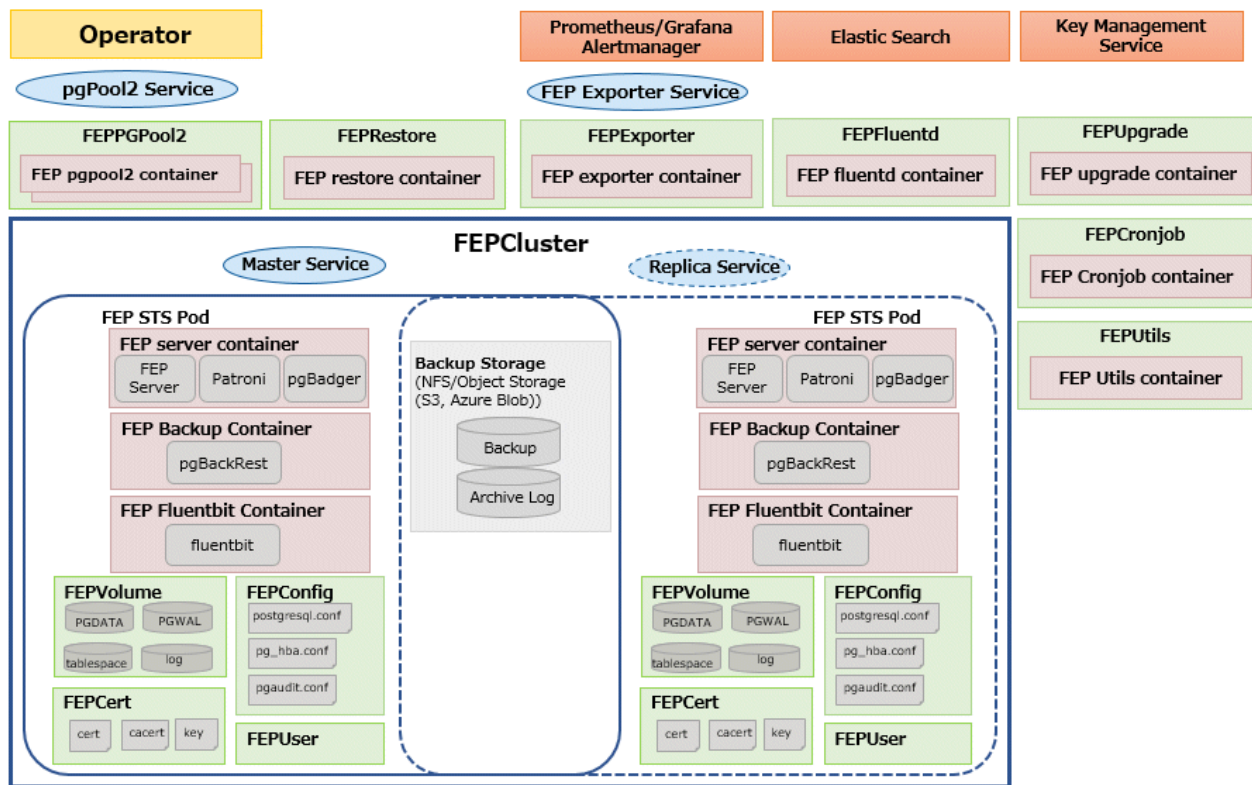
Task	Design required to operate FEP	Where to find
FEP setup	Required.	2.3.1 Deployment
High availability configuration	Recommended. (When checking or changing the behavior of high availability. However, even by default, constant high availability operation is possible.)	2.3.2 High Availability
Volume settings	Recommended. (When setting the volume. However, even by default, allocate a fixed volume.)	2.3.3 Configurable Volume per Cluster
Pgpool-II setup	Recommended. (When using Pgpool-II.)	2.3.4 Deploying Pgpool-II and Connect to FEPCluster from Operator
Backup/restore settings	Recommended. (When using a backup and restore.)	2.3.5 Scheduling Backup from Operator 2.3.6 Perform PITR and Latest Backup Restore from Operator
Monitoring & Alert(FEPEXporter)	Recommended. (When using Monitoring and Alert)	2.3.8 Monitoring & Alert (FEPEXporter)
Scaling Replicas	Recommended (When using scaling feature)	2.3.9 Scaling Replicas
Key management system	Recommended. (When the key management system manages the master encryption key for transparent data encryption)	2.3.11 Transparent Data Encryption Using a Key Management System

2.2 System Configuration Design

This section describes the system configuration.

2.2.1 Server Configuration

The following is an overview diagram of the server configuration:



System component

Describes various system resources.

Configuration server type	Description
FEP operator	A container that accepts user requests and is responsible for automating database construction and operational operations.
FEP server container	A container for the FEP server.
FEP backup container	A container that performs scheduled backup operations. Created on the same Pod as the FEP server container.
FEP Fluentbit container	A container that collect FEP database CSV log and forward to fluentd container for processing.
FEP pgpool2 container	A container that uses Pgpool-II to provide load balancing and connection pooling. If you do not use it, you do not need to create it.
FEP restore container	A container that performs the restore operation. Temporarily created during a restore operation.
FEP Exporter container	A container that exposes http/https endpoint for monitoring stats scraping.
FEP Fluentd container	A container that summarise FEP log severity as metrics for Prometheus to consume. Optionally, forward log entries to Elasticsearch for detailed log analysis.
FEP upgrade container	A container that executes the major version upgrade process of the server container. A container created temporarily during the upgrade process.
FEP Cronjob container	A container that is started when the regular processing of each feature of the operator is executed.
FEP Utils container	A container for cloud-based management.
Backup storage	Storage where backup data is stored. If you do not need to obtain a backup, you do not need to create one.

Configuration server type	Description
FEPCluster	Parent CR for FEP Cluster definition and configuration.
FEPBackup	Child CR for backup configuration.
FEPVolume	Child CR for volumes.
FEPConfig	Child CR for FEP configurations.
FEPCert	Child CR for system certificates.
FEPUser	Child CR for database users.
FEPAction	CR for performing actions.
FEPExporter	CR for monitoring configuration.
FEPUpgrade	CR for major upgrade.
Master service	A service to connect to the master FEP server.
Replica service	A service to connect to the replica FEP server.
Pgpool2 service	A service for connecting to Pgpool-II.
Fepexporter service	A service to scrape metrics from all FEPCluster nodes.

2.2.2 User Account

The user accounts used by this product are as follows.

It is possible to improve security by clearly separating and managing the accounts that operate the operators for each role by the infrastructure administrator. It is also possible to manage multiple tenants on a single container management platform.

User type	User name	Description
Infrastructure administrator	Mandatory	A system administrator (superuser) who manages the container management platform that installs operators.
Database administrator	Mandatory	An administrator who performs setup, regular operation, and maintenance operations.
Confidential administrator	Mandatory	An administrator who sets appropriate privileges for each database resource for database users.
Application developer	Mandatory	Develops and executes database applications.

2.2.3 Basic Information of the Container

This section describes the basic information of the container.

FEP server container

The naming convention for the FEP server container is as below.

fujitsu-enterprise-postgres-16-server: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images).

- fujitsu-enterprise-postgres-16-server:ubi9-16-1.0
 - fujitsu-enterprise-postgres-16-server:ubi9-16-1.0-amd64
 - fujitsu-enterprise-postgres-16-server:ubi9-16-1.0-s390x
 - fujitsu-enterprise-postgres-16-server:ubi9-16-1.0-ppc64le

FEP backup container

Use the same naming convention for FEP backup containers as for FEP server containers.

fujitsu-enterprise-postgres-16-backup: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images)

- fujitsu-enterprise-postgres-16-backup:ubi9-16-1.0
 - fujitsu-enterprise-postgres-16-backup:ubi9-16-1.0-amd64
 - fujitsu-enterprise-postgres-16-backup:ubi9-16-1.0-s390x
 - fujitsu-enterprise-postgres-16-backup:ubi9-16-1.0-ppc64le

FEP restore container

Use the same naming convention for FEP restore containers as for FEP server containers.

fujitsu-enterprise-postgres-16-restore: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images)

- fujitsu-enterprise-postgres-16-restore:ubi9-16-1.0
 - fujitsu-enterprise-postgres-16-restore:ubi9-16-1.0-amd64
 - fujitsu-enterprise-postgres-16-restore:ubi9-16-1.0-s390x
 - fujitsu-enterprise-postgres-16-restore:ubi9-16-1.0-ppc64le

FEP pgpool2 container

Use the same naming convention for FEP pgpool2 containers as for FEP server containers.

fujitsu-enterprise-postgres-16-pgpool2: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images)

- fujitsu-enterprise-postgres-16-pgpool2:ubi9-16-1.0
 - fujitsu-enterprise-postgres-16-pgpool2:ubi9-16-1.0-amd64
 - fujitsu-enterprise-postgres-16-pgpool2:ubi9-16-1.0-s390x
 - fujitsu-enterprise-postgres-16-pgpool2:ubi9-16-1.0-ppc64le

FEP Exporter container

FEP Exporter container as for FEP server containers.

fujitsu-enterprise-postgres-exporter: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images)

- fujitsu-enterprise-postgres-exporter:ubi9-16-1.0
 - fujitsu-enterprise-postgres-exporter:ubi9-16-1.0-amd64
 - fujitsu-enterprise-postgres-exporter:ubi9-16-1.0-s390x

- fujitsu-enterprise-postgres-exporter:ubi9-16-1.0-ppc64le

FEP Fluentd container

FEP Fluentd container as for FEP server containers.

fujitsu-enterprise-postgres-fluentd: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images)

- fujitsu-enterprise-postgres-fluentd:ubi9-16-1.0
 - fujitsu-enterprise-postgres-fluentd:ubi9-16-1.0-amd64
 - fujitsu-enterprise-postgres-fluentd:ubi9-16-1.0-s390x
 - fujitsu-enterprise-postgres-fluentd:ubi9-16-1.0-ppc64le

FEP Fluentbit container

FEP Fluentbit container as for FEP server containers.

fujitsu-enterprise-postgres-fluentbit: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images)

- fujitsu-enterprise-postgres-fluentbit:ubi9-16-1.0
 - fujitsu-enterprise-postgres-fluentbit:ubi9-16-1.0-amd64
 - fujitsu-enterprise-postgres-fluentbit:ubi9-16-1.0-s390x
 - fujitsu-enterprise-postgres-fluentbit:ubi9-16-1.0-ppc64le

FEP Cronjob container

FEP Cronjob container as for FEP server containers.

fujitsu-enterprise-postgres-cronjob: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images)

- fujitsu-enterprise-postgres-cronjob:ubi9-16-1.0
 - fujitsu-enterprise-postgres-cronjob:ubi9-16-1.0-amd64
 - fujitsu-enterprise-postgres-cronjob:ubi9-16-1.0-s390x
 - fujitsu-enterprise-postgres-cronjob:ubi9-16-1.0-ppc64le

FEP upgrade container

FEP upgrade container as for FEP server containers.

fujitsu-enterprise-postgres-16-upgrade: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images)

- fujitsu-enterprise-postgres-16-upgrade:ubi9-16-1.0
 - fujitsu-enterprise-postgres-16-upgrade:ubi9-16-1.0-amd64
 - fujitsu-enterprise-postgres-16-upgrade:ubi9-16-1.0-s390x
 - fujitsu-enterprise-postgres-16-upgrade:ubi9-16-1.0-ppc64le

FEP Utils container

FEP Utils container as for FEP server containers.

fujitsu-enterprise-postgres-16-utils: *OS-FEPBaseVersion-MajorVersion.MinorVersion-ARCH*

For each *Version*, specify the following:

Field	Values	Description
<i>OS</i>	ubi9	
<i>FEPBaseVersion</i>	16	
<i>MajorVersion</i>	1,2, ...	To be used when major change in image, including server patch application
<i>MinorVersion</i>	0,1,2 ...	To be used when minor changes in image, e.g bug fix in container script

The first publishing will expect following names / tagging (Manifest and Child images)

- fujitsu-enterprise-postgres-16-utils:ubi9-16-1.0
- fujitsu-enterprise-postgres-16-utils:ubi9-16-1.0-amd64
- fujitsu-enterprise-postgres-16-utils:ubi9-16-1.0-s390x
- fujitsu-enterprise-postgres-16-utils:ubi9-16-1.0-ppc64le

2.3 Design Perspective for Each Feature

This section describes the design of each feature.

postgresql-cfg format

A postgresql-cfg represent ConfigMap for containing postgresql parameters. The file is used to contain the parameters which need to be reflected in postgresql.conf of the instance. Since patroni ignores all parameters which are not known by OSS postgresql.conf, an approach is defined to treat FEP Parameters in a special way.

The content of the ConfigMap is defined by key=value format. The following table shows the detail:

Spec	Example	Comment
The content may have multiple key/ value pairs	foo=bar foo1=bar1	-
The value cannot have space unless quoted.	foo=bar bar2	Invalid
The quoted value cannot have another value after	foo='bar bar2' something	Invalid
The key value pair must have a '=' sign	-	-
White spaces are allowed before/ after/between the key value pair	foo = bar	-
Any content after '#' will be ignored	# this is a comment foo=bar #this is a comment	-
The value may be quoted by single quotes	foo='bar bar2'	-
Single quote can be escaped by two single quotes	foo='It's ok'	Note: single quotes are not supported by Patroni edit-config command

Spec	Example	Comment
Backslash '\' will be replaced by '\\' when invoking patronictl edit-config command	-	To avoid command line escape
When a key value pair is invalid, it will be ignored. the update continue to process next pair	foobar foo2=bar2	The 'foobar' will be ignored
The container script does not validate the key and value as long as they are in correct format.	-	-

It is recommended to use the psql's show command to verify parameter is setting correctly.

2.3.1 Deployment

Information for the FEPCluster

Equivalent Kubernetes command: `kubectl apply -f FEPClusterCR.yaml`

This operation will create a FEPCluster with supplied information in FEPClusterCR.yaml.

Refer to "FEPCluster parameter" in the Reference for details.

2.3.2 High Availability

Describes the settings for using the highly available features.

Arbitration

Patroni is used to control and monitor FEP instance startup, shutdown, status and trigger failover should the master instance fails. It plays a significant role in the solution. If the Patroni process dies, especially on master POD, without notice, the Pod will not update the Patroni cluster lock. This may trigger an unwanted failover to one of the Replica, without corresponding corrective action on the running master. This can create a split brain issue. It is important to monitor Patroni's status to make sure it is running. This is done using liveness probe. Important to note that this is not expected to be configured by end user.

```
livenessProbe:
  httpGet:
    scheme: HTTP
    path: /liveness
    port: 25001
  initialDelaySeconds: 30
  periodSeconds: 6
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
```

2.3.3 Configurable Volume per Cluster

Cluster node (Pod) volumes are created according to the values set in the storage section of `fehChildCrVal` in the FEPCluster custom resource.



Note

- After you create the FEPCluster for the first time, you cannot add new volumes later or modify the storageClass or accessModes.
- You can resize the initially created volume only if the underlying storageClass supports dynamic resizing.

The following is the schema for the storage section of the FEPCluster customer resource:

Field	Mandatory	Sub-Field	Default	Description
archivewalVol	No	size	1Gi	Volume size of the archive log. Refer to "Estimating Database Disk Space Requirements" in the Fujitsu Enterprise Postgres Installation and Setup Guide for Server to help you design the size.
		storageClass	Defaults to platform default if omitted	SC is only set at start
		accessModes	Defaults to ReadWriteOnce if omitted	Access mode is only set at start
backupVol	No	size	2Gi	Volume size of the backup. Estimate based on the following formula: (full backup generations + incr backup generations + 1) * dataVol size
		storageClass	Defaults to platform default if omitted	SC is only set at start
		accessModes	Defaults to ReadWriteOnce if omitted	Access mode is only set at start
dataVol	Yes	size	2Gi	Volume size of the data. Refer to "Estimating Database Disk Space Requirements" in the Fujitsu Enterprise Postgres Installation and Setup Guide for Server and base the design on table/index size.
		storageClass	Defaults to platform default if omitted	SC is only set at start
		accessModes	Defaults to ReadWriteOnce if omitted	Access mode is only set at start
logVol	No	size	1Gi	Volume size of the log. If you change the log output level (default: WARNING) or enable the audit log feature, measure the actual amount of log output in a test environment.
		storageClass	Defaults to platform default if omitted	SC is only set at start
		accessModes	Defaults to ReadWriteOnce if omitted	Access mode is only set at start
tablespaceVol	No	size	512Mi	Volume size of the tablespace. When using tablespaces, as with dataVol, you should refer to "Estimating Database Disk Space Requirements" in the Fujitsu Enterprise Postgres Installation and Setup Guide for Server for information on sizing.

Field	Mandatory	Sub-Field	Default	Description
		storageClass	Defaults to platform default if omitted	SC is only set at start
		accessModes	Defaults to ReadWriteOnce if omitted	Access mode is only set at start
walVol	Yes	size	1200Mi	Volume size of the transaction log. Refer to "Estimating Database Disk Space Requirements" in the Fujitsu Enterprise Postgres Installation and Setup Guide for Server to help you design the size. Note that the default value for max_wal_size is 1 GB.
		storageClass	Defaults to platform default if omitted	SC is only set at start
		accessModes	Defaults to ReadWriteOnce if omitted	Access mode is only set at start

The 'accessMode' is been incorporated for the inclusion of pgBadger layer later. Giving it a shared volume capability will allow pgBadger Container to read logs from multiple server instance (master / replica) and expose it via a WebServer.

2.3.3.1 Disk Space Management

Due to a sudden increase in queries, etc., the amount of data and WAL will increase, and the disk capacity will be compressed, which may cause the database operation to stop. If the disk usage exceeds the threshold, or if database operation has stopped due to insufficient disk space, use the following methods to resolve the insufficient disk space.

- Increasing disk space
- Reducing disk usage

2.3.3.1.1 Increasing Disk Space

There are two ways to increase disk space:

- Expanding disk capacity

If you are using a volume that can use the PVC extension function of Kubernetes, expand the disk capacity and solve the lack of space.

- Migrating to a database cluster with a large disk capacity

If you are using a volume that cannot be used with the PVC extension function of Kubernetes, or if the volume cannot be expanded further due to the upper limit, etc., consider migrating the database cluster. Migrating data to a cluster that uses large-capacity disks solves the lack of capacity.

Expanding disk capacity

Expand your disk capacity with the Kubernetes PVC extension. Only disks that support the PVC expansion function can be expanded. Check the specifications of the CSI driver you are using to see if the disk supports PVC extensions.

Disk capacity expansion can be performed manually by the user at any time, or automatically by the operator when the usage exceeds the threshold in cooperation with the monitoring function.

Manual expansion can expand a PVC by changing the storage definition of the FEPCluster custom resource. Rewrite the custom resource and expand the PVC when AlertManager gives you a notification that the disk usage exceeds the threshold or the database stops due to lack of disk space. Refer to "[5.3.2 Resizing PVCs](#)" for more information on manual expansion. Also, refer to "Default Alert Rule" in the Reference for an example definition of AlertManager's alert rule.

Automatic expansion does not require database administrator monitoring or manual maintenance work (volume capacity expansion) until the expansion limit is reached.

For more information on auto expansion, refer to ["2.3.3.2 Configuring PVC Auto Expansion"](#).

If you are using a disk that does not support the PVC expansion function or if the disk capacity cannot be expanded, Refer to ["Migrating to a database cluster with a large disk capacity"](#) and ["2.3.3.1.2 Reducing Disk Usage"](#).

Migrating to a database cluster with a large disk capacity

Use the backup/restore function to construct a new database cluster on another disk and migrate the data. Use this method when:

- When the alert manager issues a notification that the disk usage exceeds the threshold
- When the database stops due to lack of disk space

Insufficient disk space can be resolved by changing to a disk with a larger capacity.

When migrating to a new database cluster, you can build a new FEPCluster and restore data by setting spec.changeParams of the FEPRestore custom resource and changing the definition from the restore source. For details, refer to "FEPRestore Custom Resource Parameters" in the Reference.

2.3.3.1.2 Reducing Disk Usage

Execute the REINDEX statement on the data storage destinations (dataVol, walVol, tablespaceVol) as preventive maintenance for insufficient disk space. For details, refer to "Reorganizing Indexes" in the Fujitsu Enterprise Postgres Operation Guide.

Consider reducing the amount of disk usage as preventive maintenance for insufficient capacity of the transaction log storage destination and backup data storage destination. If the capacity of the transaction log storage destination is insufficient, review the log file rotation settings and output level, and consider changing them. If the backup data or transaction log archive storage space is insufficient, consider reducing the number of backup generations saved.

You can reduce the number of backup generations by specifying "backup_expire" for spec.fepAction.type of the FEPAAction custom resource. For details, refer to "FEPAAction Custom Resource Parameters" in the Reference.

2.3.3.2 Configuring PVC Auto Expansion

By setting the FEPCluster custom resource spec.fepChildCrVal.storage.autoresize.enable to true, you can enable the PVC auto-grow feature that automatically expands disk space when disk usage exceeds a threshold.

The following two conditions must be met to enable the PVC auto-expansion function.

- Specify a volume that supports the PVC expansion function in StorageClass
- Specify true in the allowVolumeExpansion field in StorageClass

Check the specifications of the CSI driver you are using to see if the specified volume supports the PVC expansion function.

We also need Prometheus to monitor storage usage. Scrape the metrics captured by the kubelet below with Prometheus.

- kubelet_volume_stats_used_bytes (Volume used capacity (bytes))
- kubelet_volume_stats_capacity_bytes (Volume capacity (bytes))

Make sure that the scrape_config section of your Prometheus config file references /metrics, which the kubelet serves over https, for each node. Check the Prometheus documentation for more details.

The PVC auto-expansion feature can be enabled/disabled even after building the FEPCluster.

Enabling PVC auto-expansion builds a fep-tuning pod containing a pvc-auto-resize container. The pvc-auto-resize container periodically retrieves metrics from Prometheus for each defined PVC. If the PVC volume usage rate exceeds the defined threshold, the definition of the FEPCluster custom resource is automatically rewritten. The target PVC is automatically extended by rewriting the custom resource.

If the FEPCluster is configured with multiple units, the PVC will be expanded if the volume usage rate of even one unit exceeds the threshold.

The following parameters are used in the PVC auto expansion feature. For details of each parameter, refer to the Reference.

- spec.fep.autoTuning section: Prometheus connection information for retrieving metrics

- spec.fepChildCrVal.storage.autoresize section: Storage common extended settings
- spec.fepChildCrVal.storage.xxxVol section: definition and individual extension settings for each storage

In the expansion settings, it is possible to define the volume utilization threshold, amount of size expansion, upper limit of size that can be expanded, and so on.

Below is an example of defining a FEPCluster custom resource when enabling the PVC auto expansion feature.

```
spec:
  fep:
    autoTuning:
      prometheus:
        prometheusUrl: http://prometheus-prometheus-oper-prometheus.prometheus.svc:9090
  fepChildCrVal:
    storage:
      autoresize:
        enable: True
        threshold: 20
        increase: 20
      dataVol:      # Use the data volume as is defined under autoresize
        size: 10Gi
        storageClass: resizable-storage
      walVol:      # wal volume changes threshold and expansion limit
        size: 2Gi
        storageClass: resizable-storage
        threshold: 50
        storageLimit: 10
      backupVol:   # backup volume does not expand
        size: 20Gi
        storageClass: share-storage
        accessModes: ReadWriteMany
        storageLimit: 0
```

Concept of combination with monitoring feature

The PVC auto-expansion feature allows you to limit the amount of storage that can be expanded. However, there is a possibility that more data than expected may occur and the amount of data may exceed the set upper limit. To avoid this, we recommend using the monitoring function in conjunction with the PVC auto-expansion function.

In the alert rule created by default when using the FEPExporter function, an alert will be sent when the volume usage rate exceeds 90%. The default threshold for PVC autogrowth is 80%. As a result, even if the volume usage increases when the disk expansion limit is reached and automatic expansion is not performed, an alert will be sent from AlertManager, so you can be aware of the lack of disk space.

By setting the autogrowth threshold to a value lower than the alert rule threshold, you can keep more safe disk space.

2.3.4 Deploying Pgpool-II and Connect to FEPCluster from Operator

Equivalent Kubernetes command: `kubectl create FEPpgpool2`

This operation will create a FEP pgpool2 container with supplied information.

Refer to “FEPPgpool2 Custom Resource Parameters” in the “Reference” for more information.

2.3.5 Scheduling Backup from Operator

When creating a FEPCluster, users can obtain scheduled backups by setting up backup definitions. Users can also modify the backup schedule by modifying the Backup custom resource that was created.

A backup definition includes the following:

- Acquisition time (Specify in crontab format)
- Backup type (Full or incremental backups)

Backup is taken on master Pod only.

Backup processing is performed by pgBackRest.

Parameter can be set to pgbackrestParams in CR definition.

The maximum number of backup schedules is 5.

See the pgBackRest User's Guide for details on the parameters.

However, some parameters are limited. Details are given below.

- [2.3.5.1 Important Setting Items](#)
- [2.3.5.2 Parameters that cannot be Set](#)
- [2.3.5.3 Restricted Parameters](#)
- [2.3.5.4 About Sections in the Config File](#)

2.3.5.1 Important Setting Items

Here are the important parameters for setting pgBackRest. This parameter sets the retention period of backup information. If automatic backup is set and this parameter is not set, the risk of overflowing the backup area increases.

Parameter	Overview of parameters	Setting value
Full Retention Option (repo retention -full)	Specify number of full backups to keep No default (should be set according to user backup policy)	natural number
Full Retention Type Option (repo retention-full-type)	spec.retention -full Specifies whether the setting is a number of retention days (time) or a number of retention generations (count) No default (should be set according to user backup policy)	time/count

The following is a sample CR example of changing the backup retention period (How long the PITR is valid) to 30 days after a FEPCluster deployment by setting the above parameters.

```
apiVersion: fep.fujitsu.io/v1
kind: FEPBackup
metadata:
  name: fepcluster-backup
spec:
  pgBackrestParams: |
    # define custom pgbackrest.conf parameters below to override defaults.
    [global]
    repo-retention-full = 30
    repo-retention-full-type = time
  ...
```

2.3.5.2 Parameters that cannot be Set

The following parameters in the pgBackRest Configuration Reference are not configurable.

Parameter	Overview of parameters	Reason
Copy Archive Option (--archive -copy)	Copy the WAL segments needed for consistency to the backup	To use internal fixed values
Check Archive Mode Option (--archive-mode-check)	Check the PostgreSQL archive_mode setting.	Limited to backup from master

Parameter	Overview of parameters	Reason
Backup from Standby Option (--backup-standby)	Back up from the standby cluster	Limited to backup from master
Stop Auto Option (--stop-auto)	Stops a previously failed backup on a new backup.	Because they are 9.6 not supported in
pgBackRest Command Option (--cmd)	pgBackRest command	To use internal fixed values
SSH client command Option (--cmd-ssh)	Path to ssh client executable	Not using ssh
Compress Option (--compress)	Use File Compression	For obsolete options (Use compress-type option instead)
Config Option (--config)	pgBackRest configuration file.	To use internal fixed values
Config Include Path Option (--config-include-path)	Path to additional pgBackRest configuration files.	To use internal fixed values
Config Path Option (--config-path)	Base path of pgBackRest configuration files.	To use internal fixed values
Delta Option (--delta)	Restore or Backup with Checksum	For new restores only
Dry Run Option (--dry-run)	Execute a dry-run for the command.	command-line only option
Lock Path Option (--lock-path)	Path where the lock file is stored	To use internal fixed values
Keep Alive Option (--sck -keep-alive)	Enable keep-alive messages on socket connections	To use internal fixed values
Spool Path Option (--spool-path)	Path to store temporary data for asynchronous archive-push and archive-get commands	For automatic determination from FEPCluster CR values
Stanza Option (--stanza)	Defines the stanza.	To use internal fixed values
Console Log Level Option (--log-level-console)	Console Log Level	It is not expected to operate on Pod.
Std Error Log Level Option (--log-level-stderr)	Stderr log level	It is not expected to operate on Pod.
Log Path Option (--log-path)	Log File Destination	For automatic determination from FEPCluster CR values
Repository Host Option (--repo-host)	Repository host for remote operations via SSH	Repository Host is not used
Repository Host Command Option (--repo-host-cmd)	Path of pgBackRest on Repository Host	
Repository Host Configuration Option (--repo-host-config)	Repository Host Configuration File Path	
Repository Host Configuration Include Path Option (--repo-host-config-include-path)	Repository hosts configuring include path	
Repository Host Configuration Path Option (--repo-host-config-path)	Repository Host Configuration Path	
Repository Host Port Option (--repo-host-port)	Repository host port when "repo-host" is configured	

Parameter	Overview of parameters	Reason
Repository Host User Option (--repo-host-user)	Repository host user when "repo-host" is configured	
Repository Path Option (--repo-path)	Path where backups and archives are stored	For automatic determination from FEPCluster CR values
Archive Retention Option (--repo-retention-archive)	The number of consecutive WAL backups to keep.	This option is not recommended, and WAL retention is controlled by the Full Retention Option and Full Retention Type Option.
Archive Retention Type Option (--repo-retention-archive-type)	Backup Type for WAL Retention	It is recommended not to change from the default.
Differential Retention Option (--repo-retention-diff)	Number of incremental backups to keep	No incremental backups
Archive Mode Option (--archive-mode)	Retains or disables the archive for the restored cluster.	To use internal fixed values
Exclude Database Option (--db-exclude)	Restore excluding the specified databases.	To restore the entire FEP cluster, including all databases
Include Database Option (--db-include)	Restore only the specified database	To restore the entire FEP cluster, including all databases
Link All Option (--link-all)	Restore all symbolic links.	To use internal fixed values
Link Map Option (--link-map)	Changes the destination of a symbolic link.	To use internal fixed values
Recovery Option Option (--recovery-option)	Setting options in postgresSQL recovery.conf	To use internal fixed values
Tablespace Map Option (--tablespace-map)	Restoring tablespace to a specified directory	For automatic determination from FEPCluster CR values
Map All Tablespaces Option (--tablespace-map-all)	Restores all tablespaces to the specified directory	No tablespace required because there is only one tablespace per FEPCluster
TLS Server Address Option (--tls-server-address)	TLS server address.	TLS server not used
TLS Server Authorized Clients Option (--tls-server-auth)	TLS server authorized clients.	
TLS Server Certificate Authorities Option (--tls-server-ca-file)	TLS server certificate authorities.	
TLS Server Certificate Option (--tls-server-cert-file)	TLS server certificate file.	
TLS Server Key Option (--tls-server-key-file)	TLS server key file.	
TLS Server Port Option (--tls-server-port)	TLS server port.	
PostgreSQL Database Option (--pg-database)	PostgreSQL database.	To use internal fixed values
PostgreSQL Host Option (--pg-host)	PostgreSQL host for remote operations via SSH	No SSH connection required

Parameter	Overview of parameters	Reason
PostgreSQL Host Command Option (--pg-host-cmd)	Path of pgBackRest exe on the PostgreSQL host	To use internal fixed values
PostgreSQL Host Configuration Option (--pg-host-config)	Path of the pgBackRest configuration file	To use internal fixed values
PostgreSQL Host Configuration Include Path Option (--pg-host-config-include-path)	Setting pgBackRest on PostgreSQL host include path	To use internal fixed values
PostgreSQL Host Configuration Path Option (--pg-host-config-path)	Path to configure pgBackRest on the PostgreSQL host	To use internal fixed values
PostgreSQL Host Port Option (--pg-host-port)	SSH Port Specification	No SSH connection required
PostgreSQL Host User Option (--pg-host-user)	The logon user when hosting PostgreSQL, if pg-host is set.	No SSH connection required
PostgreSQL Path Option (--pg-path)	PostgreSQL data directory.	For automatic determination from FEPCluster CR values
PostgreSQL Port Option (--pg-port)	PostgreSQL Ports	For automatic determination from FEPCluster CR values
PostgreSQL Socket Path Option (--pg-socket-path)	PostgreSQL Unix socket path	For automatic determination from FEPCluster CR values
PostgreSQL Database User Option (--pg-user)	PostgreSQL database user	To use internal fixed values

2.3.5.3 Restricted Parameters

Of the parameters in the pgBackRest Configuration Reference, the following parameters limit the configurable values.

Parameter	Overview of parameters	Possible Values
repoX-gcs-key-type	The type of key file you specify when using Google Cloud Storage	service

2.3.5.4 About Sections in the Config File

In FEPCluster CR, you can write the contents of pgbackrest.conf, but the setting for stanza (Backup space for pgBackRest) is specified internally.

The following sections are not allowed;

[stanza: command] , [stanza]

2.3.6 Perform PITR and Latest Backup Restore from Operator

There are two types of restore: one is to restore backup data to an existing FEPCluster, and the other is to create a new FEPCluster and restore backup data.

The former retains the attributes of the FEPCluster, such as IP address and name, while the latter is created from scratch.

The restore process deploys a FEP restore container. The FEP restore container performs the pgBackRest restore operation from the backup data to be restored to the master server of the FEPCluster. After the data is restored to the master server, the FEPCluster is created by synchronizing the data to two replica servers.

If user create a new FEPCluster, the newly created FEPCluster will inherit the settings of the source cluster, unless otherwise specified

User can also create a cluster with different settings from the source cluster by including the settings in FEPCluster CR.

Switching connections to the new cluster

The restore creates a new FEPCluster. If necessary, you need to set up Pgpool-II and change the access point of the application to the new cluster or the new Pgpool-II.

About recovering a failed FEPCluster

Even if the existing FEPCluster fails and the FEP is not running, if the volume of the backup area is safe, it is possible to restore from the backup data.

2.3.7 FEP Unique Feature Enabled by Default

Enable the following FEP features:

- Data masking
- Transparent Data Encryption (TDE)
- Fixed statistics

Data masking

The Data masking is enabled by default in the example FEPClster CR (in OpenShift UI). The postgresql.conf in container contains the following parameters:

```
shared_preload_libraries = 'pgx_datamasking,pg_prewarm'  
session_preload_libraries = 'pg_prewarm'  
max_worker_processes= 20
```

The user can overwrite these values in config map.

TDE

TDE is enabled by default. Select one of the following as the keystore to store the master encryption key used for transparent data encryption.

- File-based keystore
- External key management service

If you use a key management service as your keystore, you can change the keystore to another key management service even after you deploy the FEP cluster. You cannot change from a file-based keystore to a key management service, or from a key management service to a file-based keystore.

Refer to "[2.3.11 Transparent Data Encryption Using a Key Management System](#)" for the design perspective when using a key management system.

Fixed statistics

Fixed statistics(pg_dbms_stats) is enabled by default.

A dbms_stats schema is created in each database.

The list of statistics backups and the list of currently fixed objects can be referenced with pguser defined in the FEPCluster custom resource.

It also takes a backup of statistics by default.

During a major upgrade, the backup statistics maintained in the database are no longer available. The immobilization status will also be released. Take a backup and fix it again after executing the major upgrade.

2.3.8 Monitoring & Alert (FEPExporter)

As the operator is level 5 certified, the system expose various metrics about its operand i.e. FEP containers.

FEP generates lot of useful database statistics via various views. The default statistics can be further augmented by using extensions like pg_stat_statements.

FEPEXporter container by default is configured to extract useful database statistics and make the metrics available to Prometheus on the platform. External components and utilities can be used to visualise, analyse, trigger alerts and take operational decision based on exposed metrics.

FEPEXporter also sets default alert rules based on Prometheus metrics which are useful for active monitoring of FEP cluster.

2.3.8.1 FEPEXporter Custom Resource

Refer to "FEPEXporter Custom Resource" in the Reference for FEPEXporter Custom Resource parameters.

- Custom queries to scrape metrics can be added in CR in optional section.
- Custom Prometheus alert rules are created by user manually.

2.3.8.2 Change to FEPCluster CR - metrics user

User may define pgMetricsUser, pgMetricsPassword and pgMetricsUserTls in target FEPCluster. If it is defined, FEPEXporter will use metrics user details to connect to FEP cluster machines. All metrics user fields are optional and can be omitted in FEPCluster.

Refer to "FEPCluster Parameter" in the Reference for FEPCluster parameters.

2.3.8.3 FEPEXporter CR auto-create for FEPCluster

User may define enableMonitoring flag as part of FEPCluster CR to monitor FEPCluster. It will automatically create FEPCluster specific FEPEXporter so metrics scraping for FEPCluster will work.

Refer to "FEPCluster Parameter" in the Reference for FEPCluster parameters.

- FEPEXporter will be named as <cluster-name>-fepexporter.
- Once FEPEXporter created automatically, user can modify it manually from FEPEXporter CR.
- If FEPCluster will be deleted, it will delete dependent FEPEXporter as well.
- MTLS for FEPEXporter will only supported when tls configuration defined for both Prometheus & FEPEXporter specs.

2.3.9 Scaling Replicas

Auto scale out occurs when the average database CPU utilization or number of connections exceeds the threshold. Select whether the criteria for auto scale out is CPU usage or the number of connections, depending on the resource that is the bottleneck of the database.

The maximum number of replica containers, excluding the master container, is 15.

Scale out based on CPU utilization

Performs a scale out if the average CPU utilization of all pods (primary pods and all replica pods) in the FEPCluster exceeds the threshold for a period of time.

CPU utilization is calculated with the value specified in spec.fep.mcSpec.requests.cpu specified for the FEPCluster custom resource as the denominator.

Scale out based on the number of connections

Performs a scale out if the average number of connections for all pods (primary pods and all replica pods) in the FEPCluster exceeds the threshold for a period of time.

Specify the threshold for the number of connections to perform automatic scale-out with a value less than or equal to the max_connections parameter of the FEP server.

The prerequisites for using the scale out feature based on the number of connections are as follows.

- The monitoring feature (see "[2.3.8 Monitoring & Alert \(FEPEXporter\)](#)") is enabled.
- Metrics for the number of FEP server connections are collected by the monitoring feature.
- A custom metrics server is installed in the OCP/Kubernetes cluster.

- The custom metrics server publishes the average number of connections collected by the monitoring feature.

When using the scale out feature based on the number of connections, the auto scale out feature requests the custom metrics server for metrics associated with the following Kubernetes resources.

- kind: FEPCluster
- apiVersion: fep.fujitsu.io/v2
- name: Name of FEP Cluster
- namespace: The name of the namespace in which FEP Cluster is deployed

The name of the requested metric is the name specified in the metricName parameter.

This metric should represent the average number of connections for each pod in the specified FEPCluster.

Limitations

- If you want to use the scale out feature based on the number of connections, deploy FEPExporter according to the procedure of "[4.3 Deploying FEPExporter](#)".
- If FEPCluster metrics are collected by FEPExporter in standalone mode (see "[4.4 FEPExporter in Standalone Mode](#)"), the scale out feature based on the number of connections is not available.



Note

When using the auto scale out feature, the FEPCluster sync mode should be "off".

Precautions when designing auto scale out

- The auto scale out feature adds replicas one at a time. In addition, additional replicas take time to service, depending on the environment and the amount of data stored. As a result, replica growth may not be able to keep up with the increased load.
- Even if the auto scale out feature increases the number of replicas, incoming requests are not given priority to those replicas. As a result, existing FEP instances may continue to be temporarily overloaded after the number of replicas increases.
- The auto scale out feature increases the number of replica requests that can be handled only by reference requests to the database. Requests with updates continue to be processed on the primary FEP instance. Therefore, the auto scale out feature may not reduce the load on the primary FEP instance.
- Currently, the auto scale out feature does not delete replicas (reduce the number of replicas). If the load decreases after the number of replicas increases due to a temporary increase in load, the number of replicas remains increased. If necessary, manually change the number of replicas.

2.3.9.1 Change to FEPCluster CR - auto scale out

If you want to use Auto Scale Out, set the parameter to FEPClusterCR.

Refer to "FEPCluster Parameter" in the Reference for FEPCluster parameters.

2.3.10 Disaster Recovery

By using object storage, data can be migrated to database clusters in different container environments. Even if it is difficult to operate in a container environment with a database cluster deployed due to a disaster, etc., it is possible to continue operation in a different container environment.

Available disaster recovery methods include the backup/restore method and the hot standby method.

Backup/restore method

Build a new container environment after a disaster (cold standby) and restore data from object storage. Compared to the method described later, this method does not require the construction of two container environments, so it is possible to keep costs down. It takes recovery time to execute.

Hot standby method

Before a disaster occurs, start the container environment of the production environment and the container environment of the disaster recovery environment. By implementing disaster recovery in a hot standby configuration, business systems can be restored more quickly in the event of a disaster.

You can also use Velero to back up a system in a Kubernetes environment to object storage for disaster recovery that restores the system to a different Kubernetes.

2.3.11 Transparent Data Encryption Using a Key Management System

Fujitsu Enterprise Postgres provides unique features that enhance the security of PostgreSQL. These security features help users keep their data safe from unauthorized access. One such security feature is Transparent Data Encryption (TDE), which encrypts data at rest, i.e. data stored on disk/persistent volume.

In contrast, TDE's default format stores the master encryption key in a password-protected file. A key management system allows you to store your master encryption key (MEK) in a cloud-based keystore, taking your security to the next level.

The key management system that can be used with transparent data encryption is one of the following:

- Key management server using KMIP protocol
- AWS key management service (x86 only)
- Azure key management service (x86 only)

Refer to "[Appendix D Key Management System Available for Transparent Data Encryption](#)" for detailed key management system requirements.

Transparent data encryption using a key management system can only be configured when the FEPCluster is first created. Users cannot configure an existing FEPCluster for transparent data encryption using a key management system.

If the master encryption key on the key management system is lost, the encrypted/backup data cannot be decrypted. As long as the data encrypted with the master encryption key remains valid, the master encryption key must also be available and maintained on a key management system.

If you have encrypted backups with old encryption key, you must keep the old encryption key available after the master encryption key is rotated. Otherwise, you will not be able to open the database restored from backup.

In addition, the key custodian must retain the referenced master encryption key for as long as the data encrypted under the old master encryption key remains valid.

2.3.12 Database Role Management

In order to manage data access control, you can easily implement database role privilege and expiration management.

Operators can easily create roles related to database operations, assign privileges, and manage the expiration dates of database roles with login privileges in order to manage data access control.

Databases contain important data such as personal information, and data protection is important.

Data protection is defined in security protocols and is an important aspect of operations.

In order to protect data from being viewed by a third party, it is necessary to properly set the access control of database roles.

In this feature, it is recommended to divide into the following database roles.

- Database administrator: Construction/operation of database system
- Confidential administrator: Set appropriate privileges for each database resource

- General users: End users of the database

By preparing multiple database operators/administrators and assigning privileges to each, it is possible to distribute privileges. This makes it possible to prevent data from being referenced or tampered with by users with strong privileges.

This section describes the roles of database roles created by this feature.

Database administrators can perform operations related to database operations, such as referencing system tables and canceling back-end queries.

Confidential administrators grant appropriate privileges to tables and roles to prevent third parties from viewing data. With this feature, it is possible to grant the confidential administrator the privilege to use the confidentiality management feature, and to grant the appropriate privilege to each database resource.

In addition, it is not recommended to use roles with SUPERUSER or BYPASSRLS privilege that can see all data for data protection. Therefore, in this feature, the SUPERUSER (postgres) password is isolated by hiding it, and the SUPERUSER and BYPASSRLS privileges are not granted to the created database role.

2.3.12.1 Creating Roles Related to Database Operation

2.3.12.1.1 Quarantine SUPERUSER

Create a database role "postgres" with SUPERUSER privileges for the operator when building the database.

By omitting "spec.fepChildCrVal.sysUsers.pgAdminPassword" in the FEPCluster custom resource, the postgres role password is created with a random value, making it impossible for general users to use SUPERUSER privileges. However, a separate method is provided to use the "postgres" role when the administrator needs SUPERUSER privileges for database operations. Therefore, monitor for unexpected usage using the audit feature of pgAudit.

2.3.12.1.2 Database Administrator Role

Database role for database management. Defining this role is mandatory.

The user name and password are defined in "pguser" and "pgpassword" under spec.fepChildCrVal.sysUsers in the FEPCluster custom resource. Has CREATE DATABASE privilege and can see system tables/cancel backend queries.

The database administrator role has the following privileges.

- NOSUPERUSER
- NOREPLICATION
- NOBYPASSRLS
- CREATEDB
- INHERIT
- LOGIN
- CREATEROLE

However, NOCREATEROLE privileges are granted when the confidential administrator role is created.

I also belong to the following roles:

- pg_monitor
- pg_signal_backend

2.3.12.1.3 Confidential Administrator Role

A database role that uses the confidentiality management feature to set appropriate privileges for each database resource for database users. Creating this role is optional.

User name and password are defined in "pgSsecurityUser" and "pgSsecurityPassword" under "spec.fepChildCrVal.sysUsers" of FEPCluster custom resource.

Confidential administrator roles can be defined after building FEPCluster. However, you cannot change the role name or delete the role after defining this role.

This role holds the following privileges.

- LOGIN
- CREATEROLE
- NOSUPERUSER
- NOREPLICATION
- NOBYPASSRLS
- NOCREATEDB
- NOINHERIT

The Confidential administrator role has ALL privileges for the database defined in the FEPCluster custom resource "spec.fepChildCrVal.sysUsers.pgdb", and can create database objects such as tables in the target database.

Confidential administrator roles are assigned the following privileges necessary to operate the confidentiality management feature of Fujitsu Enterprise Postgres, so the confidentiality management feature can be used immediately after the role is created.

- CREATEROLE privilege
- SELECT privilege, INSERT privilege, UPDATE privilege, and DELETE privilege to all tables included in the extension of confidentiality management feature

Grant ownership to the confidential administrator role for the database objects managed by the confidentiality management feature.

In addition, by granting the privileges required for the confidential administrator role to operate the confidentiality management feature to other database roles, the number of users who perform confidential management can be increased and the privileges can be distributed.

2.3.12.2 Expiration Management of Database Roles with Login Privileges

You can control who can connect to the database by managing role expiration.

This section describes two methods for managing role expiration:

- Policy-based password operation
- Password operation with the VALID UNTIL clause

In policy-based password operation, you define a policy as a profile and assign the profile to a role. It is possible to define policies other than expiration exactly as follows:. By assigning the same profile to a role, you can apply the same policy at once.

- Set a password life time
- Restrict password reuse
- Lock accounts that have failed to log in continuously

For password operations with the VALID UNTIL clause, only an expiration date can be set. An operator can force a VALID UNTIL clause to be specified within a specified interval of time when a CREATE or ALTER ROLE command is executed.

2.3.12.2.1 Policy-based Password Operation

For information about how to create and apply profiles that define policies, refer to "Policy-based Login Security" in the Fujitsu Enterprise Postgres Operation Guide.

When FEPCluster is deployed, users for streaming replication (User specified in spec.fepChildCrVal.sysUsers.pgrepuser of the FEPCluster custom resource) are automatically assigned membership in the pgx_update_profile_status group role. This ensures that the user lock status and password status detected on the standby server are applied to the entire cluster immediately after the FEPCluster is built.

Operators can use the monitoring feature to monitor metrics such as expiration.

When you deploy FEPExporter, the following metrics collection and alert rules are enabled:

To deploy the FEPEXporter, refer to ["4.3 Deploying FEPEXporter"](#) or ["4.4 FEPEXporter in Standalone Mode"](#).

For definitions of metric queries and alert rules, refer to "Default Metric Queries" and "Default Alert Rules" in the Reference.

Metrics Collected by Default

- Number of all roles
- Number of enabled roles
- Number of roles in grace period
- Number of roles that have expired
- Number of roles locked

You can also add metrics queries to monitor, for example, how many roles expire within a specified date.

Default Alert Rules

- When Expired Role is 1 or More
- When the Grace Period Expired Role is 1 or More
- When the locked role is 1 or more

When you are notified of an alert, you can verify the database role in question from the system catalog.

For more information about the system catalogs for managing profile information, refer to "System Catalogs" in the Fujitsu Enterprise Postgres Operation Guide.

2.3.12.2 Password Operation with the VALID UNTIL Clause

You can manage password expiration for database roles with login privileges.

When defining passwords for database roles with login privileges in the CREATE ROLE or ALTER ROLE statements, it is possible to force them to expire within a specified period.

Specify the following parameters in the FEPCluster custom resource to enable this feature.

