A screenshot of a computer

Description automatically generated

**Kubernetes - PODs**

**Step-01: PODs Introduction**

* What is a POD ?
* What is a Multi-Container POD?

**Step-02: PODs Demo**

**Get Worker Nodes Status**

* Verify if kubernetes worker nodes are ready.

# Get Worker Node Status

kubectl get nodes

# Get Worker Node Status with wide option

kubectl get nodes -o wide

**Create a Pod**

* Create a Pod

# Template

kubectl run <desired-pod-name> --image <Container-Image> --generator=run-pod/v1

# Replace Pod Name, Container Image

kubectl run my-first-pod --image stacksimplify/kubenginx:1.0.0 --generator=run-pod/v1

* **Important Note:** Without **--generator=run-pod/v1** it will create a pod with a deployment which is another core kubernetes concept which we will learn in next few minutes.
* **Important Note:**
  + With **Kubernetes 1.18 version**, there is lot clean-up to **kubectl run** command.
  + The below will suffice to create a Pod as a pod without creating deployment. We dont need to add **--generator=run-pod/v1**

kubectl run my-first-pod --image stacksimplify/kubenginx:1.0.0

**List Pods**

* Get the list of pods

# List Pods

kubectl get pods

# Alias name for pods is po

kubectl get po

**List Pods with wide option**

* List pods with wide option which also provide Node information on which Pod is running

kubectl get pods -o wide

**What happened in the backgroup when above command is run?**

1. Kubernetes created a pod
2. Pulled the docker image from docker hub
3. Created the container in the pod
4. Started the container present in the pod

**Describe Pod**

* Describe the POD, primarily required during troubleshooting.
* Events shown will be of a great help during troubleshooting.

# To get list of pod names

kubectl get pods

# Describe the Pod

kubectl describe pod <Pod-Name>

kubectl describe pod my-first-pod

**Access Application**

* Currently we can access this application only inside worker nodes.
* To access it externally, we need to create a **NodePort Service**.
* **Services** is one very very important concept in Kubernetes.

**Delete Pod**

# To get list of pod names

kubectl get pods

# Delete Pod

kubectl delete pod <Pod-Name>

kubectl delete pod my-first-pod

**Step-03: NodePort Service Introduction**

* What are Services in k8s?
* What is a NodePort Service?
* How it works?

**Step-04: Demo - Expose Pod with a Service**

* Expose pod with a service (NodePort Service) to access the application externally (from internet)
* **Ports**
  + **port:** Port on which node port service listens in Kubernetes cluster internally
  + **targetPort:** We define container port here on which our application is running.
  + **NodePort:** Worker Node port on which we can access our application.

# Create a Pod

kubectl run <desired-pod-name> --image <Container-Image> --generator=run-pod/v1

kubectl run my-first-pod --image stacksimplify/kubenginx:1.0.0 --generator=run-pod/v1

# Expose Pod as a Service

kubectl expose pod <Pod-Name> --type=NodePort --port=80 --name=<Service-Name>

kubectl expose pod my-first-pod --type=NodePort --port=80 --name=my-first-service

# Get Service Info

kubectl get service

kubectl get svc

# Get Public IP of Worker Nodes

kubectl get nodes -o wide

* **Access the Application using Public IP**

http://<node1-public-ip>:<Node-Port>

* **Important Note about: target-port**
  + If target-port is not defined, by default and for convenience, the **targetPort** is set to the same value as the **port** field.

# Below command will fail when accessing the application, as service port (81) and container port (80) are different

kubectl expose pod my-first-pod --type=NodePort --port=81 --name=my-first-service2

# Expose Pod as a Service with Container Port (--taret-port)

kubectl expose pod my-first-pod --type=NodePort --port=81 --target-port=80 --name=my-first-service3

# Get Service Info

kubectl get service

kubectl get svc

# Get Public IP of Worker Nodes

kubectl get nodes -o wide

* **Access the Application using Public IP**

http://<node1-public-ip>:<Node-Port>

**Step-05: Interact with a Pod**

**Verify Pod Logs**

# Get Pod Name

kubectl get po

# Dump Pod logs

kubectl logs <pod-name>

kubectl logs my-first-pod

# Stream pod logs with -f option and access application to see logs

kubectl logs <pod-name>

kubectl logs -f my-first-pod

* **Important Notes**
  + Refer below link and search for **Interacting with running Pods** for additional log options
  + Troubleshooting skills are very important. So please go through all logging options available and master them.
  + **Reference:** <https://kubernetes.io/docs/reference/kubectl/cheatsheet/>

