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.
**Abbreviations**

The following abbreviations are used in this manual:

<table>
<thead>
<tr>
<th>Full Name</th>
<th>Abbreviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUJITSU Software Enterprise Postgres for Kubernetes</td>
<td>FEP or</td>
</tr>
<tr>
<td>FUJITSU Software Enterprise Postgres</td>
<td>FUJITSU Enterprise Postgres</td>
</tr>
<tr>
<td>Vertical Clustered Index</td>
<td>VCI</td>
</tr>
<tr>
<td>Transparent Data Encryption</td>
<td>TDE</td>
</tr>
<tr>
<td>Point in time recovery</td>
<td>PITR</td>
</tr>
<tr>
<td>Custom Resource</td>
<td>CR</td>
</tr>
<tr>
<td>Custom Resource Definition</td>
<td>CRD</td>
</tr>
<tr>
<td>Persistent Volume</td>
<td>PV</td>
</tr>
<tr>
<td>Universal Base Image</td>
<td>UBI</td>
</tr>
<tr>
<td>OpenShift Container Platform</td>
<td>OCP</td>
</tr>
<tr>
<td>Mutual TLS</td>
<td>MTLS</td>
</tr>
</tbody>
</table>

**Abbreviations of manual titles**

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

<table>
<thead>
<tr>
<th>Full Manual Title</th>
<th>Abbreviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUJITSU Software Enterprise Postgres for Kubernetes Release Notes</td>
<td>Release Notes</td>
</tr>
<tr>
<td>FUJITSU Software Enterprise Postgres for Kubernetes Overview</td>
<td>Overview</td>
</tr>
<tr>
<td>FUJITSU Software Enterprise Postgres for Kubernetes Reference</td>
<td>Reference</td>
</tr>
</tbody>
</table>

**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 1.0: March 2022
## Contents

### Chapter 1 System Requirements

1.1 Components Embedded ................................................................. 1  
1.2 CPU .......................................................................................... 1  
1.3 Supported Platform .................................................................. 1  
1.4 Collaboration Tool .................................................................... 1  

### Chapter 2 Overview of Operator Design

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 ......................................... 9  
2.3.1 Deployment .......................................................................... 10  
2.3.2 High Availability ................................................................. 10  
2.3.3 Configurable Volume per Cluster ......................................... 11  
2.3.4 Deploying Pgpool-II and Connect to FEPCluster from Operator .................................................................................. 12  
2.3.5 Scheduling Backup from Operator ......................................... 12  
2.3.5.1 Important Setting Items ................................................... 12  
2.3.5.2 Parameters that cannot be Set .......................................... 13  
2.3.5.3 Restricted Parameters .................................................... 16  
2.3.5.4 About Sections in the Config File .................................... 16  
2.3.6 Perform PITR and Latest Backup Restore from Operator............... 16  
2.3.7 FEP Unique Feature Enabled by Default ................................. 16  
2.3.8 Monitoring & Alert (FEPExporter) ......................................... 17  
2.3.8.1 FEPExporter Custom Resource ...................................... 17  
2.3.8.2 Change to FEPCluster CR - metrics user ........................ 17  
2.3.8.3 FEPExporter CR auto-create for FEPCluster .................... 17  
2.3.9 Scaling Replicas .................................................................... 17  
2.3.9.1 Change to FEPCluster CR - auto scale out ....................... 18  
2.3.10 Disaster Recovery ................................................................. 18  

### Chapter 3 Operator Installation

3.1 Using the OperatorHub ................................................................. 20  
3.1.1 Pre-requisite ................................................................. 20  
3.1.2 Deploying Operator .............................................................. 21  
3.2 Using the Helm Chart ................................................................. 23  
3.2.1 Deploying Operator .............................................................. 23  
3.2.2 Upgrading Operators ......................................................... 23  
3.3 Using the Rancher UI ................................................................. 24  
3.3.1 Pre-requisite ................................................................. 24  
3.3.2 Register Helm Chart Repository ................................. 25  
3.3.3 Deploying Operator .............................................................. 27  
3.4 Implement Collaborative Monitoring Tools ............................. 28  
3.5 Implement Client ............................................................... 29  

### Chapter 4 Deployment Container

4.1 Deploying FEPCluster using Operator ....................................... 30  
4.2 Deploy a Highly Available FEPCluster ...................................... 34  
4.3 Deploying FEPExporter ............................................................ 36  
4.4 FEPExporter in Standalone Mode ............................................. 38  
4.5 Configuration FEP to Perform MTLS ........................................ 40  
4.5.1 Manual Certificate Management ....................................... 41  
4.5.2 Automatic Certificate Management ..................................... 44  
4.5.3 Deploy FEPCluster with MTLS support .............................. 49  
4.5.4 Configurable Parameters .................................................... 56
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.

<table>
<thead>
<tr>
<th>No</th>
<th>Component</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red Hat UBI minimal</td>
<td>8</td>
<td>Meant to provide base OS image for the container</td>
</tr>
<tr>
<td>2</td>
<td>FUJITSU Enterprise Postgres Server</td>
<td>14.0</td>
<td>To provide server capabilities</td>
</tr>
<tr>
<td>3</td>
<td>Patroni</td>
<td>2.1.2</td>
<td>To provide HA capabilities and other management to the Cluster</td>
</tr>
</tbody>
</table>

1.2 CPU

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

<table>
<thead>
<tr>
<th>No</th>
<th>CPU architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x86</td>
</tr>
<tr>
<td>2</td>
<td>s390x</td>
</tr>
</tbody>
</table>

1.3 Supported Platform

It supports running on the following platforms.

<table>
<thead>
<tr>
<th>No</th>
<th>Platform</th>
<th>Version</th>
<th>Support environments are as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OpenShift Container Platform</td>
<td>4.6, 4.7, 4.8, 4.9</td>
<td>- Azure Kubernetes Service</td>
</tr>
<tr>
<td>2</td>
<td>OpenShift Container Storage</td>
<td>4.6, 4.7, 4.8, 4.9</td>
<td>- Amazon EKS</td>
</tr>
<tr>
<td>3</td>
<td>Kubernetes</td>
<td>1.21, 1.22</td>
<td>- Rancher Kubernetes Engine (on Linux hosts)</td>
</tr>
</tbody>
</table>

Supports storage supported by OpenShift or Kubernetes (AKS, EKS and RKE).
However, backup and archive WAL volumes require shared storage, such as NFS.

1.4 Collaboration Tool

Supports integration with the following tools.

<table>
<thead>
<tr>
<th>No</th>
<th>Tool</th>
<th>Version</th>
<th>How to obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prometheus</td>
<td>- OpenShift</td>
<td>- OpenShift</td>
</tr>
<tr>
<td>2</td>
<td>AlertManager</td>
<td>The version installed OpenShift</td>
<td>Preinstalled with OpenShift</td>
</tr>
<tr>
<td>No</td>
<td>Tool</td>
<td>Version</td>
<td>How to obtain</td>
</tr>
<tr>
<td>----</td>
<td>----------------------</td>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Kubernetes v0.52.1 and later</td>
<td>- Kubernetes prometheus-operator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Rancher</td>
<td><a href="https://github.com/prometheus-operator/prometheus-operator">https://github.com/prometheus-operator/prometheus-operator</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The version provided by Rancher Monitoring Chart</td>
<td>- Rancher Using the Rancher Monitoring Chart</td>
</tr>
<tr>
<td>3</td>
<td>Grafana</td>
<td>- OpenShift and Kubernetes 3.10.3 and later</td>
<td>- OpenShift Provided by OperatorHub</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Rancher</td>
<td>- Kubernetes grafana-operator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The version provided by Rancher Monitoring Chart</td>
<td><a href="https://github.com/grafana-operator/grafana-operator">https://github.com/grafana-operator/grafana-operator</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Rancher Using the Rancher Monitoring Chart</td>
</tr>
<tr>
<td>4</td>
<td>Helm</td>
<td>3.7.2 and later</td>
<td>- Kubernetes only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Helm Web Site</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="https://helm.sh/docs/intro/install/">https://helm.sh/docs/intro/install/</a></td>
</tr>
<tr>
<td>5</td>
<td>Rancher</td>
<td>v2.6.2 and later</td>
<td>Rancher Web Site</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><a href="https://rancher.com/">https://rancher.com/</a></td>
</tr>
<tr>
<td>6</td>
<td>Prometheus Adapter</td>
<td>- OpenShift and Kubernetes</td>
<td>- OpenShift and Kubernetes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Confirmed the operation with v0.9.1.</td>
<td>Prometheus Adapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Rancher</td>
<td><a href="https://github.com/kubernetes-sigs/prometheus-adapter">https://github.com/kubernetes-sigs/prometheus-adapter</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The version provided by Rancher Monitoring Chart</td>
<td>- Rancher Using the Rancher Monitoring Chart</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Design required to operate FEP</th>
<th>Where to find</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEP setup</td>
<td>Required.</td>
<td>2.3.1 Deployment</td>
</tr>
<tr>
<td>High availability configuration</td>
<td>Optional. (When checking or changing the behavior of high availability. However, even by default, constant high availability operation is possible.)</td>
<td>2.3.2 High Availability</td>
</tr>
<tr>
<td>Volume settings</td>
<td>Optional. (When setting the volume. However, even by default, allocate a fixed volume.)</td>
<td>2.3.3 Configurable Volume per Cluster</td>
</tr>
<tr>
<td>Pgpool-II setup</td>
<td>Optional. (When using Pgpool-II.)</td>
<td>2.3.4 Deploying Pgpool-II and Connect to FEPCluster from Operator</td>
</tr>
<tr>
<td>Backup/restore settings</td>
<td>Optional. (When using a backup and restore.)</td>
<td>2.3.5 Scheduling Backup from Operator</td>
</tr>
<tr>
<td>Monitoring &amp; Alert(FEPExporter)</td>
<td>Optional. (When using Monitoring and Alert )</td>
<td>2.3.6 Perform PITR and Latest Backup Restore from Operator</td>
</tr>
<tr>
<td>Scaling Replicas</td>
<td>Optional. (When using scaling feature )</td>
<td>2.3.9 Scaling Replicas</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Configuration server type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEP operator</td>
<td>A container that accepts user requests and is responsible for automating database construction and operational operations.</td>
</tr>
<tr>
<td>FEP server container</td>
<td>A container for the FEP server.</td>
</tr>
<tr>
<td>FEP backup container</td>
<td>A container that performs scheduled backup operations. Created on the same Pod as the FEP server container.</td>
</tr>
<tr>
<td>FEP Fluentbit container</td>
<td>A container that collect FEP database CSV log and forward to fluentd container for processing.</td>
</tr>
<tr>
<td>FEP pgpool2 container</td>
<td>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.</td>
</tr>
<tr>
<td>FEP restore container</td>
<td>A container that performs the restore operation. Temporarily created during a restore operation.</td>
</tr>
<tr>
<td>FEP Exporter container</td>
<td>A container that exposes http/https endpoint for monitoring stats scraping.</td>
</tr>
<tr>
<td>FEP Fluentd container</td>
<td>A container that summarise FEP log severity as metrics for Prometheus to consume. Optionally, forward log entries to Elasticsearch for detailed log analysis.</td>
</tr>
</tbody>
</table>
### 2.2.2 User Account

The user accounts used by this product are as follows.

