# Workloads

Kubernetes Objects - apiVersion, Kind, metadata, and Spec

## Pod

alias k=kubectl

k api-resources

k explain pod.spec.containers

k explain pod.spec.containers --recursive

### One-container-per-Pod

Create a file - pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

kubectl apply -f pod.yaml

kubectl get pods

kubectl describe pod nginx

# Looks for events and lifecyle

### Multi-Container-Pod

apiVersion: v1

kind: Pod

metadata:

name: multicontainer-pods

spec:

containers:

#Container 01

- name: web

image: httpd

ports:

- containerPort: 80

#Container 02

- name: redis

image: redis

### Init-Container

apiVersion: v1

kind: Pod

metadata:

name: purple

spec:

containers:

- command:

- sh

- -c

- echo The app is running! && sleep 3600

image: busybox:1.28

name: purple-container

# Adding 2 init containers to execute sleep commands

initContainers:

- command:

- sh

- -c

- sleep 60

image: busybox:1.28

name: warm-up-1

- command: ["sh", "-c", "sleep 120"]

image: busybox:1.28

name: warm-up-2

kubectl get pods -w

# Both the init containers will get executed before the main container is started

# NAME READY STATUS RESTARTS AGE

# purple 0/1 Init:1/2 0 2m41s

# After 3mins(60+120seconds), the output will be

# NAME READY STATUS RESTARTS AGE

# purple 1/1 Running 0 3m7s

### Static-Pod

In worker01,

sudo ls /etc/kubernetes/

sudo mkdir /etc/kubernetes/manifests

sudo ls /etc/kubernetes/

sudo vi /etc/kubernetes/manifests/pod1

Create a yaml file in

apiVersion: v1

kind: Pod

metadata:

name: static-web

spec:

containers:

- name: web

image: nginx

ports:

- name: web

containerPort: 80

protocol: TCP

In the master node,

k get pods -A

The pod will appear in default ns

Delete the static pod file in worker01

sudo rm /etc/kubernetes/manifests/pod1

In the master node,

k get pods -A

The pod will Disappear in default ns

### Resource Limits

apiVersion: v1

kind: Pod

metadata:

name: rl-pod

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

resources:

requests: # Minimum Value

memory: "100Mi"

