Section 38 L:

DEPLOYMENTS IN KUBERNETES

[illegible]

Scaling Application in Kubernetes.

Difference b/w Pod & Deployment.

In short, a pod is the core building block for running application in a Kubernetes cluster,
A deployment is a management tool used to control the way pods behave.

what is pod :-

In Kubernetes, a pod is either a single container or a group of related containers that share storage and networking resources.

Pods are the smallest application building blocks within a Kubernetes cluster. A Pod could host an entire application, or it could host part of one.

A developer or administrator creates the pod or pods necessary to run an application, & Kubernetes automatically manage them. Kubernetes decides which nodes -- or -- servers -- within the cluster should host each pod, & it automatically restarts pods if they fail.

Page No.
Date

By default, Kubernetes runs one instance of each pod you create. if desired, however, you can create multiple instances of the same pod using Replicasets which are defined via a deployment.

In pod we need to upgrade manually.

What is a deployment?

1) Deployment is a management tool.

2) It doesn't necessarily refer to the deployment of applications or services. Rather, a deployment is a file that defines a pod's desired behaviour or characteristics.

3) Administrators use deployments to specify what they want to happen with their applications. Then, Kubernetes automatically performs the steps necessary to create the specified state.

For example :- a deployment file can define the number of copies -- or replicas, as they're known in Kubernetes -- of a given pod. or a deployment can upgrade an existing pod to a new application version by updating the 'base container image'

By using Kubernetes to operationalize defined behavior, admins don't have to perform all the steps required to achieve a desired state. They can apply a change or update across the entire cluster with one short policy file, which saves time & increases the scalability of Kubernetes clusters.

For instance, instead of manually upgrading the Container image version in a pod, you can use a deployment to tell Kubernetes to do it automatically with a single file & Command.

[243]. Udemy

Scaling Application in Kubernetes:

1. What is App Scaling.
2. Stateless Application.
3. Stateful Application.
4. Hands Demo.

APP Scaling :-

1) Application Scalability is the potential of an application to grow in time, being able to efficiently handle more & more requests per minute.

Whenever we are getting a more traffic application should have the capability to grow in size & whenever we are getting the less traffic applications would have the capability to shrink in size.

2) in Kubernetes, a User can Scale pods.

In Kubernetes, the application will execute in a Container & Container will execute within a pod.

So in K8s, we have a capability to Scale the pods & we can Scale the pods up to a number of pods.

The pod Scaling Could be the Manual Scaling or it Could be the automated process.

(3) Pods Can be Scale Vertically or horizontally.

Horizontal Scaling :- means adding more number of nodes or going to Spin up more instances of my application.

Vertical Scale :- in Existing Running application/instances we are adding more Resources (like : Ram, CPU...)

(4) Now each application Cannot be Scaled & there are a few Constraints about the application Scaling.

So to understand these Constraints; we need to understand the Stateless & Stateful application.

Stateless Application :-

(1) Application program that does not Save client data generated in One Session for use in the next Session with that client

(2) Stateless applications Can be Scaled Horizontally.

(3) New pods Can be Created for Stateless Applications

(4) Generally 90% front end services are the Stateless application.

STATEFULL APPLICATION :-

1) Statefull is just opposite of Stateless.

2) Stateful application will maintain the State of my application, they will save the State of the client data & whenever they are getting the input, the output will be depend on the State of your application.

(3) Database System is a typical Example of a Stateful application.

(4) Stateful application Cannot be Scaled horizontally because we cannot split or break the file System & Create the new instances for my database.

Statefull application Can be Scaled Verticaly.

Scaling in Kubernetes :-

(1) Replication Controller Can be used to manage the App Scaling.

(2) Replication Controller Ensure that a Specified no of Pod replicas are running at any point of time.

(3) Replication Controller make sure that a pod or a set of pods is always up & available

Replication Controller is used for Manual Pod Creation.

Lab [224]

(1) Go to the Script store directory.

cd myscript.

