PNDA Helm Deployment on Rancher based K8S Cluster

Rancher based Kubernetes Cluster Set-up on OpenStack VMs with NFS Storage class

Introduction

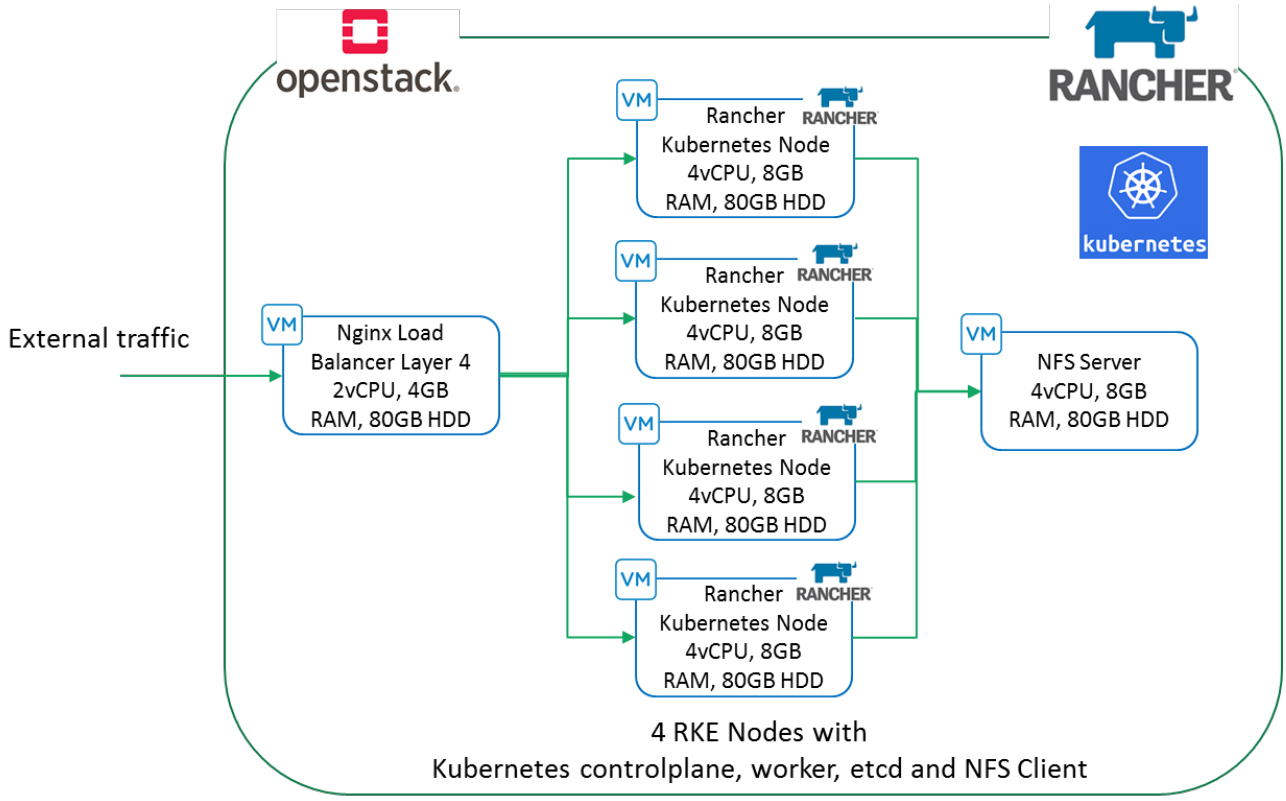
This document describes how you can achieve PNDA Helm Deployment on Rancher based Kubernetes cluster over OpenStack based (Kilo) VMs.

There are two parts to this document,

1. Rancher based K8S cluster set-up on OpenStack (along with NFS as Storage class)
2. Helm-based PNDA deployment on K8S cluster

Rancher based K8S cluster set-up on OpenStack

In the first part, we need to set-up a Rancher based Kubernetes cluster with the necessary software components. Following is the high-level depiction of Rancher Kubernetes cluster components using OpenStack VMs,



Cluster Components and Versions

All Rancher Kubernetes cluster components are deployed across 6 OpenStack VMs. Resource configuration for each OpenStack VM is shown in the diagram.