- Specify "fsep_operator_security" in shared_preload_libraries of spec.fepChildCrVal.customPgParams
- "spec.fepChildCrVal.sysUsers.passwordValid.days" specifies the number of days that can be specified from the time the password is changed to the expiration date

FEPCluster custom resource definition example

```
fepChildCrVal:
  customPgParams: |
    shared_preload_libraries='pgx_datamasking,pg_prewarm,pg_stat_statements,fsep_operator_security'
    ...
  sysUsers:
    passwordValid:
      days: 30
    pgdb: mydb
    pgpassword: mydbpassword
    pguser: mydbuser
```

When this feature is enabled, the password expiration for pgpassword and pgSecurityPassword, as defined in spec.fepChildCrVal.sysUsers in the FEPCluster custom resource, is defined after the length of time specified in spec.fepChildCrVal.sysUsers.passwordValid.days since the password was changed.

However, passwords defined in pgAdminPassword, pgreplpassword, pgRewindUserPassword, and pgMetricsUserPassword are database role passwords required for database operation management, so their expiration dates are not managed.

In addition, if the CREATE ROLE or ALTER ROLE statement defines/changes the password for a database role that has login privileges, and the password for the database role does not expire or is longer than the length of time specified by spec.fepChildCrVal.sysUsers.passwordValid.days, the executed SQL will fail.

`spec.fepChildCrVal.sysUsers.passwordValid.days` can be defined or changed after building the `FEPCluster`. Changing this parameter updates the password expiration period for all managing database roles that have not expired.

Removing `spec.fepChildCrVal.sysUsers.passwordValid.days` or setting it to 0 will stop password expiration management.

By using the `FEPExporter` custom resource feature, it is possible to monitor the password expiration date of database roles and send an alert using `AlertManager` when there is a database role that is about to expire or has passed.

2.3.13 User Synchronization Using `ldap2pg`

`ldap2pg` manages database roles in an LDAP-configured system by applying management information such as users on the LDAP server to the database server. `ldap2pg` allows automatic synchronization of LDAP directories with FEP user accounts. The synchronisation is done by `ldap2pg` on a regular basis. The frequency of synchronisation can be defined as a cronjob like entry. Connectivity from `ldap2pg` to LDAP directory and FEP to LDAP directory must support LDAP, LDAPS and LDAP over TLS.

Built-in Operator users such as `postgres`, `repluser`, `rewind_user`, `mydbuser` should not be created/deleted/modified by `ldap2pg` as it may interfere with normal operation of FEP Operator and FEP Cluster. The default `ldap2pg.yml` supplied by the FEP Operator excludes these users.

To use `ldap2pg`, you need the following:

LDAP connection info

Details about LDAP directory, include `ldap uri`, `basedn`, `binddn` must be defined and made available to `ldap2pg`. The `ldap.conf` is used to store such information. It will be available to any software that needs connectivity information to LDAP directory, including `ldap2pg` and FEP.

Details of `ldap.conf` will be stored in a secret and mounted to `fep-patroni` under `/etc/openldap/ldap.conf`. End user should pre-create this secret before enabling `ldap2pg` synchronisation. If this secret is not present when `ldap2pg` is enabled, Operator will create a sample `ldap.conf` with connectivity info commented out. End user can update the secret with corresponding LDAP connectivity info. This updated information will be reflected in `fep-patroni` when `kubelet` performs `syncPod` function.

The Secret can be named as user-preferred, e.g., `<fep-cluster>-ldapconf`. Its key must be “`ldap.conf`” and its value should contain LDAP configuration.

Trusted CA bundle for LDAP directory

If the LDAP directory is using certificates signed by private CA to secure connections, the chain of certificates up to the root CA must be provided for `ldap2pg` and FEP as trusted certificates for a secure connection, be it `ldaps` or `ldap over TLS`. This chain of certificates will be stored in a configmap and mounted to `fep-patroni` under `/tls/ldap/ca.crt`. The use of such trusted certificates is optional but is recommended to authenticate the authenticity of the LDAP servers.

The ConfigMap can be named as user-preferred, e.g., `<fep-cluster>-cacrt`. Its key must be “`ca.crt`” and its value should contain LDAP server certificates which are provided by end-user.

`ldap2pg.yml`

Configuration file for `ldap2pg` to interact with LDAP directory and FEP.

Details of `ldap2pg.yml` will be stored in a configmap and mounted to `fep-patroni` under `/tmp/.config/ldap2pg.yml`. End user should pre-create this configmap before enabling `ldap2pg` synchronisation. If this configmap is not present when `ldap2pg` is enabled, Operator will create a sample `ldap2pg.yml`. End user can update the configmap. The updated information will be reflected in `fep-patroni` when `kubelet` performs `syncPod` function.

The ConfigMap can be named as user-preferred, e.g., `<fep-cluster>-ldap2pgyaml`. Its key must be “`ldap2pg.yml`” and its value should contain `ldap2pg` configuration.

2.3.13.1 How to Update `FEPCluster` CR for `ldap2pg`

This section describes instructions for users to update `FEPCluster` CR to support `ldap2pg`.

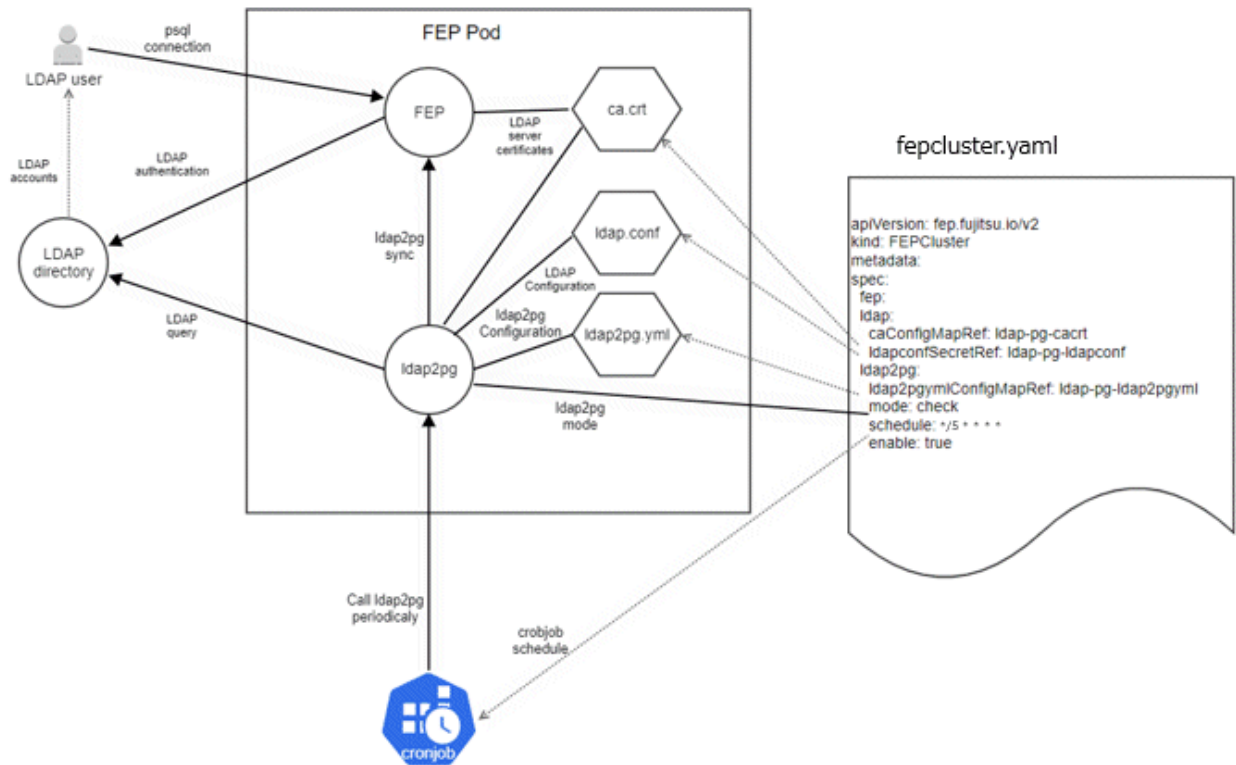
First, it will need to create resources below for both LDAP configuration and `ldap2pg` configuration.

- Secret for LDAP Configuration (`ldap.conf`)
- ConfigMap for LDAP Server Certificates (`ca.crt`)

- ConfigMap for ldap2pg Configuration (ldap2pg.yml)

Second, it will need to add spec.ldap and spec.ldap2pg entries to a FEPCluster CR below to support ldap2pg (refer to figure below).

- FEPCluster with ldap2pg enabled (fepcluster.yaml)



Follow the steps below to update the FEPCluster CR for ldap2pg.

Each step includes the approach by using OpenShift console and command lines.

1. Create a Secret for LDAP configuration (ldap.conf)

LDAP configuration includes LDAP connection info.

In this example, we assume that the LDAP server is running at ldap-server.testprj.svc, listening for LDAPS connection on port 636. The base DN is dc=example,dc=org. The LDAP user cn=admin,dc=example,dc=org is used to connect to LDAP server and the password is "mypassword".

Sample LDAP configuration (ldap.conf)

```
$ cat ldap.conf

#
# LDAP Defaults
#

#Refer to https://www.openldap.org/software/man.cgi?query=ldap.conf for full configuration
details
# This file should be world readable but not world writable.

#BINDDN: This is the user to bind to LDAP server
BINDDN cn=admin,dc=example,dc=org

#BASE: Set it to the base DN where search will begin
BASE dc=example,dc=org
```

```
#PASSWORD: Password of user as defined in BINDDN
PASSWORD mypassword

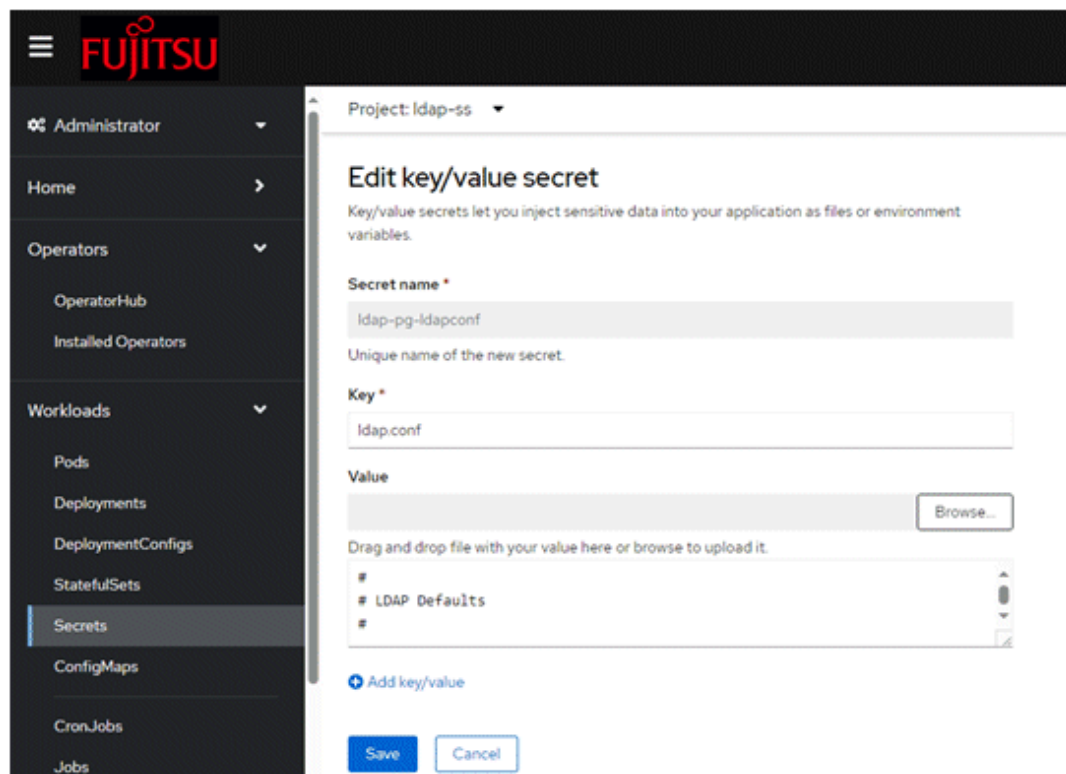
#URI: Specify the URI of LDAP server
URI ldaps://ldap-server.testprj.svc:636

#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never
TIMEOUT         15

# TLS certificates (needed for GnuTLS)
# TLS_CACERT     /tls/ldap/ca.crt

# Specifies what checks to perform on server certificates in a TLS session. We are
# being relax here even if server certificate cannot be verified.
TLS_REQCERT allow
```

- Create a Secret by OpenShift.



- Create a Secret by command-lines

```
$ kubectl -n ldap-ss create secret generic ldap-ss-ldapconf --from-file=ldap.conf
secret/mysecret created
```

2. Create a ConfigMap with chain of certificates (ca.crt) that sign the LDAP server certificate

Sample LDAP server certificates (ca.crt)

```
$ cat ca.crt

-----BEGIN CERTIFICATE-----
xxx root cert xxx
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----  
yyy intermediate cert yyy  
-----END CERTIFICATE-----
```

- Create a ConfigMap by OpenShift.

Project: ldap-ss

Create ConfigMap

Config maps hold key-value pairs that can be used in pods to read application configuration.

Configure via: ☒ Form view ☐ YAML view

Name *

ldap-ss-cacrt

A unique name for the ConfigMap within the project

☐ Immutable
Immutable, if set to true, ensures that data stored in the ConfigMap cannot be updated

Data

Data contains the configuration data that is in UTF-8 range

Remove key/value

Key *

ca.crt

Value

ca.crt.txt Browse...

Drag and drop file with your value here or browse to upload it.

-----BEGIN CERTIFICATE-----

Add key/value

Create Cancel

- Create a ConfigMap by command-lines

```
$ kubectl -n ldap-ss create configmap ldap-ss-cacrt --from-file=ca.crt  
configmap/ldap-ss-cacrt created
```

3. Create a ConfigMap for ldap2pg.yml file

The following users are managed by operators, so specify them in roles_blacklist_query in the postgres section to remove them from ldap2pg management.

- postgres user
- Username specified in spec.fepChildCrVal.sysUsers of FEPCluster custom resource

For other ldap2pg.yml settings, refer to "ldap2pg" in the Fujitsu Enterprise Postgres Installation and Setup Guide for Client.

Sample ldap2pg configuration (postgres section of ldap2pg.yml)

```
postgres:  
  roles_blacklist_query: [postgres, repluser, rewind_user, pwsyncuser, pg_*, pgx_*, ]  
  databases_query: [postgres]
```


- Create a ConfigMap by OpenShift.

- Create a ConfigMap by command-lines

```
$ kubectl -n ldap-ss create configmap ldap-ss-ldap2pgyaml --from-file=ldap2pg.yaml
configmap/ldap-ss-ldap2pgyaml created
```

4. Add spec.ldap and spec.ldap2pg entries to FEPCluster CR (fepcluster.yaml)

It will need to add the entries below to FEPCluster CR during the creation and updating of FEPCluster.

Key	Value	Description
spec.ldap.caConfigMapRef	ldap-ss-cacrt	Refer to step 1
spec.ldap.ldapconfSecretRef	ldap-ss-ldapconf	Refer to step 2
spec.ldap2pg.ldap2pgyamlConfigMapRef	ldap-ss-ldap2pgyaml	Refer to step 3
spec.ldap2pg.mode		Default value is "check"
spec.ldap2pg.schedule		Default value is '*/* * * *'
spec.ldap2pg.enable		Default value is true

You can disable ldap2pg feature by setting the spec.ldap2pg.enable to false.

Sample FEPCluster CR with ldap2pg changes (ldap2pg is enabled) (fepcluster.yaml)

```
$ cat fepcluster.yaml

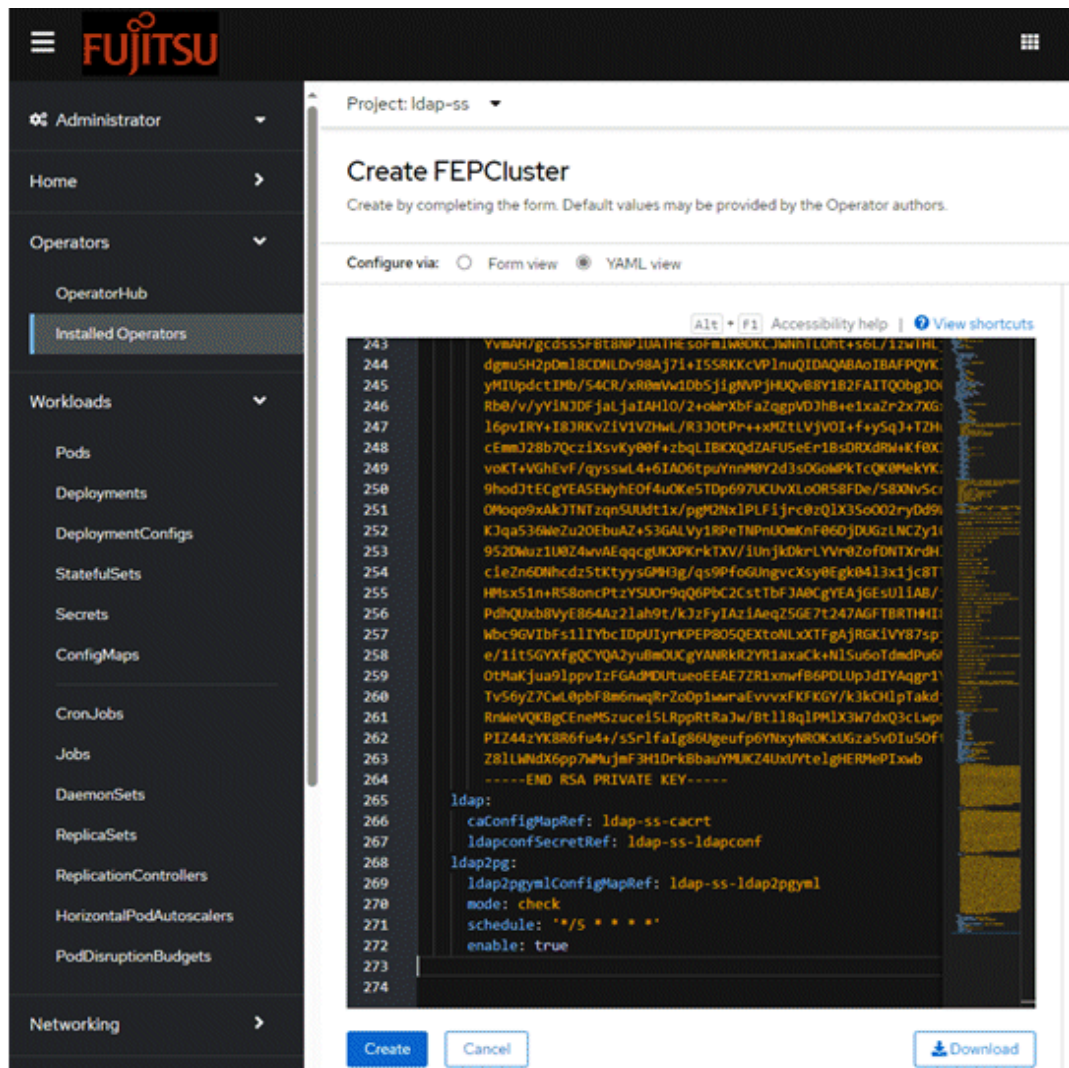
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  name: ldap-ss
```

```

namespace: ldap-ss
spec:
  ldap:
    caConfigMapRef: ldap-ss-cacrt
    ldapconfSecretRef: ldap-ss-ldapconf
  ldap2pg:
    ldap2pgyamlConfigMapRef: ldap-ss-ldap2pgyaml
    mode: check
    schedule: '*/* * * * *'
    enable: true

```

- Update FEPCluster CR by OpenShift.



- Update FEPCluster CR by command-lines

```
$ kubectl apply -f fepcluster.yaml
```

2.3.14 Fixed Statistics

Fixed statistics means tuning the intended statistics so that they are always used to prevent performance degradation due to changes in query plans. This is achieved by the `pg_dbms_stats` extension. Operator can schedule statistics to be fixed. The user simply defines a schedule in the FEPCluster custom resource, and the operator automatically fixes the statistics that the user intended.

The following figure shows the procedure for executing the schedule. For more information, refer to "5.14 Operation of Fixed Statistics".

STEP1: Verify the performance of the statistics in the FEPCluster validation environment

STEP2: Store statistics from FEPCluster in validation environment to object storage

STEP3: Define a schedule for fixing statistics to FEPCluster custom resources in a production environment

STEP4: Deploy FEPCluster custom resources for production and schedule statistics consolidation

The binary file download of statistics from object storage occurs as needed.

You can also back up statistics on a regular basis. In the event of performance degradation due to changes in query plan, the backed up statistics can be used to achieve performance stabilization.

pg_dbms_stats allows you to do the following:

- Backup
- Restore
- Lock
- Unlock
- Purge
- Cleanup
- Import

Use FEPAAction to perform various operations. For more information, refer to "FEPAAction Specific Operation Details" in the Reference.

Chapter 3 Operator Installation

This chapter describes how to install FEP operator.

Refer to "[6.5 Assigned Resources for Operator Containers](#)" for more information about the resources assigned to installed operator containers and how to change them.

"<x.y.z>" in the screen example indicates the version level of the operator. Also, "<X>" indicates the product version of Fujitsu Enterprise Postgres.



If you are using anti-virus software, exclude the directory for resource allocation from virus scanning.

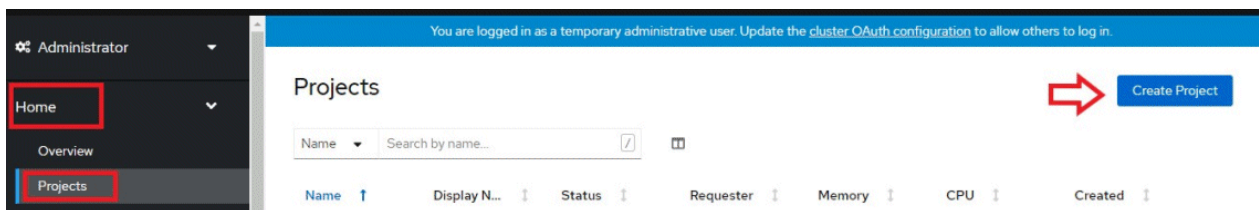
3.1 Using the OperatorHub

Describes how to use OperatorHub to install FEP operators into a new namespace on Openshift.

3.1.1 Pre-requisite

A project on openshift is essentially a namespace. It is a good practice to install FEP in a separate name space. On the RedHat OpenShift platform, click "Home" under "Projects" main menu and hence click on "Create Project".

(Screen Shot 1 and 2 - Create Project on OCP - *for ref.*)



In the dialog box, specify a unique name for your namespace and an optional display name and description.

Create Project

Name * ?

Display name

Description

Cancel Create



Note

Operator installation needs Prometheus to be pre-installed in the Openshift cluster.

3.1.2 Deploying Operator

Once operator is certified by RedHat, it is made available on OperatorHub on all RedHat OpenShift container platform.

On OpenShift platform, logon with credentials that has privileges to install operator. Click on OperatorHub on menu item under Operators and type filter keyword "Fujitsu Enterprise Postgres Operator" to find Fujitsu Enterprise Postgres Operator.

The screenshot shows the Red Hat OpenShift OperatorHub interface. The left sidebar contains a navigation menu with the following items: Administrator, Home, Overview, Projects, Search, API Explorer, Events, Operators (selected), OperatorHub (selected), Installed Operators, Workloads, Pods, Deployments, DeploymentConfigs, StatefulSets, and Secrets. The main content area displays the 'OperatorHub' page. At the top, there is a header with the Red Hat OpenShift logo and a dropdown menu for the project 'fep-install-test'. Below the header, the 'OperatorHub' section is visible, featuring a search bar and a list of operators. The 'Fujitsu Enterprise Postgres Operator' is highlighted in the search results, showing its logo and name. The operator is provided by Fujitsu and is an OpenShift Operator for Fujitsu Enterprise Postgres.

Click on Fujitsu Enterprise Postgres Operator to install operator. It will bring up details page with install button as below.



Fujitsu Enterprise Postgres Operator

<x.y.z> provided by Fujitsu

Install

Latest version
<x.y.z>

Capability level

- Basic Install
- Seamless Upgrades
- Full Lifecycle
- Deep Insights
- Auto Pilot

Source
Certified

Provider
Fujitsu

Repository
N/A

Container image
quay.io/fujitsu/fujitsu-enterprise-postgres-operator

Fujitsu Enterprise Postgres <X> delivers an enterprise-grade PostgreSQL on OpenShift Container Platform.

This solution provides the flexibility of a hybrid cloud solution while delivering an enhanced distribution of PostgreSQL to support enterprise-level workloads and provide improved deployment and management, availability, performance, data governance and security.

Available as a multi-architecture container built for both amd64, s390x and ppc64le.

The download and Use of the Product is strictly subject to the terms of the End User License Agreement with Fujitsu Limited found at <https://www.fast.fujitsu.com/fujitsu-enterprise-postgres-license-agreements>. Where the Product that has been embedded as a whole or part into a third party program, only Authorised Customers may download and use the Product.

Click on "Install" button, to bring up following screen to choose namespace and approval strategy. Select "A specific namespace on the cluster" and choose desired namespace. Leave everything else to default and click install.

Administrator

Home

Overview

Projects

Search

API Explorer

Events

Operators

OperatorHub

Installed Operators

Workloads

Networking

Storage

Builds

Observe

Compute

OperatorHub > Operator Installation

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel *

stable

Installation mode *

All namespaces on the cluster (default)

This mode is not supported by this Operator

A specific namespace on the cluster

Operator will be available in a single Namespace only.

Installed Namespace *

fepp-install-test

Update approval *

Automatic

Manual

Install

Cancel

Fujitsu

Fujitsu Enterprise Postgres Operator

provided by Fujitsu

Provided APIs

FEPC FEPCluster

Not available

FEPA FEPAAction

Not available

FEPE FEPEExporter

Not available

FEPL FEPLLogging

Not available

FEPP FEPPgpool2Cert

Not available

FEPP FEPPgpool2

Not available

FEPR FEPRRestore

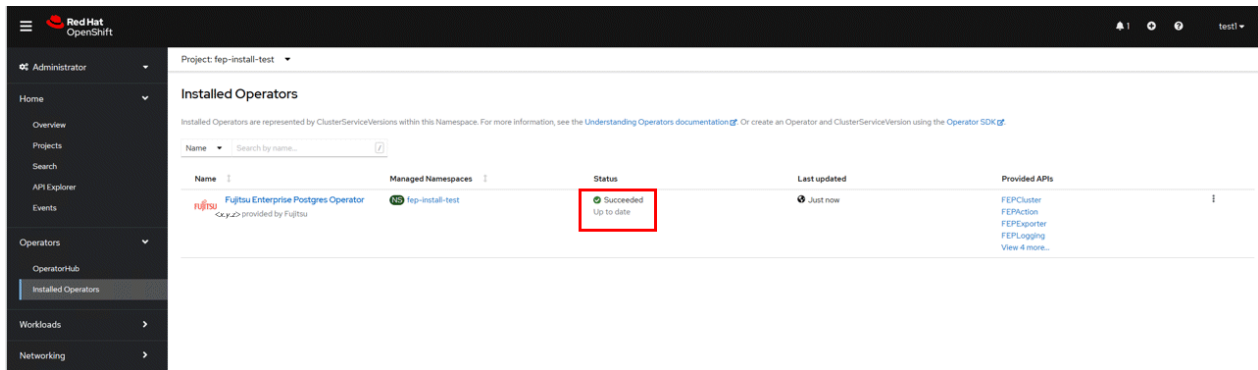
Not available

FEPU FEPUUpgrade

Not available

- 36 -

Wait still installation is complete and status changes to "Succeeded".



3.1.3 Upgrading Operators

If you select Automatic for "Update approval" on the "Install Operator" screen under "3.1.2 Deploying Operator", the operator will be automatically upgraded when a new version is published.

If Manual is selected for "Update approval", perform the following steps to upgrade manually.

1. Select the installed "Fujitsu Enterprise Postgres Operator" from [Operators] - [Installed Operators].
2. Follow the on-screen instructions to upgrade your operator from [Subscription details] in the [Subscription] tab

3.2 Using the Helm Chart

Describes how to install FEP operators into a new namespace on Kubernetes using the Helm feature.

3.2.1 Deploying Operator

1. Add a Helm Chart repository for the operator.

```
helm repo add fep-repo https://fujitsu.github.io/fep-operator-helm/v1
```

2. Create a namespace to install the operator.

```
kubectrl create namespace fep-operator
```



Note

Operator installation needs Prometheus to be installed in the Kubernetes cluster in advance.

3. Run the helm command to install the operator.

```
helm install fep-operator-release fep-repo/fujitsu-enterprise-postgres-operator --namespace fep-operator
```

3.2.2 Upgrading Operators

1. Refresh Helm Chart repository information.

```
helm repo update
```

2. Check the Helm Chart version of the latest operator.

```
helm search repo fujitsu-enterprise-postgres-operator
```

3. Run the helm command to upgrade the operator.

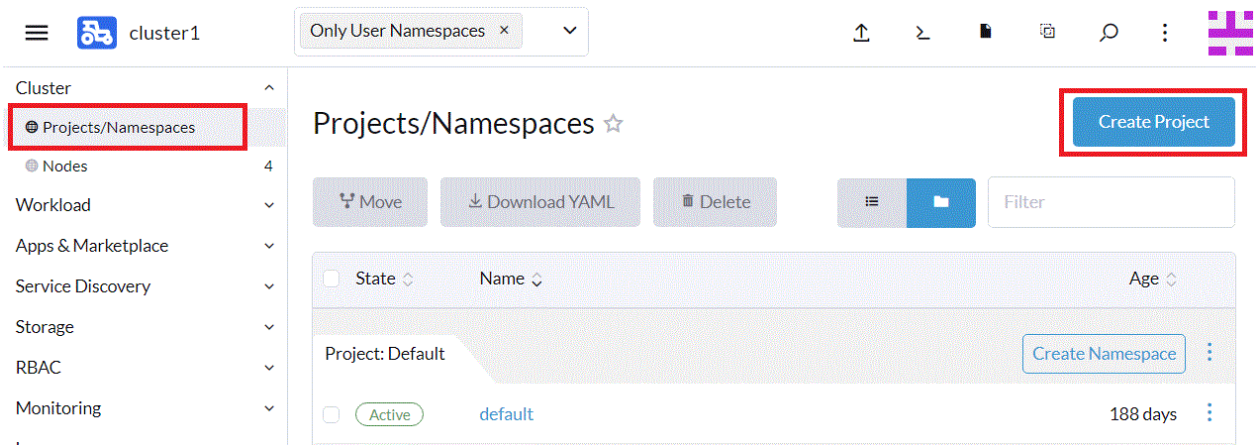
```
helm upgrade fep-operator-release fep-repo/fujitsu-enterprise-postgres-operator --namespace fep-operator
```

3.3 Using the Rancher UI

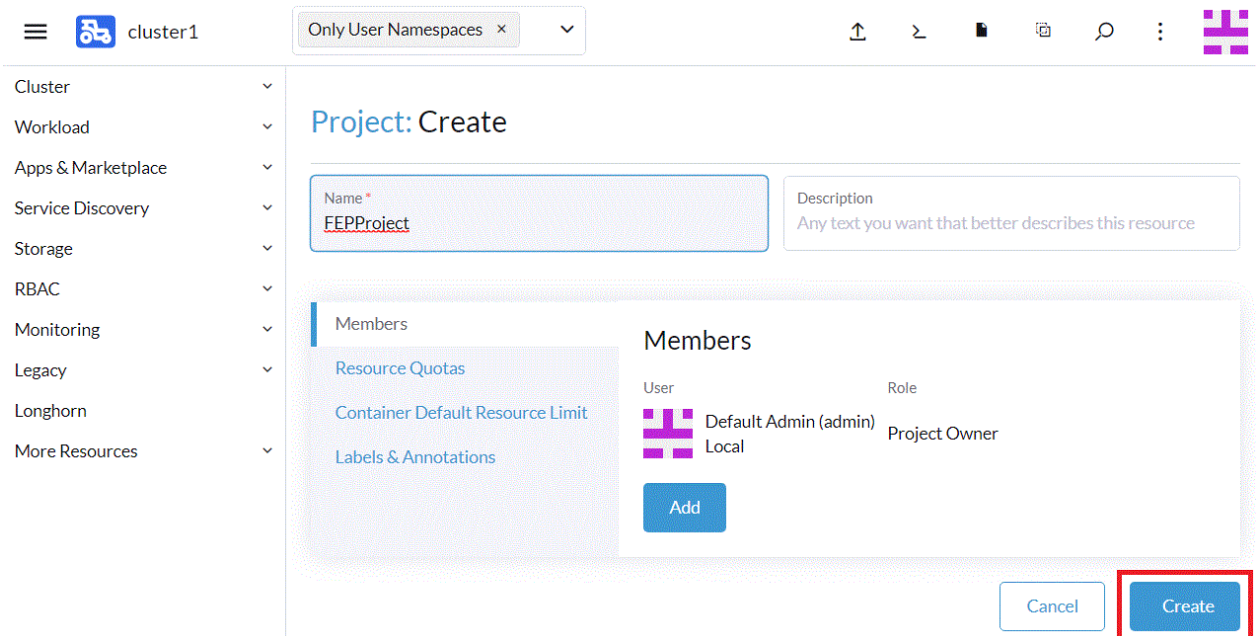
Describes how to install FEP operators into a new namespace on Rancher UI.

3.3.1 Pre-requisite

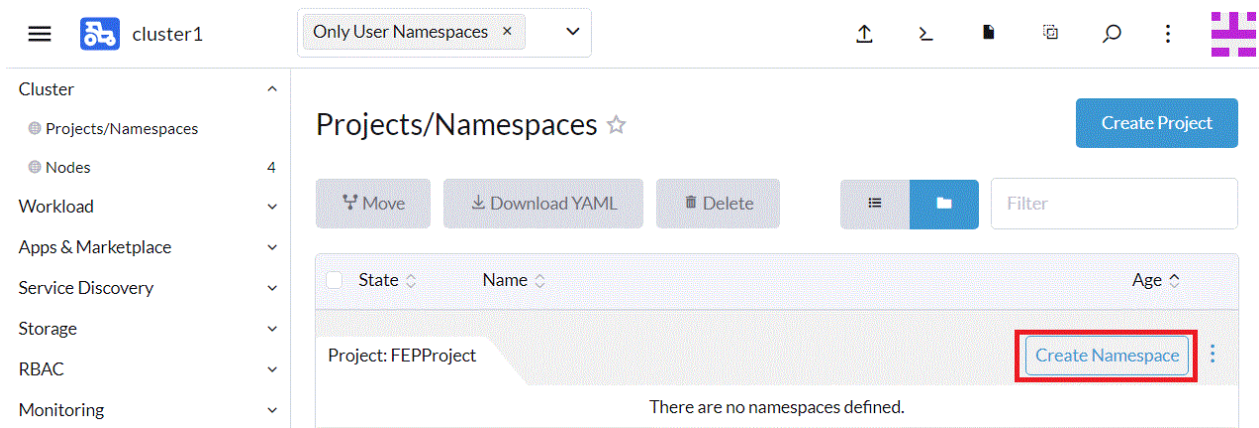
Create a project and its associated namespace on the Rancher UI. We recommend that you install FEP in a different namespace. In the Rancher UI, click [Projects/Namespaces], then click [Create Project] that appears.



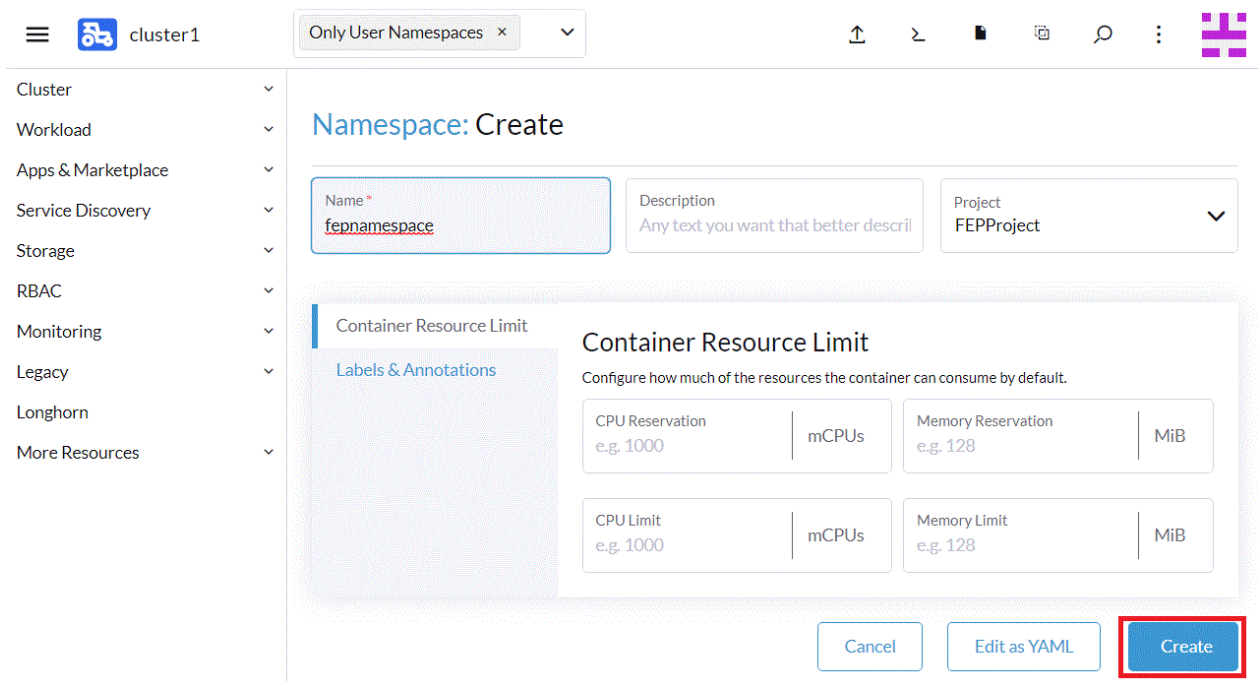
Specify a unique name for the project and click [Create].



Click [Create Namespace] displayed on the specified project.



Specify a unique name in the namespace and click [Create].



3.3.2 Register Helm Chart Repository

Register the Helm Chart repository of the operator feature on the Rancher UI.

In the Rancher UI, click [Apps & Marketplace], then click [Repositories] that appears.

cluster1 Only User Namespaces

Cluster
Workload
Apps & Marketplace
Charts
Installed Apps 4
Repositories 3
Recent Operations 0
Service Discovery
Storage
RBAC
Monitoring
Legacy

A chart repository is a Helm repository or Rancher git based application catalog. It provides the list of available charts in the cluster.

Repositories ☆

Create

Refresh Download YAML Delete Filter

State	Name	Type	URL	Branch	Age
Active	Partners	git	https://git.rancher.io/partner-charts	main	188 days
Active	Rancher	git	https://git.rancher.io/charts	release-v2.6	188 days

Click [Create] to create the Helm Chart repository.

cluster1 Only User Namespaces

Cluster
Workload
Apps & Marketplace
Charts
Installed Apps 4
Repositories 3
Recent Operations 0

A chart repository is a Helm repository or Rancher git based application catalog. It provides the list of available charts in the cluster.

Repositories ☆

Create

Refresh Download YAML Delete Filter

Enter the unique name of the catalog and the URL of the catalog below, and click [Create].

https://fujitsu.github.io/fep-operator-helm/v1

cluster1

Cluster
Workload
Apps & Marketplace
Charts
Installed Apps
Repositories
Recent Operations
Service Discovery
Storage
RBAC
Monitoring
Legacy
Longhorn
More Resources

Only User Namespaces

⬆ ⬇ 📄 🔍 ⋮

Repository: Create

Name *

fep-operator-helm

Description

Any text you want that better describes this resource

Target

☒ http(s) URL to an index generated by Helm

☐ Git repository containing Helm chart or cluster template definitions

Index URL *

https://fujitsu.github.io/fep-operator-helm/v1

Authentication

None

Labels

Add Label

Annotations

Add Annotation

Cancel

Create

3.3.3 Deploying Operator

On the Rancher UI, apply the operator function Helm Chart to the project / namespace created in ["3.3.1 Pre-requisite"](#) and install the operator.

From the leftmost tab, click [Charts], then click [fujitsu-enterprise-postgres-operator].

cluster1

Only User Namespaces x

Cluster

Workload

Apps & Marketplace

Charts

Installed Apps

Repositories

Recent Operations

Service Discovery

Storage

Charts

fep-operator-helm

All Categories

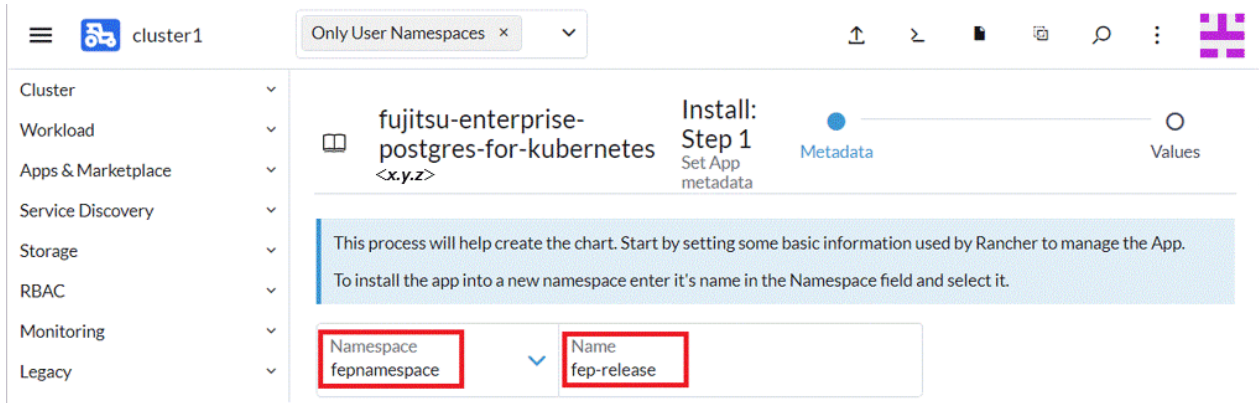
Filter

fujitsu-enterpris...
Helm Chart for
Installation of
Fujitsu Enterprise...

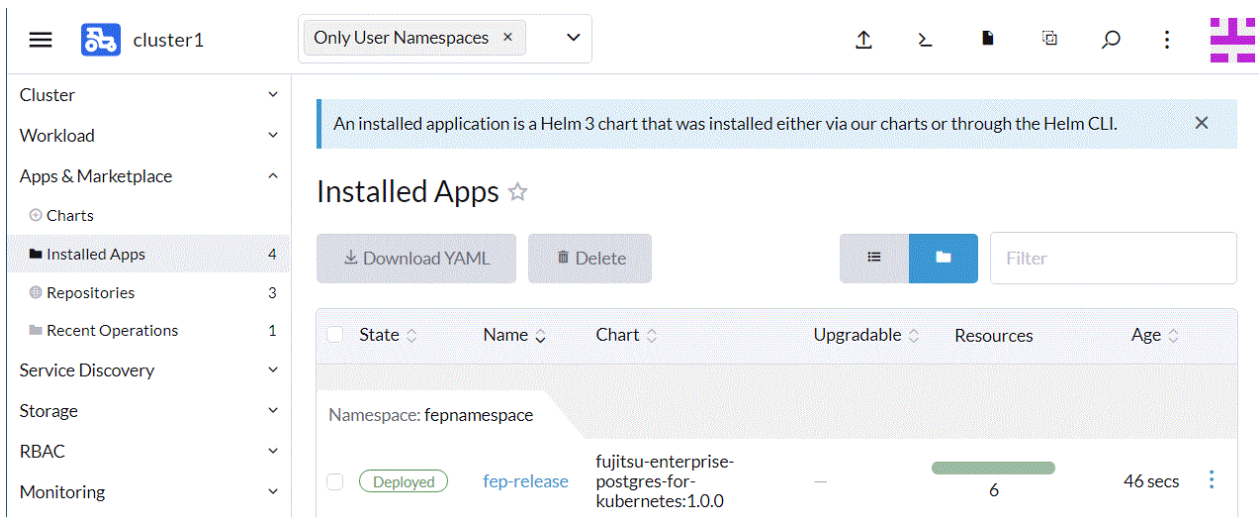
Click Install on the screen that appears.



Change the [Namespace] item to the name created in "3.3.1 Pre-requisite", enter the release name in the [Name] item, click [Next], and then click [Install] on the next screen.



The operator is deployed on the target namespace.



3.3.4 Upgrading Operators

1. Update the Helm Chart repository.

In the Rancher UI, click [Apps & Marketplace] and then click [Repositories] that appears.

Click Refresh to update the repository.

2. Upgrade the operator.

In the Rancher UI, click Apps & Marketplace, then click Installed Apps.

Select the operator you installed and follow the on-screen instructions to upgrade the operator.

3.4 Implement Collaborative Monitoring Tools

3.4.1 Implement GAP Stack

There is a pre-requisite for running FEPEXporter.

- GAP(Grafana, AlertManager, Prometheus) stack is installed on host OpenShift or Kubernetes cluster
- FEPCluster that needs to be scraped is deployed and running properly
- FEPCluster has following setting postgresql.conf:
 - pg_stats_statements library pre-loaded
 - track_activities and track_counts are turned on

For Prometheus and AlertManager, use the monitoring stack preinstalled on Openshift. Please refer to the following for deployment information.

(https://docs.openshift.com/container-platform/4.11/monitoring/monitoring-overview.html#understanding-the-monitoring-stack_monitoring-overview)

For Grafana, install and use the GrafanaOperator provided by OperatorHub. Grafana is not exposed by OperatorHub in s390x and ppc64le, so use Helm to build Grafana. Detailed instructions are available at the following site for your reference.

(<https://www.postgresql.fastware.com/knowledge-base/how-to/setting-up-grafana-on-ibm-linuxone>)

Grafana comes pre-installed on OpenShift, but it is recommended to use Grafana published in OperatorHub to customize the dashboard and monitor FEP performance information.

You can also use the sample dashboards published below.

