Achieving carbon-neutral IT systems with Kubernetes, Operator, and Rust

White paper

Achieving **carbon-neutral IT systems** with Kubernetes, Operator, and Rust

To address climate change, countries around the world have set a goal of achieving carbon neutrality by 2050.

This presents a challenge for the IT industry, because with the dizzying progress of the information society, power consumption of data centers is expected to increase around 850 times by 2050.

As a result of these targets, the index of CO2 emission has begun to be adopted as an evaluation point of IT systems.

In this paper, we propose a way to significantly reduce the power consumption of software and IT systems running in data centers in order to achieve carbon neutrality.

We will look at how three primary technologies, Kubernetes, Operator, and Rust have the potential to achieve high-density resource aggregation with the objective of reducing compute demand and hence power consumption.

In our tests, the proposed approach reduced resource usage by approximately 1/500 compared to IT systems consisting of VM-based Python applications.

This white paper proposes the adoption of the Kubernetes, Operator, and Rust technologies as standards for software and IT systems in order to achieve carbon neutrality.

The current statistics from climate change studies point to a dramatic situation

Have you ever felt that the issue of climate change has become more commonly highlighted on TV and the Internet in recent years?

In fact, climate change disasters have increased worldwide to unprecedented levels[*1]. Hazards due to climate change include extreme weather events such as floods and droughts caused by large typhoons and heavy rain. One cause for the increase in extreme weather is the progress of global warming resulting from the increase in CO2 emissions.

In order to resolve this global challenge, the Paris Agreement was adopted in 2015, under which the world agreed on common goals such as achieving a balance between anthropogenic greenhouse gas emissions and removals by sinks in the second half of this century[*2]. More than 120 countries and regions have set a goal of becoming carbon neutral by 2050. In other words, the total shall become zero when the amount of greenhouse gas absorbed through afforestation and forest management is subtracted from the amount of emissions.

According to the United Nations report "The Closing Window"[*3], emissions of carbon dioxide equivalent (C0$_2$e) in 2020 were 54 gigatons. Of this, 14 gigatons, or about 26%, were generated from electricity and heat generation. These are set to be reduced through the use of renewable energy and nuclear power.

More specifically in the field of IT systems , the "Impact of the Development of an Information Society on Energy Consumption" report published by the Center for Low Carbon Society Strategy[*4] estimates that the power consumption of data centers will increase approximately 7 times by 2030, and more alarmingly, 850 times by 2050 when compared to 2018.

To address these issues, **Fujitsu**, **Microsoft**, and **Amazon Web Services** are improving the energy efficiency of their data centers by using AI and optimizing cooling[*5*6*7]. But while this reduction in the power consumption of the data centers themselves is a necessary step in the right direction, it is countered by the increase in the number of applications requiring server resources. In fact, the total floor space of data centers is estimated to increase 73% worldwide over the next four years [*8].

Therefore, in order to effectively reduce power consumption, it is necessary to also reduce the server resource consumption by the applications running on the data center and to stop unnecessary servers.

For this reason, the concept of evaluating CO2 emissions from software and IT systems has been attracting attention. For example, the **Green Software Foundation** has published a specification for Software Carbon Intensity (SCI)[*9], a method for scoring software systems based on carbon emissions, and has established new software evaluation criteria. The specification refers to efficient software execution and reduced resource consumption as one of the important ways to reduce carbon emissions.

In addition, Fujitsu, Microsoft, and Amazon Web Services have started providing CO2 emission visualization services[*10*11*12], proving that organizations are serious about their commitment to reducing their and customers' carbon footprint. In the future, on top of performance and reliability, additional indicators will be used to evaluate software and IT systems CO2 emissions, and organizations will be aware of these indicators when selecting software and services.

This paper proposes a method for realizing significant power saving in software and IT systems running in data centers, in order to reduce your organization's carbon footprint.