<table>
<thead>
<tr>
<th>User type</th>
<th>User name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure administrator</td>
<td>Mandatory</td>
<td>A system administrator (superuser) who has root privileges on all the servers that make up this product.</td>
</tr>
<tr>
<td>Database administrator</td>
<td>Mandatory</td>
<td>Install, set up, start, stop, and perform operation and maintenance of this product.</td>
</tr>
<tr>
<td>Application developer</td>
<td>Mandatory</td>
<td>Develops and executes database applications.</td>
</tr>
</tbody>
</table>

### 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.


For each *Version*, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>ubi8</td>
<td></td>
</tr>
<tr>
<td>FEPBaseVersion</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>MajorVersion</td>
<td>1, 2, ...</td>
<td>To be used when major change in image, including server patch application</td>
</tr>
<tr>
<td>MinorVersion</td>
<td>0, 1, 2 ...</td>
<td>To be used when minor changes in image, e.g bug fix in container script</td>
</tr>
</tbody>
</table>

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

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

```text
```

For each *Version*, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OS</code></td>
<td>ubi8</td>
<td></td>
</tr>
<tr>
<td><code>FEPBaseVersion</code></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td><code>MajorVersion</code></td>
<td>1,2, ...</td>
<td>To be used when major change in image, including server patch application</td>
</tr>
<tr>
<td><code>MinorVersion</code></td>
<td>0,1,2 ...</td>
<td>To be used when minor changes in image, e.g. bug fix in container script</td>
</tr>
</tbody>
</table>

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

- `fujitsu-enterprise-postgres-14-backup:ubi8-14-1.0`
  - `fujitsu-enterprise-postgres-14-backup:ubi8-14-1.0-amd64`
  - `fujitsu-enterprise-postgres-14-backup:ubi8-14-1.0-s390x`

FEP restore container

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

```text
```

For each *Version*, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OS</code></td>
<td>ubi8</td>
<td></td>
</tr>
<tr>
<td><code>FEPBaseVersion</code></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td><code>MajorVersion</code></td>
<td>1,2, ...</td>
<td>To be used when major change in image, including server patch application</td>
</tr>
<tr>
<td><code>MinorVersion</code></td>
<td>0,1,2 ...</td>
<td>To be used when minor changes in image, e.g. bug fix in container script</td>
</tr>
</tbody>
</table>

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

- `fujitsu-enterprise-postgres-14-restore:ubi8-14-1.0`
  - `fujitsu-enterprise-postgres-14-restore:ubi8-14-1.0-amd64`
  - `fujitsu-enterprise-postgres-14-restore:ubi8-14-1.0-s390x`
FEP pgpool2 container

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


For each Version, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>ubi8</td>
<td></td>
</tr>
<tr>
<td>FEPBaseVersion</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>MajorVersion</td>
<td>1,2, ...</td>
<td>To be used when major change in image, including server patch application</td>
</tr>
<tr>
<td>MinorVersion</td>
<td>0,1,2 ...</td>
<td>To be used when minor changes in image, e.g. bug fix in container script</td>
</tr>
</tbody>
</table>

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

- fujitsu-enterprise-postgres-14-pgpool2:ubi8-14-1.0
- fujitsu-enterprise-postgres-14-pgpool2:ubi8-14-1.0-amd64
- fujitsu-enterprise-postgres-14-pgpool2:ubi8-14-1.0-s390x

FEP Exporter container

FEP Exporter container as for FEP server containers.


For each Version, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>ubi8</td>
<td></td>
</tr>
<tr>
<td>FEPBaseVersion</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>MajorVersion</td>
<td>1,2, ...</td>
<td>To be used when major change in image, including server patch application</td>
</tr>
<tr>
<td>MinorVersion</td>
<td>0,1,2 ...</td>
<td>To be used when minor changes in image, e.g. bug fix in container script</td>
</tr>
</tbody>
</table>

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

- fujitsu-enterprise-postgres-exporter:ubi8-14-1.0
- fujitsu-enterprise-postgres-exporter:ubi8-14-1.0-amd64
- fujitsu-enterprise-postgres-exporter:ubi8-14-1.0-s390x

FEP Fluentd container

FEP Fluentd container as for FEP server containers.


For each Version, specify the following:
<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS</strong></td>
<td>ubi8</td>
<td></td>
</tr>
<tr>
<td><strong>FEPBaseVersion</strong></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td><strong>MajorVersion</strong></td>
<td>1,2, ...</td>
<td>To be used when major change in image, including server patch application</td>
</tr>
<tr>
<td><strong>MinorVersion</strong></td>
<td>0,1,2 ...</td>
<td>To be used when minor changes in image, e.g. bug fix in container script</td>
</tr>
</tbody>
</table>

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

- fujitsu-enterprise-postgres-fluentd:ubi8-14-1.0
  - fujitsu-enterprise-postgres-fluentd:ubi8-14-1.0-amd64
  - fujitsu-enterprise-postgres-fluentd:ubi8-14-1.0-s390x

**FEP Fluentbit container**

FEP Fluentbit container as for FEP server containers.


For each Version, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS</strong></td>
<td>ubi8</td>
<td></td>
</tr>
<tr>
<td><strong>FEPBaseVersion</strong></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td><strong>MajorVersion</strong></td>
<td>1,2, ...</td>
<td>To be used when major change in image, including server patch application</td>
</tr>
<tr>
<td><strong>MinorVersion</strong></td>
<td>0,1,2 ...</td>
<td>To be used when minor changes in image, e.g. bug fix in container script</td>
</tr>
</tbody>
</table>

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

- fujitsu-enterprise-postgres-fluentbit:ubi8-14-1.0
  - fujitsu-enterprise-postgres-fluentbit:ubi8-14-1.0-amd64
  - fujitsu-enterprise-postgres-fluentbit:ubi8-14-1.0-s390x

**FEP Cronjob container**

FEP Cronjob container as for FEP server containers.


For each Version, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS</strong></td>
<td>ubi8</td>
<td></td>
</tr>
<tr>
<td><strong>FEPBaseVersion</strong></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td><strong>MajorVersion</strong></td>
<td>1,2, ...</td>
<td>To be used when major change in image, including server patch application</td>
</tr>
</tbody>
</table>
The first publishing will expect following names / tagging (Manifest and Child images)
- fujitsu-enterprise-postgres-cronjob:ubi8-14-1.0
  - fujitsu-enterprise-postgres-cronjob:ubi8-14-1.0-amd64
  - fujitsu-enterprise-postgres-cronjob:ubi8-14-1.0-s390x

FEP Upgrade container

FEP Upgrade container as for FEP server containers.

For each Version, specify the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>ubi8</td>
<td></td>
</tr>
<tr>
<td>FEPBaseVersion</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>MajorVersion</td>
<td>1,2, ...</td>
<td>To be used when major change in image, including server patch application</td>
</tr>
<tr>
<td>MinorVersion</td>
<td>0,1,2 ...</td>
<td>To be used when minor changes in image, e.g bug fix in container script</td>
</tr>
</tbody>
</table>

The first publishing will expect following names / tagging (Manifest and Child images)
- fujitsu-enterprise-postgres-14-upgrade:ubi8-14-1.0
  - fujitsu-enterprise-postgres-14-upgrade:ubi8-14-1.0-amd64
  - fujitsu-enterprise-postgres-14-upgrade:ubi8-14-1.0-s390x

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:

<table>
<thead>
<tr>
<th>Spec</th>
<th>Example</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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

Any content after ‘#’ will be ignored

The value may be quoted by single quotes

The value may be quoted by single quotes

Any content after ‘#’ will be ignored

Single quote can be escaped by two single quotes

Backslash ‘\’ will be replaced by ‘\’ when invoking patronictl edit-config command

When a key value pair is invalid, it will be ignored. The update continues to process next pair

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.

```yaml
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

Volumes for the cluster nodes(pods) are initially created in accordance with the values set in `fepChildCrVal` storage section of the parent FEPCluster CR.

The parent FEPCluster CR creates a child FEPVolume CR with the respective startup values and the relevant controller(FEPVolume Controller) takes care of creating the required volumes.

Any subsequent change to the volumes need to go thru the FEPVolume CR thus created.

Below is the schema of the FEPVolume CR:

<table>
<thead>
<tr>
<th>Field</th>
<th>Mandatory</th>
<th>Sub-Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>archivewalVol</td>
<td>No</td>
<td>size</td>
<td>1Gi</td>
<td>Size of the volume, expandable later</td>
</tr>
<tr>
<td></td>
<td></td>
<td>storageClass</td>
<td>Defaults to platform default if omitted</td>
<td>SC is only set at start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>accessModes</td>
<td>Defaults to ReadWriteOnce if omitted</td>
<td>Access mode is only set at start</td>
</tr>
<tr>
<td>backupVol</td>
<td>No</td>
<td>size</td>
<td>2Gi</td>
<td>Size of the volume, expandable later</td>
</tr>
<tr>
<td></td>
<td></td>
<td>storageClass</td>
<td>Defaults to platform default if omitted</td>
<td>SC is only set at start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>accessModes</td>
<td>Defaults to ReadWriteOnce if omitted</td>
<td>Access mode is only set at start</td>
</tr>
<tr>
<td>dataVol</td>
<td>Yes</td>
<td>size</td>
<td>2Gi</td>
<td>Size of the volume, expandable later</td>
</tr>
<tr>
<td></td>
<td></td>
<td>storageClass</td>
<td>Defaults to platform default if omitted</td>
<td>SC is only set at start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>accessModes</td>
<td>Defaults to ReadWriteOnce if omitted</td>
<td>Access mode is only set at start</td>
</tr>
<tr>
<td>logVol</td>
<td>No</td>
<td>size</td>
<td>1Gi</td>
<td>Size of the volume, expandable later</td>
</tr>
<tr>
<td></td>
<td></td>
<td>storageClass</td>
<td>Defaults to platform default if omitted</td>
<td>SC is only set at start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>accessModes</td>
<td>Defaults to ReadWriteOnce if omitted</td>
<td>Access mode is only set at start</td>
</tr>
<tr>
<td>tablespaceVol</td>
<td>No</td>
<td>size</td>
<td>512Mi</td>
<td>Size of the volume, expandable later</td>
</tr>
<tr>
<td></td>
<td></td>
<td>storageClass</td>
<td>Defaults to platform default if omitted</td>
<td>SC is only set at start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>accessModes</td>
<td>Defaults to ReadWriteOnce if omitted</td>
<td>Access mode is only set at start</td>
</tr>
<tr>
<td>walVol</td>
<td>Yes</td>
<td>size</td>
<td>1200Mi</td>
<td>Size of the volume, expandable later</td>
</tr>
<tr>
<td></td>
<td></td>
<td>storageClass</td>
<td>Defaults to platform default if omitted</td>
<td>SC is only set at start</td>
</tr>
</tbody>
</table>
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.

### 2.3.4 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.

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