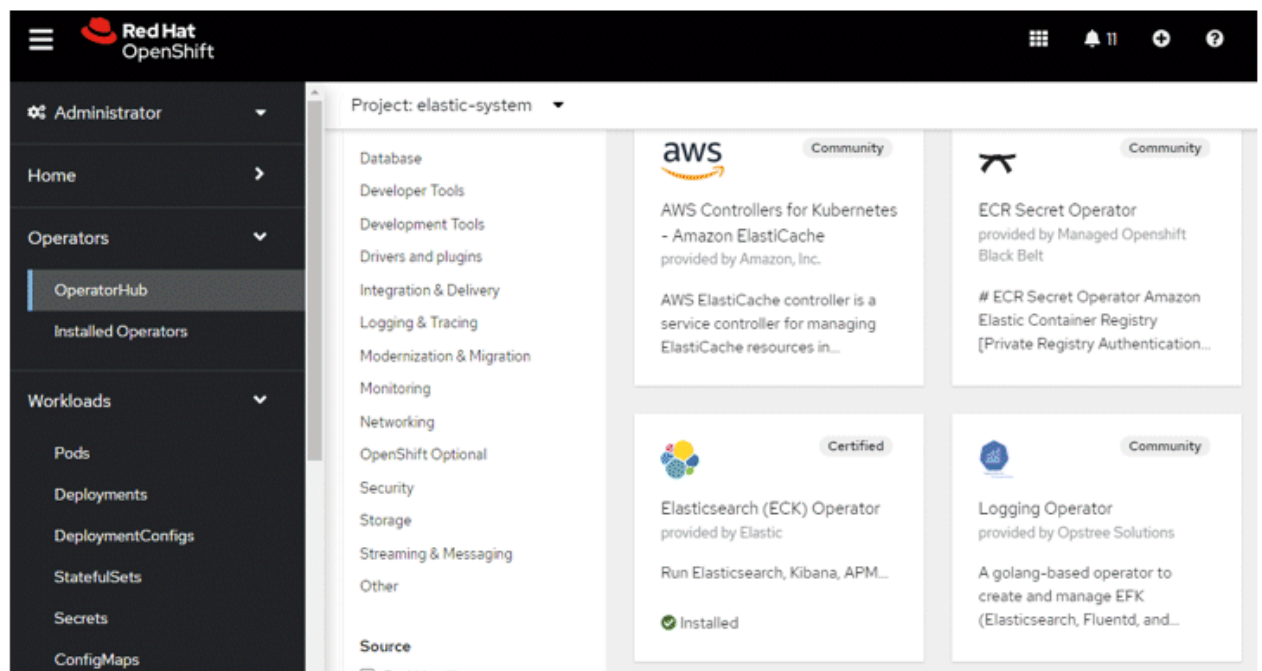
<https://github.com/fujitsu/fep-operator-examples/blob/v4/Monitoring/dashboard/fep-dashboard.json>

3.4.2 Implement Elastic Cloud on Kubernetes

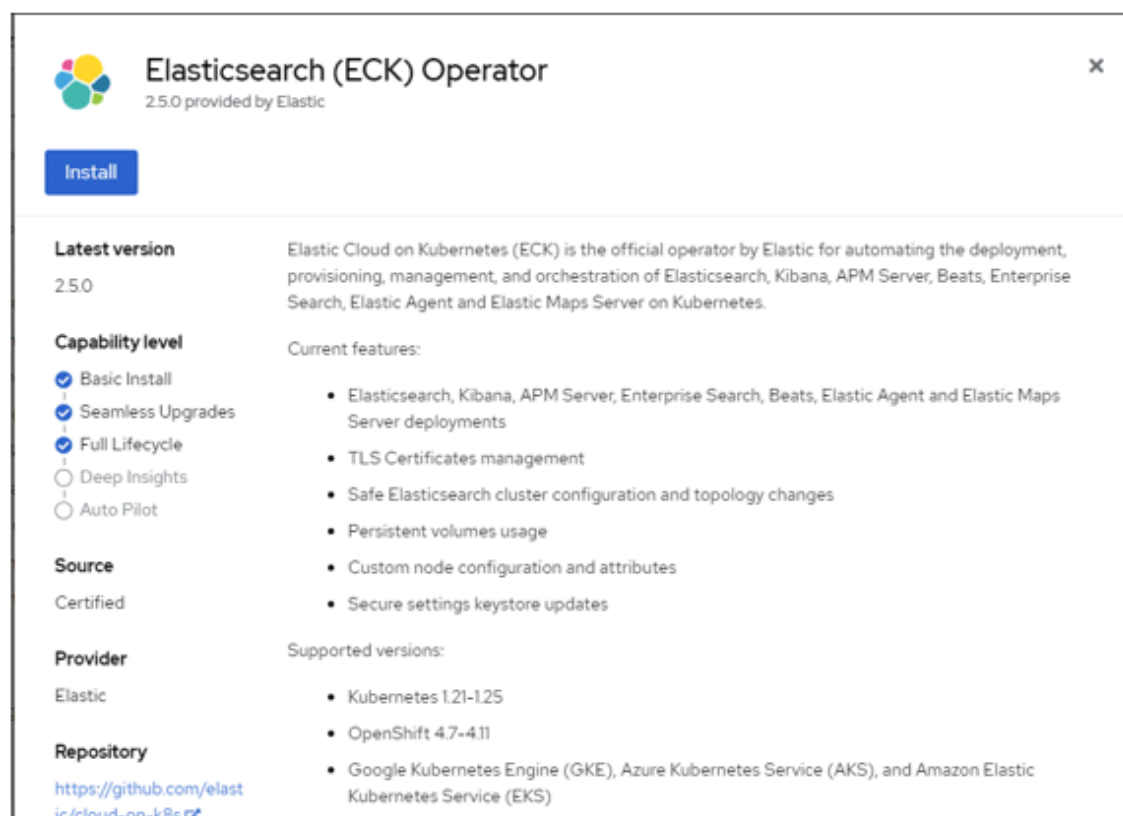
3.4.2.1 Deploy ECK Operator

1. Create namespace(project) elastic-system.

2. In OperatorHub, install Elasticsearch (ECK) Operator provided by Elastic.



3. Click Install to start proceed.



4. Change the Installation mode to "A specific namespace on the cluster" and select namespace "elastic-system". Click Install to complete the installation.

OperatorHub > Operator Installation

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either automatic updates.

Update channel * ⓘ

☒ stable

Installation mode *

☐ All namespaces on the cluster (default)
Operator will be available in all Namespaces.

☒ A specific namespace on the cluster
Operator will be available in a single Namespace only.

Installed Namespace *

PR elastic-system ▼

Update approval * ⓘ

☒ Automatic

☐ Manual

Elasticsearch (ECK) Operator
provided by Elastic

Provided APIs

AS APM Server

APM Server instance

E Elasticsearch Cluster

Instance of an Elasticsearch cluster

ES Enterprise Search

Enterprise Search instance

3.4.2.2 Deploy Elasticsearch Cluster

1. In Installed Operators, select "Elasticsearch (ECK) Operator".
2. Select "Elasticsearch Cluster" and "Create Elasticsearch".

Project: elastic-system ▼

Installed Operators > Operator details

 **Elasticsearch (ECK) Operator**
2.5.0 provided by Elastic

Actions ▼

description Events All instances APM Server Elasticsearch Cluster Enterprise Search Kibana Beats

Elasticsearchs Create Elasticsearch

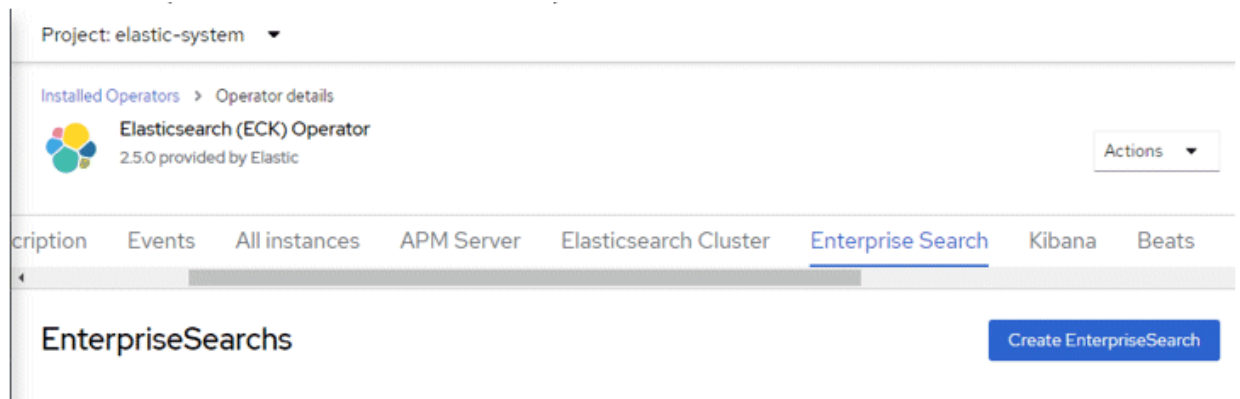
3. In YAML view, enter the following details and click "Create".

```
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: quickstart
```

```
spec:
  version: 8.5.2
  nodeSets:
  - name: default
    count: 1
    config:
      node.store.allow_mmap: false
```

3.4.2.3 Deploy Enterprise Search

1. In Installed Operators, select "Elasticsearch (ECK) Operator".
2. Select "Enterprise Search" and "Create EnterpriseSearch".

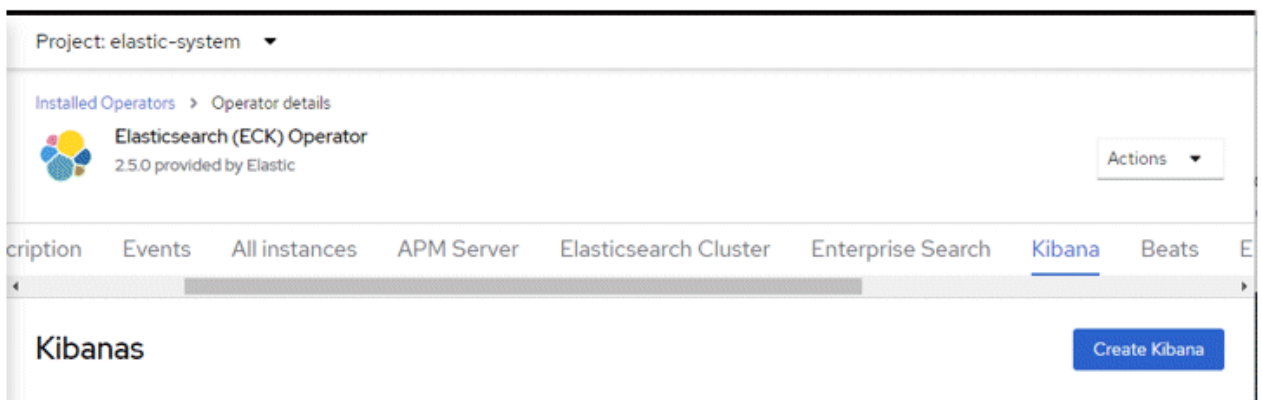


3. In YAML view, enter the following details and click "Create".

```
apiVersion: enterprisearch.k8s.elastic.co/v1
kind: EnterpriseSearch
metadata:
  name: enterprise-search-quickstart
spec:
  version: 8.5.2
  count: 1
  elasticsearchRef:
    name: quickstart
```

3.4.2.4 Deploy Kibana

1. In Installed Operators, select "Elasticsearch (ECK) Operator".
2. Select "Kibana" and "Create Kibana".



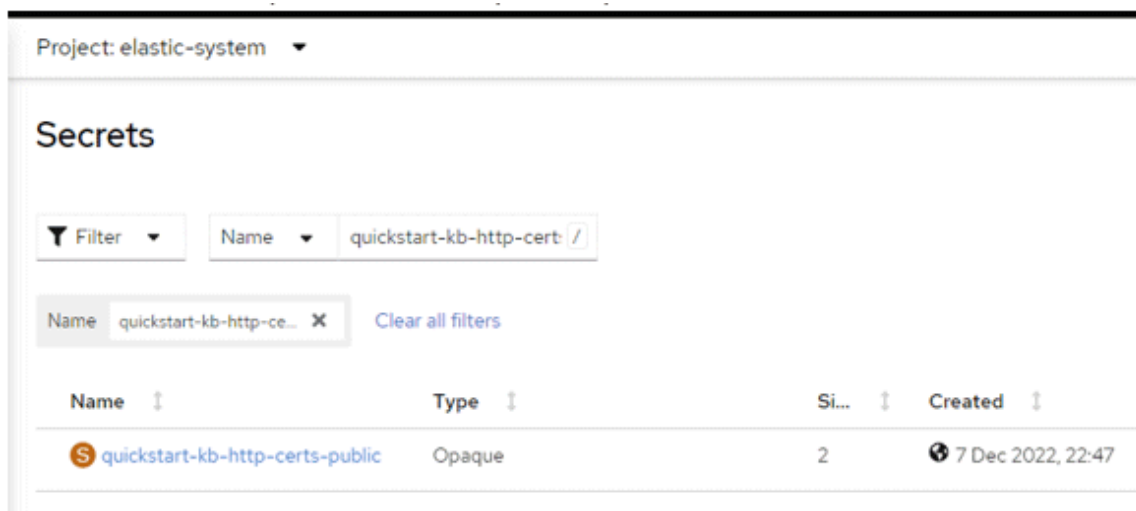
3. In YAML view, enter the following details and click "Create".

```
piVersion: kibana.k8s.elastic.co/v1
kind: Kibana
metadata:
  name: quickstart
spec:
  count: 1
  elasticsearchRef:
    name: quickstart
  enterpriseSearchRef:
    name: enterprise-search-quickstart
  version: 8.5.2
```

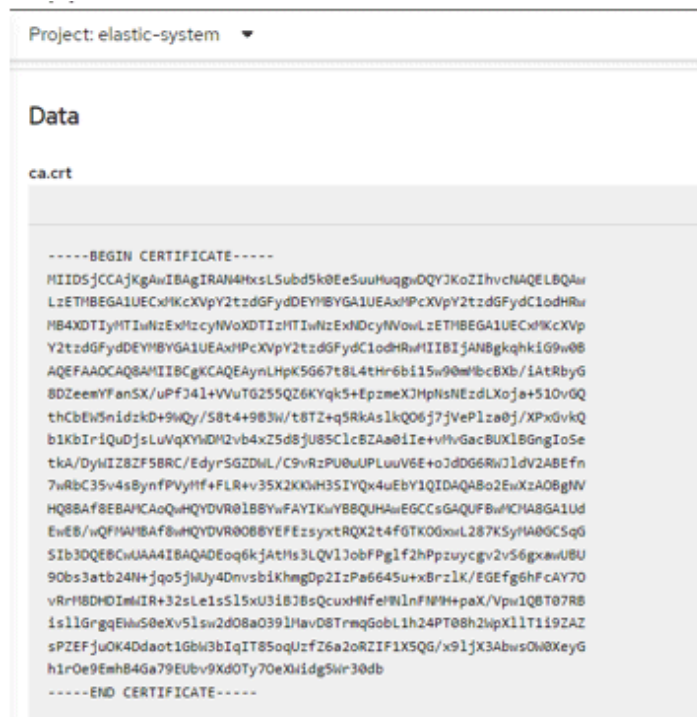
3.4.2.5 Expose Kibana using OpenShift Route

1. Obtain CA certificate that signs Kibana certificate.

Locate the secret quickstart-kb-http-certs-public.



Copy the content of ca.crt.



2. Create route for Internet access.

Navigate to Networking -> Route and select "Create Route".



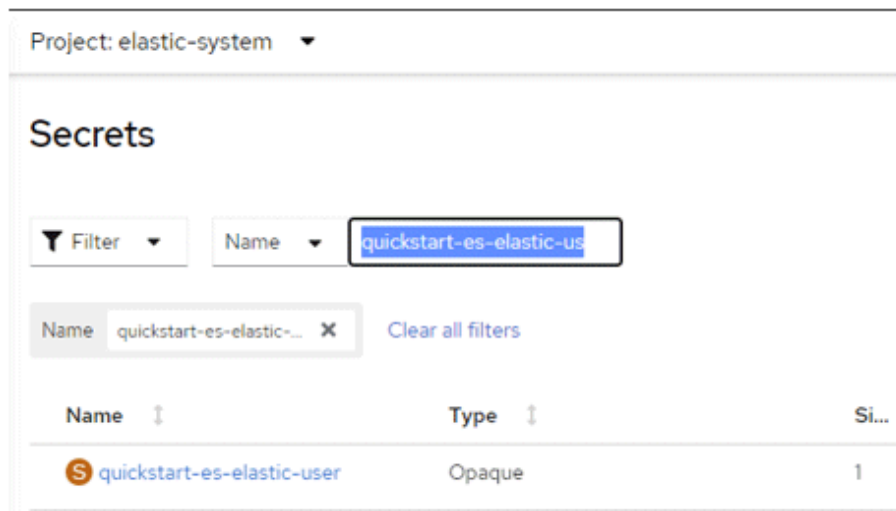
Fill in the details

Key	Value
Name	kibana
Hostname	Leave empty
Path	/
Service	quickstart-kb-http
Target port	5601 -> 5601 (TCP)
Secure Route	selected
TLS termination	Re-encrypt
Insecure traffic	Redirect
Destination CA certificate	Content of ca.crt in previous step

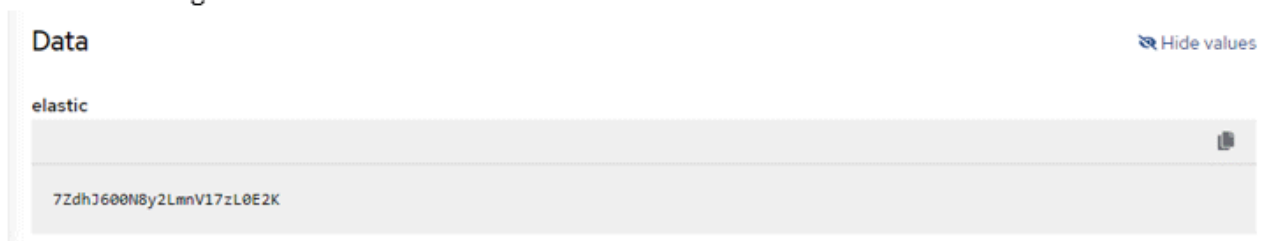
3.4.2.6 Login to Kibana

1. Obtain the Elasticsearch/Kibana login details

Locate the secret quickstart-es-elastic-user

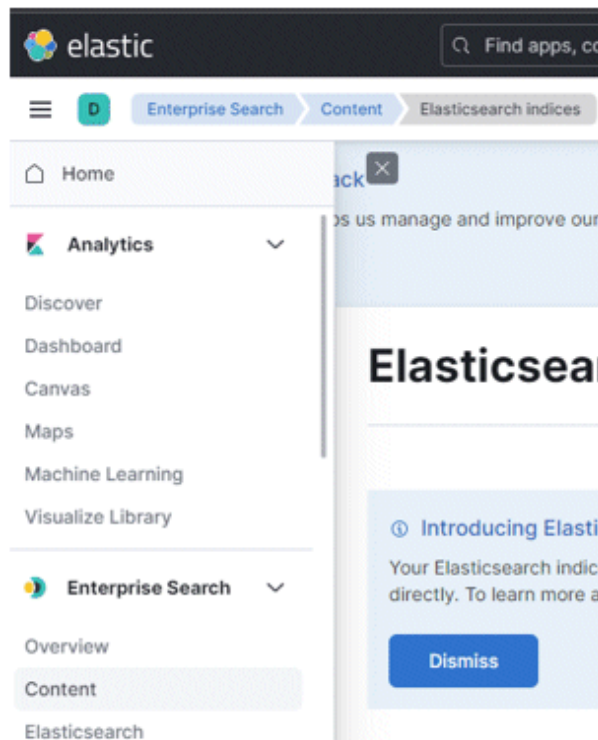


2. Observe the login details from the secret



3. Visit the URL as created in the Route above and use the above credential to login

4. Select the collapsed menu icon on top left corner and select Enterprise Search -> Content



5. If fluentd is forwarding logs to this elastic cluster, you will find the indexes here.

3.5 Implement Client

To use the FEP client, use the media or download the rpm module from the following site.

<https://www.postgresql.fastware.com/fujitsu-enterprise-postgres-client-download>

Chapter 4 Deployment Container

This chapter describes container deployment.

CR templates (sample files) for operating databases using Fujitsu Enterprise Postgres Operator are published in the following repository on GitHub. By using templates, you can easily deploy containers.

<https://github.com/fujitsu/fep-operator-examples>



Note

- Each volume of a Pod created by a FEPCluster deployment is sized by default for the following operations:

- Data size: 1 GB
- Daily update: about 50 MB

Refer to "[2.3.3 Configurable Volume per Cluster](#)" to design each volume size according to actual operation.

- If you are using anti-virus software, exclude all volumes that make up the FEPCluster from virus scanning.

4.1 Deploying FEPCluster using Operator

To deploy a FEPCluster in given namespace, follow these steps.

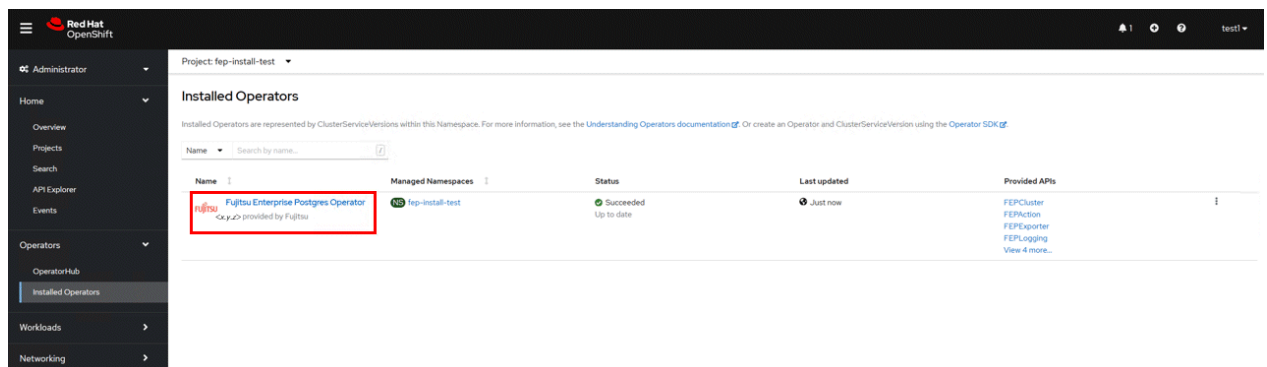
"<x.y.z>" in the screen example indicates the version level of the operator. Also, "<X>" indicates the product version of Fujitsu Enterprise Postgres.



Note

If you are deploying on a Kubernetes cluster, Refer to "Custom Resource Parameters" in the Reference to create and apply a yaml file.

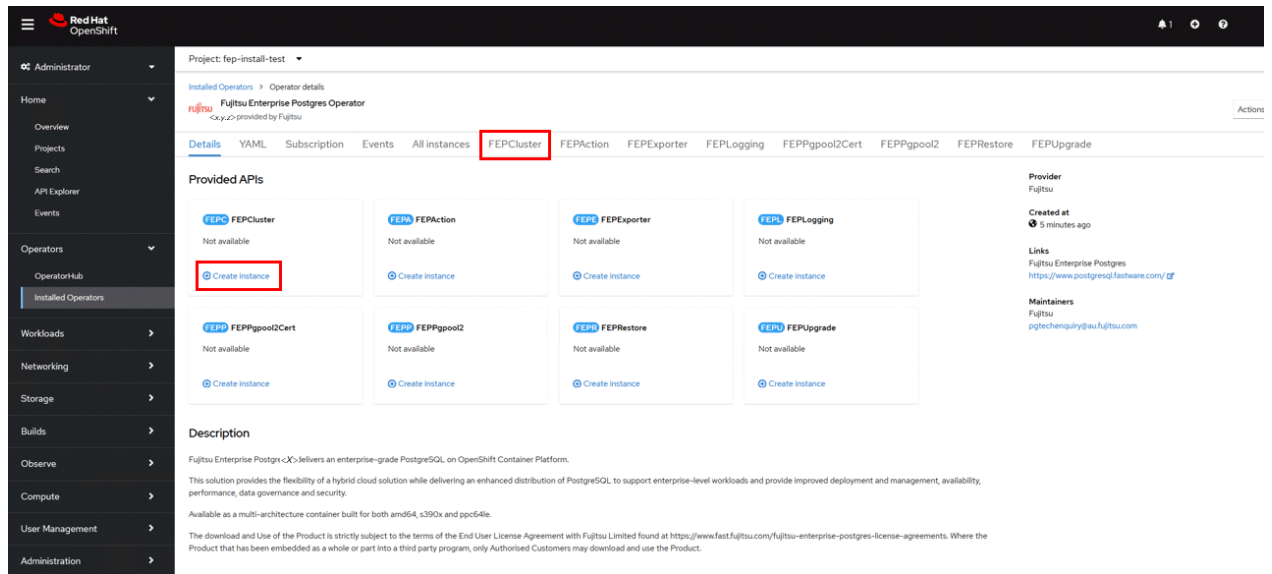
1. Under "Operators" menu item, click on "**Installed Operators**". You would see the installed FEP operator deployed in "[Chapter 3 Operator Installation](#)". Click on the name of operator.



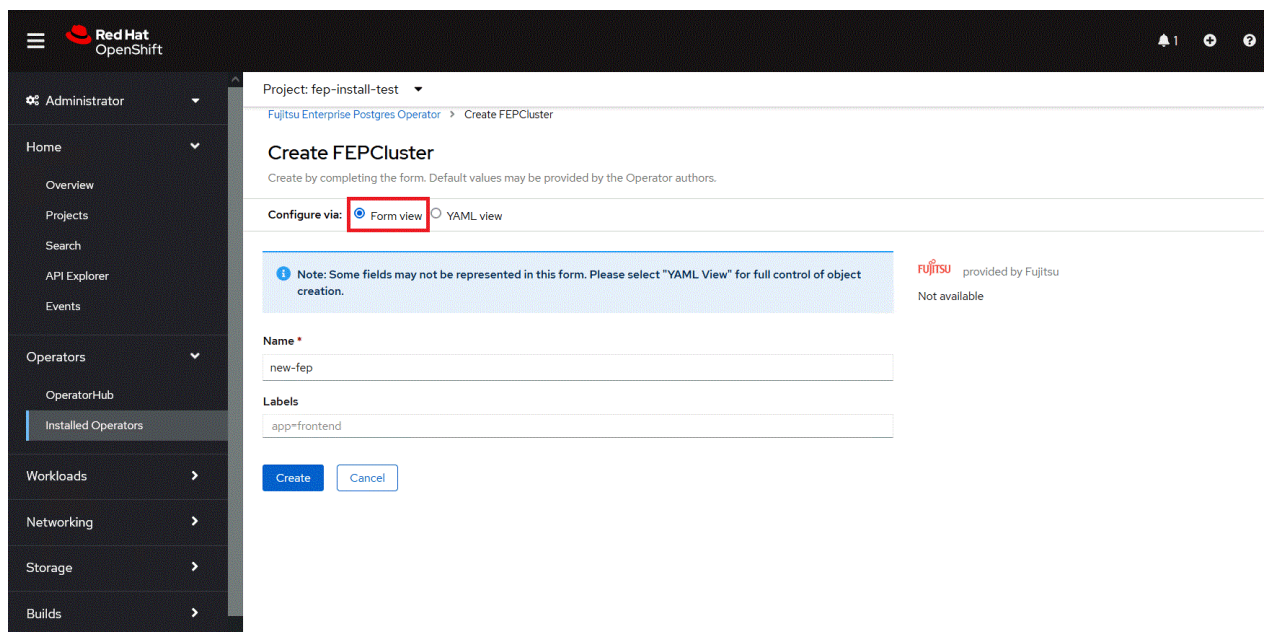
2. It will display a page with all CRs this operator supports. FEPCluster is the main CR and all others are child CR. We would create the main CR and all other CRs will be created automatically by Operator.
To create Cluster CR, either
(1) Click on "**Create Instance**" under FEPCluster.

OR

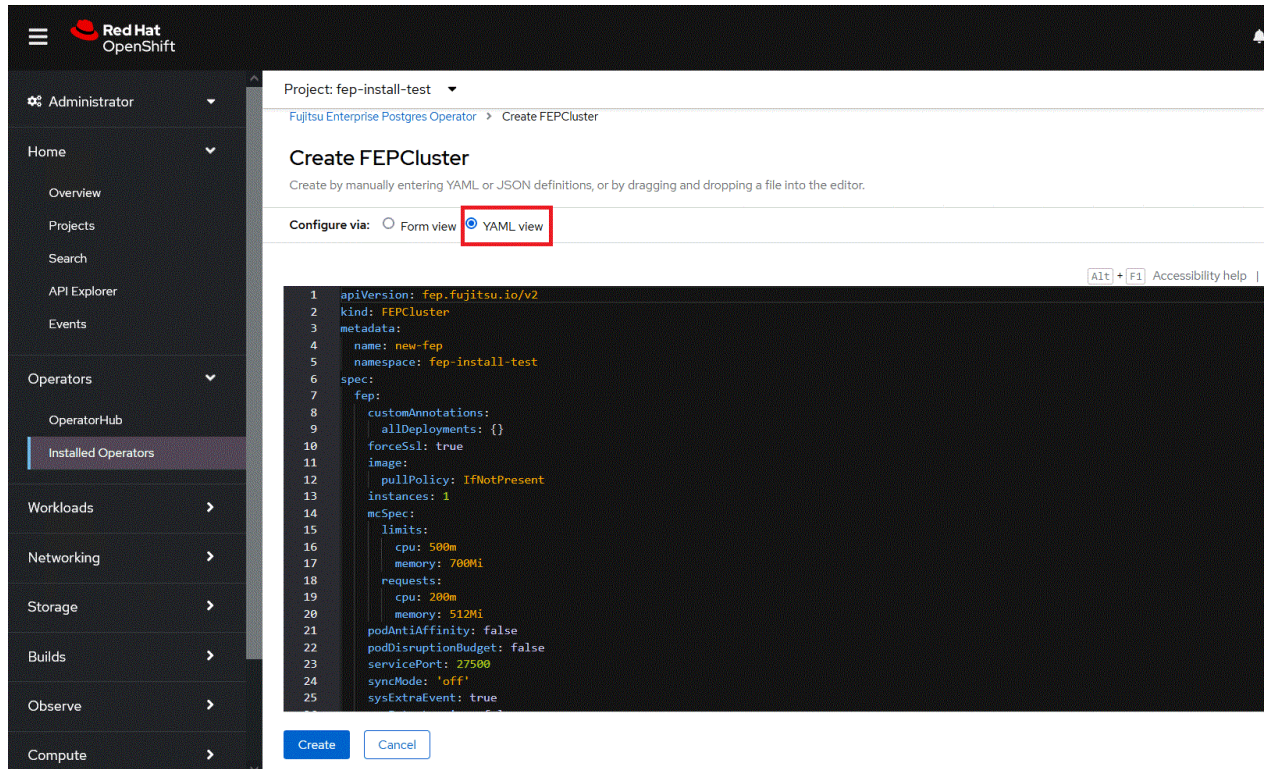
(2) Click on **"FEPCluster"** on top and then click on **"Create FEPCluster"** on the next page.



3. This will bring to "Create FEPCluster" page. Here you have two options to configure. The first one is Form View. At the moment, in Form View, one can change only the name of cluster being deployed. The default name is "new-fep". This name must be unique within a namespace.



4. In YAML View, starting value of CR is visible and one can choose to modify parameters before creating CR. Refer to the Reference for details of parameters.



The FEPCluster custom resource allows you to define the container's CPU, Memory, disk size, etc.

You can define each resource size individually, or you can use the following parameters to define the allocations for each resource in bulk.

Parameter	Description
spec.fep.databaseSize	Small, medium, and large define the following values for cpu/memory: small: 500m/700Mi medium: 2/4Gi large: 4/16Gi
spec.fepChildCrVal.storage.dataSize	Specifies the size of the data storage PV. Estimate and define the size of the backup storage area when the backup is enabled

We recommend that you use transparent data encryption to store your data.

Tablespace PVs are mounted in/database/tablespaces/tbpace1.

After building the database cluster, create tablespaces and tables as follows:

```
# Creating an Encrypted Tablespace
CREATE TABLESPACE secure_tablespace LOCATION '/database/tablespaces/tbpace1' WITH
(tablespace_encryption_algorithm = 'AES256');

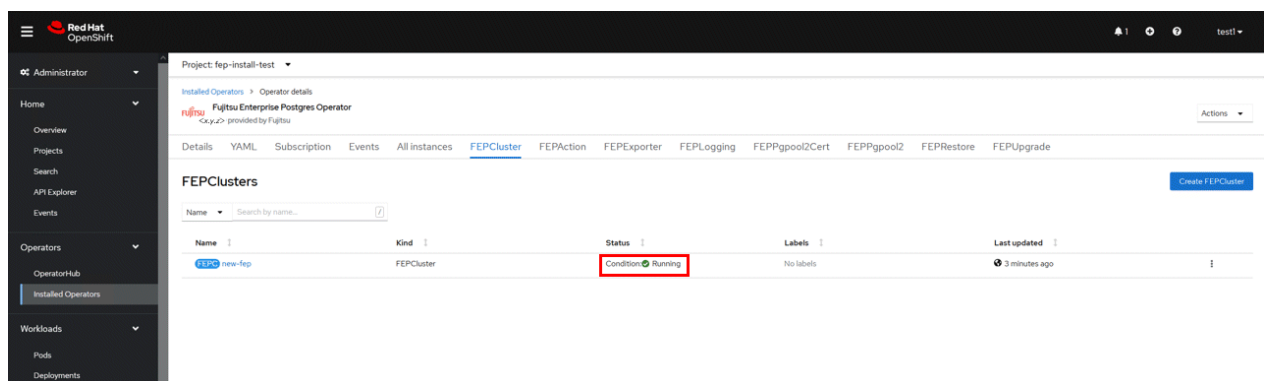
# Create Table in Encrypted Tablespace
CREATE TABLE secure_table (id int, pref text, city text, data text) TABLESPACE secure_tablespace;
```

Operator provides features such as backup, audit logging, and monitoring.

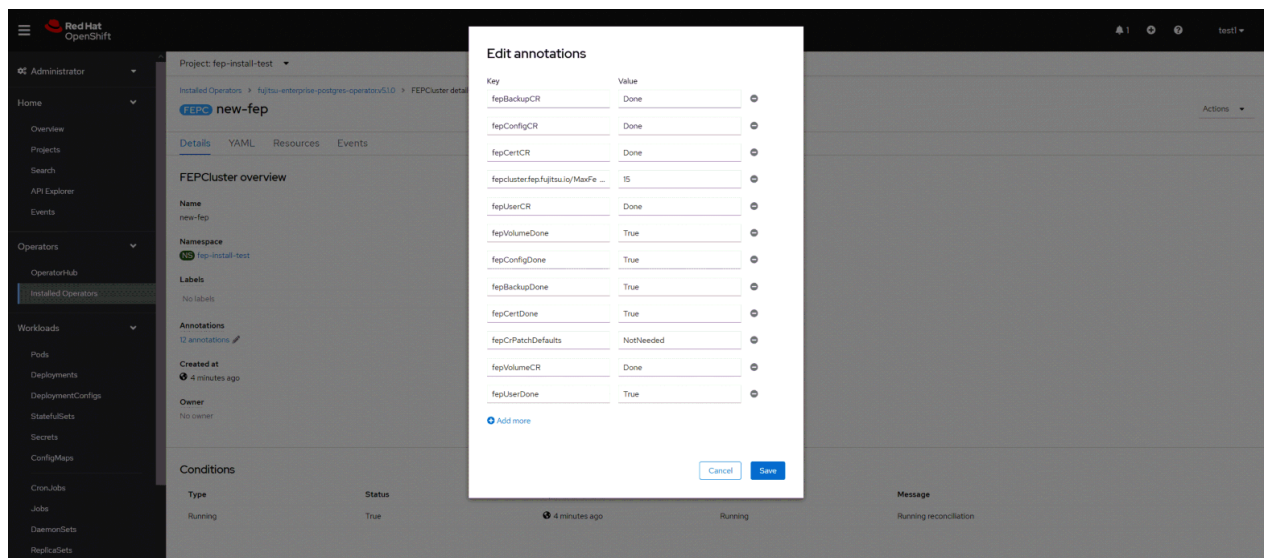
These features can be enabled with the following parameters:

Parameter	Description
spec.fep.monitoring.enable	When true, monitoring feature is enabled.
spec.fep.pgAuditLog.enable	When true, audit log collection is enabled.
spec.fepChildCrVal.backup.type	local enables backup. The backup data is stored in the PV.
spec.fepChildCrVal.autoscale.scaleout.policy	cpu_utilization enables the ability to autoscale out replicas when CPU utilization exceeds a threshold.

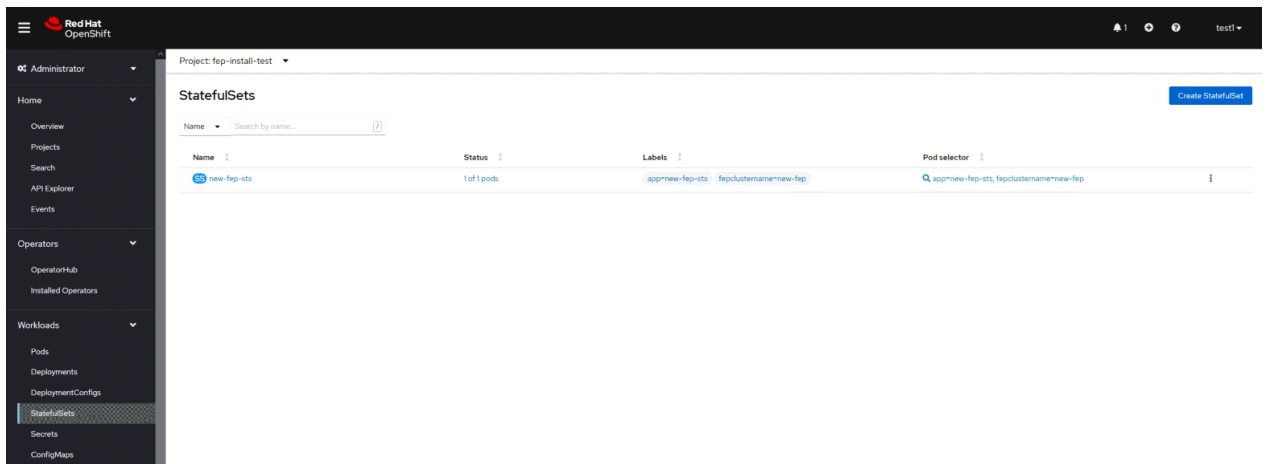
5. When "Create" is clicked on either of the two pages above, the operator creates FEPCluster CR, and there after one by one FEPClusterBackup, FEPClusterConfig, FEPClusterVolume, FEPClusterUser, and FEPClusterCert child CRs are created automatically. The starting values for child CRs are taken from the "fepChildCrVal" section of the FEPCluster CR YAML file. Modifying value in FEPCluster "fepChildCrVal" section. Operator reflects changes from FEPCluster parent CR to respective child CRs. Only allowable changes are reflected in child CRs. Child CRs are marked internal objects and hence will not be visible on the OCP console. However, you can check child CRs using command-line tools.



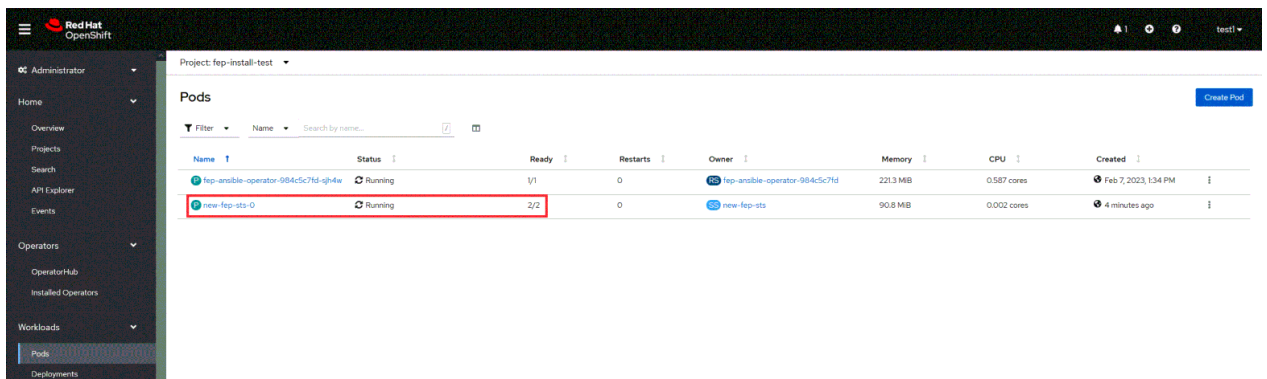
6. In FEPCluster CR, annotations are added to indicate that child CRs are created successfully and has initialised properly. It may take some time to complete.



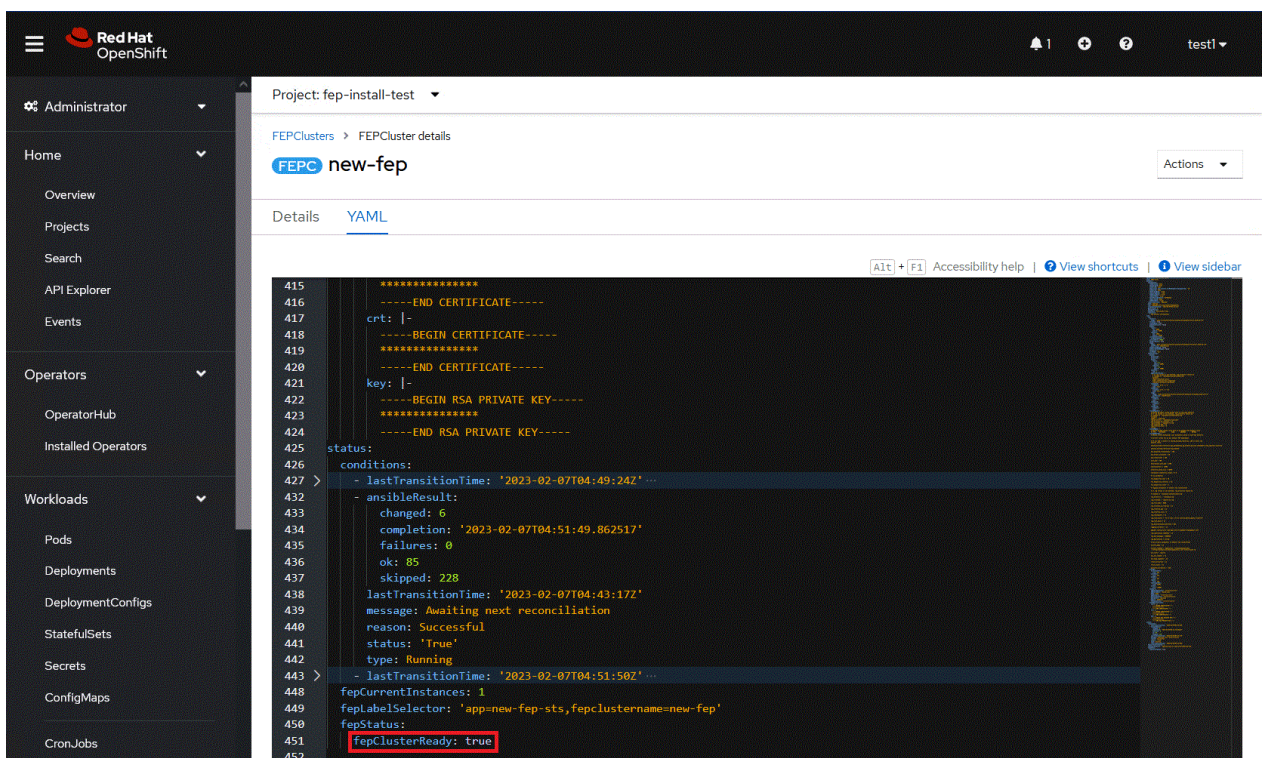
- Once child CRs are marked done in annotations, operator creates StatefulSet for the cluster.



- StatefulSet will start one FEP instance at one time and will wait it to be ready before starting next one.



- Once all instances of FEP servers are started, the operator marks a flag "fepClusterReady" under "status.fepStatus" section of CR to be **true**, indicating that FEPCluster is ready for use. Looking at YAML of FEPCluster CR, it would look like as below:



10. Operator also masks the sensitive fields like passwords, passphrase, certificates and keys in FEPCluster `fepChildCrVal` and also in respective child CRs.

4.2 Deploy a Highly Available FEPCluster

In a highly available FEP cluster, load balancing is possible by distributing read queries to replica instances.

In addition, if the master instance fails, the user can switch to the replica instance immediately to localize the business interruption period.

In a highly available configuration, you can select the synchronization mode for the replica instance. Synchronous replication is recommended for systems that cannot tolerate data loss in the event of a master instance failure.

Because multiple instances are created in a highly available configuration, licenses are required for each.

To deploy a highly available FEPCluster in given namespace, follow these steps:

[Prerequisites]

If the FEP cluster is running in HA mode, the backup and archive WAL volumes must be configured with shared storage (NFS, etc.) that supports ReadWriteMany. See the Openshift documentation for instructions on setting up shared storage. Also, the reference procedure is described in "[Appendix C Utilize Shared Storage](#)", so please check if necessary.

If you do not have shared storage, you can remove the backup section and the backup and archive volume sections to disable the backup feature and deploy the FEP cluster.



Note

If you are deploying on a Kubernetes cluster, Refer to "Custom Resource Parameters" in the Reference to create and apply a yaml file.

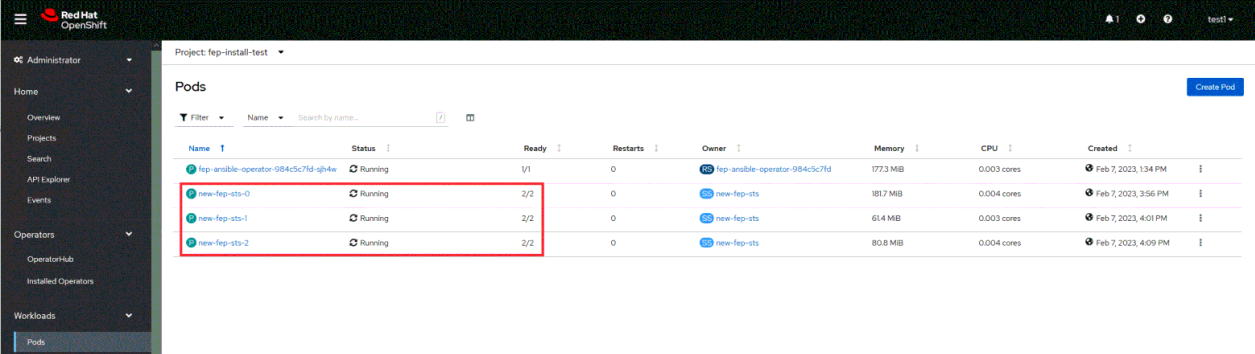
1. It is the same as the procedure from step 1 to step 3 in "[4.1 Deploying FEPCluster using Operator](#)".
2. Instead of step 4 in "[4.1 Deploying FEPCluster using Operator](#)", change to the YAML view and specify '3' for the "instances" parameter of "fep" in "spec". Specify the storage class for the prepared shared storage for the backup and archive WAL volumes.

The screenshot shows the Red Hat OpenShift console interface. On the left is a sidebar with navigation menus: Administrator, Home (Overview, Projects, Search, API Explorer, Events), Operators (OperatorHub, Installed Operators), and Workloads (Pods, Deployments, DeploymentConfigs, StatefulSets, Secrets). The main panel displays the 'Create FEPCluster' form for the 'fep-install-test' project. The form is in 'YAML view' mode. The YAML configuration is as follows:

```
1 kind: FEPCluster
2 apiVersion: fep.fujitsu.io/v2
3 metadata:
4   name: new-fep
5   namespace: fep-install-test
6 spec:
7   fep:
8     customAnnotations:
9       allDeployments: {}
10    forceSsl: true
11    image:
12    pullPolicy: IfNotPresent
13    instances: 3
14    mcSpec:
15      limits:
16        cpu: 500m
17        memory: 768Mi
18      requests:
19        cpu: 200m
20        memory: 512Mi
21    podAntiAffinity: false
22    podDisruptionBudget: false
23    servicePort: 27500
```

3. It is the same as the procedure from step 5 to step 10 in "[4.1 Deploying FEPCluster using Operator](#)".

- Three pods deployed and ready for a highly available FEPCluster.



Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
fep-ansible-operator-984c5c7fd-gh4w	Running	1/1	0	fep-ansible-operator-984c5c7fd	1773 MB	0.003 cores	Feb 7, 2023, 1:34 PM
new-fep-sts-0	Running	2/2	0	new-fep-sts	1817 MB	0.004 cores	Feb 7, 2023, 3:56 PM
new-fep-sts-1	Running	2/2	0	new-fep-sts	614 MB	0.003 cores	Feb 7, 2023, 4:01 PM
new-fep-sts-2	Running	2/2	0	new-fep-sts	808 MB	0.004 cores	Feb 7, 2023, 4:09 PM

Information

You can determine whether the master or replica pod is the master or replica pod by issuing the following command:

```
$ oc get pod -L feprole
```

NAME	READY	STATUS	RESTARTS	AGE	FEPROLE
fep-ansible-operator-88f7fb4b-5jh85	1/1	Running	0	24m	
new-fep-sts-0	2/2	Running	0	17m	master
new-fep-sts-1	2/2	Running	0	15m	replica
new-fep-sts-2	2/2	Running	0	13m	replica

4.3 Deploying FEPExporter

To deploy a FEPExporter, follow these steps.

"<x.y.z>" in the screen example indicates the version level of the operator.

Note

If you are deploying on a Kubernetes cluster, Refer to "Custom Resource Parameters" in the Reference to create and apply a yaml file.

- Administrator

Home

Overview

Projects

Search

API Explorer

Events

Operators

OperatorHub

Installed Operators

Workloads

Pods

Deployments

DeploymentConfigs

StatefulSets

Secrets

ConfigMaps

Project: fep-install-test

Fujitsu Enterprise Postgres Operator > Create FEPCluster

Create FEPCluster

Create by completing the form. Default values may be provided by the Operator authors.

