FUJITSU
Enterprise Postgres for Kubernetes
Technical presentation
Cloud and container markets growth

Kubernetes are gaining attention as virtualization technologies that make it easier to deploy and scale applications to shorten development cycles and streamline operations.

- Ideal for digital services infrastructure
- Responsiveness to rapid system changes and growth
- Reduced operational maintenance load
Fujitsu supports the transition to meet hybrid/multi-cloud requirements

FUJITSU Enterprise Postgres
- Enhanced security
- Enhanced performance
- Enhanced reliability

FUJITSU Enterprise Postgres for Kubernetes
- Reduced operational load
- Easy deployment
- No vendor-lock-in

Start small and grow
- Quickly deploy from development to operations
- Easily scale as your business grows

Reduced operational load
- Reduce the operational load on Database Administrators by automating operations such as failover, recovery, and backup.

Avoid vendor lock-in
- Leverage open container technology to move to the cloud without being locked to cloud vendors.

Security & reliability for business continuity
- Protects data from threats such as theft and falsification and ensures stable operation.

Ensured quality and compatibility
- Certified as a Red Hat OpenShift operator.
What is FUJITSU Enterprise Postgres Operator?

What is an Operator?
Operators automate the lifecycle of target software.

FUJITSU Enterprise Postgres Operator

- Deploys and manages the following in a Red Hat OpenShift Container Platform environment:
  - FEPCluster, pgpool2cluster, backup, restore, exporter
- Manages operations consistently according to predefined Custom Resources (CR).
  - CR is a YAML file that defines the configuration of a system.
Operator features

Provides operator services to automate the creation and operation of databases on your container management infrastructure.

Operators make it easy to deploy anywhere in multi-cloud and hybrid cloud environments.

<table>
<thead>
<tr>
<th>Operator features</th>
<th>Container deployment</th>
<th>HA features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Backup/Recovery</td>
<td>Configuration change</td>
</tr>
<tr>
<td></td>
<td>Minor version upgrade</td>
<td>FEP feature</td>
</tr>
<tr>
<td></td>
<td>Monitoring &amp; Alert</td>
<td>Scaling replicas</td>
</tr>
</tbody>
</table>

Red Hat OpenShift Container Platform

- Physical
- Virtual
- Private cloud
- Public cloud
Easy deployment

- Easily build highly available FUJITSU Enterprise Postgres clusters
  - The Operator execute requests in minutes and provides the functionality needed to take full advantage of FUJITSU Enterprise Postgres.
  - The Operator deploys standalone and highly available FUJITSU Enterprise Postgres clusters in pre-defined configurations and start with a small workload.
  Configuration parameters can be tuned during and after deployment to ensure that the instances are suitable for the workload.
Load balancing

- Deploying Pgpool-II and connecting to the cluster from Operator
  - Users can deploy the Pgpool-II container and access the database via Pgpool-II to use load-balancing and connection pooling features.
  - Multiple Pgpool-II containers can be deployed for load sharing and high availability. Users can request a Kubernetes service to distribute their work across multiple Pgpool-II containers.
High availability features
- Automatic failover
- Automatic recovery
- Manual switchover

Automatic failover overview
- High availability and failover management are provided by Patroni.
- If Patroni detects a failure in the cluster, such as the crash of a Postgres process or the outage of a container running the database, Patroni automatically conducts failover.
Backup and restore

- Reliable, affordable backup with object storage
  - The backup container is deployed as a complement to each server pod.
  - The backup runs at user-specified scheduled times (similarly to crontab).
  - pgBackRest is used for backup and WAL archiving.
    - Automatic backup
      - Full backup / incremental backup
    - Backup location
      - NFS fixed volumes
      - AWS S3 compatible storage
      - Red Hat OpenShift Container Storage
    - Point-in-time recovery (PITR)
      - Restore the cluster with point-in-time recovery from manual or automatic backup.
  - Restore type
    - Restore backup data to an existing cluster.
    - Create a new cluster and restore backup data.
Disaster Recovery - Overview

