## Contents

1. What’s New
2. EDB Containers
3. Preparing to Deploy an Advanced Server Container
   3.1 Deployment Prerequisites
   3.1.1 Creating a Local Repository
4. Deploying a Container
   4.1 Deploying Containers using Operators
   4.2 Deploying Containers using a Helm Chart
   4.3 Deploying Containers using Templates
   4.4 Deploying from the Docker Command Line
      4.4.1 Creating a Container at the Docker Command Line
         4.4.1.1 Deploying Advanced Server in a Container (Master Node)
         4.4.1.2 Deploying Advanced Server in a Container (Standby Node)
         4.4.1.3 Deploying pgPool in a Container
         4.4.1.4 Deploying BART in a Container
5. Using the OpenShift Console
   5.1 Scaling an Advanced Server Deployment
   5.2 Using a ConfigMap Object to Customize a Configuration
   5.3 Performing a Rolling Update
   5.4 Performing a Manual Backup
      5.4.1 Restoring a Cluster from Backup
      5.4.2 Performing a Point-in-Time Recovery (PITR)
   5.5 Preparing an Advanced Server Cluster for Monitoring
      5.5.1 Deploying a PEM Server
      5.5.2 Deploying a PEM Server using a Helm Chart
      5.5.3 Launching a PEM Agent on Advanced Server Cluster
   5.6 Removing a Project
      5.6.1 Retaining a Project with No Pods
   5.7 Upgrading an Advanced Server Container
6. Reference
   6.1 Uploading Containers to Network-Isolated Environments
   6.2 Remotely Accessing an Advanced Server Container
EDB Postgres Platform for Containers allows you to use a Docker-formatted container to deploy and manage EDB Postgres Advanced Server (Advanced Server) and supporting components in a Red Hat OpenShift environment. OpenShift automation provides an environment in which you can easily:

- Deploy or disable Advanced Server instances as needed.
- Automatically scale an Advanced Server instances to meet application requirements.
- Easily ensure Failover Manager protection for your data.
- Utilize load balancing to distribute read and write requests across the available servers.
- Deploy Advanced Server instances with custom configurations in a container environment.

The EDB Postgres Platform for Containers automates the deployment of containers that include Advanced Server and the following supporting components:

- EDB Failover Manager (EFM)
- EDB Postgres Backup and Recovery Tool (BART)
- pgPool (connection pooling for Postgres databases)
- EDB Postgres Enterprise Manager (PEM)

For detailed information and documentation for each component, please visit the EnterpriseDB website. Sample scripts and yaml files for EDB Postgres containers are available at the EnterpriseDB website.
CHAPTER 1

What’s New

The following changes are added to EDB™ Postgres on Kubernetes OpenShift Guide guide to create version 2.7:

• Starting this release, the containers can be deployed in OpenShift using the following methods in addition to using templates:
  – Operator
  – Helm chart
• Locale support for Advanced Server
To deploy a container, you must have access to the registry in which the containers are stored (containers.enterprisedb.com). To receive credentials that allow you to access the container registry, please submit a request at: https://www.enterprisedb.com/general-inquiry-form. The following containers are available in the EDB container registry at containers.enterprisedb.com:

**EDB Postgres Advanced Server with Failover Management and Monitoring components:**
- containers.enterprisedb.com/edb/edb-as:v12
- containers.enterprisedb.com/edb/edb-as:v11
- containers.enterprisedb.com/edb/edb-as-gis:v11
- containers.enterprisedb.com/edb/edb-as-gis:v12
- containers.enterprisedb.com/edb/edb-as:v10
- containers.enterprisedb.com/edb/edb-as-gis:v10
- containers.enterprisedb.com/edb/edb-as-gis23:v10
- containers.enterprisedb.com/edb/edb-as:v9.6

**EDB Postgres Advanced Server only:**
- containers.enterprisedb.com/edb/edb-as-lite:v12
- containers.enterprisedb.com/edb/edb-as-lite:v11
- containers.enterprisedb.com/edb/edb-as-lite:v10

**EDB Backup and Recovery Tool (BART):**
- containers.enterprisedb.com/edb/edb-bart:v2.5

**EDB Postgres Enterprise Manager (PEM):**

**EDB PgPool:**
- containers.enterprisedb.com/edb/edb-pgpool:v4.0
Note: You can also download all of these images as tarballs from the EDB website.

The containers use the UBI (Universal Base Image) image from Red Hat as the base OS. For more information about UBI images, refer to: https://developers.redhat.com/products/rhel/ubi/
Preparing to Deploy an Advanced Server Container

Red Hat OpenShift is an open source container application platform based on the Kubernetes container orchestrator for enterprise application development and deployment. EDB Postgres Platform for Containers is supported on Red Hat OpenShift Container Platform (previously called OpenShift Enterprise), Red Hat OpenShift Origin, and Red Hat OpenShift Online environments.

OpenShift supports multi-node Advanced Server clusters that include deployments of Advanced Server (with EDB Failover Manager), EDB Postgres BART, and pgPool.

EDB Postgres Platform for Containers are supported on OpenShift version 3.6 or later. For information about OpenShift Container Platform and OpenShift Online, please visit OpenShift website.

For information about OpenShift Origin, please visit OpenShift Origin website.
3.1 Deployment Prerequisites

To deploy a container, you must have access to the repository in which the containers are stored (containers.enterprisedb.com). To receive credentials that allow you to access the container repository, please submit a request.

Containers can be deployed on OpenShift using one of the three methods:

- Operators
- Helm charts
- Templates

Some prerequisite steps and advanced management tasks must be performed at the OpenShift command line. For detailed information about using the OpenShift command line, see the project documentation.

Before deploying the EDB Postgres Platform for Containers using any of the three methods, you must:

Create an OpenShift Project

Use the OpenShift command line utility to login and create a project. Include the \(-u\) flag and specify a user name when connecting and enter the password when prompted:

\[oc login -u <user_name>\]

Then, create a project:

\[oc new-project <project_name>\]

Where \(project_name\) specifies the name of the new project.

3.1.1 Creating a Local Repository

To create a local repository that contains EDB Postgres images, you must create a local Docker registry. For example, the following command creates and starts a registry on localhost:5000:

\[docker run -d -p 5000:5000 --restart=always --name registry registry:2\]

The following commands will push the Advanced Server image from the EnterpriseDB repository to a local repository (localhost:5000):

\[docker pull containers.enterprisedb.com/edb/edb-as:v10\]
\[docker tag containers.enterprisedb.com/edb/edb-as:v10 localhost:5000/edb/edb-as:v10\]
\[docker push localhost:5000/edb/edb-as:v10\]

Use similar steps to create pgPool (edb-pgpool:v3.5) and BART (edb-bart:v2.0) repositories.