7x

Increase in energy consumption
by data centers by 2030*

850x

Increase in energy consumption
by data centers by 2050*

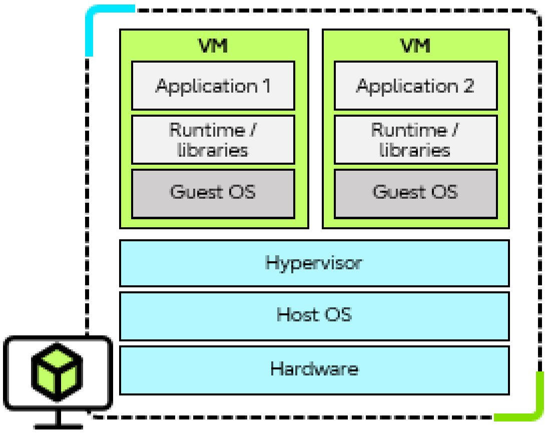
* Compared to 2018

Transforming IT systems toward the realization of energy efficiency

Resource aggregation techniques in current IT systems

Virtual machines (VMs) have been optimized as a way to streamline resources by aggregating multiple applications on a single server.

VMs can use hardware-level virtualization to create multiple operating system environments, such as Linux and Windows, on a single host machine via a *hypervisor*. As a result, a large number of applications can be aggregated on one server, and resource efficiency is improved as a whole system.



Each virtual machine requires a copy of the OS kernel

However, services that were initially small increase the number of VMs to manage as systems become more complex. Also, since each VM requires a guest OS, the extra hardware resources used by each VM need to be factored in.

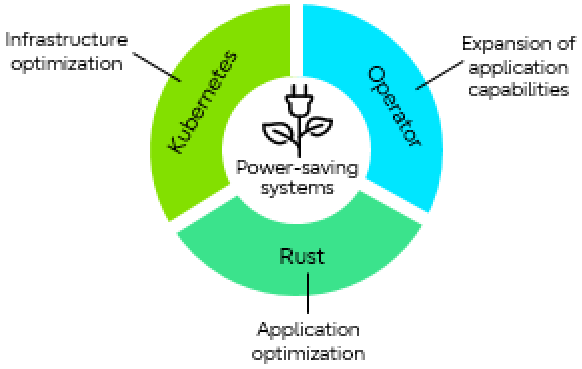
Application resource optimization

In order to achieve significant resource reduction in the system as a whole, it is necessary to optimize the resources used by the application itself, since the effect of aggregation alone is limited. That means aggregating more applications on one machine.

However, software optimization technology depends largely on the skills of programmers, and it will be difficult to optimize all applications in a general way. In this paper, we focus on programming languages for application development.

Power saving through synergy of primary technologies

By combining Kubernetes, Operator, and Rust, resource usage can be significantly reduced, helping direct the system toward carbon neutrality.



The first element of our proposal is to use a container orchestration engine for infrastructure optimization, and we suggest **Kubernetes** (k8s) since it is currently the most widely used technology in this category. This makes it possible to achieve greater resource aggregation compared to VMs.

However, applications that can be managed by Kubernetes are basically *stateless* applications, and the presence of *stateful* applications that hold persistent data is a bottleneck in resource aggregation on Kubernetes. That's where Kubernetes **Operator** comes in. The Operator technology allows managed applications to extend to stateful applications.

As a result, all applications can be managed on Kubernetes, but there is one challenge – while the autoscaling feature ensures that applications always have optimal resource allocation in response to load conditions, extra resources need to be reserved in case load conditions become high. Therefore, each organization (company or project) using the Kubernetes cluster generates surplus resources, which leads to a decrease in the aggregation rate. We discuss the solution later in this paper..

Finally, we propose using **Rust** to optimize resource usage by the applications themselves. Rust is a system programming language that aims to replace C/C++ offering the same levels of performance, reliability, and productivity[*13]. While Rust's memory and thread safety are often noted, this paper focuses on its resource efficiency, which is equivalent to C/C++.

