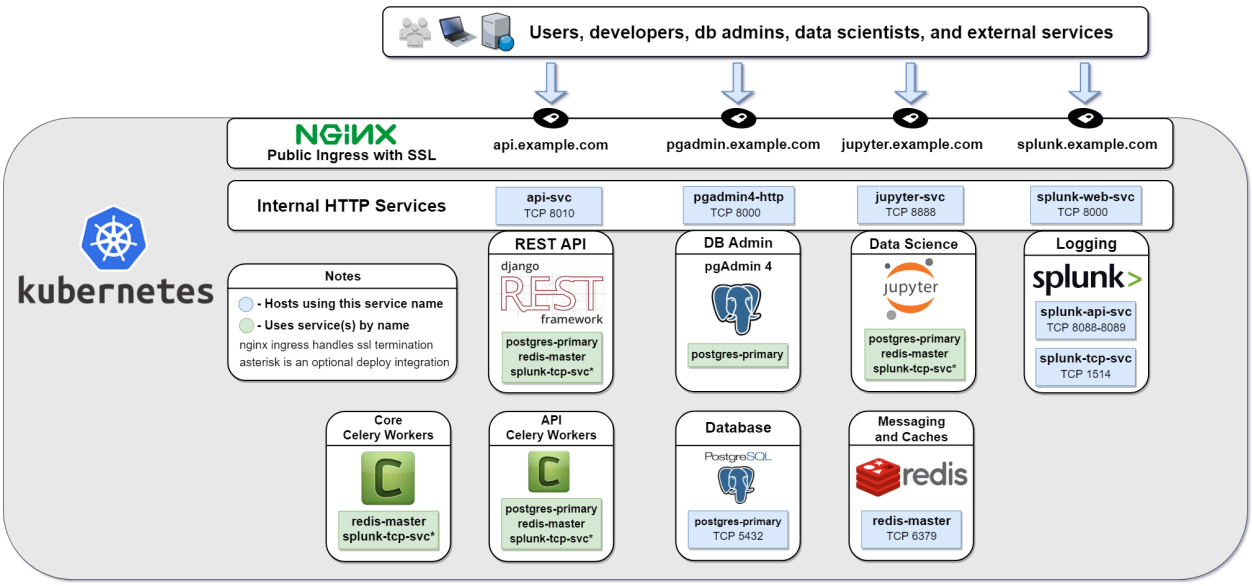
Deploy to Kuberenetes Documentation

Release 1.0.0

Jay Johnson

Mar 12, 2019

Contents



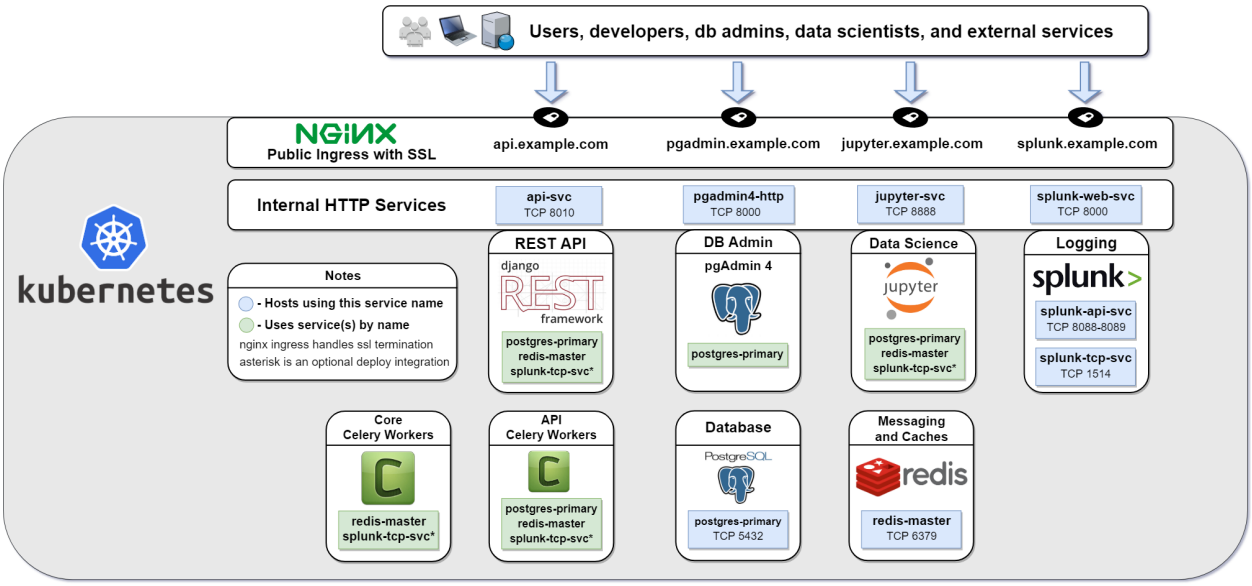
Install and manage a Kubernetes cluster (version 1.13.4) with helm on a single CentOS 7 vm or in multi-host mode that runs the cluster on 3 CentOS 7 vms. Once running, you can deploy a distributed, scalable python stack capable of delivering a resilient REST service with JWT for authentication and Swagger for development. This service uses a decoupled REST API with two distinct worker backends for routing simple database read and write tasks vs long-running tasks that can use a Redis cache and do not need a persistent database connection. This is handy for not only simple CRUD applications and use cases, but also serving a secure multi-tenant environment where multiple users manage long-running tasks like training deep neural networks that are capable of making near-realtime predictions.

This guide was built for deploying the AntiNex stack of docker containers and the Stock Analysis Engine on a Kubernetes single host or multi-host cluster.

- Managing a Multi-Host Kubernetes Cluster with an External DNS Server
- Cert Manager with Let's Encrypt SSL support
- A Native Ceph Cluster for Persistent Volume Management with KVM
- A Third-party Rook Ceph Cluster for Persistent Volumes
- Minio S3 Object Store
- Redis
- Postgres
- Django REST API with JWT and Swagger
- Django REST API Celery Workers
- Jupyter
- Core Celery Workers
- pgAdmin4
- (Optional) Splunk with TCP and HEC Service Endpoints

CHAPTER 1

Deploying a Distributed AI Stack to Kubernetes on CentOS



Install and manage a Kubernetes cluster (version 1.13.4) with helm on a single CentOS 7 vm or in multi-host mode that runs the cluster on 3 CentOS 7 vms. Once running, you can deploy a distributed, scalable python stack capable of delivering a resilient REST service with JWT for authentication and Swagger for development. This service uses a decoupled REST API with two distinct worker backends for routing simple database read and write tasks vs long-running tasks that can use a Redis cache and do not need a persistent database connection. This is handy for not only simple CRUD applications and use cases, but also serving a secure multi-tenant environment where multiple users manage long-running tasks like training deep neural networks that are capable of making near-realtime predictions.

This guide was built for deploying the AntiNex stack of docker containers and the Stock Analysis Engine on a Kubernetes single host or multi-host cluster.

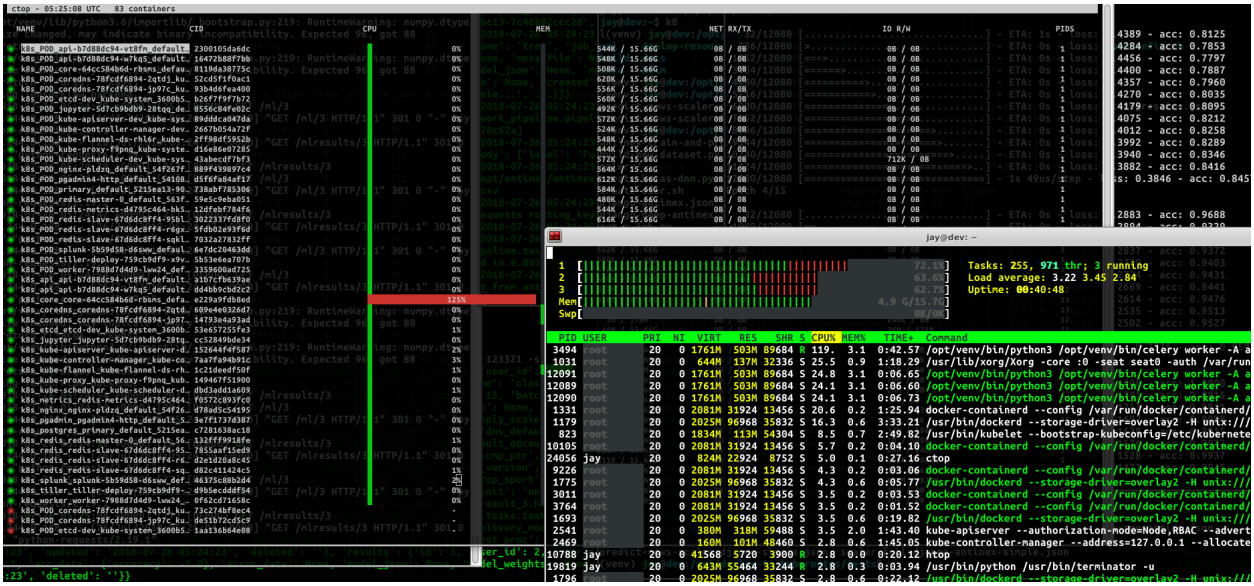
- Managing a Multi-Host Kubernetes Cluster with an External DNS Server
- Cert Manager with Let's Encrypt SSL support

- A Native Ceph Cluster for Persistent Volume Management with KVM
- A Third-party Rook Ceph Cluster for Persistent Volumes
- Minio S3 Object Store
- Redis
- Postgres
- Django REST API with JWT and Swagger
- Django REST API Celery Workers
- Jupyter
- Core Celery Workers
- pgAdmin4
- (Optional) Splunk with TCP and HEC Service Endpoints

CHAPTER 2

Getting Started

Note: Please ensure for single-vm hosting that the CentOS machine has at least 4 CPU cores and more than 8 GB ram. Here is a screenshot of the CPU utilization during AI training with only 3 cores:



2.1 Overview

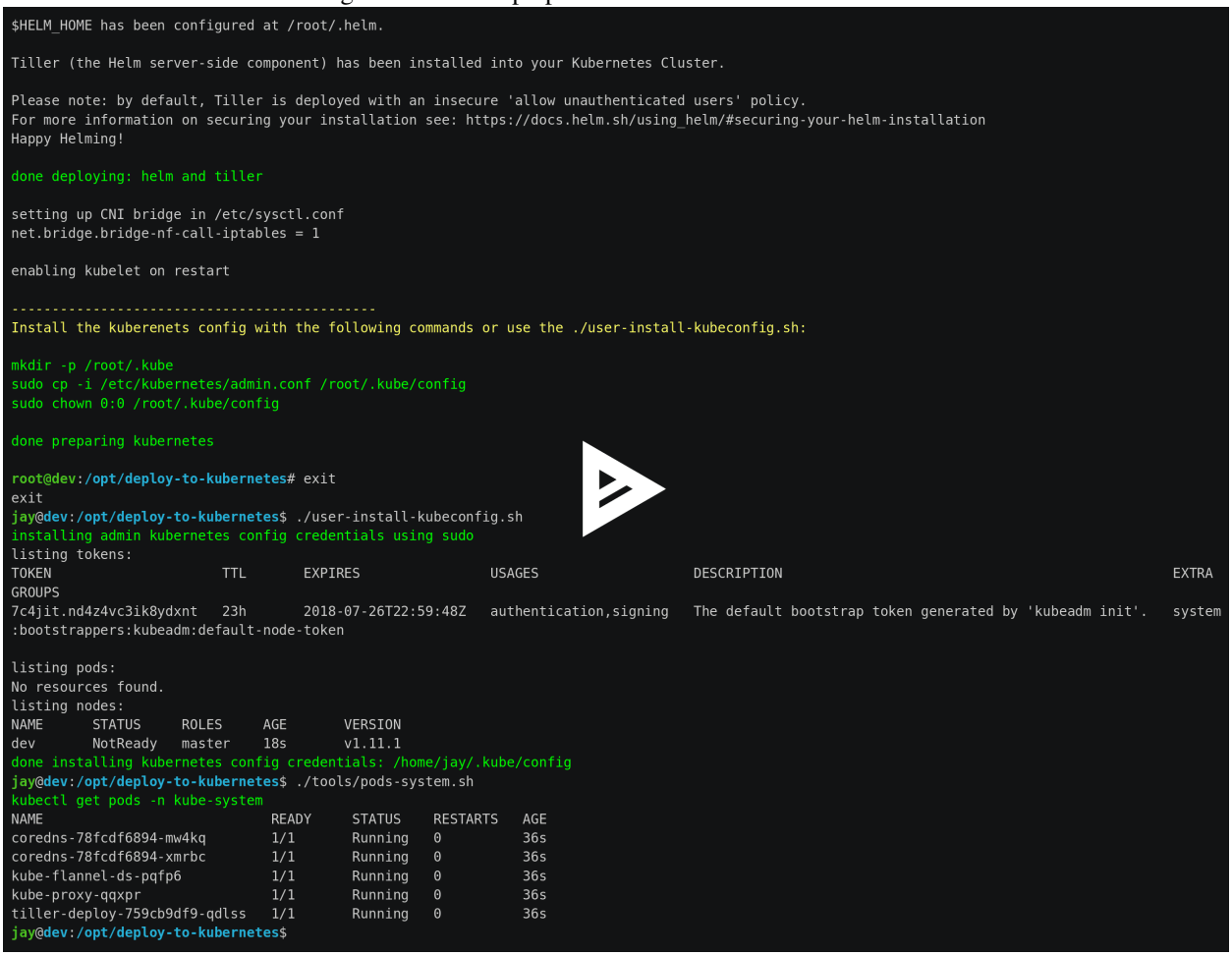
This guide installs the following systems and a storage solution Rook with Ceph cluster (default) or NFS volumes to prepare the host for running containers and automatically running them on host startup:

- Kubernetes
- Helm and Tiller

- Minio S3 Storage
- Persistent Storage Volumes using Rook with Ceph cluster or optional NFS Volumes mounted at: `/data/k8/redis`, `/data/k8/postgres`, `/data/k8/pgadmin`
- Flannel CNI

2.2 Install

Here is a video showing how to prepare the host to run a local Kubernetes cluster:



Preparing the host to run Kubernetes requires run this as root

```
sudo su
./prepare.sh
```

Note: This has only been tested on CentOS 7 and Ubuntu 18.04 and requires commenting out all swap entries in `/etc/fstab` to work

Warning: This guide used to install the cluster on Ubuntu 18.04, but after seeing high CPU utilization after a few days of operation this guide was moved to CentOS 7. The specific issues on Ubuntu were logged in `journalctl`

`-xe` and appeared to be related to "volumes not being found" and "networking disconnects".

