Crunchy Data PostgreSQL Operator

March 2018

Crunchy Data

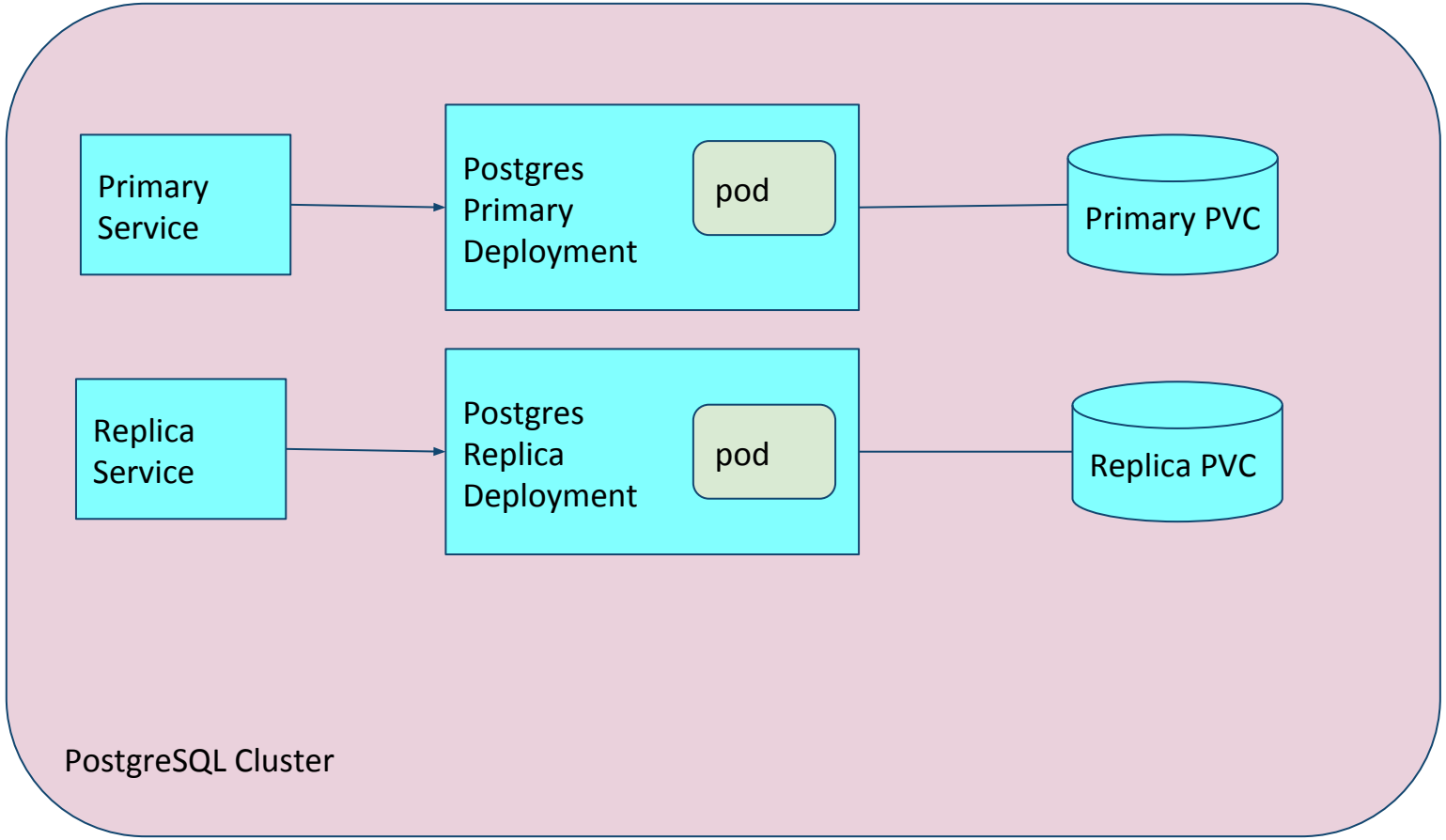
- Industry leader in providing enterprise PostgreSQL support and open source solutions
- Commitment to Community & Open Source
- Crunchy Certified PostgreSQL
  - 100% Open Source PostgreSQL
  - Emphasis in Data Security and Compliance
  - Common Criteria EAL 2+ Certified
- We're hiring!
  - https://www.crunchydata.com/
  - DBAs, Systems Engineers, Container Experts

