Crunchy Data
PostgreSQL Operator

September 2017
Crunchy Data

Commitment to Community and Open Source

Emphasis on Data Security and Compliance

PostgreSQL for Public, Private and Hybrid Cloud
What is an Operator?
Operator Basics

- **Open Source** - project at [https://github.com/crunchydata/postgres-operator](https://github.com/crunchydata/postgres-operator)
- **Controller** - an Operator is essentially a controller component you run on your Kubernetes or OpenShift cluster
- **Automation** - the Operator implements the necessary workflow orchestration logic to enable the automation of PostgreSQL related tasks
- **Leverages the Kube API** - the Operator is implemented in golang and leverages the Kubernetes Client API heavily to look for events, update labels, create containers, etc. [https://github.com/kubernetes/client-go](https://github.com/kubernetes/client-go)
- **Command Line Interface** - the Operator has a Command Line Interface that lets users manipulate the Operator and define metadata used to catalog your PostgreSQL deployments
- **Deployment** - the Operator runs as a standard Deployment, it watches for Third Party Resources that we define to manage PostgreSQL deployments
- **Third Party Resources** - the operator makes use of Kubernetes Third Party Resources currently to store metadata about PostgreSQL deployments, this will change to using Kubernetes Custom Resource Definitions in future versions.
- **Custom PostgreSQL Cluster Definitions** - a template based approach for allowing users an ability to define the makeup of a PostgreSQL cluster rather than the current default
Operator Basics
Operator Basics
Why?
Why Do I Need This?

- **Automation** - working with databases requires different workflows that involve multiple steps and verifications, the Operator allows for automating those sort of workflows in a consistent way and can reduce human errors; working with large numbers of complex database deployments can get time consuming without some sort of automation.

- **Standard Practices** - advanced users will want to have a standard set of database objects created so the ability to categorize databases with metadata and also build/apply SQL policies for them is useful to support a set of standards.

- **Ease of Use** - a simple CLI that is focused solely on deploying and managing a set of PostgreSQL clusters enables users to focus on getting real work done instead of having to build their own set of scripts or adjust to learning the details of kubectl commands.

- **Large Scale Deployments** - the Operator supports environments where you might have possibly hundreds of PostgreSQL clusters and being able to query these assets in a managed way is useful.

- **Complex Orchestrations** - the Operator provides a means of developing advanced database orchestration logic using a standards-based approach.
Building Blocks
Building Blocks

- https://github.com/crunchydata/crunchy-containers/
- crunchy-postgres
- crunchy-proxy
- crunchy-backup
- crunchy-upgrade
Installation

- [https://github.com/CrunchyData/postgres-operator/blob/master/docs/build.asciidoc](https://github.com/CrunchyData/postgres-operator/blob/master/docs/build.asciidoc)
- **Requirements**
  - Kubernetes 1.5.3+
  - OpenShift Origin 1.5.1+ and OpenShift Container Platform 3.5
  - PostgreSQL 9.5+ Container (crunchy-postgres)
  - PostgreSQL Backup Container (crunchy-backup)
  - PostgreSQL Upgrade Container (crunchy-upgrade)
- **Building the project**
- **Get packaged dependencies/build from source**
- **Bash completion**
- **View operator resources**
  - `kubectl get thirdpartyresources`
  - `kubectl get pgclusters`
  - `kubectl get pgbackups`
  - `kubectl get pgupgrades`
  - `kubectl get pgpolicies`
  - `kubectl get pgclones`
Installing Kubernetes

- [https://gist.github.com/xenophenes/05552cadbba42c6abcc039aad7273e60](https://gist.github.com/xenophenes/05552cadbba42c6abcc039aad7273e60)
  - (https://goo.gl/22fopq)
- Minikube
  - minikube addons disable default-storageclass
Features
Create a Database Cluster