**Connect to Container in a POD**

* **Connect to a Container in POD and execute commands**

# Connect to Nginx Container in a POD

kubectl exec -it <pod-name> -- /bin/bash

kubectl exec -it my-first-pod -- /bin/bash

# Execute some commands in Nginx container

ls

cd /usr/share/nginx/html

cat index.html

exit

* **Running individual commands in a Container**

kubectl exec -it <pod-name> env

# Sample Commands

kubectl exec -it my-first-pod env

kubectl exec -it my-first-pod ls

kubectl exec -it my-first-pod cat /usr/share/nginx/html/index.html

**Step-06: Get YAML Output of Pod & Service**

**Get YAML Output**

# Get pod definition YAML output

kubectl get pod my-first-pod -o yaml

# Get service definition YAML output

kubectl get service my-first-service -o yaml

**Step-07: Clean-Up**

# Get all Objects in default namespace

kubectl get all

# Delete Services

kubectl delete svc my-first-service

kubectl delete svc my-first-service2

kubectl delete svc my-first-service3

# Delete Pod

kubectl delete pod my-first-pod

# Get all Objects in default namespace

kubectl get all

**Kubernetes - ReplicaSets**

**Step-01: Introduction to ReplicaSets**

* What are ReplicaSets?
* What is the advantage of using ReplicaSets?

**Step-02: Create ReplicaSet**

**Create ReplicaSet**

* Create ReplicaSet

kubectl create -f replicaset-demo.yml

* **replicaset-demo.yml**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: my-helloworld-rs

labels:

app: my-helloworld

spec:

replicas: 3

selector:

matchLabels:

app: my-helloworld

template:

metadata:

labels:

app: my-helloworld

spec:

containers:

- name: my-helloworld-app

image: stacksimplify/kube-helloworld:1.0.0

**List ReplicaSets**

* Get list of ReplicaSets

kubectl get replicaset

kubectl get rs

**Describe ReplicaSet**

* Describe the newly created ReplicaSet

kubectl describe rs/<replicaset-name>

kubectl describe rs/my-helloworld-rs

[or]

kubectl describe rs my-helloworld-rs

**List of Pods**

* Get list of Pods

#Get list of Pods

kubectl get pods

kubectl describe pod <pod-name>

# Get list of Pods with Pod IP and Node in which it is running

kubectl get pods -o wide

**Verify the Owner of the Pod**

* Verify the owner reference of the pod.
* Verify under **"name"** tag under **"ownerReferences"**. We will find the replicaset name to which this pod belongs to.

kubectl get pods <pod-name> -o yaml

kubectl get pods my-helloworld-rs-c8rrj -o yaml

**Step-03: Expose ReplicaSet as a Service**

* Expose ReplicaSet with a service (NodePort Service) to access the application externally (from internet)

# Expose ReplicaSet as a Service

kubectl expose rs <ReplicaSet-Name> --type=NodePort --port=80 --target-port=8080 --name=<Service-Name-To-Be-Created>

kubectl expose rs my-helloworld-rs --type=NodePort --port=80 --target-port=8080 --name=my-helloworld-rs-service

# Get Service Info

kubectl get service

kubectl get svc

# Get Public IP of Worker Nodes

kubectl get nodes -o wide

* **Access the Application using Public IP**

http://<node1-public-ip>:<Node-Port>/hello

**Step-04: Test Replicaset Reliability or High Availability**

* Test how the high availability or reliability concept is achieved automatically in Kubernetes
* Whenever a POD is accidentally terminated due to some application issue, ReplicaSet should auto-create that Pod to maintain desired number of Replicas configured to achive High Availability.

# To get Pod Name

kubectl get pods

# Delete the Pod

kubectl delete pod <Pod-Name>

# Verify the new pod got created automatically

kubectl get pods (Verify Age and name of new pod)

**Step-05: Test ReplicaSet Scalability feature**

* Test how scalability is going to seamless & quick
* Update the **replicas** field in **replicaset-demo.yml** from 3 to 6.