cpu: "250m" # 1 core = 1000m

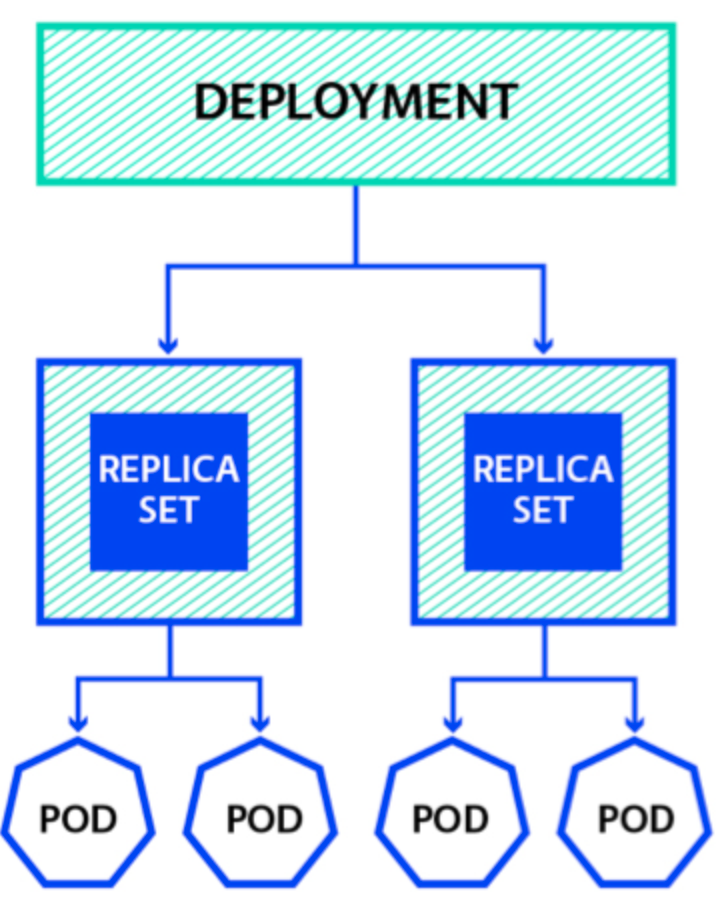
limits: # Maximum Value

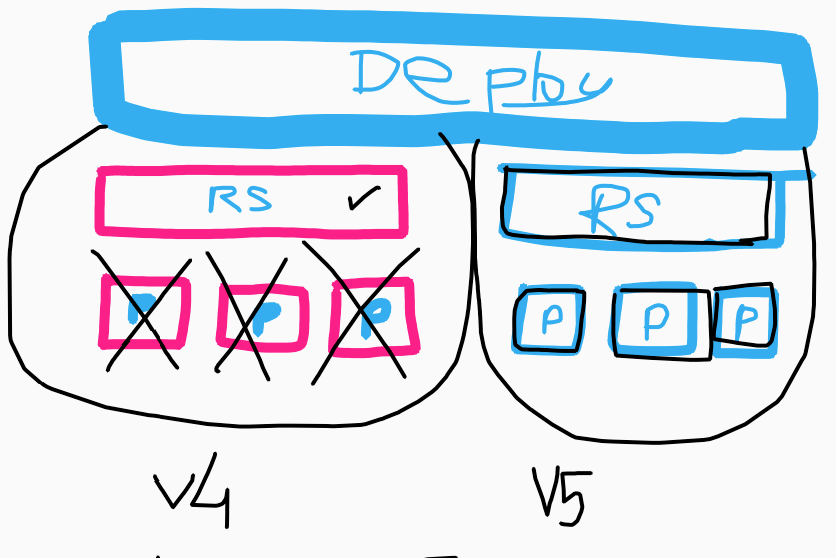
memory: "128Mi"

cpu: "300m"

k describe pod rl-pod

## Deployment





apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

kubectl apply -f deploy.yaml

kubectl get deployments

kubectl rollout status deployment nginx-deployment

kubectl get rs

### Update Deployment

kubectl set image deployment nginx-deployment nginx=nginx:1.16.1

kubectl rollout status deployment nginx-deployment

kubectl get rs

kubectl get pods | grep nginx-deployment

kubectl describe deployment nginx-deployment

Setting wrong image

kubectl set image deployment nginx-deployment nginx=nginx:xxxxxx

kubectl rollout status deployment nginx-deployment

Waiting for rollout to finish: 1 out of 3 new replicas has been updated...

