Diagram

Description automatically generated

Container orchestration tools –

Kubernetes (cloud neutral)

Google Kubernetes Engine (free tier) GKE

GCP

Why docker?

Operation team does not need to know what prog lang u used, what config it needs ? he doesn’t need to know all this 🡪 this is what docker enables

Same packaging for all types of applications

Graphical user interface, text, application, email

Description automatically generated

Deploy, expose, load balance

In28min/hello-world-rest-api:0.0.1.RELEASE is an image

Kubernetes is like auto monitoring

You tell Kubernetes that you want 3 instances.. if 1 goes down, another 1 will automatically come up

Kubernetes can autoscale 🡪 can choose max

You can deploy new versions without ANY downtime!

Diagram

Description automatically generated

Where do the microservices run ?

A Kubernetes cluster – a group of servers

“we don’t need to worry about creating Kubernetes cluster ,, google takes care of it for us”

Now, we need to create a google cloud account

We want to deploy a simple rest api to Kubernetes

What do we need?

1. Google cloud free tier account
2. Kubernetes cluster
3. Then can deploy our rest api

Diagram

Description automatically generated

Just like our managers at work – they manage the worker nodes

Now we want to create a cluster

Graphical user interface, text, application, email, Teams

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Workloads – manage the containers that you want to deploy into the cluster

Services – give external world access to applications that are deployed to Kubernetes clusters

Storage – persistent data storage for your application

Creating a cluster

1. change name of cluster

Cluster created

K8S – it just means Kubernetes

Kubernetes in Greek means helmsman of a ship -> provides direction, manages things

Google calls Kubernetes GKE – google Kubernetes engine

Graphical user interface, text, application

Description automatically generated

Cluster is ready – it has 3 nodes, 3 cpus etc etc

Text, table

Description automatically generated

Kubernetes has something installed on each of these nodes to manage them

Deploy first spring boot app to Kubernetes cluster

1. Connect to Kubernetes cluster (cmd line – use google cloud shell )

Graphical user interface, text, application, email

Description automatically generated

1. Kubectl 🡪 Kubernetes command to interact with the cluster
2. You just need to connect to the Kubernetes cluster and use kubectl to interact with it
3. We now want to deploy an application to the Kubernetes cluster (about damn time)
4. Kubectl create deployment hello-world-rest-api –image=in28min/hello-world-rest-api:0.0.1.RELEASE 🡪 this is the docker image !

(where is this image coming from)

1. So Kubernetes can somehow communicate with dockerhub
2. This will deploy the application to the Kubernetes cluster
3. But we still need to EXPOSE it to outside world
4. Kubectl expose deployment hello-world-rest-api –type=LoadBalancer –port=8080
5. Exposed

