Minimal docs Documentation

Release 0.1.0

Carol Willing

Mar 10, 2018

Zero to JupyterHub

Though I created these cheatsheets based on the "official" JupyterHub documentation, these cheatsheets are designed for my personal use.

> **Warning:** **Use at your own peril.** When in doubt, the "official" docs are the definitive source of truth.

CHAPTER 1

Cheatsheet: Create k8s cluster

1.1 Google Cloud

1. Log in to https://console.cloud.google.com
2. Enable the Container Engine API.
3. Install gcloud SDK
4. From terminal:

```
# Install kubectl
gcloud components install kubectl

# create a K8s cluster on Google Cloud
gcloud container clusters create <YOUR_CLUSTER> \
    --num-nodes=3 \
    --machine-type=n1-standard-2 \
    --zone=us-central1-b

# test if your cluster is initialized
kubectl get node

# give your account super-user permissions
kubectl create clusterrolebinding cluster-admin-binding \
    --clusterrole=cluster-admin \
    --user=<your-email-address>
```

1.2 Microsoft Azure

1. Install and initialize the **Azure command-line tools** from azure-cli github repo.
2. From terminal:

```
# authenticate azure account
az login

# specify an Azure resource group and create if needed
# if needed, view valid locations: az account list-locations
export RESOURCE_GROUP=<YOUR_RESOURCE_GROUP>
export LOCATION=<YOUR_LOCATION>
az group create --name=${RESOURCE_GROUP} --location=${LOCATION}

# install kubectl
az acs kubernetes install-cli

# create k8s cluster
export CLUSTER_NAME=<YOUR_CLUSTER_NAME>
export DNS_PREFIX=<YOUR_PREFIX>
az acs create --orchestrator-type=kubernetes \
  --resource-group=${RESOURCE_GROUP} \
  --name=${CLUSTER_NAME} \
  --dns-prefix=${DNS_PREFIX}

# authenticate kubectl
az acs kubernetes get-credentials \
  --resource-group=${RESOURCE_GROUP} \
  --name=${CLUSTER_NAME}

# test if your cluster is initialized
kubectl get node
```

CHAPTER 2

Cheatsheet: Setting up Helm

From terminal:

```
# run Helm's installer script
curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get | bash

# initialize helm on your K8s cluster (run only once per K8s cluster)
      kubectl --namespace kube-system create sa tiller
      kubectl create clusterrolebinding tiller \
          --clusterrole cluster-admin \
          --serviceaccount=kube-system:tiller
      helm init --service-account tiller

# verify versions >= 2.4.1
helm version
```

CHAPTER 3

Cheatsheet: Setting up JupyterHub

3.1 Prepare configuration file `config.yaml`

1. Create a file called `config.yaml`.
2. Create two random hex strings to use as security tokens:

```
openssl rand -hex 32
openssl rand -hex 32
```

3. Insert these lines into the `config.yaml` file. Substitute each occurrence of `RANDOM_STRING_N` below with the output of `openssl rand -hex 32`.

   config.yaml file

```
hub:
  # output of first execution of 'openssl rand -hex 32'
  cookieSecret: "RANDOM_STRING_1"
proxy:
  # output of second execution of 'openssl rand -hex 32'
  secretToken: "RANDOM_STRING_2"
```

4. Save the `config.yaml` file.

3.2 Install JupyterHub

1. From terminal:

```
# add jupyterhub helm repository
helm repo add jupyterhub https://jupyterhub.github.io/helm-chart/
helm repo update
```

(continues on next page)

(continued from previous page)

```
# install chart
helm install jupyterhub/jupyterhub \
    --version=v0.4 \
    --name=<YOUR-RELEASE-NAME> \
    --namespace=<YOUR-NAMESPACE> \
    -f config.yaml

# check hub and proxy pods are running
kubectl --namespace=<YOUR_NAMESPACE> get pod

# find IP to access JupyterHub (external IP for proxy-public service)
kubectl --namespace=<YOUR_NAMESPACE> get svc
# alternative verbose command for IP
# kubectl --namespace=<YOUR_NAMESPACE> describe svc proxy-public
```

2. To use JupyterHub:
    - enter external IP for the `proxy-public` service into a browser.
    - entering any username and password combination as JupyterHub is running with a default *dummy* authenticator

3.2.1 Notes

- `--name` For a class called *data8* you might wish set the name to **data8-jupyterhub**. Find out the name by using `helm list`.
- `--namespace` is to identify a particular application
- We recommend using the same value for `--name` and `--namespace`
- If you get a `release exists` error, then `helm delete --purge <YOUR-RELEASE-NAME>`. Reinstall by repeating this step. If it persists, `kubectl delete <YOUR-NAMESPACE>` and try again.
- If you get a time out error, add a `--timeout=SOME-LARGE-NUMBER` parameter to the `helm install` command.

CHAPTER 4

Cheatsheet: Local development using minikube

After installing minikube:

```
minikube start
eval $(minikube docker-env)  # use docker daemon inside minikube

git clone git@github.com:jupyterhub/zero-to-jupyterhub-k8s.git
cd zero-to-jupyterhub-k8s
python3 -m venv .            # create virtual environment
pip install ruamel.yaml      # install dependency

# build docker images in minikube
./build.py build

# edit minikube-conf.yaml, and, if desired, create an additional file { -f config.
↪yaml }
helm upgrade --wait --install --namespace=hub hub jupyterhub/ -f minikube-config.yaml

minikube service --namespace=hub proxy-public
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search