Following are the versions of various components that can be used,

Software	Version
CentOS	7.6
Docker	19.03.1
Rancher	2.2.7
Kubernetes	1.14.5
Helm	2.14.3

Tiller	2.14.3

Docker Installation

Install Docker on all the Rancher Kubernetes Engine nodes including Load Balancer node. Execute the following commands on all the nodes. Following commands assume “CentoOS” as underlying OS, but you can also use Ubuntu.

$ yum install -y yum-utils device-mapper-persistent-data lvm2

$ yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

$yum install docker-ce docker-ce-cli containerd.io

$ systemctl start docker

$ systemctl enable docker

$ usermod -aG docker centos

Following are the cluster components that need to be installed on the respective node or VMs as depicted in the above diagram,

- NGINX Layer 4 (TCP) Load Balancer
- Rancher Kubernetes Engine Nodes
- Helm and Tiller
- Rancher Server
- Network File System (NFS)

NGINX Layer 4 (TCP) Load Balancer

NGINX is configured as Layer 4 load balancer (TCP) that forwards connections to one of the Rancher nodes. The important point here is not to use one of the Rancher nodes as the load balancer. Following steps are executed on the Nginx node or VM.

Step 1: Create a repo file as below -

```
[root@mohseen01 rancher]# vi /etc/yum.repos.d/nginx.repo
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=0
enabled=1
```

Step 2: List the repo with the below command to check for any errors -

```
[root@mohseen01 rancher]# vi /etc/yum.repos.d/nginx.repo
[root@mohseen01 rancher]#
[root@mohseen01 rancher]#
[root@mohseen01 rancher]# yum repolist
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
base                                                                                                | 3.6 kB  00:00:00
extras                                                                                              | 3.4 kB  00:00:00
ius                                                                                                 | 1.3 kB  00:00:00
nginx                                                                                               | 2.9 kB  00:00:00
updates                                                                                             | 3.4 kB  00:00:00
(1/2): nginx/7/x86_64/primary_db                                                                    |  49 kB  00:00:00
(2/2): ius/x86_64/primary                                                                           | 163 kB  00:00:01
ius                                                                                                             829/829
repo id                                                repo name                                                 status
base/7/x86_64                                          CentOS-7 - Base                                           10,019
extras/7/x86_64                                        CentOS-7 - Extras                                            435
ius/x86_64                                             IUS for Enterprise Linux 7 - x86_64                          829
nginx/7/x86_64                                         nginx repo                                                   166
updates/7/x86_64                                       CentOS-7 - Updates                                         2,500
repolist: 13,949
```

Step 3: Install Nginx package with the below command -

```
[root@mohseen01 rancher]# yum install nginx
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
Resolving Dependencies
--> Running transaction check
---> Package nginx.x86_64 1:1.16.1-1.el7.ngx will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch                   Version
================================================================================
Installing:
 nginx                  x86_64                 1:1.16.1-1.el7.ngx

Transaction Summary
================================================================================
Install  1 Package

Total download size: 766 k
Installed size: 2.7 M
Is this ok [y/d/N]: y
Downloading packages:
nginx-1.16.1-1.el7.ngx.x86_64.rpm
Running transaction check
Running transaction test
```

Step 4: Update the configuration file "/etc/nginx/nginx.conf" with the IP addresses of your Rancher Kubernetes nodes as shown below -

```
worker_processes 4;
worker_rlimit_nofile 40000;

events {
    worker_connections 8192;
}

stream {
    upstream rancher_servers_http {
        least_conn;
        server 10.20.14.212:80 max_fails=3 fail_timeout=5s;
        server 10.20.14.238:80 max_fails=3 fail_timeout=5s;
    }
    server {
        listen     80;
        proxy_pass rancher_servers_http;
    }

    upstream rancher_servers_https {
        least_conn;
        server 10.20.14.212:443 max_fails=3 fail_timeout=5s;
        server 10.20.14.238:443 max_fails=3 fail_timeout=5s;
    }
    server {
        listen     443;
        proxy_pass rancher_servers_https;
    }
}
```

Please refer to link for any additional details - Nginx Load Balancer Set-up

Rancher Kubernetes Engine Nodes

Next activity is to install Rancher Kubernetes Engine on all the designated Kubernetes nodes. Following steps can be executed from any node or VM (**Installer node**) that has access to all the Rancher Kubernetes Engine Nodes.

