**Jenkins & Kubernates**

What is Jenkins ?

* Jenkins is a continuous integration tool used to build (compile and test) code and deploy it to the production.
* Jenkins provides hundreds of plugins to support building, deploying, and automating any project.
* It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools like SVN, Git to automate build.

What is continuous integration ?

* Continuous integration is a process in which all development work is integrated as early as possible. The resulting artifacts are automatically created and tested. This process allows to identify errors as early as possible.

|  |  |
| --- | --- |
| **Before continuous integration** | **After continuous integration.** |
| The entire source code was built and then tested. | Every commit made in source code is built and tested. |
| Developers must wait for the test results. | Developers know the test result of every commit made in the source code on the run. |
| No feedback. | Feedback is present. |

What is Pipeline?

Pipeline is a workflow with the group of events or jobs that are chained and integrated with each other in sequence.

* Pipeline is a process of continuous delivery automation using Jenkins’s job(items).
* Each job contains some processing inlet and outlets.
* Build – Deploy – Test – Release
* Continuous delivery Pipeline

Types of Pipelines ?

* Build Plugin Pipeline
* Declarative pipeline
* Scripted pipeline

**Kubernetes**

What is Kubernetes?

* Kubernetes is an open-source container-orchestration engine or container management tool, it automates deploying, scaling, and managing containerized application.
* K**ubernete**s => k8s
* Initially developed by google and now it is maintaining by Cloud Native Computing Foundation.
* Developed with Go Lang.

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**Cluster** : A Kubernetes cluster is a set of nodes that run containerized applications. Kubernetes cluster are comprised of one master node and several worker nodes. The master node controls the state of cluster: for example, which applications are running and their corresponding container images. The master node is the origin for all task assignments. Its co-ordinates processes such as :

* Scheduling and scaling applications.
* Maintaining a clusters state
* Implementing the updates

The worker nodes are the components that run these applications. Worker nodes perform tasking assigned by the master node.

**Pod** : Smallest deployment unit in Kubernetes. Containers lives inside the pods.

Graphical user interface, application

Description automatically generated

**Replica Set** : Replica sets ensure a specific number of pods are running at all the times. If one pod is down, then replica set replace that pod with new pod with different IP address.

Graphical user interface, application

Description automatically generated

**Service** : Role of service is to provide always available external interface to the applications which are running inside the pods. The service basically allows your application to receive traffic through a lifetime permanent IP address. And also, service gets the traffic first and service redirects the requests to pods based on their availability. Suppose if our back-end server pod is destroyed and then a new pod with new IP address will replace, but our front-end server won’t know the IP address of new pod, then it becomes a problem. Service provides a DNS name with which our front-end and back-end pods can communicate.

**Deployments** : Deployments are Kubernetes objects that are used for managing pods. You can scale your application by increasing the number of running pods, or update the running application using Deployment Object.

Below one is a command to create a deployment via terminal, it says create a deployment with name spring-boot-k8s, use image vishalkumar392/student-service:latest

kubectl create deployment spring-boot-k8s --image=vishalkumar392/student-service:latest --port=5000

Graphical user interface, application

Description automatically generated

Secrets and ConfigMap is present outside the pods, so all pods can access the secrets. All sensitive information like passwords can be kept in secrets and plain text can be place in ConfigMap.

**ETCD** : Kubernetes uses etcd as a key-value database store. It stores the configuration of the Kubernetes cluster in etcd. It stores all the secret and ConfigMap data inside etcd database.

Max limit is 1 mb to store secrets.

Graphical user interface, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated

|  |  |
| --- | --- |
| **Kubernetes commands** | **Usage** |
| minikube start --driver=docker | Start the minikube and create a cluster, master node using docker driver. |
| minikube status | Shows the status of minikube |
| kubectl cluster-info | Gives info about the cluster, and status of master node components. |
| kubectl get node | Shows the info about all nodes. |
| minikube docker-env | After hitting this command, you will get a command i.e.  **eval $(minikube -p minikube docker-env)** which is used to point your shell to minikube’s docker-daemon. |
| kubectl create deployment spring-boot-k8s --image=vishalkumar392:1.0.0 --port=8000 | This command is used to create a deployment with name spring-boot-k8s using image. It will create a pod. |
| kubectl get deployment | It shows all deployments. |
| kubectl describe deployment spring-boot-k8s | It describes the deployment. |
| kubectl delete deployment spring-boot-k8s | It deletes the deployment. |
| kubectl get pods | It shows the pods. |
| kubectl logs (pod name) | It shows the logs for a pod. |
| kubectl expose deployment spring-boot-k8s --type=NodePort --port=8000 | It exposes a deployment by creating a service using service type NodePort. |
| kubectl get service | It shows all the services. |
| minikube service <service name> - -url | It exposes the service DNA url, with which we can connect to containers in pods. |
| kubectl port-forward service/spring-boot-k8s 7080:8000 | It is used to forward the host port to another port number. |
| minikube dashboard | It opens minikube dashboard. |
| kubectl apply -f deployment.yaml | It creates a deployment from yaml file. |
| kubectl apply -f service.yaml | It creates a service from yaml file. |
| kubectl get nodes -o wide | It shows the internal IP address of node. |

**ConfigMap and secrets** :

1. Create a file with name config-map.yaml and place below content in it and save it.

**apiVersion**: v1

**kind**: ConfigMap

**metadata**:

**name**: config-map

**data**:

**config-key**: Hello vishal

**host.docker.internal**: host.docker.internal

1. Create a config map using command **kubectl create -f config-map.yaml**
2. To see the details for config map use command **kubectl get configmap,** **kubectl describe configmap**
3. Then we have to apply the deployment using **kubectl apply -f deployment.yaml** . deployment file has to inside the current directory or ls . Below is the deployment file .

Graphical user interface, text, application

Description automatically generated

1. Create a service using service.yaml file with command **kubectl apply -f service.yaml** .

Graphical user interface, text, application

Description automatically generated

1. Use this command to see url to hit the containers inside the pods. **kubectl port-forward service/springboot-k8s-svc 5000:5000**