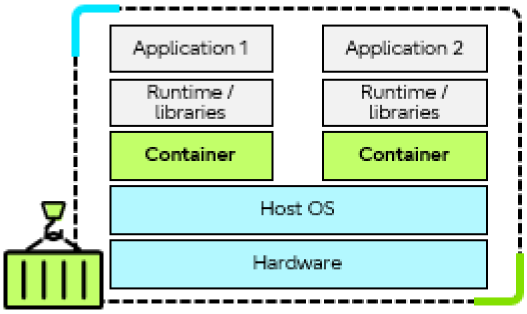
By combining **Kubernetes**, **Operator**, and **Rust**, it is possible to achieve the concentration of high-density application resources and ultimately obtain significant power savings for the entire IT system that will help the journey toward carbon neutrality.

Achieving infrastructure optimization using Kubernetes

Resource aggregation with Kubernetes

Kubernetes is a technology used to manage containers, which is known as *container orchestration*.

Container technology is highly portable and often used in microservice architectures such as those utilized in DX systems. This is because the necessary runtime/libraries are packaged together, and can be easily deployed on any infrastructure with container execution capabilities. With Kubernetes, you can automate operations such as container scaling, upgrades, and self-healing.



Containers save resources by sharing the kernel of the host OS

Container technology allows multiple applications to run in separate environments on a single machine or VM. Here, the containers share the kernel of their host OS and consume no extra resources compared to the VM. As a result, computer resources can be better aggregated than when using VMs.

Previous research results on resource aggregation using Kubernetes show that there was an effect of aggregating more than 86% of computer network resources compared to the use of VMs[*14].

Expanding coverage with Operator

Kubernetes was initially created to manage stateless applications only – the ability to support and run stateful applications was added later, and they represent a bottleneck during resource aggregation.

Stateful applications require additional operational knowledge to accommodate automation tasks such as scaling, upgrades, and self-healing. For example, consider a case where a database is created using a primary/standby configuration. In case the primary server fails, it is difficult to implement the ensuing operations (fencing the failed primary server, promoting the standby server to primary, and adding a new standby server) using only the standard functionalities of Kubernetes. Kubernetes is not able to pre-automate tasks that require such a level of application-specific operational knowledge.

The Operator technology allows software developers to implement the following 5 levels of functionality[*15] for stateful applications and manage them on Kubernetes, depending on the maturity of operational automation. Operators expand the scope of applications to be aggregated, and fit into one of the capability levels below.

- Level 1: Basic install
- Level 2: Seamless upgrades
- Level 3: Full lifecycle
- Level 4: Deep insight
- Level 5: Auto pilot

With level 5 capability, Operators deliver the autoscaling capability to optimize resource usage required to minimize energy consumption.

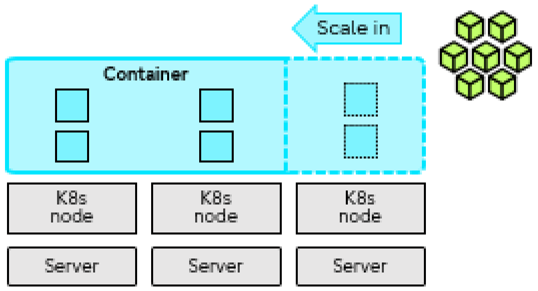
Operator maturity model of Red Hat OpenShift

Level 1	Level 2	Level 3	Level 4	Level 5
Basic install	**Seamless upgrades**	**Full lifecycle**	**Deep insights**	**Auto pilot**
Automated application provisioning and config management	Patch and minor version upgrades supported	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

Kubernetes resource optimization - Challenges and solutions

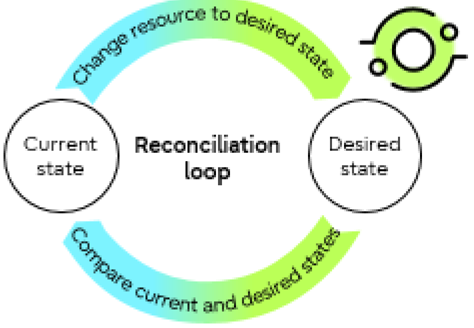
Fujitsu is constantly researching and evaluating how to optimize resources. An initial concept is to determine how operator technology can be used to automate the operation of applications on Kubernetes, whether stateless or stateful. As a result, the autoscaling capability enables applications on Kubernetes clusters to always maintain optimal resource allocation according to load conditions.

