EDB™ Postgres on Kubernetes

Release 2.7

Operator User's Guide

Feb 11, 2020

Contents

This document contains information on how to install and use the EDB Postgres Operator for deploying EDB containers in the Google Kubernetes Engine (GKE) environment. When implementing the Operator functionality, following components are used:

- **EPAS Operator container**:

  ```
  containers.enterprisedb.com/edb/epas-operator:  v1.0
  ```

- **EPAS Operator API Server container**:

  ```
  containers.enterprisedb.com/edb/epas-apiserver:v1.0
  ```

- **EPAS Operator CLI Tool (Linux)**:

  ```
  epasctl
  ```

The Operator is installed as a single pod that includes the Operator and API Server containers. Once the Operator is installed, end users can use the Operator CLI tool (epasctl) to deploy and manage EDB containers in their respective namespaces.

Two user roles are defined for deploying EDB Containers using Operators:

- **Administrator**: Responsible for installing the Operator with the appropriate system resources.
- **End User**: Uses the functions provided by Operator via the CLI tool.

CHAPTER 1

What's New

The following changes are added to EDB™ Postgres Containers Operator guide to create version 2.7:

- Starting this release, the containers can be deployed in OpenShift using the Operator.

CHAPTER 2

Installing the Advanced Server Operator on GKE and OpenShift

Before installing the Operator, download the latest Helm client version.

To install the Operator, complete the following steps as a user with administrative rights

1. Download the following files:
   - **Sample helm chart:** epas-operator-1.0.2.tgz
   - **Sample values.yaml file:** sample-operator-values.yaml
2. Configure the Operator as required by creating an `operator-values.yaml` file, e.g. `my-operator-values.yaml` based on the `sample-operator-values.yaml` file provided. For more information about configuring an Operator, see *Using Advanced Server Operator*.
3. Complete the required prerequisites based on your requirements as specified in the custom operator values yaml file, e.g., *my-operator-values.yaml*. For more information on prerequisites, refer to the relevant sections of the configuration file in *Using Advanced Server Operator*.
4. - *(OpenShift only)* Create a new Openshift project for your Operator Deployment

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

   - *(GKE only)* Create a new Kubernetes namespace on GKE for your Operator Deployment

     ```
     kubectl create namespace <my-operator-namespace>
     ```

5. Once the operator values yaml file is updated and the necessary prerequisites are met, install the operator in a namespace of your choice with your custom configuration using the helm chart provided (Helm3 commands provided below):

   ```
   helm install <helm-deployment-name> epas-operator-1.0.2.tgz \    -f
   <my-operator-values.yaml> \ --namespace <my-operator-namespace>
   ```

6. After the Operator is successfully installed, verify that the Operator service is available by running the command:

   ```
   kubectl get svc epas-operator -n <my-operator-namespace>
   ```

   Note: The `EXTERNAL-IP` and `PORT` are generated after running the Operator service command:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
epas-operator	NodePort	x.x.x.x	<external-ip>	<port> :xxxxx/TCP	10s

7. After running the Operator service command, the API Server URL for the CLI tool, (`epasctl`), would be:

```
https://<external-ip>:<port>
```

Note: The end users will need this URL in order to use the CLI tool, `epasctl`, as mentioned in *Installing the Advanced Server Operator CLI (epasctl)*.

CHAPTER 3

Configuring the Advanced Server Operator

Before installing the Operator, you must set the required prerequisites and settings in the configuration file (`operator values yaml`). Refer to the sample configuration file `sample-operator-values.yaml` for examples of the values used in the configuration file.

The following section provides examples of settings and prerequisites required for the configuration file:

Header Sections

Section	Description
licensing	Verifies that you accept the End-User License Agreement.
image	Stores details related to the EDB container images, such as container registry URI, registry credentials, image pull policy, and registry secret name.
config	Stores configuration details of the different components, such as database, queryrouter, backup, archiving, monitoring, persistence, and resource specifications.

Licensing section

Prerequisite: The user must set the value of `acceptEULA` to `Yes` to use the Operator.

Parameter(s)	Possible Values	Description
acceptEULA	Yes, No	Indicates if the user accepts the End-User License Agreement.