The most basic function of the Operator is to let a user create a new Database “Cluster”. Here Cluster refers to a set of resources (services, Deployments, Pods, etc.)

```
ubuntu:operator$ pgo create cluster isengard
created PgCluster isengard
ubuntu:operator$ pgo show cluster isengard
cluster : isengard (9.6.3)
├── deployment : isengard
│    └── replicaset : isengard-3576348909
│         └── pod : isengard-3576348909-zrs2x (Running on kubeminion-01) (0/1)
│              └── service : isengard (10.108.213.82)

ubuntu:operator$ pgo show cluster isengard
cluster : isengard (9.6.3)
├── deployment : isengard
│    └── replicaset : isengard-3576348909
│         └── pod : isengard-3576348909-zrs2x (Running on kubeminion-01) (1/1)
│              └── service : isengard (10.108.213.82)

ubuntu:operator$ kubectl describe pod isengard-3576348909-zrs2x
Name:        isengard-3576348909-zrs2x
Namespace:   default
Node:        kubeminion-01/172.28.128.3
```
Test a Database

You can test the database cluster with a simple test as follows:

```
ubuntu:operator$ pgo test isengard

cluster : isengard (9.6.3)
psql -p 5432 -h 10.108.213.82 -U master postgres is working
psql -p 5432 -h 10.108.213.82 -U postgres postgres is working
psql -p 5432 -h 10.108.213.82 -U testuser userdb is working
```
Delete a Database Cluster

You can delete the set of Database Cluster resources by name as follows:

```bash
ubuntu:operator$ pgo delete cluster isengard
delete pgcluster isengard
ubuntu:operator$ kubectl get pvc isengard-pvc
NAME    STATUS   VOLUME       CAPACITY  ACCESSMODES  AGE
isengard-pvc Bound   crunchy-pv25 1Gi       RWX      20h
ubuntu:operator$ kubectl delete pvc isengard-pvc
persistentvolumeclaim "isengard-pvc" deleted
```
Backup a Database

You can execute a backup job, jobs can run for a very long time so the Operator is able to watch for their completion using the Kubernetes Watch API, when complete the Operator is notified:

```
ubuntu:operator$ pgo backup isengard
created PgBackup isengard
ubuntu:operator$ pgo show backup isengard

backup information for isengard...

pgbackup : isengard
- PVC Name:
- PVC Access Mode: ReadWriteMany
- PVC Size: 100M
- CCP_IMAGE_TAG: centos7-9.6-1.4.1
- Backup Status: submitted
- Backup Host: isengard
- Backup User: master
- Backup Pass: password
- Backup Port: 5432

backup job pods for database isengard...
- backup-isengard-lzhjp (pvc isengard-backup-pvc)
```
Show Backup PVCs

This displays the contents of a PVC (pgo show pvc) - not just the backup pvc as displayed here. This is the equivalent of the kubectl get pvc command.

```
ubuntu:operator$ kubectl get pvc
NAME         STATUS   VOLUME        CAPACITY ACCESSMODES AGE
crunchy-pvc  Bound   crunchy-pv48 1Gi    RWX    1h
example-pvc  Bound   crunchy-pv9  1Gi    RWX    1h
foo-pvc      Bound   crunchy-pv2  1Gi    RWX    1h
isengard-backup-pvc Bound crunchy-pv29 1Gi    RWX    2m
isengard-pvc  Bound   crunchy-pv25 1Gi    RWX    1h
```

```
ubuntu:operator$ pgo show pvc isengard-backup-pvc
isengard-backup-pvc
  /example
  /isengard
  /isengard-backups
```

```
ubuntu:operator$ pgo show pvc isengard-backup-pvc --pvc-root=isengard-backups
isengard-backup-pvc/isengard-backups/2017-09-04-19-44-31
```
Restore from Backup

Create a restored database cluster from a previous backup using passed in variables to define the backup pvcs.

```bash
ubuntu:operator$ pgo show pvc isengard-backup-pvc --pvc-root=isengard-backups
isengard-backup-pvc/isengard-backups
   /2017-09-04-19-44-31

ubuntu:operator$ pgo create cluster isengard-restored --backup-pvc=isengard-backup-pvc
   --backup-path=isengard-backups/2017-09-04-19-44-31 --secret-from=isengard
created PgCluster isengard-restored
```
Show Restored Cluster