Step 1:

Create the rancher-cluster.yml file -

Using the below sample file, create the rancher-cluster.yml file. Replace the IP Addresses in the nodes list with the IP address or DNS names of the nodes you created.

```
[root@mohseen01 rancher]# cat rancher-cluster.yml
nodes:
  - address: 10.20.14.212
    internal_address: 192.168.220.190
    user: centos
    role: [controlplane,worker,etcd]
  - address: 10.20.14.238
    internal_address: 192.168.220.199
    user: centos
    role: [controlplane,worker,etcd]

services:
  etcd:
    snapshot: true
    creation: 6h
    retention: 24h
[root@mohseen01 rancher]#
```

Step 2: Download and install **'rke**' utility -

Download the utility as shown in the below diagram based on your node hardware type.

```
[root@mohseen01 rancher]# wget https://github.com/rancher/rke/releases/download/v0.2.7/rke_linux-amd64
--2019-08-20 06:57:10--  https://github.com/rancher/rke/releases/download/v0.2.7/rke_linux-amd64
Resolving github.com (github.com)... 13.234.210.38
Connecting to github.com (github.com)|13.234.210.38|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/108337180/68186d00-b84b-11e9-85af-2bdb91fb7bdd?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20190820%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20190820T065710Z&X-Amz-Expires=300&X-Amz-Signature=15deeae708eedd62a0f2d7735b1cf13dc166d9a37ef21c365164b2e8c9259359&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Drke_linux-amd64&response-content-type=application%2Foctet-stream [following]
--2019-08-20 06:57:10--  https://github-production-release-asset-2e65be.s3.amazonaws.com/108337180/68186d00-b84b-11e9-85af-2bdb91fb7bdd?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20190820%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20190820T065710Z&X-Amz-Expires=300&X-Amz-Signature=15deeae708eedd62a0f2d7735b1cf13dc166d9a37ef21c365164b2e8c9259359&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Drke_linux-amd64&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.85.83
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|52.216.85.83|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 37905942 (36M) [application/octet-stream]
Saving to: 'rke_linux-amd64'

100%[=============================================================================================================>] 37,905,942  2.23MB/s   in 16s

2019-08-20 06:57:28 (2.23 MB/s) - 'rke_linux-amd64' saved [37905942/37905942]

[root@mohseen01 rancher]#
```

To install '**rke**' using the following command –

$ install rke_linux-amd64 /usr/bin/rke

Step 3: Run '**rke**' utility on the Rancher Cluster configuration.

$ rke up --config ./rancher-cluster.yml

When finished, it should end with the line: ***Finished building Kubernetes cluster successfully.***

Step 4: Install **Kubectl** utility –

For managing your Kubernetes cluster you need to use **Kubectl** utility.

$ curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl

$ install kubectl /usr/bin/kubectl

Step 5: Testing Your Cluster

'**rke**' command execution in **Step 3** above should have created a file kube_config_rancher-cluster.yml. This file has the credentials for **kubectl** and **helm** that you need to use. Update the .bash_profile file with as below -

$ export KUBECONFIG=$(pwd)/kube_config_rancher-cluster.yml

$ source .bash_profile

Test your connectivity with **kubectl** and see if all your nodes are in **Ready** state as shown below –

```
[root@mohseen01 ~]# kubectl get nodes
NAME            STATUS   ROLES                      AGE   VERSION
10.20.14.212    Ready    controlplane,etcd,worker   18h   v1.14.5
10.20.14.238    Ready    controlplane,etcd,worker   18h   v1.14.5
[root@mohseen01 ~]#
```

Please refer to the link for any additional details such as connecting to and testing your Kubernetes cluster and for checking the health of cluster Pods - Rancher Kubernetes Engine Installation

Helm and Tiller

Following steps need to be executed from any node or VM (**Installer node**) that has access to all Rancher Kubernetes Engine nodes.