Table of Contents

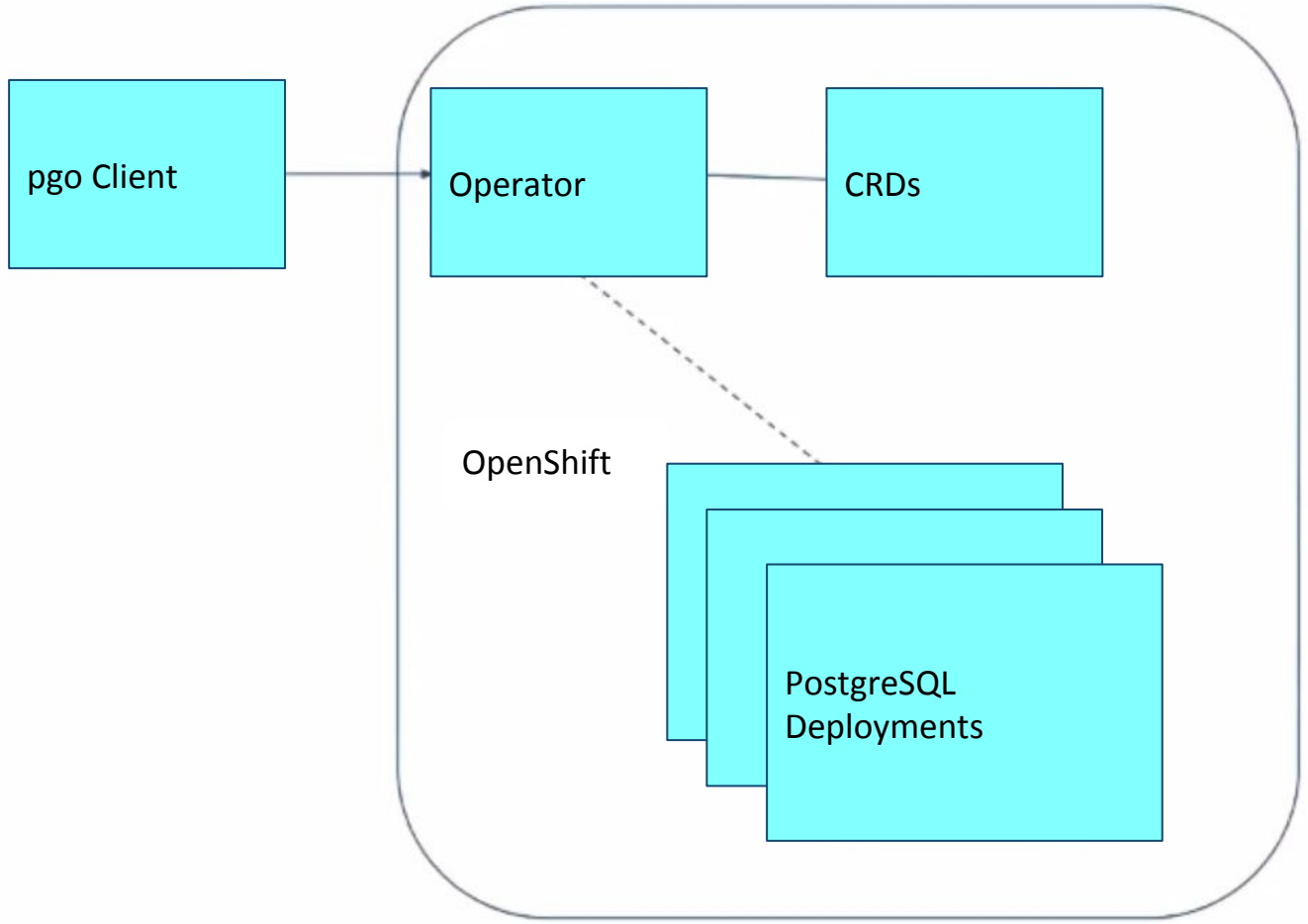
Operator Basics

- Open Source
  - https://github.com/crunchydata/postgres-operator
  - /docs/operator-docs.asciidoc
  - /docs/design.asciidoc
- Controller
- Automated
- Leverages the Kube API
  - https://github.com/kubernetes/client-go
- Command Line Interface
- Deployment
- Custom Resource Definitions

PostgreSQL Deployment



PostgreSQL Deployment



Why?