However, extra nodes need to be reserved in the cluster to handle high load conditions. This means that organizations (companies or projects) using Kubernetes clusters need to set aside extra servers, thus increasing the amount of resources set aside for the system.



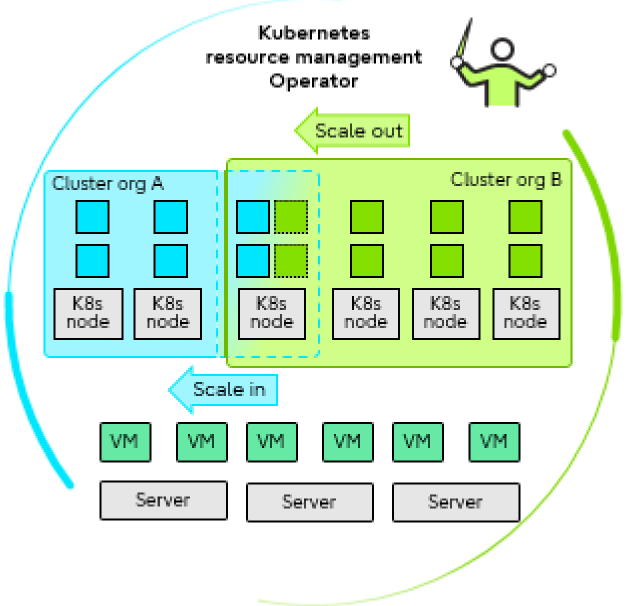
The illustration above depicts a potential **situation** in which nodes for a Kubernetes cluster are deployed on physical servers, and as the system autoscales the organization's application due to a workload decrease, containers are removed from a node, resulting in its physical server becoming redundant. Therefore, in situations where many organizations build Kubernetes clusters, this problem needs to be solved in order to achieve the resource aggregation efficiency mentioned above.

The Operator technology is utilized again as the solution. Operators use a mechanism called the **Reconciliation Loop**. While this technology is primarily used for resources running on Kubernetes, it is technically applicable to managing resources running using other technologies.


Change resource to desired state
Current state
Reconciliation loop
Desired state
Compare current and desired states

Current state is the resource allocation in place for each Kubernetes cluster, and the *desired state* is the optimal node allocation to each cluster. This makes it possible to automatically adjust node allocation to the clusters according to their resource usage.

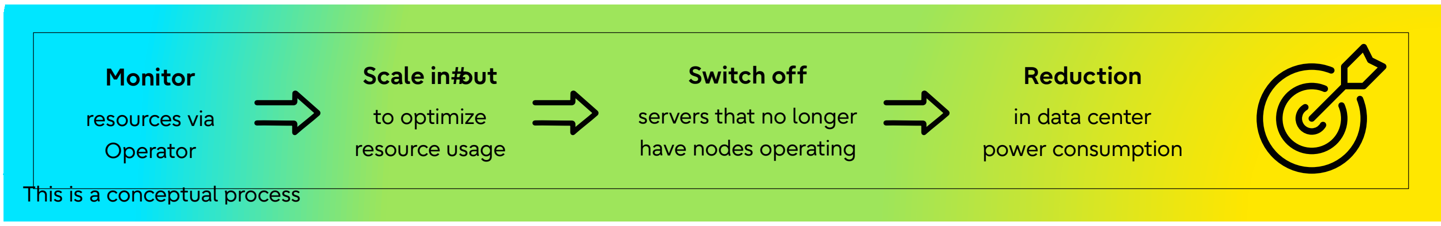
An example is illustrated below.



