Crunchy PostgreSQL Operator Open Service Broker (pgo-osb)

Latest Release: v4.6.3, 2021-05-28

General

The **pgo-osb** project is an implementation of the Open Service Broker API. This implementation uses the Crunchy PostgreSQL Operator as a means to provision services, in this case the service is a PostgreSQL database cluster.

pgo-osb allows users to also bind to a *service instance* which when invoked will return PostgreSQL credentials to a user they can use to connect to the PostgreSQL database instance.

Also, users can *deprovision* a PostgreSQL database cluster using the OSB API.

The **pgo-osb** broker was developed using the *OSB Starter Pack* and associated libraries.

See the following:

- Open Service Broker API
- `osb-broker-lib`.
- `go-open-service-broker-client`
- service-catalog

Compatibility

Starting with **pgo-osb** version 4.0.0, the release schedule and version number for **pgo-osb** will be aligned with the release schedule and version number for the the Crunchy PostgreSQL Operator. Therefore, to ensure compatibility between **pgo-osb** and the PostgreSQL Operator, please ensure the version number for **pgo-osb** matches the version number of the PostgreSQL Operator deployed in your environment. For instance, if you are using **pgo-osb** v4.6.3, please ensure the Crunchy PostgreSQL Operator v4.6.3 is also deployed in your environment.

Prerequisites

golang 1.9 or above is required to build this project.

Running the **pgo-osb** service broker assumes you have successfully deployed the PostgreSQL Operator. See the PostgreSQL Operator documentation for documentation on deploying the PostgreSQL Operator:

https://access.crunchydata.com/documentation/postgres-operator/

Please note that if **pgo-osb** is deployed to a different namespace than the PostgreSQL Operator, DNS must be utilized when specifying the URL for the PostgreSQL Operator API server. This is done using environment variable `PGO_APISERVER_URL` in the **pgo-osb** `deployment.yaml` file (located in directory `$OSB_ROOT/deploy`). For instance, if the PostgreSQL Operator is deployed to namespace `pgo`, the `PGO_APISERVER_URL` environment variable would be set in this file as follows:

```
- --PGO_APISERVER_URL
- "https://postgres-operator.pgo.svc.cluster.local:8443"
```

However, if **pgo-osb** is deployed to the same namespace as the PostgreSQL Operator, then the PostgreSQL Operator service name can simply be utilized:

```
- --PGO_APISERVER_URL
- "https://postgres-operator:8443"
```

Additionally, **pgo-osb** must also be configured with the certificates needed to properly authenticate into and trust the PostgreSQL Operator API server. When installing the PostgreSQL Operator API server these certificates are automatically generated, and must be copied into directory `$OSB_ROOT/deploy` prior to deploying **pgo-osb**. This allows the certificates to be stored in a secret that can be utilized by **pgo-osb** when accessing the PostgreSQL Operator API server. For instance, if the PostgreSQL Operator was installed using the `bash` installation method, the certificates can be copied as follows:

```
cp $PGOROOT/conf/postgres-operator/server.crt $PGOROOT/conf/postgres-operator/server.key $OS
```

Or if the PostgreSQL Operator was installed using Ansible, then the certificates can be copied from your home directory as follows:

```
cp "${HOME}"/.pgo/"${PGO_OPERATOR_NAMESPACE}"/output/server.crt $OSB_ROOT/deploy
cp "${HOME}"/.pgo/"${PGO_OPERATOR_NAMESPACE}"/output/server.pem $OSB_ROOT/deploy/server.key
```

This example also assumes you have created a Kube namespace called *demo*. Adjust OSB_NAMESPACE to suit your specific namespace value. And lastly, the example assumes you are using the PostgreSQL Operator default RBAC account called `username` with a password of `password`. If this is not the case then you will need to adjust the example service instance `service-instance.yaml`.