- Store backups in object storage for data recovery to different OCPs
  - configurations: multi-region, multi-vendor
  - RPO: Ensures data recovery in less than five minutes in the event of a disaster
  - RTO: data volume dependent
Disaster Recovery - Use case

- Data can be restored between OCPs built between different regions.
  - Example: When OCPs in the US East region are damaged, the database can be restored to OCPs in the US West region and continue operation.
- Ability to restore data between OCPs from different vendors
  - Example: Migrate on-premises OCP DB containers to cloud OCP
Disaster Recovery - configuration point

- Object Storage design
  - For disaster environments where storage to store backups must be designed to meet backup/restore requirements, the OCP environment and storage to store backups are in separate regions

- Restore time
  - Estimate business downtime in the event of a disaster by measuring restore times prior to operation, as it depends on the amount of data
Configuration changes

● Easy configuration and resource changes
  ● Changing parameters
    ● Configuration files
      ● postgresql.conf
      ● pg_hba.conf
      ● pgaudit.conf
    ● Depending on the parameter, the change will take effect in either of the following:
      ● Immediately after the change
      ● After the server process is restarted
  ● Modifying resources (CPU, memory)
    ● The allocated resources for the following containers can be changed by the FEPCluster custom resource:
      ● Server containers
      ● Backup containers
      ● Pgpool-II containers
Data security is one of the top concerns for enterprises – FUJITSU Enterprise Postgres for Kubernetes has the right tools for that.

**Transparent Data Encryption**

All data in FUJITSU Enterprise Postgres can be encrypted using Advanced Encryption Standard, a PCI DSS-compliant 256-bit encryption technology that is standard for the credit card industry.

**Data Masking**

Data masking minimizes security risk by enabling user-based confidentiality, altering original data while maintaining its usability. Redaction is applied via powerful, user-friendly policies.

**Dedicated Audit Log**

FUJITSU Enterprise Postgres extends PostgreSQL’s auditing by allowing the system to save audit records to a separate file – the Dedicated Audit Log.
Logical replication

- The logical replication feature of PostgreSQL is supported.
- Logical replication provides fine-grained control over data replication and security.

Use case
- Users can replicate data between different architectures, such as between public clouds and IBM LinuxONE™.
Scaling replicas - Overview

- Manual scale-out/scale-in
  - Scale out/in can be performed in the OpenShift web console in a few steps.
  - Database health can be also be checked using the web console

- Automatic scale-out
  - Dynamic database cluster expansion is conducted according to pre-defined scaling policies. Build and operational costs are reduced to respond to rapid transaction growth.
Scaling replicas – Use case

- Manual scale-in
  - Reduce pods manually during off-season with fewer transactions.

- Automatic scale-out
  - Maintain performance under temporary high load conditions.

![Diagram of scaling replicas]

- CPU utilization
  - High CPU usage (e.g., >80%)

- Automatic policy setting
- Manual request

- Operator

- Exporter pod

- Master pod

- Replica pod
Scaling replicas - Auto scale-out configuration

- Auto scale-out can be configured with the following threshold
  - Average CPU utilization of the cluster
  - Average number of connections to the cluster
- Maximum number of replicas that can be scaled out
  - Up to 15 replicas

Note
- When using the auto scale-out feature, consider the synchronous mode.
- When the workload on the system decreases, users should consider scaling in to reduce redundant resources. This is performed manually by editing FEPCluster CR. For details, refer to the FUJITSU Enterprise Postgres for Kubernetes User’s Guide.
The monitoring feature enables DBAs to manage the container environment to meet the system needs.