CHAPTER 3

Validate

1. Install Kubernetes Config

   Run as your user

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

   Or use the script:

```
./user-install-kubeconfig.sh
```

2. Check the Kubernetes Version

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"11", GitVersion:"v1.11.1",␣
↪GitCommit:"b1b29978270dc22fecc592ac55d903350454310a", GitTreeState:"clean",␣
↪BuildDate:"2018-07-17T18:53:20Z", GoVersion:"go1.10.3", Compiler:"gc", Platform:
↪"linux/amd64"}
The connection to the server localhost:8080 was refused - did you specify the␣
↪right host or port?
```

3. Confirm the Kubernetes Pods Are Running

```
kubectl get pods -n kube-system
```

```
NAME                              READY     STATUS    RESTARTS   AGE
coredns-78fcdf6894-k8srv          1/1       Running   0          4m
coredns-78fcdf6894-xx8bt          1/1       Running   0          4m
etcd-dev                          1/1       Running   0          3m
kube-apiserver-dev                1/1       Running   0          3m
kube-controller-manager-dev       1/1       Running   0          3m
kube-flannel-ds-m8k9w             1/1       Running   0          4m
kube-proxy-p4blg                  1/1       Running   0          4m
```

(continues on next page)

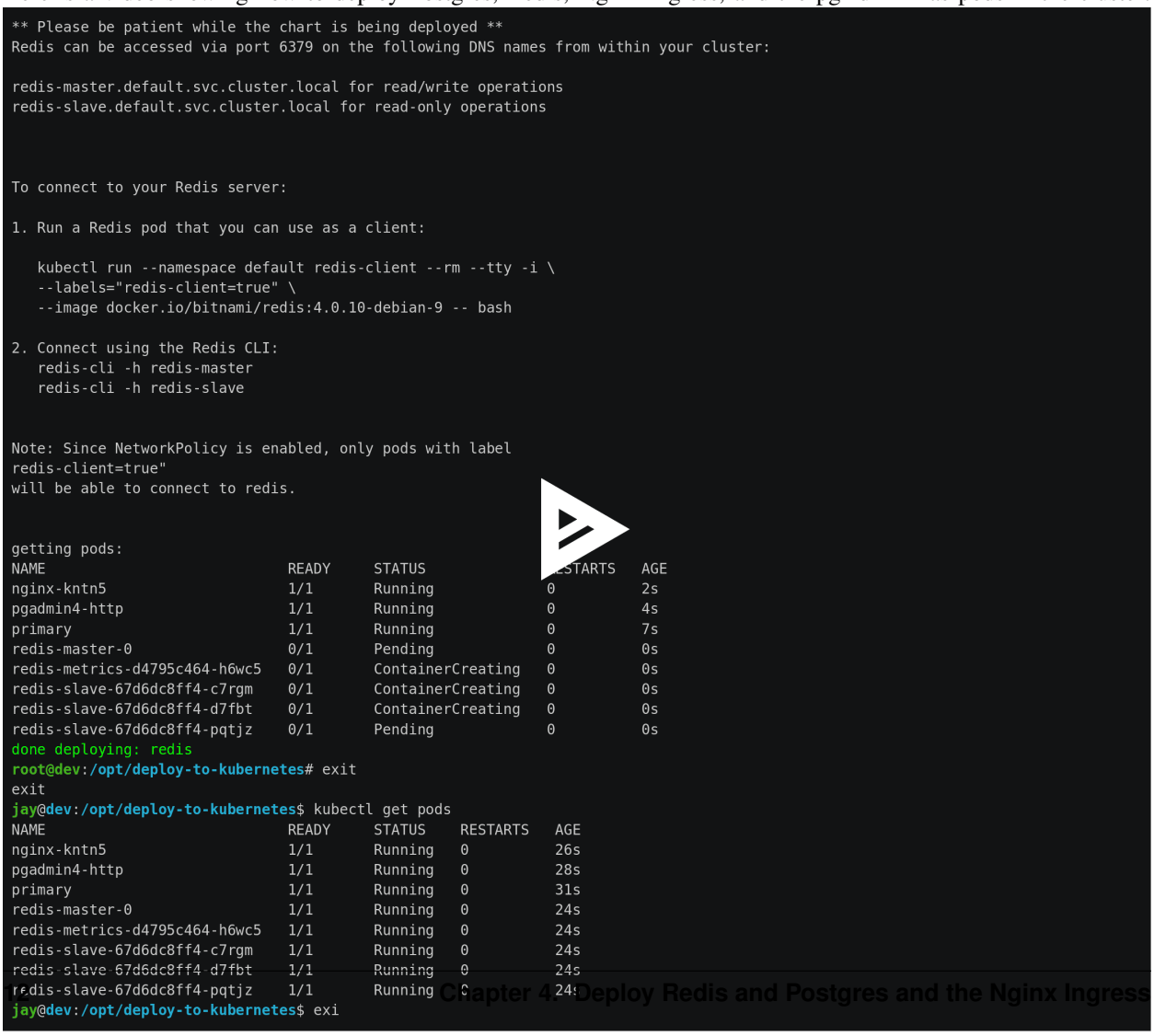
(continued from previous page)

```
kube-scheduler-dev                1/1       Running   0          3m
tiller-deploy-759cb9df9-wxvp8     1/1       Running   0          4m
```

CHAPTER 4

Deploy Redis and Postgres and the Nginx Ingress

Here is a video showing how to deploy Postgres, Redis, Nginx Ingress, and the pgAdmin4 as pods in the cluster:



Note: Postgres, pgAdmin4 and Redis use Rook Ceph to persist data

Here are the commands to deploy Postgres, Redis, Nginx Ingress, and pgAdmin4 in the cluster:

Note: Please ensure helm is installed and the tiller pod in the `kube-system` namespace is the `Running` state or Redis will encounter deployment issues

Install Go using the ./tools/install-go.sh script or with the commands:

```
# note go install has only been tested on CentOS 7 and Ubuntu 18.04:
sudo su
GO_VERSION="1.11"
GO_OS="linux"
GO_ARCH="amd64"
go_file="go${GO_VERSION}.${GO_OS}-${GO_ARCH}.tar.gz"
curl https://dl.google.com/go/${go_file} --output /tmp/${go_file}
export GOPATH=$HOME/go/bin
export PATH=$PATH:$GOPATH:$GOPATH/bin
tar -C $HOME -xzf /tmp/${go_file}
$GOPATH/go get github.com/blang/expenv
# make sure to add GOPATH and PATH to ~/.bashrc
```

```
./user-install-kubeconfig.sh
./deploy-resources.sh
```

If you want to deploy splunk you can add it as an argument:

```
./deploy-resources.sh splunk
```

If you want to deploy splunk with Let's Encrypt make sure to add `prod` as an argument:

```
./deploy-resources.sh splunk prod
```

CHAPTER 5

Start Applications

Here is a video showing how to start the Django REST Framework, Celery Workers, Jupyter, and the AntiNex Core as pods in the cluster:

```
jay@dev:/opt/deploy-to-kubernetes$ source tools/bash_colors.sh && anmt "this will start the stack in kubernetes. pl
ease note that depending on the host's resources this may take some time to download the containers and start them
all up. also you can deploy the optional splunk pod by adding it as an argument: ./start.sh splunk"
this will start the stack in kubernetes. please note that depending on the host's resources this may take some time
 to download the containers and start them all up. also you can deploy the optional splunk pod by adding it as an a
rgument: ./start.sh splunk
jay@dev:/opt/deploy-to-kubernetes$ ./start.sh
starting api: https://github.com/jay-johnson/deploy-to-kubernetes/blob/master/api/run.sh
--------------------------------------------------------------------------------
deploying api: https://github.com/jay-johnson/deploy-to-kubernetes/blob/master/api

applying secrets
secret/api-secret created

applying deployment: ./api/deployment.yml
deployment.extensions/api created

applying service
service/api-svc created

applying ingress
ingress.extensions/api-ingress created

done deploying: api

starting core: https://github.com/jay-johnson/deploy-[illegible]tes/blob/master/core/run.sh
--------------------------------------------------[illegible]----------
deploying core: https://github.com/jay-johnson/deploy[illegible]-kubernetes/blob/master/core

applying secrets
secret/core-secret created

applying deployment: ./core/deployment.yml
deployment.extensions/core created

done deploying: core

starting worker: https://github.com/jay-johnson/deploy-to-kubernetes/blob/master/worker/run.sh
--------------------------------------------------------------------------------
deploying worker: https://github.com/jay-johnson/deploy-to-kubernetes/blob/master/worker

applying secrets
```

Start all applications as your user with the command:

```
./start.sh
```

If you want to deploy the splunk-ready application builds, you can add it as an argument:

```
./start.sh splunk
```

If you want to deploy the splunk-ready application builds integrated with Let's Encrypt TLS encryption, just add `prod` as an argument:

```
./start.sh splunk prod
```

Note: The Cert Manager is set to staging mode by default and requires the `prod` argument to prevent accidentally

getting blocked due to Lets Encrypt rate limits

5.1 Confirm Pods are Running

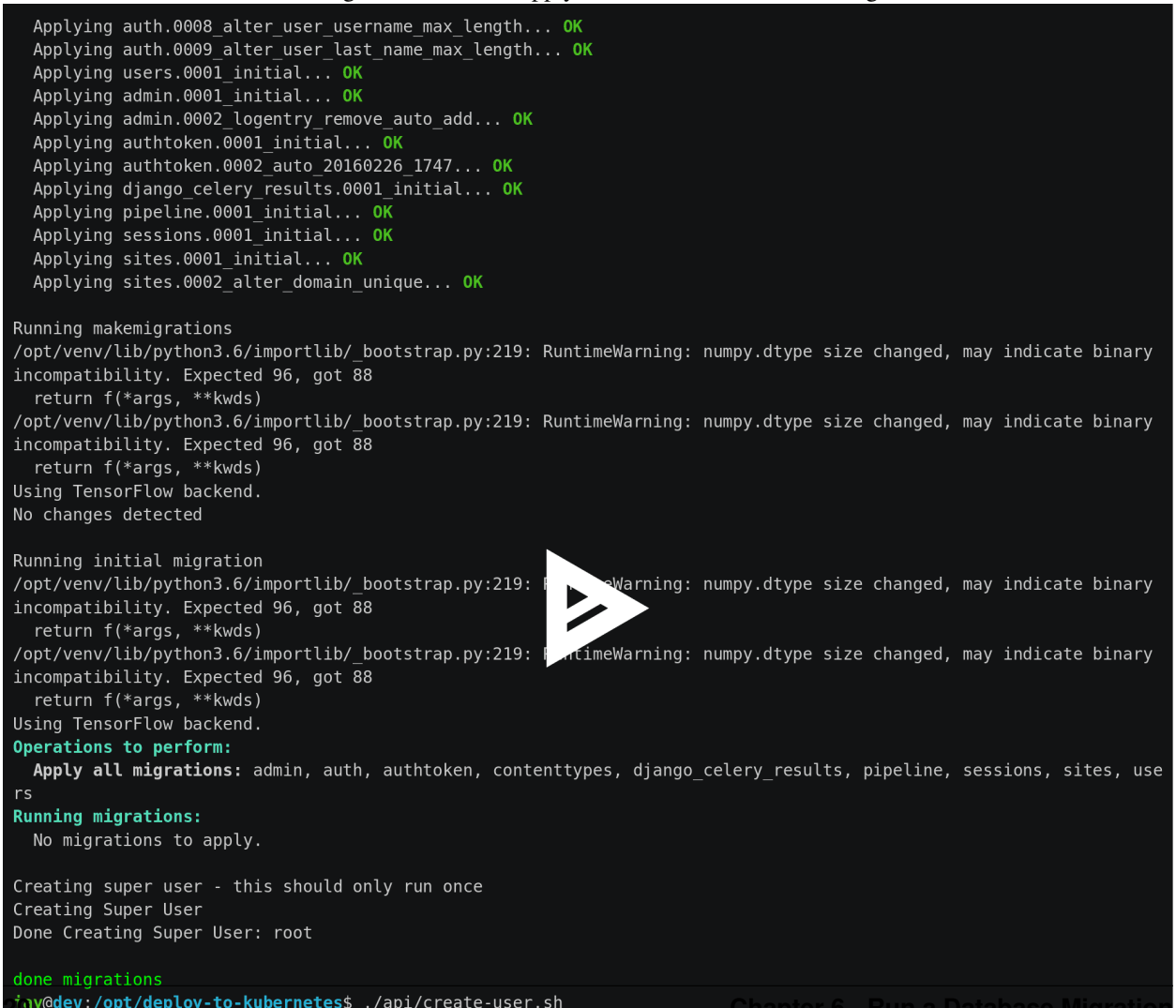
Depending on how fast your network connection is the initial container downloads can take a few minutes. Please wait until all pods are `Running` before continuing.

```
kubectl get pods
```

CHAPTER 6

Run a Database Migration

Here is a video showing how to apply database schema migrations in the cluster:



To apply new Django database migrations, run the following command:

```
./api/migrate-db.sh
```

CHAPTER 7

Add Ingress Locations to /etc/hosts

When running locally (also known in these docs as `dev` mode), all ingress urls need to resolve on the network. Please append the following entries to your local `/etc/hosts` file on the `127.0.0.1` line:

```
sudo vi /etc/hosts
```

Append the entries to the existing `127.0.0.1` line:

```
127.0.0.1   <leave-original-values-here> api.example.com jupyter.example.com pgadmin.
↪example.com splunk.example.com s3.example.com ceph.example.com minio.example.com
```

CHAPTER 8

Using the Minio S3 Object Store

By default, the Kubernetes cluster has a Minio S3 object store running on a Ceph Persistent Volume. S3 is a great solution for distributing files, datasets, configurations, static assets, build artifacts and many more across components, regions, and datacenters using an S3 distributed backend. Minio can also replicate some of the AWS Lambda event-based workflows with Minio bucket event listeners.

For reference, Minio was deployed using this script:

```
./minio/run.sh
```

8.1 View the Verification Tests on the Minio Dashboard

Login with:

- access key: `trexaccesskey`
- secret key: `trex123321`

https://minio.example.com/minio/s3-verification-tests/

8.2 Test Minio S3 with Bucket Creation and File Upload and Download

1. Run from inside the API container

   ```
   ./api/ssh.sh
   source /opt/venv/bin/activate && run_s3_test.py
   ```

   Example logs:

```
creating test file: run-s3-test.txt
connecting: http://minio-service:9000
checking bucket=s3-verification-tests exists
upload_file(run-s3-test.txt, s3-verification-tests, s3-worked-on-2018-08-12-15-21-
↪02)
upload_file(s3-verification-tests, s3-worked-on-2018-08-12-15-21-02, download-run-
↪s3-test.txt)
download_filename=download-run-s3-test.txt contents: tested on: 2018-08-12␣
↪15:21:02
exit
```

2. Run from outside the Kubernetes cluster

Note: This tool requires the python `boto3` pip is installed

```
source ./minio/envs/ext.env
./minio/run_s3_test.py
```

3. Verify the files were uploaded to Minio

   https://minio.example.com/minio/s3-verification-tests/

CHAPTER 9

Using the Rook Ceph Cluster

By default, the Kubernetes cluster is running a Rook Ceph cluster for storage which provides HA persistent volumes and claims.

You can review the persistent volumes and claims using the Ceph Dashboard:

https://ceph.example.com

CHAPTER 10

Create a User

Create the user `trex` with password `123321` on the REST API.