```yaml
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Overview of parameters</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy Archive Option (--archive-copy)</td>
<td>Copy the WAL segments needed for consistency to the backup</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Check Archive Mode Option (--archive-mode-check)</td>
<td>Check the PostgreSQL archive_mode setting.</td>
<td>Limited to backup from master</td>
</tr>
<tr>
<td>Backup from Standby Option (--backup-standby)</td>
<td>Back up from the standby cluster</td>
<td>Limited to backup from master</td>
</tr>
<tr>
<td>Stop Auto Option (--stop-auto)</td>
<td>Stops a previously failed backup on a new backup.</td>
<td>Because they are 9.6 not supported in</td>
</tr>
<tr>
<td>SSH client command Option (--cmd-ssh)</td>
<td>Path to ssh client executable</td>
<td>Not using ssh</td>
</tr>
<tr>
<td>Compress Option (--compress)</td>
<td>Use File Compression</td>
<td>For obsolete options (Use compresse-type option instead)</td>
</tr>
<tr>
<td>Config Option (--config)</td>
<td>pgBackRest configuration file.</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Config Include Path Option (--config-include-path)</td>
<td>Path to additional pgBackRest configuration files.</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Config Path Option (--config-path)</td>
<td>Base path of pgBackRest configuration files.</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Delta Option (--delta)</td>
<td>Restore or Backup with Checksum</td>
<td>For new restores only</td>
</tr>
<tr>
<td>Dry Run Option (--dry-run)</td>
<td>Execute a dry-run for the command.</td>
<td>command-line only option</td>
</tr>
<tr>
<td>Lock Path Option (--lock-path)</td>
<td>Path where the lock file is stored</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Keep Alive Option (--sck-keep-alive)</td>
<td>Enable keep-alive messages on socket connections</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Spool Path Option (--spool-path)</td>
<td>Path to store temporary data for asynchronous archive-push and archive-get commands</td>
<td>For automatic determination from FEPCluster CR values</td>
</tr>
<tr>
<td>Stanza Option (--stanza)</td>
<td>Defines the stanza.</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Console Log Level Option (--log-level-console)</td>
<td>Console Log Level</td>
<td>It is not expected to operate on Pod.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Overview of parameters</td>
<td>Reason</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Std Error Log Level Option (--log-level-stderr)</td>
<td>Stderr log level</td>
<td>It is not expected to operate on Pod.</td>
</tr>
<tr>
<td>Log Path Option (--log-path)</td>
<td>Log File Destination</td>
<td>For automatic determination from FEPCluster CR values</td>
</tr>
<tr>
<td>Azure Repository Account Option (--repo-azure-account)</td>
<td>Azure account used to store the repository</td>
<td>Azure storage is not supported</td>
</tr>
<tr>
<td>Azure Repository Container Option (--repo-azure-container)</td>
<td>Azure repository container. Azure container used to store the repository.</td>
<td></td>
</tr>
<tr>
<td>Azure Repository Endpoint Option (--repo-azure-endpoint)</td>
<td>Azure repository endpoint.</td>
<td></td>
</tr>
<tr>
<td>Azure Repository Key Option (--repo-azure-key)</td>
<td>Azure Repository Shared Key or Shared Access Signature</td>
<td></td>
</tr>
<tr>
<td>Azure Repository Key Type Option (--repo-azure-key-type)</td>
<td>Azure Repository Key Type</td>
<td></td>
</tr>
<tr>
<td>GCS Repository Bucket Option (--repo-gcs-bucket)</td>
<td>GCS repository bucket.</td>
<td>GCS is not supported</td>
</tr>
<tr>
<td>GCS Repository Endpoint Option (--repo-gcs-endpoint)</td>
<td>GCS repository endpoint.</td>
<td></td>
</tr>
<tr>
<td>GCS Repository Key Option (--repo-gcs-key)</td>
<td>GCS repository key.</td>
<td></td>
</tr>
<tr>
<td>GCS Repository Key Type Option (--repo-gcs-key-type)</td>
<td>GCS repository key type.</td>
<td></td>
</tr>
<tr>
<td>Repository Host Option (--repo-host)</td>
<td>Repository host for remote operations via SSH</td>
<td>Repository Host is not used</td>
</tr>
<tr>
<td>Repository Host Command Option (--repo-host-cmd)</td>
<td>Path of pgBackRest on Repository Host</td>
<td></td>
</tr>
<tr>
<td>Repository Host Configuration Option (--repo-host-config)</td>
<td>Repository Host Configuration File Path</td>
<td></td>
</tr>
<tr>
<td>Repository Host Configuration Include Path Option (--repo-host-config-include-path)</td>
<td>Repository hosts configuring include path</td>
<td></td>
</tr>
<tr>
<td>Repository Host Configuration Path Option (--repo-host-config-path)</td>
<td>Repository Host Configuration Path</td>
<td></td>
</tr>
<tr>
<td>Repository Host Port Option (--repo-host-port)</td>
<td>Repository host port when &quot;repo-host&quot; is configured</td>
<td></td>
</tr>
<tr>
<td>Repository Host User Option (--repo-host-user)</td>
<td>Repository host user when &quot;repo-host&quot; is configured</td>
<td></td>
</tr>
<tr>
<td>Repository Path Option (--repo-path)</td>
<td>Path where backups and archives are stored</td>
<td>For automatic determination from FEPCluster CR values</td>
</tr>
<tr>
<td>Archive Retention Option (--repo-retention-archive)</td>
<td>The number of consecutive WAL backups to keep.</td>
<td>This option is not recommended, and WAL retention is controlled by the Full Retention Option and Full Retention Type Option.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Overview of parameters</td>
<td>Reason</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Archive Retention Type Option (--repo-retention-archive-type)</td>
<td>Backup Type for WAL Retention</td>
<td>It is recommended not to change from the default.</td>
</tr>
<tr>
<td>Differential Retention Option (--repo-retention-diff)</td>
<td>Number of incremental backups to keep</td>
<td>No incremental backups</td>
</tr>
<tr>
<td>Archive Mode Option (--archive-mode)</td>
<td>Retains or disables the archive for the restored cluster.</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Exclude Database Option (--db-exclude)</td>
<td>Restore excluding the specified databases.</td>
<td>To restore the entire FEP cluster, including all databases</td>
</tr>
<tr>
<td>Include Database Option (--db-include)</td>
<td>Restore only the specified database</td>
<td>To restore the entire FEP cluster, including all databases</td>
</tr>
<tr>
<td>Link All Option (--link-all)</td>
<td>Restore all symbolic links.</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Link Map Option (--link-map)</td>
<td>Changes the destination of a symbolic link.</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Recovery Option Option (--recovery-option)</td>
<td>Setting options in postgresql recovery.conf</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>Tablespace Map Option (--tablespace-map)</td>
<td>Restoring tablespace to a specified directory</td>
<td>For automatic determination from FEPCluster CR values</td>
</tr>
<tr>
<td>Map All Tablespaces Option (--tablespace-map-all)</td>
<td>Restores all tablespaces to the specified directory</td>
<td>No tables required because there is only one table space per FEPCluster</td>
</tr>
<tr>
<td>PostgreSQL Database Option (--pg-database)</td>
<td>PostgreSQL database.</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>PostgreSQL Host Option (--pg-host)</td>
<td>PostgreSQL host for remote operations via SSH</td>
<td>No SSH connection required</td>
</tr>
<tr>
<td>PostgreSQL Host Command Option (--pg-host-cmd)</td>
<td>Path of pgBackRest exe on the PostgreSQL host</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>PostgreSQL Host Configuration Option (--pg-host-config)</td>
<td>Path of the pgBackRest configuration file</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>PostgreSQL Host Configuration Include Path Option (--pg-host-config-include-path)</td>
<td>Setting pgBackRest on PostgreSQL host include path</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>PostgreSQL Host Configuration Path Option (--pg-host-config-path)</td>
<td>Path to configure pgBackRest on the PostgreSQL host</td>
<td>To use internal fixed values</td>
</tr>
<tr>
<td>PostgreSQL Host Port Option (--pg-host-port)</td>
<td>SSH Port Specification</td>
<td>No SSH connection required</td>
</tr>
<tr>
<td>PostgreSQL Host User Option (--pg-host-user)</td>
<td>The logon user when hosting PostgreSQL, if pg-host is set.</td>
<td>No SSH connection required</td>
</tr>
<tr>
<td>PostgreSQL Path Option (--pg-path)</td>
<td>PostgreSQL data directory.</td>
<td>For automatic determination from FEPCluster CR values</td>
</tr>
<tr>
<td>PostgreSQL Port Option (--pg-port)</td>
<td>PostgreSQL Ports</td>
<td>For automatic determination from FEPCluster CR values</td>
</tr>
<tr>
<td>PostgreSQL Socket Path Option (--pg-socket-path)</td>
<td>PostgreSQL Unix socket path</td>
<td>For automatic determination from FEPCluster CR values</td>
</tr>
<tr>
<td>PostgreSQL Database User Option (--pg-user)</td>
<td>PostgreSQL database user</td>
<td>To use internal fixed values</td>
</tr>
</tbody>
</table>
2.3.5.3 Restricted Parameters

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Overview of parameters</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository Type Option (--repo-type)</td>
<td>Type of storage to use for the repository</td>
<td>posix/s3</td>
</tr>
</tbody>
</table>

2.3.5.4 About Sections in the Config File

In FEPBackup 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 FEPRestore 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)

Data masking

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

```bash
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. For details on how to specify the passphrase, refer to "FEPCluster parameter" in the Reference.
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 OSS (pg_backrest) functionality to store backup data in object storage, data can be migrated to a database cluster in a different OCP environment.
Even if it is difficult to operate in an OCP environment with a database cluster due to a disaster, it is possible to continue operating in a different OCP environment.
Chapter 3 Operator Installation

This chapter describes how to install FEP operator.

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.
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 14” to find FUJITSU Enterprise Postgres 14 Operator.

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

Wait still installation is complete and status changes to “Succeeded”.
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.

   ```
   kubectl 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. Updating Custom Resource Definitions
   Update custom resource definitions before upgrading Helm Chart.

   ```
   helm show crds fep-repo/fujitsu-enterprise-postgres-operator > crds.yaml
   kubectl apply -f crds.yaml
   ```
4. 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].

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>fepnamespace</td>
<td>Any text you want that better descri</td>
<td>FEPProject</td>
</tr>
</tbody>
</table>

### 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.
Click [Create] to create the Helm Chart repository.

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
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].

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.

3.4 Implement Collaborative Monitoring Tools

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/4.9/monitoring/monitoring-overview.html#understanding-the-monitoring-stack_monitoring-overview)

For Grafana, install and use the Grafana Operator provided by OperatorHub for x86. Grafana is not exposed by OperatorHub in s390x, so use Helm to build Grafana. Detailed instructions are available at the following site for your reference.


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.