Image section

Prerequisite: The user must obtain the `EDB registry username and password` to use the Operator.

Parameter(s)	Possible Values	Description
credentials/	containers.enterprisedb.com	Credentials required to access the EDB container registry:
registry	*Provided by EDB*	EDB container registry URI
username	*Provided by EDB*	EDB registry username
password		EDB registry password

Continued on next page

Table 3.2 – continued from previous page

Parameter(s)	Possible Values	Description
epository	edb	
pullPolicy	Always IfNotPresent Never	For more detail, refer to https://kubernetes.io/docs/concepts/configuration/overview/
pullSecret	edb-regsecret	Name of the registry secret used for downloading container images from the EDB registry by Kubernetes. The secret is automatically created upon installation based on the credentials provided earlier.

Config (header) section

Prerequisite: The default `serviceaccount` of an Operator namespace must be granted `cluster-admin` role.

For example, enter the following command:

```
kubectl create clusterrolebinding <operator-namespace>-cluster-admin-binding
--clusterrole=cluster-admin --serviceaccount=<operator-namespace>:default
```

Parameter(s)	Possible Values	Description
version	1.0	Initial version of Advanced Server Operator
namespace	User-specified	Namespace where the operator will be deployed
userNamespaces	Comma-separated user-specified list	The namespace(s) where the end-users will deploy Advanced Server clusters

Config (service) section

Prerequisite: If a service type of `NodePort` is used, the allowable port range for NodePort services (`30000-32767`) must be open on the kubernetes cluster.

Parameter(s)	Possible Values	Description
type	NodePort, LoadBalancer	The type of Kubernetes service to be used for accessing the Operator
port	User-specified	The port of the Kubernetes service

Config (database) section

Prerequisite: If `antiAffinity` is set to `true`, the number of pods in an Advanced Server cluster cannot exceed the number of nodes in the kubernetes cluster.

Parameter(s)	Possible Values	Description
antiAffinity	true, false	Indicates if anti-affinity rule should be applied between pods of the same Advanced Server cluster. For more detail, refer to: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity

Config (database/services) section

Prerequisite: If a service type of `NodePort` is used, the allowable port range for NodePort services (`30000-32767`) must be open on the Kubernetes cluster.

Parameter(s)	Possible Values	Description
master/	NodePort, LoadBalancer	The type of Kubernetes service to be used for accessing the master pod.
type	User-specified	The port of the Kubernetes service.
port		
standby/	NodePort, LoadBalancer	The type of Kubernetes service to be used for accessing the standby pods.
type	User-specified	The port of the Kubernetes service.
port		

Config (database/probes) section

Parameter(s)	Possible Values	Description
readiness/	true, false	Indicates whether readiness probes will be used.
enabled	User-specified	For details on using readiness probes, refer to the Kubernetes documentation at: https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/
initialDelaySeconds		
periodSeconds		
timeoutSeconds		
failureThreshold		
successThreshold		
liveness/	true, false	Indicates whether liveness probes will be used.
enabled	User-specified	For details on using liveness probes, refer to the Kubernetes documentation at: https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/
initialDelaySeconds		
periodSeconds		
timeoutSeconds		
failureThreshold		
successThreshold		

Config (queryrouter) section

Prerequisite: If antiAffinity is set to `true`, the number of PgPool replicas supporting the same Advanced Server cluster cannot exceed the number of nodes in the Kubernetes cluster.

Parameter(s)	Possible Values	Description
enabled	true, false	Indicates if query routing will be used (via PgPool)
image	edb-pgpool	Name of the PgPool container image
imageTag	v4.0	Version of PgPool running in the container
replicas	User-specified	Number of replicas of the PgPool container to deploy
witnessNode	true, false	Indicates the PgPool containers should be used as witness nodes for EDB Failover Manager
antiAffinity	true, false	Indicates whether anti-affinity rule should be applied between replicas of the PgPool container. For more detail, refer to: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity

Config (queryrouter/service) section

Prerequisite: If a service type of `NodePort` is used, the allowable port range for `NodePort services (``30000-32767`) must be open on the kubernetes cluster.