Step 1: Install Helm and Tiller on the installer node

$ wget https://get.helm.sh/helm-v2.14.3-linux-amd64.tar.gz

$ tar zxvf helm-v2.14.3-linux-amd64.tar.gz

```
[root@mohseen01 rancher]# wget https://get.helm.sh/helm-v2.14.3-linux-amd64.tar.gz
--2019-08-20 09:29:15--  https://get.helm.sh/helm-v2.14.3-linux-amd64.tar.gz
Resolving get.helm.sh (get.helm.sh)... 152.199.39.106, 2606:2800:247:b40:171d:1a2f:2077:f6b
Connecting to get.helm.sh (get.helm.sh)|152.199.39.106|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26533763 (25M) [application/x-tar]
Saving to: 'helm-v2.14.3-linux-amd64.tar.gz'

100%[=============================================================================================================================>] 26,533,763  15.1MB/s   in 1.7s

2019-08-20 09:29:18 (15.1 MB/s) - 'helm-v2.14.3-linux-amd64.tar.gz' saved [26533763/26533763]

[root@mohseen01 rancher]# tar zxvf helm-v2.14.3-linux-amd64.tar.gz
linux-amd64/
linux-amd64/helm
linux-amd64/README.md
linux-amd64/LICENSE
linux-amd64/tiller
[root@mohseen01 rancher]# curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.t
xt`/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 40.9M  100 40.9M    0     0  9194k      0  0:00:04  0:00:04 --:--:-- 9510k
[root@mohseen01 rancher]# ls -lrt
total 67896
drwxr-xr-x. 2 root root       60 Jul 30 16:35 linux-amd64
-rw-r--r--. 1 root root 26533763 Jul 30 16:37 helm-v2.14.3-linux-amd64.tar.gz
-r--------. 1 root root     1675 Aug 20 09:18 mohseen.pem
-rw-r--r--. 1 root root 42985504 Aug 20 09:32 kubectl
[root@mohseen01 rancher]# install kubectl /usr/bin/kubectl
[root@mohseen01 rancher]# install linux-amd64/helm /usr/bin/helm
[root@mohseen01 rancher]# install linux-amd64/tiller /usr/bin/tiller
[root@mohseen01 rancher]#
```

$ install linux-amd64/helm /usr/bin/helm

$ install linux-amd64/tiller /usr/bin/tiller

Step 2: Copy or Install **Kubectl**, **Helm** and **Tiller** binaries on all the Rancher Kubernetes Nodes to the same location as described in the above steps.

Step 3: Finishing Tiller installation on Cluster from the Installer node –

- Create the ServiceAccount in the kube-system namespace.
- Create the ClusterRoleBinding to give the tiller account access to the cluster.
- Finally, use helm to install the tiller service

$ kubectl -n kube-system create serviceaccount tiller

$ kubectl create clusterrolebinding tiller --clusterrole=cluster-admin --serviceaccount=kube-system:tiller

$ helm init --service-account tiller

Test your Tiller installation –

```
[root@mohseen01 ~]# kubectl -n kube-system  rollout status deploy/tiller-deploy
deployment "tiller-deploy" successfully rolled out
[root@mohseen01 ~]#
```

Important Files

- rancher-cluster.yml: The RKE cluster configuration file.
- kube_config_rancher-cluster.yml: The Kubeconfig file for the cluster, this file contains credentials for full access to the cluster.

- rancher-cluster.rkestate: The Kubernetes Cluster State file, this file contains credentials for full access to the cluster.

Rancher Server

In this activity, Rancher server and UI gets installed as one of the Pods inside Kubernetes cluster.