Multiple Kubernetes nodes can be deployed on one server by adding them to VMs divided into blocks. This increases the flexibility of Kubernetes node allocation. While VM aggregation requires extra resources in order to include the guest OS, aggregation of Kubernetes nodes increases both the isolation of clusters and the aggregation rate of application containers.

The **Kubernetes resource management Operator** monitors the resources being used by each organization cluster – applications on each cluster are automatically scaled out or in according to the load condition, reducing resource redundancy.

In the example above, the nodes that are no longer necessary for the organization A cluster are automatically reassigned to the organization B cluster. As a result, the entire system in the data center has fewer idle servers. Finally, if the Operator detects a server in which there are no nodes operating, it automatically turns its power off, thus reducing power consumption by the data center.



Monitor
resources via Operator

Scale in#out
to optimize resource usage

Switch off
servers that no longer have nodes operating

Reduction
in data center power consumption

This is a conceptual process

Fujitsu Enterprise Postgres for Kubernetes extends the capability of Kubernetes auto-scaling even further, allowing an additional metric – on top of the standard metric *CPU utilization* provided by Kubernetes, the *number of connections to the database is available to the Fujitsu Enterprise Postgres Operator*, thus increasing your control over cloud workloads.

By extending this resource optimization technology to all container platforms and other cloud platforms, we can help reduce global power consumption and help organizations in their commitment to carbon neutrality. Visit postgresql.fastware.com/fujitsu-enterprise-postgres-for-kubernetes for details.

Optimizing application resource utilization using Rust

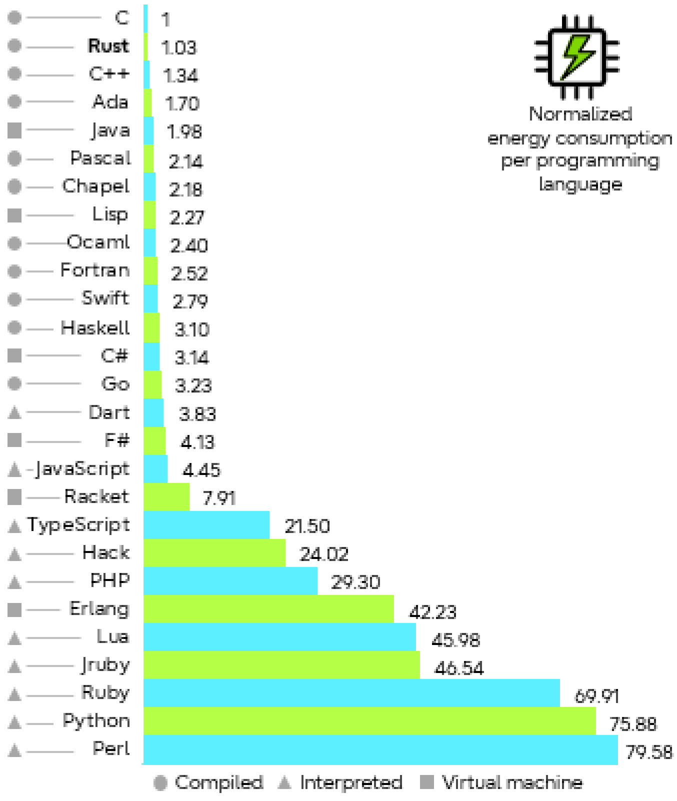
Relationship between Rust and resource efficiency

As mentioned above, resources can be more efficiently allocated by Kubernetes automatically adjusting cluster resources according to workload.

The other part of the equation is to reduce the amount of resources required by applications. Reducing the amount of resources used by each container allows Kubernetes to aggregate more containers.

Optimizing software so that it uses fewer resources relies on the skills of individual programmers, and thus is not a versatile approach. A more robust method is to focus on the programming language itself.

The graph below is from a research paper[*16] that investigated the energy efficiency of several programming languages.



In terms of energy efficiency, the top three are: C, Rust, and C++, respectively. C/C++ is known for its resource efficiency, which is one of the reasons it is used to develop embedded software for devices with limited battery capacity. However, one of its main problems is the difficulty of memory management.