```
./api/create-user.sh
```

CHAPTER 11

Deployed Web Applications

Here are the hosted web application urls. These urls are made accessible by the included nginx-ingress.

CHAPTER 12

View Django REST Framework

Login with:

- user: `trex`
- password: `123321`

https://api.example.com

CHAPTER 13

View Swagger

Login with:

- user: `trex`
- password: `123321`

https://api.example.com/swagger

CHAPTER 14

View Jupyter

Login with:

- password: `admin`

https://jupyter.example.com

CHAPTER 15

View pgAdmin

Login with:

- user: `admin@admin.com`
- password: `123321`

https://pgadmin.example.com

CHAPTER 16

View Minio S3 Object Storage

Login with:

- access key: `trexaccesskey`
- secret key: `trex123321`

https://minio.example.com

CHAPTER 17

View Ceph

https://ceph.example.com

CHAPTER 18

View Splunk

Login with:

- user: `trex`
- password: `123321`

https://splunk.example.com

CHAPTER 19

Training AI with the Django REST API

These steps install the AntiNex python client for training a deep neural network to predict attack packets from recorded network data (all of which is already included in the docker containers).

1. Create a virtual environment and install the client

```
virtualenv -p python3 /opt/venv && source /opt/venv/bin/activate
pip install antinex-client
```

2. Watch the application logs

   From a separate terminal, you can tail the Django REST API logs with the command:

```
./api/logs.sh
```

   From a separate terminal, you can tail the Django Celery Worker logs with the command:

```
./worker/logs.sh
```

   From a separate terminal, you can tail the AntiNex Core Worker logs with the command:

```
./core/logs.sh
```

   Note: Use `ctrl + c` to stop these log tailing commands

CHAPTER 20

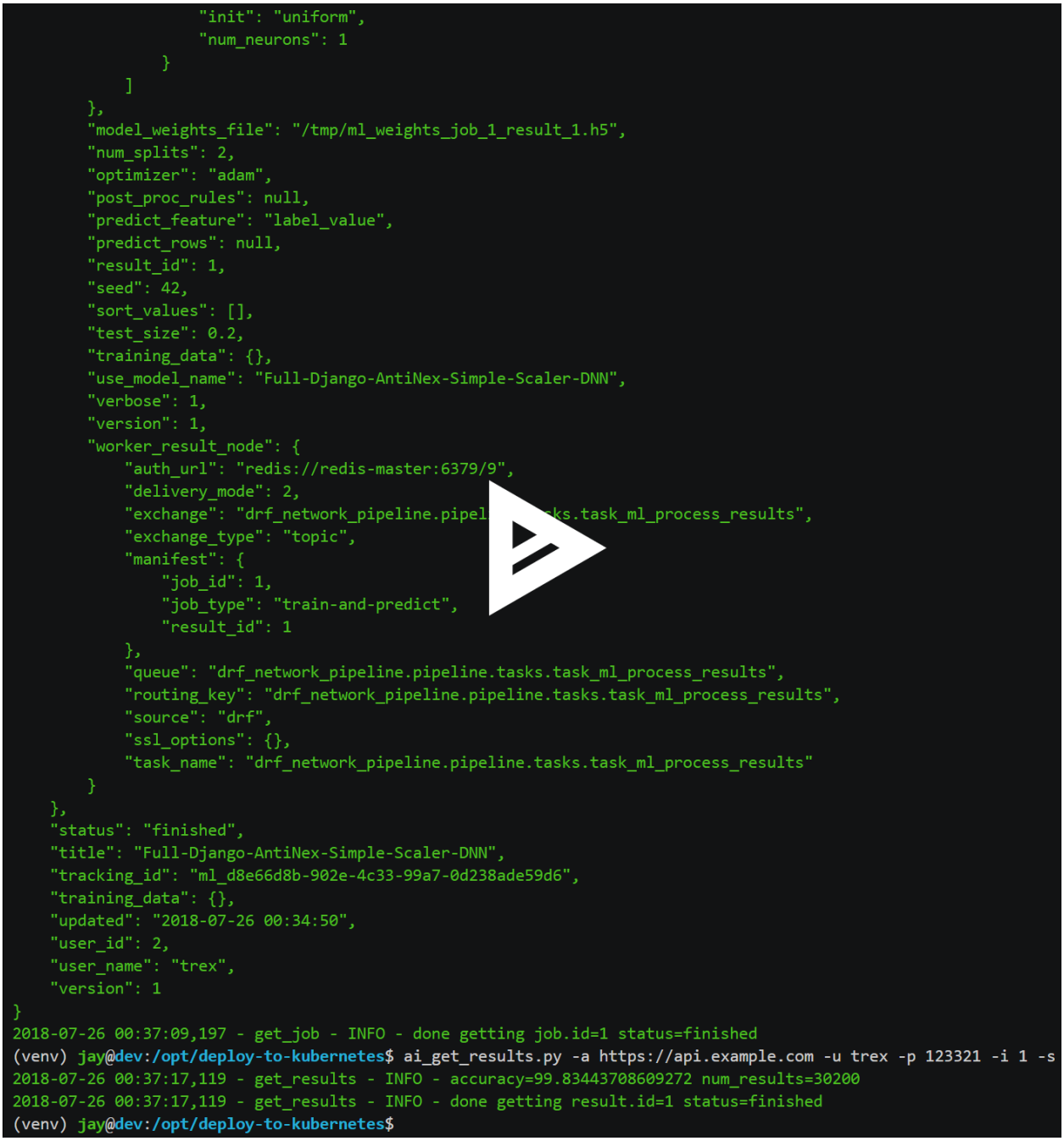
Train a Deep Neural Network on Kubernetes

With virtual environment set up, we can use the client to train a deep neural network with the included datasets:

Note: this can take a few minutes to finish depending on your hosting resources

```
ai -a https://api.example.com -u trex -p 123321 -s -f ./tests/scaler-full-django-
↪antinex-simple.json
```

While you wait, here is a video showing the training and get results:


"init": "uniform",
"num_neurons": 1
}
]
},
"model_weights_file": "/tmp/ml_weights_job_1_result_1.h5",
"num_splits": 2,
"optimizer": "adam",
"post_proc_rules": null,
"predict_feature": "label_value",
"predict_rows": null,
"result_id": 1,
"seed": 42,
"sort_values": [],
"test_size": 0.2,
"training_data": {},
"use_model_name": "Full-Django-AntiNex-Simple-Scaler-DNN",
"verbose": 1,
"version": 1,
"worker_result_node": {
"auth_url": "redis://redis-master:6379/9",
"delivery_mode": 2,
"exchange": "drf_network_pipeline.pipel
ks.task_ml_process_results",
"exchange_type": "topic",
"manifest": {
"job_id": 1,
"job_type": "train-and-predict",
"result_id": 1
},
"queue": "drf_network_pipeline.pipeline.tasks.task_ml_process_results",
"routing_key": "drf_network_pipeline.pipeline.tasks.task_ml_process_results",
"source": "drf",
"ssl_options": {},
"task_name": "drf_network_pipeline.pipeline.tasks.task_ml_process_results"
}
},
"status": "finished",
"title": "Full-Django-AntiNex-Simple-Scaler-DNN",
"tracking_id": "ml_d8e66d8b-902e-4c33-99a7-0d238ade59d6",
"training_data": {},
"updated": "2018-07-26 00:34:50",
"user_id": 2,
"user_name": "trex",
"version": 1
}
2018-07-26 00:37:09,197 - get_job - INFO - done getting job.id=1 status=finished
(venv) jay@dev:/opt/deploy-to-kubernetes$ ai_get_results.py -a https://api.example.com -u trex -p 123321 -i 1 -s
2018-07-26 00:37:17,119 - get_results - INFO - accuracy=99.83443708609272 num_results=30200
2018-07-26 00:37:17,119 - get_results - INFO - done getting result.id=1 status=finished
(venv) jay@dev:/opt/deploy-to-kubernetes$

CHAPTER 21

Get the AI Job Record

```
ai_get_job.py -a https://api.example.com -u trex -p 123321 -i 1
```

CHAPTER 22

Get the AI Training Job Results

```
ai_get_results.py -a https://api.example.com -u trex -p 123321 -i 1 -s
```

CHAPTER 23

Standalone Deployments

Below are steps to manually deploy each component in the stack with Kubernetes.

CHAPTER 24

Deploy Redis

```
./redis/run.sh
```

Or manually with the commands:

```
echo "deploying persistent volume for redis"
kubectl apply -f ./redis/pv.yml
echo "deploying Bitnami redis stable with helm"
helm install \
    --name redis stable/redis \
    --set rbac.create=true \
    --values ./redis/redis.yml
```

24.1 Confirm Connectivity

The following commands assume you have `redis-tools` installed (`sudo apt-get install redis-tools`).

```
redis-cli -h $(kubectl describe pod redis-master-0 | grep IP | awk '{print $NF}') -p␣
↪6379
10.244.0.81:6379> info
10.244.0.81:6379> exit
```

24.2 Debug Redis Cluster

1. Examine Redis Master

   ```
   kubectl describe pod redis-master-0
   ```

2. Examine Persistent Volume Claim

```
kubectl get pvc
NAME                       STATUS    VOLUME                                     ␣
↪CAPACITY   ACCESS MODES   STORAGECLASS      AGE
redis-ceph-data            Bound     pvc-1a88e3a6-9df8-11e8-8047-0800270864a8   ␣
↪8Gi        RWO            rook-ceph-block   46m
```

3. Examine Persistent Volume

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM␣
↪POLICY   STATUS    CLAIM                                 STORAGECLASS      REASON   ␣
↪ AGE
pvc-1a88e3a6-9df8-11e8-8047-0800270864a8   8Gi        RWO            Delete       ␣
↪    Bound     default/redis-ceph-data           rook-ceph-block              46m
```

24.3 Possible Errors

1. Create the Persistent Volumes

```
Warning  FailedMount       2m                kubelet, dev       MountVolume.SetUp␣
↪failed for volume "redis-pv" : mount failed: exit status 32
```

```
./pvs/create-pvs.sh
```

24.4 Delete Redis

```
helm del --purge redis
release "redis" deleted
```

24.5 Delete Persistent Volume and Claim

1. Delete Claim

```
kubectl delete pvc redis-data-redis-master-0
```

2. Delete Volume

```
kubectl delete pv redis-pv
persistentvolume "redis-pv" deleted
```

CHAPTER 25

Deploy Postgres

25.1 Install Go

Using Crunchy Data's postgres containers requires having go installed. Go can be installed using the ./tools/install-go.sh script or with the commands:

```
# note go install has only been tested on CentOS 7 and Ubuntu 18.04:
sudo su
GO_VERSION="1.11"
GO_OS="linux"
GO_ARCH="amd64"
go_file="go${GO_VERSION}.${GO_OS}-${GO_ARCH}.tar.gz"
curl https://dl.google.com/go/${go_file} --output /tmp/${go_file}
export GOPATH=$HOME/go/bin
export PATH=$PATH:$GOPATH:$GOPATH/bin
tar -C $HOME -xzf /tmp/${go_file}
$GOPATH/go get github.com/blang/expenv
# make sure to add GOPATH and PATH to ~/.bashrc
```

25.2 Start

Start the Postgres container within Kubernetes:

```
./postgres/run.sh
```

25.3 Debug Postgres

1. Examine Postgres

```
kubectl describe pod primary

Type    Reason     Age   From               Message
----    ------     ----  ----               -------
Normal  Scheduled  2m    default-scheduler  Successfully assigned default/primary␣
↪to dev
Normal  Pulling    2m    kubelet, dev       pulling image "crunchydata/crunchy-
↪postgres:centos7-10.4-1.8.3"
Normal  Pulled     2m    kubelet, dev       Successfully pulled image
↪"crunchydata/crunchy-postgres:centos7-10.4-1.8.3"
Normal  Created    2m    kubelet, dev       Created container
Normal  Started    2m    kubelet, dev       Started container
```

2. Examine Persistent Volume Claim

```
kubectl get pvc
NAME                       STATUS    VOLUME                                     ␣
↪CAPACITY   ACCESS MODES   STORAGECLASS      AGE
pgadmin4-http-data         Bound     pvc-19031825-9df8-11e8-8047-0800270864a8   ␣
↪400M       RWX            rook-ceph-block   46m
primary-pgdata             Bound     pvc-17652595-9df8-11e8-8047-0800270864a8   ␣
↪400M       RWX            rook-ceph-block   46m
```

3. Examine Persistent Volume

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM␣
↪POLICY   STATUS    CLAIM                              STORAGECLASS      REASON  ␣
↪ AGE
pvc-17652595-9df8-11e8-8047-0800270864a8   400M       RWX            Delete      ␣
↪    Bound     default/primary-pgdata             rook-ceph-block             47m
pvc-19031825-9df8-11e8-8047-0800270864a8   400M       RWX            Delete      ␣
↪    Bound     default/pgadmin4-http-data         rook-ceph-block             47m
```

CHAPTER 26

Deploy pgAdmin

Please confirm go is installed with the Install Go section.

26.1 Start

Start the pgAdmin4 container within Kubernetes:

```
./pgadmin/run.sh
```

26.2 Get Logs

```
./pgadmin/logs.sh
```

26.3 SSH into pgAdmin

```
./pgadmin/ssh.sh
```

CHAPTER 27

Deploy Django REST API

Use these commands to manage the Django REST Framework pods within Kubernetes.

27.1 Start