Operator Configuration

The standalone and ha service plans require custom storage and container resource configurations in the PostgreSQL Operator's `pgo.yaml` definition. Refer to the Operator documentation:

https://access.crunchydata.com/documentation/postgres-operator/latest/configuration/pgo-yaml-configuration/

The Open Service Broker will request custom storage and container resources corresponding to the size of plan, using the names `osbsmall`, `osbmedium`, `osblarge`. For example, the `standalone_md` plan will use disk sizes defined by the `osbmedium` custom storage definition and the memory and CPU limits defined by the `osbmedium` container resource definition.

Example configuration descriptions:

```
Storage:
  osbsmall:
    AccessMode:  <based on environment>
    Size:  300M
    StorageType:  <based on environment>
    StorageClass:  <based on environment>
    Fsgroup:  26
  osbmedium:
    AccessMode:  <based on environment>
    Size:  600M
    StorageType:  <based on environment>
    StorageClass:  <based on environment>
    Fsgroup:  26
  osblarge:
    AccessMode:  <based on environment>
    Size:  2G
    StorageType:  <based on environment>
    StorageClass:  <based on environment>
    Fsgroup:  26
ContainerResources:
  osbsmall:
    RequestsMemory:  512Mi
    RequestsCPU:  0.1
    LimitsMemory:  512Mi
    LimitsCPU:  1.0
  osbmedium:
    RequestsMemory:  1Gi
    RequestsCPU:  0.5
    LimitsMemory:  1Gi
    LimitsCPU:  2.0
  osblarge:
    RequestsMemory:  2Gi
    RequestsCPU:  1.0
    LimitsMemory:  2Gi
    LimitsCPU:  4.0
```

Build

To build the **pgo-osb** broker, place these additional environment variables into your `.bashrc` as they are used in the various scripts and deployment templates:

```
export GOPATH=$HOME/odev
export GOBIN=$GOPATH/bin
export PATH=$GOBIN:$PATH
export OSB_NAMESPACE=demo
export OSB_CMD=kubectl
export OSB_ROOT=$GOPATH/src/github.com/crunchydata/pgo-osb
export OSB_BASEOS=centos7
export OSB_VERSION=4.6.3
export OSB_IMAGE_TAG=$OSB_BASEOS-$OSB_VERSION
export OSB_IMAGE_PREFIX=crunchydata
```

Install the dep dependency tool:

```
mkdir $GOPATH/bin $GOPATH/src/github.com/crunchydata $GOPATH/pkg -p
curl https://raw.githubusercontent.com/golang/dep/master/install.sh | sh
```

Get the code:

```
cd $GOPATH/src/github.com/crunchydata
git clone https://github.com/crunchydata/pgo-osb.git
cd pgo-osb
```

Deploy Service Catalog

Install the service catalog into your Kubernetes cluster by following this link:

https://svc-cat.io/docs/install/

Instructions on that link are provided to also install the very useful `svcat` utility for inspecting and working with the service catalog.

Deploy

Deploy the **pgo-osb** broker:

```
make setup
make image
make deploy
```

Verify your deployment has been successful with:

```
kubectl get pod --selector=app=pgo-osb
```

which has output similar to:

```
NAME                          READY     STATUS    RESTARTS   AGE
pgo-osb-69c76578b9-v7s9k      1/1       Running   0          16m
```

Working with the pgo-osb

To use the **pgo-osb** broker, please follow the following instructions.

Note that if you want to specify a specific namespace for where your PostgreSQL cluster is deployed to, you can use the `PGO_CLUSTER_NAMESPACE` environmental variable. Otherwise, **pgo-osb** will search across all namespaces to look up where the cluster exists.

Show Available Plans

```
svcat marketplace
```

which has output similar to:

```
       CLASS             PLANS        DESCRIPTION
+-----------------+---------------+--------------+
  pgo-osb-service   standalone_lg   The pgo osb!
                    ha_lg
                    default
                    ha_sm
                    standalone_sm
                    ha_md
                    standalone_md
```

Note: Additional services installed in your environment may be listed as well.

