Install and use Crunchy PostgreSQL for OpenShift operator for simple todo app on OpenShift

1. Install `Crunchy PostgreSQL for OpenShift operator`
2. Install PostgreSQL Operator Monitoring
3. Install the `pgo` client
4. We are going to create a cluster named `hippo`
    - 1 Pgcluster
    - 1 bouncer (part of Pgcluster)
    - enable monitoring (part of Pgcluster)
    - enable tls (not force)
    - 1 Pgreplica
5. Deploy a simple todo application (use our cluster)
6. See Monitroring of the cluster

OpenShift

Start OpenShift

```
crc start
```

Login to OpenShift via oc

```
oc login -u kubeadmin -p <password> https://api.crc.testing:6443
```

Open OpenShift in defaul browser

```
crc console
```

Installation of Crunchy PostgreSQL for OpenShift operator

Create the namespace `pgo` before installing the `Crunchy PostgreSQL for OpenShift operator`.

```
oc create namespace pgo
```

We will install the `Crunchy PostgreSQL for OpenShift` operator via OperatorHub on OpenShift.

Install PostgreSQL Operator Monitoring

Table 1. Default Service ports.

Service	Port
Grafana	3000
Prometheus	9090
Alertmanager	9093

Install postgres-operator-metrics from local version (disable_fsgroup=true)

```
oc apply -f postgres-operator-metrics.yml
```

Install postgres-operator-metrics directly from github (issues with scc on OpenShift)

```
oc apply -f https://raw.githubusercontent.com/CrunchyData/postgres-
operator/v4.6.2/installers/metrics/kubectl/postgres-operator-metrics.yml
```

```
grafana_admin_password: "admin"
grafana_admin_username: "admin"
```

Installation of the pgo client

Install the pgo Client

```
#!/bin/bash

curl https://raw.githubusercontent.com/CrunchyData/postgres-
operator/v4.6.2/deploy/install-bootstrap-creds.sh > install-bootstrap-creds.sh
curl https://raw.githubusercontent.com/CrunchyData/postgres-
operator/v4.6.2/installers/kubectl/client-setup.sh > client-setup.sh

chmod +x install-bootstrap-creds.sh client-setup.sh

echo "Create user ..."
PGO_CMD=oc ./install-bootstrap-creds.sh

echo "Setup pgp with user ..."
PGO_CMD=oc ./client-setup.sh
```

Add this to `~/.bashr` or something like it

```
export PGOUSER=$HOME/.pgo/$PGO_OPERATOR_NAMESPACE/pgouser
export PGO_CA_CERT=$HOME/.pgo/$PGO_OPERATOR_NAMESPACE/client.crt
export PGO_CLIENT_CERT=$HOME/.pgo/$PGO_OPERATOR_NAMESPACE/client.crt
export PGO_CLIENT_KEY=$HOME/.pgo/$PGO_OPERATOR_NAMESPACE/client.key
```

Add pgo to path

```
export PATH="$HOME/.pgo/$PGO_OPERATOR_NAMESPACE:$PATH"
```

or add an alias

```
alias pgo=$HOME/.pgo/pgo/pgo
```

The client needs to be able to reach the PostgreSQL Operator API from outside the OpenShift cluster. Create an external service or forward a port locally.

```
oc -n pgo expose deployment postgres-operator
oc -n pgo create route passthrough postgres-operator --service=postgres-operator
```

For the pgo client to be able to access the api

```
oc -n pgo port-forward svc/postgres-operator 8443:8443
```

TLS for the `hippo` cluster

Create ca.crt, server.crt, server.key via script

```
./createTlsFiles.sh
```

Add `secret postgresql-ca` and `hippo-tls-keypair` to the namespace `pgo` .

Create postgresql-ca

```
kubectl create secret generic postgresql-ca -n pgo --from-file=ca.crt=ca.crt
```

Create {{ item.cluster_name }}-tls-keypair

```
kubectl create secret tls hippo-tls-keypair -n pgo --cert=server.crt --key=server.key
```

Create the `hippo` cluster

1 Pgcluster

- 1 bouncer (part of Pgcluster)
- enable monitoring (part of Pgcluster)
- enable tls (not force)

Create Pgcluster

```
oc apply -f hippo-pgcluster.yaml
```

Create a managed for our todo application

```
pgo create user hippo \
  --username=micbn --password=SuperSecret1 --managed
```

`pgBouncer` is enabled by updating the `pgBouncer` node in hippo-pgcluster.yaml.

```
pgBouncer:
  limits: null
  replicas: 1
  resources: null
  serviceType: ""
  tlsSecret: {{ item.cluster_name }}-tls-keypair
```

Monitriong is enabled by adding `exporter: true` to hippo-pgcluster.yaml.

Create Pgreplica

```
oc apply -f hippo-pgclusterreplicas-rpl1.yaml
```

Test the `hippo` cluster

```
pgo test -n pgo hippo
```

Do we have access to pgBouncer

Port forward to the cluster hippo

```
oc -n pgo port-forward svc/hippo-pgbouncer 5432:5432
```

Port forward to the cluster hippo

```
PGPASSWORD=SuperSecret1 psql -h localhost -p 5432 -U micbn hippo
```

Result is something like this if tls is present

```
psql (13.2)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
compression: off)
Type "help" for help.

hippo=>
```

User & Roles

It is possible to create users as `managed` and `not managed`.

- The `managed` have the username/password in `secrets` like `hippo-primaryuser-secret`
- The `not manage` have username/password only in PostgreSQL

Get a list of user/password for a cluster via the pgo client

```
pgo show user hippo --show-system-accounts
```

WARNING	`pgo show user hippo --show-system-accounts` will only show password for `managed` users.

Create a managed for our todo application

```
pgo create user hippo --username=micbn --password=SuperSecret1 --managed
```

ToDo app

We have a nice small app to test connection to a cluster.

todo-app/README.adoc

See monitroring of the cluster

Create route for Prometheus

```
oc -n pgo create route passthrough crunchy-prometheus --service=crunchy-prometheus
```

Port forward to Prometheus

```
oc -n pgo port-forward svc/crunchy-prometheus 9090:9090
```

Create route for Grafana

```
oc -n pgo create route passthrough crunchy-grafana --service=crunchy-grafana
```

Port forward to Grafana

```
oc -n pgo port-forward svc/crunchy-grafana 3000:3000
```

Create route for Alertmanager

```
oc -n pgo create route passthrough crunchy-alertmanager --service=crunchy-alertmanager
```

Port forward to Alertmanager

```
oc -n pgo port-forward svc/crunchy-alertmanager 9093:9093
```

Table 2. Default Service ports.

Service	Port
Grafana	3000
Prometheus	9090
Alertmanager	9093

Links

- Red Hat CodeReady Containers
- Crunchy Data
- Crunchy PostgreSQL Operator