some kubectl commands:

kubectl version --short

kubectl port-forward nginx 80:80

kubectl scale deploy nginx --replicas 2

kubectl get pods -l run=nginx

kubectl label node "nodename" demoserver=true

kubectl get node "nodename" --show-labels

kubectl exec -it podname -- sh

kubectl cluster-info

kubectl -n 'namespace name' get pods

kubectl drain 'nodename' (to put the host in maintenance mode)

kubectl config view (to get the Kubernetes URL for jenkins)

\***Namespace and Context in K8 :-**

kubectl config get-contexts

kubectl config set-context kubesys --namespace=kube-system --user=kubernetes-admin --cluster=kubernetes

where,

kubesys is context name

kubectl config current-context

kubectl config use-context kubesys

Kubectl get secrets

kubectl get secret 'secretname' -o yaml (to get output in yaml format)

kubectl create secret generic 'secretname' --from-literal=username=kubeadmin --from-literal=password=mypassword

kubectl get configmap/cm

kubectl create configmap demo-configmap-1 --from-literal=channel.name=justmeandopensource --from-literal=owner="faraz"

echo -n 'kubeadmin' | base64 (encrypted username and password you will get with this)

sudo snap install microk8s --classic (to create single node k8 cluster)

apiVersion: apps/v1 (need to use if your cluster is above 1.16 version)

**To install nfs in linux for persistent volume practice:-**

sudo pacman -S nfs-utils

sudo systemctl enable nfs-server

sudo systemctl start nfs-server

sudo mkdir -p /srv/nfs/kubedata

sudo chmod -R 777 /srv/nfs/kubedata

sudo vi /etc/exports ---> add /srv/nfs/kubedata \*(rw,sync,no\_subtree\_check,insecure)

sudo exportfs -v

kubectl get pv

**Gcloud Commands :-**

gcloud compute ssh masterip

gcloud beta container clusters update cluster-1 --zone us-central1-b --enable-autoprovisioning --min-cpu 2 --min-memory 1 --max-cpu 4 --max-memory 4

gcloud beta container clusters update cluster-1 --zone us-central1-b --node-pool default-pool --num-nodes 2

**command to create persistent disk :-**

gcloud compute disks create –size=1GiB –zone=us-central1-a mongodb

---------------------------------------------------------------------------------------------------------------------------------------

**Creating Configmap using below yaml or command**

apiVersion: v1

kind: ConfigMap

metadata:

name: demo-configmap

data

channel.name: "justmeandopensource"

channel.owner: "Venkat Nagappan"

kubectl create configmap demo-configmap-1 --from-literal=channel.name=justmeandopensource --from-literal=owner="faraz"

------------------------------------------------------------------------------------------------------------------------------------------

**Creating secret using below yaml or command**

apiVersion: v1

kind: Secret

metadata:

name: secret-demo

type: Opaque

data:

username:

password:

kubectl create secret generic 'secretname' --from-literal=username=kubeadmin --from-literal=password=mypassword

------------------------------------------------------------------------------------------------------------------------------------------

apiversion v1

kind pod

metadata

name nginx

spec

containers

- image nginx

name nginx

------------------------------------------------------------------------------------------------------------------------------------------

apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2

kind: Deployment

metadata:

name: nginx-deployment

spec:

selector:

matchLabels:

app: nginx

replicas: 2 # tells deployment to run 2 pods matching the template

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.7.9

ports:

- containerPort: 80

targetPort: http

protocol: TCP

name: http

------------------------------------------------------------------------------------------------------------------------------------------

apiVersion: extensions/v1beta1

kind: deployment

metadata:

labels:

run: nginx

name: nginx-deployment

spec:

replicas: 2

selector:

matchLabels:

run: nginx

template:

metadata:

labels:

run: nginx

spec:

containers:

- image: nginx

name: nginx

nodeSelector: (if you want to deploy app on any selected node)

demoserver: "true" (label which you have given for node)

------------------------------------------------------------------------------------------------------------------------------------------

apiVersion: extensions/v1beta1

kind: ReplicaSet

metadata:

labels:

run: nginx

name: nginx-replicaset

spec:

replicas: 2

selector:

matchLabels:

run: nginx

template:

metadata:

labels:

run: nginx

spec:

containers:

- image: nginx

name: nginx

------------------------------------------------------------------------------------------------------------------------------------------ apiVersion: extensions/v1beta1

kind: Deployment

metadata:

labels:

run: nginx

name: nginx-deployment

spec:

replicas: 2

selector:

matchLabels:

run: nginx

template:

metadata:

labels:

run: nginx

spec:

containers:

- image: nginx

name: nginx

---

apiVersion: v1

kind: Service

metadata:

name: nginx-deployment

namespace: default

labels:

app: nginx

spec:

type: NodePort

ports:

- name: http

port: 80

protocol: TCP

targetPort: 80

------------------------------------------------------------------------------------------------------------------------------------------

Go to Kubernetes.io and check things there

apiVersion: batch/v1

kind: Job

metadata:

name: helloworld

spec:

completions: 2 ( ye non-parallel hai, agar ye line add krt hain to ek pod bnaega jb wo complete ho jaega to same dusra bnaega sequential one after another)

parallelism: 2 ( 2 iteration of this job, max it can run 2 jobs in parallel, agar 10 dete hain to 5 iteration ek sath run krega )

backoffLimit: 2 ( agar fail job hoga to 3 bar fail hone pe container nai bnaega wahin ruk jaega. )

activeDeadLineSeconds: 10 ( 10 sec se zada ye job nai chlega chahe jo ho, container terminate ho jaega )

template:

spec:

containers:

- image: busybox

name: busybox

command: ["echo", "hello kubernetes !!!"] / "sleep", "60" - (ye command alag hai)

restartPolicy: never

----------------------------------------------------------------------------------------------------------------------------------------

apiVersion: batch/v1

kind: CronJob

metadata:

name: helloworld-cron

spec: ( this is job specification )

schedule: "\*\*\*\*\*" (month, year, hour sec those things wecan set here) google pe wiki cron check kro

suspend: false/true (to suspend and run the cron job)

concurrencyPolicy: allow(default)/forbid/replace

jobTemplate:

spec:

template:

spec:

containers:

- image: busybox

name: busybox

command: ["echo", "hello kubernetes !!!"] / "sleep", "60" - (ye command alag hai)

restartPolicy: never

command to resume the cronjob from suspend state: #kubectl patch cronjob helloworld-cron -p '{"spec":{"suspend":false}}

------------------------------------------------------------------------------------------------------------------------------------------

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

labels:

run: nginx

name: nginx-deploy

spec:

replicas: 1

selector:

matchLabels:

run: nginx

template:

metadata:

labels:

run: nginx

spec: (pod specification)

volumes:

- name: shared-volume

emptyDir: {} (volume type)

initContainers

- name: busybox

image: busybox

volumeMounts:

- name: shared-volume

mountPath: /nginx-data

command: ["/bin.bash"]

args: ["-c", "echo '<h1>Hello kubernetes<h1/>' > /nginx-data/index.html"]

containers:

-image: nginx

name: nginx

====================================================================================

**"Example yaml to mention secret and configmap in POD"**

apiVersion: v1

kind: Pod

metadata:

name: busybox

spec:

containers:

- image: busybox

name: busybox

command: ["/bin/sh"]

args: ["-c", "sleep 600"]

env:

- name: myusername - name: CHANNELOWNER

valueFrom: valueFrom:

secretKeyRef: configMapKeyRef:

name: Secret-demo name: demo-configmap

key: username key: channel.owner

===================================================================================

**Cluster rolling update yaml and with command :-**

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

annotations:

kubernetes.io/change-cause: "Updated to latest nginx docker version"

labels:

run: nginx

name: nginx-deploy

spec:

replicas: 4

selector:

matchLabels:

run: nginx

strategy:

type: RollingUpdate type: recreate (if you are doing update activity in dev environment)

rollingUpdate:

maxSurge: 0

maxUnavailable: 1

minReadySeconds: 5

revisionHistoryLimit: 10

template

metadata:

label:

run: nginx

spec:

containers:

- image: nginx:1.14

name: nginx

# kubectl set image deployment nginx-deploy nginx=nginx:1.15 --record (record= full command in rollout history)

# kubectl rollout status deployment nginx-deploy

# kubectl rollout history deployment nginx-deploy --revision 1/2/3

# kubectl annotate deployment nginx-deploy kubernetes.io/change-cause="Updated to version 1.14.2" (to get change cause in rollout history)

# kubectl rollout undo deployment nginx-deploy --to-revision=2 (to revert rollout on any version you want)

# kubectl rollout pause/resume deployment nginx-deploy (to pause rolling update in between)

===================================================================================

**Example for Persistent Volume in yaml :-**

apiVersion: v1

kind: PersistentVolume

metadata:

labels:

type: local

name: pv-nfs-pv1

spec:

storageClassName: manual

capacity:

storage: 1Gi

accessModes:

- ReadWriteMany:

nfs:

server: <nfs server ip>

path: "/srv/nfs/kubedata"

apiVersion: v1

kind: PersistentColumeClaim

metadata:

name: pvc-nfs-pv1

spec:

storageClassName: manual

accessModes:

- ReadWriteMany

resources:

requests:

storage: 500Mi

**how to use above in pods:-**

spec:

volumes:

- name: www

persistentVolumeClaim:

claimName: pvc-nfs-pv1 (efs-storage-claim-jenkins)

containers:

- image: nginx

name: nginx

volumeMounts:

- name: www

mountPath: /usr/share/nginx/html

===================================================================================

Rancher tutorial :-

docker run -itd --restart=unless-stopped -p 80:80 -p 443:443 -v /opt/rancher:/var/lib/rancher rancher/rancher:latest

kubectl -n cattle-prometheus get all

kubectl -n cattle-prometheus edit svc access-grafana

kubectl -n cattle-prometheus edit svc access-prometheus

https://hooks.slack.com/services/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 – slackwebhook

kind: Deployment

metadata:

labels:

app: rancher

name: rancher-deployment

namespace: cattle-system

spec:

replicas: 1

selector:

matchLabels:

app: rancher

template:

metadata:

labels:

app: rancher

spec:

containers:

- image: rancher/rancher:latest

name: rancher

ports:

- containerPort: 80

volumeMounts:

- name: rancher

mountPath: /var/lib/rancher

volumes:

- name: rancher

hostPath:

path: /opt/rancher

type: DirectoryOrCreate

nodeSelector:

assign: rancher

---

apiVersion: v1

kind: Service

metadata:

name: rancher-deployment

namespace: cattle-system

labels:

app: rancher

spec:

type: NodePort

ports:

- name: http

port: 80

protocol: TCP

targetPort: 80

nodePort: 31612

selector:

app: rancher

**HPA.YAML**

{{- if .Values.autoscaling.enabled }}

apiVersion: autoscaling/v2beta1

kind: HorizontalPodAutoscaler

metadata:

name: {{ include "atlas.fullname" . }}

labels:

{{- include "atlas.labels" . | nindent 4 }}

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: {{ include "atlas.fullname" . }}

minReplicas: {{ .Values.autoscaling.minReplicas }}

maxReplicas: {{ .Values.autoscaling.maxReplicas }}

metrics:

{{- if .Values.autoscaling.targetCPUUtilizationPercentage }}

- type: Resource

resource:

name: cpu

targetAverageUtilization: {{ .Values.autoscaling.targetCPUUtilizationPercentage }}

{{- end }}

{{- if .Values.autoscaling.targetMemoryUtilizationPercentage }}

- type: Resource

resource:

name: memory

targetAverageUtilization: {{ .Values.autoscaling.targetMemoryUtilizationPercentage }}

{{- end }}

{{- end }}

**SERVICE ACCOUNT.YAML**

{{- if .Values.serviceAccount.create -}}

apiVersion: v1

kind: ServiceAccount

metadata:

name: {{ include "atlas.serviceAccountName" . }}

labels:

{{- include "atlas.labels" . | nindent 4 }}

{{- with .Values.serviceAccount.annotations }}

annotations:

{{- toYaml . | nindent 4 }}

{{- end }}

{{- end }}

**SERVICE.YAML**

apiVersion: v1

kind: Service

metadata:

name: {{ include "atlas.fullname" . }}

labels:

{{- include "atlas.labels" . | nindent 4 }}

spec:

type: {{ .Values.service.type }}

ports:

- port: {{ .Values.service.port }}

targetPort: http

protocol: TCP

name: http

selector:

{{- include "atlas.selectorLabels" . | nindent 4 }}

**DEPLOYMENT.YAML**

apiVersion: apps/v1

kind: Deployment

metadata:

name: {{ include "atlas.fullname" . }}

labels:

{{- include "atlas.labels" . | nindent 4 }}

spec:

{{- if not .Values.autoscaling.enabled }}

replicas: {{ .Values.replicaCount }}

{{- end }}

selector:

matchLabels:

{{- include "atlas.selectorLabels" . | nindent 6 }}

template:

metadata:

{{- with .Values.podAnnotations }}

annotations:

{{- toYaml . | nindent 8 }}

{{- end }}

labels:

{{- include "atlas.selectorLabels" . | nindent 8 }}

spec:

{{- with .Values.imagePullSecrets }}

imagePullSecrets:

{{- toYaml . | nindent 8 }}

{{- end }}

serviceAccountName: {{ include "atlas.serviceAccountName" . }}

securityContext:

{{- toYaml .Values.podSecurityContext | nindent 8 }}

containers:

- name: {{ .Chart.Name }}

securityContext:

{{- toYaml .Values.securityContext | nindent 12 }}

image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"

imagePullPolicy: {{ .Values.image.pullPolicy }}

ports:

- name: http

containerPort: 7070

protocol: TCP

# livenessProbe:

# httpGet:

# path: /

# port: http

# readinessProbe:

# httpGet:

# path: /

# port: http

resources:

{{- toYaml .Values.resources | nindent 12 }}

{{- with .Values.nodeSelector }}

nodeSelector:

{{- toYaml . | nindent 8 }}

{{- end }}

{{- with .Values.affinity }}

affinity:

{{- toYaml . | nindent 8 }}

{{- end }}

{{- with .Values.tolerations }}

tolerations:

{{- toYaml . | nindent 8 }}

{{- end }}

**VALUES.YAML**

# Default values for kl-otp-framework.

# This is a YAML-formatted file.

# Declare variables to be passed into your templates.

replicaCount: 1

image:

repository: 146776836293.dkr.ecr.ap-south-1.amazonaws.com/khosla-labs

pullPolicy: Always

# Overrides the image tag whose default is the chart appVersion.

tag: "atlas"

imagePullSecrets: []

nameOverride: ""

fullnameOverride: ""

serviceAccount:

# Specifies whether a service account should be created

create: true

# Annotations to add to the service account

annotations: {}

# The name of the service account to use.

# If not set and create is true, a name is generated using the fullname template

name: ""

podAnnotations: {}

podSecurityContext: {}

# fsGroup: 2000

securityContext: {}

# capabilities:

# drop:

# - ALL

# readOnlyRootFilesystem: true

# runAsNonRoot: true

# runAsUser: 1000

service:

type: ClusterIP

port: 7070

ingress:

enabled: false

annotations: {}

# kubernetes.io/ingress.class: nginx

# kubernetes.io/tls-acme: "true"

hosts:

- host: chart-example.local

paths: []

tls: []

# - secretName: chart-example-tls

# hosts:

# - chart-example.local

resources:

# We usually recommend not to specify default resources and to leave this as a conscious

# choice for the user. This also increases chances charts run on environments with little

# resources, such as Minikube. If you do want to specify resources, uncomment the following

# lines, adjust them as necessary, and remove the curly braces after 'resources:'.

limits:

cpu: 100m

memory: 600Mi

requests:

cpu: 50m

memory: 350Mi

autoscaling:

enabled: true

minReplicas: 1

maxReplicas: 10

targetCPUUtilizationPercentage: 50

targetMemoryUtilizationPercentage: 60

nodeSelector: {}

tolerations: []