kubectl get rs

kubectl get pods | grep nginx-deployment

kubectl describe deployment

kubectl rollout history deployment nginx-deployment

kubectl rollout history deployment nginx-deployment --revision=2

### Rolling Back to a Previous Revision

kubectl rollout undo deployment nginx-deployment

kubectl rollout history deployment nginx-deployment

kubectl rollout history deployment nginx-deployment --revision=4

kubectl get deployment nginx-deployment

kubectl describe deployment nginx-deployment

# Check container image version/tag

k rollout undo deployment nginx-deployment --to-revision=1

1 -> nginx:1.14.2

2 -> nginx:1.16.1

3 -> nginx:xxxx

undo = 3-1 = 2

1 -> nginx:1.14.2

3 -> nginx:xxxx

4 -> nginx:1.16.1

undo = 1

3 -> nginx:xxxx

4 -> nginx:1.16.1

5 -> nginx:1.14.2

### 

### Scaling deployment

kubectl scale deployment nginx-deployment --replicas=10

kubectl scale deployment nginx-deployment --replicas=1

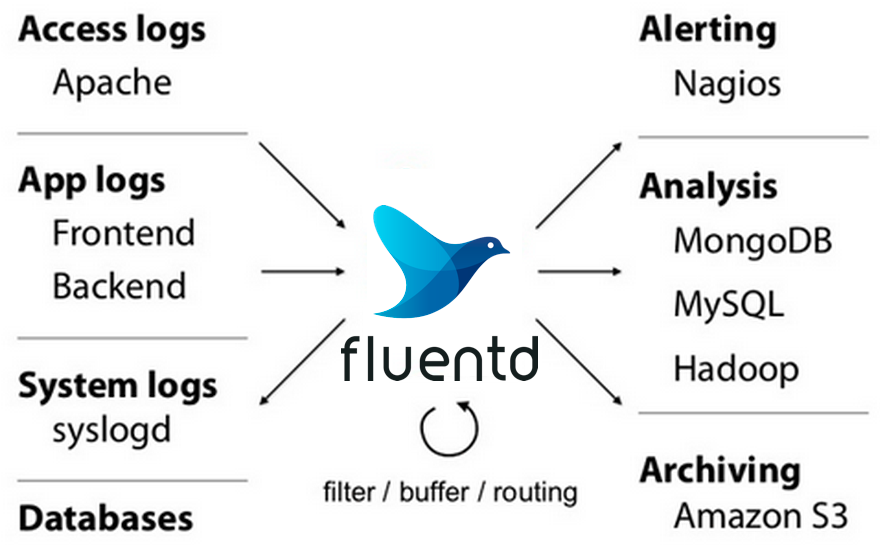
kubectl get deployment nginx-deployment

kubectl get rs

kubectl get pods | grep nginx-deployment

kubectl describe deployment nginx-deployment

## Daemon Set



kubectl apply -f https://k8s.io/examples/controllers/daemonset.yaml

kubectl get ds -n kube-system

kubectl describe ds fluentd-elasticsearch -n kube-system

kubectl get pods -o wide -n kube-system | grep fluentd

## Jobs

apiVersion: batch/v1

kind: Job

metadata:

name: pi

spec:

template:

spec:

containers:

- name: pi

image: busybox:1.28

imagePullPolicy: IfNotPresent

command:

- /bin/sh

- -c

- date; echo Hello from the Kubernetes cluster

restartPolicy: Never

backoffLimit: 4

kubectl apply -f job.yaml

kubectl describe jobs/pi

## CronJobs

apiVersion: batch/v1

kind: CronJob

metadata:

name: hello

spec:

schedule: "\* \* \* \* \*"

jobTemplate:

spec:

template:

spec:

containers:

- name: hello

image: busybox:1.28

imagePullPolicy: IfNotPresent

command:

- /bin/sh

- -c

- date; echo Hello from the Kubernetes cluster

restartPolicy: OnFailure

kubectl create -f cronjob.yaml

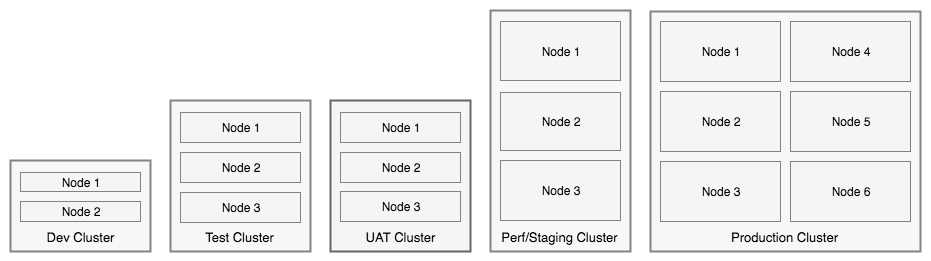
kubectl get cronjob hello

kubectl get jobs -w

kubectl delete -f cronjob.yaml

## Configuration basics

Introduction:



[All JAVA Spring Configuration item](https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html#application-properties.data.spring.datasource.username)

[MySQL DB Configuration](https://github.com/spring-guides/gs-accessing-data-mysql/blob/main/complete/src/main/resources/application.properties)

a[pplication.properties](https://github.com/spring-guides/gs-accessing-data-mysql/blob/main/complete/src/main/resources/application.properties)

spring.datasource.url=jdbc:mysql:// \

${MYSQL\_HOST:localhost}:3306/db\_example

spring.datasource.username=springuser

spring.datasource.password=ThePassword

spring.datasource.driver-class-name =com.mysql.jdbc.Driver

For each environment, we define the individual application.properties



## Env

env.yaml  
apiVersion: v1

kind: Pod

metadata:

name: envar-demo

labels:

purpose: demonstrate-envars

spec:

containers:

- name: envar-demo-container

image: gcr.io/google-samples/node-hello:1.0

env:

- name: DEMO\_GREETING

value: "Hello from the environment"

- name: DEMO\_FAREWELL

value: "Such a sweet sorrow"

kubectl apply -f env.yaml

kubectl exec envar-demo -- printenv

## ConfigMaps

config-map.yaml

apiVersion: v1

kind: ConfigMap

metadata:

name: game-demo

data:

*# property-like keys; each key maps to a simple value*

player\_initial\_lives: "3"

ui\_properties\_file\_name: "user-interface.properties"

*# file-like keys*

game.properties: |

*enemy.types=aliens,monsters*

*player.maximum-lives=5*

user-interface.properties: |

*color.good=purple*

*color.bad=yellow*

*allow.textmode=true*

k apply -f config-map.yaml

k describe cm game-demo

**Using EnvFrom**

cm-pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: dapi-test-pod02

spec:

containers:

- name: test-container

image: k8s.gcr.io/busybox

command: [ "/bin/sh", "-c", "env" ]

envFrom:

- configMapRef:

name: game-demo

restartPolicy: Never

k apply -f cm-pod.yaml

## Secrets

apiVersion: v1

kind: Secret

metadata:

name: mysecret

type: kubernetes.io/basic-auth

stringData:

username: admin

password: t0p-Secret

k apply -f secret

k describe secret mysecret

apiVersion: v1

kind: Pod

metadata:

name: dapi-test-pod03

spec:

containers:

- name: test-container

image: k8s.gcr.io/busybox

command: [ "/bin/sh", "-c", "env" ]

envFrom:

- secretRef:

name: mysecret

restartPolicy: Never

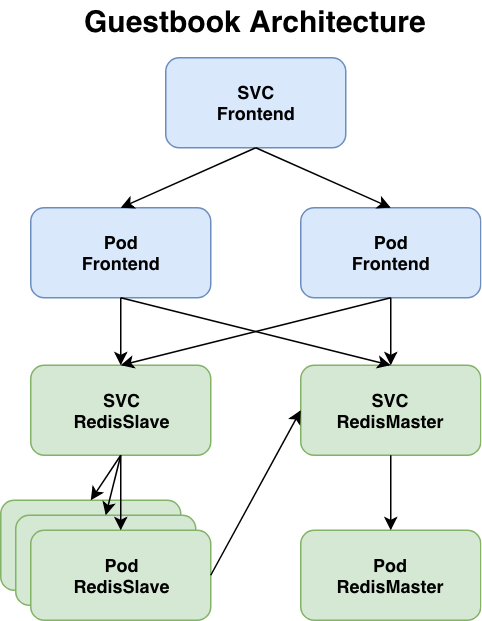
k apply -f secret-pod

k logs dapi-test-pod03

## Labels & Selectors

Check Labels of Nodes and Workloads

kubectl get pods --show-labels



wget https://raw.githubusercontent.com/sparkmbt/sparkmbt/main/kubesample.yaml

kubectl apply -f kubesample.yaml

Using Selectors,

kubectl get pods -l tier=backend

kubectl get pods -l tier=frontend

### Set-based selectors

kubectl get pods -l 'tier notin (backend, frontend)'

kubectl get pods -l 'tier in (backend), role notin (slave)'

## Monitoring Pods/Nodes: Metrics Server

kubectl apply -f<https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>

kubectl get pods -n kube-system

​​wget -c https://gist.githubusercontent.com/initcron/1a2bd25353e1faa22a0ad41ad1c01b62/raw/008e23f9fbf4d7e2cf79df1dd008de2f1db62a10/k8s-metrics-server.patch.yaml

kubectl patch deploy metrics-server -p "$(cat k8s-metrics-server.patch.yaml)" -n kube-system

