This runbook explains how to setup your environment to run the demos in the well-demonstrated video. *Please note: At this time, this runbook is limited to MacOS environments.*

## Kubernetes Terminology

If you want to learn more about Kubernetes objects, please see <https://kubernetes.io/docs/concepts/>

## Install Prerequisites

### Create an ssh key

This creates an ssh key that you can use to connect to your worker nodes.

ssh-keygen

Press Enter 3 times to accept the default choices

### Install kubectl

kubectl is a command-line utility for managing a Kubernetes cluster.

$ sudo curl -kLo /usr/local/bin/kubectl "https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/kubectl"

$ sudo chmod +x /usr/local/bin/kubectl

$ kubectl version --short --client

### Install the Authenticator

This binary is used by kubectl to pass your aws credentials to the Kubernetes API server.

$ sudo curl -kLo /usr/local/bin/heptio-authenticator-aws "https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/kubectl"

$ sudo chmod +x /usr/local/bin/heptio-authenticator-aws

$ sudo curl -kLo /usr/local/bin/aws-iam-authenticator https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/aws-iam-authenticator

$ sudo chmod +x /usr/local/bin/aws-iam-authenticator

### Install demo-magic (optional)

The demo makes use of demo-magic, a handy shell script that allows you to give repeatable demos from a terminal.  Press ENTER to execute each line in the scripts that are part of this well-demonstrated package.

1. Install demo-magic by cloning the GitHub repository, git clone <https://github.com/paxtonhare/demo-magic.git>
2. Copy the .sh files from the WorkDocs folder to the demo-magic directory that got created when you cloned the repository.
3. Make the .sh file executable, e.g. chmod +x <file\_name>
4. Execute the scripts in sequence starting with k8s-demo.sh

If you don't want to use demo-magic, you can type the commands into a terminal

### Install PV (required for demo-magic)

brew install pv

### install JQ

jq is a json parser.

brew install jq

## Create an EKS cluster

eksctl is a command-line utility for provisioning an EKS cluster.

1. Install the eksctl command line utility.  Instructions for installing eksctl can be found at [eksctl.io](http://eksctl.io).

i.e. “$ brew install weaveworks/tap/eksctl”

1. Create an EKS cluster.  The following command will create a basic 2 node EKS cluster in the Oregon (us-west-2) region

$ eksctl create cluster

eksctl installs kubectl and the heptio-authenticator a.k.a the aws-iam-authenticator. It also creates a kubeconfig file and places it in the ~/.kube directory.

## Create a default storage class

A storage class is a Kubernetes object that is used by a PersistentVolume (PV) or PersistentVolumeClaim (PVC) to provision different types of persistent storage volumes, such as gp2 or io1.

$ cat > storageclass.yaml << EOF  
kind: StorageClass   
apiVersion: storage.k8s.io/v1  
metadata:  
  name: gp2  
provisioner: kubernetes.io/aws-ebs  
parameters:  
  type: gp2  
reclaimPolicy: Delete  
mountOptions:  
  - debug  
EOF

Apply the manifest to your cluster.

$ kubectl apply -f storageclass.yaml

### Set GP2 as the default storage class

Default is used when no storage class is specified.

$ kubectl patch storageclass gp2 -p '{"metadata": {"annotations":{"[storageclass.kubernetes.io/is-default-class](http://storageclass.kubernetes.io/is-default-class)":"true"}}}'

## Install Helm

Helm is mechanism for packaging and deploying application onto a Kubernetes cluster.  For additional background on Helm, see <https://docs/helm.sh>.

### install from binary

1. To install the Helm client, download the binary from <https://github.com/kubernetes/helm/releases> and unpack it, e.g. tar -zxvf helm-v2.10.0-darwin-amd64.tgz
2. Move the helm binary to your path, e.g. mv darwin-amd64/helm /usr/local/bin/helm

### From homebrew

brew install kubernetes-helm

### Create a service account for tiller

Type the following into a terminal

cat > rbac-config.yaml << 'EOF'  
apiVersion: v1  
kind: ServiceAccount  
metadata:  
  name: tiller  
  namespace: kube-system  
---  
apiVersion: rbac.authorization.k8s.io/v1beta1  
kind: ClusterRoleBinding  
metadata:  
  name: tiller  
roleRef:  
  apiGroup: rbac.authorization.k8s.io  
  kind: ClusterRole  
  name: cluster-admin  
subjects:  
  - kind: ServiceAccount  
    name: tiller  
    namespace: kube-system  
EOF

Create the service account by applying the manifest

$ kubectl create -f rbac-config.yaml

Output:  
serviceaccount "tiller" created  
clusterrolebinding "tiller" created

Initialize helm

$ helm init --service-account tiller

## Tear down; uninstall & cleanup

### Delete helm chart

#List the existing helm charts

$ helm list

#Delete chart:

$ helm delete --purge <chart\_name>

### Delete service

$ kubectl delete svc/nginx-http

### Delete the cluster

#Get your cluster name

$ eksctl get cluster

#Delete the cluster

eksctl delete cluster --name <your cluster name>

### Remove local binaries

$ rm -fr /usr/local/bin/kubectl

$ rm -fr ~/.kube

$ rm -fr ~/conf.d

$ rm -fr /usr/local/bin/heptio-authenticator-aws

$ rm -fr /usr/local/bin/aws-iam-authenticator

$ brew uninstall weaveworks/tap/eksctl

$ rm -fr /usr/local/bin/eksctl