For information about creating a local Docker registry, please see the Docker documentation.
You can use a number of methods to deploy EDB Postgres Containers:

- Deployment with Operators
- Deployment with Helm Charts
- Deployment with Templates
- Deployment at the Docker Command Line

### 4.1 Deploying Containers using Operators

For information about deploying containers using Operators, see the Postgres Containers Operator Guide.

### 4.2 Deploying Containers using a Helm Chart

Before deploying the EDB Postgres Platform for Containers using Helm charts, you must install the latest Helm Client. Then, to deploy EDB Postgres Containers on OpenShift using Helm charts, complete the following steps:

1. **Create an OpenShift project.**

2. **Create and Configure a Storage Class:** Before creating any volumes resources, you must create a storage class for your OpenShift environment. For more information, see the OpenShift Storage Class documentation for more information.

**Sample storage class examples**

AWS ElasticBlockStore (EBS) Object Definition:

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
name: slow
```
To create an EBS Storageclass on Openshift, run the following command:

```bash
oc apply -f aws-ebs-storageclass.yaml
```

GCE PersistentDisk (gcePD) Object Definition:

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: slow
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
  zone: us-central1-a
```

To create a gcePD Storageclass on Openshift, run the following command:

```bash
oc apply -f gce-pd-storageclass.yaml
```

GlusterFS Object Definition:

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: slow
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://127.0.0.1:8081"
  restuser: "admin"
  secretName: "heketi-secret"
  secretNamespace: "default"
  gidMin: "40000"
  gidMax: "50000"
```

To create a GlusterFS Storageclass on Openshift, run the following command:

```bash
oc apply -f glusterfs-storageclass.yaml
```

3. Deploy an Advanced Server Helm Chart

To deploy the containers in your environment, complete the following steps:

Step 1 - Visit the EnterpriseDB website to sign up for credentials that allow you to access the containers repository at:

```bash
containers.enterprisedb.com
```

Step 2 - Download the following files:

- Sample helm chart: edb-2.7.0.tgz
- Sample values.yaml file: sample-values.yaml
Step 3 - Create your own `values.yaml` file with customized input values using the sample `values.yaml` as a template (e.g. `myvalues.yaml`).

Step 4 - Create a project namespace.

```
oc new-project <project-namespace>
```

Step 5 - Enter the following command to deploy the containers (i.e. install the helm chart) with your custom inputs:

```
helm install <helm-chart-name> edb-2.7.0.tgz -f myvalues.yaml
--namespace <project-namespace>
```

Where `myvalues.yaml` is the user-supplied `values.yaml` file containing the desired parameter values.

Step 6 - Once the helm chart is successfully installed, the following EDB containers should be available in your environment:

- **EDB Postgres Advanced Server pod (includes EDB Failover Manager)**
  - Container image: `edb-as:v12 / edb-as:v11`
  - Number of replicas: 3
  - Deployed as: Stateful Set
  - Each Advanced Server pod is deployed on a separate node (anti-affinity rule applied)

- **EDB PgPool pod**
  - Container image: `edb-pgpool:v4.0`
  - Number of replicas: 2
  - Deployed as: Replica Set
  - Each PgPool container is set on a separate node (anti-affinity rule applied)

- **EDB PEM Server (Postgres Enterprise Manager) pod**
  - Container image: `edb-pemserver:v7.12`
  - Number of replicas: 1
  - Deployed as: Replica Set

- **(Optionally) EDB BART (Backup and Recovery Tool) pod**
  - Container image: `edb-bart:v2.5`
  - Number of replicas: 1
  - Deployed as: Replica Set
4.3 Deploying Containers using Templates

A template is a .yaml file that describes a set of objects that can be parameterized and processed to produce a list of objects for creation by OpenShift Container Platform. To create and upload a template to your current project's template library, use the command:

```
    oc create -f template-yaml-file
```

After creating the template, you can customize parameter values during deployment. You can download sample scripts, templates, and related yaml files for the examples shown in the guide to use as a starting point.

To deploy EDB Postgres Containers on OpenShift using Templates, complete the following steps:

1. **Create an OpenShift project.**
2. Grant the required access privileges.
   Use the following commands to grant cluster role privileges to the default service account associated with the project:
   ```
   oadm policy add-cluster-role-to-user
   edit system:serviceaccount:<project_name>:default
   
   oc policy add-role-to-user view
   system:serviceaccount:<project_name>:default
   ```
   Where `project_name` specifies the name of the project.
3. Create and Configure the Volume Resources.
   Advanced Server and BART containers require persistent storage. In OpenShift, persistent storage is provided through persistent volume (PV) and persistent volume claims (PVC). Before deploying an Advanced Server or BART container, an administrative user must provision persistent storage by creating the required persistent volumes, and developers must create persistent volume claims for the project to request PV resources. The administrator can choose from a variety of storage backends such as local volumes or shared volumes (NFS, glusterFS, etc.) to configure the persistent volumes. The volume on which Advanced Server resides maybe either local or shared, but BART must reside on a non-local volume.
   For OpenShift Enterprise and OpenShift Origin environments, the following sections provide examples of creating and configuring PVs and PVCs using two different storage backends, local volumes, and glusterfs-based shared volumes.
   For more information about OpenShift volumes, please see the OpenShift documentation.
   If you are using an OpenShift Online environment, use the web console to create the persistent volumes. For more information, click here.

   **Using Local Volumes**
   To use a local volume, the administrator must create the directories in which the volume will reside and ensure that the permissions on the directory allow write access.

   **Note:** Local volumes are not recommended for production environments.

   Use the following commands to configure the directory:
   ```
   mkdir <path_to_volume_directory>
   chmod -R 777 <path_to_volume_directory>
   ```
chcon -Rt svirt_sandbox_file_t <path_to_volume_directory>

Where:

path_to_volume_directory is the complete path to the directory in which the volume will reside.

**Note:** The directory must be created on each OpenShift node, and necessary privileges granted in OpenShift to use the local (hostPath) volumes.

Once the directory (directories in case of multiple nodes), the following sample YAML files create a persistent volume and volume claim for local volumes.

**The local persistent-volume.yaml file**

Include the following content when defining a local persistent volume; customize the content for your host(s):

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: <persistent_volume_name>
spec:
  capacity:
    storage: 10Gi
  accessModes:
  - ReadWriteOnce
hostPath:
  path: <path_to_volume_directory>
persistentVolumeReclaimPolicy: Retain
```

Where:

- **persistent_volume_name** is the name of the persistent volume.
- **path_to_volume_directory** is the directory in which the volume will reside.