Configure via: ☐ Form view ☒ YAML view

1 kind: FEPCluster

2 apiVersion: fep.fujitsu.io/v2

3 metadata:

4 name: new-fep

5 namespace: fep-install-test

6 spec:

7 fep:

8 customAnnotations:

9 allDeployments: {}

10 forceSsl: true

11 monitoring:

12 enable: true

13 image:

14 pullPolicy: IfNotPresent

15 instances: 1

16 mcSpec:

17 limits:

18 cpu: 500m

19 memory: 700Mi

20 requests:

21 cpu: 200m

22 memory: 512Mi

23 podAntiAffinity: false

24 podDisruptionBudget: false

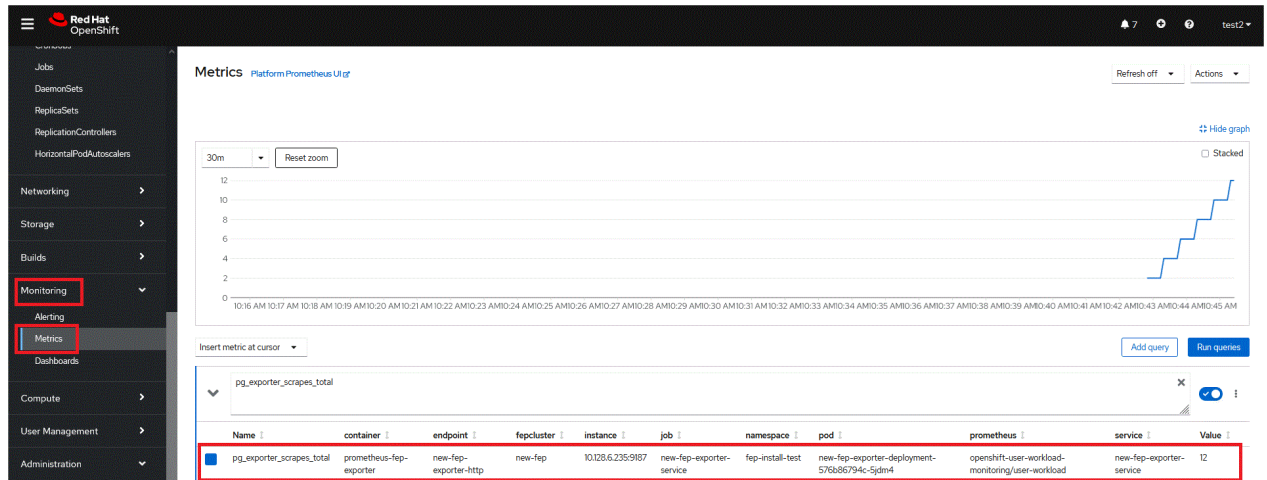
25 servicePort: 27500

26 syncMode: 'off'

- 
- The screenshot shows the Red Hat OpenShift console interface. On the left is a dark sidebar with navigation links: Administrator, Home, Overview, Projects, Search, API Explorer, Events, Operators, OperatorHub, Installed Operators (highlighted), Workloads, Pods, and Deployments. The main content area is titled 'Project: fep-install-test'. Below this, it says 'Installed Operators' and 'Operator details'. The 'Fujitsu Enterprise Postgres Operator' is listed, with a note 'Operator provided by Fujitsu'. A tab bar shows various options: Details, YAML, Subscription, Events, All instances, FEPCluster, FEPAAction, FEPExporter (selected), FEPLogging, FEPPgpool2Cert, FEPPgpool2, FEPRestore, and FEPUUpgrade. Below the tabs, there's a section for 'FEPExporters' with a search bar. A table lists the installed operators. The first entry, 'new-fep-exporter', is highlighted with a red box. The table has columns for Name, Kind, Status, Labels, and Last updated. The 'new-fep-exporter' row shows 'FEPExporter' as the kind, 'Running' as the status with a green checkmark, 'No labels' as labels, and 'Just now' as the last updated time.
- | Name | Kind | Status | Labels | Last updated |
|------------------|-------------|---------|-----------|--------------|
| new-fep-exporter | FEPExporter | Running | No labels | Just now |

- 58 -

- Now user can see scrape FEPEXporter specific metrics on Openshift Platform in monitoring section area using PROMQL to specify a metrics of interest



Note

- User can set `fep.monitoring.enable` to true or false on an already instantiated cluster as well to achieve desired results
- `pgMetricsUser` can be defined later on a running FEPCluster with monitoring enabled and can force FEPEXporter to use `pgMetricsUser` by mere restarting it (refer `restartRequired`). However, MTLS can not be configured in this case and user is expected to grant specific permission to `pgMetricsUser` for all the database objects which are expected to be use while scraping information.
- For MTLS to be forced, ensure `usePodName` and `pg_hba.conf` is been set appropriately.
- FEPEXporter default metrics expects few following in `postgresql.conf`
 - `pg_stats_statements` library pre-loaded
 - `track_activities` and `track_counts` are turned on
 - Monitoring user needs permission on `pg_stat_*` views
- FEPEXporter pod specification related to CPU memory can be changed. After changing resources specification, set `restartRequired` flag to true. FEPEXporter will be restarted with new specifications
- FEP Monitoring is closely integrated with Prometheus available on platform. User should ensure that on openshift platform monitoring is enabled for user-defined projects (Refer: <https://docs.openshift.com/container-platform/4.11/monitoring/enabling-monitoring-for-user-defined-projects.html>). For platforms other than openshift, ensure Prometheus is installed before deployment of FEP operator

4.4 FEPEXporter in Standalone Mode

FEPEXporter is an independent CR; hence it does not necessarily depend on main FEPCluster CR. To deploy a FEPEXporter in given namespace follow the below step.

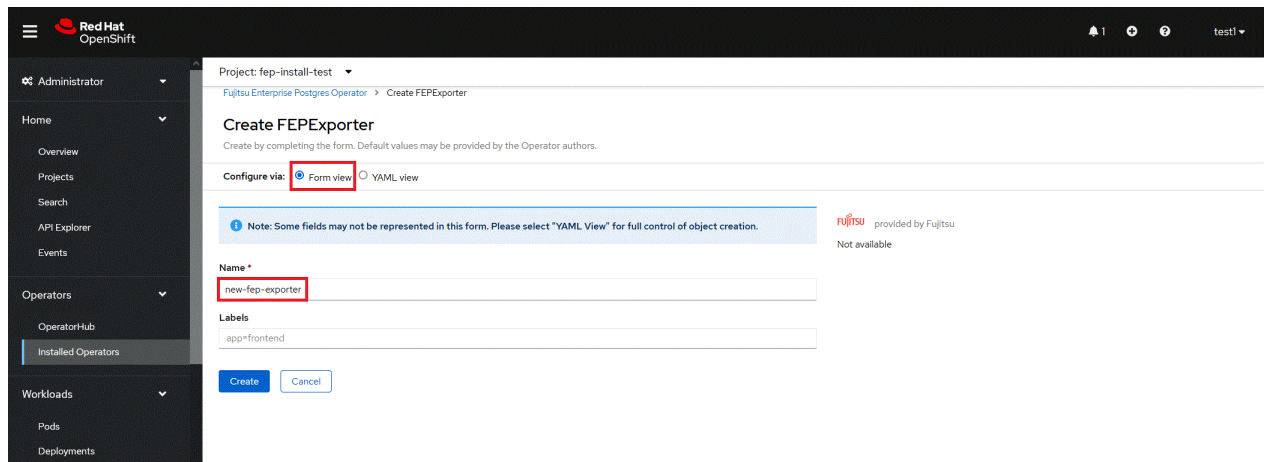


Note

If you are deploying on a Kubernetes cluster, Refer to "Custom Resource Parameters" in the Reference to create and apply a yaml file.

- To create FEPEXporter CR, either
 - Click on **"Create Instance"** under FEPEXporter.
 - OR
 - Click on **"FEPEXporter"** on top and then click on **"Create FEPEXporter"** on the next page.

2. In Form View, one can change only the name of cluster being deployed. The default name is "new-fep-exporter". This name must be unique within a namespace.
3. FEPEXporter scrapes metrics for FEPCluster within same namespace.



Project: fep-install-test

Fujitsu Enterprise Postgres Operator > Create FEPEXporter

Create FEPEXporter

Create by completing the form. Default values may be provided by the Operator authors.

Configure via: ☒ Form view ☐ YAML view

Note: Some fields may not be represented in this form. Please select "YAML View" for full control of object creation.

Name *

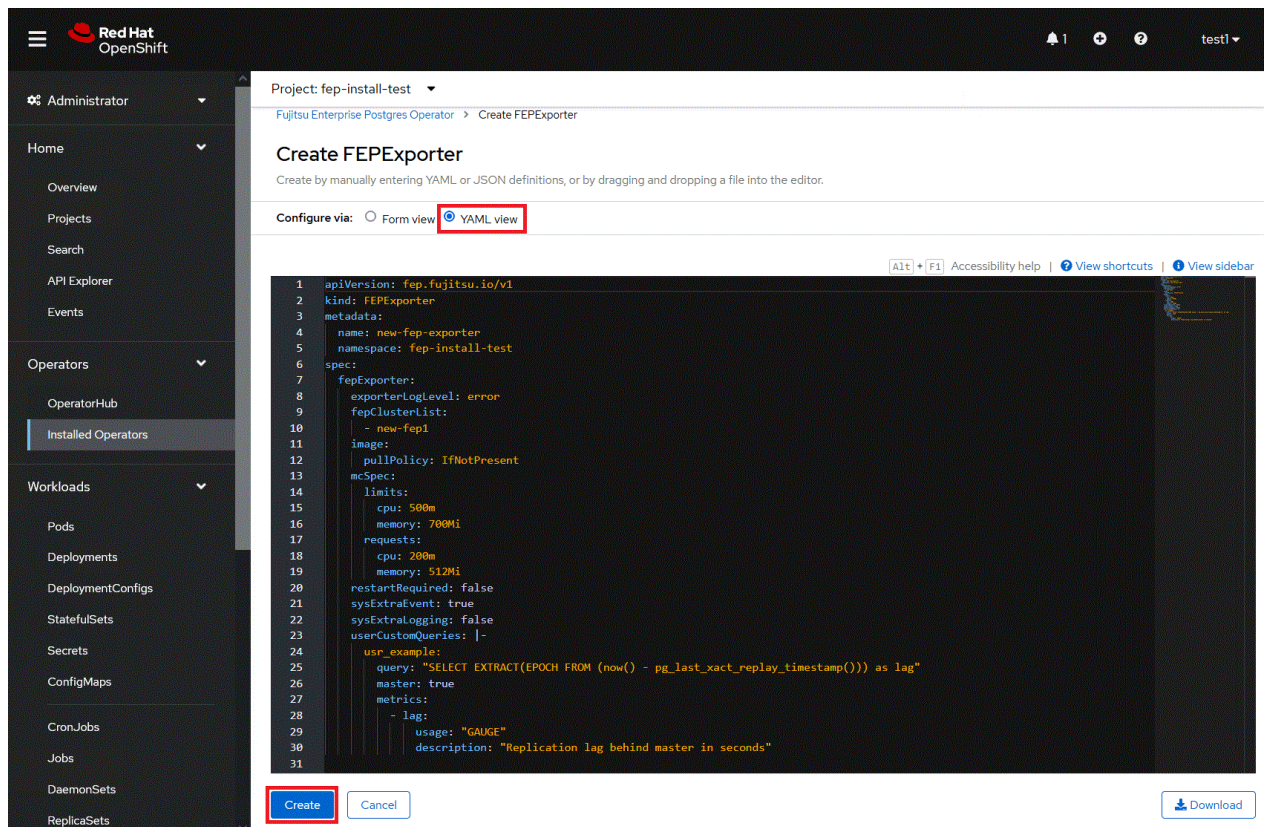
new-fep-exporter

Labels

app=frontend

[Create](#) [Cancel](#)

4. In YAML View, starting value of FEPEXporter CR is visible and one can choose to modify parameters before creating CR. Refer to the Reference for details of parameters.



Project: fep-install-test

Fujitsu Enterprise Postgres Operator > Create FEPEXporter

Create FEPEXporter

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

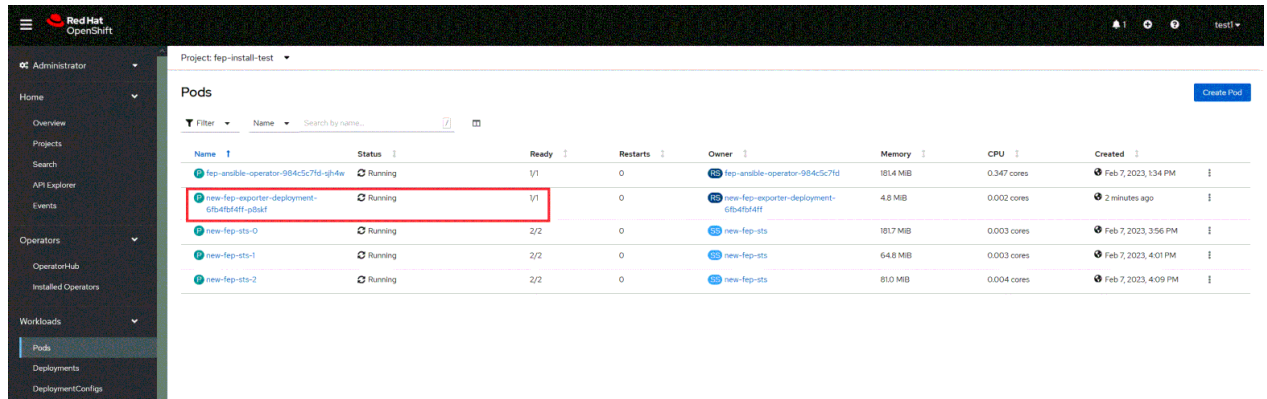
Configure via: ☐ Form view ☒ **YAML view**

```

1  apiVersion: fep.fujitsu.io/v1
2  kind: FEPEXporter
3  metadata:
4    name: new-fep-exporter
5    namespace: fep-install-test
6  spec:
7    fepExporter:
8      exporterLogLevel: error
9      fepClusterList:
10       - new-fep1
11    image:
12      pullPolicy: IfNotPresent
13    mcSpec:
14      limits:
15        cpu: 500m
16        memory: 700Mi
17      requests:
18        cpu: 200m
19        memory: 512Mi
20    restartRequired: false
21    sysExtraEvent: true
22    sysExtraLogging: false
23    userCustomQueries: |-
24      usr_example:
25        query: "SELECT EXTRACT(EPOCH FROM (now() - pg_last_xact_replay_timestamp())) as lag"
26        master: true
27        metrics:
28          - lag:
29            usage: "GAUGE"
30            description: "Replication lag behind master in seconds"
31  
```

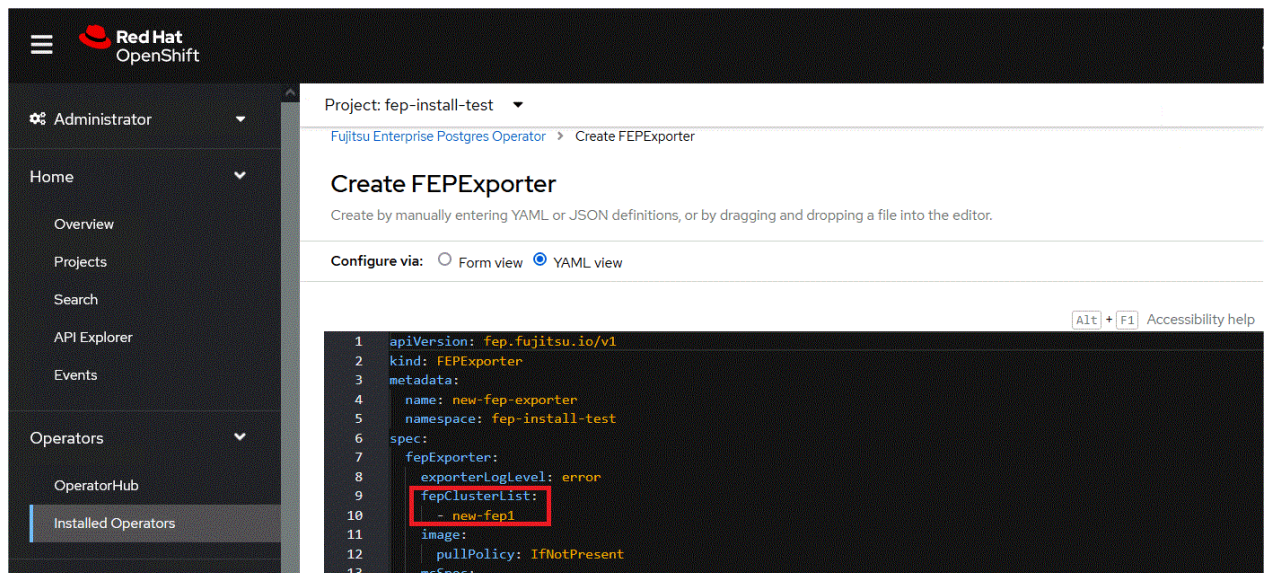
[Create](#) [Cancel](#) [Download](#)

- When clicked on the "Create" button. It will create FEPEXporter pod with other resource like secret, service, configmap for data source queries.



Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
fep-ansible-operator-984c5c7fd-qh4w	Running	1/1	0	fep-ansible-operator-984c5c7fd	1814 MB	0.347 cores	Feb 7, 2023, 1:34 PM
new-fep-exporter-deployment-6fb4fb4ff-gbaf	Running	1/1	0	new-fep-exporter-deployment-6fb4fb4ff	4.8 MB	0.002 cores	2 minutes ago
new-fep-sts-0	Running	2/2	0	new-fep-sts	1817 MB	0.003 cores	Feb 7, 2023, 3:56 PM
new-fep-sts-1	Running	2/2	0	new-fep-sts	64.8 MB	0.003 cores	Feb 7, 2023, 4:01 PM
new-fep-sts-2	Running	2/2	0	new-fep-sts	81.0 MB	0.004 cores	Feb 7, 2023, 4:09 PM

- Specify the name of the FEPCluster in spec.fepExporter.fepClusterList of FEPEXporter. Before targeting cluster, Check the FEPCluster status and FEP StatefulSet are in running condition.



Project: fep-install-test

Fujitsu Enterprise Postgres Operator > Create FEPEXporter

Create FEPEXporter

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

Configure via: ☐ Form view ☒ YAML view

```

1 apiVersion: fep.fujitsu.io/v1
2 kind: FEPEXporter
3 metadata:
4   name: new-fep-exporter
5   namespace: fep-install-test
6 spec:
7   fepExporter:
8     exporterLogLevel: error
9     fepClusterList:
10      - new-fep1
11   image:
12     pullPolicy: IfNotPresent
13   mcSpec:

```

- It will recreate FEPEXporter pod with a new dataresource secret. It will list down all the database with statistics of specified FEPCluster in monitoring section.
- If fepClusterList has more than one clusters listed, current exporter will collect metrics for all of those listed.
- Multiple FEPEXporters can be deployed within one namespace with their own cluster list to collect metrics from.

4.5 Deploying FEPClusters with Cloud-based Secret Management



The cloud-based secret management feature cannot be used together with the following parameters.

- spec.fepChildCrVal.sysUsers.pgSecurityUser
- spec.fepChildCrVal.sysTde.tdek
- spec.fepChildCrVal.sysUsers.passwordValid.days

4.5.1 Installing Secret Store CSI Driver Using Helm Charts

Install Secret Store CSI Driver from Helm chart.

Add helm chart repository.

```
helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
```

Install with helm command.

```
helm install csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver --namespace kube-system --set enableSecretRotation=true --set rotationPollInterval=30s
```



Information

- Setting enableSecretRotation=true enables auto rotation of secret. i.e if value of secret gets changed in one of the external secret store (Azure/AWS/GCP/HashiCorp vault) then the updated value will be reflected in the FEPCluster as well.
- Setting rotationPollInterval=30s enables rotation poll interval which checks how frequently the mounted secrets for all pods need to be resynced to the latest.
- For OpenShift cluster to allow CSI type volumes to be mounted in container,system Security Context Constraints needs to be patched. Patch the volumes section to include CSI for providers(nonroot,anyuid,hostmount-anyuid,machine-api-termination-handler,hostaccess,node-exporter,privileged,privileged-genevallogging,restricted).
- In scarios where existing OpenShift is upgraded kindly verify that CSI is included in system Security Context Constraints for the above mentioned providers.

4.5.2 Installing and Configuring Azure Provider for Secret Store CSI Driver

4.5.2.1 Install Azure Provider drivers using helm chart

```
helm repo add csi-secrets-store-provider-azure https://azure.github.io/secrets-store-csi-driver-provider-azure/charts
```

Note: By default when installing Azure Provider ; secret-store-csi-driver installation is set to true by default. If secret-store-csi-driver is already installed as per steps in "[4.5.1 Installing Secret Store CSI Driver Using Helm Charts](#)" execute below command.

```
helm install csi csi-secrets-store-provider-azure/csi-secrets-store-provider-azure --namespace kube-system -set secrets-store-csi-driver.install=false
```

Note: If secret-store-csi-driver is not installed as per step "[4.5.1 Installing Secret Store CSI Driver Using Helm Charts](#)". Execute below command to install azure provider along with secret-store-csi-driver.

```
helm install csi csi-secrets-store-provider-azure/csi-secrets-store-provider-azure --namespace kube-
system --set secrets-store-csi-driver.enableSecretRotation=true --set secrets-store-csi-
driver.rotationPollInterval=30s
```

4.5.2.2 Create Secret to Access Azure Key vault

```
kind: Secret
apiVersion: v1
metadata:
  name: <Secret Name>
  namespace: <WHERE FEP CLUSTER TO BE INSTALLED>
  labels:
    secrets-store.csi.k8s.io/used: 'true'
data:
  clientid: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```


mycert.pem

 **Note**

4.5.3 Installing and Configuring AWS Provider for Secret Store CSI Driver

4.5.3.1 Install AWS Provider drivers using helm chart

- 64 -

```
helm install -n kube-system secrets-provider-aws aws-secrets-manager/secrets-store-csi-driver-provider-aws -namespace kube-system
```

4.5.3.2 Setup EKS cluster along with service account with necessary IAM roles and permission to access Secret Manager

Follow below link to setup IAM roles and EKS for CSI.

<https://github.com/aws/secrets-store-csi-driver-provider-aws>

Create IAM role trust policy to access Secret Manager

```
Create IAM role trust policy to access Secret Manager
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789:oidc-provider/oidc.eks.ap-southeast-3.amazonaws.com/id/ABCD1234567"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.ap-southeast-3.amazonaws.com /id/ ABCD1234567:sub": "system:serviceaccount:myns:mysa",
          "oidc.eks.ap-southeast-3.amazonaws.com /id/ ABCD1234567:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

4.5.3.3 Store Secret in AWS Secrets Manager

```
aws secretsmanager create-secret --name <Secret Name> --secret-string <Secret Value>
```

4.5.3.4 Store Cert in AWS Secrets Manager

Certificate should be in below format before uploading cert to AWS Secrets Manager i.e it should be one .pem file (key, crt and CA in one file)

(Refer "[mycert.pem](#)" for sample certificate format)

```
aws secretsmanager create-secret --name <Secret Name> --secret-binary fileb://<File Name>
```



Note

Only single key value for secret to be stored in Secret Manager.

4.5.4 Installing GCP Provider for Secret Store CSI Driver

4.5.4.1 Install GCP Provider drivers using Kubernetes

```
wget https://raw.githubusercontent.com/GoogleCloudPlatform/secrets-store-csi-driver-provider-gcp/main/deploy/provider-gcp-plugin.yaml
```

```
kubectl apply -f provider-gcp-plugin.yaml -namespace kube-system
```

4.5.4.2 Configure GCP secret manager and IAM

Create Service Account:

```
gcloud iam service-accounts create my-secret-acc;
```

Attach SecretManagerAdmin policy to the new service account

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member="serviceAccount: my-secret-acc @$PROJECT_ID.iam.gserviceaccount.com" \
--role="roles/secretmanager.admin" \
--condition="None";
```

Generate a key for your new service account

```
gcloud iam service-accounts keys create iam-key.json \
--iam-account=" my-secret-acc @$PROJECT_ID.iam.gserviceaccount.com";
```

4.5.4.3 Create Secret to access GCP Secret manager

Use keys generated from "[4.5.4.2 Configure GCP secret manager and IAM](#)" (iam-key.json file)

```
kubectl create secret generic <secret-name> --from-file=<iam-key.json>
```

4.5.4.4 Store secret in GCP Secret manager

```
gcloud secrets create <secret name> --data-file="/path/to/file"
```

4.5.4.5 Store Cert in GCP Secret manager

Certificate should be in below format before uploading cert to GCP Secret Manager i.e it should be one .pem file (key, crt and CA in one file)

(Refer "[mycert.pem](#)" for sample certificate format)

```
gcloud secrets create <secret name> --data-file="/path/to/file"
```



Note

Only single key value for secret to be stored in Secret Manager.

4.5.5 Installing HashiCorp Vault Provider for Secret Store CSI Driver

4.5.5.1 Install HashiCorp Provider drivers using helm chart

```
helm repo add hashicorp https://helm.releases.hashicorp.com
```

```
helm install vault hashicorp/vault --set "server.enabled=false" --set "injector.enabled=false" --set "csi.enabled=true"
```

4.5.5.2 Configure Kubernetes Authentication for HashiCorp Vault

```
vault auth enable kubernetes
```

```
vault write auth/kubernetes/config \
token_reviewer_jwt="$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" \
kubernetes_host="https://$KUBERNETES_PORT_443_TCP_ADDR:443" \
kubernetes_ca_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
```

4.5.5.3 Store Secret in HashiCorp Vault

```
vault enable secret
```

```
vault kv put secret/<path> <secret name>=<secret value>
```

4.5.5.4 Store Cert in HashiCorp Vault

Certificate should be in below format before uploading cert to HashiCorp Vault i.e it should be one .pem file (key, crt and CA in one file)
(Refer "[mycert.pem](#)" for sample certificate format)

```
Vault kv put secret/<path> <secret name>=@<path to cert.pem>
```

4.5.5.5 Create policy and role to access the secrets from HashiCorp Vault

Policy:

```
vault policy write <policy name> - <<EOF
path "secret/database/credentials" {
capabilities = ["read", "write", "update", "delete"]
}
EOF
```

Role:

```
vault write auth/kubernetes/role/<role name> \
bound_service_account_names=* \
bound_service_account_namespaces=* \
policies=<policy name> \
ttl=24h
```

Note: access can be restricted by assigning <fed-cluster>-sa service account to bound_service_account_names and also can be namespace restricted by assigning value to bound_service_account_namespaces



Only single key value for secret to be stored in HashiCorp vault.

4.5.6 Configuring FEPCluster to use Provider for Secret Store Driver

To enable use of Secret Store CSI driver, a new parameter "secretStore" under spec.fepChildCrVal section in the FEPCluster CR. Under secretStore.csi user should define the details to connect to external Seret store(Azure,AWS,GCP and HashiCorp Vault) and the list of secrets in that secret store. The definition of spec.fepChildCrVal.secretStore parameter will differ depending on the type of provider that is used.

4.5.6.1 Azure Provider for Secret Store CSI Driver

```
spec:
  ...
  fepChildCrVal:
```

```

secretStore:
  method: csi
  csi:
    providerName: azure
    azureProvider:
      keyvaultname:
      tenantid:
      credentials:
      fepSecrets:
        - pgadminpassword: pgadminpassword
        - tdepassphrase: passphrase
        - systemCertificates: systemCerts
        - pguser: pgusername
        - pgpassword: pgpwd
        - pgdb: pgdbsecret
        - pgrepluser: pgrepluser
        - pgreplpassword: pgreplpassword
        - pgRewinduser: pgRewinduser
        - pgRewindpassword: pgRewindpassword
        - pgMetricsUser: metricsuser
        - pgMetricsPassword: metricspwd
        - patronitls: patronicrt
        - patronitlscacrt: patronica
        - postgresqls: postgrescrt
        - postgresqlscacrt: postgresca
        - pgAdminTls: admincrt
        - pgAdminTlscacrt: adminca
        - pgAdminTls_privateKeyPassword: adminpvtkey
        - pgRewindUserTls: rewindcrt
        - pgRewindUserTlscacrt: rewindca
        - pgRewindUserTls_privateKeyPassword: rwndpvtkey
        - pgrepluserTls: replcrt
        - pgrepluserTlscacrt: replca
        - pgrepluserTls_privateKeyPassword: replpvtkey
        - pgMetricsUserTls: metricscrt
        - pgMetricsUserTlscacrt: metricsca
        - pgMetricsUserTls_privateKeyPassword: adminpvtkey
      fepCustomCerts:
        - userName: user1
          userCrt: user1crt
          userCa: user1ca
        - userName: mydbuser
          userCrt: mydbusercrt
          userCa: mydbuserca

```

Note: The parameters which are in black in fepSecrets are mandatory.

4.5.6.2 AWS Provider for Secret Store CSI Driver

```

spec:
  ...
  fepChildCrVal:
    secretStore:
      method: csi
      csi:
        providerName: aws
        awsProvider:
          region:
          roleName:
          fepSecrets:
            - pgadminpassword: pgadminpassword
            - tdepassphrase: passphrase
            - systemCertificates: systemCerts

```

```

- pguser: pgusername
- pgpassword: pgpwd
- pgdb: pgdbsecret
- pgrepluser: pgrepluser
- pgreplpassword: pgreplpassword
- pgRewinduser: pgRewinduser
- pgRewindpassword: pgRewindpassword
- pgMetricsUser: metricsuser
- pgMetricsPassword: metricspwd
- patronitls: patronicrt
- patronitlscacrt: patronica
- postgrestls: postgrescrt
- postgrestlscacrt: postgresca
- pgAdminTls: admincrt
- pgAdminTlscacrt: adminca
- pgAdminTls_privateKeyPassword: adminpvtkey
- pgRewindUserTls: rewindcrt
- pgRewindUserTlscacrt: rewindca
- pgRewindUserTls_privateKeyPassword: rwndpvtkey
- pgrepluserTls: replcrt
- pgrepluserTlscacrt: replca
- pgrepluserTls_privateKeyPassword: replpvtkey
- pgMetricsUserTls: metricscrt
- pgMetricsUserTlscacrt: metricsca
- pgMetricsUserTls_privateKeyPassword: adminpvtkey
fepCustomCerts:
  - userName: user1
    userCrt: user1crt
    userCa: user1ca
  - userName: mydbuser
    userCrt: mydbusercrt
    userCa: mydbuserca

```

Note: The parameters which are in black in fepSecrets are mandatory.

4.5.6.3 GCP Provider for Secret Store CSI Driver

```

spec:
  ...
  fepChildCrVal:
    secretStore:
      method: csi
      csi:
        providerName: gcp
        gcpProvider:
          credentials:
            fepSecrets:
              - pgadminpassword: pgadminpassword
              - tdepassphrase: passphrase
              - systemCertificates: systemCerts
              - pguser: pgusername
              - pgpassword: pgpwd
              - pgdb: pgdbsecret
              - pgrepluser: pgrepluser
              - pgreplpassword: pgreplpassword
              - pgRewinduser: pgRewinduser
              - pgRewindpassword: pgRewindpassword
              - pgMetricsUser: metricsuser
              - pgMetricsPassword: metricspwd
              - patronitls: patronicrt
              - patronitlscacrt: patronica
              - postgrestls: postgrescrt
              - postgrestlscacrt: postgresca

```



```

- pgAdminTls: admincrt
- pgAdminTlscacrt: adminca
- pgAdminTls_privateKeyPassword: adminpvtkey
- pgRewindUserTls: rewindcrt
- pgRewindUserTlscacrt: rewindca
- pgRewindUserTls_privateKeyPassword: rwndpvtkey
- pgrepluserTls: replcrt
- pgrepluserTlscacrt: replca
- pgrepluserTls_privateKeyPassword: replpvtkey
- pgMetricsUserTls: metricscrt
- pgMetricsUserTlscacrt: metricsca
- pgMetricsUserTls_privateKeyPassword: adminpvtkey
fepCustomCerts:
  - userName: user1
    userCrt: user1crt
    userCa: user1ca
  - userName: mydbuser
    userCrt: mydbusercrt
    userCa: mydbuserca

```

Note: The parameters which are in black in fepSecrets are mandatory.

4.5.6.4 HashiCorp Vault Provider for Secret Store CSI Driver

```

spec:
  ...
  fepChildCrVal:
    secretStore:
      method: csi
      csi:
        providerName: vault
        vaultProvider:
          roleName: "database"
          vaultAddress: "http://vault-url-addr:8765"
          fepSecrets:
            - pgadminpassword: pgadminpassword
            - tdepassphrase: passphrase
            - systemCertificates: systemCerts
            - pguser: pgusername
            - pgpassword: pgpwd
            - pgdb: pgdbsecret
            - pgrepluser: pgrepluser
            - pgreplpassword: pgreplpassword
            - pgRewinduser: pgRewinduser
            - pgRewindpassword: pgRewindpassword
            - pgMetricsUser: metricsuser
            - pgMetricsPassword: metricspwd
            - patronitls: patronicrt
            - patronitlscacrt: patronica
            - postgrestls: postgrescrt
            - postgrestlscacrt: postgresca
            - pgAdminTls: admincrt
            - pgAdminTlscacrt: adminca
            - pgAdminTls_privateKeyPassword: adminpvtkey
            - pgRewindUserTls: rewindcrt
            - pgRewindUserTlscacrt: rewindca
            - pgRewindUserTls_privateKeyPassword: rwndpvtkey
            - pgrepluserTls: replcrt
            - pgrepluserTlscacrt: replca
            - pgrepluserTls_privateKeyPassword: replpvtkey
            - pgMetricsUserTls: metricscrt
            - pgMetricsUserTlscacrt: metricsca
            - pgMetricsUserTls_privateKeyPassword: adminpvtkey

```

```
fepCustomCerts:
- userName:user1
  userCrt: user1crt
  userCa: user1ca
- userName: mydbuser
  userCrt: mydbusercrt
  userCa: mydbuserca
```

Note: The parameters which are in black in fepSecrets are mandatory.

4.6 Deploying a customized FEP server container image

4.6.1 Requirements

The procedures documented below assume the use of docker command to build container image. Building container images using alternative tools such as podman is beyond the scope of this document.

4.6.2 Build custom FEP image with extension

Before building a new custom FEP Server container image, it is important to understand several build instructions specific to that image.

- FEP server container image is built on top of UBI8/UBI9 minimal image, ubi-minimal
- USER is default to 26

UBI8/UBI9 minimal image uses microdnf as package manager. Microdnf does not support installing RPM packages from remote URL or local file, only from a YUM repository. If you want to install RPM package that is not in YUM repository, first download the package and install it with rpm. However, rpm has the drawback that it does not resolve dependencies. The only way to resolve this problem is to install dnf first and use dnf to install packages from remote URL or local file.

As USER is default to 26, it does not have the permission to install RPM packages or write files to system directory such as /usr/bin, /usr/local/bin, etc. To workaround this issue, first set USER to root to continue the customization and set it back to 26.

```
FROM: quay.io/fujitsu/fujitsu-enterprise-postgres-16-server:ubi9-16-1.0

USER root

RUN ... (customization)

USER 26
```

4.6.3 Adding SQLite Foreign Data Wrapper to FEP Server Container

We will demonstrate adding the SQLite Foreign Data Wrapper module to FEP Server container.

1. Create Dockerfile

```
#use FEP 16 image as a base to compile sqlite_fdw
FROM quay.io/fujitsu/fujitsu-enterprise-postgres-16-server:ubi9-16-1.0 as compile-sqlite_fdw

#change the user with root privilege
USER root

# install build tools
RUN microdnf -y install cmake gcc-c++ libtool clang which openssl-devel git llvm gettext

# Install sqlite_fdw build require
RUN microdnf install -y sqlite-devel

# Download sqlite_fdw source
RUN curl -sSL https://github.com/pgspider/sqlite_fdw/archive/refs/tags/v2.3.0.tar.gz | tar -zxf -
```

```
# Compile sqlite_fdw
RUN cd /sqlite_fdw-2.3.0 && \
    make install USE_PGXS=1

#Use base image is from FEPContainer to build the custom image
FROM quay.io/fujitsu/fujitsu-enterprise-postgres-16-server:ubi9-16-1.0

#change the user with root privilege
USER root

#copy the prepared OSS extension binaries to FEP server lib folder
COPY --from=compile-sqlite_fdw /opt/fsepv16server64/lib/sqlite_fdw.so /opt/fsepv16server64/lib/
COPY --from=compile-sqlite_fdw /opt/fsepv16server64/lib/bitcode/sqlite_fdw /opt/fsepv16server64/lib/bitcode/
COPY --from=compile-sqlite_fdw /opt/fsepv16server64/share/extension/sqlite_fdw* /opt/fsepv16server64/share/extension/

# Install sqlite_fdw run time dependencies
RUN microdnf install -y sqlite-libs

#change the user to postgresql
USER 26
```

2. Build custom image

```
docker build -f Dockerfile -t my.registry/my-repo/fep-16-server-sqlite_fdw:ubi9-16-1.0
```

3. Push image to custom container registry

```
docker push my.registry/my-repo/fep-16-server-sqlite_fdw:ubi9-16-1.0
```

4.6.4 Create FEP Cluster with custom image

If the custom container registry requires authentication, create a pull secret with the name quay-pull-secret. FEP Operator will use this pull secret to download container image.

```
kind:Secret
apiVersion:v1
metadata:
  name:quay-pull-secret
  namespace:fep-container-ct
data:
  .dockerconfigjson:~>-
  xxxxxxCI6ICIiCiAgICB9CiAgfQp9
type:kubernetes.io/dockerconfigjson
```

Create FEP Cluster CR

```
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  name: sqlite-fdw
  namespace: fep-container-ct
spec:
  fep:
    forceSsl: true
  ...
  image:
    image: 'my.registry/my-repo/fep-16-server-sqlite_fdw:ubi9-16-1.0'
  ...
```

```
instances: 1
```

Deploy FEPCluster

```
oc apply -f sqlite_fdw.yaml
```

Create extension

```
postgres# CREATE EXTENSION sqlite_fdw;
CREATE EXTENSION
postgres=#
```

4.7 Configuration FEP to Perform MTLS

All three traffic can be secured by using TLS connection protected by certificates:

- Postgres traffic from Client Application to FEPCluster
- Patroni RESTAPI within FEPCluster
- Postgres traffic within FEPCluster (e.g. replication, rewind)

4.7.1 When Using an Automatically Generated Certificate

4.7.1.1 How to Create a Certificate

Operators can automatically generate certificates and use MTLS for communication between containers to improve safety.

If cert-manager is installed in the Kubernetes environment, use cert-manager; if cert-manager is not installed, use the openssl command in the container to generate the certificate.

If the certificate is automatically generated, the default connection method to PostgreSQL is only mutual authentication TLS communication.

If you want to change the authentication method for each user, define the connection method for each user in spec.fepChildCrVal.customPgHba.

To change the automatic certificate generation feature, create the ConfigMap "fepopr-cert-method" below before installing the operator. cert-manager, openssl, and disable can be specified for data.fepopr-cert-method. To disable the automatic certificate generation feature, specify disable.

```
$ cat > /tmp/cert-method.yaml <<EOF
kind: ConfigMap
apiVersion: v1
metadata:
  name: fepopr-cert-method
immutable: true
data:
  fepopr-cert-method: disable
EOF

$ kubectl -n <namespace> apply -f /tmp/cert-method.yaml
```

If you want to use a manually created certificate, refer to "[4.7.2 When Using Your Own Certificate](#)". If you want to use your own certificate, disable the automatic certificate generation feature.

4.7.1.2 How to Create a Client Certificate

The automatic certificate generation feature automatically generates system user certificates for database management. If you want to create a certificate for a general user who connects to the database, refer to the following steps.

This section explains how to create a Secret resource containing a certificate to be mounted in the client container. The created Secret resource can mount on the client's Pod and used to connect to FEPCluster using certificate authentication.

When using cert-manager to generate certificates

1. Verify that the Issuer resource "<namespace>-ca-issuer" exists.
2. Create a template for the Certificate resource.

```
$ cat mydbuser-cert.yaml
apiVersion:cert-manager.io/v1
kind:Certificate
metadata:
  name: "<fepclustername>-<username>-cert"
  namespace: "<namespace>"
spec:
  commonName: "< username >"
  issuerRef:
    name: <namespace>-ca-issuer
    secretName: "<fepclustername>-<username>-cert"
```

3. Apply the Certificate resource.

```
$ kubectl apply -f mydbuser-cert.yaml
```

When using openssl to generate certificates

1. Get the CA certificate stored in the Secret "fepopr-root-secret".

```
$ kubectl -n <namespace> get secret fepopr-root-secret-o jsonpath='{.data.tls\.key}' | base64
-d > <namespace>-ca.key

$ kubectl -n <namespace> get secret fepopr-root-secret-o jsonpath='{.data.tls\.cert}' |
base64 -d > <namespace>-ca.crt
```

2. Create the certificate using the following command.

```
$ serial=$(openssl rand -hex 8)
$ openssl genrsa -out <fepclustername>-<username>.key 2048
$ openssl req -new -key <fepclustername>-<username>.key -out <fepclustername>-<username>.csr -
subj "/CN=<username>"
$ openssl x509 -req -in <fepclustername>-<username>.csr -CA <namespace>-ca.crt -CAkey
<namespace>-ca.key -set_serial "0x${serial}" -out <fepclustername>-<username>.crt -days 365
```

3. Create a secret to store the certificate.

```
$ kubectl -n <namespace> create secret tls <fepclustername>-<username>-cert --cert
<fepclustername>-<username>.crt -key=<fepclustername>-<username>.key
```

4.7.2 When Using Your Own Certificate

Here, we provide two methods to create certificates for securing the TLS connection and provide mutual authentication. The first method is to create and renew certificate manually. The second method is to use CertManager to create an automatically renew certificate.



Note

The following considerations apply to client connections to a database cluster in an MTLS configuration:.

- Distribute the Root certificate for server (validation) that you specified when you created the MTLS database cluster to the client machines.
- Create and use a new client certificate.
- If the server root certificate and the client root certificate are different, a server-side configuration update is required.

4.7.2.1 Manual Certificate Management

Overview of Procedures

The procedures to enable MTLS communication are listed below:

1. Create a self signed certificate as CA
2. Create Configmap to store CA certificate
3. Create a password for protecting FEP Server private key (optional)
4. Create FEP Server private key
5. Create FEP Server certificate signing request
6. Create FEP Server certificate signed by CA
7. Create TLS Secret to store FEP Server certificate and key
8. Create private key for Patroni
9. Create certificate signing request for Patroni
10. Create certificate signed by CA for Patroni
11. Create TLS secret to store Patroni certificate and key
12. Create private key for "postgres" user client certificate
13. Create certificate signing request for "postgres" user client certificate
14. Create client certificate for "postgres" user
15. Create TLS secret to store "postgres" certificate and key
16. Repeat step 12-15 for "repluser" and "rewinduser"



Note

- The information in the manual is only an example, and in operation, use a certificate signed by a certificate authority (CA) that the user can trust.
- When working on a Kubernetes cluster, replace the oc command with the kubectl command.

Creating a CA Certificate

1. Create a self signed certificate as CA

```
openssl genrsa -aes256 -out myca.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....++++
.....++++
e is 65537 (0x010001)
Enter pass phrase for myca.key: 0okm9ijn8uhb7ygv
Verifying - Enter pass phrase for myca.key: 0okm9ijn8uhb7ygv

cat << EOF > ca.cnf
```

```
[req]
distinguished_name=req_distinguished_name
x509_extensions=v3_ca
[v3_ca]
basicConstraints = critical, CA:true
keyUsage=critical,keyCertSign,digitalSignature,cRLSign
[req_distinguished_name]
commonName=Common Name
EOF

openssl req -x509 -new -nodes -key myca.key -days 3650 -out myca.pem -subj "/O=My Organization/OU=CA /CN=My Organization Certificate Authority" -config ca.cnf
Enter pass phrase for myca.key: 0okm9ijn8uhb7ygv
```

2. Create Configmap to store CA certificate

```
oc create configmap cacert --from-file=ca.crt=myca.pem -n my-namespace
```

3. Create a password for protecting FEP Server private key (optional)

```
oc create secret generic mydb-fep-private-key-password --from-literal=keypassword=abcdefghijk -n my-namespace
```

Creating a Server Certificate

4. Create FEP Server private key

```
openssl genrsa -aes256 -out fep.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for fep.key: abcdefghijk
Verifying - Enter pass phrase for fep.key: abcdefghijk
```

5. Create FEP Server certificate signing request

```
cat << EOF > san.cnf
[SAN]
subjectAltName = @alt_names
[alt_names]
DNS.1 = *.my-namespace.pod
DNS.2 = *.my-namespace.pod.cluster.local
DNS.3 = mydb-primary-svc
DNS.4 = mydb-primary-svc.my-namespace
DNS.5 = mydb-primary-svc.my-namespace.svc
DNS.6 = mydb-primary-svc.my-namespace.svc.cluster.local
DNS.7 = mydb-replica-svc
DNS.8 = mydb-replica-svc.my-namespace
DNS.9 = mydb-replica-svc.my-namespace.svc
DNS.10 = mydb-replica-svc.my-namespace.svc.cluster.local
EOF

openssl req -new -key fep.key -out fep.csr -subj "/CN=mydb-headless-svc" -reqexts SAN -config
<(cat /etc/pki/tls/openssl.cnf <(cat san.cnf))
Enter pass phrase for fep.key: abcdefghijk
```




Note

The cluster name and namespace must be changed appropriately.

If you are connecting from outside the OCP cluster, you must also include the host name used for that connection.

6. Create FEP Server certificate signed by CA

```
openssl x509 -req -in fep.csr -CA myca.pem -CAkey myca.key -out fep.pem -days 365 -extfile
<(cat /etc/pki/tls/openssl.cnf <(cat san.cnf)) -extensions SAN -CAcreateserial # all in one line
Signature ok
subject=/CN=mydb-headless-svc
Getting CA Private Key
Enter pass phrase for myca.key: 0okm9ijn8uhb7ygv
```

7. Create TLS Secret to store FEP Server certificate and key

```
oc create secret generic mydb-fep-cert --from-file=tls.crt=fep.pem --from-file=tls.key=fep.key -n
my-namespace
```

8. Create private key for Patroni

At the moment, FEP container does not support password protected private key for Patroni.

```
openssl genrsa -out patroni.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
```

9. Create certificate signing request for Patroni

```
cat << EOF > san.cnf
[SAN]
subjectAltName = @alt_names
[alt_names]
DNS.1 = *.my-namespace.pod
DNS.2 = *.my-namespace.pod.cluster.local
DNS.3 = mydb-primary-svc
DNS.4 = mydb-primary-svc.my-namespace
DNS.5 = mydb-replica-svc
DNS.6 = mydb-replica-svc.my-namespace
DNS.7 = mydb-headless-svc
DNS.8 = mydb-headless-svc.my-namespace
EOF

openssl req -new -key patroni.key -out patroni.csr -subj "/CN=mydb-headless-svc" -reqexts SAN -
config <(cat /etc/pki/tls/openssl.cnf <(cat san.cnf)) # all in one line
```



Note

The cluster name and namespace must be changed appropriately.

If you are connecting from outside the OCP cluster, you must also include the host name used for that connection.

10. Create certificate signed by CA for Patroni

```
openssl x509 -req -in patroni.csr -CA myca.pem -CAkey myca.key -out patroni.pem -days 365 -extfile
<(cat /etc/pki/tls/openssl.cnf <(cat san.cnf)) -extensions SAN -CAcreateserial # all in one line
Signature ok
subject=/CN=mydb-headless-svc
Getting CA Private Key
Enter pass phrase for myca.key: 0okm9ijn8uhb7ygv
```

11. Create TLS secret to store Patroni certificate and key

```
oc create secret tls mydb-patroni-cert --cert=patroni.pem --key=patroni.key -n my-namespace
```

Creating a User Certificate

12. Create private key for "postgres" user client certificate

At the moment, SQL client inside FEP server container does not support password protected certificate.