Step 1: Add the Helm Chart Repository -

$ helm repo add rancher-stable https://releases.rancher.com/server-charts/stable

Step 2: Install cert-manager from Kubernetes Helm chart repository -

$ helm install stable/cert-manager --name cert-manager --namespace kube-system --version v0.5.2

Step 3: Wait for cert-manager to be rolled out -

$ kubectl -n kube-system rollout status deploy/cert-manager

Step 4: Install Rancher Server – Stable version

$ helm install rancher-stable/rancher --name rancher --namespace cattle-system --set hostname="your_host_name" --set ingress.tls.source=letsEncrypt --set letsEncrypt.email="your-email@domain.com"

This should print a message – **Rancher Server has been installed**. This will be followed by the Rancher Admin UI URL that you can browse.

Important Note: The hostname specified in the above command needs to be registered with your organization's DNS.

Step 5: Rancher Rollout

$ kubectl –n cattle-system rollout status deploy/rancher

```
[root@mohseen01 ~]# kubectl -n cattle-system rollout status deploy/rancher
deployment "rancher" successfully rolled out
[root@mohseen01 ~]#
```

This will roll out the Rancher deployment.

Step 6: Check Ingress

$ kubectl -n cattle-system get ingress

```
[root@mohseen01 ~]# kubectl get ingress -n cattle-system
NAME                       HOSTS              ADDRESS                      PORTS     AGE
cm-acme-http-solver-qqzsv  mohseen01.xor.in   10.20.14.212,10.20.14.238    80        19m
rancher                    mohseen01.xor.in   10.20.14.212,10.20.14.238    80, 443   20m
[root@mohseen01 ~]#
```

Step 7: Check Rancher Deployment Status

$ kubectl -n cattle-system get deploy rancher

```
[root@mohseen01 ~]# kubectl -n cattle-system get deploy rancher
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
rancher   3/3     3            3           21m
[root@mohseen01 ~]#
```

Network File System (NFS) Server

Following are general steps to get the NFS server and share created -

Step 1: Set-up a NFS Server on one of the nodes or VMs.

Step 2: Create and export a NFS Share

Step 3: Mount the exported NFS share on all the Rancher Kubernetes Nodes

Kubernetes NFS-Client Provisioner

The nfs-client is an automatic provisioner that use your existing and already configured NFS server to support dynamic provisioning of Kubernetes Persistent Volumes via Persistent Volume Claims. You can use Helm Chart to deploy the nfs-client.

$ helm install stable/nfs-client-provisioner --set nfs.server=x.x.x.x --set nfs.path=/nfsshare

For additional details source code, please refer to - https://github.com/kubernetes-incubator/external-storage/tree/master/nfs-client

For additional details please on Helm refer to - https://github.com/helm/charts/tree/master/stable/nfs-client-provisioner

Helm-based PNDA deployment on K8S cluster

This is the second part of PNDA Helm Deployment on Rancher based K8s Cluster. In this document, we will see how to configure Kubectl for cluster communication and add persistent volume claims for the required components.

Prerequisites

Before configuration of Kubectl for cluster communication, be sure your host machine meets the following prerequisites:

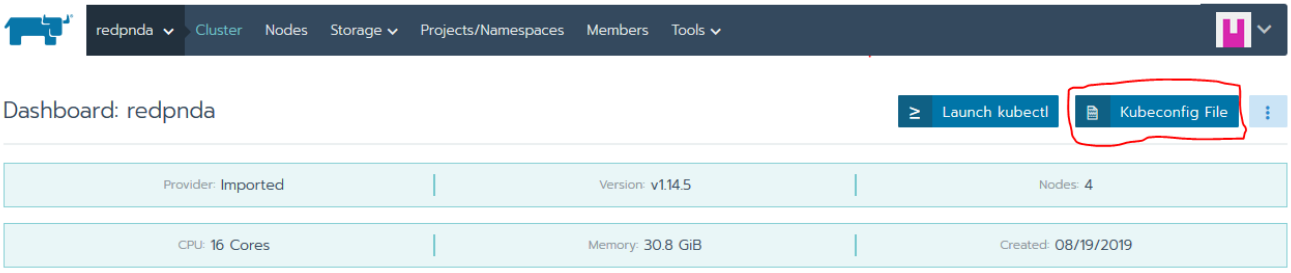
1. Installation and a functional state of Rancher Kubernetes cluster.
2. Installation of Docker and Kubectl on edge node to communicate Pods, Deployments, Services and PVCs.
3. Installation of Helm on edge node for the deployment of PNDA application on Rancher based Kubernetes cluster.