### 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.

4.1 Deploying FEPCluster using Operator

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

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.
(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.

5. When "Create" is clicked on either of the two pages above, the operator creates FEPCluster CR, and thereafter one by one FEPBackup, FEPConfig, FEPVolume, FEPUser, and FEPCert 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.

7. Once child CRs are marked done in annotations, operator creates StatefulSet for the cluster.
8. StatefulSet will start one FEP instance at one time and will wait it to be ready before starting next one.

9. 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.

3. It is the same as the procedure from step 5 to step 10 in "4.1 Deploying FEPCluster using Operator".
4. Three pods deployed and ready for a highly available FEPCluster.

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
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
<th>FEPROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>fep-ansible-operator-88f7fb4b-5jh85</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>24m</td>
<td></td>
</tr>
<tr>
<td>new-fep-sts-0</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>17m</td>
<td>master</td>
</tr>
<tr>
<td>new-fep-sts-1</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>15m</td>
<td>replica</td>
</tr>
<tr>
<td>new-fep-sts-2</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>13m</td>
<td>replica</td>
</tr>
</tbody>
</table>

### 4.3 Deploying FEPExporter

To deploy a FEPExporter, follow these steps:

**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. In order to deploy FEPExporter managed by Operator, it is as easy as setting `fep.monitoring.enable` to `true` in FEPCluster CR at the time of deployment.

2. FEPExporter will be created automatically under the name `<cluster-name>-fepexporter`. And it will list show all the database with statistics of specified FEPcluster.

3. FEPExporter spawned by FEP Operator in aforementioned way will scrape metrics by default from the Master and standby instances and make it available to Prometheus.

4. User can configure MTLS to be used for HTTP endpoint used by Prometheus for metrics scraping as well as connection from FEPExporter to database.
   a. If `pgMetricsUser`, `pgMetricsPassword` and `pgMetricsUserTls` is defined in FEPcluster, FEPExporter will hence use these for securing connection to the postgres instances. In absence of these parameters, FEPExporter will use `pgAdminUser` (i.e. super user).
   b. User can configure Prometheus.tls and FEPExporter.tls to ensure that metrics end point (`/metrics`) by FEPExporter is also used with MTLS (Refer to “FEPExporter Custom Resource” in the Reference for details of fields)

5. User can also configure basic authentication by specifying a secret that contains username & password. (Refer to “FEPExporter Custom Resource” in the Reference for details of fields)
6. 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 used 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/4.6/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.

1. To create FEPExporter CR, either
   (1) Click on “Create Instance” under FEPExporter.
   OR
   (2) 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.

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

6. Targeting the name of FEPcluster in FEPExporter cluster list. Before targeting cluster, Check the FEPCluster status and FEP StatefulSet are in running condition.

7. It will recreate FEPExporter pod with a new datasource secret. It will list down all the database with statistics of specified FEPcluster in monitoring section.

8. If fepClusterList has more than one clusters listed, current exporter will collect metrics for all of those listed.

9. Multiple FEPExporters can be deployed within one namespace with their own cluster list to collect metrics from.

4.5 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)
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.

### 4.5.1 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

```bash
openssl genrsa -aes256 -out myca.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
```
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

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

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
6. Create FEP Server certificate signed by CA

```bash
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
```

6. **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.

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

```bash
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.

```bash
openssl genrsa -out patroni.key 2048
Generating RSA private key, 2048 bit long modulus
...............................................+++...............................................+++...

43
```

9. Create certificate signing request for Patroni

```bash
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
```
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

```bash
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: 0km9ijn8uhb7ygv
```

11. Create TLS secret to store Patroni certificate and key

```bash
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.

```bash
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

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

14. Create client certificate for "postgres" user

```bash
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: 0km9ijn8uhb7ygv
```

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

```bash
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"

4.5.2 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 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
EOF
```
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-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
EOF
```
Create postgres user client certificate

```bash
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

```bash
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 `feplogging-dev`.

```bash
cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: fluentd-cert
  namespace: feplogging-dev
spec:
  subject:
    commonName: "nfl-fluentd-headless-service"
    dnsNames:
      - 'nfl-fluentd-headless-service'
      - 'nfl-fluentd-headless-service.feplogging-dev'
      - 'nfl-fluentd-headless-service.feplogging-dev.svc'
EOF
```
Create FEPLogging client(prometheus) certificate

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

Create FEPLogging client(fluentbit) certificate

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

Create FEPEXporter 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: fepexporter-cert
  namespace: my-namespace
spec:
  subject:
    commonName: "exp1-service"
EOF
```
Create FEPExporter user client(prometheus) certificate

```bash
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
```

4.5.3 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**.

```yaml
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/fep-server-test:ubi8-mtls-enhance_15'
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,vci,pgaudit,pg_prewarm'
  session_preload_libraries='vci,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

# vci parameters in default fep installation
vci.enable = on
vci.maintenance_work_mem = 256MB
vci.max_local_ros = 64MB
vci.force_max_parallelism = off

# 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: rewinder
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-----
MIIEowIBAAKCAQAEADFkImha8CIJ1VcwXbBP1L+/DmS9/ipRhQQHxfO5x7j90nse
IHdFde+Qx2GX8kAiA7VykF6kfacwBYTATUlxDgwWtRZKVRp+hk2DIj2wPcjr14m
nTFi6a2zvAPugDezHc9F8/dchYj3cw81X0kU6xamqrKQY1xQ468Nk10q0wh06aK
AHF4ewfCr8ot44AdIAJcU26S1RKSZ+RzUrz+3Py+j907Enj1Yr332KUkW30pU
9dpIneyXBN/p76eC3j3MetYwqmpV/pHq8Y8bpxqSyoyRhqQDSts14dt1eca2e4j
u7OotcPKl2EJL96e8BqalLycg91pbAMQ15w0gQIDAQABAcIABAcQ213pucimXrrQ
fQxANMjgNYK4JjgXC860UwF01u4ubKw5V532hLSPPWLS+a0lA1w1bNozSbBV0u8G
64Wwr99b3/ctJqZ26/U2UTbHPU+Ogh24qhwF5QU8kXZEU11o3yPofslqijX9G
FF0fLeLCV8nL3JK9RiaDXxXbEYPWrYu3JbMFCpAxAzV2PrNxsF9PKyNWHnBPc08z5