affinity: {}

**Helm Charts practice**

Below helm usefull commands under **https://docs.helm.sh** :-

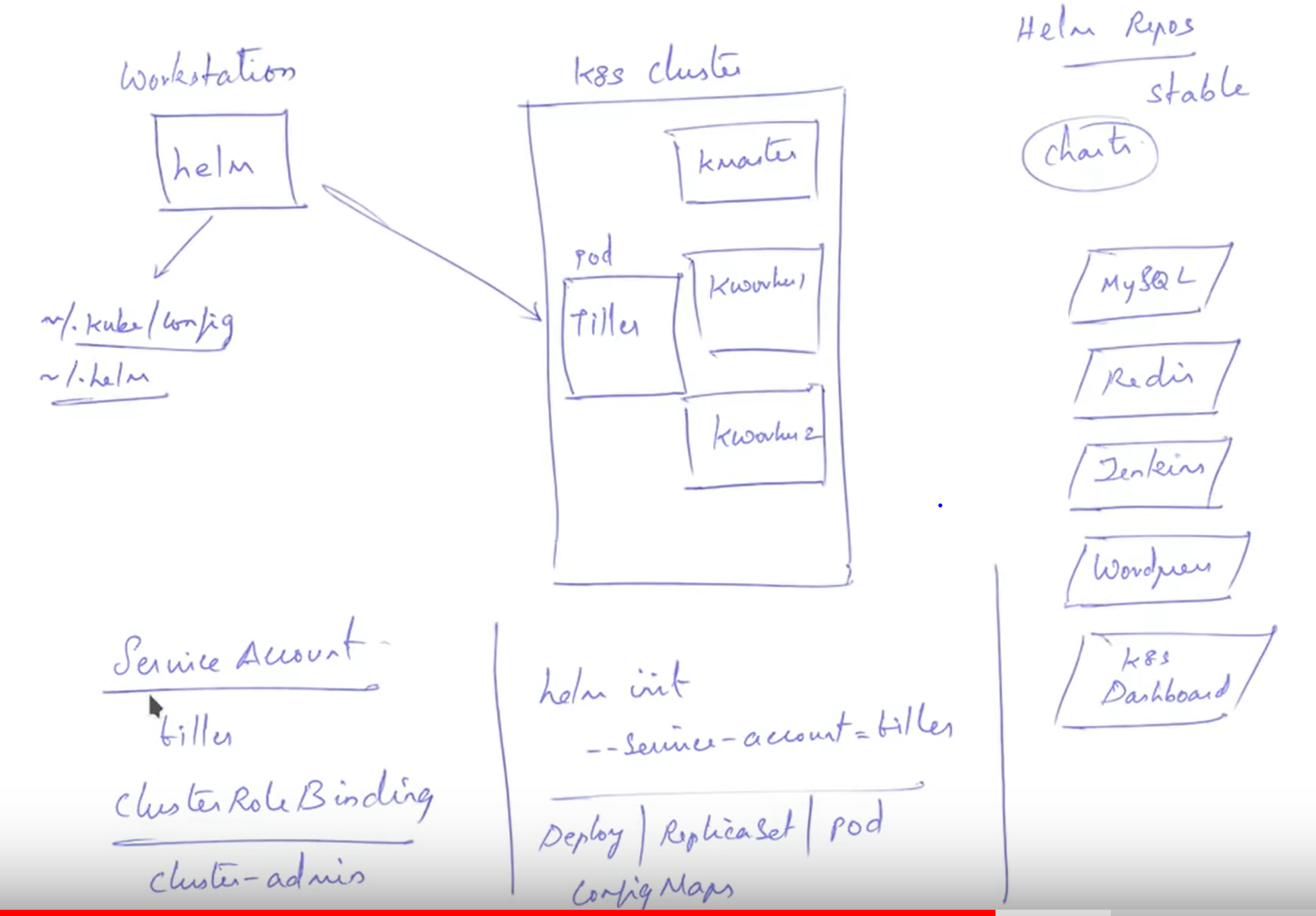
1. Helm help
2. Helm Install
3. Helm fetch
4. Helm list
5. Helm status
6. Helm search
7. Helm repo update
8. Helm upgrade
9. Helm rollback
10. Helm delete (--purge)
11. Helm reset (--force)

Tiller :- server side components for this we need service account

* Kubectl -n kube-system create serviceaccout tiller
* kubectl create clusterrolebinding tiller --clusterrole cluster-admin --serviceaccount=kube-system:tiller
* helm init--service-account tiller
* Kubectl -n kube-system get pods (you will get helm pods in kube system ns)
* Helm help
* Helm list
* Helm search stable/Jenkins
* Helm fetch Jenkins
* Tar zxf Jenkins.tar.gz
* Ls /Jenkins/templates
* Kubectl -n kube-system get deploy,replicaset,pod,serviceaccount,clusterolebinding | grep tiller
* Helm reset –remove-helm-home (to remove .helm dir and not exactly deleting tiller components from cluster)
* **To migrate chart from helm v2 to helm v3** :- copy link of helm v3 from browser
* Untar and move helm dir to /usr/local/bin/helm3
* Which helm3
* Helm3 plugin list
* Helm3 plugin install <https://github.com/helm/helm-2to3> (this plugin weneed to migrate repo)
* Helm3 2to3 --help
* Helm3 2to3 move –help
* Helm3 2to3 move config –help
* Helm3 2to3 move config –dry-run (--dry-run will tell what this command will do)
* Helm3 2to3 move config (it will migrate the repo from helm2 to 3)
* helm3.1 2to3 --help
* helm3.1 2to3 convert --help
* helm3.1 2to3 convert phpmyadmin --dry-run
* helm3.1 2to3 convert phpMyAdmin (will migrate chart from helm2 to helm3
* helm3.1 2to3 cleanup (will delete helmv2 from cluster and its tiller also)
* kubectl -n kube-system get pods
* cd /usr/local/bin/ -> rm -rf helm -> mv helm3.1 helm -> which helm -> helm list -> helm version --short
* helm uninstall phpMyAdmin
* helm repo add stable https://kubernetes-charts.storage.googleapis.com/
* helm search repo phpMyAdmin (in v3 we need to give repo after search)
* helm repo update
* helm install phpMyAdmin stable/phpMyAdmin