kubectl get pods -n kube-system

### Metrics Server Commands

kubectl top pods --all-namespaces

kubectl top node

kubectl top node master

## Horizontal Pod Autoscaling(HPA)

kubectl autoscale deploy nginx-deployment --min=3 --max=5 --cpu-percent=40

kubectl get hpa

## Self Healing Pods - Probes: Liveness and Readiness

## Liveness Probe

## Detects when a pod enters an undesired state

## Tells kubernetes when to actually restart a pod

## Readiness Probe

## Used it check if pod is ready to serve traffic

## Readiness probes runs on the container during its whole lifecycle.

## Can be used to check startups and external dependeicies

## 

## liveness.yaml

## apiVersion: v1

## kind: Pod

## metadata:

## labels:

## test: liveness

## name: liveness-exec

## spec:

## containers:

## - name: liveness

## image: k8s.gcr.io/busybox

## args:

## - /bin/sh

## - -c

## - touch /tmp/healthy; sleep 30; rm -f /tmp/healthy; sleep 600

## livenessProbe:

## exec:

## command:

## - cat

## - /tmp/healthy

## initialDelaySeconds: 5

## periodSeconds: 5

## 

## k apply -f liveness.yaml

## For the first 30 seconds of the container's life, there is a /tmp/healthy file. So during the first 30 seconds, the command cat /tmp/healthy returns a success code. After 30 seconds, cat /tmp/healthy returns a failure code. Later, verify that the container has been restarted.

## Readiness.yaml

## apiVersion: v1

## kind: Pod

## metadata:

## labels:

## test: readiness

## name: readiness-exec

## spec:

## containers:

## - name: readiness

## image: k8s.gcr.io/busybox

## args:

## - /bin/sh

## - -c

## - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600

## readinessProbe:

## exec:

## command:

## - cat

## - /tmp/healthy

## initialDelaySeconds: 5

## periodSeconds: 5

## 

## Readiness and liveness probes can be used in parallel for the same container. Using both can ensure that traffic does not reach a container that is not ready for it, and that containers are restarted when they fail.

## labsuser@master:~/workloads$ k get pod liveness-exec -w

## NAME READY STATUS RESTARTS AGE

## liveness-exec 1/1 Running 0 4s

## liveness-exec 1/1 Running **1 (2s ago)** 77s

## 

## labsuser@master:~/workloads$ k get pod readiness-exec -w

## NAME READY STATUS RESTARTS AGE

## readiness-exec 1/1 Running 0 13s

## readiness-exec **0/1** Running 0 45s

## Namespaces

## Cluster {Worker1(8gb mem) & Worker2(8gb mem)} - 16 GB Mem(total) & 4 Cpu

## Namespace

## Fb

## Grouping Resources - {Pod, deploy, cm, secret, jobs, cj}

## RBAC - Team

## Manager & Developer - View

## DevOps - Create & View

## Quota

## Whatsapp

## Instagram

## 

## kubectl get namespace

## kubectl get ns

## 

## kubectl delete -f kubesample.yaml

## 

## kubectl create ns cube01

## kubectl create -f kubesample.yaml -n cube01

## 

## kubectl get pods -l tier=backend -n cube01

## 

## k get all -n cube01

## 

## k delete ns cube01

## 

# Helm

QuickStart: <https://helm.sh/docs/intro/quickstart/>

sudo snap install helm --classic

helm version

helm repo add bitnami https://charts.bitnami.com/bitnami

helm repo update

helm search repo bitnami

#Can view many big tools are available

Charts:

<https://github.com/bitnami/charts/tree/master/bitnami>

Nginx -<https://github.com/bitnami/charts/tree/master/bitnami/nginx/templates>

helm install test-nginx bitnami/nginx

k get deploy

nginx-1631818923

k get pod

helm uninstall nginx-1631818923

k get deploy

## Creating application deployment using Helm

Traditional Method: without using Helm

<https://github.com/IBM/guestbook/tree/master/v2>

<https://github.com/IBM/guestbook/blob/master/v2/guestbook-deployment.yaml>

Helm way:

<https://github.com/IBM/helm101/tree/master/charts/guestbook>

<https://github.com/IBM/helm101/blob/master/charts/guestbook/templates/guestbook-deployment.yaml>

git clone https://github.com/IBM/helm101.git

cd /home/labsuser/troubleshooting/helm101/charts

helm install guestbook-demo ./guestbook/

k get deploy

k get svc

k get pods

## 