Crunchy Postgres Operator

Installation Guide

DRAFT

Crunchy Data Solutions, Inc.

April 5, 2018

Table of Contents

1 Project Structure

To perform an installation of the operator, first create the project structure as follows on your host, here we assume a local directory called **odev**:

```
export GOPATH=$HOME/odev
mkdir -p $HOME/odev/src $HOME/odev/bin $HOME/odev/pkg
mkdir -p $GOPATH/src/github.com/crunchydata/
```

Next, get a tagged release of the source code:

```
cd $GOPATH/src/github.com/crunchydata
git clone https://github.com/CrunchyData/postgres-operator.git
cd postgres-operator
git checkout 2.6
```

2 Get Images and Binaries

To pull prebuilt versions from Dockerhub of the **postgres-operator** containers, specify the image versions, and execute the following Makefile target:

```
export CO_IMAGE_PREFIX=crunchydata
export CO_IMAGE_TAG=centos7-2.6
make pull
```

To pull down the prebuilt **pgo** binaries, download the **tar.gz** release file from the following link:

- GITHUB RELEASES
- extract (e.g. tar xvzf postgres-operator.2.6-rc1.tar.gz)

```
cd $HOME
tar xvzf ./postgres-operator.2.6-rc1.tar.gz
```

- copy **pgo** client to somewhere in your path (e.g. cp pgo /usr/local/bin)

3 Installation Prerequsites

To run the operator and the **pgo** client, you will need the following:

- a running Kube cluster
- a kubectl client installed and in your PATH and configured to connect to your Kube cluster (e.g. export KUBECONFIG=/etc/kubernetes/admin.conf)
- a Kube namespace created and set to where you want the operator installed, for this install we assume a namespace of **demo** has been created

```
kubectl create -f examples/demo-namespace.json
kubectl config set-context $(kubectl config current-context) --namespace=demo
kubectl config view | grep namespace
```

4 Basic Installation

The basic installation uses the default operator configuration settings, these settings assume you want to use HostPath storage on your Kube cluster for database persistence. Other persistent options are available but require the Advanced Installation below.

4.1 Create HostPath Directory

The default Persistent Volume script assumes a default HostPath directory be created called /**data**:

```
sudo mkdir /data
sudo chown 777 /data
```

Create some sample Persistent Volumes using the following script:

```
export CO_NAMESPACE=demo
export CO_CMD=kubectl
export COROOT=$GOPATH/src/github.com/crunchydata/postgres-operator
go get github.com/blang/expenv
$COROOT/pv/create-pv.sh
```

4.2 Deploy the Operator

Next, deploy the operator to your Kube cluster:

```
cd $COROOT
make deployoperator
```

Instead of using the bash script you can also deploy the operator using the provided Helm chart:

```
cd $COROOT/chart
helm install ./postgres-operator
helm ls
```

4.3 Verify Operator Status

To verify that the operator is deployed and running, run the following:

```
kubectl get pod --selector=name=postgres-operator
```

You should see output similar to this:

```
NAME                          READY   STATUS  RESTARTS AGE
postgres-operator-56598999cd-tbg4w 2/2 Running 0   1m
```

There are 2 containers in the operator pod, both should be **ready** as above.

The operator creates the following Custom Resource Definitions:

```
kubectl get crd
NAME                      AGE
pgbackups.cr.client-go.k8s.io 2d
pgclusters.cr.client-go.k8s.io 2d
pgingests.cr.client-go.k8s.io 2d
pgpolicies.cr.client-go.k8s.io 2d
pgreplicas.cr.client-go.k8s.io 2d
pgtasks.cr.client-go.k8s.io 2d
pgupgrades.cr.client-go.k8s.io 2d
```

At this point, the server side of the operator is deployed and ready.

The complete set of environment variables used in the installation so far are:

```
export CO_IMAGE_PREFIX=crunchydata
export CO_IMAGE_TAG=centos7-2.6
export GOPATH=$HOME/odev
export GOBIN=$GOPATH/bin
export PATH=$PATH:$GOBIN
export COROOT=$GOPATH/src/github.com/crunchydata/postgres-operator
export CO_CMD=kubectl
```

You would normally add these into your **.bashrc** at this point to be used later on or if you want to redeploy the operator.