Graphical user interface, text, application, chat or text message

Description automatically generated

1. docker run -p 8080:8080 in28min/hello-world-rest-api:0.0.1.RELEASE
3. kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE
4. kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080
5. kubectl scale deployment hello-world-rest-api --replicas=3
6. kubectl delete pod hello-world-rest-api-58ff5dd898-62l9d
7. kubectl autoscale deployment hello-world-rest-api --max=10 --cpu-percent=70
8. kubectl edit deployment hello-world-rest-api #minReadySeconds: 15
9. kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE
11. gcloud container clusters get-credentials in28minutes-cluster --zone us-central1-a --project solid-course-258105
12. kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE
13. kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080
14. kubectl set image deployment hello-world-rest-api hello-world-rest-api=DUMMY\_IMAGE:TEST
15. kubectl get events --sort-by=.metadata.creationTimestamp
16. kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE
17. kubectl get events --sort-by=.metadata.creationTimestamp
18. kubectl get componentstatuses
19. kubectl get pods --all-namespaces
21. kubectl get events
22. kubectl get pods
23. kubectl get replicaset
24. kubectl get deployment
25. kubectl get service
27. kubectl get pods -o wide
29. kubectl explain pods
30. kubectl get pods -o wide
32. kubectl describe pod hello-world-rest-api-58ff5dd898-9trh2
34. kubectl get replicasets
35. kubectl get replicaset
37. kubectl scale deployment hello-world-rest-api --replicas=3
38. kubectl get pods
39. kubectl get replicaset
40. kubectl get events
41. kubectl get events --sort.by=.metadata.creationTimestamp
43. kubectl get rs
44. kubectl get rs -o wide
45. kubectl set image deployment hello-world-rest-api hello-world-rest-api=DUMMY\_IMAGE:TEST
46. kubectl get rs -o wide
47. kubectl get pods
48. kubectl describe pod hello-world-rest-api-85995ddd5c-msjsm
49. kubectl get events --sort-by=.metadata.creationTimestamp
51. kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE
52. kubectl get events --sort-by=.metadata.creationTimestamp
53. kubectl get pods -o wide
54. kubectl delete pod hello-world-rest-api-67c79fd44f-n6c7l
55. kubectl get pods -o wide
56. kubectl delete pod hello-world-rest-api-67c79fd44f-8bhdt
58. gcloud container clusters get-credentials in28minutes-cluster --zone us-central1-c --project solid-course-258105
59. docker login
60. docker push in28min/mmv2-currency-exchange-service:0.0.11-SNAPSHOT
61. docker push in28min/mmv2-currency-conversion-service:0.0.11-SNAPSHOT
63. kubectl create deployment currency-exchange --image=in28min/mmv2-currency-exchange-service:0.0.11-SNAPSHOT
64. kubectl expose deployment currency-exchange --type=LoadBalancer --port=8000
65. kubectl get svc
66. kubectl get services
67. kubectl get pods
68. kubectl get po
69. kubectl get replicaset
70. kubectl get rs
71. kubectl get all
73. kubectl create deployment currency-conversion --image=in28min/mmv2-currency-conversion-service:0.0.11-SNAPSHOT
74. kubectl expose deployment currency-conversion --type=LoadBalancer --port=8100
76. kubectl get svc --watch
78. kubectl get deployments
80. kubectl get deployment currency-exchange -o yaml >> deployment.yaml
81. kubectl get service currency-exchange -o yaml >> service.yaml
83. kubectl diff -f deployment.yaml
84. kubectl apply -f deployment.yaml
86. kubectl delete all -l app=currency-exchange
87. kubectl delete all -l app=currency-conversion
89. kubectl rollout history deployment currency-conversion
90. kubectl rollout history deployment currency-exchange
91. kubectl rollout undo deployment currency-exchange --to-revision=1
93. kubectl logs currency-exchange-9fc6f979b-2gmn8
94. kubectl logs -f currency-exchange-9fc6f979b-2gmn8
96. kubectl autoscale deployment currency-exchange --min=1 --max=3 --cpu-percent=5
97. kubectl get hpa
99. kubectl top pod
100. kubectl top nodes
101. kubectl get hpa
102. kubectl delete hpa currency-exchange
104. kubectl create configmap currency-conversion --from-literal=CURRENCY\_EXCHANGE\_URI=http://currency-exchange
105. kubectl get configmap
107. kubectl get configmap currency-conversion -o yaml >> configmap.yaml
109. watch -n 0.1 curl http://34.66.241.150:8100/currency-conversion-feign/from/USD/to/INR/quantity/10
111. docker push in28min/mmv2-currency-conversion-service:0.0.12-SNAPSHOT
112. docker push in28min/mmv2-currency-exchange-service:0.0.12-SNAPSHOT

kubectl get events 🡪 a lot of events have happened

1. creating cluster (node)
2. Pod 🡪 pull image , create container

Smallest deployable unit 🡪 each pod contains eg 2 containers

Containers live inside pods

Each pod has its own unique IP address

No. of containers in a pod

1 node 🡪 many pods 🡪 each pod 🡪 many containers

1 deployment 🡪 many replica set 🡪 1 replica set 🡪 many pods

Separate dev from qa 🡪 use namespaces !

1. ReplicaSet 🡪

Ensures that a specific number of pods are running at all times

Even if u kill a pod, another will be created

This is the magic of the replicaset

1. Deployment 🡪

Update deployment version with 0 downtime

Even if u made a mistake with deployment, it will continue to run

It will create a new recplicaset that will create a new pod and monitor

Does staggered deployment until all v2 are up, once all v2 are up, then all v1 can go down

Update 1 pod at a time

QUICK REVIEW TIME

1. Service 🡪 a pod is a throwaway unit that might get deleted

The pod’s ip address will always be changing

Provides a permanent ip address to the consumer no matter which ever pods die

LoadBalancer

Kubernetes uses single responsibility pattern

Kubectl get events to see what is happening in the background behind the scenes

Kubectl explain replicaset 🡺 ensures that a certain number of pods are running at any one time

Master node 🡪 distributed database , (eg. Desired state eg I want 5 instances of application A)

* + API server (how kubectl talks to the cloud )
  + Scheduler (which pods will go into which nodes)
  + Controller (actual state matches desired state )

Worker node

* + Node agent (monitors what happens on node and report to manager)
  + Kube proxy -> exposing services
  + Container runtime (docker) 🡪 to run container inside your pods

Gcloud 🡪 what is this for

* + install on ur local machine
  + So no need to use google cloud shell on the internet
  + Can use command line on your computer
  + Now need to install kubectl on local machine !
  + All this is so that from our local machine we can connect to the google cloud cluster
    1. Download the currency conv service and exchange service into eclipse
    2. What changes done ?

Disabled spring cloud starter config etc

Disabled service discovery etc

Cos Kubernetes can do all this internally

* + 1. Kubernetes can only talk to docker over internet.. so u need to push ur images to the net docker.hub.com
    2. When u launch up a new pod, <service\_name>\_SERVICE\_HOST environment variable is automatically created
    3. You can use deployment.yml to deploy
    4. Eg. Can specify number of replicas in the yml file
    5. Service discovery, load balancing