- 51 -
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.

```yaml
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/fep-server-test:ubi8-mtls-enhance_13'
    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,vci,pgaudit,pg_prewarm"
      session_preload_libraries="vci,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

# vci parameters in default fep installation
vci.enable = on
vci.maintenance_work_mem = 256MB
vci.max_local_ros = 64MB
vci.force_max_parallelism = off

# 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-----
MIIEowIBAAKCAQEA0DFkImha8CIJiVcwXbBp1L/+dS9/ivepR8QHx0fso7X7s0Nae
IHDpD+Qx2G8KAhA1vYyk6fWacbfYBATU1xdOWmTw823KVRhP+kZ1Dj1wPcJe14mr
mTP6I1a6aUqHdezf9F/3chYj3cw81X0u6axamqFkQY1QxH848KN1Qow0s6kAk
AHF4eWfcR80t4x4DA1AJCuzC5RSKZEcURz5o+py+j907Enj3rY3KUHw30p9u
9dSepney9BxP76cX3MnetWtgmpv/p+h48bpQg0fYRhyRgsQsSRC14dteicaSe24j
uTOotcFkZELH86u8gaLCyG91pbAMQj5w8r8IQDAQAbc1BAcQg213poucm1Exgrq
f4kAoxNdqMYK4j5jQXBC60u10fQ85x5323LSPnLwLs+cOAI1AwN0sBCvOu86G
64Jw9rw9vc/cQ/vZQ0/f2uBnPfo+Gq8q4wF5QUBXZEU11Oz3ysFoaflagYj9G
F5F0gfLVC8nL39KcIADxKxBgEyPnYru393FkPcpAaZv2P3rXpp59yKynWm18Fc8o2z5
TfJ4/fbn/3fJ1AVyVwxWqQp0Lx57hcQ7yW/2kJRyQmdk1K790OLwKdWvDQg
qraore2J5wnxG05rs/1/LdKtv1778gwk96jgQkybk5v0+DKozj9r1L0Gg0KXO1D
znS2JRAEcqYiEAOqf/fxtPdsnGiall2/zwewxar.dj/WeoEBFECb6/y470Aau8xN8
16PcvfQPc7ouJT7cn2JY5m5b5WEOBdEIAeo711vm/QFtM+suaPdSt/CtCHrY
92C0nxQm7yF34UXElkMaya8E1pV/7dm5u0al+m4eE9XMYST1wqBkyEcVEAlh4X
rcf1J7a5aCCL7k2U4dT2D2NE5BSGK9i0+0xL6t7Si2Kzu6mSszh+3m0K4v1Hf
/gkdxvQ5mSwuUlfhoyEg3yQ8S6av3GstRsp003j7BE8C20Vp0FntN72mdz2f/
ZYz5cKuVlYh9gqeyYh7rceON9A8bkJcKDGmHpyyLaEcgyBALD0TPgDrYm1Dmd1qH
FFo4etk/T7B7YI1ktQ8j1KqtQhIzPz4q4W7yUhyjQ5a4QXQyS1iy9PfYeP/rEc3JFrd
+o2s3Hhyumekgmpu2sjiyptrmFebN74g3t4JdaQHJoco+GdcEyvNy9Hin9pN51p7E
fuEm14d3gH0Ehs7hQVAAKBQgDRU3xmLXc9oKrNBiyenderGLL1qLqBQsYt181T
2UfizGWLw+PFC1AmpkxdR0VnqQpilriR1E1bpApa0aog2eaya/LJscdj/5j52uc8
W3JOn1jpkqasaTf57B8p57m6THYO15F50evoi/a8s13nRuiquk+m+TsUHY5FJDhe
aQdK4QK6bCohIfteyrV+MNwwazk6D1Cnb886fv23W4RAT8BYYX3w9YfnvJ4Hv8/B
m0WgyK1Uqphra8nMUC790Fyo9ePArQ3x57LtJ191bmKtkQex63fyyQyxH35XrL
JsktmA42EcGmosi2isjB7COKH32r3w0J5CgaInQxj1dbGrwh5Qpn
-----END RSA PRIVATE KEY-----
crt: |
-----BEGIN CERTIFICATE-----
MIID2CAsCgwH4BaG1QdfyTe4dZkJf1YJrJTA8bKgkhi9iG908QAqFADBX
MgRwTqVDQ8KWe5Ne8Bqgpmhbl6YXrp24xcZnABgVbAgTAk6MN34w4LXYDvQoD
EyNs5E8Pcdmbl6YXrp24gQVd1gmaWnHdugQXVo0q03AG5T54dXMDQy
MDADMQ0lVAvD0t1D1xUM0gQ5AmdQ1gOvWgDEBMBQGA1UEawmXK15ajChtkR0Jn
ZdCA9Is9eWJYoZEsQhhgAQEBQBBQAgEPFADCAQQcggEBANAxJC3vAWV1T1XM12
TS/vw5xvf4QqYEBB8Xzuce04e0j7p3H183RexvMdhI/Cgi1GvcpI+phH2nMwEwV1
MQ4hfSxQvM51UT4fpgqY19sD3a9eJpzk+iCmtprni1A4Xs3P3F3X1WJ93MPNv9
JFOs5mpqygry3CUB+PDZCKNNMIUrCgBhxelnwqQ/DreOMQAASXNFgkJtUsKmRLVe
Wft59Dve/douJ46dx9251BN9VPxAsJ3s1nWf6u+fu9x9hrWMLY1QqVf6EmKp
W8hnaEMKYEAT7KqPe8e8Zg8Mnxm17zqwXLD59Crxz+rnx1lGI7cnBv2AwdEDjeC
K/EClAwEaAsoO3jC3C5z7tBZMVNSHDEAeK38ygrq6B8Qc0DATMmbV8HRRMBAF8EAAJ
MGI1Bq8pNRHXqQwqCwVqyzF3a92dlbK5JjaGctkRjLbVt525jbiVzodGy
LmSxV2s9ghqMl5bG1aV8Zxc3m3CfJg7qRM6GzG1cXMz3JfK
MnZoNy1wCGOObu8YkX11oZ2bKqBcvYzcy5hd9vYh2nLXBO5y7zmOZsOUnhXk
Yi21n0knz4vdcZmolY2nLXB0Yyo5rdmOZsOLUM1kX
-----END CERTIFICATE-----
cacct: |
4.5.4 Configurable Parameters

To enable MTLS, make changes to the following parameters.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec.fep.usePodName</td>
<td>True</td>
<td>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.</td>
</tr>
<tr>
<td>spec.fep.patroni.tls.certificateName</td>
<td>&lt;secret-name&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>spec.fep.patroni.tls.caName</td>
<td>&lt;configmap-name&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>spec.fep.postgres.tls.certificateName</td>
<td>&lt;secret-name&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>Key</td>
<td>Value</td>
<td>Details</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>spec.fep.postgres.tls.caName</td>
<td>&lt;configmap-name&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>spec.fep.postgres.tls.privateKeyPassword</td>
<td>&lt;secret-name&gt;</td>
<td>Name of Kubernetes secret that contains the password for the private key for Postgres Server.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.sysUsers.pgAdminTls.certificateName</td>
<td>&lt;secret-name&gt;</td>
<td>Name of Kubernetes secret that contains the certificate in tls.crt and private key in tls.key for &quot;postgres&quot; 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.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.sysUsers.pgAdminTls.caName</td>
<td>&lt;configmap-name&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.sysUsers.pgAdminTls.sslMode</td>
<td>verify-full</td>
<td>For MTLS, this value must be set to verify-full. If only TLS is required, this can be set to verify-ca or prefer.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.sysUsers.pgrepluserTls.certificateName</td>
<td>&lt;secret-name&gt;</td>
<td>Name of Kubernetes secret that contains the certificate in tls.crt and private key in tls.key for &quot;repluser&quot; 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.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.sysUsers.pgrepluserTls.caName</td>
<td>&lt;configmap-name&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.sysUsers.pgrepluserTls.sslMode</td>
<td>verify-full</td>
<td>For MTLS, this value must be set to verify-full. If only TLS is required, this can be set to verify-ca or prefer.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.sysUsers.pgRewindUserTls.certificateName</td>
<td>&lt;secret-name&gt;</td>
<td>Name of Kubernetes secret that contains the certificate in tls.crt and private key in tls.key for &quot;rewinduser&quot; 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.</td>
</tr>
<tr>
<td>Key</td>
<td>Value</td>
<td>Details</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>spec.fepChildCrVal.sysUsers.pgRewindUserTls.caName</td>
<td>&lt;configmap-name&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.sysUsers.pgRewindUserTls.sslMode</td>
<td>verify-full</td>
<td>For MTLS, this value must be set to verify-full. If only TLS is required, this can be set to verify-ca or prefer.</td>
</tr>
</tbody>
</table>

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.

### 4.6 Replication Slots

#### 4.6.1 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.

   ```yaml
spec:
  fep:
    forceSel: true
    replicationSlots: |
      myslot1:
        type: logical
        database: db1
        plugin: pgoutput
      myslot2:
        type: logical
        database: db1
        plugin: pgoutput
    podAntiAffinity: false
   ```

b. Add section postgres under spec.fep as shown below.
   ```yaml
   postgres:
   ```
   caName = enter the name of configmap created for the CA
   certificateName = secret created by the end user that contains server certificate

   ```yaml
   caName: cacert
   certificateName: my-fep-cert
   ```

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

   ```bash
   archive_mode = on
   ```

   ```bash
   archive_command = 'pgbackrest --stanza=backupstanza
   --config/database/userdata/pgbackrest.conf archive-push %'
   ```

   ```bash
   wal_level = logical
   ```

   ```bash
   max_wal_senders = 12
   ```

   ```bash
   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.

   ```bash
   customPgHba: |
   ```
   ```bash
   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

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

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.

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

![Image showing ls -lrt /tmp/custom_certs/]

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:

```
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

4.7 FEP Logging

FEPCluster generates log files over the lifetime of execution. These log files can be useful for understanding cluster healthiness 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 to a analytics platform such as Elasticsearch.

There are two steps to enable monitoring and forwarding.

1. FEPLogging 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 monitoring the FEP server log file on persistent volume and forward to the FEP Logging instance.

Multiple FEPClusters can forward logs to single FEPLogging instance.

User can have two types of connection between FEPCluster & FEPLogging
- 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 "4.5.2 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.

### 4.7.1 FEPLogging Configuration

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

#### 4.7.1.1 FEPLogging Custom Resources - spec

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

Following is a sample template:

```yaml
spec:
  fepLogging:
    elastic:
      authSecret:
        secretName: elastic-auth
        passwordKey: password
        userKey: username
      host: elastic-passthrough.apps.openshift.com
      logstashPrefix: postgres
      port: 443
      scheme: https
      readTimeout: 60s
      sslVerify: true
      tls:
        certificateName: elastic-cert
        caName: elastic-cacert
    image:
      pullPolicy: IfNotPresent
      pullSpec:
        limits:
          cpu: 500m
          memory: 700Mi
        requests:
          cpu: 200m
          memory: 512Mi
        restartRequired: false
        sysExtraLogging: false
        scrapeInterval: 30s
        scrapeTimeout: 30s
        tls:
          certificateName: fluentd-cert
          caName: cacert
```

- 62 -
Below is the list of all parameters defined in the fepLogging section, along with their brief description:

<table>
<thead>
<tr>
<th>Custom Resource spec</th>
<th>Required/Optional</th>
<th>Change Effect</th>
<th>Updating value allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec.fepLogging.image.image</td>
<td>Optional</td>
<td>Fluentd Image of FEPLogging</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.image.pullPolicy</td>
<td>Required</td>
<td>Fluentd Image pull policy of FEPLogging</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.mcSpec.limits.cpu</td>
<td>Required</td>
<td>Max CPU allocated to fluentd container</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.mcSpec.limits.memory</td>
<td>Required</td>
<td>Max memory allocated to fluentd container</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.mcSpec.requests.cpu</td>
<td>Required</td>
<td>CPU allocation at start for fluentd container</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.mcSpec.requests.memory</td>
<td>Required</td>
<td>Memory allocation at start for fluentd container</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.sysExtraLogging</td>
<td>Required</td>
<td>To turn on extra debugging messages for operator, set value to true. It can be turned on/off at any time</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.restartRequired</td>
<td>Required</td>
<td>To restart FEPLogging instance for applying any new configuration for example after certificate rotation</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.scrapeInterval</td>
<td>Optional</td>
<td>Scrape interval for Prometheus to fetch metrics from FEPLogging instance</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.scrapeTimeout</td>
<td>Optional</td>
<td>Scrape Timeout for Prometheus to fetch metrics from FEPLogging instance</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.host</td>
<td>Optional</td>
<td>Target Elasticsearch host name</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.port</td>
<td>Optional</td>
<td>Target Elasticsearch port number</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.authSecret.othersName</td>
<td>Optional</td>
<td>Secret name which contains Elasticsearch authentication username &amp; password</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.authSecret.userKey</td>
<td>Optional</td>
<td>Username key specified in Elasticsearch authentication secret</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.authSecret.passwordKey</td>
<td>Optional</td>
<td>Password key specified in Elasticsearch authentication secret</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.logstashPrefix</td>
<td>Optional</td>
<td>Logstash prefix to differentiate index pattern in Elasticsearch. Default value is postgres</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.scheme</td>
<td>Optional</td>
<td>Connection scheme between FEPLogging &amp; Elasticsearch. Possible options http &amp; https</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.sslVerify</td>
<td>Optional</td>
<td>Set to true if you want to verify ssl certificate. If set to false then will not consider TLS certificate</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.tls.certificateName</td>
<td>Optional</td>
<td>Kubernetes secret name which holds fluentd certificate</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.elastic.tls.caName</td>
<td>Optional</td>
<td>Kubernetes configmap which holds ca.cert of Elasticsearch to verify Elasticsearch TLS connection</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.fepLogging.tls.certificateName</td>
<td>Optional</td>
<td>Kubernetes secret name which holds Fluentd certificate</td>
<td>Yes</td>
</tr>
<tr>
<td>Custom Resource spec</td>
<td>Required/Optional</td>
<td>Change Effect</td>
<td>Updating value allowed</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td>---------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>spec.fepLogging.tls.caName</td>
<td>Optional</td>
<td>Kubernetes configmap which holds cacert of Fluentd to configure MTLS between FEPLogging &amp; Prometheus</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.prometheus.tls.certificateName</td>
<td>Optional</td>
<td>Kubernetes secret name which holds Prometheus certificate</td>
<td>Yes</td>
</tr>
<tr>
<td>spec.prometheus.tls.caName</td>
<td>Optional</td>
<td>Kubernetes configmap which holds cacert of Fluentd to configure MTLS between FEPLogging &amp; Prometheus</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 4.7.1.1 Define fepLogging image

The image property is used to specify other than default Fluentd image and it’s pullPolicy from FEPLogging CR. If not specified it will use default image provided by Operator.