- Running the environment 24 x 7
- Maintaining desired database performance

Key monitoring items include:
- Physical resources
- Workload
- Connections
- Database resource / diagnostics info
- Load balance status
- Replication / backup status
- Liveliness
- Database server logs

Out-of-the-box real-time monitoring, also configurable:
- Monitoring items can be viewed on the OpenShift web console
- The monitoring report is integrated to Grafana
- Alerting can be set based on the monitoring metrics
- Trend analysis and historical data is available

Access control
- Providing a database log analysis feature using pgBadger
Monitoring and alert – Use cases

- Monitoring is provided by Prometheus and Grafana, the de-facto standard monitoring tools for Kubernetes.
- Prometheus, the monitoring software, collects CPU, memory, and disk usage resource information, as well as health status of cluster Pods. DBAs and Infrastructure Administrators can view monitoring data captured via Grafana features in an advanced graphical display.
- DBAs obtain a holistic view of their environment, from physical resources to workload
  - Alert-based proactive monitoring
  - Issue investigation driven by events (alerts)
  - Regular performance review
Monitoring – Reporting

● Contents of the report screen
  ● Select the Grafana URL from the Networking > Routes section of the OCP platform and report it on the Grafana dashboard screen.

● How data is managed
  ● Reports are archived every 2 hours and deleted after a specified period (default: 15 days)

● How to manage access
  ● Grafana allows you to setup access restrictions on a per-group basis. Read-only access and access allowing modifications to report configuration can be granted

### Default reporting metrics

<table>
<thead>
<tr>
<th>Type</th>
<th>Monitored metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Server resources</td>
<td>Average CPU usage, average memory usage</td>
</tr>
<tr>
<td>2 Database server</td>
<td>Version, database server start time</td>
</tr>
<tr>
<td></td>
<td>Effective cache, shared buffers, seq page cost, random page cost</td>
</tr>
<tr>
<td></td>
<td>Current update data, current insert data, current fetch data</td>
</tr>
<tr>
<td></td>
<td>Maintenance work mem, max worker processes</td>
</tr>
<tr>
<td></td>
<td>Max parallel workers</td>
</tr>
<tr>
<td>3 Database states</td>
<td>Conflicts/deadlocks, lock tables</td>
</tr>
<tr>
<td></td>
<td>Buffers (bgwriter), temp file (bytes), database size</td>
</tr>
<tr>
<td></td>
<td>Number of times of table ANALYZE by AUTOVACUUM</td>
</tr>
<tr>
<td></td>
<td>Checkpoint stats, cache hit rate</td>
</tr>
<tr>
<td></td>
<td>Number of tables without VACUUM in last 24 hours</td>
</tr>
<tr>
<td>4 Database transactions</td>
<td>Number of transactions lasting longer than 5 minutes</td>
</tr>
<tr>
<td></td>
<td>INSERT table data (SELECT), fetched database data, return data, DELETE data</td>
</tr>
<tr>
<td></td>
<td>UPDATE data, table UPDATED data</td>
</tr>
<tr>
<td></td>
<td>Transactions, dead rows, low performing queries</td>
</tr>
<tr>
<td></td>
<td>Sequential scans, index scans</td>
</tr>
<tr>
<td>5 Sizing</td>
<td>Schema sizes, data and WAL volume size</td>
</tr>
<tr>
<td>6 Connections</td>
<td>Active connection to max connection ratio</td>
</tr>
<tr>
<td></td>
<td>Max connections, connections in idle state for longer than 1 week</td>
</tr>
<tr>
<td>7 Database replication lag</td>
<td>Replication lag on Replica (in seconds)</td>
</tr>
</tbody>
</table>
Templates for Grafana dashboards are provided.

Report view settings
- Interval
- Namespace
- Exporter
- Database server
- Database
- Lock table
- FEP Pod name

If you want to change reporting metrics, change the metrics collected on Prometheus. If you want to change the graph display, edit it on Grafana.
Overview

- Early detection of signs of failure is important. Prometheus sends alerts with the Alert Manager (via email, Slack, or other tools) to DBAs and Infrastructure Administrators if the collected data (metrics) is unhealthy.
- Identify issues early and prevent problems before they occur.

