Crunchy Containers

Installation Guide

DRAFT

Crunchy Data Solutions, Inc.

March 22, 2018

Table of Contents

1 Project Setup

The crunchy-containers can run on different environments including:

- **Docker 1.12**
- **OpenShift Container Platform 3.6**
- **Kubernetes 1.8+**

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

These installation instructions are developed and tested for the following operating systems:

- **CentOS 7**
- **RHEL 7**

1.1 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=10
export CCP_PG_FULLVERSION=10.3
export CCP_VERSION=1.8.1
export CCP_IMAGE_PREFIX=crunchydata
export CCP_IMAGE_TAG=$CCP_BASEOS-$CCP_PG_FULLVERSION-$CCP_VERSION
export CCPROOT=$GOPATH/src/github.com/crunchydata/crunchy-containers
export CCP_CLI=kubectl
export CCP_NAMESPACE=demo
export PV_PATH=/mnt/nfsfileshare
export LOCAL_IP=$(hostname --ip-address)
export REPLACE_CCP_IMAGE_PREFIX=crunchydata
```

It will be necessary to refresh your bashrc file in order for the changes to take effect.

```
. ~/.bashrc
```

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

```
mkdir $HOME/cdev $HOME/cdev/src $HOME/cdev/pkg $HOME/cdev/bin
```

At this point, if you are installing crunchy-containers on a CentOS 7 machine, you may continue with the following instructions. If you are doing an installation on RHEL 7, please view the instructions located BELOW that are specific to RHEL environments.

1.1.1 CentOS 7

```
cd $GOPATH
sudo yum -y install golang git docker
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 1.8.1
go get github.com/blang/expenv
```

If you are a Crunchy enterprise customer, you will place the Crunchy repository key and yum repository file into the $CCPROOT/conf directory at this point. These files can be obtained through HTTPS://ACCESS.CRUNCHYDATA.COM/ on the downloads page.

1.1.2 RHEL 7

When setting up the environment on RHEL 7, there are slightly different steps that need to be taken.

```
cd $GOPATH
sudo subscription-manager repos --enable=rhel-7-server-optional-rpms
sudo yum-config-manager --enable rhel-7-server-extras-rpms
sudo yum -y install git golang
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 1.8.1
go get github.com/blang/expenv
```

If you are a Crunchy enterprise customer, you will place the Crunchy repository key and yum repository file into the $CCPROOT/conf directory at this point. These files can be obtained through HTTPS://ACCESS.CRUNCHYDATA.COM/ on the downloads page.

1.2 Creating a Demo Namespace on Kubernetes

This section will illustrate how to set up a new Kubernetes namespace called **demo**, and will then show how to provide permissions to that namespace to allow the Kubernetes examples to run within that namespace. This is a **requirement** for running the examples.

First, view currently existing namespaces:

```
$ kubectl get namespace
NAME       STATUS  AGE
default    Active  21d
kube-public Active 21d
kube-system Active 21d
```

Then, create a new namespace called **demo**:

```
$ kubectl create -f $CCPROOT/conf/demo-namespace.json
namespace "demo" created
$ kubectl get namespace demo
NAME    STATUS  AGE
demo    Active  7s
```

Then we'll set the namespace as our current location:

```
$ kubectl config set-context $(kubectl config current-context) --namespace=demo
```

We can verify that the namespace was set correctly through the following command:

```
$ kubectl config view | grep namespace:
   namespace: demo
```

1.3 Creating a Demo Namespace on OpenShift

This section will illustrate how to set up a new OpenShift project called **demo**. This is a **requirement** for running the examples.

First we will create a new project:

```
$ oc new-project demo --description="Crunchy Containers project" --display-name="
   Crunchy-Containers"
Now using project "demo" on server "https://127.0.0.1:8443".

You can add applications to this project with the 'new-app' command. For example,
   try:
```

```
   oc new-app centos/ruby-22-centos7~https://github.com/openshift/ruby-ex.git

to build a new example application in Ruby.
```

If we view the list of projects, we can see the new project has been added and is “active”.