Display information about the cluster we just restored; verify everything looks OK.

```
ubuntu:operator$ pgo show cluster isengard-restored
cluster : isengard-restored (9.6.3)
  - deployment : isengard-restored
  - replicaset : isengard-restored-3782209573
  - pod : isengard-restored-3782209573-2kcwn (Running on kubeminion-01) (1/1)
  - service : isengard-restored (10.111.150.248)
```

```
ubuntu:operator$ pgo test isengard-restored

cluster : isengard-restored (9.6.3)
psql -p 5432 -h 10.111.150.248 -U master postgres is working
psql -p 5432 -h 10.111.150.248 -U postgres postgres is working
psql -p 5432 -h 10.111.150.248 -U testuser userdb is working
```
Scale a Database

You can scale up the number of read-only replicas as follows:

```bash
ubunto:operator$ pgo scale isengard-restored --replica-count=1
scaling isengard-restored to 1
ubunto:operator$ pgo show cluster isengard-restored
cluster: isengard-restored (9.6.3)
  ├── deployment: isengard-restored
  │    └── deployment: isengard-restored(replica-picl)
  │         └── replicaset: isengard-restored-3782209573
  │                 └── replicaset: isengard-restored(replica-picl-2898964347)
  │                           └── pod: isengard-restored-3782209573-2kcwn (Running on kubeminion-01) (1/1)
  │                                                └── pod: isengard-restored(replica-picl-2898964347-ztf6j (Running on kubeminion-01) (1/1)
  └── service: isengard-restored (10.111.150.248)
      └── service: isengard-restored(replica (10.108.250.169)
```
Perform a Minor Database Upgrade

You can upgrade to a new minor release as follows:

```
ubuntu:operator$ pgo upgrade isengard-restored --ccp-image-tag=centos7-9.6-1.5.1
created PgUpgrade isengard-restored
ubuntu:operator$ pgo test isengard-restored
```

```
cluster : isengard-restored (unknown)
psql -p 5432 -h 10.111.150.248 -U master postgres is working
psql -p 5432 -h 10.111.150.248 -U postgres postgres is working
psql -p 5432 -h 10.111.150.248 -U testuser userdb is working
psql -p 5432 -h 10.108.250.169 -U master postgres is working
psql -p 5432 -h 10.108.250.169 -U postgres postgres is working
psql -p 5432 -h 10.108.250.169 -U testuser userdb is working
```
Perform a Major Database Upgrade

You can upgrade to a new major release as follows:

```
ubuntu:operator$ pgo upgrade isengard-before --upgrade-type=major --ccp-image-tag=centos7-9.6-1.5.1
```
created PgUpgrade isengard-before
Create a Database Policy

You can define a set of SQL statements which we call a Policy:

```
ubuntu:operator$ vi policy1.sql
ubuntu:operator$ cat policy1.sql
create table policy1 (id int);
ubuntu:operator$ pgo create policy policy1 --in-file=policy1.sql
created PgPolicy policy1
ubuntu:operator$ pgo show policy policy1

policy : policy1
  url :
  status :
  sql : create table policy1 (id int);
```
Apply a Database Policy

You can apply a Policy against a set of databases as follows:

```bash
ubuntu:operator$ pgo apply policy1 --selector=name=isengard
deployment : isengard
created PgPolicylog policy1isengard
ubuntu:operator$ pgo show cluster isengard
cluster : isengard (9.6.3)
  deployment : isengard
  policy: policy1
  replicaset : isengard-3576348909
  pod : isengard-3576348909-k3swg (Running on kubeminion-01) (1/1)
  service : isengard (10.111.149.154)
```
Create Policy from GitHub