(2) Vi Replication_Controller.yml

```
apiVersion: v1
Kind: ReplicationController
metadata:
  name: alipne-box-replicationcontroller
Spec:
  replicas: 3
  Selector:
    app: alipne-box
  template:
    metadata:
      name: alpine
      labels:
        app: alipne-box
    Spec:
      Containers:
        - name: alpine-box
          image: alpine
          Command: ["sleep", "3600"]
```

Selector: we are define the Label.

note: Selector & Label defined pod template Should be same.

(3) Kubectl apply -f Replication-Controller.yml.

will see the replication name.

(4) Kubectl get <entire Replication name>
Show in Step 3.

(5) we see the number of: Replication Create.

(5) Kubectl get pods -o wide.

(6) Suppose we want to scale the replica again from 3 to 6.

Kubectl Scale --replica=6 [Replication Controller name]
↑
get from Step 3.

(7) Kubectl get <entire Replication Controller name>

(8) if you want to descale the Replica's

Kubectl Scale --replica=2 [Replication Controller name]

we notice, default Created pods were terminated.

(9) To delete the pod's

Kubectl delete -f <Replication-Contr.yml>

SECTION 38:- [245]

ReplicaSet in Kubernetes.

(1) what is ReplicaSet.
(2) Difference b/w ReplicaSet Vs Replication Controller.
(3) ReplicaSet Vs Bare pods
(4) Hands On Demonstration.

Bare pods :- mean pod's which don't have any Kind of Controller, like the Replication Controller deployment or ReplicaSet.

Pods which created by pod template are called the bare pods.

ReplicaSet :-

(1) ReplicaSet is enhanced Version of Replication Controller.

ReplicaSet work same like Replication Controller. ReplicaSet has some additional features. (advanced version).

(2) Difference b/w ReplicaSet and ReplicaController:

The main difference b/w a ReplicaSet and a Replication Controller right now is the Selector Support.

So whenever you define the Replication Controller template, you have to define the Selector over here & that Selector was a Key Value pair & that Selector must be met with labels of your pod template-Label.

in ReplicaSet we can use the MATCH EXPRESSIONS in Selector where we can provide the Conditional Statement as well.

* Label Selector is used to identify a Set of object in K8s

Definitely whenever we are executing the pods & that pod is basically going to create it by a specific template. and we have defined some kind of label on that pod

The Pod is a K8s object, other K8s object will identify that object by the defined label on that particular object

So label Selector a very important feature in K8s and all the Communities object can be identified by the label Selectors.

Replica Set :-

1. ReplicaSets allow us to use "Set-based" label Selector

it means you can define the Set in Value of the label Selector & as we mentioned that the match Expression can be the Conditional. So we can use in, notin and Exist Operator that match the Kubernetes object labels.

Replica-Set

```
---
Spec:
  Replicas: 3
  Selector:
    app: alpine-box
  Template:
    metadata:
---
```

VS

Replica-Controller

```
Spec:
  Replicas: 3
  Selector:
    matchExpressions:
      - { Key: app, Operator: In, Values [test1, test2 ... ]
      - { Key: tier, Operator: NotIn, Values: [production] }
  Template:
    metadata:
---
```

Note:- Values should not have duplicates.

Bare Pods and ReplicaSet

Bare pods :- means the pods which Created without any template, I mean to Say without any Controller template like "no Controller is executing on above of that pod like the deployment or Replica Set or Replication Controller.

So while Creating the bare pods, we make Sure that Label Should not match with ReplicaSets.

Why? if barepods has Same Label, Then it Consider the Bare pods as ReplicaSet group. & might Conflict occurs.

So make Sure that Bare pod Label & ReplicaSet Label are different.

(2) ReplicaSet is not limited to Owning Pods Specified by its template - it Can acquire Other pods which have matching Labels.

Hands on Demo :- [246]

(1) ReplicaSet in Kubernetes Demo Lab. [246] :-

(1) Go to the directory where Scripts / Manifest are Stored. (cd ~)

(2) Vi replica-set.yml.

```
apiVersion : apps/v1
Kind : ReplicaSet
metadata:
    name: myapp-replicas
    labels:
      app: myapp
      tier: frontend
Spec:
    replicas: 3
    Selector:
      matchExpressions:
        - {key: tier, Operator: In, values: [frontend]}
    template:
      metadata:
        labels:
          app: myapp
          tier: frontend
      Spec:
        Containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
```

Save & Exit.