- Automation
- Standard Practices
- Ease of Use
- Large Scale Deployments
- Complex Orchestrations

Building Blocks

The Operator leverages the following containers to deploy PostgreSQL:

- **crunchy-postgres** - runs PostgreSQL 10.3 base image
- **crunchy-backup** - performs a pg_basebackup on a database container
- **crunchy-collect** - collects metrics using postgres_exporter & node_exporter
- **crunchy-upgrade** – performs minor & major PostgreSQL upgrades

Requirements

- https://github.com/CrunchyData/postgres-operator/blob/master/docs/operator-docs.asciidoc
- Kubernetes 1.7.0+
- OpenShift Origin 1.7.0+
- OpenShift Container Platform 3.7+
- Golang 1.8+
- PostgreSQL 9.6+ Container version 1.7.0 or later (e.g. centos7-10.1-1.7.0)
- PostgreSQL Backup Container version 1.7.0 or later (e.g. centos7-10.1-1.7.0)
- PostgreSQL Upgrade Container version 1.7.0 or later (e.g. centos7-10.1-1.7.0)
- PostgreSQL Metrics Collection Container version 1.7.0 or later (e.g. centos7-10.1-1.7.0)
- CentOS 7 or RHEL 7

Create a Cluster

Creating a cluster with **pgo create cluster** will create the following objects by default:

- Database pod
- PVC
- Deployment
- Service
- Replica set

```
$ pgo create cluster isengard
created Pgcluster isengard
```

```
$ pgo create cluster xraydb --series=3 --labels=project=xray
--policies=xrayapp,rlspolicy
```


Add Metrics

The **--metrics** command flag:

- adds crunchy-collect to the pod
- enables metrics collection on PostgreSQL

https://github.com/crunchydata/crunchy-containers/blob/master/docs/install.adoc

- crunchy-grafana
- crunchy-promgateway
- crunchy-prometheus

https://github.com/crunchydata/crunchy-containers/blob/master/docs/metrics.adoc

$COROOT/conf/postgres-operator/cluster/1/collect.json

```
$ pgo create cluster shire --metrics
created Pgcluster shire

$ pgo show cluster shire
cluster : shire (centos7-10.1-1.7.0)
├── pod : shire-1640328826-87cks (Running on localhost) (2/2)
│   └── pvc : shire-pvc
├── deployment : shire
└── service : shire (172.30.7.37)
```


Show Clusters

The output of **pgo show cluster** will display:

- the current status of the database pod
- the IP address of the database service
- the components of the cluster

```
$ pgo show cluster isengard
cluster : isengard (centos7-10.1-1.7.0)
├── pod : isengard-2909518585-9gj7m (Running on localhost)
(1/1)
        ├── pvc : isengard-pvc
├── deployment : isengard
└── service : isengard (172.30.35.188)
```

```
$ pgo show cluster all
```

```
$ pgo show cluster all --version=10.2
```


Test a Cluster

This command will test each service defined for the cluster using the following user accounts:

- postgres
- master
- standard user accounts

This command displays:

- connection status
- equivalent psql commands
- database IP address

```
$ pgo test isengard
psql -p 5432 -h 172.30.35.188 -U primaryuser postgres is working
psql -p 5432 -h 172.30.35.188 -U postgres postgres is working
psql -p 5432 -h 172.30.35.188 -U testuser userdb is working
```


Backup a Cluster

Backing up a cluster is performed through a Kubernetes Job which creates a **pgbackups** CRD.

```
$ pgo backup isengard
pgbackup pgbackup isengard not found so we will create it
created Pgbackup isengard

$ kubectl get pgbackups
NAME      AGE
isengard   24s

$ pgo backup  --selector=project=xray
```


Viewing Backups

Showing the PVC listing information is important for restoring your backup.

The output should show:

- PVC information
- CCPImageTag
- Backup Status
- Backup Host
- User/Password
- Port

```
$ pgo show backup isengard

pgbackup : isengard
├── PVC Name:     isengard-backup-pvc
├── PVC Access Mode: ReadWriteMany
├── PVC Size:          200M
├── CCPImageTag:       centos7-10.1-1.7.0
├── Backup Status:     completed
├── Backup Host:  isengard
├── Backup User:  primaryuser
├── Backup Pass:  password
```


Viewing Backup PVCs

Showing the PVC listing information is important for restoring your backup.

```
$ pgo show pvc isengard-backup-pvc
└── /isengard-backups

$ pgo show pvc isengard-backup-pvc --pvc-root=isengard-backups
└── /2018-02-10-03-14-30
```


Restore from Backup

This command will create a new database called **mordor** based on:

- the backup found in isengard-backups/2018-02-10-03-14-30
- the secrets of the original isengard cluster

```
$ pgo create cluster mordor
--backup-path=isengard-backups/2018-02-10-03-14-30
--backup-pvc=crunchy-pvc --secret-from=isengard
created Pgcluster mordor
```

Delete a Cluster

The standard **pgo delete cluster** command will not delete the PVC containing the data files.

```
$ pgo delete cluster isengard
deleted pgcluster isengard

$ pgo delete cluster --selector=project=xray

$ pgo delete cluster isengard --delete-data

$ pgo delete cluster isengard --delete-data --delete-backups

$ pgo delete cluster isengard --delete-data --delete-backups
--no-prompt
```


Cluster Replication

The replica deployment is set by default to 0 initialized replicas. This command allows you to scale those up.

```
$ pgo scale isengard --replica-count=1
Ok

$ psql -h <ip addr> -U postgres postgres -c 'table
pg_stat_replication'
```


Cluster Replication

The replica is combined with the primary with all actions that happen to the cluster.

```
$ pgo show cluster isengard
cluster : isengard (centos7-10.1-1.7.0)
├── pod : isengard-2909518585-9gj7m (Running on localhost) (1/1)
        ├── pvc : isengard-pvc
├── pod : isengard-replica-amwh-2977300267-wlwtg (Running on localhost) (1/1)
        ├── pvc : isengard-replica-amwh-pvc
├── deployment : isengard
├── deployment : isengard-replica-amwh
├── service : isengard (172.30.35.188)
└── service : isengard-replica (172.30.80.127)
```


Cluster Replication

The replica is combined with the primary with all actions that happen to the cluster.

```
$ pgo test isengard
psql -p 5432 -h 172.30.35.188 -U primaryuser postgres is working
psql -p 5432 -h 172.30.35.188 -U postgres postgres is working
psql -p 5432 -h 172.30.35.188 -U testuser userdb is working
psql -p 5432 -h 172.30.80.127 -U primaryuser postgres is working
psql -p 5432 -h 172.30.80.127 -U postgres postgres is working
psql -p 5432 -h 172.30.80.127 -U testuser userdb is working
```


Minor Cluster Upgrade

When you run this command, it will cause the operator to:

- create a pgupgrade CRD
- delete the existing containers
- recreate them with updated Cluster.CCPImageTag from pgo.yaml

$COROOT/conf/apiserver/pgo.yaml

```
$ pgo create cluster outdated --ccp-image-tag=centos7-10.1-1.6.0
created Pgcluster outdated

$ pgo show cluster outdated
cluster : outdated (centos7-10.1-1.6.0)
...

$ pgo upgrade outdated
created Pgupgrade outdated

$ pgo show cluster outdated
cluster : outdated (centos7-10.1-1.7.0)
...
```


Major Cluster Upgrade

When you run this command, it will cause the operator to run the pg_upgrade utility and:

- create a pgupgrade CRD
- create a Kubernetes Job which runs crunchy-upgrade
- delete the existing containers
- recreate them with updated Cluster.CCPImageTag from pgo.yaml
- output database files to new PVC

```
$ pgo upgrade outdated --upgrade-type=major

$ kubectl get pgupgrade
NAME      AGE
outdated   1m

$ kubectl describe pgupgrade outdated
…
Status:
  Message:  Successfully processed Pgupgrade by controller
  State:    Processed
```

$COROOT/conf/apiserver/pgo.yaml

View Passwords

Passwords are generated if not specified in your pgo configuration.

$COROOT/deploy/create-secrets.sh

```
$ pgo show cluster isengard --show-secrets=true
...

secret : isengard-primary-secret
├── username: primaryuser
└── password: password

secret : isengard-root-secret
├── username: postgres
└── password: password

secret : isengard-user-secret
├── username: testuser
└── password: password
```


Create Policy

A policy is a SQL based series of statements that can initialize your database with specific structure or defined objects (CREATE TABLE, CREATE INDEX, etc.)

Warning:
Policies are executed as the superuser (user **postgres**) in PostgreSQL.

```
$ pgo create policy mypolicy --in-file=policy1.sql
created policy

$ pgo show policy mypolicy

policy : mypolicy
├── url :
├── status :
└── sql : create table policy1 (id int);

$ pgo create policy gitpolicy
--url=https://github.com/CrunchyData/postgres-operator/blob/master/examples/policy/gitpolicy.sql
```


Apply Policy

When you apply a policy using pgo using a selector name of **isengard**, it will:

- look up clusters with the label value of name=isengard
- apply the policy label to that cluster
- execute the policy SQL against that cluster

```
$ pgo apply mypolicy --selector=name=isengard
applied policy on isengard

$ pgo show cluster isengard
cluster : isengard (centos7-10.1-1.7.0)
├── pod : isengard-2909518585-9gj7m (Running on localhost) (1/1)
│       ├── pvc : isengard-pvc
├── deployment : isengard
├── policy: mypolicy
└── service : isengard (172.30.35.188)

$ pgo show policy all
```


User Management

Control access & authentication methods

Basic user management exists at the moment to create, delete, and manage users for specific clusters or project selectors.