Create a Service Instance

```
cd $OSB_ROOT
make provision
kubectl get serviceinstance
make provision2
kubectl get serviceinstance
```

Please note the `ServiceInstance` objects created when running the `make provision` and `make provision2` commands above will create PostgreSQL cluster's in the default namespace set for the PostgreSQL Operator according to the `PGO_NAMESPACE` environment variable set in your environment. If you would like the clusters to be provisioned in another namespace, please set the proper namespace using the `PGO_NAMESPACE` parameter in files `$OSB_ROOT/manifests/service-instance.yaml` and `$OSB_ROOT/manifests/service-instance2.yaml`.

You should see a pod with that service instance name:

```
kubectl get pod --selector=name=testinstance
kubectl get pod --selector=name=testinstance2
```

Create a Binding

```
make bind
kubectl get servicebinding
make bind2
kubectl get servicebinding
```

Display the Binding with Secrets

You can view the binding and the generated Postgres credentials using this command:

```
svcat describe binding testinstance-binding -n $OSB_NAMESPACE
```

which has output similar to:

```
  Name:        testinstance-binding
  Namespace:   demo
  Status:      Ready - Injected bind result @ <timestamp>
  Secret:      testinstance-binding
  Instance:    testinstance

Parameters:
  No parameters defined

Secret Data:
  db_host         12 bytes
  db_name         6 bytes
  db_port         4 bytes
  internal_host   12 bytes
  password        16 bytes
  uri             85 bytes
  username        30 bytes
```

Display the Binding with Secrets

```
svcat describe binding testinstance-binding --show-secrets -n $OSB_NAMESPACE
```

which has output simialr to:

```
  Name:        testinstance-binding
  Namespace:   demo
  Status:      Ready - Injected bind result @ <timestamp>
  Secret:      testinstance-binding
  Instance:    testinstance

Parameters:
  No parameters defined

Secret Data:
  db_host         10.96.22.114
  db_name         userdb
  db_port         5432
  internal_host   10.96.22.114
  password        LEYtDzLOEMZTqiRH
  uri             postgresql://userd4a4kthjhyi6to6vvz5vdh4die:LEYtDzLOEMZTqiRH@10.96.22.114:
  username        userd4a4kthjhyi6to6vvz5vdh4die
```

You can also use the `svcat` Service Catalog CLI to inspect the service catalog.

View the Service Brokers

```
svcat get brokers
```

which will have output simialr to:

```
   NAME                        URL                        STATUS
+---------+-----------------------------------------+--------+
  pgo-osb   http://pgo-osb.demo.svc.cluster.local:443   Ready
```

Get the Service Class

```
svcat get classes
```

which will have output similar to:

```
    NAME         DESCRIPTION
+----------------+-------------+
  pgo-osb-service   The pgo osb!
```

Note: Additional service classes installed in your environment may be listed as well.

View the Service Class

```
svcat describe class pgo-osb-service
```

which will have output similar to:

```
Name:          pgo-osb-service
Description:   The pgo osb!
UUID:          4be12541-2945-4101-8a33-79ac0ad58750
Status:        Active
Tags:
Broker:        pgo-osb
Plans:
    NAME              DESCRIPTION
+---------+---------------------------------+
  default   The default plan for the pgo osb service
```

View Instances in a Namespace

```
svcat get instances -n $OSB_NAMESPACE
```

which will have output similar to:

```
NAME      NAMESPACE        CLASS          PLAN     STATUS
+-----------+-----------+-----------------+---------+--------+
  testinstance   demo        pgo-osb-service   default   Ready
  testinstance2  demo        pgo-osb-service   default   Ready
```

Cleanup Examples

You can remove the bindings and instances using these commands:

```
svcat unbind testinstance -n $OSB_NAMESPACE
svcat unbind testinstance2 -n $OSB_NAMESPACE
svcat deprovision testinstance -n $OSB_NAMESPACE
svcat deprovision testinstance2 -n $OSB_NAMESPACE
```

Contributing to the Project

Want to contribute to the **pgo-osb** project? Great! We've put together as set of contributing guidelines that you can review here:

- Contributing Guidelines