Step-by-step Kubectl Configuration for Cluster Communication

Step 1: Create a .kube directory

```
pnda@pnda:~$ mkdir .kube
```

Step 2: Get the config file from the Rancher cluster and place it under .kube directory




```
pnda@pnda:~/.kube$ ls -lrt
total 8
-rw-r--r-- 1 root root 5384 Aug 20 11:45 config
pnda@pnda:~/.kube$
```

Helm repository from GIT

Download the PNDA Helm Repo from GIT repository by using the following link,

https://github.com/pndaproject/pnda-helm-repo

Configuration of Persistent Volume Claims (PVC) for PNDA

Pods use Persistent Volume Claims (PVC) to request the platform for physical storage. You must create a PersistentVolumeClaim requesting a volume of at least three gibibytes to provide read-write access. Here, we have used NFS-client for storage. For configuring NFS-client storage, you need to modify the values.yaml for the following components,

Step 1: For Redis, HDFS Name Node, HDFS Data Node, Confluent-platform and JupyterHub you have

to modify in */pnda-helm-repo-master/pnda/values.yaml*

- **Redis**

```yaml
redis:
  enabled: true
  usePassword: false
  master:
    persistence:
      enabled: true
      storageClass: nfs-client
      size: 3Gi
  slave:
    persistence:
      enabled: true
      storageClass: nfs-client
      size: 3Gi
```

- **HDFS**

```yaml
hdfs:
  enabled: true
  # The base hadoop image to use for all components.
  image:
    repository: gradiant/hadoop-base
    tag: 2.7.7
    pullPolicy: IfNotPresent
  conf:
    hdfsSite:
      dfs.replication: 1
  nameNode:
    pdbMinAvailable: 1
  dataNode:
    replicas: 1
    pdbMinAvailable: 1
  persistence:
    nameNode:
      enabled: true
      storageClass: nfs-client
      accessMode: ReadWriteOnce
      size: 4Gi
    dataNode:
      enabled: true
      storageClass: nfs-client
      accessMode: ReadWriteOnce
      size: 4Gi
  ingress:
    nameNode:
      enabled: true
```

- **Confluent-platform**

```
confluent-platform:
  cp-zookeeper:
    servers: 1
    persistence:
      enabled: true
      storageClass: nfs-client
      dataDirSize: 5Gi
      dataDirStorageClass: nfs-client
      dataLogDirSize: 5Gi
      dataLogDirStorageClass: nfs-client
  cp-kafka:
    image: confluentinc/cp-kafka
    imageTag: 5.0.1
    brokers: 1
    configurationOverrides:
      offsets.topic.replication.factor: "1"
      default.replication.factor: "1"
    persistence:
      enabled: true
      storageClass: nfs-client
      disksPerBroker: 1
      size: 5Gi
```

- **JupyterHub**

```
jupyterhub:
  enabled: true
  hub:
    uid: 0
    fsGid: 0
    image:
      name: pnda/k8s-hub
      tag: '0.8.2'
    db:
      pvc:
        storageClassName: nfs-client
        storage: 3Gi
    extraConfig:
      00-custom-singleuser-hostname: |
         c.KubeSpawner.extra_pod_config = {
            "hostname": "jupyter-{username}",
            "subdomain": "hub-subdomain"
         }
  auth:
    type: dummy
    dummy:
```

Step 2: For Package repository, have to modify values.yaml under /pnda-helm-repo-master/pnda/package-repository/values.yaml folder