# Before change

spec:

replicas: 3

# After change

spec:

replicas: 6

* Update the ReplicaSet

# Apply latest changes to ReplicaSet

kubectl replace -f replicaset-demo.yml

# Verify if new pods got created

kubectl get pods -o wide

**Step-06: Delete ReplicaSet & Service**

**Delete ReplicaSet**

# Delete ReplicaSet

kubectl delete rs <ReplicaSet-Name>

# Sample Commands

kubectl delete rs/my-helloworld-rs

[or]

kubectl delete rs my-helloworld-rs

# Verify if ReplicaSet got deleted

kubectl get rs

**Delete Service created for ReplicaSet**

# Delete Service

kubectl delete svc <service-name>

# Sample Commands

kubectl delete svc my-helloworld-rs-service

[or]

kubectl delete svc/my-helloworld-rs-service

# Verify if Service got deleted

kubectl get svc

**Pending Concept in ReplicaSet**

* We didn't discuss about **Labels & Selectors**
* This concept we can understand in detail when we are learning to write Kubernetes YAML manifest.
* So we will understand about this during the **ReplicaSets-YAML** section.

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: my-helloworld-rs

labels:

app: my-helloworld

spec:

replicas: 6

selector:

matchLabels:

app: my-helloworld

template:

metadata:

labels:

app: my-helloworld

spec:

containers:

- name: my-helloworld-app

image: stacksimplify/kube-helloworld:1.0.0

**Kubernetes - Deployment**

**Step-01: Introduction to Deployments**

* What is a Deployment?
* What all we can do using Deployment?
* Create a Deployment
* Scale the Deployment
* Expose the Deployment as a Service

**Step-02: Create Deployment**

* Create Deployment to rollout a ReplicaSet
* Verify Deployment, ReplicaSet & Pods
* **Docker Image Location:** <https://hub.docker.com/repository/docker/stacksimplify/kubenginx>

# Create Deployment

kubectl create deployment <Deplyment-Name> --image=<Container-Image>

kubectl create deployment my-first-deployment --image=stacksimplify/kubenginx:1.0.0

# Verify Deployment

kubectl get deployments

kubectl get deploy

# Describe Deployment

kubectl describe deployment <deployment-name>

kubectl describe deployment my-first-deployment

# Verify ReplicaSet

kubectl get rs

# Verify Pod

kubectl get po

**Step-03: Scaling a Deployment**

* Scale the deployment to increase the number of replicas (pods)

# Scale Up the Deployment

kubectl scale --replicas=20 deployment/<Deployment-Name>

kubectl scale --replicas=20 deployment/my-first-deployment

# Verify Deployment

kubectl get deploy

# Verify ReplicaSet

kubectl get rs

# Verify Pods

kubectl get po

# Scale Down the Deployment

kubectl scale --replicas=10 deployment/my-first-deployment

kubectl get deploy

**Step-04: Expose Deployment as a Service**

* Expose **Deployment** with a service (NodePort Service) to access the application externally (from internet)

# Expose Deployment as a Service

kubectl expose deployment <Deployment-Name> --type=NodePort --port=80 --target-port=80 --name=<Service-Name-To-Be-Created>

kubectl expose deployment my-first-deployment --type=NodePort --port=80 --target-port=80 --name=my-first-deployment-service

# Get Service Info

kubectl get svc

Observation: Make a note of port which starts with 3 (Example: 80:3xxxx/TCP). Capture the port 3xxxx and use it in application URL below.

# Get Public IP of Worker Nodes

kubectl get nodes -o wide

Observation: Make a note of "EXTERNAL-IP" if your Kubernetes cluster is setup on AWS EKS.

* **Access the Application using Public IP**

http://<worker-node-public-ip>:<Node-Port>

**Kubernetes - Update Deployments**

**Step-00: Introduction**

* We can update deployments using two options
  + Set Image
  + Edit Deployment

**Step-01: Updating Application version V1 to V2 using "Set Image" Option**

**Update Deployment**

* **Observation:** Please Check the container name in spec.container.name yaml output and make a note of it and replace in kubectl set image command

# Get Container Name from current deployment

kubectl get deployment my-first-deployment -o yaml

# Update Deployment - SHOULD WORK NOW

kubectl set image deployment/<Deployment-Name> <Container-Name>=<Container-Image> --record=true

kubectl set image deployment/my-first-deployment kubenginx=stacksimplify/kubenginx:2.0.0 --record=true

**Verify Rollout Status (Deployment Status)**

* **Observation:** By default, rollout happens in a rolling update model, so no downtime.