Rust, the second most energy-efficient language, makes memory management much simpler. Another advantage of Rust's improved memory management is the increased security of applications, since many security flaws are due to memory management issues. In fact, Google[*17] and Microsoft[*18] have been shifting development of C/C++ applications to Rust.

Rust cites performance, reliability, and productivity as benefits, with a unique reliability mechanism that guarantees memory and thread safety. This prevents the creation of security vulnerabilities.

While Rust maintains high levels of reliability and productivity, the focus of this paper is on its superior performance and efficient memory management, which reduces resource usage of containers running on Kubernetes when compared to other languages.

Rust main advantages

- Superior performance and reliability
- Memory safety
- Zero-cost abstractions
- Easier concurrent programming
- Safe control over low-level resources
- Growing number of packages
- Increasing popularity
- Vibrant community

Container web application comparison results

In order to measure the energy efficiency of Rust in a container environment, we created a simple web application using Rust and Python, and made it into a container.

The same resources (CPU/memory) were allocated for the containers, and then resource usage and performance were compared. It used Fibonacci sequence calculations to reproduce the processing load of a web application, using the environments below for Rust and Python.

	Rust	Python
Version	1.61	3.10.4
Web framework	actix-web 4.0.1[*19]	fastapi 0.78.0[*20]

Below are the test results of continuously issuing Web APIs and measuring the number of API processes per second and CPU/memory usage on containers with 0.5 cores allocated for CPU resources and 64 MB allocated as memory.

	Rust	Python
Performance [API/s]	24.82 API/s	0.33 API/s
Average CPU utilization [%]	99.376%	100%
Average memory usage [MB]	4.7 MB	25.1 MB

In terms of performance and resource utilization, our tests showed that:

- Rust can handle around 75 times more Web API requests per unit of time than Python
- Rust uses less CPU and memory than Python when running similar web applications

Because of its higher performance, the number of containers deployed to run an application implemented in Rust can be reduced to around 1/75 compared to Python and still deliver the same performance. And because of its lower resource usage, Rust application containers can be allocated fewer resources.

It is important to note that the purpose of this paper is not to determine which language is better or worse for web applications. For example, Python has made some impressive achievements in machine learning, and the Rust compiler's build tool is Python. The rule of thumb when choosing is to adopt the right programming language for each purpose.

The argument put forth in this section is simply that Rust can be used to reduce resource allocation and resource usage in container applications that run on Kubernetes environments..

Rust at a glance

● Creation	2006
● Execution type	Compiled
● Ranking	Voted most loved language for the last 7 years[*]
● Performance	Extremely fast - no runtime or garbage collector
● Productivity	Friendly compiler, integrated package manager
● Reliability	Rich type system, memory ownership model
● Used in software	Firefox, Dropbox, Cloudflare, npm, Yelp
● Sponsors	Microsoft, AWS, Google Cloud, GitHub, Mozilla

*: https://survey.stackoverflow.co/2022/#technology-most-loved-dreaded-and-wanted

Summary and future issues

This paper proposes a combination of three technologies, Kubernetes, Operator and Rust, to realize a carbon neutral energy-saving IT system.

By leveraging Kubernetes and Operator technologies, many applications, including stateful applications, can be aggregated on Kubernetes. Furthermore, the clusters can automatically adjust their resource allocations, reducing overall system resources by up to 86.4% compared to VMs. In addition, the use of Rust reduced application resources by about 1/75 compared to Python. The combined system can reduce server resources by about 1/500 compared to IT systems consisting of VM-based Python applications.

The total server power consumption in data centers in Japan was estimated in 2018 to increase from 7 TWh to 46 TWh by 2030[*4].

If the proposed method reduces the total server power consumption and CO2 by up to 1/500 in 2030, the amount of reduction would be about 12 billion L in terms of crude oil. This is equivalent to 236 crude oil tanks (about 50,000 kiloliters) at the stockpiling base. This is the amount of reduction only applied to Japan domestic data centers, and this proposal is to expand the effect further by expanding the application of these technologies worldwide.