```
openssl genrsa -out postgres.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
```

13. Create certificate signing request for "postgres" user client certificate

```
openssl req -new -key postgres.key -out postgres.csr -subj "/CN=postgres"
```

14. Create client certificate for "postgres" user

```
openssl x509 -req -in postgres.csr -CA myca.pem -CAkey myca.key -out postgres.pem -days 365
Signature ok
subject=CN = postgres
Getting CA Private Key
Enter pass phrase for myca.key: 0okm9ijn8uhb7ygv
```

15. Create TLS secret to store "postgres" certificate and key

```
oc create secret tls mydb-postgres-cert --cert=postgres.pem --key=postgres.key -n my-namespace
```

16. Repeat step 12-15 for "repluser" and "rewinduser"

4.7.2.2 Automatic Certificate Management

There are many Certificate Management tools available in the public. In this example, we will use cert-manager for the purpose.



Note

- Note that certificates created in this example are not password protected.
- When working on a Kubernetes cluster, replace the oc command with the kubectl command.

Install cert-manager

```
oc create namespace cert-manager
```

```
oc apply -f https://github.com/jetstack/cert-manager/releases/download/v1.3.0/cert-manager.yaml
```

Create a Self Signed Issuer (This can be namespace specific or cluster wise)

This example creates an Issuer, that can create self signed certificate, in namespace my-namespace.

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigned-issuer
  namespace: my-namespace
spec:
  selfSigned: {}
EOF
```

Create a Self Signed CA certificate using selfsigned-issuer

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: cacert
  namespace: my-namespace
spec:
  subject:
    organizations:
      - My Organization
    organizationalUnits:
      - CA
  commonName: "My Organization Certificate Authority"
  duration: 87600h
  isCA: true
  secretName: cacert
  issuerRef:
    name: selfsigned-issuer
EOF
```

The above command will create a self signed Root certificate and private key stored in the Kubernetes secret "cacert" in namespace my-namespace.

Create a CA Issuer with above certificate

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ca-issuer
  namespace: my-namespace
spec:
  ca:
    secretName: cacert
EOF
```

Create FEP Server certificate using above CA Issuer

Assuming FEPCluster name is mydb in namespace my-namespace.

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: mydb-fep-cert
  namespace: my-namespace
spec:
  subject:
    commonName: "mydb-headless-svc"
    dnsNames:
      - "*.my-namespace.pod"
      - "*.my-namespace.pod.cluster.local"
      - "mydb-primary-svc"
      - "mydb-primary-svc.my-namespace"
      - "mydb-primary-svc.my-namespace.svc"
      - "mydb-primary-svc.my-namespace.svc.cluster.local"
      - "mydb-replica-svc"
      - "mydb-replica-svc.my-namespace"
      - "mydb-replica-svc.my-namespace.svc"
      - "mydb-replica-svc.my-namespace.svc.cluster.local"
  duration: 8760h
  usages:
    - server auth
  secretName: mydb-fep-cert
  issuerRef:
    name: ca-issuer
EOF
```

Create Patroni certificate using above CA Issuer

Assuming FEPCluster name is mydb in namespace my-namespace.

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: mydb-patroni-cert
  namespace: my-namespace
spec:
  subject:
    commonName: "mydb-headless-svc"
    dnsNames:
      - "*.my-namespace.pod"
      - "*.my-namespace.pod.cluster.local"
      - "*.mydb-primary-svc"
      - "*.mydb-primary-svc.my-namespace"
      - "*.mydb-replica-svc"
      - "*.mydb-replica-svc.my-namespace"
  duration: 8760h
  usages:
    - server auth
  secretName: mydb-patroni-cert
  issuerRef:
    name: ca-issuer
EOF
```

Create postgres user client certificate

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: mydb-postgres-cert
  namespace: my-namespace
spec:
  subject:
    commonName: "postgres"
    duration: 8760h
  usages:
    - client auth
  secretName: mydb-postgres-cert
  issuerRef:
    name: ca-issuer
EOF
```

Create repluser user client certificate

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: mydb-repluser-cert
  namespace: my-namespace
spec:
  subject:
    commonName: "repluser"
    duration: 8760h
  usages:
    - client auth
  secretName: mydb-repluser-cert
  issuerRef:
    name: ca-issuer
EOF
```

Create FEPLogging(Fluentd) server certificate using above CA Issuer

Assuming FEPLogging name is **nfl** in namespace **fepprocessing-dev**.

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: fluentd-cert
  namespace: fepprocessing-dev
spec:
  subject:
    commonName: "nfl-fluentd-headless-service"
  dnsNames:
    - 'nfl-fluentd-headless-service'
    - 'nfl-fluentd-headless-service.fepprocessing-dev'
    - 'nfl-fluentd-headless-service.fepprocessing-dev.svc'
    - 'nfl-fluentd-headless-service.fepprocessing-dev.svc.cluster.local'
  duration: 8760h
  usages:
    - server auth
  secretName: fluentd-cert
  issuerRef:
```

```
name: ca-issuer
EOF
```

Create FELogging client(prometheus) certificate

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: prometheus-cert
  namespace: felogging-dev
spec:
  subject:
    commonName: "prometheus"
    duration: 8760h
  usages:
    - client auth
  secretName: prometheus-cert
  issuerRef:
    name: ca-issuer
EOF
```

Create FELogging client(fluentbit) certificate

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: fluentbit-cert
  namespace: felogging-dev
spec:
  subject:
    commonName: "fluentbit"
    duration: 8760h
  usages:
    - client auth
  secretName: fluentbit-cert
  issuerRef:
    name: ca-issuer
EOF
```

Create FEExporter certificate using above CA Issuer

Assuming FEP Exporter name is **exp1** in namespace **my-namespace**.

```
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: feexporter-cert
  namespace: my-namespace
spec:
  subject:
    commonName: "exp1-service"
    dnsNames:
      - 'exp1-service'
      - 'exp1-service.feexporter-dev'
      - 'exp1-service.feexporter-dev.svc'
      - 'exp1-service.feexporter-dev.svc.cluster.local'
  duration: 8760h
```

```

usages:
- server auth
secretName: fepexporter-cert
issuerRef:
  name: ca-issuer
EOF

```

Create FEPEXporter user client(prometheus) certificate

```

cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: prometheus-cert
  namespace: my-namespace
spec:
  subject:
    commonName: "prometheus"
    duration: 8760h
    usages:
    - client auth
  secretName: prometheus-cert
  issuerRef:
    name: ca-issuer
EOF

```

4.7.2.3 Deploy FEPCluster with MTLS support

Deploy FEPCluster with manual certificate management

Use the following yaml as an example to deploy a FEPCluster with Manual Certificate Management. MTLS related parameters are highlighted in **Red**.

```

apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  name: mydb
  namespace: my-namespace
spec:
  fep:
    usePodName: true
    patroni:
      tls:
        certificateName: mydb-patroni-cert
        caName: cacert
    postgres:
      tls:
        certificateName: mydb-fep-cert
        caName: cacert
        privateKeyPassword: mydb-fep-private-key-password
  forceSsl: true
  podAntiAffinity: false
  mcSpec:
    limits:
      cpu: 500m
      memory: 700Mi
    requests:
      cpu: 200m
      memory: 512Mi
  customAnnotations:
    allDeployments: {}

```

```

servicePort: 27500
image:
  image: 'quay.io/fujitsu/fujitsu-enterprise-postgres-16-server:ubi9-16-1.0'
  pullPolicy: IfNotPresent
sysExtraLogging: false
podDisruptionBudget: false
instances: 3
syncMode: 'on'
fepChildCrVal:
customPgAudit: |
  # define pg audit custom params here to override defaults.
  # if log volume is not defined, log_directory should be
  # changed to '/database/userdata/data/log'
  [output]
  logger = 'auditlog'
  log_directory = '/database/log/audit'
  [rule]
customPgHba: |
  # define pg_hba custom rules here to be merged with default rules.
  # TYPE      DATABASE      USER      ADDRESS      METHOD
  hostssl     all              all       0.0.0.0/0     cert
  hostssl     replication     all       0.0.0.0/0     cert
customPgParams: >+
  # define custom postgresql.conf parameters below to override defaults.
  # Current values are as per default FEP deployment
  shared_preload_libraries='pgx_datamasking,pgaudit,pg_prewarm'
  session_preload_libraries='pg_prewarm'
  max_prepared_transactions = 100
  max_worker_processes = 30
  max_connections = 100
  work_mem = 1MB
  maintenance_work_mem = 12MB
  shared_buffers = 128MB
  effective_cache_size = 384MB
  checkpoint_completion_target = 0.8

  # tcp parameters
  tcp_keepalives_idle = 30
  tcp_keepalives_interval = 10
  tcp_keepalives_count = 3

  # logging parameters in default fep installation
  # if log volume is not defined, log_directory should be
  # changed to '/database/userdata/data/log'
  log_directory = '/database/log'
  log_filename = 'logfile-%a.log'
  log_file_mode = 0600
  log_truncate_on_rotation = on
  log_rotation_age = 1d
  log_rotation_size = 0
  log_checkpoints = on
  log_line_prefix = '%e %t [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h'
  log_lock_waits = on
  log_autovacuum_min_duration = 60s
  logging_collector = on
  pgaudit.config_file='/opt/app-root/src/pgaudit-cfg/pgaudit.conf'
  log_replication_commands = on
  log_min_messages = WARNING
  log_destination = stderr

  # wal_archive parameters in default fep installation
  archive_mode = on
  archive_command = '/bin/true'

```



```

wal_level = replica
max_wal_senders = 12
wal_keep_segments = 64

storage:
  dataVol:
    size: 2Gi
    storageClass: nfs-client
  walVol:
    size: 1200Mi
    storageClass: nfs-client
  logVol:
    size: 1Gi
    storageClass: nfs-client
sysUsers:
  pgAdminPassword: admin-password
  pgdb: mydb
  pgpassword: mydbpassword
  pguser: mydbuser
  pgrepluser: repluser
  pgreplpassword: repluserpwd
  pgRewindUser: rewinduser
  pgRewindPassword: rewinduserpwd
  pgAdminTls:
    certificateName: mydb-postgres-cert
    caName: cacert
    sslMode: prefer

  pgrepluserTls:
    certificateName: mydb-repluser-cert
    caName: cacert
    sslMode: prefer

  pgRewindUserTls:
    certificateName: mydb-rewinduser-cert
    caName: cacert
    sslMode: prefer

tdepassphrase: tde-passphrase
systemCertificates:
  key: |-
    -----BEGIN RSA PRIVATE KEY-----
    MIIeowIBAAKCAQEAO0DFkImha8CIJiVcwXbBPLl+/DmS9/ipRhQQHxf05x7jS0nse
    IHdFd6+Qx2GX8KAiAhVykf6kfacyBYTATU1xDgwwTm82KVRPh+kZDIj2wPcJr14m
    mTP6I6a2mavUgDhezHc9F8/dchYj3cw81X0kU6xamqrKQYlXQH48NkI0qcwh06sK
    AHF4ewfCr80t44xADIA1JcU2CS1RKSZEtURZ+30Py+j907Enjp1YR33ZKUHW30pU
    9dpIneyfXBN/pT6cX3MetYwtgmpV/pHqY8pbxqGfoYRhgQDsSRC14dtlecaZeZ4j
    uT0otcPkZELHP6eu8gaLtycG9lpbAMQL5w0r8QIDAQABAOIBACq213qPuoiMExrQ
    fQXaNJmqNYK4fJqXCB6oUwf0F1u4ubkx5V532hLSPHwLs+a0lAWlbNozSoBV0u8G
    64VvrA9bv3/cJVqZZ6/UzUTbHPU+Ogh24qhwF5QU8kXZEU11To3YsPoftalgjX9G
    Ff0fLcLVC8nL3K9RiaDXxXbEYpWrYu39M3FCpAXAzV2PrNxsP9PKyNWHnBPc08z5
    tFj45/bHn+j31AVVvgWtqz0pLks57hc4Q7yW/2RoRYq2md1KI7090LNwtkWEOVqb
    qnraorh2TwGnNa0B5oX5/lJvKtLq778fw96jGqykBr0+DKozj9rlr10GgYOKDwld
    nsZJPAECgYEA+0qf/fxtPdsNGial2Z/heewvtaxjw/WoEVBFEcb6/y4Ro7aux9nB
    16FcVi79Cwfp0UTJ7cnZvYSmBk5GWE0bEIAeo61lvM/Qe1tM5+usAPd5/TcHXLye
    920Nxmq7h3F4UXEkMayak8Lpu/TdmR5u0aL+m4aEu+XMY5tLxqDCnyECgYEA1h4X
    jCPi7Ja5CHK7a2Ud4TL2DNpIBE6GSK9iQ+0xFL6Tsik2Sfu6n8mx2sh+Jm0KHTiE
    /gWHDhQZSSwiuUlfHoYEQ3Rq8S6Av3GsGtRSp003j7BE8C20Vpt0FnNTjZmdzf2/
    YZxc5KuyLh9qeY7Y7ce0sWA8JckDgMHPYzyLatECgYBALD0TPgDr8Y1vMIDmLqH
    FF04eTk/TBYIYKltgJ81KqthibeFzp4q+W7UyUhzj5a4XQ0ySlfYhFpJReTc3JEd
    r+o2SH3ymuEkqUpZZjyptrMbWN4g3t4TDjaHqo6QQbD+GdcZyNy9M1Np9N5p17E
    fUEm14dg6d3H0Ehs7QVAAQKBgQDRUx3mLXc9oKRINBiYDerGLJILQqLBQxtYl81T
    ZuFizGWL8w+PCIAMkpxDrVpWqqcGpiiuRi2ElbPap0aOg2epaY/LJscd/j5z6uc8

```

```

W3JoNljPkoRa4f0578Pv5tM6TYH0zLF5VeoIy/a8sI3hRNuiqkM/+TsUHY5FJDRh
aeDk4QKBgCOHtevvrM+WuwakzD6LNCbb8H6fvZ3WRAT8BYyz3wW9YfnV4J4uh/B1
moWYgIK2UpkrhA8scMUC790FoybQeParQ35x7Jl91bmTkkCqsX63fyqqYhx3SXRl
JSktmH4E2cGmosZisjB7COKHR32w0J5JCgaGInQxjldbGrwhZQpn
-----END RSA PRIVATE KEY-----

crt: |-
-----BEGIN CERTIFICATE-----
MIID2CCAsCgAwIBAgIQDfFYteD4kZj4Sko2iy1IJTANBgkqhkiG9w0BAQsFADBx
MRgwFgYDVQQKEw9NeSBPcmdbmbl6YXRpb24xCzAJBgNVBAsTAkNBMS4wLAYDVQQD
EyVNeSBPcmdbmbl6YXRpb24gQ2VydGlmawNhdGUGbG9yaXR5MB4XDTEyMDQy
MDAwMDQ1OVVoXDTIxMDQyMDAwMDQ1OVowGDEWMBQGA1UEAwNKi5jaGctcHRjLnBv
ZDCCASIdQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANaxZCJoWvAiCYLXMF2w
T5S/vw5kvf4qUYUEB8Xzuce40jp7HiB3RXevkMdhL/CgIgIVcpH+pH2nMAWEwE1N
cQ4MFk5vNiLUT4fpgQyI9sD3Ca9eJpkz+i0mtpmr1IA4Xsx3PRfP3XIWI93MPNV9
JF0swppqyGJcUB+PDZCNkNMITurCgBxeHlnwq/DreOMQAYANSXFNgtUSkmRLVE
Wft9D8vo/d0xJ46dWEd92SLB8N9KVPXaSJ3sn1wTf6U+nF9zHrWMLYJqVf6R6mPK
W8ahn6MkYYEA7EkQpEbZxNgmXmeI7kzqLXD5GRCxz+nrvIGi7cnBvZaWwDEJecN
K/ECaWEAAa0B3jCB2zATBgNVHSUEDDAKBggrBgEFBQcDATAMBGNVHRMBAf8EAjAA
MI61BgNVHREEga0wgaqCCWxvY2FsaG9zdIIbKi5jaGctcHRjLnBvZC5jbHVzdG9y
LmxvY2FsgHMQmL15ZGItaGVhZGxlM3Mtc3ZjghsqLm15ZGItaGVhZGxlM3Mtc3Zj
LmNoZy1wdG0ChyoubXlkYi1oZWFKbGVzcy1zdmMuY2hnLXB0Yy5zdmOCLSoubXlk
Yi1oZWFKbGVzcy1zdmMuY2hnLXB0Yy5zdmMuY2x1c3Rlc15sb2NhbDANBgkqhkiG
9w0BAQsFAAOCAQEALnhliDflu+BHp5conq4dXBwD/Ti2YR5TWQixM/0a60D4KecZ
MmaLl0T+0JJvA/j2IufZpc7dzEx5mZDKR2CRmoq10qZXqCRRtBZSxm6ARQWoYpeg
9c0l4f8roxrkMGUKVPTKUwAvbnNYhD2l6PlBPwMpkMUfQFaSEXMaPyQKhrTQxdpH
WjuS540P0lm0peYu/yiaD98LtrTXnb6jch84SKf6Vii4HAVQyMeJaw+dpkqcI2+V
Q4fkWwYSjY8BncmXCwvHDLdy+s4EXWvHafhusuUhcP4HyMb1A6hd5hJhgFSnEvLy
kLA0L9LaScxee6V756Vt9Tn1NGjwmwyQD0hnQQ==
-----END CERTIFICATE-----

cacrt: |-
-----BEGIN CERTIFICATE-----
MIIDXCcAKSgAwIBAgIRAMPzF3BNFxt9HWE+NXlFQjQwDQYJKoZIhvcNAQELBQAw
VzEYMBYGA1UEChMPTXkgT3JnYW5pemF0aw9uMQswCQYDVQQLEwJDQTEuMCAwGA1U
EAxMlTXkgT3JnYW5pemF0aw9uIENlcnRpb2ljYXRlIEF1dGhvcml0eTAeFw0yMTA0
MTkwNDQ0MjNaFw0zMTA0MTcwNDQ0MjNaMFcxGDAWBgNVBAoTD015IE9yZ2FuaXph
dGlvb2JELMAKGA1UECmQ0ExLjAsBgNVBAMTJU15IE9yZ2FuaXphdGlvbiBDZXJ0
awZpY2F0ZSBBDXR0b3JpdHkwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQc5t6CS23Gk65YMw5e4i4xHldyxkCZS67w/6LWqeILYKmfAaeL83Wwy8MHUP0b
4mahtUafEzDEOX6+URf72J8m0voLdQ5FYr1AyUoyX8U90wGFqhbEgKRqt7vZEwIe
2961fwqHh6917zi4xmt5W6ZJ5dBQvtkhzB+Pf706KBYjHoCnBBkfNVzsfZQ/1hnR
0UzimfAc7Ze+UNwhXJhinFRJ3Yur+xi0TpPk1lGXPhLgFSQheKz4KepcbQEKejb
jg0dum10BYIXZTSSbi09rNmFUVLB5DcV0vZbSrGxLjWLBt5U8N2xf2d1bvkQw+bw
Kklf90G26bAi27tujurzn3r3AgMBAAGjIzAhMA4GA1UdDwEB/wQEAwICpDAPBgNV
HRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQAM0CN3n5C/KOT4uZ4ewwKK
rHmANBPVM9u6MJB08U62HcQLeoCuDFeU8zmUjLHjsQaPX64mJZlR7T5y52gEK05A
0qsBz3pg/vJ5DJTv0698+1Q1hB9k3smQdksAim19FZqysB7J4zK/+8aJ/q2kIFvs
Jk3ekwQdQ3xfggklBQVuf76gr1v0uYlPtFpffPlfcGZ06Im6mqbajenXoR1PxPB0
+zyCS8DkgPtDu1plrUwvXCFMYw9TPbzXKlt7tLsqRXogYLnXWJDzM1n0YcND+rDm
qxenV9Ir8RqZ0XSyUyZrKa5N4dhIhrzTAiNdeU5gzynX0z67u/Iefz1iK9ZcdE3
-----END CERTIFICATE-----

```

Deploy FEPCluster with automatic certificate management

Use the following yaml as an example to deploy a FEPCluster with Automatic Certificate Management. MTLS related parameters are highlighted in **Red**.

```

apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  name: mydb
  namespace: my-namespace
spec:
  fep:

```

```

usePodName: true
patroni:
  tls:
    certificateName: mydb-patroni-cert
postgres:
  tls:
    certificateName: mydb-fep-cert
forceSsl: true
podAntiAffinity: false
mcSpec:
  limits:
    cpu: 500m
    memory: 700Mi
  requests:
    cpu: 200m
    memory: 512Mi
customAnnotations:
  allDeployments: {}
servicePort: 27500
image:
  image: 'quay.io/fujitsu/fujitsu-enterprise-postgres-16-server:ubi9-16-1.0'
  pullPolicy: IfNotPresent
sysExtraLogging: false
podDisruptionBudget: false
instances: '3'
syncMode: 'on'
fepChildCrVal:
  customPgAudit: |
    # define pg audit custom params here to override defaults.
    # if log volume is not defined, log_directory should be
    # changed to '/database/userdata/data/log'
    [output]
    logger = 'auditlog'
    log_directory = '/database/log/audit'
    [rule]
  customPgHba: |
    # define pg_hba custom rules here to be merged with default rules.
    # TYPE      DATABASE      USER      ADDRESS      METHOD
    hostssl     all             all       0.0.0.0/0     cert
    hostssl     replication    all       0.0.0.0/0     cert
  customPgParams: >+
    # define custom postgresql.conf parameters below to override defaults.
    # Current values are as per default FEP deployment
    shared_preload_libraries='pgx_datamasking,pgaudit,pg_prewarm'
    session_preload_libraries='pg_prewarm'
    max_prepared_transactions = 100
    max_worker_processes = 30
    max_connections = 100
    work_mem = 1MB
    maintenance_work_mem = 12MB
    shared_buffers = 128MB
    effective_cache_size = 384MB
    checkpoint_completion_target = 0.8

    # tcp parameters
    tcp_keepalives_idle = 30
    tcp_keepalives_interval = 10
    tcp_keepalives_count = 3

    # logging parameters in default fep installation
    # if log volume is not defined, log_directory should be
    # changed to '/database/userdata/data/log'

```

```

log_directory = '/database/log'
log_filename = 'logfile-%a.log'
log_file_mode = 0600
log_truncate_on_rotation = on
log_rotation_age = 1d
log_rotation_size = 0
log_checkpoints = on
log_line_prefix = '%e %t [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h'
log_lock_waits = on
log_autovacuum_min_duration = 60s
logging_collector = on
pgaudit.config_file='/opt/app-root/src/pgaudit-cfg/pgaudit.conf'
log_replication_commands = on
log_min_messages = WARNING
log_destination = stderr

# wal_archive parameters in default fep installation
archive_mode = on
archive_command = '/bin/true'
wal_level = replica
max_wal_senders = 12
wal_keep_segments = 64

storage:
  dataVol:
    size: 2Gi
    storageClass: nfs-client
  walVol:
    size: 1200Mi
    storageClass: nfs-client
  logVol:
    size: 1Gi
    storageClass: nfs-client
sysUsers:
  pgAdminPassword: admin-password
  pgdb: mydb
  pgpassword: mydbpassword
  pguser: mydbuser
  pgrepluser: repluser
  pgreplpassword: repluserpwd
  pgRewindUser: rewinduser
  pgRewindPassword: rewinduserpwd
  pgAdminTls:
    certificateName: mydb-postgres-cert
    sslMode: verify-full

  pgrepluserTls:
    certificateName: mydb-repluser-cert
    sslMode: verify-full

  pgRewindUserTls:
    certificateName: mydb-rewinduser-cert
    sslMode: verify-full

tdepassphrase: tde-passphrase
systemCertificates:
  key: |-
    -----BEGIN RSA PRIVATE KEY-----
    MIIeowIBAAKCAQEADfKImha8CIJiVcwXbBPll+/DmS9/ipRhQQHxf05x7jS0nse
    IHdFd6+Qx2GX8KAiAhVykf6kf6BYTATU1xDgwWTm82KVRPh+kZDIj2wPcJr14m
    mTP6I6a2mavUgDhezHc9F8/dchYj3cw81X0kU6xamqrKQYlXQH48NkI0qcwh06sK
    AHF4eWfCr80t44xADIA1JcU2CS1RKSZEtURZ+30Py+j907Enjp1YR33ZKUHW30pu
    9dpIneyfXBN/pT6cX3MetYwtgmpV/pHqY8pbxqGfoYRhgQDsSRCL4dt lecaZeZ4j

```

```
-----END RSA PRIVATE KEY-----
```

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----

- 89 -

```
+zyCS8DkgPtDu1plruwvXCFMYw9TPbzXKlt7t1sqRXogYLnXWJDzM1n0YcNd+rDm
qxenv9Ir8RqZ0XSYuUyzRka5N4dhIhrzTAiNdeU5gzynX0z67u/Iefz1iK9ZcdE3
-----END CERTIFICATE-----
```

4.7.2.4 Configurable Parameters

To enable MTLS, make changes to the following parameters.

Key	Value	Details
spec.fep.usePodName	True	For MTLS, this key must be defined and set to true. For TLS connection without MTLS, it can be omitted. However, it is recommended to set this to true as well.
spec.fep.patroni.tls.certificateName	<secret-name>	Name of Kubernetes secret that contains the certificate in tls.crt and private key in tls.key for Patroni REST API. For MTLS Patroni REST API communication, this key must be defined. The private key cannot be password protected. When using cert-manager, the secret also contains the CA bundle in ca.crt.
spec.fep.patroni.tls.caName	<configmap-name>	Name of Kubernetes configmap that contains the CA bundle. If using cert-manager, the ca.crt is already included in the secret above. In this situation, this key can be omitted.
spec.fep.postgres.tls.certificateName	<secret-name>	Name of Kubernetes secret that contains the certificate in tls.crt and private key in tls.key for Postgres server. For MTLS Postgres communication, this key must be defined. The private key can be password protected. When using cert-manager, the secret also contains the CA bundle in ca.crt.
spec.fep.postgres.tls.caName	<configmap-name>	Name of Kubernetes configmap that contains the CA bundle. If using cert-manager, the ca.crt is already included in the secret above. In this situation, this key can be omitted.
spec.fep.postgres.tls.privateKeyPassword	<secret-name>	Name of Kubernetes secret that contains the password for the private key for Postgres Server.
spec.fepChildCrVal.sysUsers.pgAdminTls.certificateName	<secret-name>	Name of Kubernetes secret that contains the certificate in tls.crt and private key in tls.key for "postgres" user. For MTLS Postgres communication, this key must be defined. The private key cannot be

Key	Value	Details
		password protected. When using cert-manager, the secret also contains the CA bundle in ca.crt.
spec.fepChildCrVal.sysUsers.pgAdminTls.caName	<configmap-name>	Name of Kubernetes configmap that contains the CA bundle. If using cert-manager, the ca.crt is already included in the secret above. In this situation, this key can be omitted.
spec.fepChildCrVal.sysUsers.pgAdminTls.sslMode	verify-full	For MTLS, this value must be set to verify-full. If only TLS is required, this can be set to verify-ca or prefer.
spec.fepChildCrVal.sysUsers.pgrepluserTls.certificateName	<secret-name>	Name of Kubernetes secret that contains the certificate in tls.crt and private key in tls.key for "repluser" user. For MTLS Postgres communication, this key must be defined. The private key cannot be password protected. When using cert-manager, the secret also contains the CA bundle in ca.crt.
spec.fepChildCrVal.sysUsers.pgrepluserTls.caName	<configmap-name>	Name of Kubernetes configmap that contains the CA bundle. If using cert-manager, the ca.crt is already included in the secret above. In this situation, this key can be omitted.
spec.fepChildCrVal.sysUsers.pgrepluserTls.sslMode	verify-full	For MTLS, this value must be set to verify-full. If only TLS is required, this can be set to verify-ca or prefer.
spec.fepChildCrVal.sysUsers.pgRewindUserTls.certificateName	<secret-name>	Name of Kubernetes secret that contains the certificate in tls.crt and private key in tls.key for "rewinduser" user. For MTLS Postgres communication, this key must be defined. The private key cannot be password protected. When using cert-manager, the secret also contains the CA bundle in ca.crt.
spec.fepChildCrVal.sysUsers.pgRewindUserTls.caName	<configmap-name>	Name of Kubernetes configmap that contains the CA bundle. If using cert-manager, the ca.crt is already included in the secret above. In this situation, this key can be omitted.
spec.fepChildCrVal.sysUsers.pgRewindUserTls.sslMode	verify-full	For MTLS, this value must be set to verify-full. If only TLS is required, this can be set to verify-ca or prefer.

It is also required to customize pg_hba.conf to perform MTLS. Below are two possible settings.

spec.fep.customPgHba	hostssl all all 0.0.0.0/0 cert hostssl replication all 0.0.0.0/0 cert
----------------------	--

The above setting will force FEP server to perform certification authentication. At the same time verify the authenticity of client certificate.

spec.fep.customPgHba	hostssl all all 0.0.0.0/0 md5 clientcert=verify-full hostssl replication repluser 0.0.0.0/0 md5 clientcert=verify-full
----------------------	---

The above setting will force FEP server to perform md5 authentication as well as verifying the authenticity of client certificate.

4.8 Replication Slots

4.8.1 Setting Up Logical Replication using MTLS

This section describes setup of logical replication.

To setup logical replication using MTLS, follow these steps:

1. Create two FEPClusters - (to act as Publisher and Subscriber) and ensure that they can communicate with each other. You can see the creation of FEPCluster in the ["4.1 Deploying FEPCluster using Operator"](#).
2. To setup Publisher, make following changes to the FEPCluster yaml of the cluster that you want to use as publisher:
 - a. Add section replicationSlots under spec.fep to create replication slots.

The "**database**" should be the name of the database for which we are setting up logical replication.

```
158 spec:
159   fep:
160     forceSsl: true
161     replicationSlots: |
162       myslot1:
163         type: logical
164         database: db1
165         plugin: pgoutput
166       myslot2:
167         type: logical
168         database: db1
169         plugin: pgoutput
170     podAntiAffinity: false
```

- b. Add section postgres under spec.fep as shown below.

caName = enter the name of configmap created for the CA

certificateName = secret created by the end user that contains server certificate

```
78 |         memory: 512Mi
79 |         customAnnotations:
80 |           allDeployments: {}
81 |         servicePort: 27500
82 |         postgres:
83 |           tls:
84 |             caName: cacert
85 |             certificateName: my-fep-cert
86 |         image:
```

- c. Change the value of wal_level parameter under spec.fepChildCrVal.customPgParams from replica to logical.

```
301 |
302 |         archive_mode = on
303 |
304 |         archive_command = 'pgbackrest --stanza=backupstanza
305 |         --config=/database/userdata/pgbackrest.conf archive-push %p'
306 |
307 |         wal_level = logical
308 |
309 |         max_wal_senders = 12
310 |
311 |         wal_keep_size = 401
```

- d. Add entry under spec.fepChildCrVal.customPgHba as shown below.

This requires the client to present a certificate and only certificate authentication is allowed.

Replace "SubClusterName" and "SubNamespace" with the appropriate values as per the Subscriber FEPCluster.

```
[rule]
customPgHba: |
  # define pg_hba custom rules here to be merged with default rules.
  # TYPE      DATABASE      USER      ADDRESS      METHOD
  hostssl all all <SubClusterName>-primary-svc.<SubNamespace>.svc.cluster.local cert
customPgParams: >
```

3. To setup Subscriber, make following changes to the FEPCluster yaml of the cluster that you want to use as subscriber:

- a. Add customCertificates under spec.fepChildCrVal as shown below.

caName = enter the name of configmap created for the CA (i.e. The CA certificate which is used to sign/authenticate the server/client certificates is mounted as a configMap called 'cacert')

certificateName = secret created by end user that contains a client certificate which can be verified by the server

username = name of the role created on publisher cluster for logical replication

```
74 |         fepChildCrVal:
75 |           customCertificates:
76 |             - caName: cacert
77 |               certificateName: my-logicalrepl-cert
78 |               userName: logicalrepluser
79 |           customPgAudit: |
80 |             # define pg audit custom params here to override defaults.
81 |             # if log volume is not defined, log_directory should be
```

4. Connect to the pod terminal of the Publisher FEPCluster and then connect to the postgres database as shown below.

```
sh-4.4$ psql -h /tmp -p 27500 -U postgres
Password for user postgres:
psql (13.1)
Type "help" for help.

postgres=#
```

5. Next, on the publisher side, connect to the database that contains the tables you want to replicate and create a role e.g., logicalrepluser and give the required permissions to this role.

Consider the below image as example only, the privileges to grant may differ as per the requirements.

```
db1=# CREATE ROLE logicalrepluser WITH REPLICATION LOGIN PASSWORD 'my_password';
CREATE ROLE
db1=# GRANT ALL PRIVILEGES ON DATABASE db1 TO logicalrepluser;
GRANT
db1=# GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO logicalrepluser;
GRANT
db1=#
```

6. At the Publisher side, create a publication and alter the publication to add the tables that need to be replicated.

```
db1=# create publication my_publication;
CREATE PUBLICATION
db1=# alter publication my_publication add table my_table;
ALTER PUBLICATION
db1=#
```

7. At the subscriber side, the custom certificates added in the above step 3.a will be mounted at the path /tmp/custom_certs/ as shown:

```
sh-4.4$ ls -rlt /tmp/custom_certs
total 0
drwxr-xr-t. 3 1001190000 root 103 Aug 10 10:08 logicalrepluser
sh-4.4$ ls -rlt /tmp/custom_certs/logicalrepluser
total 0
lrwxrwxrwx. 1 1001190000 root 14 Aug 10 10:08 tls.key -> ../data/tls.key
lrwxrwxrwx. 1 1001190000 root 14 Aug 10 10:08 tls.crt -> ../data/tls.crt
lrwxrwxrwx. 1 1001190000 root 13 Aug 10 10:08 ca.crt -> ../data/ca.crt
sh-4.4$
```

8. The structure of the table to be replicated should be present in the subscriber cluster since logical replication only replicates the data and not the table structure.

Create a subscription as shown below:

```
db1=# CREATE SUBSCRIPTION my_subscription CONNECTION 'host=fepcluster-publisher-primary-svc.ns-a.svc.cluster.local port=27500 sslcert=/tmp/custom_certs/logicalrepluser/tls.crt sslkey=/tmp/custom_certs/logicalrepluser/tls.key sslrootcert=/tmp/custom_certs/logicalrepluser/ca.crt sslmode=verify-full dbname=db1 user=logicalrepluser' PUBLICATION my_publication WITH (slot_name=myslot1, create_slot=false);
CREATE SUBSCRIPTION
```

The command in the above example is :

```
CREATE SUBSCRIPTION my_subscription CONNECTION 'host=fepcluster-publisher-primary-svc.ns-a.svc.cluster.local port=27500 sslcert=/tmp/custom_certs/logicalrepluser/tls.crt sslkey=/tmp/custom_certs/logicalrepluser/tls.key sslrootcert=/tmp/custom_certs/logicalrepluser/ca.crt sslmode=verify-full password=my_password user=logicalrepluser dbname=db1' PUBLICATION my_publication WITH (slot_name=myslot1, create_slot=false);
```

Host = primary service of the publisher FEP Cluster
sslcert, sslkey, sslrootcert = path to certificates mounted on the Subscriber FEP Cluster
user= Role created on the Publisher side
password= password for the role
dbname= database which contains the tables to be replicated

Where

Host = primary service of the publisher FEP Cluster
sslcert, sslkey, sslrootcert = path to certificates mounted on the Subscriber FEP Cluster
user= Role created on the Publisher side and used to establish logical replication connection from Subscriber to Publisher
dbname= database which contains the tables to be replicated

4.9 FEP Logging

FEPCluster generates log files and auditlog files, if configured, over the lifetime of execution. These log files can be useful for understanding cluster healthness and debugging purpose. By default, the log files are stored on persistent volume of the container. User can enable log monitoring feature by forwarding those log files and auditlog files to a analytics platform such as Elasticsearch.

There are two steps to enable monitoring and forwarding.

1. FEPLLogging Configuration - Creating FEP Logging instance
2. FEPCluster configuration - Enabling logging in FEPCluster

The FEP Logging instance is a standalone container running fluentd. It accepts log forwarded from FEP Clusters and aggregate data according to log entries severity and present that to Prometheus for monitoring and alerting purpose. It can optionally be configured to forward those logs to an Elasticsearch instance for detail analysis.

When logging is enabled on FEPCluster, a sidecar, containing fluentbit, will be deployed alongside the FEP server container. This fluentbit sidecar will monitor the FEP server log files and auditlog files on persistent volume and forward to the FEP Logging instance.

Multiple FEPClusters can forward logs to single FEPLLogging instance.

User can have two types of connection between FEPCluster & FEPLLogging

- Insecure connection: Without TLS/MTLS certificates
- Secure connection: With TLS/MTLS certificates

For the secure connections between the components, User have two options:

- User can use their own certificates
- User can generate self signed certificates (see "[4.7.2.2 Automatic Certificate Management](#)")

The FEP Logging instance can run standalone without additional component. For detail log analysis, the user can configure the FEP Logging instance to forward logs to Elastic Stack or Elastic Cloud. Please consult the [Elastic Document](#) on how to deploy a Elastic Stack or sign up to [Elastic Cloud](#).

4.9.1 FEPLLogging Configuration

This section describes how to deploy and configure FEP Logging instance via the FEPLLogging custom resource. FEPLLogging is a separate CR which will accept logs sent from FEPCluster and forwards them to Elasticsearch or Prometheus for raising alarm. User must create FEPLLogging CR before enabling FEPCluster logging feature.

4.9.1.1 FEPLLogging Custom Resources - spec

The fepLogging section needs to be added under spec to define required parameters for FEPLLogging configuration.

Following is a sample template :

```
spec:
  fepLogging:
    elastic:
      authSecret:
        secretName: elastic-auth
        passwordKey: password
        userKey: username
      host: elastic-passthrough.apps.openshift.com
      logstashPrefix: postgres
      port: 443
      scheme: https
      sslVerify: true
      tls:
        certificateName: elastic-cert
        caName: elastic-cacert
    image:
      pullPolicy: IfNotPresent
    mcSpec:
      limits:
        cpu: 500m
        memory: 700Mi
      requests:
        cpu: 200m
        memory: 512Mi
      restartRequired: false
      sysExtraLogging: false
      scrapeInterval: 30s
      scrapeTimeout: 30s
      tls:
        certificateName: fluentd-cert
        caName: cacert
    prometheus:
      ...
```

Below is the list of all parameters defined in the fepLogging section, along with their brief description

Custom Resource spec	Required/ Optional	Change Effect	Updating value allowed
spec.fepLogging.image.image	Optional	Fluentd Image of FEPLogging	Yes
spec.fepLogging.image.pullPolicy	Required	Fluentd Image pull policy of FEPLogging	Yes
spec.fepLogging.mcSpec.limits.cpu	Required	Max CPU allocated to fluentd container	Yes
spec.fepLogging.mcSpec.limits.memory	Required	Max memory allocated to fluentd container	Yes
spec.fepLogging.mcSpec.requests.cpu	Required	CPU allocation at start for fluentd container	Yes
spec.fepLogging.mcSpec.requests.memory	Required	Memory allocation at start for fluentd container	Yes
spec.fepLogging.sysExtraLogging	Required	To turn on extra debugging messages for operator, set value to true. It can be turned on/off at any time	Yes
spec.fepLogging.restartRequired	Required	To restart FEPLogging instance for applying any new configuration for example after certificate rotation	Yes
spec.fepLogging.scrapeInterval	Optional	Scrape interval for Prometheus to fetch metrics from FEPLogging instance	Yes

Custom Resource spec	Required/ Optional	Change Effect	Updating value allowed
spec.fepLogging.scrapeTimeout	Optional	Scrape Timeout for Prometheus to fetch metrics from FEPLogging instance	Yes
spec.fepLogging.elastic.host	Optional	Target Elasticsearch host name	Yes
spec.fepLogging.elastic.port	Optional	Target Elasticsearch port number	Yes
spec.fepLogging.elastic.authSecret.secretName	Optional	Secret name which contains Elasticsearch authentication username & password	Yes
spec.fepLogging.elastic.authSecret.userKey	Optional	Username key specified in Elasticsearch authentication secret	Yes
spec.fepLogging.elastic.authSecret.passwordKey	Optional	Password key specified in Elasticsearch authentication secret	Yes
spec.fepLogging.elastic.logstashPrefix	Optional	Logstash prefix to differentiate index pattern in elastic search. Default value is postgres	Yes
spec.fepLogging.elastic.auditLogstashPrefix	Optional	Logstash prefix to differentiate index pattern in elastic search for auditlog. If not specified, it will default to the same value as 'logstashPrefix'.	Yes
spec.fepLogging.elastic.scheme	Optional	Connection scheme between FEPLogging & Elasticsearch. Possible options http & https	Yes
spec.fepLogging.elastic.sslVerify	Optional	Set to true if you want to verify ssl certificate. If set to false then will not consider TLS certificate	Yes
spec.fepLogging.elastic.tls.certificateName	Optional	Kubernetes secret name which holds fluentd certificate	Yes
spec.fepLogging.elastic.tls.caName	Optional	Kubernetes configmap which holds cacert of Elasticsearch to verify Elasticsearch TLS connection	Yes
spec.fepLogging.tls.certificateName	Optional	Kubernetes secret name which holds Fluentd certificate	Yes
spec.fepLogging.tls.caName	Optional	Kubernetes configmap which holds cacert of Fluentd to configure MTLS between FEPLogging & Prometheus	Yes
spec.prometheus.tls.certificateName	Optional	Kubernetes secret name which holds Prometheus certificate	Yes
spec.prometheus.tls.caName	Optional	Kubernetes configmap which holds cacert of Fluentd to configure MTLS between FEPLogging & Prometheus	Yes

4.9.1.1.1 Define fepLogging image

The image property is used to specify other than default Fluentd image and its pullPolicy from FEPLogging CR.

If not specified it will use default image provided by Operator.

Example)

```
spec:
  fepLogging:
    image:
      image: 'quay.io/fujitsu/fujitsu-enterprise-postgres-fluentbit:ubi9-16-1.0'
      pullPolicy: IfNotPresent
```

4.9.1.1.2 Define fepLogging mcSpec

FEPLogging container Memory & CPU configuration can be provided by mcSpec properties.

Example)

```
spec:
  fepLogging:
    mcSpec:
      limits:
        cpu: 500m
        memory: 700Mi
      requests:
        cpu: 200m
        memory: 512Mi
```

4.9.1.1.3 Define fepLogging restartRequired

If FEPLogging required to be restarted to apply any new change, for example, after certificate rotation, FEPLogging container can be restarted by setting restartRequired flag as true. Default value of this flag is False. This flag will change back to false once the pod is restarted

Example)

```
spec:
  fepLogging:
    restartRequired: true
```

4.9.1.1.4 Define fepLogging scrapeInterval and scrapeTimeout

scrapeInterval and scrapeTimeout properties of FEPLogging are optional. These properties are used by Prometheus Servicemonitor to configure metrics fetching interval(scrapeInterval) and timeout of request.

Example)

```
spec:
  fepLogging:
    scrapeInterval: 30s
    scrapeTimeout: 30s
```

4.9.1.1.5 Define fepLogging elastic

To forward logs from FEPLogging(Fluentd) to Elasticsearch, need to configure elastic property. This is optional property. Elasticsearch server and certificates will be configured by user.

To configure log forwarding to Elasticsearch, the following properties are required.

- authSecret
- host
- port
- logstashPrefix
- auditLogstashPrefix
- scheme
- sslVerify
- tls(if sslVerify set to true)

Configure Elasticsearch server and use it's host name and port.

Here `tls` property is optional and works with `sslVerify` flag. To enable secure connection and `tls` verification set `sslVerify` true and provide valid `certificateName` & `caName`.

Elasticsearch `caName` is mandatory which holds CA cert of elastic search server.

Example)

```
spec:
  fepLogging:
    elastic:
      authSecret:
        passwordKey: password
        secretName: elastic-auth
        userKey: username
      host: elastic-passthrough.apps.openshift.com
      logstashPrefix: postgres
      auditLogstashPrefix: postgres
      port: 443
      scheme: https
      sslVerify: false
      tls:
        certificateName: fluentd-cert
        caName: elastic-cacert
```

4.9.1.1.6 Define authSecret for elastic

`authSecret` is the secret which contains username & password in base64 format for elastic search authentication

Example)

```
kind: Secret
apiVersion: v1
metadata:
  name: elastic-auth
  namespace: my-namespace
data:
  password: OFBobzlyRUJWOGg1Mk0xcXdaMUQ5bzQ0
  username: ZWxhc3RpYw==
type: Opaque
```

4.9.1.1.7 Define fepLogging TLS

FEPLogging has optional `TLS` property. If user wants to forward logs from FEPCluster to FEPLogging instance over a secure connection, the `TLS` configuration for FEPCluster(`remoteLogging` section) and the `TLS` configuration for FEPLogging and Prometheus are mandatory. Configuring `TLS` configuration on just `fepLogging` or Prometheus will not work.

When a self signed certificate is used, `caName` can be skipped.

Example)

```
spec:
  fepLogging:
    tls:
      certificateName: fluentd-cert
      caName: cacert
```

4.9.1.1.8 Define Prometheus TLS

If secured connection between FEPLogging and FEPCluster is required, then `TLS` configuration for FEPLogging and Prometheus are mandatory. Configuring `TLS` on just `fepLogging` or Prometheus will not work.

When a self signed certificate is used, caName can be skipped.

Example)

```
spec:
  fepLogging:
    ...
  prometheus:
    tls:
      certificateName: prometheus-cert
      caName: cacert
```

4.9.2 FEPCluster Configuration

This section describes how to enable logging in FEPCluster. FEP cluster provides a feature to forward logs to remote Fluentd(FEPLogging) and FEPLogging instance will forward the same logs to Elasticsearch(Optional) & Prometheus.

4.9.2.1 FEP Custom Resources - spec.fep.remoteLogging

The remoteLogging section needs to be added under fep to define required parameters for remoteLogging configuration.

Following is a sample template:

```
spec:
  fep
  ...
  remoteLogging:
    enable: true
    fluentdName: new-fep-logging
    tls:
      certificateName: fluentbit-cert
      caName: cacert
  ...
```

Below is the list of all parameters defined in the remoteLogging section, along with their brief description:

Custom Resource spec	Required/ Optional	Change Effect	Updating value allowed
remoteLogging.enable	Required	The 'enable' is set to true for enabling Logging feature	No
remoteLogging.fluentdName	Required	The 'fluentdName' is the name of the FEPLogging CR where logs will be forwarded	Yes
remoteLogging.tls.certificateName	Optional	Secret name which contains MTLS certs of fluentbit	No
remoteLogging.tls.caName	Optional	Cacert of Fluentd for ssl verification	No
remoteLogging.image	Optional	Fluentbit image for remoteLogging	Yes
remoteLogging.pullPolicy	Optional	Fluentbit image pull policy	Yes
remoteLogging.mcSpec.limits.cpu	Optional	CPU allocation limit for fluentbit	Yes
remoteLogging.mcSpec.limits.memory	Optional	Memory allocation limit for fluentbit	Yes
remoteLogging.mcSpec.requests.cpu	Optional	CPU allocation request for fluentbit	Yes

Custom Resource spec	Required/ Optional	Change Effect	Updating value allowed
remoteLogging.mcSpec.requests.memory	Optional	Memory allocation request for fluentbit	Yes
remoteLogging.fluentbitParameters.memBufLimit	Optional	Defines the Mem_Buf_Limit in Fluentbit. This will affect all sections that use this parameter	Yes

4.9.2.1.1 Define remoteLogging enable and fluentdName

The enable flag is used to describe that FEPCluster will enable log monitoring feature if set as true.

If enable flag set as true then fluentdName is the mandatory field. It will describe the FEPLogging CR name to which FEPCluster will forwards the logs.

If the enable flag is set as false, the FEPCluster will not enable logging feature.

Example)

```
fep:
  remoteLogging:
    enable: true
    fluentdName: new-fep-logging
```

If user wants to update existing FEPCluster with log monitoring feature then FEPCluster log_destination configuration must be set as **csvlogs**. For new cluster it will be already set.

Example)

```
fep:
  ...
  remoteLogging:
    enable: true
    fluentdName: new-fep-logging
  ...

  fepChildCrVal:
    customPgParams:
      ...
      log_destination = csvlog
      ...
```

4.9.2.1.2 Define remoteLogging tls

When FEPCluster uses secure connection for remoteLogging, then TLS section is mandatory.

In the TLS section, provide the secret name that contains certificate and private key that is used for ssl verification.

For MTLS connection caName is required to mutually validate certificate.

Example)