# Verify Rollout Status

kubectl rollout status deployment/my-first-deployment

# Verify Deployment

kubectl get deploy

**Describe Deployment**

* **Observation:**
  + Verify the Events and understand that Kubernetes by default do "Rolling Update" for new application releases.
  + With that said, we will not have downtime for our application.

# Descibe Deployment

kubectl describe deployment my-first-deployment

**Verify ReplicaSet**

* **Observation:** New ReplicaSet will be created for new version

# Verify ReplicaSet

kubectl get rs

**Verify Pods**

* **Observation:** Pod template hash label of new replicaset should be present for PODs letting us know these pods belong to new ReplicaSet.

# List Pods

kubectl get po

**Verify Rollout History of a Deployment**

* **Observation:** We have the rollout history, so we can switch back to older revisions using revision history available to us.

# Check the Rollout History of a Deployment

kubectl rollout history deployment/<Deployment-Name>

kubectl rollout history deployment/my-first-deployment

**Access the Application using Public IP**

* We should see Application Version:V2 whenever we access the application in browser

# Get NodePort

kubectl get svc

Observation: Make a note of port which starts with 3 (Example: 80:3xxxx/TCP). Capture the port 3xxxx and use it in application URL below.

# Get Public IP of Worker Nodes

kubectl get nodes -o wide

Observation: Make a note of "EXTERNAL-IP" if your Kubernetes cluster is setup on AWS EKS.

# Application URL

http://<worker-node-public-ip>:<Node-Port>

**Step-02: Update the Application from V2 to V3 using "Edit Deployment" Option**

**Edit Deployment**

# Edit Deployment

kubectl edit deployment/<Deployment-Name> --record=true

kubectl edit deployment/my-first-deployment --record=true

# Change From 2.0.0

spec:

containers:

- image: stacksimplify/kubenginx:2.0.0

# Change To 3.0.0

spec:

containers:

- image: stacksimplify/kubenginx:3.0.0

**Verify Rollout Status**

* **Observation:** Rollout happens in a rolling update model, so no downtime.

# Verify Rollout Status

kubectl rollout status deployment/my-first-deployment

**Verify Replicasets**

* **Observation:** We should see 3 ReplicaSets now, as we have updated our application to 3rd version 3.0.0

# Verify ReplicaSet and Pods

kubectl get rs

kubectl get po

**Verify Rollout History**

# Check the Rollout History of a Deployment

kubectl rollout history deployment/<Deployment-Name>

kubectl rollout history deployment/my-first-deployment

**Access the Application using Public IP**

* We should see Application Version:V3 whenever we access the application in browser

# Get NodePort

kubectl get svc

Observation: Make a note of port which starts with 3 (Example: 80:3xxxx/TCP). Capture the port 3xxxx and use it in application URL below.

# Get Public IP of Worker Nodes

kubectl get nodes -o wide

Observation: Make a note of "EXTERNAL-IP" if your Kubernetes cluster is setup on AWS EKS.

# Application URL

http://<worker-node-public-ip>:<Node-Port>

**Rollback Deployment**

**Step-00: Introduction**

* We can rollback a deployment in two ways.
  + Previous Version
  + Specific Version

**Step-01: Rollback a Deployment to previous version**

**Check the Rollout History of a Deployment**

# List Deployment Rollout History

kubectl rollout history deployment/<Deployment-Name>

kubectl rollout history deployment/my-first-deployment

**Verify changes in each revision**

* **Observation:** Review the "Annotations" and "Image" tags for clear understanding about changes.

# List Deployment History with revision information

kubectl rollout history deployment/my-first-deployment --revision=1

kubectl rollout history deployment/my-first-deployment --revision=2

kubectl rollout history deployment/my-first-deployment --revision=3

**Rollback to previous version**

* **Observation:** If we rollback, it will go back to revision-2 and its number increases to revision-4

# Undo Deployment

kubectl rollout undo deployment/my-first-deployment

# List Deployment Rollout History

kubectl rollout history deployment/my-first-deployment

**Verify Deployment, Pods, ReplicaSets**

kubectl get deploy

kubectl get rs

kubectl get po

kubectl describe deploy my-first-deployment

**Access the Application using Public IP**

* We should see Application Version:V2 whenever we access the application in browser

# Get NodePort

kubectl get svc

Observation: Make a note of port which starts with 3 (Example: 80:3xxxx/TCP). Capture the port 3xxxx and use it in application URL below.