```
./api/run.sh
```

27.2 Run a Database Migration

To apply a django database migration run the following command:

```
./api/migrate-db.sh
```

27.3 Get Logs

```
./api/logs.sh
```

27.4 SSH into the API

```
./api/ssh.sh
```

CHAPTER 28

Deploy Django Celery Workers

Use these commands to manage the Django Celery Worker pods within Kubernetes.

28.1 Start

```
./worker/run.sh
```

28.2 Get Logs

```
./worker/logs.sh
```

28.3 SSH into the Worker

```
./worker/ssh.sh
```

CHAPTER 29

Deploy AntiNex Core

Use these commands to manage the Backend AntiNex Core pods within Kubernetes.

29.1 Start

```
./core/run.sh
```

29.2 Get Logs

```
./core/logs.sh
```

29.3 SSH into the API

```
./core/ssh.sh
```

CHAPTER 30

Deploy Jupyter

Use these commands to manage the Jupyter pods within Kubernetes.

30.1 Start

```
./jupyter/run.sh
```

30.2 Login to Jupyter

Login with:

- password: `admin`

https://jupyter.example.com

30.3 Get Logs

```
./jupyter/logs.sh
```

30.4 SSH into Jupyter

```
./jupyter/ssh.sh
```

CHAPTER 31

Deploy Splunk

Use these commands to manage the Splunk container within Kubernetes.

31.1 Start

```
./splunk/run.sh
```

31.2 Login to Splunk

Login with:

- user: `trex`
- password: `123321`

https://splunk.example.com

CHAPTER 32

Searching in Splunk

Here is the splunk searching command line tool I use with these included applications:

https://github.com/jay-johnson/spylunking

With search example documentation:

https://spylunking.readthedocs.io/en/latest/scripts.html#examples

CHAPTER 33

Search using Spylunking

Find logs in splunk using the `sp` command line tool:

```
sp -q 'index="antinex" | reverse' -u trex -p 123321 -a $(./splunk/get-api-fqdn.sh) -i␣
↪antinex
```

CHAPTER 34

Find Django REST API Logs in Splunk

```
sp -q 'index="antinex" AND name=api | head 20 | reverse' -u trex -p 123321 -a $(./
↪splunk/get-api-fqdn.sh) -i antinex
```

CHAPTER 35

Find Django Celery Worker Logs in Splunk

```
sp -q 'index="antinex" AND name=worker | head 20 | reverse' -u trex -p 123321 -a $(./
↪splunk/get-api-fqdn.sh) -i antinex
```

CHAPTER 36

Find Core Logs in Splunk

```
sp -q 'index="antinex" AND name=core | head 20 | reverse' -u trex -p 123321 -a $(./
↪splunk/get-api-fqdn.sh) -i antinex
```

CHAPTER 37

Find Jupyter Logs in Splunk

```
sp -q 'index="antinex" AND name=jupyter | head 20 | reverse' -u trex -p 123321 -a $(./
↪splunk/get-api-fqdn.sh) -i antinex
```

Example for debugging `sp` splunk connectivity from inside an API Pod:

```
kubectl exec -it api-59496ccb5f-2wp5t -n default echo 'starting search' && /bin/bash -
↪c "source /opt/venv/bin/activate && sp -q 'index="antinex" AND hostname=local' -u␣
↪trex -p 123321 -a 10.101.107.205:8089 -i antinex"
```

37.1 Get Logs

```
./splunk/logs.sh
```

37.2 SSH into Splunk

```
./splunk/ssh.sh
```

CHAPTER 38

Deploy Nginx Ingress

This project is currently using the nginx-ingress instead of the Kubernetes Ingress using nginx. Use these commands to manage and debug the nginx ingress within Kubernetes.

Note: The default Yaml file annotations only work with the nginx-ingress customizations

38.1 Start

```
./ingress/run.sh
```

38.2 Get Logs

```
./ingress/logs.sh
```

38.3 SSH into the Ingress

```
./ingress/ssh.sh
```

CHAPTER 39

View Ingress Nginx Config

When troubleshooting the nginx ingress, it is helpful to view the nginx configs inside the container. Here is how to view the configs:

```
./ingress/view-configs.sh
```

CHAPTER 40

View a Specific Ingress Configuration

If you know the pod name and the namespace for the nginx-ingress, then you can view the configs from the command line with:

```
app_name="jupyter"
app_name="pgadmin"
app_name="api"
use_namespace="default"
pod_name=$(kubectl get pods -n ${use_namespace} | awk '{print $1}' | grep nginx |␣
↪head -1)
kubectl exec -it ${pod_name} -n ${use_namespace} cat /etc/nginx/conf.d/${use_
↪namespace}-${app_name}-ingress.conf
```

CHAPTER 41

Deploy Splunk

41.1 Start

To deploy splunk you can add the argument `splunk` to the ./deploy-resources.sh splunk script. Or you can manually run it with the command:

```
./splunk/run.sh
```

Or if you want to use Let's Encrypt for SSL:

```
./splunk/run.sh prod
```

CHAPTER 42

Deploy Splunk-Ready Applications

After deploying the splunk pod, you can deploy the splunk-ready applications with the command:

```
./start.sh splunk
```

42.1 Get Logs

```
./splunk/logs.sh
```

42.2 SSH into Splunk

```
./splunk/ssh.sh
```

42.3 View Ingress Config

```
./splunk/view-ingress-config.sh
```

CHAPTER 43

Create your own self-signed x509 TLS Keys, Certs and Certificate Authority with Ansible

If you have openssl installed you can use this ansible playbook to create your own certificate authority (CA), keys and certs.

1. Create the CA, Keys and Certificates

```
cd ansible
ansible-playbook -i inventory_dev create-x509s.yml
```

2. Check the CA, x509, keys and certificates for the client and server were created

```
ls -l ./ssl
```

CHAPTER 44

Deploying Your Own x509 TLS Encryption files as Kubernetes Secrets

This is a work in progress, but in `dev` mode the cert-manager is not in use. Instead the cluster utilizes pre-generated x509s TLS SSL files created with the included ansible playbook create-x509s.yml. Once created, you can deploy them as Kubernetes secrets using the deploy-secrets.sh script and reload them at any time in the future.

44.1 Deploy Secrets

Run this to create the TLS secrets:

```
./ansible/deploy-secrets.sh
```

44.2 List Secrets

```
kubectl get secrets | grep tls
tls-ceph                kubernetes.io/tls                     2         36m
tls-client              kubernetes.io/tls                     2         36m
tls-database            kubernetes.io/tls                     2         36m
tls-docker              kubernetes.io/tls                     2         36m
tls-jenkins             kubernetes.io/tls                     2         36m
tls-jupyter             kubernetes.io/tls                     2         36m
tls-k8                  kubernetes.io/tls                     2         36m
tls-kafka               kubernetes.io/tls                     2         36m
tls-kibana              kubernetes.io/tls                     2         36m
tls-minio               kubernetes.io/tls                     2         36m
tls-nginx               kubernetes.io/tls                     2         36m
tls-pgadmin             kubernetes.io/tls                     2         36m
tls-phpmyadmin          kubernetes.io/tls                     2         36m
tls-rabbitmq            kubernetes.io/tls                     2         36m
tls-redis               kubernetes.io/tls                     2         36m
tls-restapi             kubernetes.io/tls                     2         36m
```

(continues on next page)

(continued from previous page)

```
tls-s3                  kubernetes.io/tls                       2         36m
tls-splunk              kubernetes.io/tls                       2         36m
tls-webserver           kubernetes.io/tls                       2         36m
```

44.3 Reload Secrets

If you want to deploy new TLS secrets at any time, use the `reload` argument with the `deploy-secrets.sh` script. Doing so will delete the original secrets and recreate all of them using the new TLS values:

```
./ansible/deploy-secrets.sh -r
```

CHAPTER 45

Deploy Cert Manager with Let's Encrypt

Use these commands to manage the Cert Manager with Let's Encrypt SSL support within Kubernetes. By default, the cert manager is deployed only in `prod` mode. If you run it in production mode, then it will install real, valid x509 certificates from Let's Encrypt into the nginx-ingress automatically.

45.1 Start with Let's Encrypt x509 SSL Certificates

Start the cert manager in `prod` mode to enable Let's Encrypt TLS Encryption with the command:

```
./start.sh prod
```

Or manually with the command:

```
./cert-manager/run.sh prod
```

If you have splunk you can just add it to the arguments:

```
./start.sh splunk prod
```

45.2 View Logs

When using the production mode, make sure to view the logs to ensure you are not being blocked due to rate limiting:

```
./cert-manager/logs.sh
```

CHAPTER 46

Stop the Cert Manager

If you notice things are not working correctly, you can quickly prevent yourself from getting blocked by stopping the cert manager with the command:

```
./cert-manager/_uninstall.sh
```

Note: If you get blocked due to rate-limits it will show up in the cert-manager logs like:

```
I0731 07:53:43.313709       1 sync.go:273] Error issuing certificate for default/api.
↪antinex.com-tls: error getting certificate from acme server: acme:␣
↪urn:ietf:params:acme:error:rateLimited: Error finalizing order :: too many␣
↪certificates already issued for exact set of domains: api.antinex.com: see https://
↪letsencrypt.org/docs/rate-limits/
E0731 07:53:43.313738       1 sync.go:182] [default/api.antinex.com-tls] Error␣
↪getting certificate 'api.antinex.com-tls': secret "api.antinex.com-tls" not found
```

46.1 Debugging

To reduce debugging issues, the cert manager ClusterIssuer objects use the same name for staging and production mode. This is nice because you do not have to update all the annotations to deploy on production vs staging:

The cert manager starts and defines the issuer name for both production and staging as:

```
--set ingressShim.defaultIssuerName=letsencrypt-issuer
```

Make sure to set any nginx ingress annotations that need Let's Encrypt SSL encryption to these values:

```
annotations:
  kubernetes.io/tls-acme: "true"
  kubernetes.io/ingress.class: "nginx"
  certmanager.k8s.io/cluster-issuer: "letsencrypt-issuer"
```

CHAPTER 47

Troubleshooting

CHAPTER 48

Customize Minio and How to Troubleshoot

48.1 Change the Minio Access and Secret Keys

1. Change the secrets file: `minio/secrets/default_access_keys.yml`

   Change the `access_key` and `secret_key` values after generating the new base64 string values for the secrets file:

```
echo -n "NewAccessKey" | base64
TmV3QWNjZXNzS2V5
# now you can replace the access_key's value in the secrets file with the string:␣
↪TmV3QWNjZXNzS2V5
```

```
echo -n "NewSecretKey" | base64
TmV3U2VjcmV0S2V5
# now you can replace the secret_key's value in the secrets file with the string:␣
↪TmV3QWNjZXNzS2V5
```

2. Deploy the secrets file

```
kubectl apply -f ./minio/secrets/default_access_keys.yml
```

3. Restart the Minio Pod

```
kubectl delete pod -l app=minio
```

If you have changed the default access and secret keys, then you will need to export the following environment variables as needed to make sure the `./minio/run_s3_test.py` test script works:

```
export S3_ACCESS_KEY=<minio access key: trexaccesskey - default>
export S3_SECRET_KEY=<minio secret key: trex123321 - default>
export S3_REGION_NAME=<minio region name: us-east-1 - default>
```

(continues on next page)

(continued from previous page)

```
export S3_ADDRESS=<minio service endpoint: external address found with the script ./
↪minio/get-s3-endpoint.sh and the internal cluster uses the service: minio-
↪service:9000>
# examples of setting up a minio env files are in: ./minio/envs
```

48.2 View the Minio Dashboard

Login with:

- access key: `trexaccesskey`
- secret key: `trex123321`

https://minio.example.com

48.3 Get S3 Internal Endpoint

If you want to use the Minio S3 service within the cluster please use the endpoint:

```
minio-service:9000
```

or source the internal environment file:

```
source ./minio/envs/int.env
```

48.4 Get S3 External Endpoint

If you want to use the Minio S3 service from outside the cluser please use the endpoint provided by the script:

```
./minio/get-s3-endpoint.sh
# which for this documentation was the minio service's Endpoints:
# 10.244.0.103:9000
```

or source the external environment file:

```
source ./minio/envs/ext.env
```

48.5 Debugging Steps

1. Load the Minio S3 external environment variables:

   ```
   source ./minio/envs/ext.env
   ```

2. Run the S3 Verification test script

   ```
   ./minio/run_s3_test.py
   ```

3. Confirm Verification Keys are showing up in this Minio S3 bucket

   https://minio.example.com/minio/s3-verification-tests/

   If not please use the describe tools in `./minio/describe-*.sh` to grab the logs and please file a GitHub issue

48.6 Describe Pod

```
./minio/describe-service.sh
```

48.7 Describe Service

```
./minio/describe-service.sh
```

48.8 Describe Ingress

```
./minio/describe-ingress.sh
```

48.9 Uninstall Minio

```
./minio/_uninstall.sh
```

CHAPTER 49

Ceph Troubeshooting

Please refer to the Rook Common Issues for the latest updates on how to use your Rook Ceph cluster.

Note: By default Ceph is not hosting the S3 solution unless `cephs3` is passed in as an argument to `deploy-resource.sh`.

There are included troubleshooting tools in the `./rook` directory with an overview of each below:

49.1 Validate Ceph System Pods are Running

```
./rook/view-system-pods.sh

----------------------------------------
Getting the Rook Ceph System Pods:
kubectl -n rook-ceph-system get pod
NAME                                  READY     STATUS    RESTARTS   AGE
rook-ceph-agent-g9vzm                 1/1       Running   0          7m
rook-ceph-operator-78d498c68c-tbsdf   1/1       Running   0          7m
rook-discover-h9wj9                   1/1       Running   0          7m
```

49.2 Validate Ceph Pods are Running

```
./rook/view-ceph-pods.sh

----------------------------------------
Getting the Rook Ceph Pods:
kubectl -n rook-ceph get pod
NAME                                  READY     STATUS      RESTARTS   AGE
```

(continues on next page)

(continued from previous page)

