Installation Guide - Crunchy Containers for PostgreSQL

Crunchy Data Solutions, Inc.

Version 1.4.1, 2017-06-22

Table of Contents

Project Setup & Docker Installation . . . 1
- Assumptions . . . 1
- Step 1 - Project Directory Structure . . . 1
- Step 2 - Install the Host Dependencies . . . 2
- Step 3 - Build the Containers . . . 3

OpenShift Environment . . . 3
- Installation . . . 3
- Configure NFS for Persistence . . . 4
- Examples . . . 5

Kubernetes Environment . . . 6
- Installation . . . 6
- Examples . . . 6
- Tips . . . 6

Legal Notices . . . 6

Project Setup & Docker Installation

The crunchy-containers can run on different environments including:

- Standalone Docker
- OpenShift Enterprise
- Kubernetes 1.2.X

In this document we list the basic installation steps required for these environments.

Assumptions

The install assumes the following about your host configuration:

- Centos 7 or RHEL 7 VM
- user called **someuser** has been created
- **someuser** has sudo privileges with no password

Step 1 - Project Directory Structure

First add the following lines to your .bashrc file to set the project paths:

```
export GOPATH=$HOME/cdev
export GOBIN=$GOPATH/bin
export PATH=$PATH:$GOBIN
export CCP_BASEOS=centos7
export CCP_PGVERSION=9.6
export CCP_VERSION=1.4.1
export CCP_IMAGE_TAG=$CCP_BASEOS-$CCP_PGVERSION-$CCP_VERSION
export CCPROOT=$GOPATH/src/github.com/crunchydata/crunchy-containers
```

You will then need to log out and back in for the changes to your .bashrc file to take effect.

Next, set up a project directory structure and pull down the project:

```
mkdir $HOME/cdev $HOME/cdev/src $HOME/cdev/pkg $HOME/cdev/bin
cd $GOPATH
sudo yum -y install golang git docker postgresql
go get github.com/tools/godep
cd src/github.com
mkdir crunchydata
cd crunchydata
git clone https://github.com/crunchydata/crunchy-containers
cd crunchy-containers
git checkout 1.4.1
godep restore
```

If you are a Crunchy enterprise customer, you will place the CRUNCHY repo key and yum repo file into the $CCPROOT/conf directory at this point. These files can be obtained through https://access.crunchydata.com/ on the downloads page.

Step 2 - Install the Host Dependencies

Next, install system dependencies:

```
sudo yum -y update
sudo groupadd docker
sudo usermod -a -G docker someuser
```

Remember to log out of the **someuser** account for the Docker group to be added to your current session. Once it's added, you'll be able to run Docker commands from your user account.

```
su - someuser
```

You can ensure your **someuser** account is added correctly by running the following command and ensuring **docker** appears as one of the results:

```
groups
```

Before you start Docker, you might consider configuring Docker storage: This is described if you run:

```
man docker-storage-setup
```

Follow the instructions available on the main OpenShift documentation page to configure Docker storage appropriately.

My typical process for setting up Docker storage is as follows:

- add an extra IDE drive to my VM
- fdisk /dev/sd? to format the drive
- vgcreate /dev/sd?1 to create a volume group on the new drive partition
- add VG=docker-vg to /etc/sysconfig/docker-storage-setup
- run docker-storage-setup to use that new volume group

Next, we enable and start up Docker:

```
sudo systemctl enable docker.service
sudo systemctl start docker.service
```

Step 3 - Build the Containers

At this point, you have a decision to make - either download prebuilt containers from Dockerhub, **or** build the containers on your local host.

To download the prebuilt containers, make sure you can login to Dockerhub, and then run the following:

```
docker login
cd $CCPROOT
./bin/pull-from-dockerhub.sh
```

Or if you'd rather build the containers from source, perform a container build as follows:

```
cd $CCPROOT
make setup
make all
```

After this, you will have all the Crunchy containers built and are ready for use in a **standalone Docker** environment.

OpenShift Environment

Installation

See the OSE installation guide for details on how to install OSE on your host. The main instructions are here:

https://docs.openshift.com/enterprise/3.2/install_config/install/index.html

Or, if you'd prefer to install OpenShift Origin, the easiest way to get OpenShift Origin up and running is found here: https://github.com/openshift/origin/blob/master/docs/cluster_up_down.md

For examples and tips on how to run OpenShift Enterprise & Origin, please look at the examples.adoc documentation.