Creating a new user for a specific cluster:

```
$ pgo user --add-user=saruman --selector=name=isengard
adding new user saruman to isengard

$ pgo user --add-user=sauron --valid-days=30 --managed
--db=userdb --selector=name=isengard
adding new user sauron to isengard

$ pgo show cluster isengard --show-secrets
...
secret : isengard-sauron-secret
├── username: sauron
└── password: aFC6L6N8
```


Password Management

Control access & authentication methods

You can change the password of a specific user in a cluster.

```
$ pgo show cluster isengard --show-secrets
...
secret : isengard-sauron-secret
├── username: sauron
└── password: aFC6L6N8

$ pgo user --change-password=sauron --selector=name=isengard
changing password of user sauron on isengard

$  pgo show cluster isengard --show-secrets
...
secret : isengard-sauron-secret
├── username: sauron
└── password: p6zkVeez
```


Password Management

Control access & authentication methods

The ability is present to impose a maximum password age for a user.

Extend a current user's maximum password age:

$ pgo user --change-password=user0 --valid-days=10 --selector=name=xraydb1

To see user passwords that have expired past a certain number of days:

$ pgo user --expired=14 --selector=name=isengard
RoleName saruman Role Valid Until 2018-02-20T00:00:00-05:00

To update expired passwords in a cluster:

$ pgo user --update-passwords --selector=name=mycluster

Label Management

Label management allows the user to define metadata for any set of **pgcluster** objects.

```
$ pgo label --label=project=test --selector=name=mycluster1
adding label to mycluster1

$ pgo label --label=project=test --selector=name=mycluster2
adding label to mycluster2

$ pgo show cluster all --selector=project=test
cluster : mycluster1 (centos7-10.1-1.7.0)
├── pod : mycluster1-1973324183-sf7dn (Running on localhost)
(1/1)
...
cluster : mycluster2 (centos7-10.1-1.7.0)
├── pod : mycluster2-2500445412-8fjtr (Running on localhost)
(1/1)
...
```


pgo.yaml

```
Namespace: demo
Cluster:
  CCPImageTag: centos7-10.1-1.7.1
  Port: 5432
  User: testuser
  Database: userdb
  PasswordAgeDays: 60
  PasswordLength: 8
  Strategy: 1
  Replicas: 0
PrimaryStorage: storage1
BackupStorage: storage1
ReplicaStorage: storage1
```

```
Storage:
 storage1:
  AccessMode: ReadWriteMany
  Size: 200M
  StorageType: create
 storage2:
  AccessMode: ReadWriteMany
  Size: 333M
  StorageType: create
 storage3:
  AccessMode: ReadWriteMany
  Size: 440M
  StorageType: create
```

```
Pgo:
  Audit: false
  Metrics: false
  LSPVCTemplate:
/config/pgo.lspvc-template.json
  CSVLoadTemplate:
/config/pgo.load-template.json
  COImagePrefix: crunchydata
  COImageTag: centos7-2.5
  Debug: true
```

$COROOT/conf/apiserver/pgo.yaml

Roadmap - Where we've been

Where we've last been, where we're going - the highlights

Recent updates -

- v2.4:
  - Data purge feature
  - Metrics support
- v2.5:
  - Storage configuration features

https://github.com/CrunchyData/postgres-operator/releases

Roadmap - Where we're going

Where we've last been, where we're going - the highlights

Upcoming plans -

- Advanced PostgreSQL administration, automation, scalability
- Multi-namespace/multi-cluster operations
- Web User interface
- Disaster Recovery Features
- RBAC security model for end user access

https://github.com/CrunchyData/postgres-operator/releases

THANK YOU!

Learn more:

- https://github.com/CrunchyData/postgres-operator
- https://www.crunchydata.com/

Slides are available at:

- http://sarahconway.com/slides/scale2018-postgres-operator.pdf

Sarah Conway: sarah.conway@crunchydata.com
Jeff McCormick: jeff.mccormick@crunchydata.com