(3) Kubectl apply -f replica-set.yml.

We can see that.

replicaSet.apps/myapp-replicas Created.

④ To check the State of Replicas

```
Kubectl get replicaset.app/myapp-replicas
```

we see that Replicas are Created.

Desired: mean by No of Replicas mentioned in manifest
Current: mean Replicas Created.
Status: Showing No of Replicas in ready State.

or.

Short cut for Replicas Set = rs
Replica Controller = rc

```
Kubectl get rs/myapp-replicas.
```

⑤ Kubectl describe rs/myapp-replicas

⑥ Kubectl get pods -o wide

⑦ To Scale pods.

Kubectl.Scale --replicas=10 rs/myapp-replicas.

⑧ Kubectl get pods -o wide.

⑨ To de-Scale pods.

Same Command mentioned Step 7. with Reduced Replicas no

we se that latest Created Replicas will be terminated.

(10) Kubectl get pods -o wide.

(11) To delete the Complete Replica Set.

Kubectl delete -f replica-set.yml.

will delete the Replicat Set we created.

DEPLOYMENT [247]

Deployment is One level higher abstraction of Replica Set & Replication Controller.

1. what is Deployment
2. Desired State
3. Deployment use cases
4. Hand on Demos :-

what is Deployment :-

(1) Deployment is One Step higher than ReplicaSet.

Deployment is a top level abstract layer in that hierarchy & it will provide One Step higher than the Replica Set

(2) Deployment is desired State of ReplicaSet.

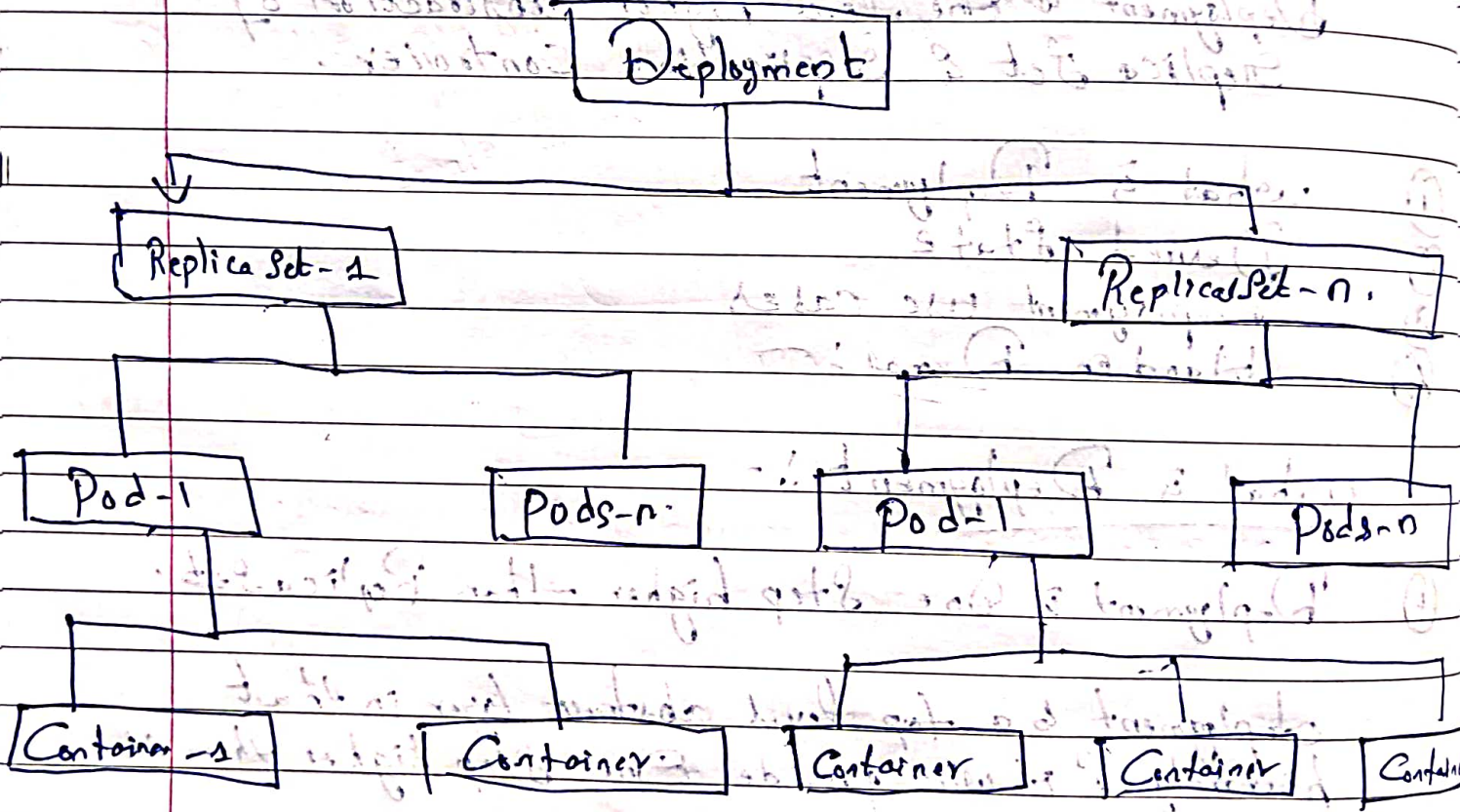
Deployment basicaly Control both deployment can Control the Replica Set & deployment can also Control the Pods.