4.4 Configure pgo Client

The **pgo** command line client requires TLS for securing the connection to the operator's REST API. This configuration is performed as follows:

```
export PGO_CA_CERT=$COROOT/conf/apiserver/server.crt
export PGO_CLIENT_CERT=$COROOT/conf/apiserver/server.crt
export PGO_CLIENT_KEY=$COROOT/conf/apiserver/server.key
```

The **pgo** client uses Basic Authentication to authenticate to the operator REST API, for authentication, add the following **.pgouser** file to your $HOME:

```
echo "username:password" > $HOME/.pgouser
```

The **pgo** client needs the URL to connect to the operator.

Depending on your Kube environment this can be done the following ways:

4.4.1 Running Kube Locally

If your local host is not set up to resolve Kube Service DNS names, you can specify the operator IP address as follows:

```
kubectl get service postgres-operator
NAME            TYPE    CLUSTER-IP EXTERNAL-IP PORT(S)    AGE
postgres-operator NodePort 10.109.184.8 <none> 8443:30894/TCP 5m

export CO_APISERVER_URL=https://10.109.184.8:8443
pgo version
```

You can also define a bash alias like:

```
alias setip='export CO_APISERVER_URL=https://`kubectl get service postgres-
   operator -o=jsonpath="{.spec.clusterIP}"`:8443'
```

This alias will set the CO_APISERVER_URL IP address for you!

4.4.2 Running Kube Remotely

Set up a port-forward tunnel from your host to the Kube remote host, specifying the operator pod:

```
kubectl get pod --selector=name=postgres-operator
NAME                        READY   STATUS  RESTARTS AGE
postgres-operator-56598999cd-tbg4w 2/2 Running 0   8m

kubectl port-forward postgres-operator-56598999cd-tbg4w 8443:8443
```

In another terminal:

```
export CO_APISERVER_URL=https://127.0.0.1:8443
pgo version
```

4.5 Verify pgo Client

At this point you should be able to connect to the operator as follows:

```
pgo version
pgo client version 2.6
apiserver version 2.6
```

pgo commands are documented on the COMMANDS page.

5 Custom Installation

Most users after they try out the operator will want to create a more customized installation and deployment of the operator.

5.1 Specify Storage

The operator will work with HostPath, NFS, and Dynamic Storage.

5.1.1 NFS

To configure the operator to use NFS for storage, a sample **pgo.yaml.nfs** file is provided. Overlay the default **pgo.yaml** file with that file:

```
cp $COROOT/examples/pgo.yaml.nfs $COROOT/conf/apiserver/pgo.yaml
```

Edit the **pgo.yaml** file to specify the NFS GID that is set for the NFS volume mount you will be using, the default value assumed is **nfsnobody** as the GID (65534). Update the value to meet your NFS security settings.

There is currently no script available to create your NFS Persistent Volumes but you can typically modify the $COROOT/pv/create-pv.sh script to work with NFS.

5.1.2 Dynamic Storage

To configure the operator to use Dynamic Storage classes for storage, a sample **pgo.yaml.storageclass** file is provided. Overlay the default **pgo.yaml** file with that file:

```
cp $COROOT/examples/pgo.yaml.storageclass $COROOT/conf/apiserver/pgo.yaml
```

Edit the **pgo.yaml** file to specify the storage class you will be using, the default value assumed is **standard** which is the name used by default within a GKE Kube cluster deployment. Update the value to match your storage classes.

Notice that the **FsGroup** setting is required for most block storage and is set to the value of **26** since the PostgreSQL container runs as UID **26**.

5.2 Change the Operator Configuration

There are many ways to configure the operator, those configurations are documented on the CONFIGURATION page.

Reasonable defaults are specified which allow users to typically run the operator at this point so you might not initially require any customization beyond specifying your storage.

5.3 Deploy and Run

At this point, you can use the Basic Installation Deploy steps to deploy the operator and run the **pgo** client.