

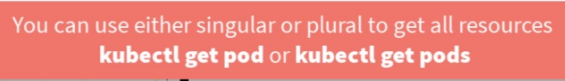
docker run -p 8080:8080 in28min/hello-world-rest-api:0.0.1.RELEASE

* Deploying application.

**kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE**

* Exposing to outside world.

**kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080**



* To see all events happening after command

**kubectl get events**

* To check all the pods getting created

**Kubectl get pods**

* To check all the replicaset getting created

**Kubectl get replicaset**

Or

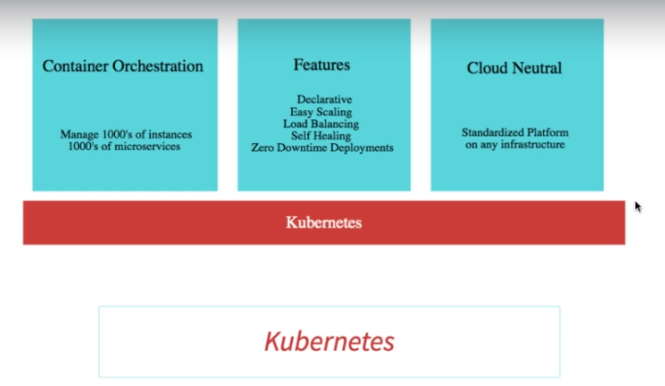
**Kubectl get rs**

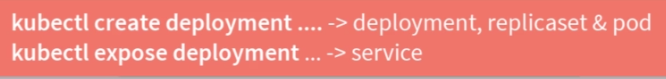
* To check all the deployment getting created

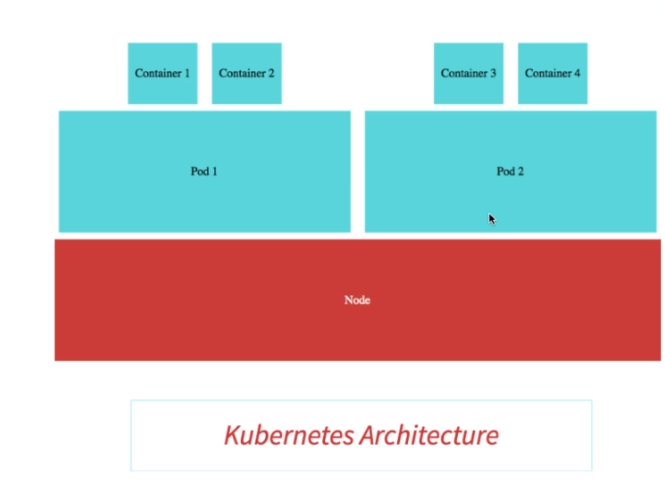
**Kubectl get deployment**

* To check all the services getting created

**Kubectl get service**

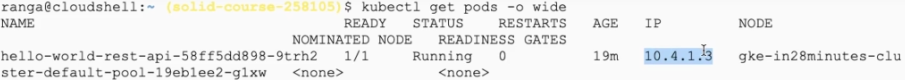






* POD is the smallest deployable unit in k8s
* Each container inside pod can talk using localhost.
* To get more details of POD

**Kubectl get pods -o wide**

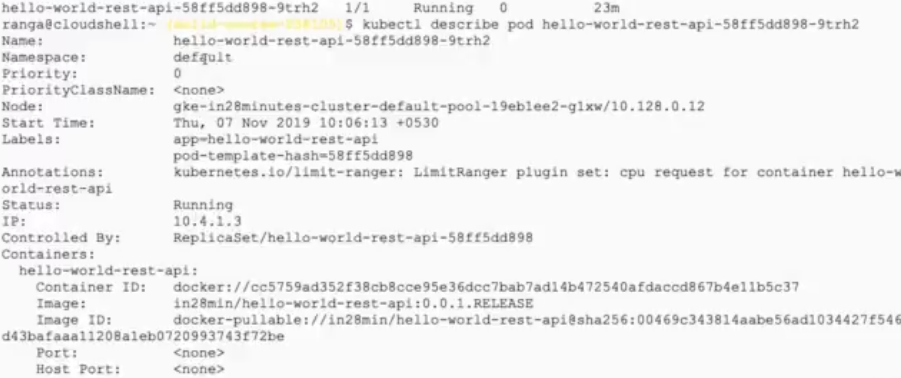


* Each pod has a unique IP address.
* 1/1 is no. of containers running.
* To get explanation of any component like pods