```
rook-ceph-mgr-a-9c44495df-7jksz       1/1       Running     0          6m
rook-ceph-mon0-rxxsl                  1/1       Running     0          6m
rook-ceph-mon1-gqblg                  1/1       Running     0          6m
rook-ceph-mon2-7xfsq                  1/1       Running     0          6m
rook-ceph-osd-id-0-7d4d4c8794-kgr2d   1/1       Running     0          6m
rook-ceph-osd-prepare-dev-kmsn9       0/1       Completed   0          6m
rook-ceph-tools                       1/1       Running     0          6m
```

49.3 Validate Persistent Volumes are Bound

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
↪STATUS    CLAIM                              STORAGECLASS      REASON    AGE
pvc-03e6e4ef-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete
↪Bound     default/certs-pv-claim             rook-ceph-block             46m
pvc-0415de24-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete
↪Bound     default/configs-pv-claim           rook-ceph-block             46m
pvc-0441307f-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete
↪Bound     default/datascience-pv-claim       rook-ceph-block             46m
pvc-0468ef73-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete
↪Bound     default/frontendshared-pv-claim    rook-ceph-block             46m
pvc-04888222-9df8-11e8-8047-0800270864a8   1Gi        RWO            Delete
↪Bound     default/staticfiles-pv-claim       rook-ceph-block             46m
pvc-1c3e359d-9df8-11e8-8047-0800270864a8   10Gi       RWO            Delete
↪Bound     default/minio-pv-claim             rook-ceph-block             46m
```

49.4 Validate Persistent Volume Claims are Bound

```
kubectl get pvc
NAME                     STATUS    VOLUME                                     ␣
↪CAPACITY   ACCESS MODES   STORAGECLASS      AGE
certs-pv-claim           Bound     pvc-03e6e4ef-9df8-11e8-8047-0800270864a8   1Gi
↪     RWO            rook-ceph-block   47m
configs-pv-claim         Bound     pvc-0415de24-9df8-11e8-8047-0800270864a8   1Gi
↪     RWO            rook-ceph-block   47m
datascience-pv-claim     Bound     pvc-0441307f-9df8-11e8-8047-0800270864a8   1Gi
↪     RWO            rook-ceph-block   47m
frontendshared-pv-claim  Bound     pvc-0468ef73-9df8-11e8-8047-0800270864a8   1Gi
↪     RWO            rook-ceph-block   47m
minio-pv-claim           Bound     pvc-1c3e359d-9df8-11e8-8047-0800270864a8   10Gi
↪     RWO            rook-ceph-block   46m
```

49.5 Create a Persistent Volume Claim

Going forward, Ceph will automatically create a persistent volume if one is not available for binding to an available Persistent Volume Claim. To create a new persistent volume, just create a claim and verify the Rook Ceph cluster created the persistent volume and both are bound to each other.

```
kubectl apply -f pvs/pv-staticfiles-ceph.yml
```

49.6 Verify the Persistent Volume is Bound

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
↪STATUS    CLAIM                           STORAGECLASS      REASON    AGE
pvc-77afbc7a-9ade-11e8-b293-0800270864a8   20Gi       RWO            Delete
↪Bound     default/staticfiles-pv-claim    rook-ceph-block             2s
```

49.7 Verify the Persistent Volume Claim is Bound

```
kubectl get pvc
NAME                    STATUS    VOLUME                                     CAPACITY
↪ ACCESS MODES   STORAGECLASS      AGE
staticfiles-pv-claim    Bound     pvc-77afbc7a-9ade-11e8-b293-0800270864a8   20Gi
↪ RWO            rook-ceph-block   11s
```

49.8 Describe Persistent Volumes

```
kubectl describe pv pvc-c88fc37b-9adf-11e8-9fae-0800270864a8
Name:            pvc-c88fc37b-9adf-11e8-9fae-0800270864a8
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by=ceph.rook.io/block
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    rook-ceph-block
Status:          Bound
Claim:           default/certs-pv-claim
Reclaim Policy:  Delete
Access Modes:    RWO
Capacity:        20Gi
Node Affinity:   <none>
Message:
Source:
    Type:       FlexVolume (a generic volume resource that is provisioned/attached
↪using an exec based plugin)
    Driver:     ceph.rook.io/rook-ceph-system
    FSType:     xfs
    SecretRef:  <nil>
    ReadOnly:   false
    Options:    map[clusterNamespace:rook-ceph image:pvc-c88fc37b-9adf-11e8-9fae-
↪0800270864a8 pool:replicapool storageClass:rook-ceph-block]
Events:          <none>
```

49.9 Show Ceph Cluster Status

```
./rook/show-ceph-status.sh

-----------------------------------------------
Getting the Rook Ceph Status with Toolbox:
kubectl -n rook-ceph exec -it rook-ceph-tools ceph status
cluster:
    id:     7de1988c-03ea-41f3-9930-0bde39540552
    health: HEALTH_OK

services:
    mon: 3 daemons, quorum rook-ceph-mon2,rook-ceph-mon0,rook-ceph-mon1
    mgr: a(active)
    osd: 1 osds: 1 up, 1 in

data:
    pools:   1 pools, 100 pgs
    objects: 12 objects, 99 bytes
    usage:   35443 MB used, 54756 MB / 90199 MB avail
    pgs:     100 active+clean
```

49.10 Show Ceph OSD Status

```
./rook/show-ceph-osd-status.sh

-----------------------------------------------
Getting the Rook Ceph OSD Status with Toolbox:
kubectl -n rook-ceph exec -it rook-ceph-tools ceph osd status
+----+-----------------------------------+-------+-------+--------+---------+-------
↪-+---------+-----------+
| id |                host               |  used | avail | wr ops | wr data | rd␣
↪ops | rd data |   state   |
+----+-----------------------------------+-------+-------+--------+---------+-------
↪-+---------+-----------+
| 0  | rook-ceph-osd-id-0-7d4d4c8794-kgr2d | 34.6G | 53.4G |    0   |     0   |    0 ␣
↪ |     0   | exists,up |
+----+-----------------------------------+-------+-------+--------+---------+-------
↪-+---------+-----------+
```

49.11 Show Ceph Free Space

```
./rook/show-ceph-df.sh

-----------------------------------------------
Getting the Rook Ceph df with Toolbox:
kubectl -n rook-ceph exec -it rook-ceph-tools ceph df
GLOBAL:
    SIZE       AVAIL      RAW USED     %RAW USED
    90199M     54756M       35443M         39.29
POOLS:
```

(continues on next page)

(continued from previous page)

```
    NAME            ID     USED      %USED     MAX AVAIL     OBJECTS
    replicapool     1        99          0        50246M          12
```

49.12 Show Ceph RDOS Free Space

```
./rook/show-ceph-rados-df.sh

-----------------------------------------------
Getting the Rook Ceph rados df with Toolbox:
kubectl -n rook-ceph exec -it rook-ceph-tools rados df
POOL_NAME   USED OBJECTS CLONES COPIES MISSING_ON_PRIMARY UNFOUND DEGRADED RD_OPS RD␣
↪ WR_OPS WR
replicapool   99      12      0     12                  0       0        0    484␣
↪381k     17 7168

total_objects    12
total_used       35443M
total_avail      54756M
total_space      90199M
```

49.13 Out of IP Addresses

Flannel can exhaust all available ip addresses in the CIDR network range. When this happens please run the following command to clean up the local cni network files:

```
./tools/reset-flannel-cni-networks.sh
```

CHAPTER 50

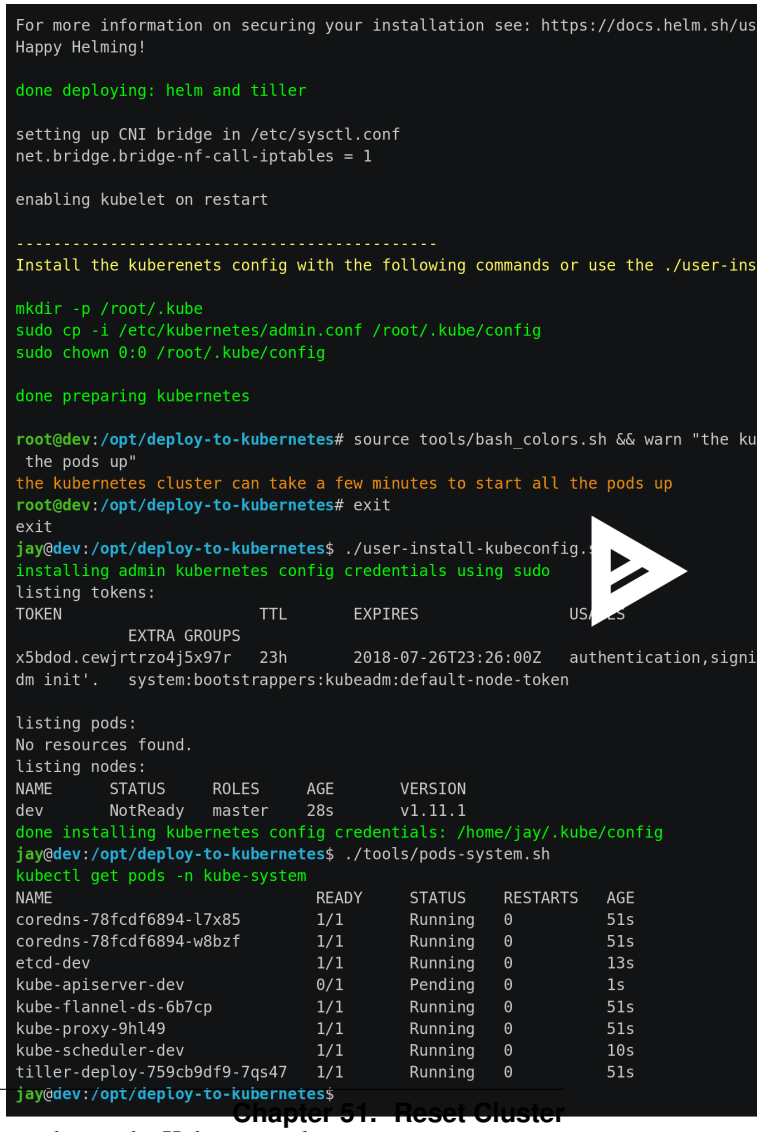
AntiNex Stack Status

Here are the AntiNex repositories, documentation and build reports:

Component	Build	Docs Link	Docs Build
REST API		Docs	
Core Worker		Docs	
Network Pipeline		Docs	
AI Utils		Docs	
Client		Docs	

CHAPTER 51

Reset Cluster



Here is a video showing how to reset the local Kubernetes cluster.
Please be careful as these commands will shutdown all containers and reset the Kubernetes cluster.

Run as root:

```
sudo su
kubeadm reset -f
./prepare.sh
```

Or use the file:

```
sudo su
./tools/cluster-reset.sh
```

Or the full reset and deploy once ready:

```
sudo su
cert_env=dev; ./tools/reset-flannel-cni-networks.sh; ./tools/cluster-reset.sh ; ./
↪user-install-kubeconfig.sh ; sleep 30; ./deploy-resources.sh splunk ${cert_env}
exit
# as your user
./user-install-kubeconfig.sh
# depending on testing vs prod:
# ./start.sh splunk
# ./start.sh splunk prod
```

CHAPTER 52

Development

Right now, the python virtual environment is only used to bring in ansible for running playbooks, but it will be used in the future with the kubernetes python client as I start using it more and more.

```
virtualenv -p python3 /opt/venv && source /opt/venv/bin/activate && pip install -e .
```

CHAPTER 53

Testing

```
py.test
```

or

```
python setup.py test
```

CHAPTER 54

License

Apache 2.0 - Please refer to the LICENSE for more details

CHAPTER 55

Running a Distributed Ceph Cluster on a Kubernetes Cluster

55.1 Overview

This guide automates installing a native ceph cluster inside a running kubernetes native cluster. It requires creating and attaching 3 additional hard drive disk images to 3 kubernetes cluster vm's (tested on CentOS 7). This guide assumes your kubernetes cluster is using `kvm` with `virsh` for running the `attach-disk` commands (it was tested with kubernetes version `1.13.3`).

By default, the disk images will be installed at: `/cephdata/m[123]/k8-centos-m[123]`. These disks will be automatically partitioned and formatted using ceph zap, and zap will format each disk using the recommended XFS filesystem.

Note: This is a work in progress and things will likely change. This guide will be updated as progress proceeds.

55.2 Background

This installer was built to replace Rook-Ceph after encountering cluster stability issues after ~30 days of uptime in 2019. The steps are taken from the Ceph Helm installer:

http://docs.ceph.com/docs/mimic/start/kube-helm/

CHAPTER 56

Add the Ceph Mon Cluster Service FQDN to /etc/hosts

Before starting, please ensure each kubernetes vm has the following entries in `/etc/hosts`:

m1

```
sudo echo "192.168.0.101    ceph-mon.ceph.svc.cluster.local" >> /etc/hosts
```

m2

```
sudo echo "192.168.0.102    ceph-mon.ceph.svc.cluster.local" >> /etc/hosts
```

m3

```
sudo echo "192.168.0.103    ceph-mon.ceph.svc.cluster.local" >> /etc/hosts
```

Note: Missing this step can result in some debugging

CHAPTER 57

Build KVM HDD Images

Change to the `ceph` directory.

```
cd ceph
```

Generate `100 GB` hdd images for the ceph cluster with 1 qcow2 image for each of the three vm's:

```
./kvm-build-images.sh
```

The files are saved here:

```
/cephdata/
├── m1
│   └── k8-centos-m1
├── m2
│   └── k8-centos-m2
└── m3
    └── k8-centos-m3
```

CHAPTER 58

Attach KVM Images to VMs

This will attach each `100 GB` image to the correct vm: `m1`, `m2` or `m3`

```
./kvm-attach-images.sh
```

CHAPTER 59

Format Disks in VM

With automatic ssh root login access, you can run this to partition, mount and format each of the new images:

Warning: Please be careful running this as it can delete any previously saved data.

Warning: Please be aware that `fdisk` can also hang and requires hard rebooting the cluster if orphaned `fdisk` processes get stuck. Please let me know if you have a way to get around this. There are many discussions like the process that would not die about this issue on the internet.

```
./_kvm-format-images.sh
```

CHAPTER 60

Install Ceph on All Kubernetes Nodes

Please add `ceph-common`, `centos-release-ceph-luminous` and `lsof` to all kubernetes node vm's before deploying ceph.

For additional set up please refer to the official ceph docs:

http://docs.ceph.com/docs/master/install/get-packages/

For CentOS 7 you can run the ./ceph/install-ceph-tools.sh script or the commands:

```
sudo rpm --import "https://download.ceph.com/keys/release.asc"
sudo yum install -y ceph-common centos-release-ceph-luminous lsof
```

CHAPTER 61

Deploy Ceph Cluster

Ceph requires running a local Helm repo server (just like the Redis cluster does) and building then installing chart to get the cluster pods running.