You can define a set of SQL statements which we call a Policy and import the file from GitHub:

```
create PgPolicy gitpolicy
```

```
$ kubectl get pgpolicies gitpolicy -o json
{
  "apiVersion": "crunchydata.com/v1",
  "kind": "PgPolicy",
  "metadata": {
    "creationTimestamp": "2017-09-06T05:23:12Z",
    "name": "gitpolicy",
    "namespace": "default",
    "resourceVersion": "1445",
    "selfLink": "/apis/crunchydata.com/v1/namespaces/default/pgpolicies/gitpolicy",
    "uid": "79ff856-92c3-11e7-8507-02c35d847e20"
  },
  "spec": {
    "name": "gitpolicy",
    "sql": "",
    "status": "",
    "url": "https://github.com/CrunchyData/postgres-operator/blob/master/examples/policy/gitpolicy.sql"
  }
}
Clone Database

Clone an existing PostgreSQL cluster, waits for the replica to complete replication, decouples the replica from the master by triggering a recovery and re-labeling the replica to become a fully functioning master, becomes a fully functioning master, creates a service for the new master and an empty replica deployment.

```
ubuntu:operator$ pgo clone isengard-restored --name=isengard2
cloned isengard-restored as isengard2
```
Multiple Clusters

Command line parameters can be passed in with multiple values, and all clusters can be viewed.

```
ubuntu:postgres-operator$ pgo create cluster frodo-baggins samwise-gamgee meriadoc-brandybuck peregrin-took
created PgCluster frodo-baggins
created PgCluster samwise-gamgee
created PgCluster meriadoc-brandybuck
created PgCluster peregrin-took
ubuntu:postgres-operator$ pgo show cluster all
cluster : frodo-baggins ()
  deployment : frodo-baggins
  replicaset : frodo-baggins-369741252
  pod : frodo-baggins-369741252-kzxcf (Running on kubeminion-01) (1/1)
  service : frodo-baggins (10.109.27.207)

cluster : meriadoc-brandybuck ()
  deployment : meriadoc-brandybuck
  replicaset : meriadoc-brandybuck-4261601008
  pod : meriadoc-brandybuck-4261601008-7q1w4 (Running on kubeminion-01) (1/1)
  service : meriadoc-brandybuck (10.102.167.28)

cluster : peregrin-took ()
  deployment : peregrin-took
  replicaset : peregrin-took-2971520704
  pod : peregrin-took-2971520704-68qjt (Running on kubeminion-01) (1/1)
  service : peregrin-took (10.97.211.184)

cluster : samwise-gamgee ()
  deployment : samwise-gamgee
  replicaset : samwise-gamgee-1756221610
  pod : samwise-gamgee-1756221610-c6kb2 (Running on kubeminion-01) (1/1)
  service : samwise-gamgee (10.102.130.103)
```
.pgo.yaml

Configuration YAML file for the postgres-operator.

https://github.com/CrunchyData/postgres-operator/blob/master/docs/config.asciidoc

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

https://kubernetes.io/docs/concepts/storage/volumes/
Strategies (Resources)

- https://github.com/CrunchyData/postgres-operator/blob/master/docs/design.asciidoc
- https://github.com/CrunchyData/postgres-operator/tree/master/conf/postgres-operator/cluster/1
- https://github.com/CrunchyData/postgres-operator/blob/master/examples/pgo.yaml.emptydir
- https://github.com/CrunchyData/postgres-operator/blob/master/docs/config.asciidoc
Summary
THANK YOU!

Learn more:
• [https://www.youtube.com/watch?v=HX10WWTRiTY](https://www.youtube.com/watch?v=HX10WWTRiTY)
• [https://github.com/CrunchyData/postgres-operator](https://github.com/CrunchyData/postgres-operator)
• [https://www.crunchydata.com/](https://www.crunchydata.com/)

Slides are available at:
• [https://www.sarahconway.com/slides/postgres-operator.pdf](https://www.sarahconway.com/slides/postgres-operator.pdf)

Sarah Conway: [sarah.conway@crunchydata.com](mailto:sarah.conway@crunchydata.com)
Jeff McCormick: [jeff.mccormick@crunchydata.com](mailto:jeff.mccormick@crunchydata.com)