Sample alert rules

- Server container/pod CPU usage is exceeding 80% of the resource limits
- Server container/pod memory usage is exceeding 80% of the resource limits
- PVC (volume) has less than 10% disk available
- Server apparently went down or is not accessible
Server log monitoring – Use case

- Alert notification is performed according to the level of log messages to immediately detect errors in the database.
  - In the past, errors in resources and database parameters could be detected, but by monitoring log messages, detailed errors in the database can be reported.
- It is possible to detect queries that affect database performance by using statistical reports obtained by analyzing log files.
  - Detection of slow queries and queries that execute frequently
  - Detecting queries generating many temporary files
  - Detecting Locked Queries
Server log monitoring - Configuration point

- **Log monitoring**
  - Set alert rules based on the severity of errors according to the customer's business operations.
    - Example: Immediate detection of FATAL/PANIC/ERROR errors. WARNING is notified when more than a certain number is detected per unit time.

- **Log analysis**
  - Preconfiguring Postgres parameters to output information to the database log is recommended
    - `log_min_duration_statement`: default "-1" (disabled) → "0" or value designed by user
    - `log_checkpoints`: default "off" → "on"
    - `log_connections`: default "off" → "on"
    - `log_disconnections`: default "off" → "on"
Deep Insight - Alerts

- Prometheus will send alerts for abnormal metrics according to the alert rules.

- Alert rules
  - Default rules

<table>
<thead>
<tr>
<th>Warning rules</th>
<th>Alert level</th>
<th>Duration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Container high CPU utilization</td>
<td>Warning</td>
<td>5 minutes</td>
<td>Server container/pod CPU utilization exceeds 80% of resource limit.</td>
</tr>
<tr>
<td>Server Container high RAM usage</td>
<td>Warning</td>
<td>30 minutes</td>
<td>Server container/pod memory usage exceeds 80% of resource limit.</td>
</tr>
<tr>
<td>PVC low disk space</td>
<td>Warning</td>
<td>5 minutes</td>
<td>PVC (volume) has less than 10% usable disks</td>
</tr>
<tr>
<td>Server Container not found</td>
<td>Warning</td>
<td>60 seconds</td>
<td>Server container/pod has been inaccessible for 60 second.</td>
</tr>
<tr>
<td>Postgresql down</td>
<td>Error</td>
<td>-</td>
<td>Server may be down or inaccessible.</td>
</tr>
<tr>
<td>Postgresql too many connections</td>
<td>Warning</td>
<td>-</td>
<td>Server container/pod connection usage exceeds 90% of available capacity.</td>
</tr>
</tbody>
</table>

- Alert rules are configurable
  - Alert levels, intervals, and thresholds can be set for any monitoring item

- Alert notification method
  - Set up notifications in Prometheus.
  - Alerts are sent via Alert Manager.
  - Integration available with email, SMS, Slack, and other systems.
When you create FUJITSU Enterprise Postgres resources, including clusters, you can see how custom resources are being created by integrating them with OpenShift standard event functionality.

Use case

- Visualize status of custom resource creation process
- Quickly identify abnormal processing and facilitate debugging

How to check events

CLI (`oc get events`)

GUI
Operator system configuration

Container that accepts user requests and automates database construction and operations.
Container for the FUJITSU Enterprise Postgres server.
Container created on the same pod as the server container to perform scheduled backup.
Container temporarily created to perform restore.
Container that uses Pgpool-II to provide load balancing and connection pooling. (optional)
Container that exposes http/https endpoint for monitoring stats scraping.
Storage for backup data. Not required if you do not need to obtain backups.

Parent CR for cluster definition and configuration.
Child CR for volumes.
Child CR for database configuration.
Child CR for system certificates.
Child CR for database users.
CR for monitoring configuration.
Service to connect to the master server.
Service to connect to the replica server.
Service for connecting to Pgpool-II.
Service to scrape metrics from all cluster nodes.
## Features and benefits