```
./run.sh
```

CHAPTER 62

Watch all Ceph Logs with Kubetail

With kubetail installed you can watch all the ceph pods at once with:

```
./logs-kt-ceph.sh
```

or manually with:

```
kubetail ceph -c cluster-log-tailer -n ceph
```

CHAPTER 63

Show Pods

View the ceph cluster pods with:

```
./show-pods.sh
-----------------------------------------------------
Getting Ceph pods with:
kubectl get pods -n ceph

NAME                                        READY   STATUS      RESTARTS   AGE
ceph-mds-85b4fbb478-wjmxb                   1/1     Running     1          4m38s
ceph-mds-keyring-generator-pvh4l            0/1     Completed   0          4m38s
ceph-mgr-588577d89f-w8p8v                   1/1     Running     1          4m38s
ceph-mgr-keyring-generator-76l5r            0/1     Completed   0          4m38s
ceph-mon-429mk                              3/3     Running     0          4m39s
ceph-mon-6fvv6                              3/3     Running     0          4m39s
ceph-mon-75n4t                              3/3     Running     0          4m39s
ceph-mon-check-549b886885-cb64q             1/1     Running     0          4m38s
ceph-mon-keyring-generator-q26p2            0/1     Completed   0          4m38s
ceph-namespace-client-key-generator-bbvt2   0/1     Completed   0          4m38s
ceph-osd-dev-vdb-96v7h                      1/1     Running     0          4m39s
ceph-osd-dev-vdb-g9zkg                      1/1     Running     0          4m39s
ceph-osd-dev-vdb-r5fxr                      1/1     Running     0          4m39s
ceph-osd-keyring-generator-6pg77            0/1     Completed   0          4m38s
ceph-rbd-provisioner-5cf47cf8d5-kbfvt       1/1     Running     0          4m38s
ceph-rbd-provisioner-5cf47cf8d5-pwj4s       1/1     Running     0          4m38s
ceph-rgw-7b9677854f-8d7s5                   1/1     Running     1          4m38s
ceph-rgw-keyring-generator-284kp            0/1     Completed   0          4m38s
ceph-storage-keys-generator-bc6dq           0/1     Completed   0          4m38s
```

CHAPTER 64

Check Cluster Status

With the cluster running you can quickly check the cluster status with:

```
./cluster-status.sh
-------------------------------------------------
Getting Ceph cluster status:

kubectl -n ceph exec -ti ceph-mon-check-549b886885-cb64q -c ceph-mon -- ceph -s
cluster:
    id:     aa06915f-3cf6-4f74-af69-9afb41bf464d
    health: HEALTH_OK

services:
    mon: 3 daemons, quorum master1.example.com,master2.example.com,master3.example.com
    mgr: master2.example.com(active)
    mds: cephfs-1/1/1 up  {0=mds-ceph-mds-85b4fbb478-wjmxb=up:active}
    osd: 3 osds: 3 up, 3 in
    rgw: 1 daemon active

data:
    pools:   7 pools, 148 pgs
    objects: 208 objects, 3359 bytes
    usage:   325 MB used, 284 GB / 284 GB avail
    pgs:     148 active+clean
```

CHAPTER 65

Validate a Pod can Mount a Persistent Volume on the Ceph Cluster in Kubernetes

Run these steps to walk through integration testing your kubernetes cluster can host persistent volumes for pods running on a ceph cluster inside kubernetes. This means your data is backed to an attached storage disk on the host vm in:

Note: If any of these steps fail please refer to the Kubernetes Ceph Cluster Debugging Guide

```
ls /cephdata/*/*
/cephdata/m1/k8-centos-m1   /cephdata/m2/k8-centos-m2   /cephdata/m3/k8-centos-m3
```

65.1 Create PVC

```
kubectl apply -f test/pvc.yml
```

65.2 Verify PVC is Bound

```
kubectl get pvc | grep test-ceph
test-ceph-pv-claim        Bound    pvc-a715256d-38c3-11e9-8e7c-525400275ad4   1Gi      ␣
↪    RWO            ceph-rbd          46s
```

65.3 Create Pod using PVC as a mounted volume

```
kubectl apply -f test/mount-pv-in-pod.yml
```

65.4 Verify Pod has Mounted Volume inside Container

```
kubectl describe pod ceph-tester
```

65.5 Verify Ceph is Handling Data

```
./cluster-status.sh
```

```
./show-ceph-osd-status.sh

---------------------------------------------
Getting Ceph osd status:
kubectl -n ceph exec -it ceph-rgw-7b9677854f-lcr77 -- ceph osd status
+----+---------------------+-------+-------+--------+---------+--------+---------+----
↪-------+
| id |         host        |  used | avail | wr ops | wr data | rd ops | rd data |  ␣
↪state   |
+----+---------------------+-------+-------+--------+---------+--------+---------+----
↪-------+
| 0  | master2.example.com |  141M | 94.8G |    0   |     0   |    1   |    16   |␣
↪exists,up |
| 1  | master1.example.com |  141M | 94.8G |    0   |     0   |    0   |     0   |␣
↪exists,up |
| 2  | master3.example.com |  141M | 94.8G |    0   |     0   |    0   |     0   |␣
↪exists,up |
+----+---------------------+-------+-------+--------+---------+--------+---------+----
↪-------+
```

65.6 Delete Ceph Tester Pod

```
kubectl delete -f test/mount-pv-in-pod.yml
```

65.7 Recreate Ceph Tester Pod

```
kubectl apply -f test/mount-pv-in-pod.yml
```

65.8 View Logs from Previous Pod

```
kubectl logs -f $(kubectl get po | grep ceph-tester | awk '{print $1}')
```

Notice the last entries in the log show the timestamp changed in the logs like:

```
kubectl logs -f $(kubectl get po | grep ceph-tester | awk '{print $1}')
total 20
drwx------    2 root     root         16384 Feb 25 07:31 lost+found
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--    1 root     root            29 Feb 25 07:33 updated
Filesystem                Size      Used Available Use% Mounted on
/dev/rbd0               975.9M      2.5M    957.4M   0% /testing
last update:
Mon Feb 25 07:33:34 UTC 2019
Mon Feb 25 08:29:27 UTC 2019
```

65.9 Cleanup Ceph Tester Pod

```
kubectl delete -f test/mount-pv-in-pod.yml
kubectl delete -f test/pvc.yml
```

CHAPTER 66

Kubernetes Ceph Cluster Debugging Guide

66.1 Confirm Ceph OSD pods are using the KVM Mounted Disks

If the cluster is in a `HEALTH_WARN` state with a message about `low on available space`:

```
./cluster-status.sh
-------------------------------------------------
Getting Ceph cluster status:

kubectl -n ceph exec -ti ceph-mon-kjcqq -c ceph-mon -- ceph -s
cluster:
    id:     747d4fc1-2d18-423a-96fe-43419f8fe9cd
    health: HEALTH_WARN
            mons master2.example.com,master3.example.com are low on available space
```

Then please confirm the vms all mounted the correct storage disks for ceph. This could be due to your `/etc/fstab` entries failing to mount (say after a cluster reboot), which we can quickly check with:

```
./check-kvm-disk-mounts.sh
```

If you see something like:

```
-------------------------------------------------
Checking Ceph OSD Pod Mountpoints for /dev/vdb1:

checking: ceph-osd-dev-vdb-5dv8l
kubectl -n ceph exec -it ceph-osd-dev-vdb-5dv8l -- df -h /var/lib/ceph/
failed: ceph-osd-dev-vdb-5dv8l is using /dev/mapper/centos-root
checking: ceph-osd-dev-vdb-s77lh
kubectl -n ceph exec -it ceph-osd-dev-vdb-s77lh -- df -h /var/lib/ceph/
failed: ceph-osd-dev-vdb-s77lh is using /dev/mapper/centos-root
checking: ceph-osd-dev-vdb-vxvd7
kubectl -n ceph exec -it ceph-osd-dev-vdb-vxvd7 -- df -h /var/lib/ceph/
failed: ceph-osd-dev-vdb-vxvd7 is using /dev/mapper/centos-root
```

(continues on next page)

(continued from previous page)

```
detected at least one Ceph OSD mount failure
Please review the Ceph debugging guide: https://deploy-to-kubernetes.readthedocs.io/
↪en/latest/ceph.html#confirm-ceph-osd-pods-are-using-the-kvm-mounted-disks for more␣
↪details on how to fix this issue
```

Then the correct storage disk(s) failed to mount correctly, and ceph is using the wrong disk for extended, persistent storage on the vm. This can put your ceph cluster into a `HEALTH_WARN` state as seen above in the cluster status script.

To fix this error, please either use the `./_kvm-format-images.sh` (if you are ok reformatting all previous ceph data on the disks) or manually with the following steps:

1. Fix `/etc/fstab` on all vms

   Warning: Only run these steps when the cluster can be taken down as it will interrupt services

   Confirm the `/etc/fstab` entry has the correct value:

   ```
   cat /etc/fstab | grep vdb1
   /dev/vdb1 /var/lib/ceph  xfs     defaults    0 0
   ```

   For any vm that does not have the `/etc/fstab` entry, please run these commands as root to set them up manually:

2. Delete the bad mountpoint: `/var/lib/ceph`

   ```
   rm -rf /var/lib/ceph
   ```

3. Add the new `/dev/vdb` entry to `/etc/fstab`

   ```
   sudo echo "/dev/vdb1 /var/lib/ceph  xfs     defaults    0 0" >> /etc/fstab
   ```

4. Mount the disk

   ```
   mount /dev/vdb1 /var/lib/ceph
   ```

5. Uninstall Ceph

   Warning: Running `./_uninstall.sh` will impact any pods using the `ceph-rbd` storageClass

   ```
   ./_uninstall.sh
   ```

6. Reinstall Ceph or Reboot all impacted vms

   ```
   ./run.sh
   ```

7. Confirm the Mounts Worked

   ```
   ./check-kvm-disk-mounts.sh
   ```

66.2 The ceph-tester failed to start

If your integration test fails mounting the test persistent volume follow these steps to try and debug the issue:

Check if the `ceph-mon` service is missing a ClusterIP:

```
get svc -n ceph
NAME        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
ceph-mon    ClusterIP   None            <none>        6789/TCP   11m
ceph-rgw    ClusterIP   10.102.90.139   <none>        8088/TCP   11m
```

See if there is a log in the `ceph-tester` showing the error.

```
kubectl describe po ceph-tester
```

May show something similar to this for why it failed:

```
server name not found: ceph-mon.ceph.svc.cluster.local
```

If `ceph-mon.ceph.svc.cluster.local` is not found, manually add it to `/etc/hosts` on all nodes.

m1 node:

```
# on m1 /etc/hosts add:
192.168.0.101    ceph-mon.ceph.svc.cluster.local
```

Confirm connectivity

```
telnet ceph-mon.ceph.svc.cluster.local 6789
```

m2 node:

```
# on m2 /etc/hosts add:
192.168.0.102    ceph-mon.ceph.svc.cluster.local
```

Confirm connectivity

```
telnet ceph-mon.ceph.svc.cluster.local 6789
```

m3 node:

```
# on m3 /etc/hosts add:
192.168.0.103    ceph-mon.ceph.svc.cluster.local
```

Confirm connectivity

```
telnet ceph-mon.ceph.svc.cluster.local 6789
```

If connectivity was fixed on all the kubernetes nodes then please `./_uninstall.sh` and then reinstall with `./run.sh`

If not please continue to the next debugging section below.

66.3 Orphaned fdisk Processes

If you have to use the `./_uninstall.sh -f` to uninstall and re-partition the disk images, there is a chance the partition tool `fdisk` can hang. If this happens it should hang the `./_uninstall.sh -f` and be detected by the user or the script (hopefully).

If your cluster hits this issue I have to reboot my server.

Note: This guide does not handle single kubernetes vm outages at the moment.

For the record, here's some attempts to kill this process:

```
root@master3:~# ps auwwx | grep fdisk
root     18516  0.0  0.0 112508   976 ?        D    06:33   0:00 fdisk /dev/vdb
root     21957  0.0  0.0 112704   952 pts/1    S+   06:37   0:00 grep --color fdisk
root@master3:~# kill -9 18516
root@master3:~# ps auwwx | grep fdisk
root     18516  0.0  0.0 112508   976 ?        D    06:33   0:00 fdisk /dev/vdb
root     22031  0.0  0.0 112704   952 pts/1    S+   06:37   0:00 grep --color fdisk
```

```
root@master3:~# strace -p 18516
strace: Process 18516 attached
# no more logs after waiting +60 seconds
strace: Process 18516 attached
^C
^C
^C
^C^Z
[1]+  Stopped                 strace -p 18516
# so did strace just die by touching that pid?
```

What is `fdisk` using on the filesystem?

Notice multiple `ssh pipe` resources are in use below. Speculation here: are those pipes the `fdisk` wait prompt over a closed ssh session (I am guessing but who knows)?