**Example**

```yaml
spec:
  fepLogging:
    image:
      image: 'quay.io/fujitsu/fujitsu-enterprise-postgres-14-fluentbit:ubi814-0.0'
      pullPolicy: IfNotPresent
```

### 4.7.1.2 Define fepLogging mcSpec

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

**Example**

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

### 4.7.1.3 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**

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

### 4.7.1.4 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
```

### 4.7.1.1.5 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
- scheme
- sslVerify
- tls (if sslVerify set to true)

Configure Elasticsearch server and use its 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
      port: 443
      scheme: https
      sslVerify: false
      tls:
        certificateName: fluentd-cert
        caName: elastic-cacert
```

### 4.7.1.1.6 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
```
4.7.1.7 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)

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

4.7.1.8 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)

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

4.7.2 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.

4.7.2.1 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:

```yaml
spec:
  fep
    ...
  remoteLogging:
    enable: true
    fluentdName: new-fep-logging
    tls:
      certificateName: fluentbit-cert
```
Below is the list of all parameters defined in the remoteLogging section, along with their brief description:

<table>
<thead>
<tr>
<th>Custom Resource spec</th>
<th>Required/Optional</th>
<th>Change Effect</th>
<th>Updating value allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>remoteLogging.enable</td>
<td>Required</td>
<td>The ‘enable’ is set to true for enabling Logging feature</td>
<td>No</td>
</tr>
<tr>
<td>remoteLogging.fluentdName</td>
<td>Required</td>
<td>The ‘fluentdName’ is the name of the FEPLogging CR where logs will be forwarded</td>
<td>Yes</td>
</tr>
<tr>
<td>remoteLogging.tls.secretName</td>
<td>Optional</td>
<td>Secret name which contains MTLS certs of fluentbit</td>
<td>No</td>
</tr>
<tr>
<td>remoteLogging.tls.caName</td>
<td>Optional</td>
<td>Cacert of Fluentd for ssl verification</td>
<td>No</td>
</tr>
</tbody>
</table>

### 4.7.2.1.1 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)

```yaml
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)

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

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

### 4.7.2.1.2 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
cName: ca-cert`

**Note**
The Elasticsearch server is configured by user and it is NOT part of FEPLogging deployment by operator.

### 4.7.3 FEPLogging Operations

#### 4.7.3.1 Log Forwarding to Elasticsearch

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

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

#### 4.7.3.2 Log severity based Alarms/Metrics

FEPLogging feature is used for raising alarm/alert based on postgres severity counts as well. While user creates FEPLogging CR, Operator will forward real time counts of various postgres severity 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 FEPLogging implementation as listed below:

- FEPLogErrorMessage
- FEPLogFatalMessage
- FEPLogPanicMessage
- 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.

4.7.4 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.

4.8 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.

4.8.1 FEP Custom Resources - spec.fep.pgBadger

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

### 4.8.2 Define pgBdager 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 pgBdager report will be created and uploaded.

Example)

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

### 4.8.3 Define pgBdager 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)

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

### 4.8.4 Define Endpoint for Uploading Report

**Web server url**

Both http and https are supported.

Example)

```yaml
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
```

- 70 -
Create a secret by using the wrapped text. The key must be 'basic_auth'.

Example)

```yaml
kind: Secret
apiVersion: v1
metadata:
  name: pgbadger-endpoint-auth
  namespace: fep-container-ct
data:
  basic_auth: YW1GemIyNTNPbXBoYzI5dWR3PT0=
type: Opaque
```

Add the secret name in the endpoint definition.

Example)

```yaml
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)

```bash
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)

```bash
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)

```yaml
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)

```yaml
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)

```yaml
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)

```yaml
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 endpoint section will be converted to curl command parameters. The following table shows the mapping:

<table>
<thead>
<tr>
<th>curl command parameter</th>
<th>User configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>[URL]</td>
<td>Endpoint url</td>
</tr>
<tr>
<td>--cert</td>
<td>webserver.pem</td>
</tr>
<tr>
<td></td>
<td>included in the secret referred in customCertificateName</td>
</tr>
<tr>
<td>--key</td>
<td>webserver.key</td>
</tr>
<tr>
<td></td>
<td>included in the secret referred in customCertificateName</td>
</tr>
<tr>
<td>--cacert</td>
<td>webca.pem</td>
</tr>
<tr>
<td></td>
<td>included in the configmap referred in customCertificateName</td>
</tr>
<tr>
<td>--form “uploadfile=@@path/to/report”</td>
<td>Endpoint fileUploadParameter</td>
</tr>
<tr>
<td>--header &quot;Authorization: Basic passxxxx”</td>
<td>Endpoint authentication configmap</td>
</tr>
<tr>
<td>--insecure</td>
<td>When endpoint.insecure is set to true</td>
</tr>
</tbody>
</table>

### 4.8.5 Uploaded File on Web Server

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

<table>
<thead>
<tr>
<th>incremental mode</th>
<th>Uploaded file name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>[fep cluster name]-sts-[pod index].zip</td>
<td>pgbadger-test3-sts-0.zip</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgbadger-test3-sts-1.zip</td>
</tr>
<tr>
<td>False</td>
<td>[fep cluster name]-sts-[pod index].html</td>
<td>pgbadger-test3-sts-0.html</td>
</tr>
</tbody>
</table>
The zip file contains a folder of pgbadger incremental report.

Example)

```sql
\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.
Chapter 5 Post-Deployment Operations

This chapter describes the operation after deploying the container.

5.1 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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>.metadata.name</td>
<td>Name of Cluster</td>
</tr>
<tr>
<td>AGE</td>
<td>Elapsed time</td>
<td>Indicates the amount of time that has elapsed since the cluster was created</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Custom Resource spec</th>
<th>Change effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>.spec.fep.instances: n</td>
<td>Increase the number of nodes in the cluster to n.</td>
</tr>
</tbody>
</table>

This will impact behaviour for values in fep section only.

All other values except spec.fepChildCrVal.backup.image.image must be changed from respective child CR.

Delete FEPCluster

Equivalent Kubernetes command: kubectl delete FEPCluster <cluster_name>

This operation will remove the FEPCluster by the cluster_name and all Child CRs (FEPVolume, FEPConfig, FEPCert & FEPUser) & 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.
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 the "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 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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>.metadata.name</td>
<td>Name of pgpool2</td>
</tr>
</tbody>
</table>

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>`

Specify updated parameters in the format described in "2.3.4 Deploying Pgpool-II and Connect to FEPCluster from Operator". Only following parameters would change for Operations that can be performed here.

<table>
<thead>
<tr>
<th>Custom Resource spec</th>
<th>Change Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>.spec.count: n</td>
<td>Increase the number of nodes in the cluster to n.</td>
</tr>
<tr>
<td>.spec.serviceport</td>
<td>Change the TCP port for connecting to the Pgpool-II.</td>
</tr>
<tr>
<td>.spec.statusport</td>
<td>Change the TCP port for connecting to the PCP process.</td>
</tr>
<tr>
<td>.spec.limits.cpu</td>
<td>Change limits of cpus.</td>
</tr>
<tr>
<td>.spec.limits.memory</td>
<td>Change limits of memory.</td>
</tr>
<tr>
<td>.spec.requests.cpu</td>
<td>Change requests of cpus.</td>
</tr>
<tr>
<td>.spec.requests.memory</td>
<td>Change requests of memory.</td>
</tr>
<tr>
<td>.spec.fepclustername</td>
<td>Change fepcluster to connect.</td>
</tr>
<tr>
<td>.spec.customhba</td>
<td>Change pool_hba.conf file.</td>
</tr>
<tr>
<td>.spec.custompcp</td>
<td>Change pcp.conf file.</td>
</tr>
<tr>
<td>.spec.customsslkey</td>
<td>Change key content</td>
</tr>
<tr>
<td>.spec.customsslcert</td>
<td>Change the contents of the public x 509 certificate.</td>
</tr>
<tr>
<td>.spec.customsslcacert</td>
<td>Change the contents of the CA root certificate in PEM format.</td>
</tr>
</tbody>
</table>

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.

5.3 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: fepbackup
  status: ok
cipher: none
db (current)
  wal archive min/max (12-1): 000000010000000000000001/000000010000000000000005
  full backup: 20201125-025043F
    wal start/stop: 0000000100000000000000003 / 00000001000000000000000000000003
    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: 000000010000000000000005 / 00000001000000000000000000000005
    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>

Specify updated parameters in the format described in "2.3.5 Scheduling Backup from Operator". Only following parameters would change for Operations that can be performed here.

<table>
<thead>
<tr>
<th>Custom Resource spec</th>
<th>Change Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec.schedule.num</td>
<td>Change the Number of Registered Backup Schedules</td>
</tr>
<tr>
<td>spec.scheduleN.schedule</td>
<td>Change the scheduled backup time</td>
</tr>
<tr>
<td>spec.scheduleN.type</td>
<td>Change the scheduled backup type</td>
</tr>
<tr>
<td>spec.pgBackrestParams</td>
<td>Change pgBackRest parameters</td>
</tr>
<tr>
<td>spec.scheduleN.repo</td>
<td>If you specified more than one repository for spec.pgBackrestParams, select the repository in which to store the backup data. The default is 1.</td>
</tr>
</tbody>
</table>

- 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)
5.4 Configure MTLS Setting

5.4.1 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.

5.5 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.
5.5.1 Monitoring FEP Operator and Operands

The monitoring of FEP operators and operands are achieved by Prometheus' standard alive and resource monitoring.

<table>
<thead>
<tr>
<th>Metrics name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alive monitoring</td>
<td>Can monitor Pod status</td>
</tr>
<tr>
<td>Resource monitoring</td>
<td>You can monitor the following resource status</td>
</tr>
<tr>
<td></td>
<td>- CPU Usage</td>
</tr>
<tr>
<td></td>
<td>- CPU Quota</td>
</tr>
<tr>
<td></td>
<td>- Memory Usage</td>
</tr>
<tr>
<td></td>
<td>- Memory Quota</td>
</tr>
<tr>
<td></td>
<td>- Current Network Usage</td>
</tr>
<tr>
<td></td>
<td>- Receive Bandwidth</td>
</tr>
<tr>
<td></td>
<td>- Transmit Bandwidth</td>
</tr>
<tr>
<td></td>
<td>- Rate of Received Packets</td>
</tr>
<tr>
<td></td>
<td>- Rate of Transmitted Packets</td>
</tr>
<tr>
<td></td>
<td>- Rate of Received Packets Dropped</td>
</tr>
<tr>
<td></td>
<td>- Rate of Transmitted Packets Dropped</td>
</tr>
</tbody>
</table>

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 Catalog page to see which version you are currently using, which can be updated, and to check for security vulnerabilities.

5.5.2 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.

5.5.2.1 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.