Package repository

Pnda/package-repository/values.yaml

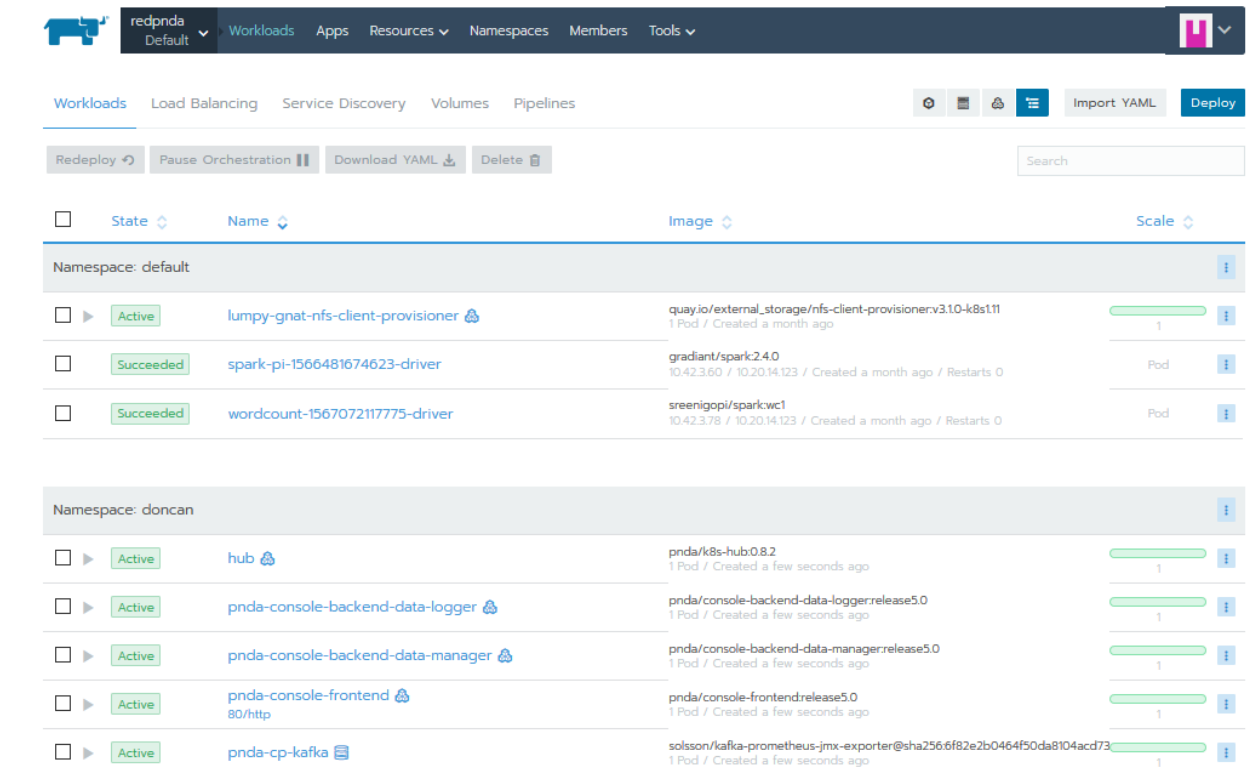
```
image:
  repository: sgopired/packrep
  tag: pr
  pullPolicy: IfNotPresent

service:
  type: ClusterIP
  port: 8888

conf:
  logLevel: INFO

resources:
  requests:
    memory: "128Mi"
    cpu: "10m"
  limits:
    memory: "1024Mi"
    cpu: "1000m"

persistence:
  enabled: true
  storageClass: nfs-client
  accessMode: ReadWriteOnce
  size: 5Gi
```