```
root@master3:~# lsof -p 18516
COMMAND   PID USER   FD   TYPE DEVICE  SIZE/OFF      NODE NAME
fdisk   18516 root  cwd    DIR  253,0       271 100663361 /root
fdisk   18516 root  rtd    DIR  253,0       285        64 /
fdisk   18516 root  txt    REG  253,0    200456  33746609 /usr/sbin/fdisk
fdisk   18516 root  mem    REG  253,0 106070960      1831 /usr/lib/locale/locale-
↪archive
fdisk   18516 root  mem    REG  253,0   2173512  33556298 /usr/lib64/libc-2.17.so
fdisk   18516 root  mem    REG  253,0     20112  33556845 /usr/lib64/libuuid.so.1.3.0
fdisk   18516 root  mem    REG  253,0    261488  33556849 /usr/lib64/libblkid.so.1.1.0
fdisk   18516 root  mem    REG  253,0    164240  33556291 /usr/lib64/ld-2.17.so
fdisk   18516 root    0r  FIFO    0,9       0t0    847143 pipe
fdisk   18516 root    1w  FIFO    0,9       0t0    845563 pipe
fdisk   18516 root    2w  FIFO    0,9       0t0    845564 pipe
fdisk   18516 root    3u   BLK 252,16     0t512      1301 /dev/vdb
root@master3:~#
```

Stop `strace` that will prevent `gdb` tracing next:

```
root@master3:~# ps auwwx | grep 26177
root     14082  0.0  0.0 112704   952 pts/0    S+   07:02   0:00 grep --color 26177
root     26177  0.0  0.0   7188   600 ?        S    06:41   0:00 strace -p 18516
root@master3:~# kill -9 26177
```

`gdb` also hangs when trying this stackoverflow:

```
gdb -p 18516
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-110.el7
```

(continues on next page)

(continued from previous page)

```
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Attaching to process 18516
```

If a vm gets to this point then the server gets rebooted.

Here are other operational debugging tools that were used with cluster start up below:

66.4 Check osd pods

When setting up new devices with kubernetes you will see the `osd` pods failing and here is a tool to describe one of the pods quickly:

```
./describe-osd.sh
```

66.5 Watch the Ceph Mon Logs with Kubetail

```
kubetail ceph-mon -c cluster-log-tailer -n ceph
```

66.6 Attach Successful but Mounting a Ceph PVC fails

Even if the cluster is stable, your pv's can attach but fail to mount due to:

```
Events:
Type     Reason                  Age                  From                       ␣
↪Message
----     ------                  ----                 ----                       ---
↪----
Normal   Scheduled               3m25s                default-scheduler          ␣
↪Successfully assigned default/busybox-mount to master3.example.com
Normal   SuccessfulAttachVolume  3m25s                attachdetach-controller    ␣
↪AttachVolume.Attach succeeded for volume "pvc-907ae639-3880-11e9-85a5-525400275ad4"
Warning  FailedMount             82s                  kubelet, master3.example.com ␣
↪Unable to mount volumes for pod "busybox-mount_default(24ac4333-3881-11e9-85a5-
↪525400275ad4)": timeout expired waiting for volumes to attach or mount for pod
↪"default"/"busybox-mount". list of unmounted volumes=[storage]. list of unattached␣
↪volumes=[storage default-token-6f9vj]
Warning  FailedMount             45s (x8 over 109s)   kubelet, master3.example.com ␣
↪MountVolume.WaitForAttach failed for volume "pvc-907ae639-3880-11e9-85a5-
↪525400275ad4" : fail to check rbd image status with: (executable file not found in
↪$PATH), rbd output: ()
```

To fix this please:

1. Install `ceph-common` on each kubernetes node.
2. Uninstall the ceph cluster with:

```
./_uninstall.sh -f
```

3. Delete Remaining pv's

```
kubectl delete --ignore-not-found pv $(kubectl get pv | grep ceph-rbd | grep -v␣
↪rook | awk '{print $1}')
```

66.7 Previous Cluster Cleanup Failed

Please run the `_uninstall.sh` if you see this kind of error when running the `cluster-status.sh`:

```
./cluster-status.sh
-------------------------------------------------
Getting Ceph cluster status:

kubectl -n ceph exec -ti ceph-mon-p9tvw -c ceph-mon -- ceph -s
2019-02-24 06:02:12.468777 7f90f6509700  0 librados: client.admin authentication␣
↪error (1) Operation not permitted
[errno 1] error connecting to the cluster
command terminated with exit code 1
```

CHAPTER 67

OSD Issues

When debugging ceph `osd` issues, please start by reviewing the pod logs with:

```
./logs-osd-prepare-pod.sh
```

67.1 OSD Pool Failed to Initialize

Depending on how many disks and the capacity of the ceph cluster, your first time creating the `osd pool` startup may hit an error during this command:

```
kubectl -n ceph exec -ti ${pod_name} -c ceph-mon -- ceph osd pool create rbd 256
```

With an error like:

```
creating osd pool
Error ERANGE:  pg_num 256 size 3 would mean 840 total pgs, which exceeds max 600 (mon_
↪max_pg_per_osd 200 * num_in_osds 3)
command terminated with exit code 34
initializing osd
rbd: error opening default pool 'rbd'
Ensure that the default pool has been created or specify an alternate pool name.
command terminated with exit code 2
```

Please reduce the number at the end of the `ceph osd pool create rbd 256` to:

```
kubectl -n ceph exec -ti ${pod_name} -c ceph-mon -- ceph osd pool create rbd 100
```

67.2 OSD Pod Prepare is Unable to Zap

To fix this error below, make sure the `ceph-overrides.yaml` is using the correct `/dev/vdb` path:

```
Traceback (most recent call last):
File "/usr/sbin/ceph-disk", line 9, in <module>
    load_entry_point('ceph-disk==1.0.0', 'console_scripts', 'ceph-disk')()
File "/usr/lib/python2.7/dist-packages/ceph_disk/main.py", line 5717, in run
    main(sys.argv[1:])
File "/usr/lib/python2.7/dist-packages/ceph_disk/main.py", line 5668, in main
    args.func(args)
File "/usr/lib/python2.7/dist-packages/ceph_disk/main.py", line 4737, in main_zap
    zap(dev)
File "/usr/lib/python2.7/dist-packages/ceph_disk/main.py", line 1681, in zap
    raise Error('not full block device; cannot zap', dev)
ceph_disk.main.Error: Error: not full block device; cannot zap: /dev/vdb1
```

67.3 OSD unable to find IP Address

To fix this error below, make sure to either remove the `network` definitions in the `ceph-overrides.yaml`.

```
+ exec /usr/bin/ceph-osd --cluster ceph -f -i 2 --setuser ceph --setgroup disk
2019-02-24 08:53:40.592021 7f4313687e00 -1 unable to find any IP address in networks
↪'172.21.0.0/20' interfaces ''
```

CHAPTER 68

Cluster Status Tools

68.1 Show All

```
./show-ceph-all.sh
```

68.2 Show Cluster Status

```
./show-ceph-status.sh
```

```
-----------------------------------------------
Getting Ceph status:
kubectl -n ceph exec -it ceph-rgw-7b9677854f-k6hj7 -- ceph status
cluster:
    id:     384880f1-23f3-4a83-bff8-93624120a4cf
    health: HEALTH_OK

services:
    mon: 3 daemons, quorum master1.example.com,master2.example.com,master3.example.com
    mgr: master3.example.com(active)
    mds: cephfs-1/1/1 up  {0=mds-ceph-mds-85b4fbb478-9fhf4=up:active}
    osd: 3 osds: 3 up, 3 in
    rgw: 1 daemon active

data:
    pools:   6 pools, 48 pgs
    objects: 208 objects, 3359 bytes
    usage:   324 MB used, 284 GB / 284 GB avail
    pgs:     48 active+clean
```

68.3 Show Ceph DF

```
./show-ceph-df.sh
```

```
------------------------------------------------
Getting Ceph df:
kubectl -n ceph exec -it ceph-rgw-7b9677854f-k6hj7 -- ceph df
GLOBAL:
    SIZE     AVAIL     RAW USED     %RAW USED
    284G      284G          323M          0.11
POOLS:
    NAME                    ID     USED      %USED     MAX AVAIL     OBJECTS
    .rgw.root               1      1113          0        92261M           4
    cephfs_data             2         0          0        92261M           0
    cephfs_metadata         3      2246          0        92261M          21
    default.rgw.control     4         0          0        92261M           8
    default.rgw.meta        5         0          0        92261M           0
    default.rgw.log         6         0          0        92261M           0
```

68.4 Show Ceph OSD Status

```
./show-ceph-osd-status.sh
```

```
Getting Ceph osd status:
kubectl -n ceph exec -it ceph-rgw-7b9677854f-k6hj7 -- ceph osd status
+----+---------------------+-------+-------+--------+---------+--------+---------+----
↪-------+
| id |         host        |  used | avail | wr ops | wr data | rd ops | rd data |  ⌴
↪state   |
+----+---------------------+-------+-------+--------+---------+--------+---------+----
↪-------+
| 0  | master2.example.com |  107M | 94.8G |    1   |    18   |    0   |    13   |⌴
↪exists,up |
| 1  | master1.example.com |  107M | 94.8G |    3   |   337   |    0   |     0   |⌴
↪exists,up |
| 2  | master3.example.com |  108M | 94.8G |    5   |   315   |    1   |   353   |⌴
↪exists,up |
+----+---------------------+-------+-------+--------+---------+--------+---------+----
↪-------+
```

68.5 Show Ceph Rados DF

```
./show-ceph-rados-df.sh
```

```
Getting Ceph rados df:
kubectl -n ceph exec -it ceph-rgw-7b9677854f-k6hj7 -- rados df
POOL_NAME           USED OBJECTS CLONES COPIES MISSING_ON_PRIMARY UNFOUND DEGRADED RD_
↪OPS RD    WR_OPS WR
.rgw.root           1113       4      0     12                  0       0        0  ⌴
↪ 12 8192      4 4096
```

(continues on next page)

(continued from previous page)

```
cephfs_data             0        0       0       0                    0        0        0
↪  0     0       0     0
cephfs_metadata      2246       21       0      63                    0        0        0
↪  0     0      42  8192
default.rgw.control     0        8       0      24                    0        0        0
↪  0     0       0     0

total_objects    33
total_used       323M
total_avail      284G
total_space      284G
```

CHAPTER 69

Uninstall

To uninstall the ceph cluster and leave the mounted KVM disks `/dev/vdb` untouched:

```
./_uninstall.sh
```

69.1 Uninstall and Reformat KVM Images

To uninstall the ceph cluster and reformat the mounted KVM disks `/dev/vdb`:

> **Warning:** Running this will destroy all data across the cluster by reformatting the /dev/vdb block devices in each vm

```
./_uninstall.sh -f
```

CHAPTER 70

Managing a Multi-Host Kubernetes Cluster with an External DNS Server

This guide is for managing a multi-host Kubernetes cluster deployed across 3 CentOS 7 vms. Once running, you can access the sample applications from outside the cluster with the included DNS nameserver (bind9).

70.1 Overview

Set up 3 CentOS 7 vms and run an external DNS (using bind9) for a distributed, multi-host Kubernetes cluster that is accessible on the domain: `example.com`

70.2 Background

Why did you make this?

Before using DNS, I was stuck managing and supporting many DHCP IP addresses in `/etc/hosts` like below. This ended up being way more time consuming than necessary. So I made this guide for adding a DNS server over a multi-host Kubernetes cluster.

```
############################################################
#
# find the MAC using: ifconfig | grep -A 3 enp | grep ether | awk '{print $2}'
#
# MAC address:  08:00:27:37:80:e1
192.168.0.101   m1 master1 master1.example.com api.example.com ceph.example.com mail.
↪example.com minio.example.com pgadmin.example.com s3.example.com www.example.com
#
# MAC address:  08:00:27:21:80:19
192.168.0.102   m2 master2 master2.example.com jupyter.example.com
#
# MAC address:  08:00:27:21:80:29
192.168.0.103   m3 master3 master3.example.com splunk.example.com
```

70.3 Allocate VM Resources

1. Each vm should have at least 70 GB hard drive space
2. Each vm should have at least 2 CPU cores and 4 GB memory
3. Each vm should have a bridge network adapter that is routeable
4. Take note of each vm's bridge network adapter's MAC address (this will help finding the vm's IP address in a router's web app or using network detection tools)

70.4 Install CentOS 7

Install CentOS 7 on each vm and here is the CentOS 7 DVD download page

1. Additional notes
   - I use the multihost/_reset-cluster-using-ssh.sh script to reset the cluster using ssh.
   - I recently moved from running on Virtualbox with Ubuntu 18.04 to KVM with CentOS 7, and am tracking the changes in the `multihost` directory. This includes how each vm's bridge network adapter uses a `ifcg-eth0` interface and starter scripts to make this process repeatable. Please note, it will continue to be a work in progress.
     - create a vm with kvm
     - start m1 vm
     - start m2 vm
     - start m3 vm
     - m1 directory
     - m2 directory
     - m3 directory

70.5 Prepare VMs

1. This command needs to run as root and will prepare the CentOS vm for running Kubernetes.

   Use this script to prepare a CentOS 7 vm for running in this cluster.

```
./centos/prepare.sh
```

2. Confirm Kube Proxy Kernel Modules are Loaded

   The vanilla CentOS 7 installer does not install the required kernel modules. By running the `centos/prepare-vm.sh` script, each vm's kernel should support the required kube proxy kernel odules:

```
lsmod | grep ip_vs
ip_vs_sh               12688  0
ip_vs_wrr              12697  0
ip_vs_rr               12600  0
ip_vs                 141473  6 ip_vs_rr,ip_vs_sh,ip_vs_wrr
```

(continues on next page)

(continued from previous page)

```
nf_conntrack          133053  9 ip_vs,nf_nat,nf_nat_ipv4,nf_nat_ipv6,xt_conntrack,
↪nf_nat_masquerade_ipv4,nf_conntrack_netlink,nf_conntrack_ipv4,nf_conntrack_ipv6
libcrc32c              12644  5 xfs,ip_vs,libceph,nf_nat,nf_conntrack
```

70.6 Install Kubernetes

Install Kubernetes on each vm using your own tool(s) of choice or the deploy to kubernetes tool I wrote. This repository builds each vm in the cluster as a master node, and will use kubeadm join to add **master2.example.com** and **master3.example.com** to the initial, primary node **master1.example.com**.

CHAPTER 71

Start All Kubernetes Cluster VMs

1. Start Kubernetes on the Master 1 VM

   Once Kubernetes is running on your initial, primary master vm (mine is on **master1.example.com**), you can prepare the cluster with the commands:

```
# ssh into your initial, primary vm
ssh 192.168.0.101
```

   If you're using the deploy-to-kubernetes repository to run an AI stack on Kubernetes, then the following commands will start the master 1 vm for preparing the cluster to run the stack:

```
# make sure this repository is cloned on all cluster nodes to: /opt/deploy-to-
↪kubernetes
# git clone https://github.com/jay-johnson/deploy-to-kubernetes.git /opt/deploy-
↪to-kubernetes
sudo su
# for preparing to run the example.com cluster use:
cert_env=dev; cd /opt/deploy-to-kubernetes; ./tools/reset-flannel-cni-networks.sh;
↪ ./tools/cluster-reset.sh ; ./user-install-kubeconfig.sh
```

2. Confirm only 1 Cluster Node is in the Ready State

```
kubectl get nodes -o wide --show-labels
```

3. Print the Cluster Join Command on Master 1

```
kubeadm token create --print-join-command
```

4. Join Master 2 to Master 1

```
ssh 192.168.0.102
sudo su
kubeadm join 192.168.0.101:6443 --token <token> --discovery-token-ca-cert-hash
↪<hash>
exit
```

5. Join Master 3 to Master 1

```
ssh 192.168.0.103
sudo su
kubeadm join 192.168.0.101:6443 --token <token> --discovery-token-ca-cert-hash
↪<hash>
exit
```

71.1 Verify the Cluster has 3 Ready Nodes

1. Set up your host for using kubectl

```
sudo apt-get install -y kubectl
```

2. Copy the Kubernetes Config from Master 1 to your host

```
mkdir -p 775 ~/.kube/config >> /dev/null
scp 192.168.0.101:/root/.kube/config ~/.kube/config
```

3. Verify the 3 nodes (vms) are in a Status of Ready in the Kubernetes cluster