### 5.5.2.2 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.

There are 2 levels of default server metrics defined by FEP Exporter

<table>
<thead>
<tr>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default mandatory</td>
<td>Are collected by FEP Exporter. These are kept enabled by default by FEP Exporter and can not be disabled by end user.</td>
</tr>
<tr>
<td>Default useful</td>
<td>Useful focused metrics for health and performance metrics. Can be disabled by end user.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Metrics name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>pg_stat_bwriter_*</td>
<td>Maps to view in Statistic Collector</td>
</tr>
<tr>
<td>pg_stat_database_*</td>
<td>Maps to view in Statistic Collector</td>
</tr>
<tr>
<td>pg_stat_database_conflicts_*</td>
<td>Maps to view in Statistic Collector</td>
</tr>
<tr>
<td>pg_stat_archiver_*</td>
<td>Maps to view in Statistic Collector</td>
</tr>
<tr>
<td>pg_stat_activity_*</td>
<td>Maps to view in Statistic Collector</td>
</tr>
<tr>
<td>pg_stat_replication_*</td>
<td>Maps to view in Statistic Collector</td>
</tr>
<tr>
<td>pg_replication_slots_*</td>
<td>Maps to System Catalog pg_replication_slots</td>
</tr>
<tr>
<td>pg_settings_*</td>
<td>Maps to System Catalog pg_settings</td>
</tr>
<tr>
<td>pg_locks_*</td>
<td>Maps to System Catalog pg_locks</td>
</tr>
<tr>
<td>pg_exporter_*</td>
<td>Exposes exporter metrics:</td>
</tr>
<tr>
<td></td>
<td>- last_scrape_duration_seconds (Duration of the last scrape of metrics from PostgresSQL)</td>
</tr>
<tr>
<td></td>
<td>- scrapes_total (Total number of times PostgresSQL was scraped for metrics)</td>
</tr>
<tr>
<td></td>
<td>last_scrape_error (Whether the last scrape of metrics from PostgresSQL resulted in an error; 1 for error &amp; 0 for success)</td>
</tr>
<tr>
<td>pg_*</td>
<td>Exposes exporter metrics</td>
</tr>
<tr>
<td></td>
<td>- pg_up ( set to 1 if the connection to service is success, 0 otherwise )</td>
</tr>
<tr>
<td></td>
<td>- pg_static ( can be used to fetch label short_version / version containing postgres server version information )</td>
</tr>
</tbody>
</table>

Default useful metrics

There are certain useful queries which are additionally added to evaluate the health of the Database system.

<table>
<thead>
<tr>
<th>Metrics name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>pg_capacity_connection_*</td>
<td>Metrics on connections e.g. txns running for 1 hour</td>
</tr>
<tr>
<td>pg_capacity_schema_*</td>
<td>Metrics on disk space of schema</td>
</tr>
<tr>
<td>pg_capacity_tbspace_*</td>
<td>Metrics on disk space of tablespace</td>
</tr>
<tr>
<td>pg_capacity_tblvacuum_*</td>
<td>Metrics on tables without vacuum for days</td>
</tr>
<tr>
<td>pg_capacity_longtx_*</td>
<td>Number of transactions running longer than 5 minutes</td>
</tr>
<tr>
<td>pg_performance_locking_detail_*</td>
<td>Details of processes in blocked state</td>
</tr>
<tr>
<td>pg_performance_locking_*</td>
<td>Number of processes in blocked state</td>
</tr>
<tr>
<td>pg_replication_*</td>
<td>Replication lag behind master in seconds</td>
</tr>
<tr>
<td></td>
<td>Provides the ability to check for the most current data in a reference replica</td>
</tr>
<tr>
<td></td>
<td>To solve the problem, it is necessary to consider measures such as increasing network resources and reducing the load</td>
</tr>
<tr>
<td>pg_postmaster_*</td>
<td>Time at which postmaster started</td>
</tr>
<tr>
<td>pg_stat_user_tables_*</td>
<td>Important statistics from pg_stat_user_tables</td>
</tr>
</tbody>
</table>
### Metrics name Details

- **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_*** Fetched bloat in tables

**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.

Refer an example below.

---

5.5.2.3 Default Alerts

There are few basic alert rules which are setup by the FEP Operator as below

<table>
<thead>
<tr>
<th>Alert rule</th>
<th>Alert Level</th>
<th>Condition persistence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContainerHighCPUUsage</td>
<td>Warning</td>
<td>5 mins</td>
<td>FEP server container/POD CPU usage is exceeding 80% of the resource limits</td>
</tr>
<tr>
<td>ContainerHighRAMUsage</td>
<td>Warning</td>
<td>30 mins</td>
<td>FEP server container/POD memory usage is exceeding 80% of the resource limits</td>
</tr>
<tr>
<td>PVCCLowDiskSpace</td>
<td>Warning</td>
<td>5 mins</td>
<td>A FEP PVC (volume) has less than 10% disk available</td>
</tr>
<tr>
<td>ContainerDisappeared</td>
<td>Warning</td>
<td>60 seconds</td>
<td>FEP server container/POD has disappeared since last 60 seconds</td>
</tr>
<tr>
<td>PostgresqlDown</td>
<td>Error</td>
<td>-</td>
<td>FEP server apparently went down or not accessible</td>
</tr>
<tr>
<td>PostgresqlTooManyConnections</td>
<td>Warning</td>
<td>-</td>
<td>FEP server container/POD connection usage is beyond 90% of its available capacity</td>
</tr>
</tbody>
</table>

**The alerts are based on statistics/metrics. If a platform statistics are incorrect, it may raise an incorrect alarm.**

---
e.g. if the Storage Driver is not showing correct metrics for bytes usage for a PV, system may end up raising incorrect alarm of PVCLowDiskSpace. This behaviour can be seen with NFS storage.

You can configure any alert by adding alert rules to other monitoring items.

### 5.5.2.4 Graphical user interface

User can build their custom dashboard using default and custom metrics.

An example Grafana dashboard screenshot is shown below

![Grafana Dashboard Screenshot](image)

### 5.5.3 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 fep_exporter schema in the postgres database on the FEP server.

<table>
<thead>
<tr>
<th>Table/View name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>pgbackrest_info_backup</td>
<td>Backup Processing Status</td>
</tr>
</tbody>
</table>

#### 5.5.3.1 pgbackrest_info_backup view

Contains one line per backup for information about the state of the backup.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>text</td>
<td>Information identifying the backup</td>
</tr>
<tr>
<td>type</td>
<td>text</td>
<td>full: full backup, incr: incremental backup</td>
</tr>
<tr>
<td>prior</td>
<td>text</td>
<td>Label of the backup that should be applied first (For incremental backups only)</td>
</tr>
<tr>
<td>database_size</td>
<td>bigint</td>
<td>Database size</td>
</tr>
<tr>
<td>database_size_comp</td>
<td>bigint</td>
<td>Database size (After Compression)</td>
</tr>
<tr>
<td>backup_size</td>
<td>bigint</td>
<td>Backup size</td>
</tr>
<tr>
<td>backup_size_comp</td>
<td>bigint</td>
<td>Backup size (After Compression)</td>
</tr>
<tr>
<td>archive_start</td>
<td>text</td>
<td>Range of WALs required for restore (Start)</td>
</tr>
<tr>
<td>archive_stop</td>
<td>text</td>
<td>Range of WALs required for restore (End)</td>
</tr>
</tbody>
</table>
### 5.5.4 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 fep_exporter schema in the postgres database on the FEP server.

<table>
<thead>
<tr>
<th>Table/View name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>pgpool2_stat_load_balance</td>
<td>Load Balance Information in pgpool2</td>
</tr>
<tr>
<td>pgcluster_stat_replication</td>
<td>Replication State</td>
</tr>
<tr>
<td>pgpool2_stat_conn_pool</td>
<td>Connection Pool State for pgpool2</td>
</tr>
<tr>
<td>pgpool2_stat_sql_command</td>
<td>SQL Command Statistics</td>
</tr>
</tbody>
</table>

#### 5.5.4.1 pgpool2_stat_load_balance view

Contains one row for MasterService and one row for ReplicaService.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_id</td>
<td>integer</td>
<td>database node id (0 or 1)</td>
</tr>
<tr>
<td>status</td>
<td>text</td>
<td>status (up or down)</td>
</tr>
<tr>
<td>lb_weight</td>
<td>double</td>
<td>load-balancing weight</td>
</tr>
<tr>
<td>role</td>
<td>text</td>
<td>role (primary or standby)</td>
</tr>
<tr>
<td>last_status_change</td>
<td>timestamp</td>
<td>last status change time</td>
</tr>
</tbody>
</table>

#### 5.5.4.2 pgpool2_stat_conn_pool view

Indicates the state of the connection pool. Contains connection pool information for each pcpool2 instance.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pgpool2_node_id</td>
<td>integer</td>
<td>pgpool2 node id (0 - the number of pgpool2 instance - 1)</td>
</tr>
<tr>
<td>pool_pid</td>
<td>integer</td>
<td>The PID of the displayed Pgpool-II process</td>
</tr>
<tr>
<td>start_time</td>
<td>timestamp</td>
<td>The timestamp of when this process was launched</td>
</tr>
<tr>
<td>pool_id</td>
<td>integer</td>
<td>The pool identifier (should be between 0 and max_pool - 1)</td>
</tr>
<tr>
<td>backend_id</td>
<td>integer</td>
<td>The backend identifier (should be between 0 and the number of configured backends minus one)</td>
</tr>
<tr>
<td>role</td>
<td>text</td>
<td>role (primary or standby)</td>
</tr>
<tr>
<td>database</td>
<td>text</td>
<td>The database name for this process's pool id connection</td>
</tr>
<tr>
<td>username</td>
<td>text</td>
<td>The user name for this process's pool id connection</td>
</tr>
<tr>
<td>Column</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>create_time</td>
<td>timestamp</td>
<td>The creation time and date of the connection</td>
</tr>
<tr>
<td></td>
<td>with timezo</td>
<td></td>
</tr>
<tr>
<td>majorversion</td>
<td>integer</td>
<td>The protocol version numbers used in this connection</td>
</tr>
<tr>
<td>minorversion</td>
<td>integer</td>
<td>The protocol version numbers used in this connection</td>
</tr>
<tr>
<td>pool_counter</td>
<td>integer</td>
<td>Counts the number of times this pool of connections (process) has been used by clients</td>
</tr>
<tr>
<td>pool_connected</td>
<td>boolean</td>
<td>True (1) if a frontend is currently using this backend</td>
</tr>
</tbody>
</table>

5.5.4.3 pgpool2_stat_sql_command view

Represents SQL command statistics.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_id</td>
<td>integer</td>
<td>The backend identifier (should be between 0 and the number of configured backends minus one)</td>
</tr>
<tr>
<td>role</td>
<td>text</td>
<td>role (primary or standby)</td>
</tr>
<tr>
<td>select_cnt</td>
<td>integer</td>
<td>The numbers of SQL command: SELECT</td>
</tr>
<tr>
<td>insert_cnt</td>
<td>integer</td>
<td>The numbers of SQL command: INSERT</td>
</tr>
<tr>
<td>update_cnt</td>
<td>integer</td>
<td>The numbers of SQL command: UPDATE</td>
</tr>
<tr>
<td>delete_cnt</td>
<td>integer</td>
<td>The numbers of SQL command: DELETE</td>
</tr>
<tr>
<td>ddl_cnt</td>
<td>integer</td>
<td>The numbers of SQL command: DDL</td>
</tr>
<tr>
<td>other_cnt</td>
<td>integer</td>
<td>The numbers of SQL command: others</td>
</tr>
<tr>
<td>panic_cnt</td>
<td>integer</td>
<td>The numbers of failed commands</td>
</tr>
<tr>
<td>fatal_cnt</td>
<td>integer</td>
<td>The numbers of failed commands</td>
</tr>
<tr>
<td>error_cnt</td>
<td>integer</td>
<td>The numbers of failed commands</td>
</tr>
</tbody>
</table>

5.6 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.

5.6.1 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.

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.