**Kubectl explain pods**

* To get specific pod details

**Kubectl describe pod podname**



193. Step 08 - Understanding ReplicaSets in Kubernetes

**What is the role that they play**?

1. Replica sets ensure that a specific number of pods are running at all the time.

* To create replica

**kubectl scale deployment hello-world-rest-api --replicas=3**

* To delete any pod

**kubectl delete pod hello-world-rest-api-58ff5dd898-62l9d**

* To check all events happening while cration of replicaset

**Kubectl get events - -sort-by=.metadata.creationTimestamp**

194. Step 09 - Understanding Deployment in Kubernetes

**Let's try and understand why do we need a deployment?**

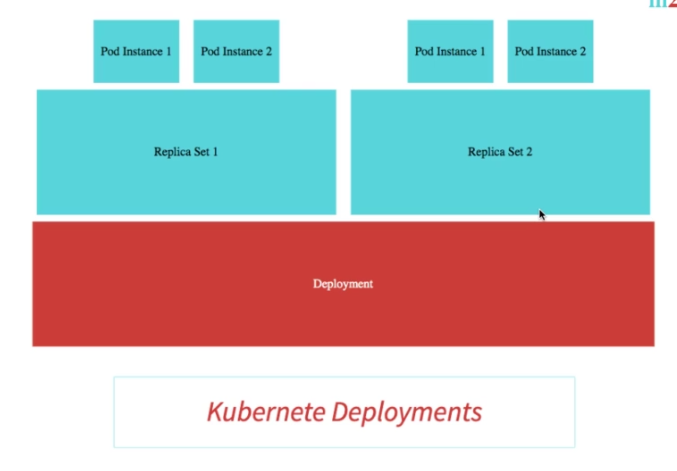
Now, let's say, I have a specific version of application deployed, let's call it V1and let's say we'd want to update to a new version,let's say, V2. Typically, what do we want when we are updating applications?We would want zero downtime.

* To see image associated with replicaset

**kubectl get rs -o wide**

* To deploy another version of application

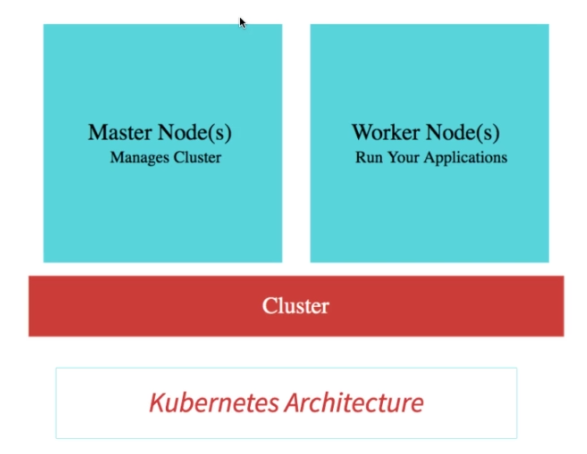
**kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE**

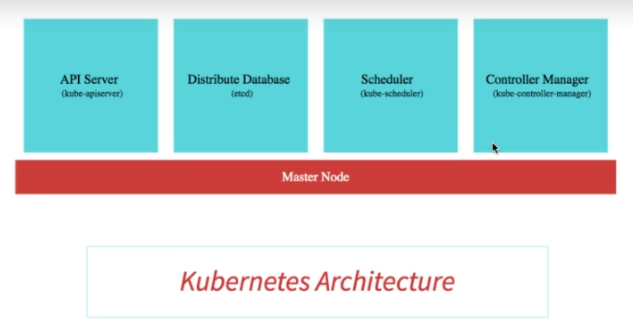


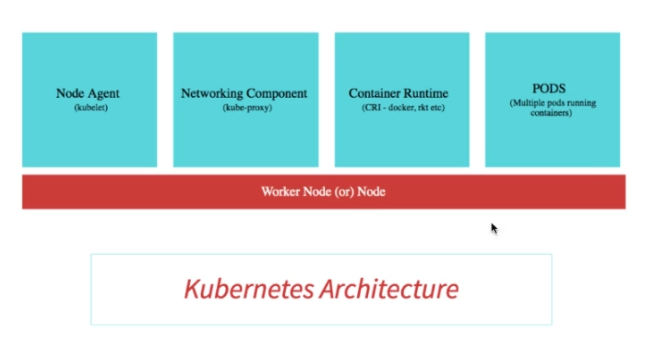
196. Step 11 - Understanding Services in Kubernetes

The role of a service is to provide an always-available external interface to the applications which are running inside the pods. A service basically allows your application to receive traffic through a permanent lifetime IP address.