Configure NFS for Persistence

NFS is required for some of the OpenShift examples, those dealing with backups and restores will require a working NFS for example.

First, if you are running your NFS system with SELinux in enforcing mode, you will need to run the following command to allow NFS write permissions:

```
sudo setsebool -P virt_use_nfs 1
```

Next, you will need to set the permissions of your NFS path so that your pods can have write access. For the Crunchy examples, the **nfsnobody** GUI was chosen as an example. Pods will reference the **nfsnobody** GID (65534) as a security context **supplementalGroup** attribute. This setting will allow the pod to have group permissions of 65534 and therefore be able to write to the NFS persistent volumes.

The permissions on the NFS path are set as follows:

```
drwxrwx---.   3 nfsnobody nfsnobody   23 Dec 16 11:28 nfsfileshare
```

Most of the Crunchy containers run as the postgres UID (26), but you will notice that when **supplementalGroups** are specified, the pod will include the nfsnobody group in the list of groups for the pod user.

The case of Amazon file systems is different, for that you use the **fsGroup** security context setting but the idea for allowing write permissions is the same.

Here are the instructions I use when setting up NFS:

http://www.itzgeek.com/how-tos/linux/centos-how-tos/how-to-setup-nfs-server-on-centos-7-rhel-7-fedora-22.html

OpenShift NFS examples can be found here:

https://github.com/openshift/origin/tree/master/examples/wordpress/nfs

The examples specify a test NFS server running at IP address 192.168.0.103.

On that server, the /etc/exports file looks like this:

```
/nfsfileshare *(rw,sync)
```

Test your NFS configuration out by mounting a local directory:

```
mount 192.168.0.114:/nfsfileshare /mnt/nfsfileshare
```

if you are running your client on a VM, you will need to add 'insecure' to the exportfs file on the NFS server, this is because of the way port translation is done between the VM host and the VM instance.

see this for more details:

http://serverfault.com/questions/107546/mount-nfs-access-denied-by-server-while-mounting

Examples

For running the examples that require persistent volumes, you will need to run the following script:

```
cd $CCPROOT/examples/pv
./create-pv.sh
./create-pvc.sh
```

If you are wanting to run the examples on a Minishift instance you will need to create the PVs using hostPath as follows:

```
oc login -u system:admin
./create-pv.sh hostpath
oc login -u developer
./create-pvc.sh
```

Additional steps are required to allow persistence to work on minishift including:

```
oc login -u system:admin
oc edit scc restricted
```

Above, you will change runAsUser.Type strategy to RunAsAny.

On the boot2docker instance running Minishift, you will need to set the host path permissions as follows:

```
chmod 777 /mnt/sda1/data
```

The NAMESPACE environment variable is set to indicate which OpenShift project you want various example objects to use. This variable is set to **default** within the **examples/envvars.sh** script. Set this to match your project configuration.

See here to view the documentation showing various examples.

Kubernetes Environment

Installation

I recommend using kubeadm or minikube to try the examples out.

See the following links for installation instructions:

- https://github.com/Kubernetes/minikube
- http://linoxide.com/containers/setup-kubernetes-kubeadm-centos/
- https://kubernetes.io/docs/getting-started-guides/kubeadm/

Examples

The namespace is set for the examples within the **examples/envvars.sh** script and defaults to **default**. Set this variable according to your project configuration.

Note, some of the examples assume an NFS file system for creating persistent volumes. See above for details on setting NFS permissions and the use of **supplementalGroups** within pod specs.

Visit the examples documentation for different use cases and examples.

Tips

Make sure your hostname resolves to a single IP address in your /etc/hosts file! If not, the NFS examples will not work.

You should see a single IP address returned from this command:

```
hostname --ip-address
```

```
sudo PATH=$PATH ALLOW_PRIVILEGED=true ./hack/local-up-cluster.sh
```

Note: specifying ALLOW_PRIVILEGED=true is required if you are running in SELinux enforcing mode. This allows you to specify the following in your pod spec to run the container as privileged:

```
"securityContext": {
    "privileged": true
},
```

Legal Notices