and that Control is being done by the declarative manifest which defined in the deployment.

(3) By Deployment manifest, you can Control the Replica Set & the pods..

(4) Smallest unit of Deployment, a Pod, runs Containers. Each pod has it Own IP address & shares a PID namespace, network & hostname.

Hierrachy of Deployment



Page No.
Date

USE Case of Deployment :- 2

1. Create Deployment :- Deploy Application Pods
2. Update Deployment :- Push new Version of Application in Controlled Manner.
3. Rolling upgrade :- Upgrade Application in Zero Downtime.

Basically the rolling upgrade is a part of the update deployment. & it will upgrade your application Containers in a rolling manner.

it means if five pods are running as a pod of application deployment will basically upgrade the pods one by one.

in that case, my application won't face any zero downtime because if one instances in the upgrade in the maintenance window. Then other instances are up & they are serving the traffic.

4. rollback features :- Rollback the upgrade in case of unstable upgrade. Revise the Deployment state.

There are some issues & customers are facing some unnecessary glitches. then you can simply roll back your Complete deployment & it will also revise your deployment version.

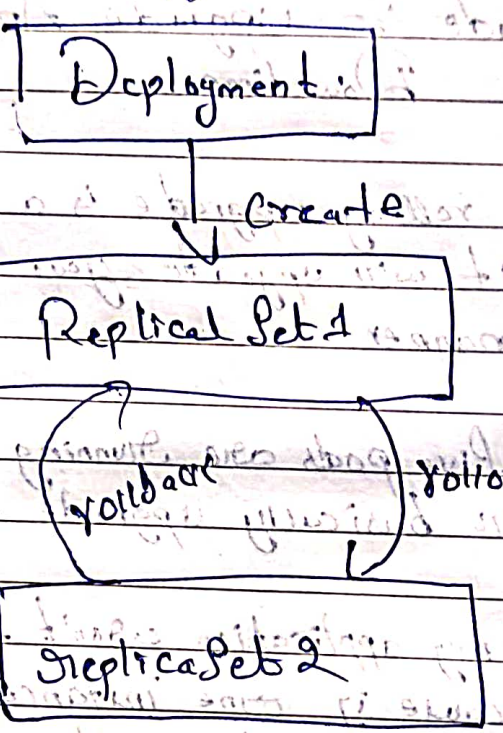
Rollback is also done in the rolling Intert manner or rolling Rollback manner.