198. Step 13 - Understanding Kubernetes Architecture - Master Node and Nodes

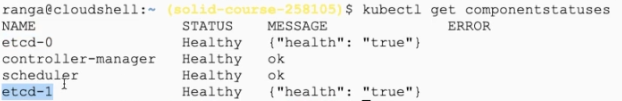






* To get status of Kubernetes component statues –

**kubectl get componentstatuses**



kubectl autoscale deployment hello-world-rest-api --max=10 --cpu-percent=70

kubectl edit deployment hello-world-rest-api #minReadySeconds: 15

gcloud container clusters get-credentials in28minutes-cluster --zone us-central1-a --project solid-course-258105

kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE

kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080

kubectl set image deployment hello-world-rest-api hello-world-rest-api=DUMMY\_IMAGE:TEST

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl get componentstatuses

kubectl get pods --all-namespaces

kubectl get events

kubectl get pods

kubectl get replicaset

kubectl get deployment

kubectl get service

kubectl get pods -o wide

kubectl explain pods

kubectl get pods -o wide

kubectl describe pod hello-world-rest-api-58ff5dd898-9trh2

kubectl get replicasets

kubectl get replicaset

kubectl scale deployment hello-world-rest-api --replicas=3

kubectl get pods

kubectl get replicaset

kubectl get events

kubectl get events --sort.by=.metadata.creationTimestamp

kubectl get rs

kubectl get rs -o wide

kubectl set image deployment hello-world-rest-api hello-world-rest-api=DUMMY\_IMAGE:TEST

kubectl get rs -o wide

kubectl get pods

kubectl describe pod hello-world-rest-api-85995ddd5c-msjsm

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl get pods -o wide

kubectl delete pod hello-world-rest-api-67c79fd44f-n6c7l

kubectl get pods -o wide

kubectl delete pod hello-world-rest-api-67c79fd44f-8bhdt

gcloud container clusters get-credentials in28minutes-cluster --zone us-central1-c --project solid-course-258105

docker login

204. Step 16 - Deploy Microservices to Kubernetes & Understand Service Discovery

docker push in28min/mmv2-currency-exchange-service:0.0.11-SNAPSHOT

docker push in28min/mmv2-currency-conversion-service:0.0.11-SNAPSHOT

kubectl create deployment currency-exchange --image=in28min/mmv2-currency-exchange-service:0.0.11-SNAPSHOT

kubectl expose deployment currency-exchange --type=LoadBalancer --port=8000

kubectl get svc

kubectl get services

kubectl get pods

kubectl get po

kubectl get replicaset

kubectl get rs

kubectl get all

kubectl create deployment currency-conversion --image=in28min/mmv2-currency-conversion-service:0.0.11-SNAPSHOT