This will allow IT systems adopting our proposal to significantly reduce power consumption, and by extending data center resource optimization technology to all container and cloud platforms, reduce global power consumption and make strides toward carbon neutrality.

The challenge for the future is that since the proposed method is based on container technology, it is necessary to promote a container shift assuming cloud-native technologies for the IT systems' existing assets. While newly constructed systems and applications can be developed with containers as their targets, existing systems require time and effort to be containerized, for rebuilding applications and complicated network management due to silos created when adopting containers. As a result, the hurdle of container shift becomes higher.

It is possible to use container conversion tools for applications running on VMs and service mesh technology, but ultimately it is necessary to accumulate know-how on container shift in practice.

In order to make the most of the effectiveness of the proposed method in the future, it is necessary for individual customers, not only enterprises, to actively use IT systems running in data centers through cloud systems. In order to achieve this, it is necessary to create an era in which customers are conscious of their household carbon footprint, and it is necessary to develop CO2 emissions visualization services such as the one created by Fujitsu[*10].

This will empower humankind to engage in green activities by actively shifting their IT appliances to thin clients to save energy and leverage the energy efficient data centers proposed in this paper.

How Fujitsu Enterprise Postgres can help decarbonize compute

The planet has been warming at a steady pace and it's projected to keep doing so. While technology has played a part in this situation, it can become part of the solution.

Fujitsu takes sustainability seriously and wants to help organizations reduce their environmental footprint. No matter where your IT system is in the journey to carbon neutrality, we can help you. Read more in our brochure **Greener technology : How Fujitsu Enterprise Postgres can help decarbonize compute** available here.

Documents and reference materials cited in this white paper

*1: New York Times. "A Summer of Climate Disasters".
https://www.nytimes.com/2022/09/07/briefing/climate-change-heat-waves-us-europe.html

*2: United Nations Climate Change. "Paris Agreement".
https://unfccc.int/files/essential_background/convention/application/pdf/english_paris_agreement.pdf

*3: United Nations. "The Closing Window"
https://www.unep.org/resources/emissions-gap-report-2022

*4: Center for Low Carbon Society Strategy. "Impact of the progress of the information society on energy consumption (Vol. 2)".
https://www.jst.go.jp/lcs/pdf/fy2020-pp-03.pdf

*5: Fujitsu Limited. "Improve Power Usage Effectiveness (PUE) at Our Data Centers > Case Studies ".
https://www.fujitsu.com/global/about/environment/pue/case-studies/

*6: Microsoft. "Hydrogen fuel cells could provide emission free backup power at datacenters, Microsoft says".
https://news.microsoft.com/innovation-stories/hydrogen-fuel-cells-could-provide-emission-free-backup-power-at-datacenters-microsoft-says/

*7: Amazon Web Services. "The Carbon Reduction Opportunity of Moving to Amazon Web Services".
https://sustainability.aboutamazon.com/carbon-reduction-aws.pdf

*8: Research and Markets. "Global Data Centre Trends - 2022 to 2026".
https://www.researchandmarkets.com/reports/5647291/global-data-centre-trends-2022-to-2026

*9: Green Software Foundation. "Software Carbon Intensity (SCI) Specification".
https://github.com/Green-Software-Foundation/software_carbon_intensity

*10: Fujitsu. "Fujitsu and Ridgelinez Launch Services to Visualize Customers' Carbon Footprint in Supply Chain".
https://www.fujitsu.com/global/about/resources/news/press-releases/2021/1224-01.html

*11: Microsoft . "Calculating My Carbon Footprint - Microsoft Sustainability".
https://www.microsoft.com/en-us/sustainability/emissions-impact-dashboard

*12: Amazon Web Services. "Carbon Footprint Reporting.
https://aws.amazon.com/aws-cost-management/aws-customer-carbon-footprint-tool/

*13: Rust. "A language empowering everyone to build reliable and efficient software".
https://www.rust-lang.org/