**The local persistent-volume-claim.yaml File**

Include the following content when defining a local persistent volume claim:

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <persistent_volume_claim_name>
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
  volumeName: <persistent_volume_name>
```

Where:

- **persistent_volume_claim_name** is the name of the persistent volume claim.
- **persistent_volume_name** is the name of the associated persistent volume.

**Using Shared Volumes**

To use shared volumes based on a glusterfs-based storage backend, use the following the YAML files to create the endpoint, the persistent volume, and the volume claim files.
The glusterfs-endpoint.yaml file

```yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: <glusterfs_endpoint_name>
subsets:
- addresses:
  - ip: <glusterfs-node1-ipaddress> # node1 IP
    ports:
    - port: 1
  - addresses:
    - ip: <glusterfs-node2-ipaddress> # node2 IP
    ports:
    - port: 1
```

Where:
- **<glusterfs_endpoint_name>** is the name of the glusterfs endpoint.
- **<glusterfs-node1-ipaddress>** is the IP address of node 1 of the glusterfs cluster.
- **<glusterfs-node2-ipaddress>** is the IP address of node 2 of the glusterfs cluster.

The shared persistent-volume.yaml file

Include the following content when defining a shared persistent volume; customize the content for your host(s):

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: <persistent_volume_name>
spec:
  capacity:
    storage: 4Gi
  accessModes:
  - ReadWriteMany
  glusterfs:
    endpoints: <glusterfs_endpoint_name>
    path: <glusterfs_volume>
    readOnly: false
    persistentVolumeReclaimPolicy: Retain
```

Where:
- **<persistent_volume_name>** is the name of the persistent volume.
- **<glusterfs_endpoint_name>** is the name of the glusterfs endpoint.
- **<glusterfs_volume>** is the name of the glusterfs volume.

The shared persistent-volume-claim.yaml file

Include the following content when defining a shared persistent volume claim:

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <persistent_volume_claim_name>
spec:
  accessModes:
  - ReadWriteMany
```

Where:
- **<persistent_volume_claim_name>** is the name of the persistent volume claim.
resources:
  requests:
    storage: 2Gi
  volumeName: <persistent_volume_name>

Where:

- `<persistent_volume_claim_name>` is the name of the persistent volume claim.
- `<persistent_volume_name>` is the name of the associated persistent volume.

4. Create a Registry Secret

Use the `oc secrets` command to create an OpenShift Secret object that contains connection information for the EnterpriseDB repository:

- Create a `.docker/config.json` file in your home directory using the following format:

  ```
  auths:
    <registry-url>:
      auth: "<encrypted_repo_user_name:repo_user_password>"
      email: "<user@email>"
  ```

  Where `<registry-url>` is the URL of the EDB container registry. Auth is the encrypted username:password for the EDB container registry.

  To generate the `encrypted_repo_user_name:repo_user_password`, run `echo -n 'repo_user_name:repo_user_password' | base64` on the command line, substituting `repo_user_name` with your EDB repository username and `repo_user_password` with your EDB repository password. `<user@email>` specifies the email address of the user.

- Create the secret from your local `.docker/config.json` file:

  ```bash
  $ oc create secret generic <secret_name> --from-file=.dockerconfigjson --type=kubernetes.io/dockerconfigjson
  ```

  Where: `secret_name` specifies the name of the secret.

  You must add the secret to your service account; include the default keyword to use the default service account:

  ```bash
  oc secrets add serviceaccount/default secrets/<secret_name> --for=pull
  ```

  The following command creates a secret that can be mounted in a pod:

  ```bash
  oc secrets add serviceaccount/builder secrets/<secret_name>
  ```

  You can also use OpenShift Secret objects to manage sensitive information such as passwords. For more information about using OpenShift Secret objects, see the OpenShift documentation.
4.4 Deploying from the Docker Command Line

You can use the Docker command-line client to deploy, manage, and use an Advanced Server container. We recommend including the following docker command options when using the command line:

- `-d`
  The `-d` option forces docker to run in the background as a daemon. This option is optional.

- `--privileged`
  The `--privileged` option may be required if local security settings do not permit mounting a local volume with read-write options. As an alternative, we recommend allowing your security settings to permit the container to have read-write access to the mounted volume. If applicable, adjust your SELinux settings to allow access.

- `--restart=always`
  This option specifies your restart preferences for the container. Refer to the Docker documentation for details about the supported options. For more information about docker commands and command options, see Docker base commands.

4.4.1 Creating a Container at the Docker Command Line

Use the `docker run` command to deploy a container at the command line. You can specify environment variables at the command line (when invoking the docker run command), or (for convenience) in a file located in the network mountpoint. See the Docker documentation for a complete list of command options.

The number of nodes that you must deploy will depend on the features that you wish to use with your system. For example, if you are using EFM functionality, you will need to create at least one master and two standby nodes.

When you use the Docker command line to create standby nodes or containers that provide complimentary functionality for Advanced Server, you specify information about the configuration of the cluster, and the names of master or standby nodes of the cluster. Please note that information provided in the container configuration must be consistent across all nodes of the cluster.
4.4.1.1 Deploying Advanced Server in a Container (Master Node)

The master node of a cluster contains an instance of Advanced Server and a supporting Failover Manager deployment. When deploying a master node, include the following syntax:

```
sudo docker run \
  --name "master_name" \
  -e DATABASE_NAME="db_name" \
  -e PGPORT=db_listener_port \n  -e DATABASE_USER="db_user_name" \n  -e DATABASE_USER_PASSWORD="db_user_password" \n  -e ENTERPRISEDB_PASSWORD="enterprisedb_password" \n  -e LOCALEPARAMETER="locale" \n  -e NAMESERVER="nameserver_ipaddress" \n  -e REPL_USER="repl_user_name" \n  -e REPL_PASSWORD="repl_user_password" \n  -e ENABLE_HA_MODE=Yes|No \n  -e ENABLE_ARCHIVE_MODE=Yes|No \n  -v host_data_volume:/edbvolume \n  -v host_archive_volume:/edbarchive \n  -p host_port:db_listener_port \n  -e ACCEPT_EULA=Yes|No \n  -d containers.enterprisedb.com/edb/edb-as:v12
```

Include the --name option to specify the name of the master node of the cluster. Include the -e option with the docker run command to specify values for environment variables used to configure your container. Include the following values when deploying a master node that includes Advanced Server and Failover Manager:

- Use the DATABASE_NAME environment variable to specify the name of the Advanced Server database.
- Use the PGPORT environment variable to specify the listener port of the Advanced Server database (by default, 5444).
- Use the DATABASE_USER environment variable to specify the name of a database superuser that will be created when the database is initialized; by default, the database superuser is named enterprisedb.
- If you specify the default (enterprisedb), the user will be associated with the password provided by the ENTERPRISEDB_PASSWORD environment variable.
- Use the DATABASE_USER_PASSWORD environment variable to specify the password associated with the database superuser if you specify a db_user_name other than enterprisedb. Please note that this password should not be changed after the pod is initialized.
- Use the ENTERPRISEDB_PASSWORD environment variable to specify the password associated with the default database superuser (enterprisedb). During the installation process, the container creates a database superuser named enterprisedb. Please note that this password should not be changed after the pod is initialized.
- Use the LOCALEPARAMETER environment variable to specify the locale that will be used by the container.
- Use the INITDBOPTS parameter to provide non-default options for the initdb operation, e.g., this parameter can be used to disable Oracle compatibility.
- Use the NAMESERVER environment variable to specify the identity of a name server that will be used for notifications from Failover Manager.
- Use the REPL_USER environment variable to specify the name of the Postgres streaming replication user.
- Use the REPL_PASSWORD environment variable to specify the password associated with the replication user.
- Use the ENABLE_HA_MODE environment variable to specify whether failover should be enabled.
- Use the ENABLE_ARCHIVE_MODE environment variable to specify whether archiving should be enabled. If ENABLE_HA_MODE is set to Yes, then ENABLE_ARCHIVE_MODE must be set to Yes also.
Map the mount point of the data volume of the container, /edbvolume, to a directory on the host system using the -v option; for example, the following option:

```
-v /data/edbas:/edbvolume
```

Edges the mount point of the data volume of the container, /edbvolume, to a directory on the host system using the -v option; for example, the following option:

```
-v /archive/edbas:/edbarcive
```

If ENABLE_ARCHIVE_MODE is set to Yes, then map the mount point of the archive volume of the container, /edbarcive, to a directory on the host system using the -v option; for example, the following option:

```
-v /archive/edbas:/edbarcive
```

Use the ACCEPT_EULA environment variable to indicate if you accept or decline the EnterpriseDB license. The license is available for review at:

```
https://www.enterprisedb.com/node/4509
```

Note: You must accept the license agreement to deploy a container.

Include the -p option to enable port forwarding from the container to the host machine; for example, `-p 5445:5444` makes the listener port 5444 in the container available on port 5444 on the host system.

Include the -d option to indicate that the container should be deployed in detached mode; this enables background processing for the deployment.

After providing values for the environment variable and including any docker options, specify the repository name and image you wish to use.
4.4.1.2 Deploying Advanced Server in a Container (Standby Node)

A standby container hosts an instance of Advanced Server that is replicating the Advanced Server master node. If Failover Manager detects a failure on the part of the master node, a standby will assume the role of master node for the cluster. When deploying a standby node, include the following syntax:

```bash
sudo docker run --name "standby_name" \
  -e DATABASE_NAME="db_name" \n  -e PGPORT=db_listener_port \n  -e DATABASE_USER="db_user_name" \n  -e DATABASE_USER_PASSWORD="db_user_password" \n  -e ENTERPRISEDB_PASSWORD="enterprisedb_password" \n  -e LOCALEPARAMETER="locale" \n  -e NAMESERVER="nameserver_ipaddress" \n  -e REPL_USER="repl_user_name" \n  -e REPL_PASSWORD="repl_user_password" \n  -e MASTER_HOST "
    (\n      -e MASTER_PORT=5444 \n      -e ENABLE_HA_MODE=Yes|No \n      -e ENABLE_ARCHIVE_MODE=Yes|No \n      -v host_data_volume:/edbvolume \n      -v host_archive_volume:/edbarchive \n      -p host_port:db_listener_port \n      -e ACCEPT_EULA=Yes|No \n      -d containers.enterprisedb.com/edb/edb-as:v12
    )" 
```

Include the `--name` option to specify the name of the master node of the cluster. Include the `-e` option with the docker run command to specify values for environment variables used to configure your container. Include the following values when deploying a master node that includes Advanced Server and Failover Manager:

- **DATABASE_NAME**: environment variable to specify the name of the Advanced Server database.
- **PGPORT**: environment variable to specify the listener port of the Advanced Server database (by default, 5444).
- **DATABASE_USER**: environment variable to specify the name of a database superuser that will be created when the database is initialized; by default, the database superuser is named enterprisedb.

If you specify the default (enterprisedb), the user will be associated with the password provided by the **ENTERPRISEDB_PASSWORD** environment variable.

- **DATABASE_USER_PASSWORD**: environment variable to specify the password associated with the database superuser if you specify a db_user_name other than enterprisedb. Please note that this password should not be changed after the pod is initialized.

- **ENTERPRISEDB_PASSWORD**: environment variable to specify the password associated with the default database superuser (enterprisedb). During the installation process, the container creates a database superuser named enterprisedb. Please note that this password should not be changed after the pod is initialized.

- **LOCALEPARAMETER**: environment variable to specify the locale that will be used by the container.

- **INITDBOPTS**: parameter to provide non-default options for the initdb operation, e.g., this parameter can be used to disable Oracle compatibility.

- **NAME SERVER**: environment variable to specify the identity of a name server that will be used for notifications from Failover Manager.

- **REPL_USER**: environment variable to specify the name of the Postgres streaming replication user.

- **REPL_PASSWORD**: environment variable to specify the password associated with the replication user.
Use the `MASTER_HOST` environment variable to specify the ip address of the master node of the replication cluster. The ip address of the master node can be retrieved using the docker inspect command the cluster based on the name of the master node as shown below:

```bash
sudo docker inspect --format '{{ .NetworkSettings.IPAddress }}' master_name
```

Use the `STANDY_HOSTS` environment variable to specify a comma-separated list of ip addresses of the standby nodes in the replication cluster.

Use the `ENABLE_HA_MODE` environment variable to specify whether failover should be enabled.

Use the `ENABLE_ARCHIVE_MODE` environment variable to specify whether archiving should be enabled. If `ENABLE_HA_MODE` is set to Yes, then `ENABLE_ARCHIVE_MODE` must be set to Yes also.

Map the mount point of the data volume of the container, `/edbvolume`, to a directory on the host system using the `-v` option; for example, the following option:

```bash
-v /data/edbas:/edbvolume
```

If `ENABLE_ARCHIVE_MODE` is set to Yes, then map the mount point of the archive volume of the container, `/edbarcive`, to a directory on the host system using the `-v` option; for example, the following option:

```bash
-v /archive/edbas:/edbarcive
```

Use the `ACCEPT_EULA` environment variable to indicate if you accept or decline the EnterpriseDB license. The license is available for review at:

https://www.enterprisedb.com/node/4509

Note: You must accept the license agreement to deploy a container.

Include the `-p` option to enable port forwarding from the container to the host machine; for example, `-p 5445:5444` makes the listener port 5444 in the container available on port 5445 on the host system.

Include the `-d` option to indicate that the container should be deployed in detached mode; this enables background processing for the deployment.

After providing values for the environment variable and including any docker options, specify the repository name and image you wish to use.

Replace `master_name` with the name of the master node of the cluster.
4.4.1.3 Deploying pgPool in a Container

pgPool is a connection pooler for Advanced Server. When deploying a pgPool container, include the following syntax:

```
docker run --name "container_name" \
   -e DATABASE_NAME="db_name" \
   -e PGPORt=db_listener_port \
   -e ENTERPRISEDB_PASSWORD="enterprisedb_password" \
   -e REPL_USER="repl_user_name" \
   -e REPL_PASSWORD="repl_user_password " \
   -e MASTER_HOST=master_ipaddress \
   -e STANDBY_HOSTS=standby_ipaddress_list \
   -e ACCEPT_EULA=Yes|No \
   -d containers.enterprisedb.com/edb/edb-pgpool:v4.0
```

Include the --name option to specify the name of the container in which pgPool resides.

Use the -e option with the docker run command to specify values for environment variables used to configure your container. Include the following values when deploying a master node that includes Advanced Server and Failover Manager:

- **DATABASE_NAME** environment variable to specify the name of the Advanced Server database.
- **PGPORT** environment variable to specify the listener port of the Advanced Server database (by default, 5444).
- **ENTERPRISEDB_PASSWORD** environment variable to specify the password associated with the default database superuser (enterprisedb). During the installation process, the container creates a database superuser named enterprisedb. Please note that this password should not be changed after the pod is initialized.
- **REPL_USER** environment variable to specify the name of the Postgres streaming replication user.
- **REPL_PASSWORD** environment variable to specify the password associated with the replication user.
- **MASTER_HOST** environment variable to identify the master node of the replication cluster. The clause uses a call to docker inspect to retrieve the address of the master node of the cluster:

  ```
  MASTER_HOST="`sudo docker inspect --format '{{ .NetworkSettings.IPAddress }}' master_name`".
  ```

- **STANDBY_HOSTS** environment variable to specify a comma-separated list of ip addresses of the standby nodes in the replication cluster.
- **ACCEPT_EULA** environment variable to indicate if you accept or decline the EnterpriseDB license. The license is available for review at:

  https://www.enterprisedb.com/node/4509

Note: You must accept the license agreement to deploy a container.

Include the -d option to indicate that the container should be deployed in detached mode; this enables background processing for the deployment.

After providing values for the environment variable and including any docker options, specify the repository name and image you wish to use.
4.4.1.4 Deploying BART in a Container

EDB Postgres Backup and Recovery Tool (BART) simplifies backup management for Advanced Server databases. When deploying BART in a container, include the following syntax:

```bash
docker run \
    --name "container_name" \
    -e DATABASE_NAME=db_name \
    -e BART_HOST_ADDRESS=host_address \
    -e PGHOST=`sudo docker inspect --format '{{.NetworkSettings.IPAddress }}' master_name` \
    -e PGPORT=db_listener_port \
    -e REPL_USER=repl_user_name \
    -e REPL_PASSWORD=repl_user_password \
    -e BART_AUTOMATE_BACKUP=Yes|No \
    -e BART_NUM_BACKUPS_TO_KEEP=2 \
    -e BART_COMPRESS_BACKUP=Yes|No \
    -e ACCEPT_EULA=Yes \
    -v hostvolume:/edbbackup \
    -d containers.enterprisedb.com/edb/edb-bart:v2.5
```

Include the `--name` option to specify the name of the container in which BART will reside.

Include the `-e` option with the `docker run` command to specify values for environment variables used to configure your container. Include the following values when deploying a master node that includes Advanced Server and Failover Manager:

- **DATABASE_NAME**: Specify the name of the Advanced Server database that will be backed up.
- **BART_HOST_ADDRESS**: Provide the IP address or location of the BART host.
- **PGHOST**: Specify the IP address of the master node of the replication cluster.
- **PGPORT**: Specify the listener port of the Advanced Server database (by default, 5444).
- **REPL_USER**: Specify the name of the Postgres streaming replication user.
- **REPL_PASSWORD**: Specify the password associated with the replication user.
- **BART_AUTOMATE_BACKUP**: Specify if BART should automate the backup process. If set to Yes, use the `BART_BACKUP_SCHEDULE` environment variable to specify the schedule for the backup process in cron format.
- **BART_NUM_BACKUPS_TO_KEEP**: Specify the number of backups retained by BART.
- **BART_COMPRESS_BACKUP**: Specify if BART should compress the backup. Specify yes to indicate that backups should be compressed, or no to indicate that they should not be compressed.

4.4. Deploying from the Docker Command Line
Use the `ACCEPT_EULA` environment variable to indicate if you accept or decline the EnterpriseDB license. The license is available for review at:

https://www.enterprisedb.com/node/4509

Note: You must accept the license agreement to deploy a container.

Map the mount point of the backup volume of the container, `/edbbackup`, to a directory on the host system using the `-v` option; for example, the following option:

```
-v /backup/edbas:/edbbackup
```

maps the host directory `/backup/edbas` to `/edbbackup` inside the container.

Include the `-d` option to indicate that the container should be deployed in detached mode; this enables background processing for the deployment.

After providing values for the environment variable and including any docker options, specify the repository name and image you wish to use.
CHAPTER 5

Using the OpenShift Console

After creating a project and any required templates, you can use the OpenShift console to create and manage Advanced Server projects.

Use a browser to access the OpenShift web console, and log on using your OpenShift credentials. When you have successfully authenticated, the console displays the My Projects page.

Select your project (for example, my-edbas-project) from the Projects list; the OpenShift console will navigate to the project management page.
Click the **Add to Project** button to open the **Select Image or Template** page.

Click the **Advanced Server template**. The OpenShift console opens a page that allows you to specify details for your Advanced Server deployment.
Enter the installation details for the deployment in the fields displayed under the Parameters section. The details provided are used during pod initialization; it is important to note that password changes are not allowed after a pod has been initialized.

- In the **Database Name** field, provide the name of the database that is created when the cluster is initialized.
- In the **Default database user** field, enter the name of a database superuser that is created when the database is initialized; by default, the database superuser is named `enterprisedb`.
  - If you accept the default (`enterprisedb`), the user will be associated with the password provided by an OpenShift Secret created for the `enterprisedb` password.
  - If you specify an alternate name for the database superuser, the user will be associated with the password provided via an OpenShift Secret created for the alternate user password. Optionally, you can provide the name of an alternate user with a ConfigMap.
- In the **Repl user** field, enter the name of the replication user; the default name is `repl`. Optionally, you can use a ConfigMap to provide the name of a replication user.
  - The replication user will be associated with a password provided in an OpenShift Secret created for the replication user password.
- In the **Database Port** field, enter the port that the database will monitor for connections.
- In the **Locale** field, enter the locale used by the cluster; by default, the locale is the system locale.
- In the **Email** field, enter the email address that will receive any notifications sent by Failover Manager.
- In the **Name Server for Email** parameter, provide the identity of a name server that will be used for notifications from Failover Manager.
- Use the **Enable Monitoring** parameter to indicate whether monitoring should be enabled; valid options are Yes or No (default).
- Use the **Enable High Availability** parameter to indicate whether high availability mode should be enabled; valid options are Yes (default) or No.
• Use the Initdb Options parameter to provide non-default options for the initdb operation, e.g., this parameter can be used to disable Oracle compatibility.

• Use the Reuse Data Volume parameter to indicate if pre-existing data found in the data volume during database startup should be reused; valid options are Yes (default) or No.

• Use the Enable Archive Mode parameter to indicate if archiving should be turned on for the database; valid options are No (default) or Yes.

• In the Persistent Volume field, enter the name of the persistent volume definition file.

• In the Persistent Volume Claim field, enter the name of the persistent volume claim definition file.

• In the Backup Volume field, provide the name of the volume claim for BART.

• In the Backup Volume Claim field, provide the name of the persistent volume claim for BART.

• In the BART Host Address field, provide the address of the BART server host.

• In the Database Host field, provide the name of the database server that BART is backing up.

• In the Automate backup operation field, indicate if backups should be taken automatically; if you specify No, the BART Backup Schedule field and the BART Number of Backups to Keep fields will not apply, and you must manually take backups and manage backup retention.

• In the BART Backup Schedule field, provide the cron formatted schedule on which backups will be performed.

• In the BART Number of Backups to Keep field, enter the number of database backups that will be stored.

• In the Compress backup files field, specify if backup files should be stored in compressed format.

• In the Restore File field, enter the name and location of a backup file that will be restored during the deployment.

• In the Restore Directory field, enter the location of the directory to be used when performing point-in-time recovery (PITR).

• In the PEM Server field, enter the address of the PEM server.

• In the PEM Server Port field, enter the port number of the PEM database server.

• In the PEM user field, enter the name of the PEM user for accessing the PEM server.

• Enter a value in the Accept end-user license agreement field; valid options are Yes or No.

• Use the Labels section of the Parameters dialog to specify one or more tags that will be applied to the container. For more information about using labels, see the OpenShift documentation.

When you have completed the Parameters dialog, click the Create button to deploy an Advanced Server project.
Click **Close** to open the deployment status screen. In the example, four Advanced Server pods, two pgPool pods, and one BART pod are being deployed.

As OpenShift spins up the pods, the progress indicator will change from light blue to darker blue.
When the progress indicators are solid blue for all the pods, the pods are ready to accept client requests and the cluster is ready for use.

You can review a listing of all the pods in the cluster on the Applications dialog.

You can also use the OpenShift CLI to review a list of pods, along with their respective roles:

```
oc get pods -o wide -L role -L cluster-l cluster=<cluster_name>
```

Where:

- `cluster_name` specifies the name of the cluster.

The following screen shows the pod listing for a cluster named edb:
To check the log files for a pod, use the command:

```bash
oc rsh <pod_name> cat startup.log
```

Where `pod_name` specifies the name of a specific pod.
5.1 Scaling an Advanced Server Deployment

The default configuration of EDB Postgres Advanced Server for OpenShift uses EDB Postgres Failover Manager to ensure high-availability for your deployment. If a pod fails, Failover Manager detects the failure, and replaces the pod with a running node. If the failed node is the master in your replication scenario, Failover Manager promotes a standby node to the role of master before adding a replacement standby to the scenario.

To prevent disruptions in Failover Manager monitoring, an Advanced Server deployment must have at least four pods; by default, each new Advanced Server project will have four pods.

Please note: by default, the container environment will support up to 9 pods; to support 10 or more pods, you must modify the server configuration.

Manually Scaling a Pod

You can use the up arrow (to the right of the blue circle) to add new pods to your deployment when processing requirements are higher, or use the down arrow to remove unneeded pods from the deployment when processing requirements are light. Please note that when removing a pod from your deployment, OpenShift may remove the master node in your replication scenario. If Failover Manager is enabled, and the master node is removed during scaling, a standby node will be promoted to the role of master.

If you plan to remove multiple pods from a deployment, you should allow time for each pod to be completely removed before removing each additional pod to avoid interfering with Failover Manager protection.
5.2 Using a ConfigMap Object to Customize a Configuration

You can use an OpenShift ConfigMap object to create a template that specifies a custom configuration, executes SQL commands, or specifies the name of the database superuser.

- When specifying a custom parameter value for the `postgresql.conf` file, provide the modified `parameter=value` pair(s) in a file named `postgresql.conf.in`. Contents of the `postgresql.conf.in` file are appended to the current `postgresql.conf` file.

- When specifying a custom entry for the `pg_hba.conf` file, provide the entry in a file named `pg_hba.conf.in`. Contents of the `pg_hba.conf.in` file are appended to the current `pg_hba.conf` file.

- When using a ConfigMap to execute SQL commands at deployment, provide the SQL commands in a file named `dbload`.

After creating custom files, copy the files into a directory that can be accessed when deploying the container. Then, create a ConfigMap with the command:

```bash
oc create configmap <map_name> --from-file=<directory_name>
```

Where:

- `map_name` is the name of the ConfigMap.
- `directory_name` specifies the directory that contains the custom configuration files.

After creating the ConfigMap, you must modify the template file, specifying the location of the files in the volumes section:

```yaml
- name: <mount_name>
  configMap:
    name: <map_name>
```

You must also modify the `volumeMounts` section:

```yaml
volumeMounts:
  - name: <mount_name>
    mountPath: /<target_directory>
```

Where:

- `mount_name` is the name used within the template file to link the ConfigMap name with the target directory on the deployment.
- `map_name` is the name of the ConfigMap.
- `target_directory` is the name of the directory on the deployment to which the files will be copied.

Any containers deployed using the template will use the customized configuration files.

To create a ConfigMap that specifies the (non-default) name of the database superuser or replication user, include the `--from-literal` type, followed by the associated keyword, and the name of the user.

- When specifying the name of a database superuser, include the `database.user` keywords.
- When specifying the name of a replication user, include the `repl.user` keywords.

Use the following command to create the ConfigMap:

```bash
oc create configmap --from-literal=<user_type>=<user_name>
```

Where:
user_type is either database.user or repl.user.

user_name is the name of the user.

You do not need to update the template file when using a ConfigMap to specify a non-default user name.

For detailed information about customizing the postgresql.conf file, please refer to the PostgreSQL core documentation.

For detailed information about customizing the pg_hba.conf file, please refer to the PostgreSQL core documentation.

For more information about using an OpenShift ConfigMap object, please see the OpenShift documentation.
5.3 Performing a Rolling Update

When an updated version of Advanced Server becomes available, you can use a rolling update technique to upgrade your cluster. EnterpriseDB’s Docker repository will always make available the most recent version of the server; to update the server version used in your deployment, you can simply:

1. When an updated version of Advanced Server becomes available, use the OpenShift console to add new pods to the cluster; as each new pod is added, the new pod will use the updated server version.

2. Remove pods that were instantiated using the old server version (including the original master node of the replication scenario).

**Note:** To preserve the integrity of your Failover Manager scenario, you should not let the total pod count of the deployment drop below four when performing a rolling update.
5.4 Performing a Manual Backup

To perform a manual backup, use the OpenShift command line to identify the BART container:

```
oc get pods -o wide -L role -L cluster -l role=backuptool -l cluster=<cluster_name>
```

Then, use the OpenShift command line to log in to the BART container

```
oc rsh <pod_name>
```

Where `pod_name` is the value displayed for the backuptool in the list of pods.

Then, use BART to perform a backup:

```
bart backup -s <cluster_name> -Ft -z
```

Where:

- `cluster_name` specifies the name of the cluster.

For example:

```
bart backup -s edb -Ft -z
```

INFO: creating backup for server 'edb'
INFO: backup identifier: '1525640148300'
65043/65043 kB (100%), 1/1 tablespace

INFO: backup completed successfully
INFO: backup checksum: 86bd2596f9bd69f4ca1f96555ff98f0 of base.tar.gz
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1525640148300
BACKUP NAME: none
BACKUP PARENT: none
BACKUP LOCATION: /edbbackup/edb-bart-1-j1czk/pgbackup/edb/1525640148300
BACKUP SIZE: 6.73 MB
BACKUP FORMAT: tar.gz
BACKUP TIMEZONE: UTC
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
ChkSum File
86bd2596f9bd69f4ca1f96555ff98f0 base.tar.gz
TABLESPACE(s): 0
START WAL LOCATION: 0000000010000000000000000000000D
BACKUP METHOD: streamed
BACKUP FROM: master
STOP TIME: 2018-05-06 20:55:50 UTC
TOTAL DURATION: 2 sec(s)

BART creates the fully qualified filename by concatenating the BACKUP LOCATION and File fields. You can use the filename to restore the cluster if needed. In the example above, the fully qualified filename is:

```
/edbbackup/edb-bart-1-j1czk/pgbackup/edb/1525640148300/base.tar.gz
```
5.4.1 Restoring a Cluster from Backup

To restore the cluster, use the Add to Project button to deploy a new cluster; when prompted, provide the name of the backup file in the Restore File field on the Parameters dialog for the deployment.

![Restore File](image)

Fig. 5.11: Restoring a cluster from backup

5.4.2 Performing a Point-in-Time Recovery (PITR)

To perform a point-in-time recovery (PITR) from backup, complete the following steps:

1. Run the OpenShift command line to identify the BART container:
   ```
   oc get pods -o wide -L role -L cluster -l role=backuptool -l cluster=<cluster_name>
   ```
2. Use the OpenShift command line to log in to the BART container:
   ```
   oc rsh <pod_name>
   ```
   where `pod_name` is the value displayed for the backuptool in the list of pods.
3. Use BART to list backups:
   ```
   bart show-backups -s <cluster_name>
   ```
   where `cluster_name` specifies the name of the cluster.
4. Select the desired backup from the list, and restore pgdata into restore directory up to desired timestamp:
   ```
   bart restore -s cluster_name -i backupid -p /edbbackup/project/restore/edb1/pgdata -g 'timestamp'
   ```
5. Then, restore the cluster by setting RESTORE_DIR to the restored pgdata directory in the yaml file, and deploy.
5.5 Preparing an Advanced Server Cluster for Monitoring

To monitor an Advanced Server cluster with PEM, you need to:

1. Start (deploy) the PEM server container.
2. Launch the PEM agent on the Advanced Server cluster.

5.5.1 Deploying a PEM Server

Once the PEM server is deployed, you can login to access the admin console via any browser at https://<your-server-ip>:<pem-port>/pem/login.

For more information, see the Monitoring section of the sample user scenarios on the EDB website for instructions on how to deploy the PEM server.

5.5.2 Deploying a PEM Server using a Helm Chart

Download the following sample files:

- Sample helm chart: edb-pemserver-7.12.0.tgz
- Sample values.yaml file: sample-values.yaml

Create your pemserver-values.yaml file with customized input values using the sample values.yaml file as a template (e.g. my-pemserver-values.yaml).

Enter the following command to deploy the containers with your custom inputs in your namespace:

```
helm install <helm-chart-name> edb-pemserver-7.12.0.tgz -f <pemserver-values.yaml> --namespace <namespace>
```
Note: In the example, myvalues.yaml is the user supplied values.yaml file containing the desired parameter values.

5.5.3 Launching a PEM Agent on Advanced Server Cluster

Once the PEM server is deployed, to launch the PEM agent on the Advanced Server cluster, you must:

Enable monitoring on the Advanced Server cluster at startup by setting the parameter `ENABLE_MONITOR_MODE` to the value `Yes`.
5.6 Removing a Project

Use the following set of commands to remove a project:

$$\text{oc delete project } <\text{project-name}>$$

For example, to remove a project named epas-10:

$$\text{oc delete project epas-10}$$

5.6.1 Retaining a Project with No Pods

You can delete all pods in the project but retain the project for later use; use the commands:

$$\text{oc delete dc/database_name-as10}$$
$$\text{dc/database_name-pgpool}$$
$$\text{dc/database_name-bart}$$
$$\text{oc delete svc database_name-service}$$

You can reuse the Secret and ConfigMap objects created earlier.
5.7 Upgrading an Advanced Server Container

You can use the migration tool to upgrade an Advanced Server version 9.6 Container to Advanced Server version 10. The following steps detail the upgrade process:

1. Download the migration tool:
   
   `oc new-app containers.enterprisedb.com/edb/edb-admintool`

2. Open a command line to the Advanced Server container:
   
   `oc rsh <container_name>`

3. Inside the Advanced Server container, use pg_dumpall to take a logical backup. When the command completes, verify the location of the dump file, and exit back to the command line.
   
   ```
   mkdir -p /edbvolume/$DATABASE_NAME/$HOSTNAME/pgdump
   pg_dumpall -c -f /edbvolume/$DATABASE_NAME/$HOSTNAME/pgdump/${DATABASE_NAME}.dumpall.sql
   ls -l /edbvolume/$DATABASE_NAME/$HOSTNAME/pgdump/${DATABASE_NAME}.dumpall.sql
   exit
   ```

4. Run the migration tool in preview mode. This is an optional step that will not perform an actual migration, but will generate a sample of the yaml file to be used to deploy the upgraded Advanced Server container.
   
   ```
   oc rsh `oc get pod -o name -l role=admintool` migratedb
   --database=<database_name>
   --version=10
   --restore-file=<dumpall_file_name>
   --preview > <my_deployment_preview.yaml>
   ```

   Where:
   
   - `database_name` is the name of the Advanced Server database.
   - `dumpall_file_name` is the name of the pg_dumpall file created in Step 3.
   - `my_deployment_preview.yaml` specifies the name of the yaml file generated when invoking the migration tool in preview mode.

5. Run the migration tool. This step will delete the old Advanced Server cluster, deploy the upgraded Advanced Server cluster and restore data from the old cluster into the new cluster.
   
   ```
   oc rsh `oc get pod -o name -l role=admintool` migratedb
   --database=<database_name>
   --version=10
   --restore-file= <dumpall_file_name> | oc create -f -
   ```

   Where:
   
   - `database_name` is the name of the Advanced Server database.
dumpall_file_name is the name of the pg_dumpall file created in Step 3.
For detailed information about using pg_dumpall, see the PostgreSQL online documentation.
6.1 Uploading Containers to Network-Isolated Environments

If you want to deploy the containers in a network-isolated environment, perform the following steps:

1. Using a browser, download EDB container image as tarballs (.tgz file) from the EDB website.
2. Internally transfer the image tarball to the isolated environment
3. Load the image in the isolated environment
   
   ```
   docker load < image.tgz
   ```

For more information, refer to the official docker documentation.
To access the database remotely, expose pgPool via an external service:

```
oc expose dc edb1-pgpool --port=5444 --target-port=9999 --name=edb1-ext-service --type=LoadBalancer
```

Where `edb1` is the name the database and `5444` is the port this example.

Once an external IP address is allocated for the server, you can check obtain it via the following command:

```
oc get service edb1-ext-service
```

The database cluster (`edb1`) will now be accessible remotely using the external ip address and port.
6.3 Sample Scripts

You can download the scripts and related yaml files for the examples shown in the guide.
Conclusion

EDB™ Postgres Containers and Integration with OpenShift
Copyright © 2007 - 2020 EnterpriseDB Corporation. All rights reserved.
EnterpriseDB® Corporation 34 Crosby Drive, Suite 201, Bedford, MA 01730, USA
T +1 781 357 3390 F +1 978 467 1307 E info@enterprisedb.com www.enterprisedb.com

• EnterpriseDB and Postgres Enterprise Manager are registered trademarks of EnterpriseDB Corporation. EDB
  and EDB Postgres are trademarks of EnterpriseDB Corporation. Oracle is a registered trademark of Oracle, Inc.
  Other trademarks may be trademarks of their respective owners.

• EDB designs, establishes coding best practices, reviews, and verifies input validation for the logon UI for EDB
  products where present. EDB follows the same approach for additional input components, however the nature
  of the product may require that it accepts freeform SQL, WMI or other strings to be entered and submitted
  by trusted users for which limited validation is possible. In such cases it is not possible to prevent users from
  entering incorrect or otherwise dangerous inputs.

• EDB reserves the right to add features to products that accept freeform SQL, WMI or other potentially dangerous
  inputs from authenticated, trusted users in the future, but will ensure all such features are designed and tested to
  ensure they provide the minimum possible risk, and where possible, require superuser or equivalent privileges.

• EDB does not warrant that we can or will anticipate all potential threats and therefore our process cannot fully
  guarantee that all potential vulnerabilities have been addressed or considered.
Index

C
Conclusion, 43
Creating a Container at the Docker Command Line, 14
Creating a Local Repository, 6

D
Deploying a Container, 7
Deploying a PEM Server, 35
Deploying a PEM Server using a Helm Chart, 35
Deploying Advanced Server in a Container (Master Node), 15
Deploying Advanced Server in a Container (Standby Node), 17
Deploying BART in a Container, 20
Deploying Containers using a Helm Chart, 7
Deploying Containers using Operators, 7
Deploying Containers using Templates, 10
Deploying from the Docker Command Line, 14
Deploying pgPool in a Container, 19
Deployment Prerequisites, 6

E
EDB Containers, 3

L
Launching a PEM Agent on Advanced Server Cluster, 36

P
Performing a Manual Backup, 33
Performing a Rolling Update, 32
Preparing an Advanced Server Cluster for Monitoring, 35
Preparing to Deploy an Advanced Server Container, 5

R
Reference, 40
Removing a Project, 37
Retaining a Project with No Pods, 37

S
Scaling an Advanced Server Deployment, 29

U
Upgrading an Advanced Server Container, 38
Using a ConfigMap Object to Customize a Configuration, 30
Using the OpenShift Console, 22

W
What’s New, 2