```
fep:
  remoteLogging:
    enable: true
    fluentdName: new-fep-logging
  tls:
    certificateName: fluentbit-cert-secret
    caName: ca-cert
```



Note

The Elasticsearch server is configured by user and it is NOT part of FEPLlogging deployment by operator.

4.9.2.1.3 Define remoteLogging image

The image property is used to specify other than default Fluentbit image and it's pullPolicy.

If not specified it will use default image provided by Operator.

Example)

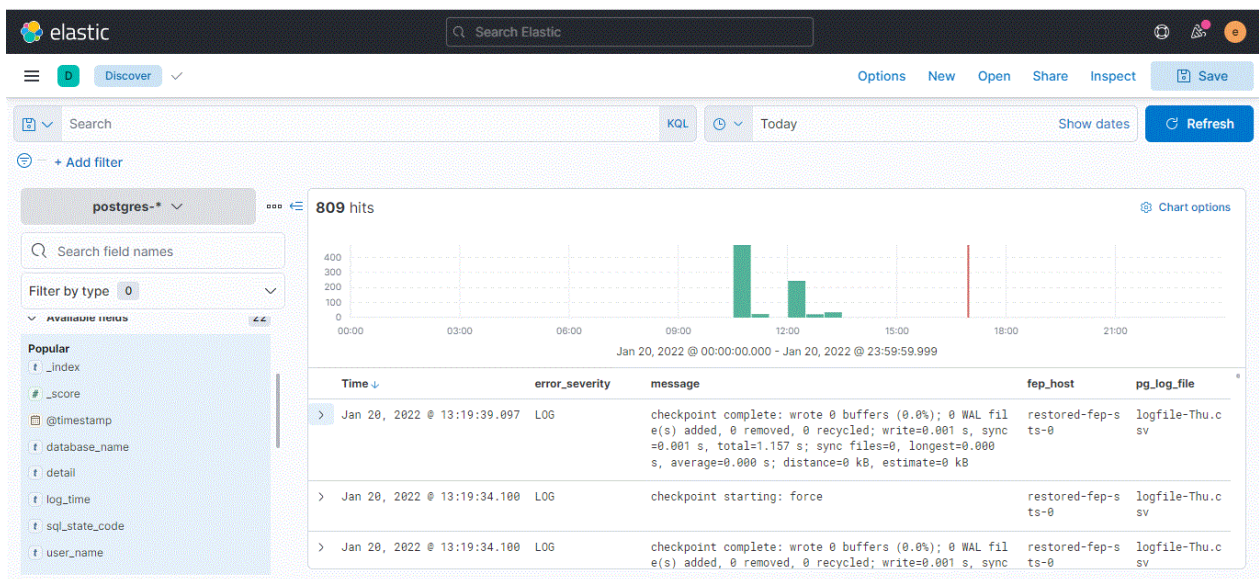
```
spec:
  fep:
    remoteLogging:
      image: 'quay.io/fujitsu/fujitsu-enterprise-postgres-fluentbit:ubi9-16-1.0'
      pullPolicy: IfNotPresent
```

4.9.3 FEPLlogging Operations

4.9.3.1 Log Forwarding to Elasticsearch

If the user has provided Elasticsearch configuration in the FEPLlogging CR, and FEPCluster is configured to send server log files and auditlog files to that FEPLlogging instance, those logs will be visible on Elasticsearch stack or Elastic Cloud. Assuming Elasticsearch has been configured with Kibana then logs will be visible in Kibana Dashbord. User can use fep log csv fields to create various Dashbord in Kiabana as well. LogstashPrefix and auditLogstashPrefix will be used to filter logs of specific FEPLlogging instance.

User can verify if FEPLlogging feature is configured properly or not by checking real time FEP logs are populating to the destination.



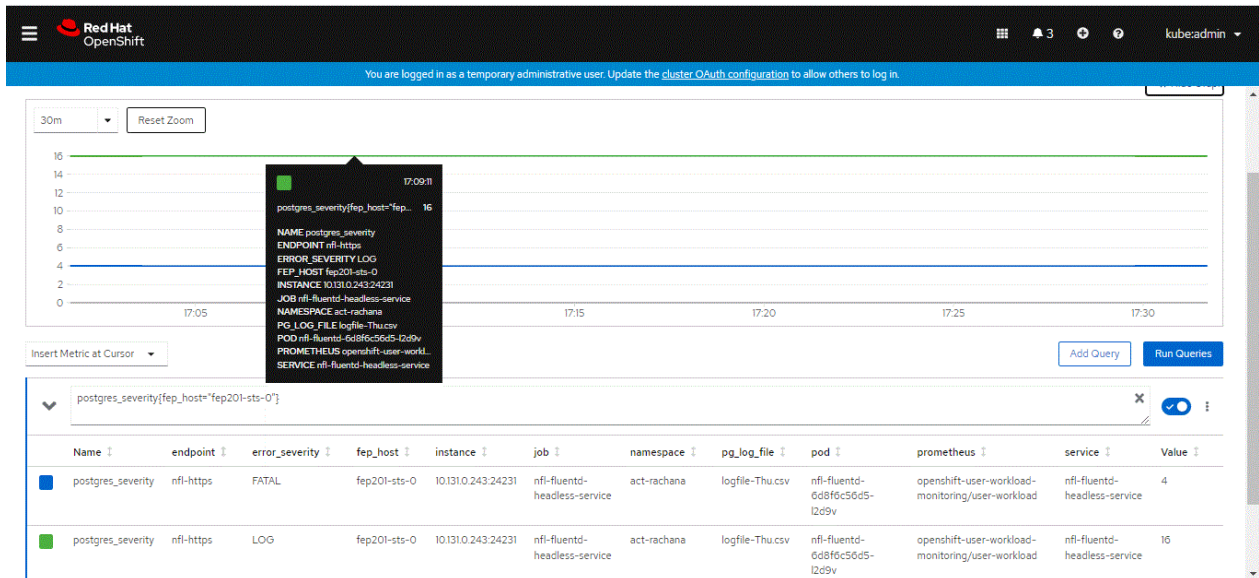
4.9.3.2 Log severity based Alarms/Metrics

FEPLlogging feature is used for raising alarm/alert based on postgres severity counts as well. While user creates FEPLlogging CR, Operator will forward real time counts of various postgres seviry metrics to Openshift managed Prometheus. Openshift managed Alertmanager can access this metrics counters and user can use them to create alerts/alarms. There are 4 default alert rules already created as part of FEPLlogging implementation as listed below:

- FEPLlogErrorMessage
- FEPLlogFatalMessage
- FEPLlogPanicMessage

- FEPLogWarningMessage

Prometheus will scrape postgres_severity counter at every 30s as default scrape interval is 30s. User can modify this scrape interval from FEPLogging CR. After each scrape interval, if any change/increment found in postgres_severity counter then alert rule will be fired. User can check counts of postgres_severity metrics anytime from Prometheus dashboard as well.



4.9.3.3 Forwarding auditlog to Elasticsearch

In order to forward auditlog to Elasticsearch, update the FEPCluster to enable creating auditlog.

Example)

```
spec:
  fep:
    fepChildCrVal:
      customPgAudit: |
        [output]
        logger = 'auditlog'
        log_directory = '/database/log/audit'
      customPgParams: |
        shared_preload_libraries='...,pgaudit'
        session_preload_libraries='...,pgaudit'
```



Information

Refer to "4.11 Automating Audit Log Operations" for information on enabling audit logs. You can also enable audit logging by configuring the spec.fep.pgAuditLog.enable parameter.

4.9.4 Limitations

- Only postgres_severity including ERROR, PANIC, FATAL and WARNING are monitored.
- External fluentd can not be used for log monitoring and log forwarding.
- External Elasticsearch is required for log forwarding.
- User must decide at deployment time whether secured connection between FEPCluster and FEPLogging is required or not. After deployment, one can switch connection from insecure to secure but can not switch from secure to insecure connection.
- User must configure FEPLogging CR first then only FEPCluster can forward logs to particular FEPLogging otherwise Logging feature will not work.

- User must set log_destination in FEPCluster CR.

4.10 Configuring pgBadger

This section describes how to configure pgBadger. FEP cluster provides a feature to create pgbadger report on defined schedule and upload the report to a web server outside.

4.10.1 FEP Custom Resources - spec.fep.pgBadger

Custom Resource spec	Change Effect
pgBadger.schedules.create	The 'create' schedule to create report and upload it to endpoint
pgBadger.schedules.cleanup	The 'cleanup' schedule to delete the report left in container
pgBadger.options.incremental	Default: false; When set to True: create incremental report in pgbadger
pgBadger.endpoint.authentication	a secret to contain authentication info to access endpoint support basic auth only
pgBadger.endpoint.customCertificateName	Client certificate reference in customCertificate CR
pgBadger.endpoint.fileUploadParameter	The file upload parameter defined by the web server Default: 'file'
pgBadger.endpoint.insecure	equivalent to curl -insecure option, default to false
pgBadger.endpoint.url	Web server url to upload the report file

4.10.2 Define pgBadger Schedules

The schedules are used to create and run a job periodically, written in Cron format.

If the schedule format is invalid, the cronjob will not be created, so no pgBadger report will be created and uploaded.

Example)

```
pgBadger :
  schedules:
    cleanup: '10 * * * *'
    create: '50 * * * *'
```

4.10.3 Define pgBadger Options

When the incremental option is set to false, pgBadger will create normal html report and upload the html file to the web server.

When the incremental option is set to true, pgBadger will create incremental report and upload a zip file to the web server.

Example)

```
pgBadger :
  options:
    incremental: true
```

4.10.4 Define Endpoint for Uploading Report

Web server url

Both http and https are supported.

Example)

```
pgBadger :
endpoint :
  url: 'https://webserver-svc:4443/cgi-bin/upload.php'
```

Web Server authentication

Only basic auth is supported

To configure web server authentication:

Create a base64 encoded text from username:password

Example)

```
$ echo -ne "myuser:mypass" | base64
```

```
amFzb2530mphc29udw==
```

Wrap the output with base64 for creating a secret

Example)

```
$ echo -ne "amFzb2530mphc29udw==" | base64
```

```
YW1GemIyNTNPbXB0YzI5dWR3PT0=
```

Create a secret by using the wrapped text. The key must be 'basic_auth'.

Example)

```
kind: Secret
apiVersion: v1
metadata:
  name: pgbadger-endpoint-auth
  namespace: fep-container-ct
data:
  basic_auth: YW1GemIyNTNPbXB0YzI5dWR3PT0=
type: Opaque
```

Add the secret name in the endpoint definition.

Example)

```
pgBadger :
endpoint :
  authentication: pgbadger-endpoint-auth
```

Web Server certificates

When certificate files are required by the web server, FEP cluster provides customCertificate CR to mount the certificates files in container.

To use certificates for web server.

Create a secret based on the cert and key files.

Example)

```
oc create secret tls webserver-cert --cert=webserver.pem --key=webserver.key
```

The webserver.pem and webserver.key are certificate files for accessing web server

Create a configmap based on the CA cert.

Example)

```
oc create configmap webserver-cacert --from-file=ca.crt=webca.pem
```

The webca.pem is the CA certificate file for accessing web server.

Define custom certificates in FEPCluster CR.

Example)

```
spec:
  fepChildCrVal:
    customCertificates:
      - userName: pgbadger-custom
        certificateName: webserver-cert
        caName: webserver-cacert
```

The userName is a reference in the pgBadger endpoint.

The certificateName is the secret created above.

The caName is the configmap created above.

Refer the custom certificate name in pgbadger endpoint.

Example)

```
pgBadger:
  endpoint:
    customCertificateName: pgbadger-custom
```

Insecure access to web server

The pgbadger CR provides an option to the web server endpoint when secure connection is not required:

Example)

```
pgBadger:
  endpoint:
    insecure: true
```

File upload parameter

This parameter specify the request parameter for uploading a file to a web server. The value of this parameter is depended on the web server implementation.

Example)

```
pgBadger:
  endpoint:
    fileUploadParameter: uploadfile
```

curl command and parameters

FEP cluster uses curl command to upload the generated report to a web server endpoint. The CR in enpoint section will be converted to curl command parameters. The following table shows the mapping:

curl command parameter	User configuration
[URL]	Endpoint url
--cert	webserver.pem included in the secret referred in customCertificateName

curl command parameter	User configuration
--key	webserver.key included in the secret referred in customCertificateName
--cacert	webca.pem included in the configmap referred in customCertificateName
--form "uploadfile=@/path/to/report"	Endpoint fileUploadParameter
--header "Authorization: Basic passxxx"	Endpoint authentication configmap
--insecure	When endpoint.insecure is set to true

4.10.5 Uploaded File on Web Server

The FEP cluster uploads the pgbadger report according to the incremental mode:

incremental mode	Uploaded file name	Example
True	[fep cluster name]-sts-[pod index].zip	pgbadger-test3-sts-0.zip pgbadger-test3-sts-1.zip
False	[fep cluster name]-sts-[pod index].html	pgbadger-test3-sts-0.html pgbadger-test3-sts-1.html

The zip file contains a folder of pgbadger incremental report.

Example)

```
\database
  \log
    \pgbadger-report
      \[years]
        \[months]
          \[weeks]
```

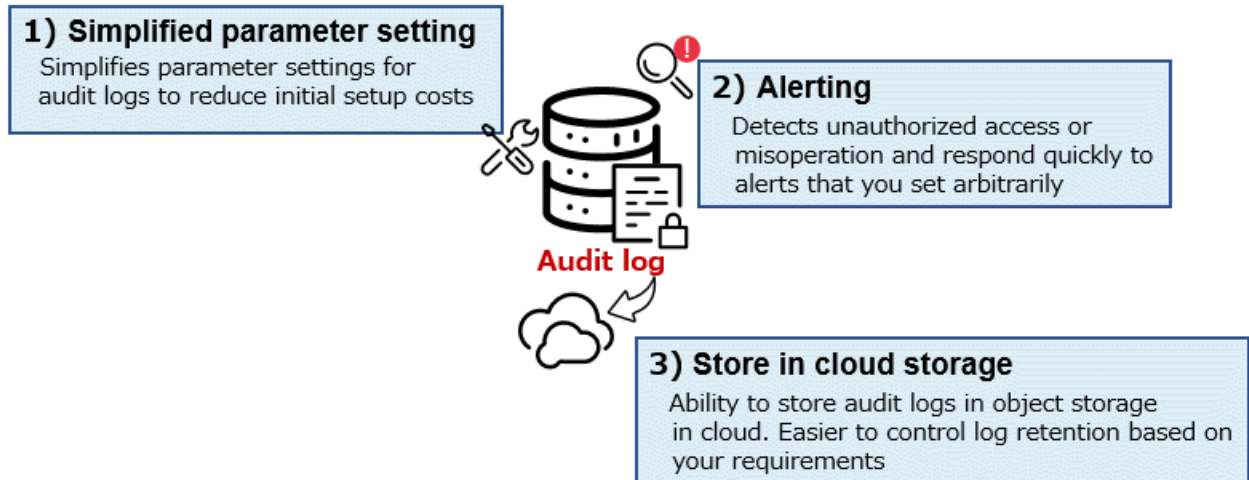


Note

- The web server is NOT included in the FEP cluster solution.
- The web server is responsible to the uploaded files according to the customer's business logic.

4.11 Automating Audit Log Operations

Simplifies the operation of your audit logs to implement operations that meet security requirements such as audits.



4.11.1 Simplifies Parameter Setting

Simplify audit log parameter settings and reduce initial setup costs. The only setting required is enabling the enable parameter. The pgaudit module is loaded when the FEP server starts and audit logs are stored in the logs directory.

By customizing the audit log configuration file as necessary, it is also possible to make settings according to operational requirements.

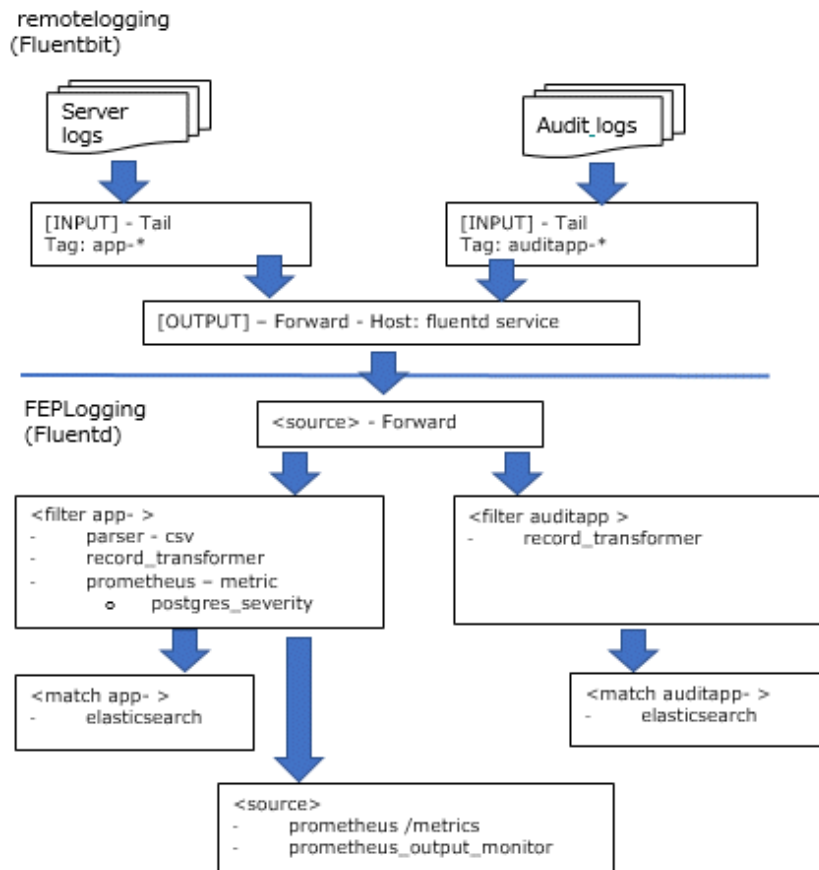
```
spec:
  fep:
    pgAuditLog:
      enable: true
```

4.11.2 Alerting

According to preset alert conditions, unauthorized access and erroneous operations can be detected at an early stage, enabling rapid response.

Audit logs are sent to Fluentd using the remotelogging function, and audit logs are monitored by setting alerts in Prometheus according to sqlstate conditions.

The alert is triggered when the 1 minute average of sqlstate(28P01) (invalid password) exceeds 50.



4.11.3 Store in Cloud Storage

Long-term retention of audit logs may be required in accordance with system or industry standard security policy requirements. However, long-term storage of logs requires the continuation of complicated operations such as disk management and rotation management.

Therefore, with this function, by saving audit logs to cloud object storage, you can easily control the saving of logs based on your requirements.

```

spec:
  fep:
    pgAuditLog:
      enable: true
      endpoint:
        protocol: s3
        url: s3://pgaudit/cluster1
        authentication: s3-secret
      schedules:
        upload: '30 * * * *'
  
```

4.12 Transparent Data Encryption Using a Key Management System

Describes how to configure transparent data encryption using a key management system.

Transparent data encryption using a key management system can only be configured when the FEPCluster is first created. Users cannot configure an existing FEPCluster for transparent data encryption using a key management system.

4.12.1 Registration of Authentication Information

4.12.1.1 When Using a KMIP Server

Save the certificate used for TLS communication between KMIP server in Secret or ConfigMap.

The Secret or ConfigMap you created gives the FEPCluster custom resource a resource name and mounts it in the FEP container.

Create a Secret to store the client certificate and private key for connecting to KMIP server.

Also, optionally create a ConfigMap to store the root certificate.

An example of registering credentials using the credentials file below is explained.

```
kmip.pem    # Client certificate for connecting to KMIP server
kmip.key    # Private key
myca.pem    # Root certificate
```

Create a Secret to store the client certificate and private key.

Specify tls.crt and tls.key as file names when mounting the client certificate and private key, respectively.

```
$ oc create secret generic kmip-cert --from-file=tls.crt=kmip.pem --from-file=tls.key=kmip.key -n kmip-demo
```

Optionally create a ConfigMap to store your root certificates.

Specify ca.crt as the file name to be mounted.

```
$ oc create configmap kmip-cacert --from-file=ca.crt=myca.pem -n my-namespace
```

4.12.1.2 When Using AWS Key Management Service

Save credentials and other settings required to connect to AWS key management services in Secrets and ConfigMaps.

Prepare two files, credentials and config, which describe credentials and other settings according to the format specified by the AWS client interface. Specifying access_key_id and secret_access_key in the credentials file is mandatory.

An example of registering authentication information using the following configuration file is explained.

```
credentials  # credentials file
config       # config file
```

Create a ConfigMap to store config files. Specify config for the key name. The name of the ConfigMap is arbitrary (here aws-kms-config).

```
$ oc create configmap aws-kms-config --from-file=config=config -n my-namespace
```

Create a secret to save the credentials file. Specify credentials for the key name. The name of the Secret is arbitrary (here aws-kms-credentials).

```
$ oc create secret generic aws-kms-credentials --from-file=credentials=credentials -n my-namespace
```



See

Refer to below for AWS client interface configuration files.

<https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html>

4.12.1.3 When using Azure Key Management Service

Save the credentials required to connect to Azure's key management service in Secret.

The available authentication methods are either authentication using passwords or authentication using client certificates.

For password-based authentication, create a YAML format file that defines a secret like the one below. The secret name is arbitrary (here azure-key-vault-passphrase). data.clientsecret contains a base64-encoded password.

```
kind: Secret
apiVersion: v1
metadata:
  name: azure-key-vault-passphrase
  namespace: my-namespace
data:
  clientsecret: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX=
type: Opaque
```

Create a secret based on the created YAML file. Here we are using a YAML file named azure-client-secret.yaml.

```
$ kubectl apply -f azure-client-secret.yaml -n my-namespace
```

For authentication using a client certificate, store the client certificate file and private key in Secret.

Here is an example of creating a Secret using the certificate file below.

```
azuremycert.pem    # PEM file containing client certificate and private key
```

Create a secret to store the client certificate. Specify azure-key-vault.crt for the key name. The secret name is arbitrary (here azure-key-vault-secret).

```
$ oc create secret generic azure-key-vault-secret --from-file=azure-key-vault.crt=azuremycert.pem -n my-namespace
```

4.12.2 Configuring FEPCluster Custom Resources

To enable TDE using a key management system, you need to set “spec.fepChildCrVal.customPgParams” and “spec.fepChildCrVal.sysTde”.

4.12.2.1 Define spec.fepChildCrVal.customPgParams

The fepChildCrVal.customPgParams section must define the following parameters:

shared_preload_libraries

Add the 'tde_kms' library to the list of libraries in shared_preload_libraries.

Example)

```
spec:
  fep:
    ...
    fepChildCrVal:
      ...
      customPgParams:
        shared_preload_libraries='pgx_datamasking,pg_prewarm,pg_stat_statements,tde_kms'
```

Do not remove 'tde_kms' library from 'shared_preload_libraries' list after cluster creation.

4.12.2.2 Define spec.fepChildCrVal.sysTde

Add a sysTde section under spec.fepChildCrVal to define the parameters required to connect to your key management system. Under sysTde there are two parameters defined:

- tdeType
- tdek

Define spec.fepChildCrVal.sysTde.tdeType

sysTde itself is an optional parameter (if sysTde is not defined, use a file-based keystore). However, if sysTde is defined by the user, sysTde.tdeType must also be defined.

If configuring TDE with a key management system, set sysTde.tdeType to "tdek".

Example)

```
sysTde:
  tdeType: tdek
```

Define spec.fepChildCrVal.sysTde.tdek.kmsDefinition

If you set sysTde.tdeType to "tdek", you must also define sysTde.tdek.

Define the connection information of the key management system in sysTde.tdek.kmsDefinition. Based on the information defined here, the operator creates the key management system connection information file used by Fujitsu Enterprise Postgres.

Information for multiple key management systems can be defined in kmsDefinition. For type, specify the type of key management system (either kmip, awskms, or azurekeyvault).

Example)

```
sysTde:
  tdeType: tdek
  tdek:
    targetKmsName: kms_conninfo1
    kmsDefinition:
      - name: kms_conninfo1
        type: kmip
  ...
```

Refer to the Reference for details of each parameter.

Specify the name of the Secret or ConfigMap created in "[4.12.1 Registration of Authentication Information](#)" in the corresponding parameter under kmsDefinition. If type is awskms, profile specifies the name of the profile to use from the profile in the AWS client interface configuration file.

Example)

```
spec:
  fep:
    ...
  fepChildCrVal:
    ...
  sysTde:
    tdeType: tdek
    tdek:
      targetKmsName: kms_conninfo1
      targetKeyId: xxxyyyzzz
      kmsDefinition:
        - name: kms_conninfo1
          type: kmip
          address: xxx.xxx.xxx.xxx
          port: 1000
          authMethod: cert
          sslpassphrase: ssl-password
          cert:
            certificateName: kmip-cert
```

```
caName: kmip-cacert
sslcrName: kmip-crl
```

Define `spec.fepChildCrVal.sysTde.tdek.targetKeyId`, `spec.fepChildCrVal.sysTde.tdek.targetKmsName`

Specify one of the key management system names defined in `kmsDefinition` in `sysTde.tdek.targetKmsName` as the name of the key management system to use as the keystore. `sysTde.tdek.targetKeyId` specifies the key ID of the encryption key within that key management system to use as the master encryption key.

4.13 Disaster Recovery in Hot Standby Configuration

By implementing disaster recovery in a hot standby configuration, business systems can be restored more quickly in the event of a disaster. This function has the following two methods.

Continuous recovery method

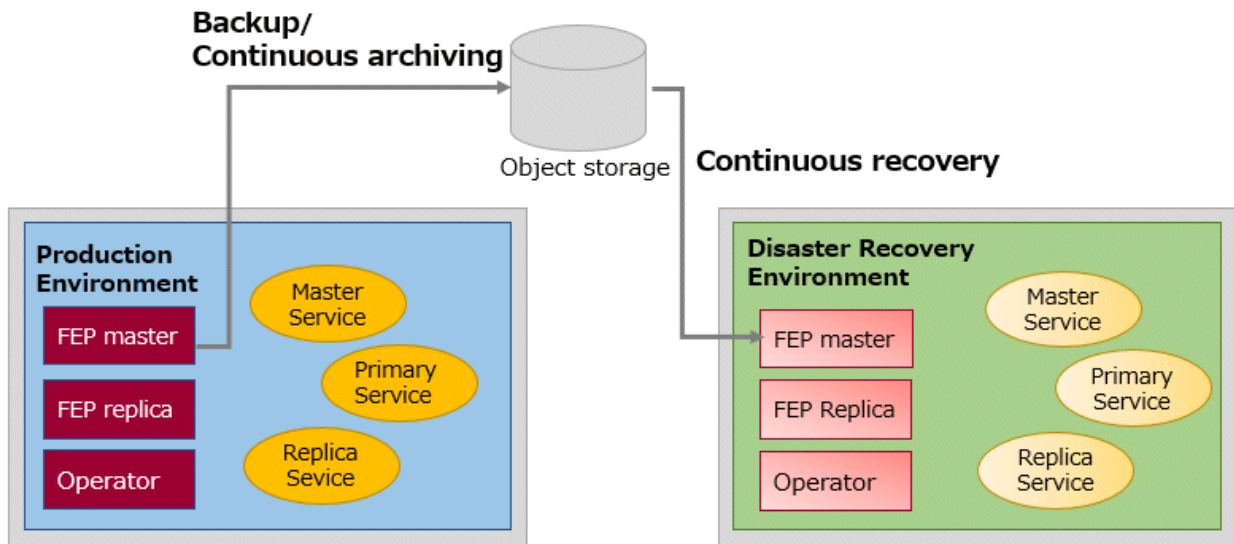
Create a container environment for production and another one for disaster recovery. Store the production environment data in object storage, and continuously restore to the disaster recovery environment. This method enables quick recovery compared to the backup/restore method, but RPO (Recovery Point Objective) increases depending on the timing of regular backups.

Streaming replication method

Similarly to the continuous recovery method, create a container environment for production and another one for disaster recovery. Use streaming replication methods to synchronize data to the disaster recovery environment. This method enables faster recovery compared to the backup/restore method, lower RPO compared to the continuous recovery method, and real-time data synchronization. However, because network settings for the streaming replication method are required, the management cost is high, and there is a slight impact on the performance of the database in the production environment.

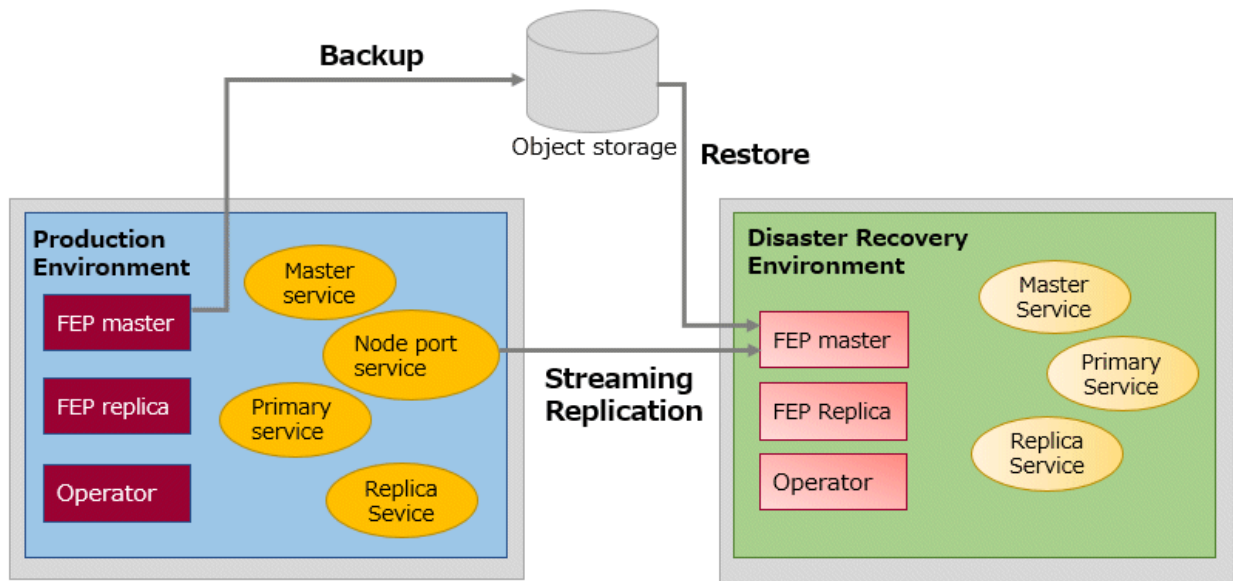
4.13.1 Continuous Recovery Method

The continuous recovery method uses object storage to synchronize production and disaster recovery environments. Specify object storage that is located in a region that you consider safe for the range of possible disasters.



4.13.2 Streaming Replication Method

The streaming replication method achieves direct data synchronization between the database in the production environment and the database in the disaster recovery environment.



4.13.3 Defining a Hot Standby Configuration

This section describes the deployment procedures for the continuous recovery method and the streaming replication method of the hot standby configuration.

4.13.3.1 Defining a Continuous Recovery Method

Custom resource definitions for FEPCluster in the production environment do not have parameters that are only used in hot standby configurations. When using the continuous recovery method, define the FEPCluster custom resource in the disaster recovery environment as follows.

```
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  ...
spec:
  fep:
    standby:
      enable: true
      method: archive-recovery
      pgBackrestConf: |
        [global]
        log-path=/database/log/backup
        repo1-type=azure
        repo1-path=< Backup path of primary/ cluster from which data is to be restored>
        repo1-azure-account=<my storage account>
        repo1-azure-container=fepbackups
        repo1-azure-key=<my storage account key >
      ...
```

4.13.3.2 Defining a Streaming Replication Method

When using the streaming replication method, define as follows.

```
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  ...
spec:
  fep:
```

```

standby:
  enable: true
  method: streaming
  streaming:
    host: <LoadBalancer IP>
    port: 27500
  pgBackrestConf: |
    [global]
    log-path=/database/log/backup
    repo1-type=azure
    repo1-path=< Backup path of primary/ cluster from which data is to be restored>
    repo1-azure-account=<my storage account>
    repo1-azure-container=fepbackups
    repo1-azure-key=<my storage account key >
...

```

For streaming replication, FEPClusterCR is defined as above, but a separate LoadBalancer must be deployed.

```

kind: Service
apiVersion: v1
metadata:
  name: my-fep-internal-svc
  namespace: sample-namespace
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: 'true'
spec:
  ports:
    - protocol: TCP
      port: 27500
  type: LoadBalancer
  selector:
    app: <my-fep-cluster>-sts
    feprole: master

```

4.13.3.3 Defining FEPCluster Custom Resources

The parameters of FEPClusterCR in the disaster recovery environment that are required to realize a hot standby configuration are shown below. For parameter details, refer to "FEPCluster Parameters" in the Reference.

- spec.fep.standby.enable
- spec.fep.standby.method
- spec.fep.standby.pgBackrestConf
- spec.fep.standby.streaming.host
- spec.fep.standby.streaming.port

4.14 Enabling Client Authentication with scram-sha-256 Authentication

This section describes step-by-step instructions on how to enable client authentication using scram-sha-256 authentication.

4.14.1 Enabling Client Authentication Using scram-sha-256 Authentication in the FEP Server Container

To enable scram-sha-256 authentication on a newly built FEP server container, deploy a FEPCluster custom resource with the following settings.

4.14.1.1 Define spec.fepChildCrVal.customPgParams

Specify scram-sha-256 for password_encryption.

```
password_encryption = 'scram-sha-256'
```

You must restart the database in the FEP server container for the configuration to take effect.

4.14.1.2 Define spec.fepChildCrVal.customPgHba

Add an entry to allow scram-sha-256 authentication.

To allow scram-sha-256 authentication, specify scram-sha-256 in the METHOD.

You must restart the database in the FEP server container for the configuration to take effect.

4.14.2 Enabling Client Authentication Using scram-sha-256 Authentication in the FEPPgpool2 Container

This section describes how to enable scram-sha-256 authentication on a newly built FEPPgpool2 container.

4.14.2.1 Creating the Resources Required to Enable scram-sha-256 Authentication

Create the resources required to enable scram-sha-256 authentication.

1. Create secret for encryption key

Create an encryption key secret (pgpoolkeySecret) containing the password to use for scram-sha-256 authentication.

The key is pgpoolkey and the value is a base64 encrypted version of the password to be used for encryption.

```
apiVersion: v1
kind: Secret
metadata:
  name: scrampgpoolkey-secret
type: Opaque
data:
  pgpoolkey: cGdwb29sa2V5cGFzc3dvcmQ=
```

After creating the secret, write the name of the secret you created in the FEPPgpool2 custom resource. This mounts the created secret in the FEPPgpool2 container as a file in the volume and passes the password used for encryption to the FEPPgpool2 container.

If you use scram-sha-256 authentication and do not create a secret, the operator automatically creates the following secret based on the information in the FEPPgpool2 custom resource.

```
apiVersion: v1
kind: Secret
metadata:
  name: "{{spec.name}}-feppgpool2-pgpoolkey"
type: Opaque
data:
  pgpoolkey: K1kx0VZxKzRrdWlUT3A2UHNQMzZlcUJ0U0Z2UUoxUklnMms2cktIY1NkekFOemZBYkhjZDFadG5VR3ZtTVR6Uw==
```

Value is set to the value of a password randomly generated by the operator and encrypted with base64. The secret name is "{{spec.name}}-feppgpool2-pgpoolkey".

Information

scram-sha-256 encryption uses an encryption key. This encryption key is used by the FEPPgpool2 container to decrypt the AES-encrypted password if it was stored in the pool_passwd file. Therefore, the encryption and decryption keys must be the same.

2. Create a secret for database user information

The user/password used for client authentication in the FEPPgpool2 container must be the same as the database user used in the database in the FEP server container.

Create a database user information secret (userinfoSecret) in the FEPPgpool2 container to inform the database user username and password information in the FEP server container.

The user name and password should describe the secret in the following format.

```
apiVersion: v1
kind: Secret
metadata:
  name: scramuserinfo-secret #Specify any name
type: Opaque
data:
  user1: dXNlcjFwYXNzd2Q=
  user2: dXNlcjJwYXNzd2Q=
```

For database users that Operator creates automatically, you do not need to create a secret because Operator retrieves the database username and password information from the FEPCluster custom resource and the FEP server container.

If you use scram-sha-256 authentication and do not create a secret, the operator automatically creates the following secret:

```
apiVersion: v1
kind: Secret
metadata:
  name: "{{spec.name}}-feppgpool2-userinfo"
type: Opaque
data:
```

The secret name is "{{spec.name}}-feppgpool2-userinfo".

4.14.2.2 Editing FEPPgpool2 Custom Resources

To enable scram-sha-256 authentication, edit and deploy a FEPPgpool2 custom resource with the following settings:

1. Set secret information

Specify scram for spec.clientAuthMethod in the FEPPgpool2 custom resource.

In addition, in the secret for the encryption key (spec.scram.pgpoolkeySecret), specify the secret name that you created in step 1 of ["4.14.2.1 Creating the Resources Required to Enable scram-sha-256 Authentication"](#). Similarly, in the database user information secret (spec.scram.userinfoSecret), specify the secret name that you created in step 2 of ["4.14.2.1 Creating the Resources Required to Enable scram-sha-256 Authentication"](#).

```
spec:
  clientAuthMethod: scram
  scram:
    pgpoolkeySecret: scrampgpoolkey-secret
    userinfoSecret: scramuserinfo-secret
```

This mounts the created secret in the FEPPgpool2 container as a file in the volume and notifies the FEPPgpool2 container of the username and password information.

2. Edit spec.customhba

Edit the spec.customhba field in the FEPPgpool2 custom resource to add an entry for scram-sha-256 authentication.

3. Edit pgpool.conf

You must edit the spec.customparams field in the FEPPgpool2 custom resource to edit the parameters related to the authentication settings.

```
enable_pool_hba=true
```

4.14.3 Enabling Client Authentication Using scram-sha-256 Authentication on Existing FEP Server and FEPpgpool2 Containers

This section describes how to enable client authentication using scram-sha-256 authentication on existing FEP server and FEPpgpool2 containers

For information about FEP server containers, refer to "[4.14.1 Enabling Client Authentication Using scram-sha-256 Authentication in the FEP Server Container](#)". At this time, if the database user created by the user already has a password encrypted with md5, change the password to one encrypted with scram-sha-256. Therefore, reset the password.

For information about the FEPpgpool2 container, refer to "[4.14.2 Enabling Client Authentication Using scram-sha-256 Authentication in the FEPpgpool2 Container](#)". If an existing FEPpgpool2 custom resource is deployed, delete the FEPpgpool2 custom resource. You do not need to delete the FEPpgpool2 custom resource if you want to change the contents of the secret specified in the FEPpgpool2 custom resource or the secret created automatically by the operator.

4.15 Backing Up Statistics

In case the query plan becomes unstable, you can periodically backup the statistics using `pg_dbms_stats`. Set `spec.fep.backupStats.enable` to true and set the backup to `spec.fep.backupStats.scheduleN` (N is an integer) in the FEPCluster custom resource. Note that `spec.fep.backupStats.enable` defaults to true.

The following are the configurable items.

- Schedule
- Whether to execute the schedule
- What to back up
- Retention
- Comment

The example below is a setting to take a backup of mydb every day at 20:00, and at the same time delete the backup from 5 days ago.

Example) Definition example of FEPCluster custom resource

```
spec:
  fep:
    backupStats:
      enable: true
      schedule1:
        enable: true
        backupSchedule: "0 20 * * *"
        targetDb: mydb
        retention: 5
        comment: "mydbBackup"
```

Additionally, if there is no backup definition in `spec.fep.backupStats.schedule1` when building FEPCluster for the first time, the backup definition will be defined in `spec.fep.backupStats.schedule1` of the FEPCluster custom resource with the following settings by default.

Schedule: 23:30 every day

Target: All databases

Retention: Statistics backed up in that database older than 7 days will be deleted.

Comment: "FepDefaultBackup"

```
spec:
  fep:
    backupStats:
      enable: true
      schedule1:
        enable: true
        backupSchedule: "30 23 * * *"
```

```
retention: 7
comment: "FepDefaultBackup"
```

If you do not want to perform scheduled backups, set `spec.fep.backupStats.scheduleN.enable` to `false`. Also, if you do not want to perform all backups, set `spec.fep.backupStats.enable` to `false`.

Chapter 5 Post-Deployment Operations

This chapter describes the operation after deploying the container.

5.1 How to Connect to a FEP Cluster

When connecting from within the same project of the OpenShift system

Service resources are used to connect to FEPCluster and FEPPgpool2 from within the same project.

A service resource provides a single endpoint for communicating with containers.

Service resources are created with the following naming conventions.

FEPCluster service

- <FEPCluster name>-primary-svc
- <FEPCluster name>-replica-svc
- <FEPCluster name>-headless-svc

FEPPGPool2 service

- <FEPPgpool2 name>-feppgpool2-svc

Example of checking service resources of FEPCluster container and FEPPgpool2 container

```
$ oc get all
```

Check where the resource type is Service (Begin with "svc").

You can also check with the oc get svc command. The following is an example.

```
$ oc get svc
NAME                                TYPE          CLUSTER-IP      EXTERNAL-IP  PORT(S)                                AGE
<FEPCluster name>-headless-svc    ClusterIP      None             <none>       27500/TCP, 25001/TCP                  24h
<FEPCluster name>-primary-svc     ClusterIP      xxx.xxx.xxx.xxx  <none>       27500/TCP, 25001/TCP                  24h
<FEPCluster name>-replica-svc     ClusterIP      yyy.yyy.yyy.yyy <none>       27500/TCP, 25001/TCP                  24h
<FEPPgpool2 name>-feppgpool2-svc NodePort       zzz.zzz.zzz.zzz <none>       9999:31707/TCP, 9998:31906/TCP        24h
```

Example of accessing FEPPgpool2 container

```
$ psql -h <FEPPgpool2 name>-feppgpool2-svc -p 9999 -c "select version();"

```

When connecting from outside the OpenShift system

Automatically creating a service with ClusterIP to connect to the deployed container. You can connect to FEP or FEP pgpool2 services from the OpenShift system's internal network. To access from outside the OpenShift system, you need to know the address of the OpenShift node.

For example, "Access the FEP pgpool2 container from an application server that is running outside the OpenShift system but is part of the Internal network".

An example of how to check the node IP in OpenShift.

```
$ oc get nodes
NAME                                STATUS    ROLES    AGE    VERSION
openshiftcluster1-cmfv8-master-0    Ready     master   370d   v1.19.0+4c3480d
openshiftcluster1-cmfv8-master-1    Ready     master   370d   v1.19.0+4c3480d
openshiftcluster1-cmfv8-master-2    Ready     master   370d   v1.19.0+4c3480d
$ oc describe nodes openshiftcluster1-cmfv8-master-0 | grep IP
InternalIP: 10.0.2.8
```

An example of verifying the service resource for the FEP pgpool2 container.

```
$ oc get all
```

Check where the resource type is Service (Begin with "svc/").

You can also see this with the oc get svc command. The following is an example.

```
$ oc get svc
NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)                                AGE
svc-feppgpool2-feppgpool2  NodePort  172.30.248.12 <none>       9999: 30537/TCP, 9998: 30489/TCP  2m5s
```

This is an example of accessing the FEP pgpool2 container.

```
$psql -h 10.0.2.8 -p 30537 -c "show pool_nodes"
```

5.2 Configuration Change

This section describes changes to the FEPCluster configuration.

List FEPCluster

Equivalent Kubernetes command: `kubectl get FEPClusters (-A)`

This operation will list all FEPClusters in a namespace, or if the -A option is specified, will list all FEPClusters in all namespace.

Default output format:

Field	Value	Details
NAME	.metadata.name	Name of Cluster
AGE	Elapsed time	Indicates the amount of time that has elapsed since the cluster was created

Example)

```
# kubectl get fepclusters -A
```

NAMESPACE	NAME	AGE
namespace1	ns1fep1	21h
namespace2	ns2fep2	22h

Update FEPCluster

Equivalent Kubernetes command: `kubectl apply -f <new_spec>`

Operations that can be performed here.

Custom Resource spec	Change effect
.spec.fep.instances: <i>n</i>	Increase the number of nodes in the cluster to <i>n</i> .
.spec.fep.image.image: 'quay.io/fujitsu/fujitsu-enterprise-postgres-16-server:ubi9-16-1.1'	Minor upgrade of FEP image to ubi9-16-1.1.
spec.fepChildCrVal.backup.image.image: 'quay.io/fujitsu/fujitsu-enterprise-postgres-16-backup:ubi9-16-1.1'	Minor upgrade of Backup image to ubi9-16-1.1.

This will impact behaviour for values in `feh` section only.
All parameters can be updated from the `FEPCluster` custom resource.

Delete FEPCluster

Equivalent Kubernetes command: `kubectl delete FEPCluster <cluster_name>`

This operation will remove the `FEPCluster` by the `cluster_name` and all Child CRs (`FEPClusterVolume`, `FEPClusterConfig`, `FEPClusterCert` & `FEPClusterUser`) & resources associated with it.



Note

Deleting a `FEPCluster` will delete all PV associated with the cluster, including backup and archived WAL volumes (except when using pre-made PV or AWS S3). This is an unrecoverable action.

5.3 FEPCluster Resource Change

5.3.1 Changing CPU and Memory Allocation Resources

Describes how to change the CPU and memory resources assigned to a pod created by a `FEPCluster`.

This allows you to scale the pod vertically through custom resources.

To modify CPU and memory resources, modify the `spec.feh.mcSpec` section(*1) of the `FEPCluster` custom resource and apply your changes.

When the changes are applied, restart the replica server with the new resource settings. If there are multiple replica servers, restart them one at a time. When all replica servers are restarted, one of them is promoted to the new master server due to a switchover. Then restart the container image on the original master server. This allows you to change resource settings for all servers with minimal disruption.

*1) Modifying this section scales up the FEP server container. For information about other container resource sections, refer to "FEPCluster Parameters" in the Reference.

5.3.2 Resizing PVCs

Describes how to resize a PVC assigned to a pod created by a `FEPCluster`.

This allows you to increase the size of the volume allocated to the pod through custom resources.

To change the PVC size, modify the size of each volume in the `spec.fehChildCrVal.storage` section of the `FEPCluster` custom resource and apply the change. These changes apply to all PVCs assigned to the pod created by the `FEPCluster`.



Note

- PVC resizing is extensible only.
- You can resize a PVC only if the `StorageClass` supports dynamic resizing.
- If the `StorageClass` does not support resizing PVCs, use the `FEPClusterRestore` custom resource to create a new `FEPCluster` to resize the PVC. For more information, refer to "FEPClusterRestore Custom Resource Parameters" in the Reference.

5.4 FEPPGPool2 Configuration Change

This section describes changes to the `FEPPGPool2` configuration.

List FEPPGPool2

Equivalent Kubernetes command: `kubectl get FEPPGPool2 (-A)`

This operation will list all FEPPGPool2 in a namespace, or if the -A option is specified, will list all FEPPGPool2 in all namespace.

Default output format:

Field	Value	Details
Name	.metadata.name	Name of pgpool2

Example)

# kubectl get feppgpool2 -A	
NAMESPACE	NAME
namespace1	fep1-pgpool2
namespace2	fep2-pgpool2

Delete FEPPGPool2

Equivalent Kubernetes command: kubectl delete FEPPGPool2 <pgpool2_name>

This operation will remove the FEPPGPool2 by the pgpool2_name.

Update FEPPGPool2

Equivalent Kubernetes command: kubectl apply -f <new_spec>

Refer to "FEPPgpool2 Custom Resource Parameters" in the Reference and specify the parameters to be updated. Only the following parameters can be specified.

Custom Resource spec	Change Effect
.spec.count: n	Increase the number of nodes in the cluster to n.
.spec.serviceport	Change the TCP port for connecting to the Pgpool-II.
.spec.statusport	Change the TCP port for connecting to the PCP process.
.spec.limits.cpu	Change limits of cpus.
.spec.limits.memory	Change limits of memory.
.spec.requests.cpu	Change requests of cpus.
.spec.requests.memory	Change requests of memory.
.spec.fepclustername	Change fepcluster to connect.
.spec.customhba	Change pool_hba.conf file.
.spec.customparams	Change pgpool2 parameters
.spec.custompcp	Change pcp.conf file.
.spec.customsslkey	Change key content
.spec.customsslcert	Change the contents of the public x 509 certificate.
.spec.customsslcert	Change the contents of the CA root certificate in PEM format.

Some of the customparams parameters, customhba and custompcp, require a restart of pgpool2.

Equivalent Kubernetes command: Kubectl apply -f <new_spec>

"pgpool2_restart" action type expects users to specify the name of the pgpool2 that they want to restart from.

Specify the metadata.Name of the FEPPGPool2 CR in the targetPgpool2Name section of the FEPACTION CR, as below:

spec:
targetPgpool2Name: fep1-pgpool2

```
fepAction:
  type: pgpool2_restart
```



Note

When updating FEPPGPool2, the Pod of FEPPGPool2 is restarted. If configured with more than one FEPPGpool2, they are rebooted sequentially. The application should be designed to reconnect the connection because the connection being connected is broken.

Update each resource used for client authentication using scram-sha-256 authentication

Set password for new database user

When you add a new user to the database, you must notify the FEPpgpool2 container of the user and password.

In this case, in addition to the information of the existing user, add the information of the new user to the secret as follows.