**How to setup Local helm chart repository**

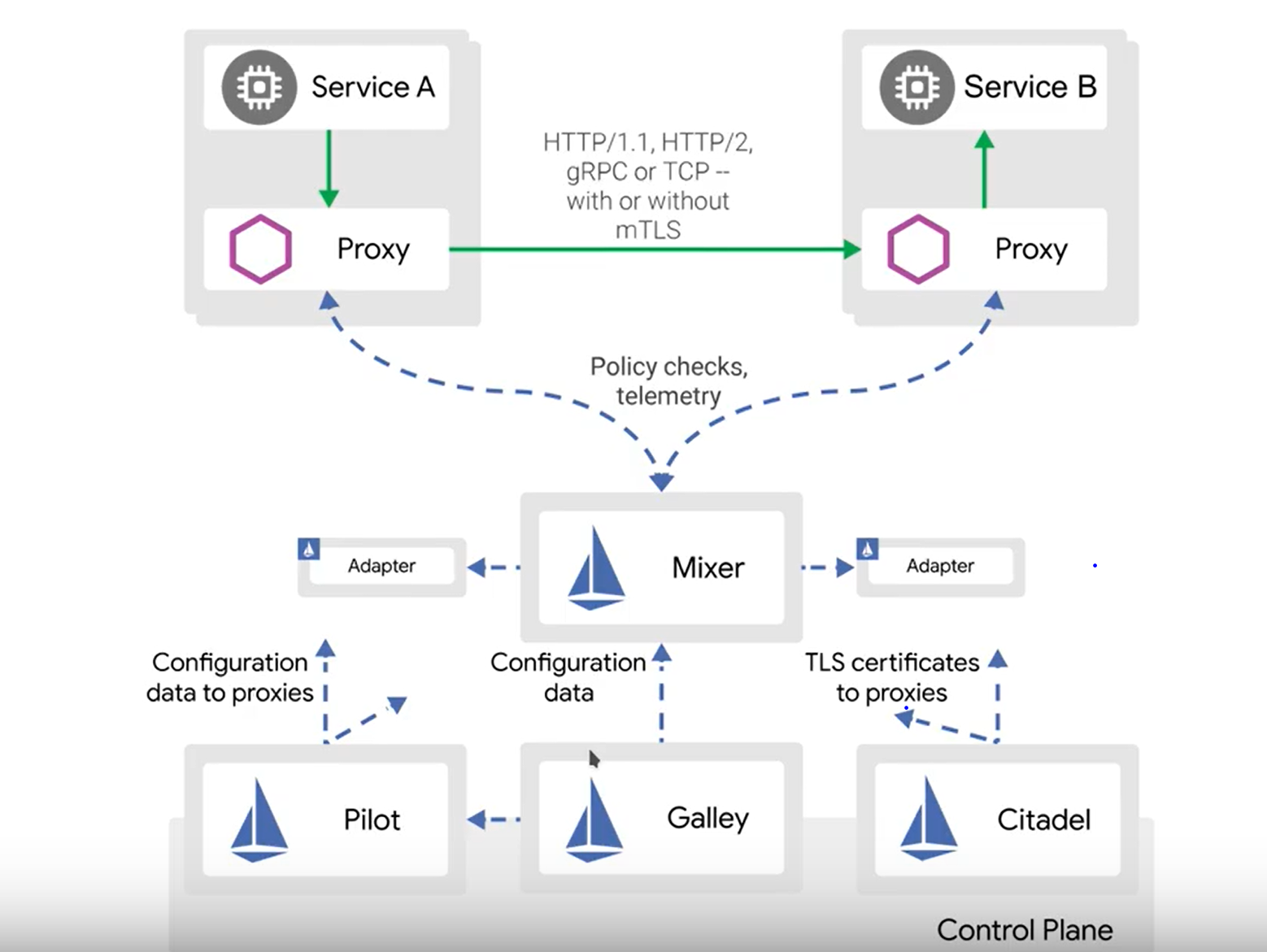
* helm package my-nginx (chartname)  to make package
* helm serve –repo-path
* Install chartmuseum using rancher
* helm plugin install <https://github.com/chartmuseum/helm-push.git>
* helm inspect values stable/Jenkins > /tmp/Jenkins.values
* helm push “chartname” chartmuseum (to upload chart in chartmuseum repo) after installing push plugins
* helm search repo chartstore/ (my chartmuseum repo name)
* curl --data-binary "@mychart-0.1.0.tgz" http://localhost:8080/api/charts



**KONTENA LENS**

1. sudo snap install kontena-lens –classic (cmd to install kontena on ubuntu)

**ISTIO in KUBERNETES**



## What's installed?

When you create or update a cluster with Istio on GKE, the following core Istio components are installed:

* [Pilot](https://istio.io/docs/concepts/traffic-management/" \l "pilot), which is responsible for service discovery and for configuring the [Envoy](https://www.envoyproxy.io/) sidecar proxies to manage your service mesh's [traffic](https://istio.io/docs/concepts/traffic-management/).
* The [Mixer](https://istio.io/docs/concepts/policy-and-control/mixer.html) components Istio-Policy and Istio-Telemetry, which enforce usage policies and gather telemetry data across the service mesh.
* The Istio [ingress gateway](https://istio.io/docs/tasks/traffic-management/ingress.html), which provides an ingress point for traffic from outside the cluster.
* **(Istio 1.0 versions only)** The Istio [egress gateway](https://istio.io/docs/tasks/traffic-management/egress.html), which allows Istio features like monitoring and routing rules to be applied to traffic exiting the mesh. The Istio egress gateway isn't installed by default in version 1.1 and later.
* [Citadel](https://istio.io/docs/concepts/security/mutual-tls.html" \l "key-management), which automates key and certificate management for Istio.
* [Galley](https://istio.io/docs/concepts/what-is-istio/" \l "galley), which provides configuration management services for Istio.

The installation also lets you add the Istio sidecar proxy to your service workloads, allowing them to communicate with the control plane and join the Istio mesh.

**WORKAROUND**

* 1. kubectl create ns istio-system
  2. helm repo add istio.io <https://storage.googleapis.com/istio-release/releases/1.1.7/charts/>
  3. cd /istio -> helm template install/kubernetes/helm/istio-init --name istio-init --namespace istio-system | kubectl apply -f –
  4. kubectl -n istio-system wait --for=condition=complete job --all
  5. kubectl get crds | grep 'istio.io\|certmanager.k8s.io' | wc -l
  6. helm template install/kubernetes/helm/istio --name istio --namespace istio-system \ --values install/kubernetes/helm/istio/values-istio-demo.yaml | kubectl apply -f –
  7. helm install --name istio --namespace istio-system --set grafana.enabled=true istio.io/istio
  8. kubectl get svc/pods -n istio-system
  9. kubectl -n istio-system edit svc
  10. helm delete –purge istio or helm delete –purge istio-init
  11. istioctl version --remote=false

# **Deploying Istio Service Mesh in Kubernetes using Istioctl**

* 1. curl -L https://istio.io/downloadIstio | sh -
  2. mv /root/istio/bin/istio usr/local/bin
  3. istioctl verify-install -> istio profile list
  4. istioctl manifest apply --set profile=demo
  5. istioctl manifest generate --set profile=demo
  6. kubectl label namespace default istio-injection=enabled
  7. istioctl profile diff default demo (to check diff btw default and demo)
  8. istioctl manifest generate --set profile=demo > /tmp/istio.yaml
  9. istioctl verify-install -f /tmp/istio.yaml
  10. kubectl delete -f /tmp/istio.yaml
  11. gcloud compute firewall-rules create allow-gateway-http --allow tcp:$INGRESS\_PORT
  12. gcloud compute firewall-rules create allow-gateway-https --allow tcp:$SECURE\_INGRESS\_PORT

**BOOKINFO MICROSERVICE APP on ISTIO**

1. kubectl create -f samples/bookinfo/platform/kube/bookinfo.yaml
2. kubectl create -f samples/bookinfo/networking/bookinfo-gateway.yaml
3. istioctl kube-inject -f samples/sleep/sleep.yaml | kubectl apply -f – (manually inject sidecar if needed)

Command to download flannel CNI to cluster :-

kubectl apply -f <https://github.com/coreos/flannel/raw/master/Documentation/kube-flannel.yml>

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Command to download metric server :-

1. Git clone[**https://github.com/kubernetes-incubator/metrics-server**](https://github.com/kubernetes-incubator/metrics-server)
2. Cd /root/metrics-server/deploy/Kubernetes/metrics-server-deployment.yaml

containers:

- name: metrics-server

image: k8s.gcr.io/metrics-server-amd64:v0.3.6

command:

- /metrics-server

- --kubelet-insecure-tls

1. git clone <https://github.com/kubernetes/autoscaler>

==================================================================

# **KinD - Kubernetes Cluster using Docker containers**

1. Donwload docker
2. Download Golang by # **wget** [**https://dl.google.com/go/go1.14.2.linux-amd64.tar.gz**](https://dl.google.com/go/go1.14.2.linux-amd64.tar.gz)
3. Tar zxf go1.14.2.linux-amd64.tar.gz -C /usr/local/bin
4. Export PATH=$PATH:/usr/local/go/bin/
5. Open in browser <https://github.com/kubernetes-sigs/kind>
6. Run in putty **GO111MODULE="on" go get** [**sigs.k8s.io/kind@v0.7.0**](mailto:sigs.k8s.io/kind@v0.7.0)
7. export PATH=$PATH:/root/go/bin
8. which kind -> kind version -> kind help
9. kind create cluster (will create only 1 node cluster - master)
10. kubectl cluster-info -> check resources
11. kind delete cluster
12. create file kindcluster.yaml for creating 1 master and 2 worker nodes also.