kubectl expose deployment currency-conversion --type=LoadBalancer --port=8100

kubectl get svc --watch

kubectl get deployments

* To get existing yaml file of deployment created by Kubernetes -

kubectl get deployment currency-exchange -o yaml >> deployment.yaml

* To get existing yaml file of service created by Kubernetes -

kubectl get service currency-exchange -o yaml >> service.yaml

* To get difference of running and current YAML we are providing -

kubectl diff -f deployment.yaml

* To apply the changes -

kubectl apply -f deployment.yaml

kubectl delete all -l app=currency-exchange

kubectl delete all -l app=currency-conversion

kubectl rollout history deployment currency-conversion

kubectl rollout history deployment currency-exchange

kubectl rollout undo deployment currency-exchange --to-revision=1

kubectl logs currency-exchange-9fc6f979b-2gmn8

* To check log in pod

kubectl logs -f currency-exchange-9fc6f979b-2gmn8

* To create centralized configuration in Kubernetes –

kubectl create configmap currency-conversion --from-

literal=CURRENCY\_EXCHANGE\_URI=http://currency-exchange

kubectl get configmap

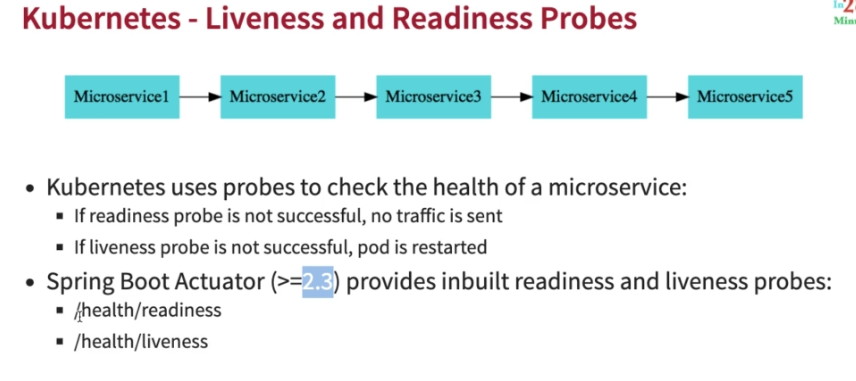
kubectl get configmap currency-conversion -o yaml >> configmap.yaml

watch -n 0.1 curl http://34.66.241.150:8100/currency-conversion-feign/from/USD/to/INR/quantity/10

docker push in28min/mmv2-currency-conversion-service:0.0.12-SNAPSHOT

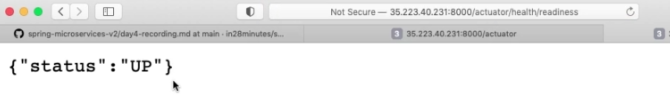
docker push in28min/mmv2-currency-exchange-service:0.0.12-SNAPSHOT

214. Step 26 - Configuring Liveness and Readiness Probes for Microservices with K8S



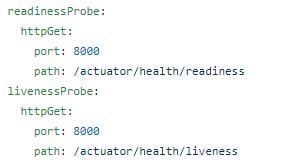
|  |
| --- |
| ## CHANGE-KUBERNETES |
|  | management.endpoint.health.probes.enabled=true |
|  | management.health.livenessState.enabled=true |
|  | management.health.readinessState.enabled=true |

* Use service ip and port to access actuator liveness and readiness to avoid down time. It will wait the containers to get ready to send traffic.



Refer

deployment-03-probes-configured.yaml



215. Step 27 - Autoscaling Microservices with Kubernetes

kubectl autoscale deployment currency-exchange --min=1 --max=3 --cpu-percent=70

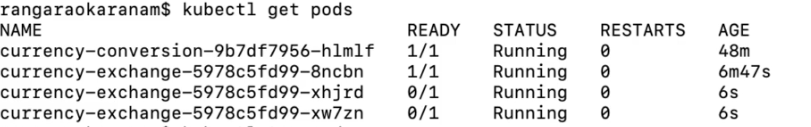
kubectl autoscale deployment currency-exchange --min=1 --max=3 --cpu-percent=5

* To see horizonal pod autoscaler created on above command -

kubectl get hpa

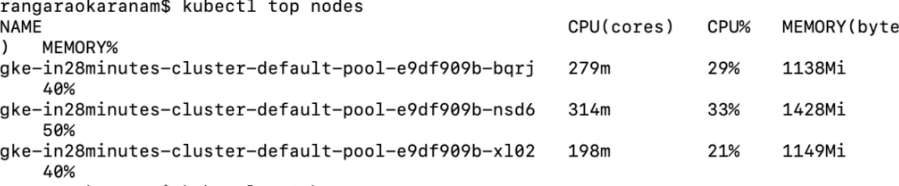
* To see cpu utilization at pod–

**kubectl top pod**



* To see cpu utilization at node–

**kubectl top nodes**



kubectl get hpa

* To delete horizonal pod autoscaler created use hpa name -

kubectl delete hpa currency-exchange

<https://github.com/in28minutes/kubernetes-crash-course/blob/master/instructions-aws-and-azure.md>