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Upgrade</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data centre</strong></td>
<td>Supported</td>
</tr>
<tr>
<td>Within one data center</td>
<td>Minor version, Major version</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>Upgrade Type</td>
</tr>
<tr>
<td>One master, two replica</td>
<td>Rolling update</td>
</tr>
<tr>
<td>One master</td>
<td></td>
</tr>
<tr>
<td><strong>Replication Type</strong></td>
<td></td>
</tr>
<tr>
<td>Asynchronous</td>
<td>Configuration Change</td>
</tr>
<tr>
<td>Synchronous</td>
<td></td>
</tr>
<tr>
<td>Logical replication</td>
<td></td>
</tr>
<tr>
<td><strong>Scaling</strong></td>
<td>Monitor &amp; Deep Insights</td>
</tr>
<tr>
<td>CPU</td>
<td>Monitoring</td>
</tr>
<tr>
<td>Memory</td>
<td>Operator metrics</td>
</tr>
<tr>
<td>Number of connections</td>
<td>Operands (i.e., FEPCluster) metrics</td>
</tr>
<tr>
<td><strong>Load Balancing</strong></td>
<td>Alert</td>
</tr>
<tr>
<td>Using Pgpool-II</td>
<td>Alert by metrics information</td>
</tr>
<tr>
<td><strong>High Availability</strong></td>
<td>Event Notification</td>
</tr>
<tr>
<td>Failover Type</td>
<td>CR creation events</td>
</tr>
<tr>
<td>Automatic</td>
<td></td>
</tr>
<tr>
<td>switchover type</td>
<td></td>
</tr>
<tr>
<td>Manual</td>
<td></td>
</tr>
<tr>
<td>Auto recovery</td>
<td></td>
</tr>
<tr>
<td>Automatic</td>
<td></td>
</tr>
<tr>
<td><strong>Backup and Restore</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Frequency</strong></td>
<td><strong>Scale-out read replica</strong></td>
</tr>
<tr>
<td>Configurable</td>
<td>Manual, Automatic</td>
</tr>
<tr>
<td><strong>Generation</strong></td>
<td><strong>Scale-in read replica</strong></td>
</tr>
<tr>
<td>Configurable</td>
<td>Manual</td>
</tr>
<tr>
<td><strong>Backup type</strong></td>
<td><strong>FUJITSU Enterprise Postgres features</strong></td>
</tr>
<tr>
<td>Full, incremental</td>
<td>Transparent Data Encryption</td>
</tr>
<tr>
<td><strong>Restore Type</strong></td>
<td>Data Masking</td>
</tr>
<tr>
<td>Latest, PITR</td>
<td>Dedicated Audit Log</td>
</tr>
<tr>
<td><strong>Restore to</strong></td>
<td>Vertical Clustered Index</td>
</tr>
<tr>
<td>New cluster, existing cluster</td>
<td>Global Meta Cache</td>
</tr>
</tbody>
</table>

**FUJITSU Enterprise Postgres features**

- Transparent Data Encryption
- Data Masking
- Dedicated Audit Log
- Vertical Clustered Index
- Global Meta Cache
In addition to OSS PostgreSQL features, FUJITSU Enterprise Postgres clusters support the features below:

<table>
<thead>
<tr>
<th>Category</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Global Meta Cache</td>
</tr>
<tr>
<td>Data security</td>
<td>Transparent Data Encryption</td>
</tr>
<tr>
<td></td>
<td>Data Masking</td>
</tr>
<tr>
<td></td>
<td>Dedicated Audit Log</td>
</tr>
<tr>
<td>High performance</td>
<td>Vertical Clustered Index</td>
</tr>
<tr>
<td></td>
<td>High-speed data load</td>
</tr>
<tr>
<td>Application interface</td>
<td>Java integration</td>
</tr>
<tr>
<td></td>
<td>ODBC integration</td>
</tr>
<tr>
<td></td>
<td>.NET framework integration</td>
</tr>
<tr>
<td></td>
<td>Embedded SQL integration (C language)</td>
</tr>
<tr>
<td></td>
<td>Embedded SQL integration (COBOL)</td>
</tr>
</tbody>
</table>
Supported platforms

The Operator is tested on the following platforms. (*: Kubernetes 1.21, 1.22 support)