Parameter(s)	Possible Values	Description
type	NodePort, LoadBalancer	The type of kubernetes service tp be used with PgPool
port	User-specified	The port of the kubernetes service

Config (queryrouter/probes) Section

For details, see the *Config (database/probes) Section* table.

Config (backup) section

Parameter(s)	Possible Values	Description
enabled	true, false	Indicates whether backup will be used
image	edb-bart	Name of the BART container image
imageTag	v2.5	Version of BART running in the container
hostAddress	localhost	Host of the BART server
persistentVolume/	User-specified	Capacity and access mode of the persistent volume to be used for backups. For more detail, refer to:
capacity	ReadWriteMany	https://kubernetes.io/docs/concepts/storage/persistent-volumes/
accessModes	User-specified	Specification of the NFS volume to be used for backup
nfsServer/		
path		
ipAddress		
persistentVolumeClaim/	User-specified	Size of the persistent volume claim to be used for backups. For more detail, refer to:
size		https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Config (archive) section

Parameter(s)	Possible Values	Description
enabled	true, false	Indicates whether database archiving will be used
persistentVolume/	User-specified	Capacity and access mode of the persistent volume to be used for backups. For more detail, refer to:
capacity	ReadWriteMany	https://kubernetes.io/docs/concepts/storage/persistent-volumes/
accessModes	User-specified	Specification of the NFS volume to be used for backup
nfsServer/		
path		
ipAddress		
persistentVolumeClaim/	User-specified	Size of the persistent volume claim to be used for backups. For more detail, refer to:

Continued on next page

Table 3.10 – continued from previous page

Parameter(s)	Possible Values	Description
size		https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Config (monitor) section

Prerequisite: The port range for NodePort services (`30000-32767`) must be open on the kubernetes cluster.

Parameter(s)	Possible Values	Description
enabled	true, false	Indicates if monitoring will be used
serverImage	edb-pemserver	Name of the PEM server container image
serverImageTag	v7.12	Version of PEM server running in the container
cidrAddress	0.0.0.0/0	CIDR address
httpPorts	30000-32767, 30000-32767	Comma-separated list of http ports of the PEM servers
httpsPorts	30000-32767, 30000-32767	Comma-separated list of https ports of the PEM servers
dbImage	edb-as	Name of the Advanced Server container image used as the backing database for PEM
dbImageTag	v11	Version of the Advanced Server container image used as the backing database for PEM
pgport	User-specified	Database port of backing database for PEM
user	User-specified	Database user of backing database for PEM
password	User-specified	Database password of backing database for PEM
enterprisedb-Password	User-specified	Database password for the default user (enterprisedb) of backing database for PEM

Config (persistence) section

Prerequisite: The `storageClass` specified must be available if persistence is enabled. Refer to the Kubernetes documentation at the link provided above for details on how to create storage classes for your specific platform.

If dynamic provisioning is NOT selected, i.e. set to false, then persistent volumes will have to be created beforehand for *each replica* and have to be labeled with the respective namespace and cluster names:

```
kubectl label <my-persistent-vol> namespace=<my-namespace>
cluster=<my-cluster>
```

Parameter(s)	Possible Values	Description
enabled	true, false	Indicates whether data will be persisted
dynamic	true, false	Indicates whether dynamic provisioning will be used to create persistent volumes.
storageClass	User-specified	Storage class to be used for the persistent volume. For more detail, refer to: https://kubernetes.io/docs/concepts/storage/storage-classes/
accessModes	ReadWriteOnce	Access mode of the persistent volume
size	User-specified	Size of the persistent volume claim to be used for data. For more detail, refer to: https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Config (resources) section

Parameter(s)	Possible Values	Description
limits/		
cpu	User-Specified	Specification for resource limits. For more detail, refer to:
memory	User-Specified	https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/
requests/	User-Specified	Specification for resource requests. For more detail, refer to:
cpu memory	User-Specified	https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

CHAPTER 4

Using the Advanced Server Operator

Once the Operator is installed by the administrator, end users can install the CLI tool, `epasctl`, on their own client machines and start deploying and managing EDB containers using the CLI tool.