```
apiVersion: v1
kind: Secret
metadata:
  name: test-secret
type: Opaque
data:
  user_name_1:cGFzc3dvcmRfMQ==
  user_name_2:cGFzc3dvcmRfMg==
  user_name_new:cGFzc3dvcmRfbmV3
```

After updating the secret, the FEPpgpool2 container automatically updates the contents of the password file pool_passwd.

If you want to edit the operator-created secret to add new user information, delete the postgres key entry. If there are still postgres key entries, the new user information will not be reflected.

Update passwords for existing database users

The FEPCluster container may update the passwords of database users for reasons such as password expiration.

In this case, keep the information of the existing user, and update the user information secret for the updated password as follows.

```
apiVersion: v1
kind: Secret
metadata:
  name: test-secret
type: Opaque
data:
  user_name_1:cGFzc3dvcmRfMQ==
  user_name_2:cGFzc3dvcmRfMl9uZXc=
```

After updating the secret, the FEPpgpool2 container automatically updates the contents of the password file pool_passwd.

Update encryption key file pgpoolkey

If you want to update the encryption key file pgpoolkey, update the contents of the secret defined in the encryption key secret pgpoolkeySecret.

```
apiVersion: v1
kind: Secret
metadata:
  name: pgpoolkey-secret
type: Opaque
data:
  pgpoolkey:bmV3LXBncG9vbGt leXBhc3N3b3Jk
```

If you want to update the encryption key file pgpoolkey, you must use the updated pgpoolkey and encrypt it again with scram-sha-256. You will also need to restart pgpool2 after updating the encryption key secret.

After updating the secret, the FEPpgpool2 container automatically updates the contents of the password file pool_passwd with the updated pgpoolkey.

5.5 Scheduling Backup from Operator

Operational status confirm

Information about the backup can be found by running the command in the FEP backup container, as shown in the example below.

```
$ oc exec pod/fepserver-XXXXX -c FEPbackup -- pgbackrest info
stanza: febackup
  status: ok
  cipher: none

  db (current)
    wal archive min/max (12-1): 00000001000000000000000001/000000010000000000000005

    full backup: 20201125-025043F
      timestamp start/stop: 2020-11-25 02:50:43 / 2020-11-25 02:50:52
      wal start/stop: 00000001000000000000000003 / 000000010000000000000003
      database size: 31.7MB, backup size: 31.7MB
      repository size: 3.9MB, repository backup size: 3.9MB

    incr backup: 20201125-025043F_20201125-025600I
      timestamp start/stop: 2020-11-25 02:56:00 / 2020-11-25 02:56:02
      wal start/stop: 00000001000000000000000005 / 000000010000000000000005
      database size: 31.7MB, backup size: 24.3KB
      repository size: 3.9MB, repository backup size: 619B
      backup reference list: 20201125-025043F
```

Update FEPBackup

Equivalent Kubernetes command: `kubectl apply -f <new_spec>`

Refer to "FEPBackup Child Custom Resource Parameters" in the Reference and specify the parameters to be updated. Only the following parameters can be specified.

Custom Resource spec	Change Effect
spec.schedule.num	Change the Number of Registered Backup Schedules
spec.scheduleN.schedule	Change the scheduled backup time
spec.scheduleN.type	Change the scheduled backup type
spec.pgBackrestParams	Change pgBackRest parameters
spec.scheduleN.repo	If you specified more than one repository for spec.pgBackrestParams, select the repository in which to store the backup data. The default is 1.



Note

- Changes made during the backup are reflected from the next backup.
- Changes to the backup schedule do not affect the application.
- If you perform any of the following update operations, be sure to obtain a backup after the update.
 - When the master encryption key is updated with `pgx_set_master_key`
 - When the encryption passphrase for transparent data encryption is updated (can be updated by the `tdeppassphrase` parameter of FEPCluster CR)

5.6 Configure MTLS Setting

5.6.1 Certification Rotation

All certificates are bounded by the time limit. At certain time, it needs to be renewed. We recommend to renew the certificate when it reaches 3/4 of its life cycle or as soon as possible if it is compromised. When a certificate is renewed, we need to rotate it inside the FEP server container. At the moment, FEP server container does not support automatic certificate rotation. Depending on which certificate has renewed, there are different procedures to handle that.

Patroni Certificate Rotation

When Patroni certificate is renewed, we have to re-deploy each and every Pod for FEP server container to pick up the new certificate. There is a down time on FEPCluster.

FEP Server Certificate Rotation

When FEP Server certificate is renewed, we can use FEPAction CR to trigger a reload of the database and FEP server will pick up the new certificate with no interruption to service.

Client certification Rotation

When any of the client certificate is renewed, FEP server container internally will use the new certificate next time it establishes a connection to FEP server. However, to avoid any unexpected interruption to service, it is recommended to re-deploy each and every Pod as soon as possible.

5.7 Monitoring

Monitoring is collecting historic data points that you then use to generate alerts (for any anomalies), to optimize databases and lastly to be proactive in case something goes wrong (for example, a failing database).

There are five key reasons to monitor FEP database.

1. Availability

It is a very simple equation that if you do not have a database in running, your application will not work. If the application is critical, it directly effects on users and the organization.

2. System Optimization

Monitoring helps to identify the system bottlenecks and according to the user can make changes to your system to see if it resolves the problem or not. To put this into perspective, there may be a situation where users see a very high load on the system. And figured out that there is a host parameter that can be set to a better value.

3. Identify Performance Problems

Proactive monitoring can help you to identify future performance problems. From the database side, it could be related to bloating, slow running queries, table and index statistics, or the vacuum being unable to catch up.

4. Business Process Improvement

Every database user has a different need and priority. Knowing the system (load, user activity, etc.) helps you to prioritize customer tasks, reporting, or downtime. Monitoring helps to make business process improvement.

5. Capacity Planning

More user or application growth means more system resources. It leads to key questions: Do you need more disk space? Do you need a new read replica? Do you need to scale your database system vertically? Monitoring helps you to understand your current system utilization—and if you have data, points spread over a few weeks or months, it helps to forecast system scaling needs.

This article describes monitoring and alerting operations using OpenShift's standard Pod alive monitoring, resource monitoring and database statistics provided by the FEP Exporter.

5.7.1 Monitoring FEP Operator and Operands

The monitoring of FEP operators and operands are achieved by Prometheus' standard alive and resource monitoring.

Metrics name	Details
Alive monitoring	Can monitor Pod status
Resource monitoring	You can monitor the following resource status - CPU Usage - CPU Quota - Memory Usage - Memory Quota - Current Network Usage - Receive Bandwidth - Transmit Bandwidth - Rate of Received Packets - Rate of Transmitted Packets - Rate of Received Packets Dropped - Rate of Transmitted Packets Dropped

By setting alert rules based on these monitoring items, operators and operands can be monitored. For the setting method, refer to the appendix in the Reference.

If an error is detected by monitoring the operator's alive, it can be dealt with by recreating the Pod.

If resource monitoring detects an error, consider allocating more resources to the Operator or Operands.

Check the Operator Hub or Red Hat Operator Catlog page to see which version you are currently using, which can be updated, and to check for security vulnerabilities.

5.7.2 Monitoring FEP Server

Monitoring and alerts system leverages standard GAP stack (Grafana, Alert manager, Prometheus) deployed on OCP and Kubernetes. GAP stack must be there before FEP operator & FEPCluster can be deployed.

Prometheus is a condensed way to store time-series metrics. Grafana provides a flexible and visually pleasing interface to view graphs of FEP metrics stored in Prometheus.

Together they let store large amounts of metrics that user can slice and break down to see how the FEP database is behaving. They also have a strong community around them to help deal with any usage and setup issues.

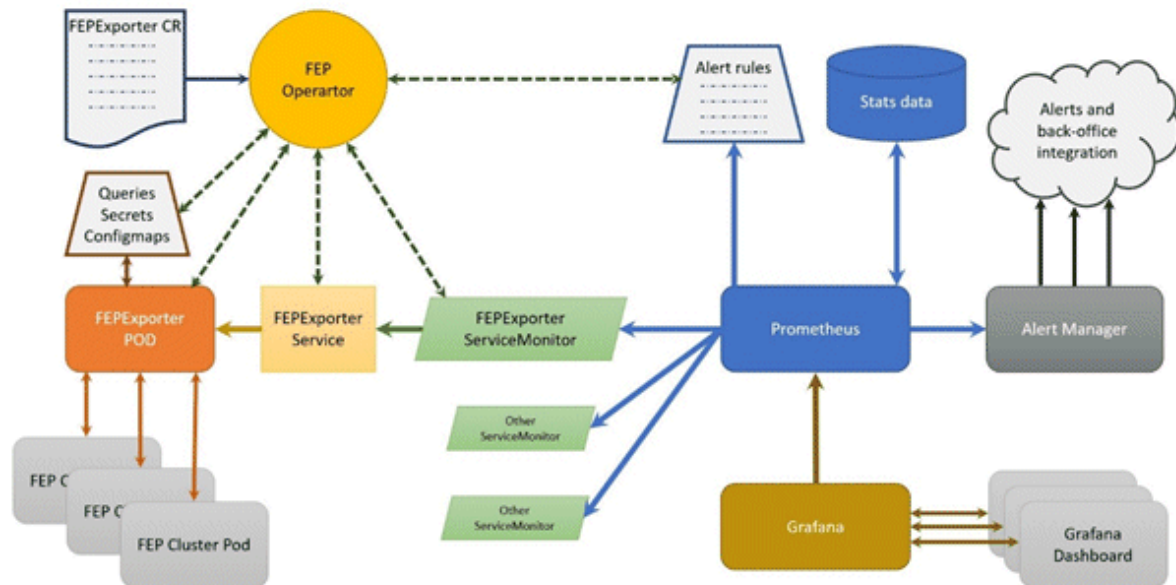
The Prometheus acts as storage and a polling consumer for the time-series data of FEP container. Grafana queries Prometheus to displaying informative and very pretty graphs.

If Prometheus rules are defined, it also evaluates rules periodically to fire alerts to Alert manager if conditions are met. Further Alert manager can be integrated with external systems like email, slack, SMS or back-office to take action on alerts raised.

Metrics from FEP Cluster(s) is collected by Prometheus through optional components deployed using FEP Exporter with default set of metrics and corresponding Prometheus rules to raise alerts. User may extend or overwrite metrics by defining their custom metrics queries and define their custom Prometheus rules for alerting.

5.7.2.1 Architecture

Block diagram of monitoring FEP server is as follows.



- FEPEXporter CR is managed by FEP Operator
- When FEPEXporter CR is created, FEP operator creates following kubernetes objects:
 - ConfigMap that contains default and custom queries to collect metrics from database cluster from each node
 - Secret containing JDBC URL for all FEPCluster nodes to connect and request metrics. This string contains authentication details as well to make JDBC connection.
 - Prometheus rules corresponding to default alert rules
 - ServiceMonitor for Prometheus to discover FEPEXporter service
 - FEPEXporter container using FEPEXporter image to scrape metrics from all FEPCluster nodes

Note

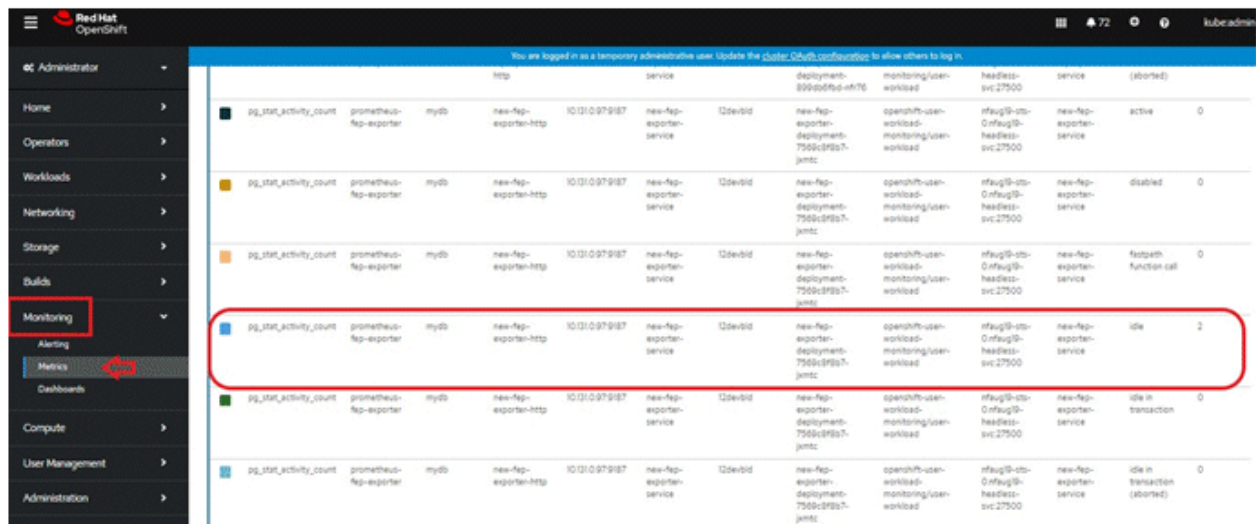
- Alert Manager integration to back-office to send mail / message / raising ticket is done by user based on their environment
- Grafana installation and integration is done by user. Use the Grafana Operator provided by OperatorHub.
- Grafana dashboard is created by user based on their requirements and design.

5.7.2.2 Default Server Metrics Monitoring

By default FEPEXporter scrapes some useful metrics for server.

Once FEPEXporter is running, user can check the collected metrics under Openshift->Monitoring->Metrics submenu.

Refer an example below.



There are 2 types of default server metrics defined by FEP Exporter.

Type	Details
Default mandatory	Are collected by FEP Exporter. These are kept enabled by default by FEP Exporter and can not be disabled by end user.
Default useful	Useful focused metrics for health and performance metrics. Can be disabled by end user.

Default mandatory metrics

These metrics are either from basic statistics view of the database or FEP Exporter own metrics;

Various metrics under this category are

Metrics name	Details
pg_stat_bgwriter_*	Maps to view in Statistic Collector
pg_stat_database_*	Maps to view in Statistic Collector
pg_stat_database_conflicts_*	Maps to view in Statistic Collector
pg_stat_archiver_*	Maps to view in Statistic Collector
pg_stat_activity_*	Maps to view in Statistic Collector
pg_stat_replication_*	Maps to view in Statistic Collector
pg_replication_slots_*	Maps to System Catalog pg_replication_slots
pg_settings_*	Maps to System Catalog pg_settings
pg_locks_*	Maps to System Catalog pg_locks
pg_exporter_*	Exposes exporter metrics: - last_scrape_duration_seconds (Duration of the last scrape of metrics from PostgreSQL) - scrapes_total (Total number of times PostgreSQL was scraped for metrics) last_scrape_error (Whether the last scrape of metrics from PostgreSQL resulted in an error; 1 for error & 0 for success)
pg_*	Exposes exporter metrics - pg_up (set to 1 if the connection to service is success, 0 otherwise)

Metrics name	Details
	- pg_static (can be used to fetch label short_version / version containing postgres server version information)

Default useful metrics

There are certain useful queries which are additionally added to evaluate the health of the Database system.

Metrics name	Details
pg_capacity_connection_*	Metrics on connections e.g. txns running for 1 hour
pg_capacity_schema_*	Metrics on disk space of schema
pg_capacity_tblspace_*	Metrics on disk space of tablespace
pg_capacity_tblvacuum_*	Metrics on tables without vacuum for days
pg_capacity_longtx_*	Number of transactions running longer than 5 minutes Review the information and consider SQL tuning and resource enhancements.
pg_performance_locking_detail_*	Details of processes in blocked state
pg_performance_locking_*	Number of processes in blocked state
pg_replication_*	Replication lag behind master in seconds Provides the ability to check for the most current data in a reference replica To solve the problem, it is necessary to consider measures such as increasing network resources and reducing the load
pg_postmaster_*	Time at which postmaster started
pg_stat_user_tables_*	Important statistics from pg_stat_user_tables
pg_statio_user_tables_*	Important statistics from pg_statio_user_tables
pg_database_*	Database size If the database runs out of space, database restore is required
pg_stat_statements_*	Statistics of SQL statements executed by server
pg_capacity_tblbloat_*	Fetches bloat in tables
pg_tde_encrypted_*	Presence or absence of transparent data encryption in the tablespace and the number of tables and indexes stored
pg_password_valid_*	Database Role Password Validity Period
pg_not_set_password_valid_*	Number of database roles with no password expiration
pg_user_profile_*	Number of database roles in each status for policy-based password operation



Note

You can tune the intervals and thresholds at which information is gathered by changing the values specified in the information gathering query. For more information, refer to the queries in the appendix of the Reference Guide, and make your own settings.

5.7.2.3 Default Alerts

There are few basic alert rules which are setup by the FEP Operator as below

Alert rule	Alert Level	Condition persistence	Description
ContainerHighCPUUsage	Warning	5 mins	FEP server container/Pod CPU usage is exceeding 80% of the resource limits
ContainerHighRAMUsage	Warning	30 mins	FEP server container/Pod memory usage is exceeding 80% of the resource limits
PVCLowDiskSpace	Warning	5 mins	A FEP PVC (volume) has less than 10% disk available
ContainerDisappeared	Warning	60 seconds	FEP server container/Pod has disappeared since last 60 seconds
PostgresqlDown	Error	-	FEP server apparently went down or not accessible
PostgresqlTooManyConnections	Warning	-	FEP server container/Pod connection usage is beyond 90% of its available capacity
PostgresqlRolePasswordCloseExpierd	Warning	-	A Postgresql role exists with a password expiration of less than 7 days
PostgresqlRolePasswordExpired	Warning	-	A Postgresql role exists with an expired password
PasswordIsGraceTimeByUserProfile	Warning	-	There is a Postgresql role that is in the grace period due to policy-based password operation
PasswordExpiredByUserProfile	Warning	-	There is a Postgresql role whose password has expired due to policy-based password operation
PasswordLockedByUserProfile	Warning	-	There is a Postgresql role that is locked by policy-based password operation

You can configure any alert by adding alert rules to other monitoring items.

Alerts are based on statistics/metrics. Incorrect platform statistics can cause false alarms. For example, when using NFS storage, the system may raise false alarms for PVCLowDiskSpace when the storage driver is not showing the correct metrics for PV byte usage.

5.7.2.4 Graphical user interface

User can build their custom dashboard using default and custom metrics.

An example Grafana dashboard screenshot is shown below



5.7.3 Monitoring FEP Backup

You can view information about the backed-up data and the status of the backup process in the FEP server tables and system views. Backup information is updated when the automatic backup process completes or when backup data is deleted as specified by retention. The following tables and views are added. The tables and views to be added are created under the `feep_exporter` schema in the postgres database on the FEP server.

Table/View name	Details
<code>pgbackrest_info_backup</code>	Backup Processing Status

5.7.3.1 `pgbackrest_info_backup` view

Contains one line per backup for information about the state of the backup.

Column	Type	Description
<code>label</code>	text	Information identifying the backup
<code>type</code>	text	full: full backup, incr: incremental backup
<code>prior</code>	text	Label of the backup that should be applied first (For incremental backups only)
<code>database_size</code>	bigint	Database size
<code>database_size_comp</code>	bigint	Database size (After Compression)
<code>backup_size</code>	bigint	Backup size
<code>backup_size_comp</code>	bigint	Backup size (After Compression)
<code>archive_start</code>	text	Range of WALs required for restore (Start)
<code>archive_stop</code>	text	Range of WALs required for restore (End)
<code>backup_start</code>	timestamp with timezon	Backup Start Time
<code>backup_stop</code>	timestamp with timezone	Backup End Time
<code>backup_exec_time</code>	interval	The duration of the backup

5.7.4 Monitoring FEP PGPool2

Information about pgpool2 activity and replication status can be found in the FEP server table and in the system view.

The pgpool2 statistics are updated according to the schedule specified in the parameter.

The tables and views that have been added are described below. The tables and views to be added are created under the `feh_exporter` schema in the postgres database on the FEP server.

Table/View name	Details
pgpool2_stat_load_balance	Load Balance Information in pgpool2
pgcluster_stat_replication	Replication State
pgpool2_stat_conn_pool	Connection Pool State for pgpool2
pgpool2_stat_sql_command	SQL Command Statistics

5.7.4.1 pgpool2_stat_load_balance view

Contains one row for MasterService and one row for ReplicaService.

Column	Type	Description
node_id	integer	database node id (0 or 1)
status	text	status (up or down)
lb_weight	double precision	load-balancing weight
role	text	role (primary or standby)
last_status_change	timestamp with time zone	last status change time

5.7.4.2 pgpool2_stat_conn_pool view

Indicates the state of the connection pool. Contains connection pool information for each pgpool2 instance.

Column	Type	Description
pgpool2_node_id	integer	pgpool2 node id (0 - the number of pgpool2 instance - 1)
pool_pid	integer	The PID of the displayed Pgpool-II process
start_time	timestamp with timezone	The timestamp of when this process was launched
pool_id	integer	The pool identifier (should be between 0 and max_pool - 1)
backend_id	integer	The backend identifier (should be between 0 and the number of configured backends minus one)
role	text	role (primary or standby)
database	text	The database name for this process's pool id connection
username	text	The user name for this process's pool id connection
create_time	timestamp with timezo	The creation time and date of the connection
majorversion	integer	The protocol version numbers used in this connection
minorversion	integer	The protocol version numbers used in this connection
pool_counter	integer	Counts the number of times this pool of connections (process) has been used by clients
pool_connected	boolean	True (1) if a frontend is currently using this backend

5.7.4.3 pgpool2_stat_sql_command view

Represents SQL command statistics.

Column	Type	Description
node_id	integer	The backend identifier (should be between 0 and the number of configured backends minus one)
role	text	role (primary or standby)
select_cnt	integer	The numbers of SQL command: SELECT
insert_cnt	integer	The numbers of SQL command: INSERT
update_cnt	integer	The numbers of SQL command: UPDATE
delete_cnt	integer	The numbers of SQL command: DELETE
ddl_cnt	integer	The numbers of SQL command: DDL
other_cnt	integer	The numbers of SQL command: others
panic_cnt	integer	The numbers of failed commands
fatal_cnt	integer	The numbers of failed commands
error_cnt	integer	The numbers of failed commands

5.8 Event Notification

The eventing mechanism introduced, is to enable operator to raise customized Kubernetes events. The custom events will be raised during the creation of custom resources. Currently following events are raised.

5.8.1 Events raised

- fepcluster - During FEPCluster CR creation
 - Event is raised when FEPVolume CR creation is initiated and when FEPVolume CR creation initiation fails.
 - Event is raised when FEPConfig CR creation is initiated and when FEPConfig CR creation initiation fails.
 - Event is raised when FEPUser CR creation is initiated and when FEPUser CR creation initiation fails.
 - Event is raised when FEPCert CR creation is initiated and when FEPCert CR creation initiation fails.
 - Event is raised when Statefulset creation is successful and Statefulset creation fails.
 - Event is raised when PDB creation is successful and when PDB creation fails.
 - Event is raised when FEPBackup CR creation is initiated and when FEPBackup CR creation initiation fails.
 - Event is raised when a FEPCluster custom resource with Velero enabled specifies the same object storage path for production and disaster recovery environments.

Please note the following child CR events are raised as part of Create FEP Cluster

- fepcert - During FEPCert CR creation
 - Event is raised when FEPCert CR creation is successful, when FEPCert CR fails annotating FEPCluster and when FEPCert CR creation fails.
- fepconfig - During FEPConfig CR creation
 - Event is raised when FEPConfig CR creation is successful, when FEPConfig CR fails annotating FEPCluster and when FEPConfig CR creation fails.

- fepvolume - During FEPVolume CR creation
 - Event is raised when FEPVolume CR creation is successful, when FEPVolume CR fails annotating FEPCluster and when FEPVolume CR creation fails.
- feppool2- During FEPPgPool2 CR creation
 - Event is raised when FEPPgPool2 CR creation is successful and when FEPPgPool2 CR creation fails.
 - Event is raised when FEPPgPool2Cert CR creation is initiated and when FEPPgPool2Cert CR creation initiation fails.

Please note the following child CR event are raised as part of Create FEP PgPool2

- feppool2cert- During FEPPgPool2Cert CR creation
 - Event is raised when FEPPgPool2Cert CR creation is successful, when FEPPgPool2Cert CR fails annotating FEPPgPool2 and when FEPPgPool2Cert CR creation fails
- feprestore - During FEPRestore CR creation
 - Event is raised when FEPRestore CR creation is successful and when FEPRestore CR creation fails.

5.8.2 Events that Occur when Custom Resources are Updated

If the "sysExtraEvent" parameter is specified in the custom resource, an event will be generated when changes to FEPCluster, FEPLogging, FEPEXporter are detected, or when the changes are applied successfully/failed.

Refer to "Operator operation event notification" in "Reference" for information about events that occur.

5.8.3 Viewing the Custom Events

The custom events can be viewed on CLI as well as the Openshift console

On cli

Executing the command

kubectl get events

OR

oc get events

Following is a snippet of the events output is ==shown when the above command is executed,

1.3m	Normal	InitiatedChildCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Started FEP Volume CR creation
1.3m	Normal	InitiatedChildCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Started FEP User CR creation
1.3m	Normal	InitiatedChildCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Started FEP Cert CR creation
1.3m	Normal	InitiatedChildCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Started FEP Backup CR creation
1.3m	Normal	SuccessfulFepVolumeCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Successfully created FEP Volume
1.3m	Normal	SuccessfulFepUserCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Successfully created FEP User
1.3m	Normal	SuccessfulFepCertCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Successfully created FEP Cert
1.3m	Normal	SuccessfulFepConfigCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Successfully created FEP Config
1.3m	Normal	SuccessfulFepBackupCronjobCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Successfully created FEP Backup Cronjob1
1.3m	Normal	SuccessfulFepBackupCronjob2Create	feplaster/new-fep-hg-12-08-21	playground-hg,	Successfully created FEP Backup Cronjob2
1.3m	Normal	SuccessfulFepVolumeCreate	feplaster/new-fep-hg-12-08-21	playground-hg,	Successfully created FEP Volume

On openshift console

For the specific project/ namespace the custom events can be viewed along with Kubernetes events under the events as shown in the following screenshot.

Project: playground-hg ▾

Events

Resources 1 ▾ Normal ▾ Filter Events by name or message... [f]

Resource All x x

Streaming events... Showing 46 events

FEPB new-fep-hg-12-08-21 Generated from fepbackups playground-hg, Successfully created FEP Backup Cronjob2	NS playground-hg	Aug 12, 5:49 pm
FEPB new-fep-hg-12-08-21 Generated from fepbackups playground-hg, Successfully created FEP Backup Cronjob1	NS playground-hg	Aug 12, 5:49 pm

5.9 Scaling Replicas

5.9.1 Automatic Scale Out

Automatic scale out occurs when the average CPU utilization or number of connections of the DB container exceeds the threshold.

The maximum number of replica containers, excluding the master container, is 15.

If the load decreases after the number of replicas increases due to a temporary increase in load, the number of replicas will remain increased. Perform manual scale in if necessary.

Specify `spec.fepChildCrVal.autoscale.scaleout` in `FEPClusterCR` when you want to perform Automatic scale out. Refer to "FEPCluster Parameters" in the Reference for information about the values to specify.

```
$ oc edit fepcluster <FEPClusterCR name>
```

5.9.2 Manual Scale In/Out

To manually scale in or out of a `FEPCluster`, edit the "`spec.fep.instances`" in `FEPClusterCR`.

The value must be between 1 and 16. (Number of instances with one master)

```
$ oc edit fepcluster <FEPClusterCR name>
```



Note

- Do not scale in from two to one replica instance when the `syncMode` is 'on'. Update SQL cannot be executed.
- Any database connections to the replica Pod that are deleted during a scale in will be forced to disconnect.

5.10 Backing Up to Object Storage

Describes how to store backup data in object storage.

5.10.1 Pre-creation of Resources

5.10.1.1 Storing CA Files (Root Certificates)

If you want to use a non-default root certificate for object storage connections, register it in `ConfigMap`.

```
$ oc create configmap storage-cacert --from-file=ca.crt=storage-ca.pem -n my-namespace
```

5.10.1.2 Storing Repository Key

When using the parameter (repo-gcs-key) of pgBackRest, register the GCS repository key in Secret.

```
$ oc create secret generic storage-key-secret --from-file=key.json=storage-key.json -n my-namespace
```

5.10.2 Defining a FEPCluster Custom Resource

List the backup settings under spec.fepChildCrVal.backup in the FEPCluster custom resource.

Specify the object storage for the backup data in pgbackrestParams. Refer to "[2.3.5 Scheduling Backup from Operator](#)" for possible values for pgbackrestParams.

Specify the ConfigMap name created in "[5.10.1.1 Storing CA Files \(Root Certificates\)](#)" for caName.

FEPCluster Custom Resource Example: Only Object Storage Used for Backup Repository

```
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  ...
spec:
  fepChildCrVal:
    backup:
      pgbackrestParams: |
        repo1-type=s3
        repo1-path=/backup/cluster1
        repo1-s3-bucket= sample-bucket
        repo1-s3-endpoint=s3.ap-northeast-1.amazonaws.com
        repo1-s3-region=ap-northeast-1
        repo1-storage-ca-file=/pgbackrest/storage-certs/ca.crt
      pgbackrestKeyParams: |
        repo1-s3-key=SAMPLEKEY
        repo1-s3-key-secret=SAMPLESECRET
      caName:
        - storage-cacert
  ...
```

If the persistent volume and object storage specified in spec.fepChildCrVal.storage.backupVol are to be used together in the backup repository, specify the object storage setting after "repo2".

If "repo1" is not defined, a permanent volume is automatically designated as the storage destination for the backup volume.

FEPCluster Custom Resource Example: When using object storage and PV

```
...
spec:
  fepChildCrVal:
    backup:
      pgbackrestParams: |
        repo2-type=s3
        repo2-path=/backup/cluster1
        repo2-s3-bucket= sample-bucket
        repo2-s3-endpoint=s3.ap-northeast-1.amazonaws.com
        repo2-s3-region=ap-northeast-1
        repo2-storage-ca-file=/pgbackrest/storage-certs/ca.crt
      pgbackrestKeyParams: |
        repo2-s3-key=SAMPLEKEY
        repo2-s3-key-secret=SAMPLESECRET
      caName:
```

```
...    - storage-cacert
```

When using object storage GCS as a backup repository, specify as follows.

For repoKeySecretName, specify the Secret created in "[5.10.1.2 Storing Repository Key](#)". Also, specify service for gcs-key-type.

FEPCluster Custom Resource Example: When using GCS as a backup repository

```
apiVersion: fep.fujitsu.io/v1
kind: FEPCluster
metadata:
  ...
spec:
  fepChildCrVal:
    backup:
      pgbackrestParams: |
        repo1-type=gcs
        repo1-path=/backup-ct/test2
        repo1-gcs-bucket=dbaas-gcs
        repo1-gcs-endpoint=localhost
        repo1-storage-ca-file=/pgbackrest/storage-certs/ca.crt
        repo1-gcs-key=/pgbackrest/storage-keys/key.json
        repo1-gcs-key-type=service
      caName:
        - storage-cacert
      repoKeySecretName:
        - storage-key-secret
  ...
```

5.11 Disaster Recovery

Available disaster recovery methods include backup/restore method, continuous recovery method, and streaming replication method.

5.11.1 Disaster Recovery by Backup/Restore Method

5.11.1.1 Disaster Recovery Prerequisites

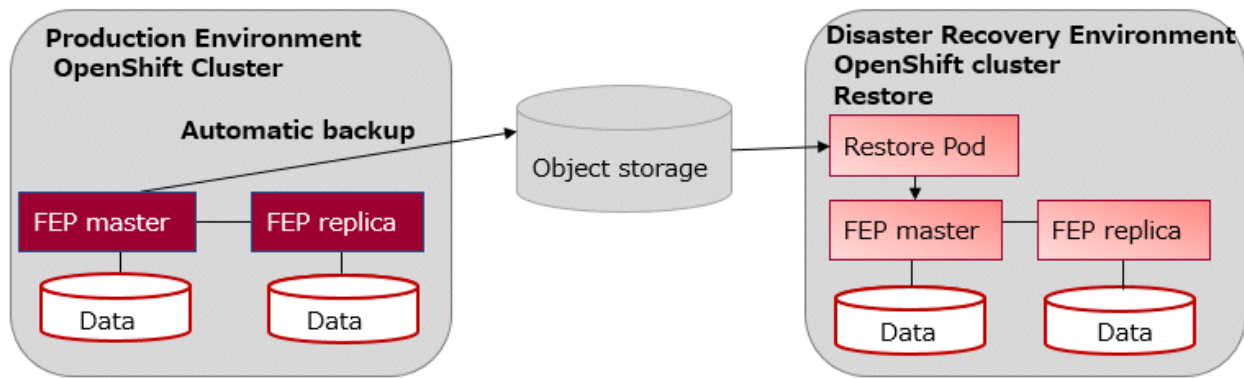
The configuration diagram of the Pod arrangement and backup repository, which are prerequisites for the backup function to perform disaster recovery using the backup/restore method, is shown below.

In FEPCluster to get a backup, specify the object storage as the backup data storage destination with spec.fepChildCrVal.backup.pgbackrestParams.

Specify object storage that is in an area that is considered safe for the scope of the expected disaster.

The definition of the FEPCluster custom resource is not inherited when performing disaster recovery.

We recommend that you save your production environment FEPCluster custom resource definitions in case of a disaster.



5.11.1.2 Performing Disaster Recovery

Describes the procedure for restoring to an OCP environment different from the restore source using the backup data stored in the object storage.

5.11.1.2.1 Pre-creation of Resources

Storing CA Files (Root Certificates)

If you want to use a non-default root certificate for object storage connections, register it in ConfigMap.

```
$ oc create configmap storage-cacert --from-file=ca.crt=storage-ca.pem -n my-namespace
```

Storing GCS Repository Key

When using the parameter (repo-gcs-key) of pgBackRest, register the GCS repository key in Secret.

```
$ oc create secret generic storage-key-secret --from-file=key.json=storage-key.json -n my-namespace
```

5.11.1.2.2 Defining a FEPCluster Custom Resource

In addition to the FEPCluster settings, specify the Restore settings below.

FEPCluster Custom Resource Example

```
apiVersion: fep.fujitsu.io/v1
kind: FEPCluster
metadata:
  ...
spec:
  fepChildCrVal:
    restore:
      pgbackrestParams: |
        repo1-type=s3
        repo1-path=/backup/cluster1
        repo1-s3-bucket=sample-bucket
        repo1-s3-endpoint=s3.ap-northeast-1.amazonaws.com
        repo1-s3-region=ap-northeast-1
        repo1-storage-ca-file=/pgbackrest/storage-certs/ca.crt
      pgbackrestKeyParams: |
        repo1-s3-key=SAMPLEKEY
        repo1-s3-key-secret=SAMPLESECRET
      caName:
        - storage-cacert
  ...
```

When using object storage GCS as a backup repository, specify as follows.

For repoKeySecretName, specify the Secret created in "[Storing GCS Repository Key](#)". Also, specify service for gcs-key-type.

```
apiVersion: fep.fujitsu.io/v1
kind: FEPCluster
metadata:
  ...
spec:
  fepChildCrVal:
    backup:
      pgbackrestParams: |
        repo1-type=gcs
        repo1-path=/backup-ct/test2
        repo1-gcs-bucket=dbaas-gcs
        repo1-gcs-endpoint=localhost
        repo1-storage-ca-file=/pgbackrest/storage-certs/ca.crt
        repo1-gcs-key=/pgbackrest/storage-key/key.json
        repo1-gcs-key-type=service
      caName:
        - storage-cacert
      repoKeySecretName:
        - storage-key-secret
    ...
```

Setting value

Field	Default	Details
spec.fepChildCrVal.restore		Define when restoring by specifying the backup data stored in the object storage.
spec.fepChildCrVal.restore.pgbackrestParams		Optional " " is fixed, and the following lines specify the parameters to set in pgbackrest.conf. Specify the object storage where the backup data is stored. If you want to use a root certificate other than the default, specify the following: repo1-storage-ca-path=/pgbackrest/storage-certs/<file name> Register the CA file in ConfigMap and specify the ConfigMap name in spec.fepChildCrVal.restore.caName.
spec.fepChildCrVal.restore.pgbackrestKeyParams		Optional " " is fixed, and the following lines specify the parameters to set in pgbackrest.conf. The value described by this parameter is masked with *****. Specify the parameter you want to mask, such as a password.
spec.fepChildCrVal.restore.caName		Optional Specify when you use a CA file other than the system default. Specify the name of the created ConfigMap in list format. The specified ConfigMap will be mounted in /pgbackrest/storage-certs.
spec.fepChildCrVal.restore.mcSpec.limits	cpu: 200m memory: 300Mi	Optional CPU and memory allocated to the container performing the restore.
spec.fepChildCrVal.restore.mcSpec.requests	cpu: 100m memory: 200Mi	Optional CPU and memory allocated to the container performing the restore.

Field	Default	Details
spec.fepChildCrVal.restore.restoretype	latest	Optional Restore Type (latest or PITR)
spec.fepChildCrVal.restore.restoredate		Optional Specify the date to restore when spec.fepChildCrVal.restore.restoretype is "PITR".
spec.fepChildCrVal.restore.restoretime		Optional Specify the time to restore when spec.fepChildCrVal.restore.restoretype is "PITR".
spec.fepChildCrVal.restore.image		Optional Image of the container to perform the restore. It is omitted by default. In this case, the URL for image is obtained from the operator container environment.
spec.fepChildCrVal.restore.imagePullPolicy	IfNotPresent	Optional

5.11.2 Disaster Recovery with Continuous Recovery Method

5.11.2.1 Disaster Recovery Prerequisites

When performing disaster recovery using the continuous recovery method, the database cluster must be configured based on the "[4.13.3.1 Defining a Continuous Recovery Method](#)". At this time, it is necessary to periodically back up the database to object storage.

5.11.2.2 Performing Disaster Recovery

In the event of a disaster, the FEPCluster in the disaster recovery environment should be promoted.

Describes the procedure for promoting a FEPCluster in a disaster recovery environment deployed in a hot standby configuration. If the production environment becomes unusable when a disaster occurs, you can promote the disaster recovery environment to the production environment by executing `FEPAction(promote_standby)`.

An example FEPAction definition is shown below.

```
...
apiVersion: fep.fujitsu.io/v1
kind: FEPAction
metadata:
  name: new-fep-promote-standby-action
  namespace: my-namespace
spec:
  fepAction:
    type: promote_standby
    sysExtraEvent: true
    sysExtraLogging: true
    targetClusterName: my-fep
...
```

5.11.3 Disaster Recovery Using Velero

This section describes disaster recovery using Velero.

5.11.3.1 Disaster Recovery Prerequisites

Prepare for the following in production environment and disaster recovery environment. Before enabling this feature in a production environment, and before restoring using this feature in a disaster recovery environment, you must prepare the following:

- Installing VeleroCLI
- Installing Velero
- Installing FEPOperator
- StorageClass, namespace, CRD, and other resources needed to build the system

If you plan to use a static PV with FEPCluster, prepare the PV before restoring to a disaster recovery environment. If you use a storage class or certificate that is different from the production environment, provide a storage class or certificate with the same name as the production environment in your disaster recovery environment. However, if the object storage certificate or secret where the database backup data is stored differs between the production environment and the disaster response environment, set a different name. For more information, refer to "FEPCluster Parameters" in the Reference.

This feature only supports x86.

5.11.3.2 Performing Disaster Recovery

Provides step-by-step instructions on how to enable this feature to perform disaster recovery from a production environment to a disaster recovery environment. It also describes how to perform disaster recovery again from a disaster recovery environment to a production environment.

FEPCluster and FEPPool2 LOG volumes cannot be restored. Before using this feature, be sure to test and validate in both production and disaster recovery environments.

5.11.3.2.1 Configuring FEPCluster Custom Resources

To use this feature, in addition to the normal configuration of FEPCluster custom resources in your production environment, configure the following to deploy on Kubernetes. For more information, refer to "FEPCluster Parameters" in the Reference.

- `fed.velero.enable`
- `fed.velero.labels`
- `fed.velero.backup`
- `fed.velero.restore`
- `fedChildCrVal.backup`

Example) FEPCluster Custom Resource Definition Example

```
spec:
  fed:
    velero:
      enable: true
      labels:
        backup-dev: my-backup1
        backup-dep: my-backup2
      backup:
        pgbackrestParams: |
          [global]
          repo1-retention-full=5
          repo1-retention-full-type=count
          repo2-retention-full=5
          repo2-retention-full-type=count
          log-path=/database/log/backup
          log-level-file=debug
          repo2-path=/velero-backup/velero-backup-for-dr2
          repo2-s3-bucket=my-s3-bucket
          repo2-s3-endpoint=s3.ap-northeast-1.amazonaws.com
          repo2-s3-region=ap-northeast-1
          repo2-type=s3
        pgbackrestKeyParams: |
          repo2-s3-key=XXXXXXXXXX
```

```

        repo2-s3-key-secret=YYYYYYYYYY
caName: DR-objectstorage-cert
repoKeySecretName: XXX

resotre:
  image:
    image: "XXX-amd64"
  pullPolicy: IfNotPresent
  mcSpec:
    limit:
      cpu: 200m
      memory: 300Mi
    request:
      cpu: 100m
      memory: 200Mi
  restoreTargetRepo: 2
  fepChildCrVal:
    backup:
      pgbackrestParams: |
        [global]
        repo1-retention-full=5
        repo1-retention-full-type=count
        repo2-retention-full=5
        repo2-retention-full-type=count
        log-path=/database/log/backup
        log-level-file=debug
        repo2-path=/ velero-backup/ velero-backup-for-dr1
        repo2-s3-bucket=my-s3-bucket
        repo2-s3-endpoint=s3.ap-northeast-1.amazonaws.com
        repo2-s3-region=ap-northeast-1
        repo2-type=s3
      pgbackrestKeyParams: |
        repo2-s3-key=XXXXXXXXXX
        repo2-s3-key-secret=YYYYYYYYYY
    caName: objectstorage-cert
    repoKeySecretName: ZZZ

  schedule:
    num: 1
    schedule1:
      schedule: "0-59/3 * * * *" #schedule1.schedule
      type: "full" #schedule1.type
    repo: 2

```

5.11.3.2.2 Velero Backup

After building FEPCluster in this environment or modifying FEPCluster custom resources, use the following command to back up the resources on kubernetes, including FEPCluster custom resources, to object storage.

Example)

```
velero backup create <Bavkup name> --selector <Backup target label>
```

This feature supports only Velero backup commands that specify the labels described in `fep.velero.labels` for FEPCluster custom resources and back up only those resources to which the labels are attached. Other commands may back up and restore unnecessary resources, resulting in build and data recovery failures. When specifying labels, do not specify the following keys to avoid confusion with existing labels.

- app
- control-plane
- name

- pod-template-hash
- vendor
- app.kubernetes.io/component
- app.kubernetes.io/instance
- app.kubernetes.io/managed-by
- app.kubernetes.io/name
- app.kubernetes.io/part-of
- control-plane
- controller-revision-hash
- fepclustername
- feprole
- statefulset.kubernetes.io/pod-name
- controller-uid
- job-name
- pod-template-hash
- release

This feature labels the following resources:

- FEPCluster Custom Resources
- Secrets required to build FEPCluster

When backing up the following resources using Velero, assign the labels specified in `spec.fep.velero.labels` (If `spec.fep.velero.labels` is omitted, the default backup-group: `fep-backup`) to the ConfigMap and Secret specified for the various custom and custom resources.

- FEPPgpool2
- FEPExporter
- FEPLogging

You can also back up manually created resources (Application system, ConfigMap, Secret, etc.) by assigning them the labels specified in `spec.fep.velero.labels` (If `spec.fep.velero.labels` is omitted, the default backup-group: `fep-backu`) and making them eligible for Velero backups.

For more information about using the Velero command, refer to the official documentation.



Note

Each time you modify a custom resource, such as FEPCluster, a Velero backup is required. The database is brought up to date with object storage. If a custom resource is in a pre-change state, it may be inconsistent with the database, causing data loss, performance degradation, security, and other issues.

5.11.3.2.3 Database Backup

Perform regular database backups.

5.11.3.2.4 Velero Restore

If business cannot continue in a production environment due to a disaster, use the following command to restore the resources on kubernetes, including the FEPCluster custom resources stored in object storage. The database data is restored to the latest archived WAL state stored in object storage.

Example)

```
velero restore create <Restore name> --from-backup <Backup name>
```

For more information about using the Velero command, refer to the official documentation.

5.11.3.2.5 Return from Disaster Recovery Environment to Production Environment

When restoring from a disaster recovery environment to a production environment again using Velero, the FEPCluster custom resources deployed in the disaster recovery environment must be modified. The FEPCluster custom resource after disaster recovery, `fev.velero.backup`, `fev.velero.restore`, defines pre-disaster recovery information. Update the information to the object storage used in the production environment. Then back up with Velero in a disaster recovery environment and restore with Velero in a production environment.

5.11.4 Disaster Recovery with Streaming Replication Method

5.11.4.1 Disaster Recovery Prerequisites

When performing disaster recovery using the streaming replication method, the database cluster must be configured based on the ["4.13.3.2 Defining a Streaming Replication Method"](#). After deployment, database data is synchronized between the production environment and the disaster recovery environment using the streaming replication method, so no operation is required during operation.

5.11.4.2 Performing Disaster Recovery

When a disaster occurs, it is necessary to promote the FEPCluster in the disaster recovery environment. For promotion of FEPCluster in a disaster recovery environment, refer to ["5.11.2.2 Performing Disaster Recovery"](#).

5.11.5 Parameter Change in Disaster Recovery Environment

If parameters (such as FEPClusterCR) are changed in the production environment, manually reflect the changes in the disaster recovery environment. However, if the database password is changed, the post-change value is automatically reflected in the disaster recovery environment, so manual reflection is not required.

5.12 Operation of Transparent Data Encryption Using Key Management System

5.12.1 Updating Custom Resource Parameters

When using a newly generated master encryption key in your key management system, update the FEPCluster custom resource `fevChildCrVal.sysTde.tdek.targetKeyId` to the ID of the new master encryption key. The operator will automatically re-enable TDE when this value is updated.

Also, if the credentials for connecting with the key management system are updated, update the corresponding values in the FEPCluster custom resource. The operator automatically performs a keystore open when the credentials are updated.

When re-enabling TDE or opening the keystore is completed, the following event will be notified.