```
kubectl get nodes -o wide --show-labels
NAME                  STATUS    ROLES     AGE       VERSION   INTERNAL-IP
↪EXTERNAL-IP   OS-IMAGE                KERNEL-VERSION               CONTAINER-
↪RUNTIME   LABELS
master1.example.com   Ready     master    7h        v1.11.2   192.168.0.101
↪<none>        CentOS Linux 7 (Core)   3.10.0-862.11.6.el7.x86_64   docker://18.
↪6.1     backend=disabled,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/
↪os=linux,ceph=enabled,datascience=disabled,frontend=enabled,kubernetes.io/
↪hostname=master1.example.com,minio=enabled,node-role.kubernetes.io/master=,
↪splunk=disabled
master2.example.com   Ready     <none>    7h        v1.11.2   192.168.0.102
↪<none>        CentOS Linux 7 (Core)   3.10.0-862.11.6.el7.x86_64   docker://18.
↪6.1     backend=enabled,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/
↪os=linux,ceph=enabled,datascience=enabled,frontend=enabled,kubernetes.io/
↪hostname=master2.example.com,minio=disabled,splunk=disabled
master3.example.com   Ready     <none>    7h        v1.11.2   192.168.0.103
↪<none>        CentOS Linux 7 (Core)   3.10.0-862.11.6.el7.x86_64   docker://18.
↪6.1     backend=enabled,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/
↪os=linux,ceph=enabled,datascience=disabled,frontend=disabled,kubernetes.io/
↪hostname=master3.example.com,minio=disabled,splunk=enabled
```

CHAPTER 72

Deploy a Distributed AI Stack to a Multi-Host Kubernetes Cluster

This will deploy the AntiNex AI stack to the new multi-host Kubernetes cluster.

72.1 Deploy Cluster Resources

1. ssh into the master 1 host:

```
ssh 192.168.0.101
```

2. Install Go

   The Postgres and pgAdmin containers require running as root with Go installed on the master 1 host:

```
# note this has only been tested on CentOS 7:
sudo su
GO_VERSION="1.11"
GO_OS="linux"
GO_ARCH="amd64"
go_file="go${GO_VERSION}.${GO_OS}-${GO_ARCH}.tar.gz"
curl https://dl.google.com/go/${go_file} --output /tmp/${go_file}
export GOPATH=$HOME/go/bin
export PATH=$PATH:$GOPATH:$GOPATH/bin
tar -C $HOME -xzf /tmp/${go_file}
$GOPATH/go get github.com/blang/expenv
# make sure to add GOPATH and PATH to ~/.bashrc
```

3. Deploy the stack's resources:

```
cert_env=dev
cd /opt/deploy-to-kubernetes; ./deploy-resources.sh splunk ceph ${cert_env}
exit
```

72.2 Start the AI Stack

1. Run the Start command

```
cert_env=dev
./start.sh splunk ceph ${cert_env}
```

2. Verify the Stack is Running

Note: This may take a few minutes to download all images and sync files across the cluster.

```
NAME                                READY     STATUS    RESTARTS   AGE
api-774765b455-nlx8z                1/1       Running   0          4m
api-774765b455-rfrcw                1/1       Running   0          4m
core-66994c9f4d-nq4sh               1/1       Running   0          4m
jupyter-577696f945-cx5gr            1/1       Running   0          4m
minio-deployment-7fdcfd6775-pmdww   1/1       Running   0          5m
nginx-5pp8n                         1/1       Running   0          5m
nginx-dltv8                         1/1       Running   0          5m
nginx-kxn7l                         1/1       Running   0          5m
pgadmin4-http                       1/1       Running   0          5m
primary                             1/1       Running   0          5m
redis-master-0                      1/1       Running   0          5m
redis-metrics-79cfcb86b7-k9584      1/1       Running   0          5m
redis-slave-7cd9cdc695-jgcsk        1/1       Running   2          5m
redis-slave-7cd9cdc695-qd5pl        1/1       Running   2          5m
redis-slave-7cd9cdc695-wxnqh        1/1       Running   2          5m
splunk-5f487cbdbf-dtv8f             1/1       Running   4          4m
worker-59bbcd44c6-sd6t5             1/1       Running   0          4m
```

3. Verify Minio is Deployed

```
kubectl describe po minio | grep "Node:"
Node:               master1/192.168.0.101
```

4. Verify Ceph is Deployed

```
kubectl describe -n rook-ceph-system po rook-ceph-agent | grep "Node:"
Node:               master3/192.168.0.103
Node:               master1/192.168.0.101
Node:               master2/192.168.0.102
```

5. Verify the API is Deployed

```
kubectl describe po api | grep "Node:"
Node:               master2/192.168.0.102
Node:               master1/192.168.0.101
```

6. Verify Jupyter is Deployed

```
kubectl describe po jupyter | grep "Node:"
Node:               master2/192.168.0.102
```

7. Verify Splunk is Deployed

```
kubectl describe po splunk | grep "Node:"
Node:               master3/192.168.0.103
```

CHAPTER 73

Set up an External DNS Server for a Multi-Host Kubernetes Cluster

Now that you have a local, 3 node Kubernetes cluster, you can set up a bind9 DNS server for making the public-facing frontend nginx ingresses accessible to browsers or other clients on an internal network (like a home lab).

1. Determine the Networking IP Addresses for VMs

   For this guide the 3 vms use the included netplan yaml files for statically setting their IPs:

   - m1 with static ip: 192.168.0.101
   - m2 with static ip: 192.168.0.102
   - m3 with static ip: 192.168.0.103

   Warning: If you do not know each vm's IP address, and you are ok with having a **network sniffing tool** installed on your host like arp-scan, then you can use this command to find each vm's IP address from the vm's bridge network adapter's MAC address:

   ```
   arp-scan -q -l --interface <NIC name like enp0s3> | sort | uniq | grep -i "
   ↪<MAC address>" | awk '{print $1}'
   ```

2. Install DNS

   Pick a vm to be the primary DNS server. For this guide, I am using `master1.example.com` with IP: `192.168.0.101`.

   For DNS this guide uses the ISC BIND server. Here is how to install BIND on CentOS 7:

   ```
   sudo apt install -y bind9 bind9utils bind9-doc dnsutils
   ```

3. Build the Forward Zone File

   Depending on how you want your Kubernetes affinity (decision logic for determining where applications are deployed) the forward zone will need to have the correct IP addresses configured to help maximize your available hosting resources. For example, I have my `master1.example.com` vm with 3 CPU cores after noticing how much the original 2 cores were being 100% utilized.

The included forward zone file uses the `example.com` domain outlined below and needs to be saved as the `root` user to the location:

```
/etc/bind/fwd.example.com.db
```

Based off the original `/etc/hosts` file from above, my forward zone file looks like:

```
;
; BIND data file for example.com
;
$TTL    604800
@   IN  SOA example.com. root.example.com. (
                20     ; Serial
            604800     ; Refresh
            86400     ; Retry
            2419200     ; Expire
            604800 )   ; Negative Cache TTL
;
;@  IN  NS  localhost.
;@  IN  A   127.0.0.1
;@  IN  AAAA    ::1

;Name Server Information
        IN      NS      ns1.example.com.
;IP address of Name Server
ns1     IN      A       192.168.0.101

;Mail Exchanger
example.com.   IN     MX   10   mail.example.com.

;A - Record HostName To Ip Address
@         IN       A      192.168.0.101
api       IN       A      192.168.0.101
ceph      IN       A      192.168.0.101
master1   IN       A      192.168.0.101
mail      IN       A      192.168.0.101
minio     IN       A      192.168.0.101
pgadmin   IN       A      192.168.0.101
www       IN       A      192.168.0.101
api       IN       A      192.168.0.102
jenkins   IN       A      192.168.0.102
jupyter   IN       A      192.168.0.102
aejupyter  IN       A      192.168.0.102
master2   IN       A      192.168.0.102
master3   IN       A      192.168.0.103
splunk    IN       A      192.168.0.103
```

Note: The API has two A records for placement on two of the vms `192.168.0.103` and `192.168.0.102`

4. Verify the Forward Zone File

```
named-checkzone example.com /etc/bind/fwd.example.com.db
zone example.com/IN: loaded serial 20
OK
```

5. Build the Reverse Zone File

Depending on how you want your Kubernetes affinity (decision logic for determining where applications are deployed) the reverse zone will need to have the correct IP addresses configured to help maximize your available hosting resources.

The included reverse zone file uses the `example.com` domain outlined below and needs to be saved as the `root` user to the location:

```
/etc/bind/rev.example.com.db
```

Based off the original `/etc/hosts` file from above, my reverse zone file looks like:

```
;
; BIND reverse zone data file for example.com
;
$TTL    604800
@   IN  SOA example.com. root.example.com. (
                20     ; Serial
            604800     ; Refresh
            86400     ; Retry
            2419200     ; Expire
            604800 )   ; Negative Cache TTL
;
;@  IN  NS  localhost.
;1.0.0  IN  PTR localhost.

;Name Server Information
        IN      NS     ns1.example.com.
;Reverse lookup for Name Server
101     IN      PTR    ns1.example.com.
;PTR Record IP address to HostName
101     IN      PTR    api.example.com.
101     IN      PTR    example.com
101     IN      PTR    ceph.example.com.
101     IN      PTR    mail.example.com.
101     IN      PTR    master1.example.com.
101     IN      PTR    minio.example.com.
101     IN      PTR    pgadmin.example.com.
101     IN      PTR    www.example.com.
102     IN      PTR    api.example.com.
102     IN      PTR    jupyter.example.com.
102     IN      PTR    aejupyter.example.com.
102     IN      PTR    jenkins.example.com.
102     IN      PTR    master2.example.com.
103     IN      PTR    master3.example.com.
103     IN      PTR    splunk.example.com.
```

Note: The API has two A records for placement on two of the vms `101` and `102`

6. Verify the Reverse Zone File

```
named-checkzone 0.168.192.in-addr.arpa /etc/bind/rev.example.com.db
zone 0.168.192.in-addr.arpa/IN: loaded serial 20
OK
```

7. Restart and Enable Bind9 to Run on VM Restart

```
systemctl restart bind9
systemctl enable bind9
```

8. Check the Bind9 status

```
systemctl status bind9
```

9. From another host set up the Netplan yaml file

   Here is the `192.168.0.101` vm's `/etc/sysconfig/network-scripts/ifcfg-eth0` network interface file that uses the external BIND server for DNS. Please edit this file as root and according to your vm's networking IP address and static vs dhcp requirements.

```
/etc/sysconfig/network-scripts/ifcfg-eth0
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
BOOTPROTO="none"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
IPV6_ADDR_GEN_MODE="stable-privacy"
NAME="eth0"
UUID="747d880d-0c18-5a9f-c0a5-e9e80cd6be46"
DEVICE="eth0"
ONBOOT="yes"
IPADDR="192.168.0.101"
PREFIX="24"
GATEWAY="192.168.0.1"
DNS1="192.168.0.100"
DNS2="8.8.8.8"
DNS3="8.8.4.4"
IPV6_PRIVACY="no"
```

10. Verify the Cluster DNS Alias Records

    The Django REST API web application has two alias records:

```
dig api.example.com | grep IN | tail -2
api.example.com.        7193    IN      A       192.168.0.101
api.example.com.        7193    IN      A       192.168.0.102
```

    Rook Ceph dashboard has one alias record:

```
dig ceph.example.com | grep IN | tail -1
ceph.example.com.       604800  IN      A       192.168.0.101
```

    Minio S3 has one alias record:

```
dig minio.example.com | grep IN | tail -1
minio.example.com.      604800  IN      A       192.168.0.101
```

    Jupyter has one alias record:

```
dig jupyter.example.com | grep IN | tail -1
jupyter.example.com.    604800  IN      A       192.168.0.102
```

pgAdmin has one alias record:

```
dig pgadmin.example.com | grep IN | tail -1
pgadmin.example.com.    604800  IN      A       192.168.0.101
```

The Kubernetes master 1 vm has one alias record:

```
dig master1.example.com | grep IN | tail -1
master1.example.com.    7177    IN      A       192.168.0.101
```

The Kubernetes master 2 vm has one alias record:

```
dig master2.example.com | grep IN | tail -1
master2.example.com.    604800  IN      A       192.168.0.102
```

The Kubernetes master 3 vm has one alias record:

```
dig master3.example.com | grep IN | tail -1
master3.example.com.    604800  IN      A       192.168.0.103
```

CHAPTER 74

Start using the Stack

With the DNS server ready, you can now migrate the database and create the first user `trex` to start using the stack.

74.1 Run a Database Migration

Here is a video showing how to apply database schema migrations in the cluster:

To apply new Django database migrations, run the following command:

```
# from /opt/deploy-to-kubernetes
./api/migrate-db.sh
```

74.2 Create a User

Create the user `trex` with password `123321` on the REST API.

```
./api/create-user.sh
```

CHAPTER 75

Deployed Web Applications

Once the stack is deployed, here are the hosted web application urls. These urls are made accessible by the included nginx-ingress.

CHAPTER 76

View Django REST Framework

Login with:

- user: `trex`
- password: `123321`

https://api.example.com

CHAPTER 77

View Swagger

Login with:

- user: `trex`
- password: `123321`

https://api.example.com/swagger

CHAPTER 78

View Jupyter

Login with:

- password: `admin`

https://jupyter.example.com

CHAPTER 79

View pgAdmin

Login with:

- user: `admin@admin.com`
- password: `123321`

https://pgadmin.example.com

CHAPTER 80

View Minio S3 Object Storage

Login with:

- access key: `trexaccesskey`
- secret key: `trex123321`

https://minio.example.com

CHAPTER 81

View Ceph

https://ceph.example.com

CHAPTER 82

View Splunk

Login with:

- user: `trex`
- password: `123321`

https://splunk.example.com

CHAPTER 83

Train AI with Django REST API

Please refer to the Training AI with the Django REST API for continuing to examine how to run a distributed AI stack on Kubernetes.

CHAPTER 84

Next Steps

- Add Heptio's Ark for disaster recovery
- Add Jenkins into the stack using Helm

84.1 More Information

After seeing high CPU utilization across the cluster, this guide was moved from Ubuntu 18.04 vms to CentOS 7.

CHAPTER 85

AntiNex Stack Status

Here are the AntiNex repositories, documentation and build reports:

Component	Build	Docs Link	Docs Build
REST API		Docs	
Core Worker		Docs	
Network Pipeline		Docs	
AI Utils		Docs	
Client		Docs	