4.1 Installing the Advanced Server Operator CLI (epasctl)

1. Install the CLI tool on client machine:

   On any Linux machine, download the CLI tarball operator-cli.tgz and expand. The tarball contains the following files:

   - `epasctl`
   - `apiserver.crt`
   - `apiserver.key`
   - `epasconfig`
   - `sample-create-epas.yaml`

2. Update the CLI configuration file (.epasconfig)

   In the `.epasconfig` file, set the following parameters:

Setting	Value	Description
apiserverUrl	https:<ipaddres>:<port>	The URL provided by the administrator after the Operator is installed as mentioned in Step 6 of Section 2.
namespace	User-specified	The namespace the user wants to use for deploying the Advanced Server cluster and related objects. The namespace has to be one of the user namespaces specified during the Operator installation as mentioned in the config(header) section of Section 2.

3. Verify the CLI installation:

   Run the following command to verify the CLI installation:

   ```
   ./epasctl version
   ```

4.2 Using the Advanced Server Operator CLI

This section provides examples on how to use the CLI tool to deploy and manage EDB containers in your designated namespace as an end-user.

4.2.1 Creating an Advanced Server Cluster

To create an Advanced Server cluster in the designated namespace, complete the following steps:

1. Create a cluster specification file - The description of the settings is provided in Cluster Specifications section. For examples, refer to the sample-create-epas.yaml file (obtained from the downloaded tarball).
2. Create the cluster by running the following command:

   ```
   ./epasctl create cluster <cluster_name> -f <cluster_spec_file>
   ```

Where `cluster_name` is the name of the cluster.

Cluster Specifications

Section	Setting	Values	Description
database	image	edb-as edb-as-lite	EPAS image to be used
	imageTag	v10, v11, v12	EPAS version
	standbyCount	User-specified	Number of standbys
	user	User-specified	Database user
	password	User-specified	Database password
	enterprisedbPassword	User-specified	Password for default user “enterprisedb”
	replUser	User-specified	Replication user
	replPassword	User-specified	Replication password
	pgport	User-specified	Database port
	localeParameter	User-specified	Locale
	nameserver	User-specified	IP address of nameserver
	enableHAMode	Yes, No	Enable high-availability (default ‘Yes’)
	enableArchiveMode	Yes, No	Enable archiving (default ‘Yes’)
	enableMonitorMode	Yes, No	Enable monitoring (PEM)
	initdbOpts	User-specified	initdb options
	reuseDataVolume	Yes, No	Reuse data if exists on volume (default ‘Yes’)
	restoreFile	User-specified	File to restore data from
	restoreDir	User-specified	Directory to restore data from
queryrouter	enabled	true, false	Enable query routing (PgPool)
backup	enabled	true, false	Enable backup
	numBackupsToKeep	User-specified	Backup retention policy
	schedule	User-specified	Backup schedule in cron format

4.2.2 Backing up an Advanced Server Cluster

You can back up an existing cluster automatically or manually.

Automatic Backup:

To automate backups, set the `schedule` in the backup section of the cluster specification file during the cluster creation as mentioned in the config section.

..note:

Automatic backups are currently available using BART only.

Manual Backup:

To perform a manual backup using BART, use the command:

```
./epasctl backup cluster <cluster_name> --method=bart
```

To perform a manual backup using pg_dump, use the command:

```
./epasctl backup cluster <cluster_name> --method=pgdump
```

4.2.3 Restoring an Advanced Server Cluster

You can restore a cluster using `BART` or `pg_restore`.The old cluster has to be deleted if exists.

Restore using BART:

To restore using BART, first list the available backups to restore from using the command:

```
./epasctl show backups <cluster_name>
```

Note the `BACKUP ID` and `TIMESTAMP` of the backups

Restoring using BART is a two-step process. In the first step, backed up data is copied into a staging area, and in the second step, that data is used to restore the cluster.