kind: Cluster

apiVersion: kind.x-k8s.io/v1alpha4

nodes:

- role: control-plane

- role: worker

- role: worker

13. kind create cluster –config kindcluster.yaml

=================================================================================

# **Weave Scope - Kubernetes Visualization & Monitoring**

1. kubectl create clusterrolebinding "cluster-admin-$(whoami)" --clusterrole=cluster-admin --user="$(gcloud config get-value core/account)"
2. kubectl apply -f "https://cloud.weave.works/k8s/scope.yaml?k8s-version=$(kubectl version | base64 | tr -d '\n')" --dry-run
3. kubectl edit svc -n weave (edit to nodePort to access app on browser)
4. Play with it.

==================================================================================

# **Velero - Backup & Restore Kubernetes Cluster**

1. wget <https://github.com/vmware-tanzu/velero/releases/download/v1.3.2/velero-v1.3.2-linux-amd64.tar.gz>
2. tar zxf velero-v1.3.2-linux-amd64.tar.gz
3. cd velero-v1.3.2-linux-amd64/ -> mv velero /usr/local/bin
4. gcloud iam service-accounts list
5. gcloud iam service-accounts keys create credentials-velero --iam-account [231721000172-compute@developer.gserviceaccount.com](mailto:231721000172-compute@developer.gserviceaccount.com)
6. velero install --provider gcp --bucket velerobucket1 --secret-file ./credentials-velero --plugins velero/velero-plugin-for-gcp:v1.0.1
7. source <(velero completion bash)  to create tab completion (tab maro to option aaye)
8. kubectl -n velero get crds (customer resource deployments)
9. velero backup create firstbackup --include-namespaces testing (will take backup of all resources of testing NS)
10. velero backup create firstbackup --exclude-namespaces testing (will take backup of whole cluster excluding testing NS)
11. velero backup create firstbackup --exclude-namespaces testing –include/exclude-resources pods,deployments (will take backup of these resources only)
12. velero backup get
13. velero restore create firstbackup-restore1 --from-backup firstbackup
14. velero restore get
15. velero backup/restore describe (backup/restore name) | less
16. velero restore delete (backup/restore name)
17. **velero schedule create firstschedule --schedule="@every 1m/1h/1d/1w" --include-namespaces testing (multiple schedule option you can take)**
18. velero create backup secondbackup –include-namespaces testing –ttl 2h (will keep backup till 2 hours)
19. velero delete backup/restore –all

**=====================================================================================**

# **K9S Kubernetes Terminal Dashboard**

1. wget <https://github.com/derailed/k9s/releases/download/v0.19.1/k9s_Linux_x86_64.tar.gz>
2. tar zxf k9s\_Linux\_x86\_64.tar.gz
3. mv k9s /usr/local/bin
4. k9s hit enter

**=====================================================================================**

# **Getting started with LXC containers**

1. snap install lxd
2. which lxc/lxd
3. dpkg -l | grep lxd
4. systemctl start lxd
5. lxc config set lxcubuntu3 limits.memory 512MB

# Deploying Kubernetes Cluster using LXC Containers

1. lxd init
2. lxc list
3. lxc profile list
4. lxc profile show default
5. lxc profile copy default k8s
6. lxc profile edit k8s → add below config to file
7. lxc launch images:centos/7 kmaster --profile k8s
8. lxc launch images:centos/7 kworker1 --profile k8s
9. cat bootstrap-kube.sh | lxc exec kmaster bash

config:

limits.cpu: "2"

limits.memory: 2GB

limits.memory.swap: "false"

linux.kernel\_modules: ip\_tables,ip6\_tables,netlink\_diag,nf\_nat,overlay

raw.lxc:"lxc.apparmor.profile=unconfined\nxlc.cap.drop=\nxlc.cgroup.devices.allow=a\nlxc.mount.auto=proc:rw

sys:rw"

security.nesting: "true"

**AMAZON EKS CLUSTER**

1. apt install awscli

2. curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

3. sudo mv /tmp/eksctl /usr/local/bin → eksctl version

4. eksctl create cluster --name first-eks –region eu-west-1 --nodegroup-name standard-nodes --node-type t2.micro --managed