- fepbackup - During FEPBackup CR creation
  - Event is raised when FEPBackup cronjob1 creation is successful and when FEPBackup cronjob1 creation fails.
  - Event is raised when FEPBackup cronjob2 creation is successful and when FEPBackup cronjob2 creation fails.
  - Event is raised when FEPBackup cronjob3 creation is successful and when FEPBackup cronjob3 creation fails.
  - Event is raised when FEPBackup cronjob4 creation is successful and when FEPBackup cronjob4 creation fails.
  - Event is raised when FEPBackup cronjob5 creation is successful and when FEPBackup cronjob5 creation fails.

- feppgpool2- 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

- feppgpool2cert- During FEPPgPool2Cert CR creation
  - Event is raised when FEPPgPool2Cert CR creation is successful, when FEPPgPool2Cert CR creation 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.

5.6.2 Viewing the custom events

The custom events can be viewed on CLI as well as the Openshift console

On cli

Executing the command

cubectl get events

OR

oc get events

Following is a snippet of the events output is ==shown when the above command is executed,

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.

- 86 -
5.7 Scaling Replicas

5.7.1 Auto Scale Out

Auto 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 Auto scale out. Refer to "FEPCluster Parameters" in the Reference for information about the values to specify.

```
$ oc edit fepcluster <FEPClusterCR name>
```

5.7.2 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.

5.8 Backing Up to Object Storage

Describes how to store backup data in object storage.

5.8.1 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
```
5.8.2 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 “5.8.1 Storing CA Files (Root Certificates)” for caName.

FEPCluster Custom Resource Example: Only Object Storage Used for Backup Repository

```yaml
apiVersion: fep.fujitsu.io/v2
kind: FEPCluster
metadata:
  ...
spec:
  fepChildeCrVal:
    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=/etc/pki/tls/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

```yaml
...
spec:
  fepChildeCrVal:
    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=/etc/pki/tls/certs/ca.crt
      pgbackrestKeyParams: |
        repo2-s3-key=SAMPLEKEY
        repo2-s3-key-secret=SAMPLESECRET
      caName: |
        - storage-cacert
  ...
```

5.9 Disaster Recovery

By using OSS (pg_backrest) functionality to store backup data in object storage, data can be migrated to a database cluster in a different OCP environment.
Even if it is difficult to operate in an OCP environment with a database cluster due to a disaster, it is possible to continue operating in a different OCP environment.

### 5.9.1 Disaster Recovery Prerequisites

The configuration diagram of the pod placement and backup repository, which are prerequisites for the backup feature for performing disaster recovery, 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.

![Configuration Diagram]

**Note**

The definition of the FEPCluster custom resource is not inherited when performing disaster recovery.

We recommend that you save your production site FEPCluster custom resource definitions in case of a disaster.

### 5.9.2 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.

#### 5.9.2.1 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
```

#### 5.9.2.2 Defining a FEPCluster Custom Resource

In addition to the FEPCluster settings, specify the Restore settings below.

FEPCluster Custom Resource Example

```yaml
apiVersion: fep.fujitsu.io/v1
kind: FEPCluster
metadata: ...
spec:
  fepChildCrVal:
    restore:
      pgbackrestParams: |
      repol-type=s3
      repol-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=/etc/pki/tls/certs/ca.crt
pgbackrestKeyParams:
  repo1-s3-key=SAMPLEKEY
  repo1-s3-key-secret=SAMPLESECRET
cName: "storage-cacert"
...
```

Setting value

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec.fepChildCrVal.restore</td>
<td></td>
<td>Define when restoring by specifying the backup data stored in the object storage.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.pgbackrestParams</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>&quot;</td>
<td>is fixed, and the following lines specify the parameters to set in pgbackrest.conf.</td>
</tr>
<tr>
<td></td>
<td>Specify the object storage where the backup data is stored.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you want to use a root certificate other than the default, specify the following:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>repo1-storage-ca-path=/pgbackrest/storage-certs/&lt;file name&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Register the CA file in ConfigMap and specify the ConfigMap name in spec.fepChildCrVal.restore.caName.</td>
<td></td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.pgbackrestKeyParams</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>&quot;</td>
<td>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.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.caName</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Specify when you use a CA file other than the system default.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specify the name of the created ConfigMap in list format.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The specified ConfigMap will be mounted in /pgbackrest/storage-certs.</td>
<td></td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.spec.limits</td>
<td>cpu: 200m memory: 300Mi</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>CPU and memory allocated to the container performing the restore.</td>
<td></td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.spec.requests</td>
<td>cpu: 100m memory: 200Mi</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>CPU and memory allocated to the container performing the restore.</td>
<td></td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.restoretype</td>
<td>latest</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Restore Type (latest or PITR)</td>
<td></td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.restoredate</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Specify the date to restore when spec.fepChildCrVal.restore.restoretype is “PITR”.</td>
<td></td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.restoretime</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Specify the time to restore when spec.fepChildCrVal.restore.restoretype is “PITR”.</td>
<td></td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.image</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Image of the container to perform the restore.</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Default</td>
<td>Details</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It is omitted by default. In this case, the URL for image is obtained from the operator container environment.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.restore.imag.ePullPolicy</td>
<td>IfNotPresent</td>
<td>Optional</td>
</tr>
</tbody>
</table>
Chapter 6 Maintenance Operations

This chapter describes the maintenance operation after deploying the container.

### 6.1 Minor Version Upgrade

Minor FEP version upgrade is done by replacing the image in FEPCluster customer resource with a new one. For the procedure, refer to "Minor Version Upgrade" in the Overview.

Update information can be found in the Red Hat catalog to see if a new FEP database server container has been released.

Updates are rolling updated, so you can localize downtime, but it is recommended that you avoid running during business hours as connected applications will result in connection errors.

**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 FEPAction CR to update FEPAction CR.

Equivalent Kubernetes command: kubectl apply -f <new_spec>

"switchover" action type expects users to specify the name of the current leader/primary pod that they want to switchover from. Specify the name in the args section under the FEPAction CR spec as below:

```yaml
spec:
  fepAction:
    args:
      - new-fep-sts-2
    type: switchover
    targetClusterName: new-fep
```

Here, new-fep-sts-2 is the current primary.

Refer to "FEPAction 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.
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:

```yaml
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"

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec.fepChildCrVal.upgrade</td>
<td>Optional</td>
<td>When this field is defined, a major version upgrade is performed.</td>
</tr>
<tr>
<td>Field</td>
<td>Default</td>
<td>Details</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>However, if spec.fepChildCrVal.restore is defined, the FEPCluster build stops.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.sourceCluster</td>
<td></td>
<td>Specify the FEPCluster CR name of the data migration source. Be sure to specify spec.fepChildCrVal.upgrade when defining it.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.mcSpec.limits</td>
<td>cpu: 200m memory: 300Mi</td>
<td>Optional Specify the maximum number of resources allocated to the upgrade execution container.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.mcSpec.requests</td>
<td>cpu: 100m memory: 200Mi</td>
<td>Optional Specify the lower limit of resources allocated to the upgrade execution container.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.image</td>
<td></td>
<td>Optional If omitted, the URL of the image is obtained from the operator container environment.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.imagePullPolicy</td>
<td>IfNotPresent</td>
<td>Optional Specify the pull policy for the container image. - Always - IfNotPresent - Never</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.source.pgAdminTls.certificateName</td>
<td></td>
<td>Optional If the data source FEPCluster used &quot;cert&quot; 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 &quot;postgres&quot; in the data source. Refer to &quot;4.5.1 Manual Certificate Management&quot; for information about creating secrets.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.destination.pgAdminTls.certificateName</td>
<td></td>
<td>Optional</td>
</tr>
</tbody>
</table>

- 95 -
<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>spec.fepChildCrVal.upgrade.storage</td>
<td>Optional</td>
<td>Defines storage for storing dump files.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.storage.storageClass</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If omitted, the default storage class of the operating environment will be used.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.storage.size</td>
<td>2Gi</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specify the size of the storage to store the dump file.</td>
</tr>
<tr>
<td>spec.fepChildCrVal.upgrade.storage.accessModes</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Storage access mode for storing dump files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>As an array of access modes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>e.g. [ReadWriteMany]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If omitted, it is treated as [ReadWriteOnce].</td>
</tr>
</tbody>
</table>

**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:

```bash
$ kubectl get event
LAST SEEN   TYPE     REASON               OBJECT                                     MESSAGE
164m        Normal  SuccessfulFepUpgrade fepupgrade/<Name of the new FEPCluster CR> <namespace>, Successfully FEP Upgrade
```

In addition, the following annotation will be added to YAML in FEPClusterCR:

```yaml
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:

```bash
$ kubectl get event
LAST SEEN   TYPE     REASON               OBJECT                                     MESSAGE
164m        Warning  FailedFepUpgrade  fepupgrade/<Name of the new FEPCluster CR> <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.

```bash
$ kubectl describe fepupgrade/<Name of the new FEPCluster CR>
```

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.
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.

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.

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

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>fep-ansible-operator-7dc5fd9bf7-4 smzk</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>20m</td>
</tr>
</tbody>
</table>

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.
Appendix A  Quantitative Values and Limitations

## A.1 Quantitative Values

Refer to the FUJITSU Software 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 "5.1 Configuration Change" and reapply. Depending on the parameters to be changed, the container may be redeployed. Refer to "5.1 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 don’t allow it to be 100% functionally capable to VM based server instance. The known limitations are as below.

<table>
<thead>
<tr>
<th>No</th>
<th>Limitation</th>
<th>Reason for Limitation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No Support for JIT</td>
<td>Since UBI8 is not having requisite LLVM libraries</td>
<td>It is not possible to enable JIT in postgreSQL.conf.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impact for the customer is that they are not able to achieve maximum performance capabilities on given CPU and memory.</td>
</tr>
<tr>
<td>2</td>
<td>FEP parallelism improvements</td>
<td>Since UBI8 is not hosting dstat binaries</td>
<td>FEP parallelism improvement is to restrict number of parallel workers in case the CPU is already busy because of other tasks/processes. It is unlikely to have too much impact on FEP container, since container is running only one process.</td>
</tr>
<tr>
<td>3</td>
<td>Crypto Express cards are not supported</td>
<td>IBM LinuxOne doesn’t support CryptoExpress cards in OpenShift container platform at this stage.</td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>No Support for Oracle foreign data wrapper</td>
<td>Oracle foreign data wrapper has dependency on Instant Client package, which is not available.</td>
<td>Oracle InstantClient package is not redistributed by FUJITSU Enterprise Postgres leading to this limitation. The functionality of Oracle Foreign data wrapper is not available to FUJITSU Enterprise Postgres on OpenShift environment.</td>
</tr>
</tbody>
</table>

### 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".

- 100 -
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.
2. Both the Statefulset and its resulting pods will be annotated with your provided annotations: archivewalVol and backupVol must be ReadWriteMany.
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)

```yaml
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)

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: < PV name >
spec:
  capacity:
    storage: < Capacity Required ex.8Gi >
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  mountOptions:
  - hard
  nfs:
```
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`

```yaml
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 ~
```