```
# When re-enabling TDE
$ kubectl get event
LAST SEEN   TYPE      REASON              OBJECT                                MESSAGE
164m        Normal    SuccessfulTdeSetMasterKey  fevconfig/<FEPClusterCR name> <namespace>, Successfully
set TDE masterKey

# When re-enabling TDE fails
$ kubectl get event
LAST SEEN   TYPE      REASON              OBJECT                                MESSAGE
164m        Warning   FailedTdeSetMasterKey    fevconfig/<FEPClusterCR name> <namespace>, Error/
Failure set TDE masterKey
```

If the process fails, review the parameters defined in the FEPCluster custom resource and re-enter the correct values.

If only the contents of the Secret or ConfigMap that stores the credentials are updated and the custom resource is not modified, open the keystore using the FEPAAction custom resource described in ["5.12.2 Update Credentials"](#).

5.12.2 Update Credentials

If the credentials in the key management system are updated, update the contents of the corresponding Secret or ConfigMap. If there are no changes to the values specified in the FEP cluster custom resource, apply the FEPAAction custom resource to update the credentials used by FEP.

Example) Definition example of FEPAAction custom resource

```
apiVersion: fep.fujitsu.io/v1
kind: FEPAAction
metadata:
  name: new-fep-action
spec:
  sysExtraLogging: false
  targetClusterName: nf-131851
  fepAction:
    type: open_tde_masterkey
```

5.12.3 Encrypting a Tablespace

If you create an encrypted tablespace, configure the encryption algorithm in runtime parameters. For example, to create a tablespace named `secure_tablespace` using AES with a 256-bit key length as the encryption algorithm, define:

```
-- Specify the encryption algorithm for the tablespace to be created below
SET tablespace_encryption_algorithm = 'AES256';
CREATE TABLESPACE secure_tablespace LOCATION '/database/tablespaces/tbpace1';
-- Specify that the tablespace to be created below is not to be encrypted
SET tablespace_encryption_algorithm = 'none';
```

Or

```
CREATE TABLESPACE tbs_tst_new LOCATION '/database/tablespaces/tbpace1' WITH
(tablespace_encryption_algorithm = 'AES256' );
```

Checking for encrypted tablespaces

You can check which tablespaces are encrypted by executing the following SQL.

```
SELECT spcname, spcencalgo FROM pg_tablespace ts, pgx_tablespaces tsx WHERE ts.oid =
tsx.spctablespace;
```

5.12.4 Backup/Restore

In case the FEP cluster is damaged or lost, backups should be made at the following times:

- When the cluster is first created
- When the master encryption key is changed

When you use the FEPRestore custom resource to create a cluster restored from backup, the restored cluster is restored with the master encryption key at the time the backup was taken on the source cluster (where the backup was created from).

If a newer master encryption key is specified in `sysTde.tdek.targetKeyId` than when the source FEPCluster was backed up, the value will be carried over to the restore destination FEPCluster custom resource, and the operator automatically re-enables TDE with the new master encryption key after data recovery.

Also, update the authentication information to the key management system before executing the restore. If your credentials are not up-to-date, FEP will not be able to connect to the key management service and restore your data.

If you mistakenly update the information for connecting to the key management system under `sysTde.tdek.kmsDefinition` after building FEPCluster, FEP will not be able to refer to the key management system when restoring data. Before executing the restore process, confirm that the correct values are described in the FEPCluster custom resource.

5.12.5 Changing Key Management System Definitions

Modify the parameters under `spec.fepChildCrVal.sysTde.tdek.kmsDefinition` in the FEPCluster custom resource if you want to add or change the connection information to the key management system.

If you make any of the following changes, the replica server will be restarted with the new parameters:. If there are multiple replica servers, they are restarted one at a time. When all replica servers are restarted, one of them is promoted to the new master server due to a switchover. The original master server's container image is then restarted. This allows you to change the definition of the key management system for all servers with minimal disruption.

- Add a new key management system definition
- Delete an existing key management system definition
- Change the order of key management system definitions
- Add, Delete, or rename ConfigMap or Secret resources that you specify as credentials

If you make changes that require a restart, temporarily disable the automatic scale out feature for the database before making the changes. The automatic scale out feature can be disabled with the `spec.fepChildCrVal.autoscale.scaleout.policy` parameter of the FEPCluster custom resource.

You cannot rename the ConfigMap/Secret resource that you currently specify as the credential for the key management system you are using as the keystore.

5.13 Confidentiality Management Feature

5.13.1 Enabling Confidentiality Management Feature

When building FEPCluster, the extension "pgx_confidential_management_support" of the confidentiality management feature was installed and set up in the following database.

- template1
- postgres
- Database specified in `spec.fepChildCrVal.sysUsers.pgdb`

In addition, when creating a confidential administrator role (`spec.fepChildCeVal.sysUsers.pgSecurityUser`), this role is assigned the following functions necessary for executing confidentiality management feature.

- CREATE ROLE
- SELECT, INSERT, UPDATE, and DELETE privileges on all tables included in the extension

Therefore, immediately after FEPCluster is built, database objects can be managed by the confidentiality management feature in a database in which the extension "pgx_confidential_management_support" is installed or in a database created from template1.

Refer to "Confidentiality Management" in the Fujitsu Enterprise Postgres Security Operation Guide for details on how to operate the security management support function.

Refer to "Tables Used by Confidentiality Management Feature" in the Fujitsu Enterprise Postgres Security Operation Guide for tables included in the extension.

In addition, if a database role other than the confidential administrator role needs to operate the confidentiality management feature, such as by preparing a database role for each schema that manages database objects using the confidentiality management feature, the confidentiality management feature assign the following privileges to the database role.

- CREATE ROLE
- SELECT, INSERT, UPDATE, and DELETE privileges on all tables included in the extension

When using the confidentiality management feature to manage database objects created by other users, it is necessary to grant ownership of the database objects to the database role that operates the confidentiality management feature.

Example) When giving ownership of the table "security_table" to the confidential administrator user "security_user"

```
ALTER TABLE security_table OWNER TO security_user;
```

The owner of the database object can be confirmed using the PostgreSQL meta-command "\d".

5.13.2 Monitoring Confidentiality Management Feature

You can forward pgAudit's audit logs to Elasticsearch using the FEP logging feature.

Analyze the transferred audit log and monitor for changes in privileges on database objects by unintended users.

For the FEP log feature, please refer to "[4.9 FEP Logging](#)".

5.14 Operation of Fixed Statistics

Describes how to periodically fix statistics in the production environment using statistics whose performance is guaranteed in the validation environment.

5.14.1 Preparing for Object Storage Connections

Deploy a secret that contains the credentials required to operate object storage.

5.14.1.1 Using S3

Secret - Name: my-aws-s3-secret
data: aws_access_key: cG9zdGdyZXM= aws_access_secret: ZnNlcA3A3A==

5.14.1.2 Using AzureBlob

Secret - Name: my-azure-blob-secret
data: azure_storage_account_name: cG9zdGdyZXM= azure_storage_account_key: ZnNlcA3A3A==

5.14.1.3 Using Google Cloud Storage

Deploy the json file required for authentication with the following command:

```
kubectl create secret generic <secret> --from-file=key.json=<json file>
```

5.14.2 Storing validation environment statistics in object storage

Run FEPAction to export the validation environment statistics and store them in object storage. Define the connection information to object storage in the FEPCluster custom resource before executing FEPAction.

The following is an example definition of FEPAction that stores a binary file that exports the database "mydatabase" in object storage AWS s3.

Example) Definition example of FEPCluster custom resource

```
spec:  
  fep:  
    fixedStats:
```



```
endpoint: #Required
protocol: s3
authentication: aws-fixedstats-credentials
```

Example) Definition example of FEPAAction custom resource

```
apiVersion: fep.fujitsu.io/v1
kind: FEPAAction
metadata:
  name: fep-action-fixdStats
spec:
  fepAction:
    targetClusterName: new-fep #target cluster
    type: fixed_stats
    args:
      fixedStatsType: export
      targetDb: mydatabase
      url: 's3://export_stats/file'
      targetStats: effective
```

5.14.3 Schedule for Fixed Statistics

Define the object storage information and schedule where the statistics are stored in the FEPCluster custom resource in the production environment. You can also add, modify, or delete schedules after they are built.

The following is an example of the FEPCluster custom resource definition for each Friday from 20:00 to 23:00 when the database “mydatabase” is pinned with statistics stored in object storage AWS s3.

Example) Definition example of FEPCluster custom resource

```
spec:
  fep:
    fixedStats:
      schedule1:
        update: true
        fixSchedule: "0 20 * * 5"
        unfixSchedule: "0 23 * * 5"
        targetDb: mydatabase
        url: 's3://export_stats/file '
      endpoint:
        protocol: s3
        authentication: my-aws-s3-secret
```

Once the statistics are downloaded from object storage, they are backed up to the database. Also, spec.fep.fixedStats.schedleN.update will be from true to false.

If you want to update the statistics that you want to freeze, change spec.fep.fixedStats.schedleN.update to ture and run the statistics import from object storage again.

5.14.4 Using Local Storage

If object storage is not available, use local storage.

5.14.4.1 Exporting Statistics Files to FEPCluster in a Validation Environment

Execute FEPAAction and export the statistics whose performance is guaranteed in the verification environment to the file system /database/userdata/action/local/ on the container.

Example) Definition example of FEPCluster custom resource

```
spec:
  fep:
```

```
fixedStats:
  endpoint: #Required
  protocol: local
```

Example) Definition example of FEPAAction custom resource

```
apiVersion: fep.fujitsu.io/v1
kind: FEPAAction
metadata:
  name: fep-action-fixdStats
spec:
  fepAction:
    targetClusterName: new-fep
    type: fixed_stats
    args:
      fixedStatsType: export
      targetDb: mydatabase
      file: file_name.dump
      targetStats: effective
```

5.14.4.2 Deploying Statistics Files to FEPCluster in a Production Environment

Deploy the statistics binary file under /database/userdata/fixedstats/action/local/ in the primary database of FEPCluster in the production environment.

```
kubect1 cp <file-to-copy> \
<namespace>/<primary-pod-name>:/database/userdata/fixedstats/action/local/ -c fep-patroni
```

5.14.4.3 Specify File Names for FEPCluster Custom Resources

Specify the name of the deployed statistics binary file for the FEPCluster custom resource.

Example) Definition example of FEPCluster custom resource

```
spec:
  fep:
    fixedStats:
      schedule1: #Required, N is the identification number
      update: true
      fixSchedule: "0 20 * * 5" #Required
      unfixSchedule: "0 23 * * 5" #Optional, If not specified, do not unfreeze
      targetDb: mydatabase #Required
      file: file_name.dump
    endpoint:
      protocol: local
```

Chapter 6 Maintenance Operations

This chapter describes the maintenance operation after deploying the container.

6.1 Minor Version Upgrade

This section explains the steps for minor upgrade of a container image.

Before upgrading each container version, please refer to "[Chapter 3 Operator Installation](#)" to update your operators. Also, please refer to RedHat's ecosystem catalog for updated information on each container.

You can upgrade the container image by changing the parameters of the FEPCluster custom resource.

Parameter	Description
spec.fep.image.image	Update the FEP container image.
spec.fepChildCrVal.backup.image.image	Update the backup container image.
spec.fep.feputils.image	Update the container image responsible for cloud secret management feature.
spec.fep.remoteLogging.image	Update the container image that handles the Log Collection feature.

After changing the custom resource parameters, if you have multiple FEPPods, restart the Pods one by one from the replica server to upgrade the container image. Once all replica servers are upgraded, one of them will be promoted to master server. The old master server's Pods are then restarted and the container images are upgraded.

These rolling upgrades allow you to upgrade all container images with minimal disruption to your business.



Note

The upgrade process will cause an outage on the cluster for the duration to upgrade both Master and Sync Replica. If there is no Sync Replica in the cluster, the outage is limited to the length of time to upgrade the Master (or actually the failover time required to take another replica been promoted by patroni).

6.2 Cluster Master Switchover

You can switch a master instance to a replica instance in the event of a master instance performance failure or planned node maintenance.

Specify "switchover" for the action type of the FEPAAction CR to update FEPAAction CR.

Equivalent Kubernetes command: `kubectl apply -f <new_spec>`

The "switchover" action type requires the user to specify the name of the target cluster on which to perform the switchover. The args section is not needed for switchovers, as FEPAAction internally identifies the pod to switch from and promotes a new master pod.

```
spec:
  fepAction:
    type: switchover
    targetClusterName: new-fep
```

Refer to "FEPAAction Custom Resource Parameters" in the Reference for more information on parameters.

6.3 Perform PITR and the Latest Backup Restore from Operator

It can be used to restore a database to a specific location due to an application failure or to prepare a duplicate database for production.

Restore process can restore data by creating a CR (FEPRestore CR) for the restore as follows:

`oc create -f [Custom Resource Files]`

Example)

```
$oc create -f config/samples/postgres_v1_restore.yaml
```

There are two methods of restoring: restoring data to an existing FEPCluster or restoring data to a new FEPCluster.

When restoring to an existing FEPCluster, information such as the FEPCluster name, IP address, and various settings remain the same.

If you restore to a new FEPCluster, the FEPCluster name is the one you specified in CR and the new IP address is also given. If the setting value is not specified, the new cluster will inherit the settings from the restore source cluster, but you can change the settings to create a new cluster by specifying them in CR.

6.3.1 Setting Item

Refer to "FEP Restore Custom Resource Parameters" in the Reference for the items to be set in a custom resource file.

6.3.2 After Restore

Switching connections to the new cluster

The restore creates a new FEPCluster. If necessary, you need to set up Pgpool-II and change the access point of the application to the new cluster or the new Pgpool-II.

Backup data of the destination cluster

PITR restores to the pre-restore time are not possible, because the backup of the destination cluster begins after the restore completes.

6.4 Major Version Upgrade

Describes the procedure for upgrading the major version of the operator and FEP container.

A major version upgrade of a FEP builds a new major version of the FEP in the same Namespace as the previous major version of the FEP. At this time, by defining the "spec.fepChildCrVal.upgrade" field in FEPClusterCR, the operator creates the upgrade execution container. The upgrade execution container uses the previous version of FEP Cluster specified in "spec.fepChildCrVal.upgrade.sourceCluster" as the data source FEPCluster and migrates the data to the newly created FEPCluster.

6.4.1 Pre-work on the Data Source FEP Cluster

Stop the running business application before executing the major version upgrade.

Next, edit "spec.fepChildCrVal.customPgHba" of the data source FEPCluster Custom Resource to allow the connection of the upgrade execution container.

The addresses that are allowed to connect are specified as follows:

```
<fep>-upgrade-pod.<fep>- upgrade-headless-svc.<namespace>.svc.cluster.local
```

<fep> specifies the name of the newly created FEPCluster Custom Resource.

The authentication method can be either trust/md5/cert.

Example of Editing a FEPCluster Custom Resource in a Data Source:

```
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  name: source-fep
  namespace: my-namespace
spec:
  fepChildCrVal:
    customPgHba: |
      host all all destination-fep-upgrade-pod. destination-fep-upgrade-headless-svc. my-
```

```
namespace.svc.cluster.local trust
...
```

6.4.2 Operator Upgrade

Describes the instructions for upgrading the operator.



Note

After an operator upgrade, any custom resource configuration changes you defined in the previous version are not reflected in the container.

6.4.2.1 Uninstalling the Old Operator

Uninstall the old operator.

Select "Uninstall Operator" from "Operators"> "Installed Operators"> "Fujitsu Enterprise Postgres <Old version> Operator"> Actions.

6.4.2.2 Installing a New Version of the Operator

Refer to "[Chapter 3 Operator Installation](#)" to install the new version of the operator.

6.4.3 Major Version Upgrade of FEP

6.4.3.1 Creating a New FEPCluster CR

Refer to the Reference to define a new major version of the FEPCluster custom resource. At this time, allow the running upgrade container to connect as you did in "[6.4.1 Pre-work on the Data Source FEP Cluster](#)".

In addition, a major version upgrade of FEP is performed by defining the "spec.fepChildCrVal.upgrade" field, as in the following example of defining a FEPCluster custom resource.

The upgrade execution container uses PV to store dump files retrieved from the FEPCluster of the data source.

If you have not enabled the automatic PV provisioning feature in your Kubernetes environment, create a PV for the upgrade in addition to the new PV for the FEPCluster before creating the FEPCluster custom resource.

Also, edit "spec.fepChildCrVal.customPgHba" to allow the connection of the upgrade execution container, as in "[6.4.1 Pre-work on the Data Source FEP Cluster](#)".

Example of Defining a FEPCluster Custom Resource to Perform an Upgrade:

```
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  name: destination-fep
  namespace: my-namespace
spec:
  fep:
    ...
  fepChildCrVal:
    upgrade
      sourceCluster: source-fep-cluster
      storage:
        size: 8Gi
      customPgHba: |
        host all all destination-fep-upgrade-pod.destination-fep-upgrade-headless-svc.my-
namespace.svc.cluster.local trust
    ...
```

FEPCluster Custom Resource Fields "spec.fepChildCrVal.upgrade"

Field	Default	Details
spec.fepChildCrVal.upgrade		Optional When this field is defined, a major version upgrade is performed. However, if spec.fepChildCrVal.restore is defined, the FEPCluster build stops.
spec.fepChildCrVal.upgrade.sourceCluster		Specify the FEPCluster CR name of the data migration source. Be sure to specify spec.fepChildCrVal.upgrade when defining it.
spec.fepChildCrVal.upgrade.mcSpec.limits	cpu: 200m memory: 300Mi	Optional Specify the maximum number of resources allocated to the upgrade execution container.
spec.fepChildCrVal.upgrade.mcSpec.requests	cpu: 100m memory: 200Mi	Optional Specify the lower limit of resources allocated to the upgrade execution container.
spec.fepChildCrVal.upgrade.image		Optional If omitted, the URL of the image is obtained from the operator container environment.
spec.fepChildCrVal.upgrade.imagePullPolicy	IfNotPresent	Optional Specify the pull policy for the container image. - Always - IfNotPresent - Never
spec.fepChildCrVal.upgrade.source.pgAdminTls.certificateName		Optional If the data source FEPCluster used "cert" as the authentication method for the Upgrade Execution Container, use the secret certificate that defines spec.fepChildCrVal.sysUsers.pgAdminTls.certificateName for the data source FEPCluster. If the above parameter is not defined, it points to the Kubernetes TLS secret containing the certificate of the Postgres user "postgres" in the data source.

Field	Default	Details
		Refer to " 4.7.2.1 Manual Certificate Management " for information about creating secrets.
spec.fepChildCrVal.upgrade.destination.pgAdminTls.certificateName		Optional If the newly created FEPCluster used the "cert" authentication method for the running upgrade container, use the secret certificate that defines the spec.fepChildCrVal.sysUsers.pgAdminTls.certificateName of the newly created FEPCluster. If the above parameter is not defined, it points to the Kubernetes TLS secret containing the certificate of the newly created Postgres user "postgres". Refer to "4.7.2.1 Manual Certificate Management" for information about creating secrets.
spec.fepChildCrVal.upgrade.storage		Optional Defines storage for storing dump files.
spec.fepChildCrVal.upgrade.storage.storageClass		Optional If omitted, the default storage class of the operating environment will be used.
spec.fepChildCrVal.upgrade.storage.size	2Gi	Optional Specify the size of the storage to store the dump file.
spec.fepChildCrVal.upgrade.storage.accessModes	ReadWriteOnce	Optional Storage access mode for storing dump files As an array of access modes. e.g. [ReadWriteMany] If omitted, it is treated as [ReadWriteOnce].



Note

Connect to the database and run the following SQL to check the size of the database in advance:

```
$ SELECT pg_size_pretty(sum(pg_database_size(datname))) AS dbsize FROM pg_database;
```

Since the pg_dumpall command used in the upgrade execution container outputs the database data as an SQL command, the file actually created is as follows.

For example, the integer type 2147483647 is 4 bytes for database data.

However, this is 10 bytes because SQL commands output them as strings. Therefore, make sure that the storage (PV) for dump files has sufficient disk space.

6.4.3.2 Verifying FEP Major Upgrade Complete

If you migrate your data to the new FEPCluster and the FEP major version upgrade is successful, the following event will be output:

```
$ kubectl get event
LAST SEEN   TYPE      REASON              OBJECT                                          MESSAGE
164m        Normal    SuccessfulFepUpgrade  fepupgrade/<Name of the new FEPClusterCR>    <namespace>,
Successfully FEP Upgrade
```

In addition, the following annotation will be added to YAML in FEPClusterCR:

```
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  annotations:
    FEPUpgradeDone: true
...
name: destination-fep-cluster
namespace: my-namespace
spec:
...
```



Note

When a major upgrade of FEP fails, an event similar to the following is output:

```
$ kubectl get event
LAST SEEN   TYPE      REASON              OBJECT                                          MESSAGE
164m        Warning    FailedFepUpgrade    fepupgrade/<Name of the new FEPClusterCR>    <namespace>, Error/
Failure in FEP Upgrade
```

Obtain the Kubernetes resource information listed in the OBJECT column, review the output messages, and then recreate the new FEPCluster custom resource.

```
$ kubectl describe fepupgrade/<Name of the new FEPClusterCR>
```

6.4.4 Updating Each Custom Resource

Describes the procedures for each custom resource used to operate the FEPCluster for the data source after the major FEP upgrade is complete.

After this process is complete, resume the suspended business applications.

6.4.4.1 Removing a FEPClusterCR for a Data Source

Delete the FEPCluster for the data source.

For the Openshift GUI console:

From "Operators" > "Installed Operators" > "Fujitsu Enterprise Postgres < New version > Operator" > "FEPCluster" > "FEPCluster name to delete" > Actions, select "Delete FEPCluster".

6.4.4.2 FEPPgpool2

Re-create FEPPgpool2 to match the version of the client with the version of the upgraded FEP.

6.4.4.3 FEPEXporter Built in Standalone Mode

Edit the FEPEXporter custom resource "spec.fepExporter.fepClusterList" to specify the new version of the FEPCluster custom resource. Refer to "FEPEXporter Custom Resource" in the Reference for more information about the parameters.

6.5 Assigned Resources for Operator Containers

The following resources are allocated by default to the operator containers provided by this product.

```
resources:
limits:
  cpu: 2
  memory: 1536Mi
requests:
  cpu: 500m
  memory: 768Mi
```

If there is only one FEPCluster custom resource managed by an operator, it can be operated with the resource assigned by default. However, when deploying and operating multiple FEPCluster custom resources, change the assigned resource of the operator container.



Note

If you have changed the resource, the resource value will revert to the default value after the operator version upgrade. Therefore, change the resource again after upgrading the operator.

6.5.1 How to Change Assigned Resources

Describes how to change the resources assigned to an operator container.

When updating resources assigned to an operator container, the operator container is recreated. At this time, the operation of already built containers such as FEPCluster will not stop.

How you change the allocated resources depends on how the operator was installed.

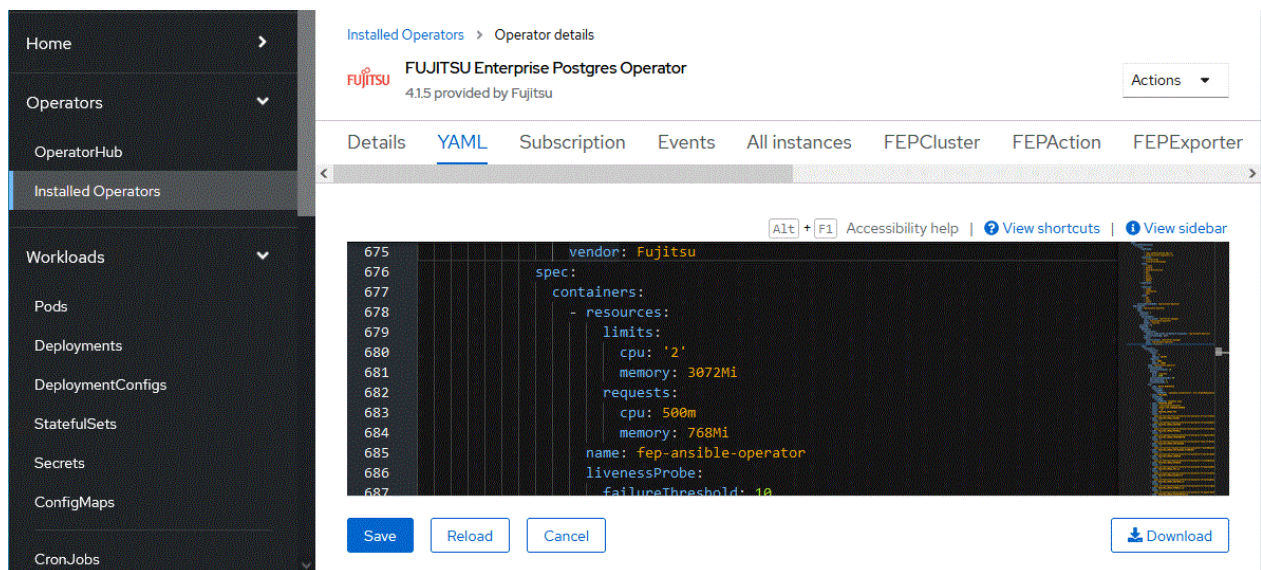
6.5.1.1 When installing using OperatorHub

If you are using an operator installed from OperatorHub To change the resources assigned to the operator container, edit the ClusterServiceVersion (CSV).

Editing the CSV "spec.install.spec.deployments[0].spec.template.spec.containers[0].resources" will recreate the operator container and apply the specified resources.

When editing CSV from the OCP GUI console

Click [Installed Operators] in the menu item under Operators and select the installed operator. On the [YAML] tab, edit the specified part of the allocation resource and click [Save].



When editing CSV from the CUI console using the OC client

Check the CSV name of the installed operator with the "oc get" command.

```
$ oc get csv
NAME                                DISPLAY                                VERSION  REPLACES  PHASE
fujitsu-enterprise-postgres-operator.v4.1.5  Fujitsu Enterprise Postgres Operator
4.1.5                                     Succeeded
```

Edit the CSV with the "oc edit" command.

```
$ oc edit csv fujitsu-enterprise-postgres-operator.v4.1.5
```

6.5.1.2 When installing using Helm Chart or RancherUI

If the operator is installed using Helm Chart or RancherUI, edit the deployment of the operator container to change the resources assigned to the operator container.

Editing the Deployment's "spec.template.spec.containers[0].resources" will recreate the operator container and apply the specified resources.

Edit the Deployment "fep-ansible-operator" with the "kubectl edit" command.

```
$ kubectl get deployment fep-ansible-operator
NAME                READY  UP-TO-DATE  AVAILABLE  AGE
fep-ansible-operator  1/1    1            1          2m10s

$ kubectl edit deployment fep-ansible-operator
```

6.6 Using SUPERUSER Privilege

6.6.1 CREATE EXTENSION

When executing the CREATE EXTENSION command to install external extensions for PostgreSQL, there are extensions that can only be installed by SUPERUSER. To install such extensions we make use of the FEPAAction custom resource.

By specifying "create_extension" in spec.fepAction.type of FEPAAction custom resource, CREATE EXTENSION can be executed for the specified FEPCluster container.

Please refer to the Reference for how to use.

6.6.2 Change Password of SUPERUSER

To update the password of SUPERUSER "postgres", specify update_admin_password for fepActionType of the FEPAction custom resource.

Recreate the password with a random value and update it.

Please refer to the Reference for how to use.

6.6.3 Using SUPERUSER

If SUPERUSER privileges are required for database operation, you can obtain the password for SUPERUSER "postgres" by following the steps below.

1. Get the base64-encoded password from the Secret with the same name as the FEPCluster custom resource name.

Example) When the FEPCluster custom resource name is new-fep

```
$ kubectl get -o yaml secret new-fep | grep PG_ADMIN_PASSWORD
PG_ADMIN_PASSWORD: YWRtaW4tcGFzc3dvcmQ=
```

2. Decode the obtained password.

```
$ echo YWRtaW4tcGFzc3dvcmQ= | base64 -d
admin-password
```



Note

.....

In order to prevent SUPERUSER from being used by a third party, please set Kubernetes Role permissions to the Secret so that only the database administrator can refer it.

.....

Chapter 7 Abnormality

This chapter describes the actions to take when an error occurs in the database or an application, while FEP is operating.

Depending on the type of error, recover from the backed-up material, reserve capacity, check the operator log, and check the FEP log.

7.1 Handling of Data Abnormalities

Recover the database cluster from the backup immediately prior to failure in any of the following cases:

- A hardware failure occurs on the data storage disk or the backup data storage disk.
- If the data on the disk is logically corrupted and the database does not work correctly
- Data corruption caused by user error

Refer to "[6.3 Perform PITR and the Latest Backup Restore from Operator](#)" for restore instructions.

7.2 Handling when the Capacity of the Data Storage Destination or Transaction Log Storage Destination is Insufficient

If you run out of space in the data storage location, first check if there are any unnecessary files on the disk, and then delete them so that you can continue working.

If deleting unnecessary files does not solve the problem, you may need to migrate the data to a larger disk.

Use a backup restore to migrate data.

You can use the FEPRestore custom resource to build a new FEPCluster and migrate data. Refer to "FEPRestore Custom Resource Parameters" in the Reference.

7.3 What to do when the Capacity of the Backup Data Storage Area is Insufficient

If you run out of space in the backup data destination, first check the disk for unnecessary files, and then delete the unnecessary files. Or reduce the backup retention generation.

By specifying backup_expire in spec.fepAction.type of the FEPAction custom resource, you can reduce the number of backup generations saved. Refer to "FEPAction Custom Resource Parameters" in the Reference for details.

7.4 Handling Access Abnormalities When Instance Shutdown Fails

If an instance fails to start or stop, refer to the Operator log and the FEP log to determine the cause.

For checking the operator log and the FEP log, refer to "[7.5 Collection of Failure Investigation Information](#)".

7.5 Collection of Failure Investigation Information

If the cause of the trouble that occurred during the construction or operation of the environment is not identified, information for the initial investigation is collected.

I will explain how to collect information for the initial investigation.

- Product log
- Operator log

Product log

FEP log

Get into the container and collect the log.

The log location is specified by `log_directory` in the custom resource `FEP Clusterspec.startupValues.customPgParam` parameter. The default is `/database/log`.

Pgpool-II log

Get into the container and collect the log.

The log location is `/var/log/pgpool/pool.log`.

Operator log

Check the operator log as follows.

Verification Example

```
$oc get po
NAME                                READY   STATUS    RESTARTS   AGE
fep-ansible-operator-7dc5fd9bf7-4  1/1     Running   0           20m
```

How to check the log

```
$oc logs pod fep-ansible-operator-7dc5fd9bf7-4 smzk -c manager
```

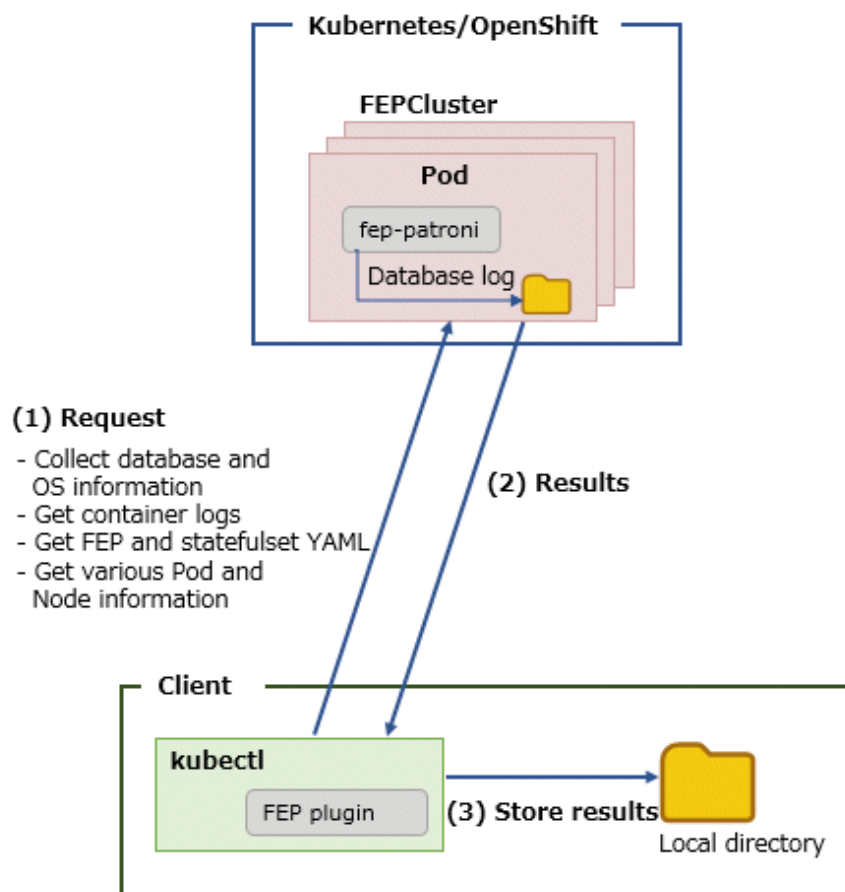
The log will be output to the console. Please check the file output by redirection.

7.6 Log Collection Tool

The Log Collection Tool is a client tool for the FEP container environment, that allows logs, and other useful environment and statistical information, to be retrieved from FEP containers running on a specified FEP cluster on Kubernetes/OpenShift. The collected information is stored locally in a filesystem directory on the client machine, for subsequent analysis.

The purpose of the tool is to obtain helpful diagnostic and troubleshooting information, in the case that an FEP containers is, for example, non-responsive, has terminated abnormally or exhibits abnormal performance.

The Log Collection Tool collects container-specific information as well as database and OS information.



7.6.1 Logs Collected

The following logs and information are collected by the Log Collection Tool:

Type of log(s)	Source
OS information and statistics, FEP catalog contents, database logs	fep-patroni (FEP server) container, in each pod
FEP yaml	FEP cluster
Statefulset yaml	FEP cluster
Container logs	Log from each FEP-related container running in the FEP cluster ("kubectl logs ..." output)
Detailed pod information, for the namespace	"kubectl get pods -n <namespace> -o wide" output
Detailed node information, for the Kubernetes cluster	"kubectl get nodes -o wide" output
Pod CPU and memory utilization, for the namespace	"kubectl top pod -n <namespace>" output
Node CPU, memory, and resource utilization, for the Kubernetes cluster	"kubectl top node" output

7.6.2 Location and Dependencies

The Log Collection Tool is the kubectl plugin "kubectl-fep". Add Log Collection Tool as subcommands to kubectl to extend the command line syntax.

Log Collection Tool are stored in the operator's Pod. Install it on the Kubernetes client machine using the command below. <destination-path> specifies the storage location of the plugin.

```
1. Check the OperatorPod name with the command below.
$ kubectl get pods --selector=name=fep-ansible-operator -o name
pod/fep-ansible-operator-5985969857-ldj2x

2. Copy the plugin.
# Linux Client
$ kubectl cp fep-ansible-operator-5985969857-ldj2x:/opt/fepopr-kubectl-plugin/kubectl-fep
<destination-path>/kubectl-fep

# Windows Client
$ kubectl cp fep-ansible-operator-5985969857-ldj2x:/opt/fepopr-kubectl-plugin/kubectl-fep.ps1
<destination-path>/kubectl-fep.ps1
```

The Log Collection Tool kubectl plugin is installed on client machines along with the kubectl/oc command for remote access to the target Kubernetes/OpenShift environment. Specify the directory where both the kubectl plugin and the kubectl/oc command are stored in the PATH environment variable. Also, set appropriate execution permissions for the plugin.

On a client machine, the kubectl/oc command supports different methods of authentication/login to the Kubernetes Cluster (e.g., username + password, TLS client certificates, OAuth token) that allow different users to execute the command, provided they have suitable privileges to the Kubernetes Cluster.

The diagnostic logs collected by the Log Collection Tool are stored locally on the filesystem on the client.

7.6.3 User Interface

The Log Collection Tool is invoked through the kubectl plugin command-line interface.

7.6.3.1 Command-Line Syntax

The command-line syntax of the Log Collection Tool is shown below.

The Log Collection Tool corresponds to the "cluster logs" subcommand of the FEP kubectl plugin.

There is a companion "cluster list" subcommand which displays the list of FEP clusters in the specified namespace.

Usage:

```
kubectl fep cluster logs -n <NAMESPACE> [-o <OUTPUT-DIRECTORY>] [-d <DATABASE>] <FEP-CLUSTER>
kubectl fep cluster list -n <NAMESPACE>
```

Note that for the OpenShift Container Platform, the OpenShift "oc" command can be used as an alternative to "kubectl", as it has the same features as the kubectl binary, but it has been further extended to natively support the OCP features.

Windows considerations

The command-line syntax for invoking the FEP kubectl plugin (but not supplying its command parameters) may differ between Operating Systems, due to the type of executable that the plugin is implemented as.

For example, under Windows, since the FEP kubectl plugin is implemented as a Powershell script the plugin needs to be invoked directly, rather than indirectly from kubectl/oc.

From a Command Prompt:

Powershell 5.1:

```
powershell kubectl-fep ...
```

Powershell 7+:

```
pwsh -command kubectl-fep ...
```

From a Powershell Window:

```
kubectl-fep ...
```

An additional consideration is the current Powershell "ExecutionPolicy" in effect, which may block Powershell scripts from being run for security reasons, if they are not signed. In this case, since the FEP kubectl plugin is a trusted script, the ExecutionPolicy can be bypassed for the script execution, using the following syntax:

Powershell 5.1:

```
powershell -executionpolicy bypass kubectl-fep ...
```

Powershell 7+:

```
pwsh -executionpolicy bypass -command kubectl-fep ...
```

7.6.4 Log Collection Directory

7.6.4.1 Command-Line Specification

The Log Collection Tool output directory may optionally be specified using the -o flag, and if the directory exists, it must be empty. If the output directory is not specified, the following default output directory is used:

Linux:

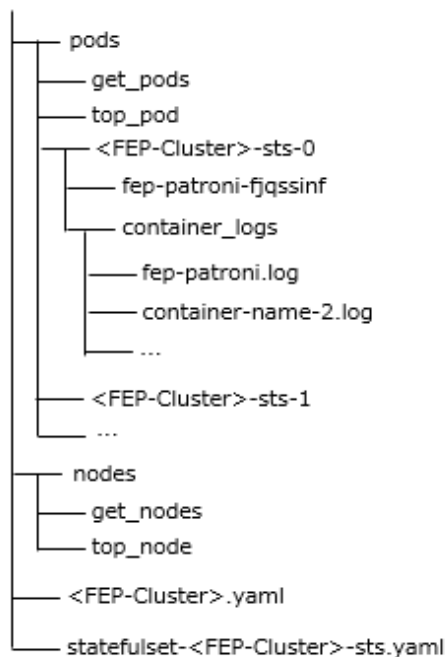
```
/tmp/fep_cluster_logs_<namespace>_<cluster>_<date>_<time>
```

Windows:

```
%TEMP%\fep_cluster_logs_<namespace>_<cluster>_<date>_<time>
```

7.6.4.2 Directory Structure

The structure of the output directory of the Log collection Tool is as follows.



Appendix A Quantitative Values and Limitations

A.1 Quantitative Values

Refer to the Fujitsu Enterprise Postgres Installation and Setup Guide for Server.

A.2 Limitations

Note

If you log in to a container and edit the configuration file directly, restarting the container may undo your changes.

If you want to change the settings, modify the custom resource files as described in "[5.2 Configuration Change](#)" and reapply. Depending on the parameters to be changed, the container may be redeployed. Refer to "[5.2 Configuration Change](#)" for details of the parameters.

Unavailable FEP features

Since FEP server container is based on other components (like UBI and Patroni), there are certain limitations that doesn't allow it to be 100% functionally capable to VM based server instance. The known limitations are as below.

No	Limitation	Reason for Limitation	Description
1	Crypto Express cards are not supported	IBM LinuxOne doesn't support CryptoExpress cards in Openshift container platform at this stage.	FEP TDEz extension cannot be used on LinuxOne Openshift environment. However, User can still use TDE on both LinuxOne Openshift environment as well as Azure (x86) Openshift environment.

Fixed parameter

Some parameters cannot be changed. Refer to "[2.3.5.2 Parameters that cannot be Set](#)".

FEP features that needs to be set when using

Refer to "[2.3.7 FEP Unique Feature Enabled by Default](#)".

Appendix B Adding Custom Annotations to FEPCluster Pods using Operator

This section describes instructions for adding custom annotations to a FEPCluster pod.

1. In YAML view of the Create FEPCluster section, add custom annotations as below and then click on Create.

Project: fep14-install-test

FUJITSU Enterprise Postgres 14 Operator > Create FEPCluster

Create FEPCluster

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

Configure via: ☐ Form view ☒ YAML view

```
1  apiVersion: fep.fujitsu.io/v2
2  kind: FEPCluster
3  metadata:
4    name: new-fep
5    namespace: fep14-install-test
6  spec:
7    fep:
8      customAnnotations:
9        allDeployments:
10         annotation1: value1
11         annotation2: value2
12      forceSsl: true
13      image:
14        pullPolicy: IfNotPresent
15      instances: 1
16      mcSpec:
17        limits:
18          cpu: 500m
19          memory: 700Mi
20        requests:
21          cpu: 200m
22          memory: 512Mi
23      podAntiAffinity: false
24      podDisruptionBudget: false
25      servicePort: 27500
26      syncNode: 'off'
27      sysExtraLogging: false
28    fepChildCrVal:
29      backup:
30        image:
31          pullPolicy: IfNotPresent
32        mcSpec:
33          limits:
34            cpu: 0.2
```

[Alt + F1 Accessibility help](#) | [View shortcuts](#)

[Create](#) [Cancel](#) [Download](#)

- Both the Statefulset and its resulting pods will be annotated with your provided annotations: archivalVol and backupVol must be ReadWriteMany.

The screenshot shows the Red Hat OpenShift console interface. On the left is a sidebar with navigation menus: Administrator, Home, Operators, Workloads, and a list of resources including Pods, Deployments, Stateful Sets (selected), Secrets, Config Maps, Cron Jobs, Jobs, Daemon Sets, Replica Sets, Replication Controllers, and Horizontal Pod Autoscalers. The main panel displays the 'Stateful Set Details' for 'new-fep-with-cust-anno-sts' in the 'install-test' project. The 'YAML' tab is active, showing the following configuration:

```
1 kind: StatefulSet
2 apiVersion: apps/v1
3 metadata:
4   annotations:
5     annotation1: value1
6     annotation2: value2
7   statusCheckAt: 'Tue Sep 7 15:23:31 UTC 2021'
8   selflink: >-
9     /apis/apps/v1/namespaces/install-test/statefulsets/new-fep-with-cust-anno-sts
10  resourceVersion: '147317819'
11  name: new-fep-with-cust-anno-sts
12  uid: 269c6888-434d-4bde-b1d4-832c16ad521c
13  creationTimestamp: '2021-09-07T15:20:55Z'
14  generation: 1
15  managedFields:
16    - manager: OpenAPI-Generator
17      operation: Update
18      apiVersion: apps/v1
19      time: '2021-09-07T15:20:55Z'
20      fieldsV1: FieldsV1
21      fieldsV1:
22        'f:metadata':
23          'f:annotations':
```

At the bottom of the YAML editor are 'Save', 'Reload', and 'Cancel' buttons, and a 'Download' button on the right.

This screenshot shows the same Red Hat OpenShift console interface, but with the 'template' tab selected in the main panel. The configuration details are as follows:

```
535 name: new-fep-with-cust-anno
536 uid: 27037431-46a9-49eb-a723-3b8c2e8aab49
537 labels:
538   app: new-fep-with-cust-anno-sts
539   fepclustername: new-fep-with-cust-anno
540 spec:
541   replicas: 1
542   selector:
543     matchLabels:
544       app: new-fep-with-cust-anno-sts
545       fepclustername: new-fep-with-cust-anno
546   template:
547     metadata:
548       creationTimestamp: null
549     labels:
550       app: new-fep-with-cust-anno-sts
551       fepclustername: new-fep-with-cust-anno
552     annotations:
553       annotation1: value1
554       annotation2: value2
555     spec:
556       restartPolicy: Always
557       serviceAccountName: new-fep-with-cust-anno-sa
```

The interface includes the same sidebar and top navigation as the previous screenshot, with the 'YAML' tab also visible in the main panel's tab bar.

Appendix C Utilize Shared Storage

Explains how to build a FEPCluster when using shared storage.

Use a disk where PV accessModes can specify ReadWriteMany.

This chapter shows an example of using NFS as PV in static provisioning.

C.1 Creating a StorageClass

Create a StorageClass.

In the OCP WebGUI screen, click "StorageClass" in the main menu "Storage", then press "Create Storage Class" > "Edit YAML" and edit YAML to create the StorageClass.

If you are using the CLI, create a yaml file and create a StorageClass with the following command:

```
$ oc create -f <file_name>.yaml
```

YAML definitions are created with reference to the following samples.

Example)

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: < StorageClass Name >
provisioner: kubernetes.io/no-provisioner
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

C.2 Creating a PersistentVolume

Create as many PersistentVolumes (PV) as you need.

On the Web GUI screen, click "PersistentVolumes" in the main menu "Storage", click "Create PersistentVolume", and edit YAML to create PV.

If you are using the CLI, create a yaml file and create a PV using the following command:

```
$ oc create -f <file_name>.yaml
```

YAML definitions are created with reference to the following samples.

The StorageClass name specifies the StorageClass created in "[C.1 Creating a StorageClass](#)".

Assign a different NFS directory for each PV.

In addition, accessModes is ReadWriteMany.

Example)

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: < PV name >
spec:
  capacity:
    storage: < Capacity Required ex.8Gi >
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  mountOptions:
    - hard
  nfs:
```

```
path: < NFS directory path (Assign a different directory for each PV) ex. /nfs/pv >
server: < IP address of the NFS server ex. 192.168.1.10>
storageClassName: < StorageClass name created in "C.1 Creating a StorageClass">
```

C.3 Creating FEPCluster

Specifies that ReadWriteMany PV is used in the YAML definition in step 4 of "[4.1 Deploying FEPCluster using Operator](#)".

In spec.fepChildCRVal.storage, specify the StorageClass and AccessModes of the PV created in "[C.2 Creating a PersistentVolume](#)".

The "spec.fepChildCRVal.storage.<Volume Type>.size" should be less than or equal to the PV allocated.

Example) Using PV created by archivewalVol and backupVol

```
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  name: t3-fep
spec:
  ~ Suppress ~
  fepChildCrVal:
    storage:
      archivewalVol:
        size: < Capacity Required ex. 8Gi >
        storageClass: <StorageClass name created in C.1 Creating a StorageClass" >
        accessModes:
          - "ReadWriteMany"
      backupVol:
        size: < Capacity Required ex. 8Gi >
        storageClass: <StorageClass name created in C.1 Creating a StorageClass" >
        accessModes:
          - "ReadWriteMany"
  ~ Suppress ~
```

Appendix D Key Management System Available for Transparent Data Encryption

Describes the key management system available for transparent data encryption.

D.1 KMIP Server

Refer to "To Connect to a key Management System Using the KMIP Protocol" in the Fujitsu Enterprise Postgres Installation and Setup Guide for Server for KMIP server requirements.

D.2 AWS Key Management Service

D.2.1 Available Services

By using the AWS KMS adapter, you can use encryption keys on the Key Management Service (hereafter referred to as AWS KMS) provided by AWS. There is no region restriction as long as it is a region supported by AWS KMS.

D.2.2 Available AWS KMS Keys

The KMS key's key spec must be "symmetric". "asymmetric" keys cannot be used. Also, the KMS key usage must be ENCRYPT_DECRYPT.

D.2.3 Required Privileges

For the KMS key to be used, the following operations must be permitted for the user accessing AWS KMS.

- Encrypt
- Decrypt
- DescribeKey

D.2.4 Key ID

The following can be specified as key IDs for the TDE key management system linkage feature.

- Key ARN

D.3 Azure Key Management Service

D.3.1 Available Services

Accessible via Azure's Key Vault API using the Azure KMS Adapter, and a key management service is available that allows you to use symmetric keys.



Refer to below for key management services for which symmetric keys are available.

<https://learn.microsoft.com/en-us/azure/key-vault/keys/about-keys#key-types-and-protection-methods>

Refer to below for Azure key management services.

<https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management#azure-key-management-services>

D.3.2 Available Keys

A symmetric key is available.

D.3.3 Available Algorithms

The following algorithms are available during encryption/decryption operations.

- A256GCM

D.3.4 Key Operation

For the key to be used, the following operations must be permitted for the user who accesses Azure's key management service.

- encrypt
- decrypt
- get

D.3.5 Key ID

The following can be specified as key IDs for the TDE key management system linkage feature.

- Key object identifier

D.3.6 Sign In

Sign in to Azure using your service principal. You will need your application ID, tenant ID, and credentials to sign in.



See

.....
Refer to below for service principals.

<https://learn.microsoft.com/en-us/cli/azure/create-an-azure-service-principal-azure-cli#4-sign-in-using-a-service-principal>
.....