```
$ oc get projects
NAME      DISPLAY NAME    STATUS
demo      Crunchy-Containers Active
myproject My Project      Active
```

If you were on a different project and wanted to switch to the demo project, you would do so by running the following:

```
$ oc project demo
Now using project "demo" on server "https://127.0.0.1:8443".
```

1.4 Installing PostgreSQL

These installation instructions assume the installation of PostgreSQL 10 through the official PGDG repository. View the documentation located HERE in order to view more detailed notes or install a different version of PostgreSQL.

Locate and edit your distributions .repo file, located:

- On CentOS: /etc/yum.repos.d/CentOS-Base.repo, [base] and [updates] sections
- On Red Hat: /etc/yum/pluginconf.d/rhnplugin.conf [main] section

To the section(s) identified above, you need to append a line (otherwise dependencies might resolve to the PostgreSQL supplied by the base repository):

```
exclude=postgresql*
```

Next, install the RPM relating to the base operating system and PostgreSQL version you wish to install. The RPMs can be found HERE.

For example, to install PostgreSQL 10 on a CentOS 7 system:

```
sudo yum -y install https://download.postgresql.org/pub/repos/yum/10/redhat/rhel
   -7-x86_64/pgdg-centos10-10-2.noarch.rpm
```

Or to install PostgreSQL 10 on a RHEL 7 system:

```
sudo yum -y install https://download.postgresql.org/pub/repos/yum/testing/10/
   redhat/rhel-7-x86_64/pgdg-redhat10-10-2.noarch.rpm
```

You'll need to update your system:

```
sudo yum -y update
```

Then, go ahead and install the PostgreSQL server package.

```
sudo yum -y install postgresql10-server.x86_64
```

2 Docker Environment

As good practice, at this point you'll update your system.

```
sudo yum -y update
```

Now we'll install Docker.

```
sudo yum -y install docker
```

After that, it's necessary to add the **docker** group and give your user access to that group (here referenced as **someuser**):

```
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.

These steps are illustrative of a typical process for setting up Docker storage. You will need to run these commands as root.

First, add an extra virtual hard disk to your virtual machine (see THIS BLOG POST for tips on how to do so).

Run this command to format the drive, where /dev/sd? is the new hard drive that was added:

```
fdisk /dev/sd?
```

Next, create a volume group on the new drive partition within the fdisk utility:

```
vgcreate docker-vg /dev/sd?
```

Then, you'll need to edit the docker-storage-setup configuration file in order to override default options. Add these two lines to **/etc/sysconfig/docker-storage-setup**:

```
DEVS=/dev/sd?
VG=docker-vg
```

Finally, run the command **docker-storage-setup** to use that new volume group. The results should state that the physical volume /dev/sd? and the volume group docker-vg have both been successfully created.

Next, we enable and start up Docker:

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

Verify that Docker version 1.12.6 was installed, as per the OpenShift 3.6 REQUIREMENTS.

```
docker version
```

2.1 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:

```
godep restore
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.

3 Configure Container Storage

The Container Suite is tested on 3 different storage backends:

- hostPath (single node testing)
- NFS (single and multi-node testing)
- Gluster (dynamic storage on separate Gluster cluster)

Other storage backends work as well including GCE, EBS, ScaleIO, and others but may require you to modify various examples or configuration.

3.1 Configure HostPath

HostPath is the simplist storage backend to setup, it only is feasible on a single node but is good for testing the examples. You set up **hostPath** storage as follows:

```
sudo mkdir /data
sudo chmod 777 /data
cd $CCPROOT/examples/pv
./create-pv.sh hostpath
./create-pvc.sh
```

This set of scripts will create 15 sample Persistent Volumes that all point to /**data**. It also creates sample Persistent Volume Claims that can be shared by various examples because these Volumes are created as ReadWriteMany.

3.2 Configure NFS for Persistence

NFS is required for some of the examples, including the backup and restore containers.

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
```

Detailed instructions that you can use for setting up a NFS server on Centos 7 are provided in the following link.