1. To restore from a specific backup identified by the backup-id, use the following commands:

   ```
   ./epasctl restore cluster <cluster-name -i backup-id>
   ```

   ```
   ./epasctl create cluster <cluster-name -f cluster_spec_file>
   --restore-data
   ```

2. To restore upto a specific timestamp, i.e. Point-in-time Recovery (PITR), use the following commands:

   ```
   ./epasctl restore cluster <cluster-name -i backup-id -t "timestamp">
   ```

   ```
   ./epasctl create cluster <cluster-name -f cluster_spec_file>
   --restore-data
   ```

Restore using pg_restore:

To restore a cluster previously backed up by pg_dump, use the command:

```
./epasctl create cluster <cluster_name -f cluster_spec_file>
--use-pgrestore
```

4.2.4 Scaling an Advanced Server Cluster

An existing cluster may be scaled by increasing or decreasing the number of standbys:

Run the following command for scaling a cluster:

```
./epasctl scale cluster <cluster_name> -standby-count=
<desired-standby-count>
```

4.2.5 Displaying Status of an Advanced Server Cluster

To display the status of an existing cluster, run the command

```
./epasctl show cluster <cluster_name>
```

4.2.6 Monitoring an Advanced Server Cluster

To enable monitoring of the cluster, set the `enableMonitorMode` setting to `Yes` in the database section of the cluster specification file.

Details of the PEM console:

Console URL: Displayed upon successful creation of the cluster

Credentials: Provided in the **.pemserverpass** file.

The URL of the PEM console can be redisplayed by running the cluster status command:

```
./epasctl show cluster <cluster_name>
```

4.2.7 Deleting an Advanced Server Cluster

To delete a cluster along with its associated objects, run the following command:

```
./epasctl delete cluster <my_cluster>
```

4.3 List of Supported Functions

Function	Example Command	Description
Create database cluster	./epasctl create cluster edb1 -f create-epas.yaml –use-pgrestore	Creates an Advanced Server cluster named edb1 with the specs provided in the input manifest create-epas.yaml
Scale database cluster	./epasctl scale cluster edb1–standby-count=3	Scales the cluster edb1 (up or down) by changing the number standbys to 3
Backup database cluster manually using BART	./epasctl backup cluster edb1–method=bart	Manually (on-demand) backs up the cluster edb1 using BART.s
Backup database cluster manually using pg_dump	./epasctl backup cluster edb1–method=pgdump	Manually (on-demand) backs up the cluster edb1using pg_dump.
Backup database cluster automatically using BART	In the backup section of the input manifest, set enabled = true and schedule=*cron-formatted-schedule*	Automatically backs up the Advanced Server cluster per provided schedule backup section of the input manifest.
Show status of database cluster	./epasctl show cluster edb1	Displays the current status of the cluster edb1
Delete database cluster	./epasctl delete cluster edb1	Deletes the cluster edb1 along with its associated objects
Provide failover capability	In the input manifest, set enableHAMode=Yes in the database section	Enables the Advanced Server cluster to function in high availability mode via EDB Failover Manager (EFM)
Monitor database cluster	In the input manifest, set enableMonitorMode=Yes in the database section	Automatically registers the Advanced Server cluster with the PEM server deployed in its namespace
Restore database cluster using pg_restore	./epasctl create cluster edb1 -f create-epas.yaml –use-pgrestore	Creates cluster edb1 with the specs provide in the input manifest create-epas.yaml and restores data from pg_dump taken previously.
Restore database cluster using BART	./epasctl restore cluster edb1 -i *backup-id* ./epasctl create cluster edb1 -f create-epas.yaml –restore-data	Restores data from backup identified by *backup-id* into a staging area. Creates cluster edb1 with the specs provided in the input manifest create-epas.yaml and restores data from staging area.
Perform Point-in-time Recovery (PITR) of database cluster using BART	./epasctl restore cluster edb1 -i *backup-id* -t *timestamp* ./epasctl create cluster edb1 -f create-epas.yaml –restore-data	Restores data from backup identified by *backup-id* and *timestamp* into a staging area. Creates cluster edb1 with the specs providedin the input manifest create-epas.yaml and restores data from staging area.
Migrate database cluster	/epasctl backup cluster edb1 –method=pgdump ./epasctl delete cluster edb Update the cluster spec file create-epas.yaml with the desired postgres version ./epasctl create cluster edb1 -f create-epas.yaml –use-pgrestore	Migrates cluster edb1 to a different postgres version using pg_dump and pg_restore.

CHAPTER 5

Conclusion

EDB™ Postgres Containers Operator guide

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

- EDB designs, establishes coding best practices, reviews, and verifies input validation for the logon UI for EDB Postgres 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