# Get Public IP of Worker Nodes

kubectl get nodes -o wide

Observation: Make a note of "EXTERNAL-IP" if your Kubernetes cluster is setup on AWS EKS.

# Application URL

http://<worker-node-public-ip>:<Node-Port>

**Step-02: Rollback to specific revision**

**Check the Rollout History of a Deployment**

# List Deployment Rollout History

kubectl rollout history deployment/<Deployment-Name>

kubectl rollout history deployment/my-first-deployment

**Rollback to specific revision**

# Rollback Deployment to Specific Revision

kubectl rollout undo deployment/my-first-deployment --to-revision=3

**List Deployment History**

* **Observation:** If we rollback to revision 3, it will go back to revision-3 and its number increases to revision-5 in rollout history

# List Deployment Rollout History

kubectl rollout history deployment/my-first-deployment

**Access the Application using Public IP**

* We should see Application Version:V3 whenever we access the application in browser

# Get NodePort

kubectl get svc

Observation: Make a note of port which starts with 3 (Example: 80:3xxxx/TCP). Capture the port 3xxxx and use it in application URL below.

# Get Public IP of Worker Nodes

kubectl get nodes -o wide

Observation: Make a note of "EXTERNAL-IP" if your Kubernetes cluster is setup on AWS EKS.

# Application URL

http://<worker-node-public-ip>:<Node-Port>

**Step-03: Rolling Restarts of Application**

* Rolling restarts will kill the existing pods and recreate new pods in a rolling fashion.

# Rolling Restarts

kubectl rollout restart deployment/<Deployment-Name>

kubectl rollout restart deployment/my-first-deployment

# Get list of Pods

kubectl get po

**Pause & Resume Deployments**

**Step-00: Introduction**

* Why do we need Pausing & Resuming Deployments?
  + If we want to make multiple changes to our Deployment, we can pause the deployment make all changes and resume it.
* We are going to update our Application Version from **V3 to V4** as part of learning "Pause and Resume Deployments"

**Step-01: Pausing & Resuming Deployments**

**Check current State of Deployment & Application**

# Check the Rollout History of a Deployment

kubectl rollout history deployment/my-first-deployment

Observation: Make a note of last version number

# Get list of ReplicaSets

kubectl get rs

Observation: Make a note of number of replicaSets present.

# Access the Application

http://<worker-node-ip>:<Node-Port>

Observation: Make a note of application version

**Pause Deployment and Two Changes**

# Pause the Deployment

kubectl rollout pause deployment/<Deployment-Name>

kubectl rollout pause deployment/my-first-deployment

# Update Deployment - Application Version from V3 to V4

kubectl set image deployment/my-first-deployment kubenginx=stacksimplify/kubenginx:4.0.0 --record=true

# Check the Rollout History of a Deployment

kubectl rollout history deployment/my-first-deployment

Observation: No new rollout should start, we should see same number of versions as we check earlier with last version number matches which we have noted earlier.

# Get list of ReplicaSets

kubectl get rs

Observation: No new replicaSet created. We should have same number of replicaSets as earlier when we took note.

# Make one more change: set limits to our container

kubectl set resources deployment/my-first-deployment -c=kubenginx --limits=cpu=20m,memory=30Mi

**Resume Deployment**

# Resume the Deployment

kubectl rollout resume deployment/my-first-deployment

# Check the Rollout History of a Deployment

kubectl rollout history deployment/my-first-deployment

Observation: You should see a new version got created

# Get list of ReplicaSets

kubectl get rs

Observation: You should see new ReplicaSet.

**Access Application**

# Access the Application

http://<node1-public-ip>:<Node-Port>

Observation: You should see Application V4 version

**Step-02: Clean-Up**

# Delete Deployment

kubectl delete deployment my-first-deployment

# Delete Service

kubectl delete svc my-first-deployment-service

# Get all Objects from Kubernetes default namespace

kubectl get all