HTTP://WWW.ITZGEEK.COM/HOW-TOS/LINUX/CENTOS-HOW-TOS/HOW-TO-SETUP-NFS-SERVER-ON-CENTOS-7-RHEL-7-FEDORA-22.HTML

> ***NOTE:*** *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.*

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

For more details on this bug, please see the following link.

HTTP://SERVERFAULT.COM/QUESTIONS/107546/MOUNT-NFS-ACCESS-DENIED-BY-SERVER-WHILE-MOUNTING

A suggested best practice for tuning NFS for PostgreSQL is to configure the PostgreSQL fstab mount options like so:

```
proto=tcp,suid,rw,vers=3,proto=tcp,timeo=600,retrans=2,hard,fg,rsize=8192,wsize
   =8192
```

Network options:

```
MTU=9000
```

If interested in mounting the same NFS share multiple times on the same mount point, look into the NOAC MOUNT OPTION.

Next, assuming that you are setting up NFS as your storage option, you will need to run the following script:

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

> ***NOTE:*** *If you elect to configure HostPath or GCE as your storage option, please view README.txt for command-line usage for the ./create-pv.sh command.*

3.3 Configure Gluster for Persistence

Setting up a Gluster cluster will offer you the ability to use dynamic storage provisioning in the examples. A set of example Gluster configuration files is found at $CCPROOT/docs/gluster.

This configuration is for a 3 node Gluster cluster which runs on a Centos7 Minimal VM deployment. See HTTPS://GITHUB.COM/CRUNCHYDATA/CRUNCHY-CONTAINERS/BLOB/MASTER/DOCS/GLUSTER/GLUSTER-SETUP.ADOC

4 OpenShift Environment

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

HTTPS://DOCS.OPENSHIFT.COM/CONTAINER-PLATFORM/3.6/INSTALL_CONFIG/INSTALL/QUICK_INSTALL.HTML

> ***NOTE:*** *If you install OpenShift Enterprise on a server with less than* `16GB` *memory and* `40GB` *of disk, the following Ansible variables need to be added to* `/.config/openshift/installer.cfg.yml` *prior to installation:*

```
openshift_check_min_host_disk_gb: '10' # min 10gb disk
openshift_check_min_host_memory_gb: '3' # min 3gb memory
```

5 Kubernetes Environment

See KUBEADM for installing the latest version of Kubernetes.

5.1 Helm (optional)

Some Kubernetes Helm examples are provided in the following directory as one option for deploying the Container Suite.

```
$CCPROOT/examples/helm/
```

Once you have your Kubernetes environment configured, it is simple to get Helm up and running. Please refer to THIS DOCUMENT to get Helm installed and configured properly.

5.2 DNS

Please see HERE to view the official documentation regarding configuring DNS for your Kubernetes cluster.

5.3 Google Cloud Environment

The PostgreSQL Container Suite was tested on Google Container Engine.

Here is a link to set up a Kube cluster on GCE: HTTPS://KUBERNETES.IO/DOCS/GETTING-STARTED-GUIDES/GCE

Setup the persistent disks using GCE disks by first editing your **bashrc** file and export the GCE settings to match your GCE environment.

```
export GCE_DISK_ZONE=us-central1-a
export GCE_DISK_NAME=gce-disk-crunchy
export GCE_DISK_SIZE=4
export GCE_FS_FORMAT=ext4
```

Then create the PVs used by the examples, passing in the **gce** value as a parameter. This will cause the GCE disks to be created:

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

Here is a link that describes more information on GCE persistent disk: HTTPS://CLOUD.GOOGLE.COM/CONTAINE
ENGINE/DOCS/TUTORIALS/PERSISTENT-DISK/

To have the persistent disk examples work, you will need to specify a **fsGroup** setting in the **Security-Context** of each pod script as follows:

```
"securityContext": {
      "fsGroup": 26
      },
```

For our PostgreSQL container, a UID of 26 is specified as the user which corresponds to the **fsGroup** value.

5.4 Tips

Make sure your hostname resolves to a single IP address in your /etc/hosts file. The NFS examples will not work otherwise and other problems with installation can occur unless you have a resolving hostname.

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

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

6 Legal Notices