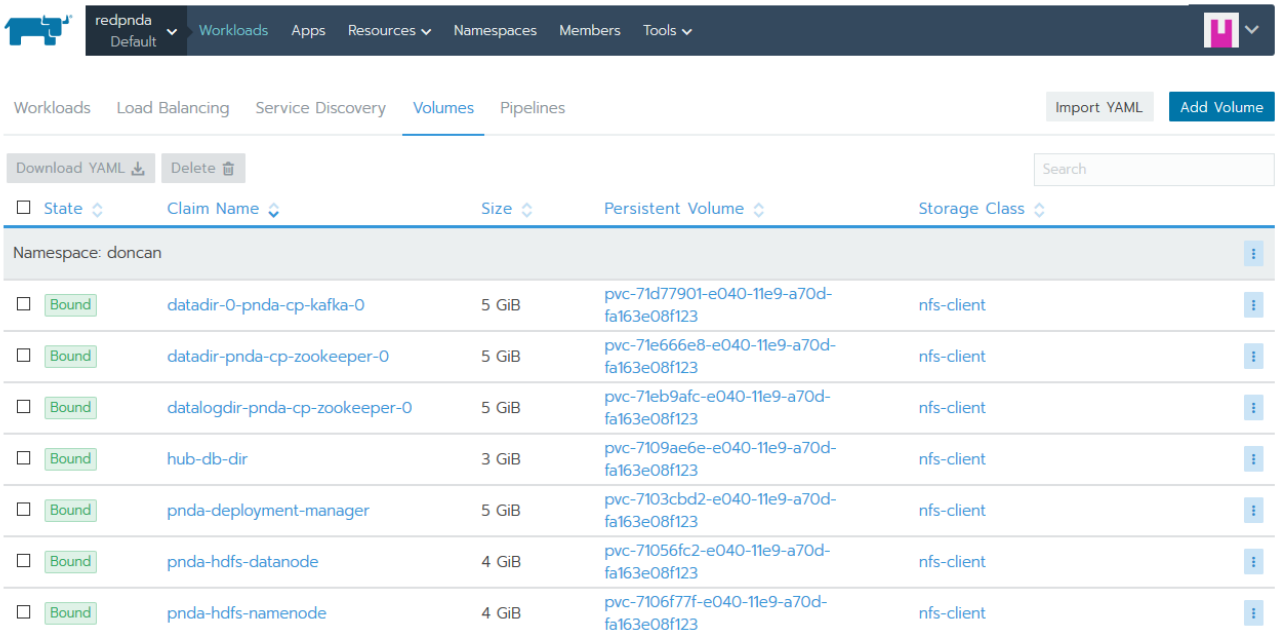
PNDA Deployment Steps

Please check the following link for PNDA deployment steps,

https://github.com/pndaproject/pnda-helm-repo

After successful PNDA deployment, you can see workloads, services and volumes under the namespace which you specify in Rancher UI, as shown below:




redpnda
Default
Workloads
Apps
Resources
Namespaces
Members
Tools
Workloads
Load Balancing
Service Discovery
Volumes
Pipelines
Import YAML
Add Volume
Download YAML
Delete
Search
State
Claim Name
Size
Persistent Volume
Storage Class
Namespace: doncan
Bound
datadir-0-pnda-cp-kafka-0
5 GiB
pvc-71d77901-e040-11e9-a70d-fa163e08f123
nfs-client
Bound
datadir-pnda-cp-zookeeper-0
5 GiB
pvc-71e666e8-e040-11e9-a70d-fa163e08f123
nfs-client
Bound
datalogdir-pnda-cp-zookeeper-0
5 GiB
pvc-71eb9afc-e040-11e9-a70d-fa163e08f123
nfs-client
Bound
hub-db-dir
3 GiB
pvc-7109ae6e-e040-11e9-a70d-fa163e08f123
nfs-client
Bound
pnda-deployment-manager
5 GiB
pvc-7103cbd2-e040-11e9-a70d-fa163e08f123
nfs-client
Bound
pnda-hdfs-datanode
4 GiB
pvc-71056fc2-e040-11e9-a70d-fa163e08f123
nfs-client
Bound
pnda-hdfs-namenode
4 GiB
pvc-7106f77f-e040-11e9-a70d-fa163e08f123
nfs-client