<table>
<thead>
<tr>
<th>Service</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Managed Kubernetes Service</td>
<td>● Red Hat OpenShift Container Platform 4.6 / 4.7 / 4.8 / 4.9</td>
</tr>
<tr>
<td></td>
<td>● Rancher Kubernetes Engine (on Linux hosts) *</td>
</tr>
<tr>
<td>Fully Managed Kubernetes Service</td>
<td>● Red Hat OpenShift Service on AWS (ROSA)</td>
</tr>
<tr>
<td></td>
<td>● Azure Red Hat OpenShift (ARO)</td>
</tr>
<tr>
<td></td>
<td>● Red Hat OpenShift on IBM Cloud</td>
</tr>
<tr>
<td></td>
<td>● Azure Kubernetes Service (AKS) *</td>
</tr>
<tr>
<td></td>
<td>● Amazon Elastic Kubernetes Service (EKS) *</td>
</tr>
</tbody>
</table>

Supported storage:

<table>
<thead>
<tr>
<th>Category</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type/interface</td>
<td>● Container Storage Interface</td>
</tr>
<tr>
<td></td>
<td>● NFS</td>
</tr>
<tr>
<td></td>
<td>● Red Hat OpenShift Container Storage</td>
</tr>
<tr>
<td>Cloud service</td>
<td>● Azure Blob Storage</td>
</tr>
<tr>
<td></td>
<td>● Amazon S3</td>
</tr>
</tbody>
</table>

CPU: x86, s390x

Components embedded:

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat UBI minimal</td>
<td>8</td>
<td>Base OS image for the container</td>
</tr>
<tr>
<td>FUJITSU Enterprise Postgres Server</td>
<td>14.0</td>
<td>Database server capabilities</td>
</tr>
</tbody>
</table>
### Pre-requisite for Kubernetes

**Collaboration tools:** Integration with the monitoring and alerting tools below is supported:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>How to obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prometheus</td>
<td>Installed version of OpenShift</td>
<td>Pre-installed with OpenShift</td>
</tr>
<tr>
<td>AlertManager</td>
<td>Kubernetes v0.52.1 and later</td>
<td>prometheus-operator</td>
</tr>
<tr>
<td></td>
<td>Rancher</td>
<td>rancher-monitoring application</td>
</tr>
<tr>
<td>Grafana</td>
<td>OpenShift 3.10.3 and later</td>
<td>Provided by OperatorHub</td>
</tr>
<tr>
<td></td>
<td>Kubernetes 3.10.3 and later</td>
<td>grafana-operator</td>
</tr>
<tr>
<td></td>
<td>Rancher</td>
<td>rancher-monitoring application</td>
</tr>
</tbody>
</table>

Integration with FUJITSU Enterprise Postgres Operator management on Kubernetes tools below is supported:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>How to obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helm</td>
<td>3.7.2 and later</td>
<td>Helm Web Site</td>
</tr>
<tr>
<td>Rancher</td>
<td>v2.6.2 and later</td>
<td>Rancher Web Site</td>
</tr>
</tbody>
</table>
Assured quality and compatibility

- Red Hat OpenShift Operator Certification
  - Level V certified (as of October 2021)
  - Asia's first commercial middleware provider
  - World's first multi-architecture operator certification

Operator maturity model of Red Hat OpenShift

Level I
- Basic Install
  - Automated application
  - Provisioning and configuration management

Level II
- Seamless Upgrades
  - Patch and minor version upgrades supported

Level III
- Full Lifecycle
  - App lifecycle, storage lifecycle (backup, failure recovery)

Level IV
- Deep Insights
  - Metrics, alerts, log processing and workload analytics

Level V
- Auto Pilot
  - Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

FUJITSU Enterprise Postgres 14 Operator
- Create instance
- Upgrade
- Automatic backup
- Automatic failover
- Monitoring
- Alerting and events
- Auto scaling

Red hat OpenShift Container Platform

https://sdk.operatorframework.io/docs/overview/operator-capabilities/
FUJITSU Enterprise Postgres for Kubernetes

For more, visit our website at
fast.fujitsu.com/fujitsu-enterprise-postgres-for-kubernetes