*14: IPSJ Digital Library. "Design and operate large-scale container environments using multi-container orchestration".
https://ipsj.ixsq.nii.ac.jp/ej/?action=repository_action_common_download&item_id=212035&item_no=1&attribute_id=1&file_no=1

*15: Red Hat. "Operator maturity model". Red Hat Customer Portal.
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html/operators/understanding-operators

*16: Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, João Saraiva. "Energy Efficiency across Programming Languages".
https://greenlab.di.uminho.pt/wp-content/uploads/2017/10/sleFinal.pdf

*17: Thomas Claburn , "Google says Android runs better when covered in Rust", The Register.
hxxxxxttps://xtech.nikkei.com/atcl/nxt/column/18/00692/042700054/

*18: Aratrika Dutta, "Rust In, C and C++ Out of Microsoft! For the Sake of Security", Analytics Insight.
https://www.analyticsinsight.net/rust-in-c-and-c-out-of-microsoft-for-the-sake-of-security/

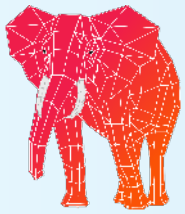
*19: The Actix Team. "ACTIX".
https://actix.rs/

*20: FastAPI. "FastAPI".
https://fastapi.tiangolo.com

Fujitsu Enterprise Postgres can help your journey

Fujitsu Enterprise Postgres is the enhanced version of PostgreSQL, for enterprises seeking a more robust, secure, and fully supported edition for business-critical applications.

It is fully compatible with PostgreSQL and shares the same operation method, interface for application development, and inherent functionality. Designed to deliver the Quality of Service that enterprises demand of their databases in the digital world, while supporting the openness and extensibility expected of open source platforms, all at a lower cost than traditional enterprise databases.



Fujitsu Enterprise Postgres

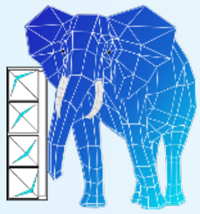
Combine the strengths of open-source PostgreSQL with the enterprise features developed by Fujitsu.

Enhanced speed, security, and support — without the costs associated with most proprietary systems.

Fujitsu Enterprise Postgres for Kubernetes

Utilize operator capabilities for provisioning and managing operations on the OpenShift Container Platform.

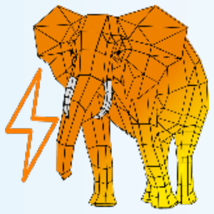
Business-ready database that integrates container operation technology for rapid development-to-production deployments.



Fujitsu Enterprise Postgres on IBM LinuxONE™

World-class platform that embraces open source and improves data security, performance, and business continuity.

The best of open source flexibility with the peace of mind that comes from knowing it is backed by Fujitsu and IBM.



Fujitsu Enterprise Postgres on IBM Power¶

Experience frictionless hybrid cloud that can help you modernize to respond faster to business demands.

Fujitsu database designed for security, performance, and reliability, combined with IBM server built for agility in the hybrid cloud.

About Fujitsu

Trust and performance for over 80 years Since it was founded back in 1935, Fujitsu has been innovating information and communication technologies (ICT) across the globe. A long line of landmark achievements and product milestones have made Fujitsu what it is today — a leading company in the ICT space.

Experience - 40 years of database development Fujitsu Enterprise Postgres is the outcome of Fujitsu's experience in developing enterprise databases for 40 years, its vision for and commitment to open source, its contributions to PostgreSQL, and its active participation in the PostgreSQL open source community.

Proudly supporting customers around the world As Japan's largest and the world's fifth largest IT services provider, Fujitsu proudly supports customers in more than 100 countries with over 126,000 Fujitsu employees. We utilize our experience and the power of ICT to shape the future of society for our customers.

Visit us at **fast fujitsu com**

Contact
Fujitsu Limited
Email: enterprisepostgresql@fujitsu.com
Website: fast.fujitsu.com
2025-03-26 WW EN