Another use Case

① Pause/Resume Deployment:

Can Pause your deployment, make changes in YAML file & then resume your deployment again.

② Deployment Rollout upgrade & Rollback update.



Whenever it is upgrading or rollback the replica, it will Create a new replica every time.

as this Operation is going in the rolling upgrade fashion so application won't face any downtime.

Page No.
Date

Lab 1 [248] Deployment in Kubernetes 1

1. Go to the directory where script are store.

2. Vi deployment.yml

```
apiversion: apps/v1.
Kind: Deployment
metadata:
  name: chef-server
  labels:
    app: chef.
Spec:
  Replicas: 3
  Selector:
    matchLabels:
      app: chef-server.
  template:
    metadata:
      labels:
        app: chef-server.
    Spec:
      Containers:
        - name: chef-server
          image: 'chef/chefdk:4.9'
          ports:
            - ContainerPort: 8080
          Command:
            - /bin/sh
          args:
            - '-c'
            - echo Hello from the chef container, sleep 3600
```

```
- name: Ubuntu
  image: 'ubuntu:18.04'
  Ports:
    - ContainerPort: 8080
  Command
    - /bin/sh
  args:
    - '-c'
    - echo Hello from the Ubuntu Container; sleep 3600.
```

Save & Exit -

(3) Kubectl apply -f deployment.yml.

We see that deployment.apps/Chef-Server. Created.

(4) State of the deployment -

Kubectl get deployment.apps/chef-Server

(5) To check if deployment in roll out

Kubectl rollout Status deployment.apps/.Chef-Server.

(6) Kubectl describle deployment.apps/Chef-Server

(7) Kubectl get rs (RS = Replica Set)

Deployment Create Replica Set

(8) Kubectl get pods --show-labels.

[249] Lab 2:
Deployments in Kubernetes 2.

(1) Rolling update:

(a) deploy new pods with lastes image.
(b) Terminate/down the first old pod & Bring first new pod.
(c) then Terimate/dow the Second old pod & Bring Second new pod.

(1) Kubectl Set image deployment/chef-Server <Container_name> = <latest image version>

Kubectl Set image deployment/chef-Server chef-Sever = chef/ChefDK:4.9.10

(2) Kubectl rollout Status deployment.apps/chef-Server

(3) Kubectl rollout history deployment.apps/chef-Server

we notice the, revision is none, it is not showing the change-cause. (Execute below command)

(4) Kubetcl Set image deployment/chef-Sever chef-Servier = chef/chefdk:4.9.13 --record

(5) Kubectl. rollout history deployment.apps/chef-server.

(6) Kubectl describe pod chef-server.

RollBack if any issues found.

(1) Kubectl rollout undo deployment.apps/chef-server

Roll Back to Specify version.

Kubectl rollout undo deployment.apps/chef-server --to-revision=1.

One more to update but this is not Recommended.

(1) Kubectl ~~run~~ edit deployment.apps/chef-server

it will Open the Yaml file in edit mode

we can edit any thing, Suppose I want to update Ubuntu Container to latest version, we can edit the deployment file.

Save & Exit (vi mode)

(2) it will start rollout immediately after changes.

(3) Kubectl rollout status deployment.apps/chef-server

(4) in edit record will not happen, so it is not recommended.

Pause & resume deployment:

(1) Kubectl rollout pause deployment.apps/chef-server

will pause the rollout, we can change in flunk.

(2) Kubectl set image deployment/chef-server chef-server = chef/chefdk:4.9.16 --record

(3) Kubectl rollout status deployment.apps/chef-server.

We can see the 0 out of 3 new replicas have been updated

Until we resume back it will not update the containers.

(4) Other changes.

Kubectl set image deployment/chef-server ubuntu=ubuntu:21.04 --record.

(5) Let change the resource on container.
Note this is not in manifest.

Container

<Container name>

Kubectl Set resources deployment/chef-server -c=chef-server
--limits=memory=250mi

(6) we did blank changes.

Kubectl rollout resume deployment.apps/chef-server

How to Scale deployment. (Manual Command)

Kubectl Scale deployment.apps/chef-server --replicas=5