**Kubetnetes {K8S}**

**Kubernetes is an open-source** [**container orchestration**](https://www.vmware.com/topics/glossary/content/container-orchestration) **platform that enables the operation of an elastic web server framework for cloud applications. Kubernetes can support data centre outsourcing to** [**public cloud**](https://www.vmware.com/topics/glossary/content/public-cloud) **service providers or can be used for web hosting at scale. Website and mobile applications with complex custom code can deploy using Kubernetes on commodity hardware to lower the costs on web server provisioning with public cloud hosts and to optimise software development processes.**

**IMP links:**

**EKS:** [**https://www.youtube.com/watch?v=v5vyC\_C-juU**](https://www.youtube.com/watch?v=v5vyC_C-juU)

[**https://www.youtube.com/watch?v=QThadS3Soig**](https://www.youtube.com/watch?v=QThadS3Soig)

[**https://www.youtube.com/watch?v=aZd0UolVwD4**](https://www.youtube.com/watch?v=aZd0UolVwD4)

**YAML:** [**https://www.youtube.com/watch?v=cdLNKUoMc6c**](https://www.youtube.com/watch?v=cdLNKUoMc6c)

**Pod**: <https://kubernetes.io/docs/concepts/workloads/pods/>

**Readiness & Liveness :** <https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/>

**Kubernetes Architecture:**

<https://www.appvia.io/blog/components-of-kubernetes-architecture>

**RC vs RS vs Deployment:**

[**https://www.mirantis.com/blog/kubernetes-replication-controller-replica-set-and-deployments-understanding-replication-options/**](https://www.mirantis.com/blog/kubernetes-replication-controller-replica-set-and-deployments-understanding-replication-options/)

[**https://www.edureka.co/community/43891/difference-between-replica-set-and-replication-controller**](https://www.edureka.co/community/43891/difference-between-replica-set-and-replication-controller)

[**https://stackoverflow.com/questions/36220388/what-is-the-difference-between-replicaset-and-replicationcontroller**](https://stackoverflow.com/questions/36220388/what-is-the-difference-between-replicaset-and-replicationcontroller)

1. **Managed by customer (self managed)**: you can install anywhere- VMware, Datacenter, ec2 etc

Customer will be responsible for everything

Ex: adding new nodes to k8s cluster, Managing Kubernetes Master, Backup, Archievle, Network, Storage, LoadBalancing, domain services, security etc

You will be responsible only code development, code deployment, writing kubernetes manifest file, docker images etc

To install kubernetes:

1. Kubeadm {bootstrap method-> to create a k8s cluster, generate a key on one of the vm (master), paste the key to another node(worker node) }

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>

1. KOPS {Kubernetes Operations}

<https://kubernetes.io/docs/setup/production-environment/tools/kops/>

1. KubeSpray:

<https://kubernetes.io/docs/setup/production-environment/tools/kubespray/>

1. **Managed Kubernetes {Managed by cloud provider}**

Ex: AWS EKS, Azure AKS, Google GKE, RedHat OpenShift etc

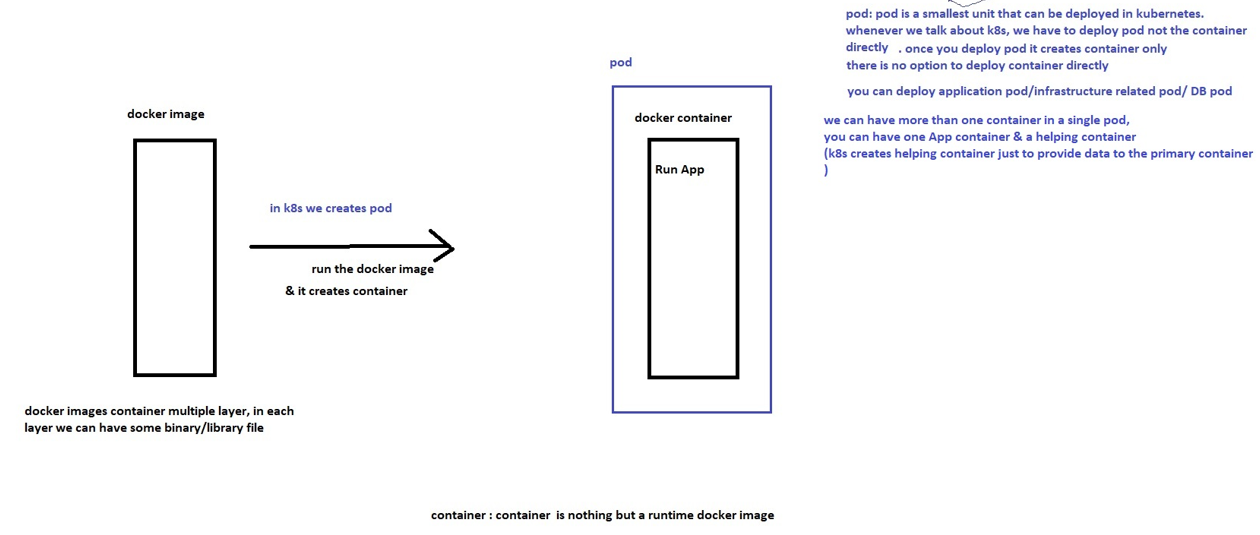
Ex: adding/removing new nodes to k8s cluster, Managing Kubernetes Master (Control panel), Backup, Archievle, Network, Storage, LoadBalancing, domain services will be managed by cloud provider

You will be responsible only code development, code deployment, writing kubernetes manifest file, docker images etc

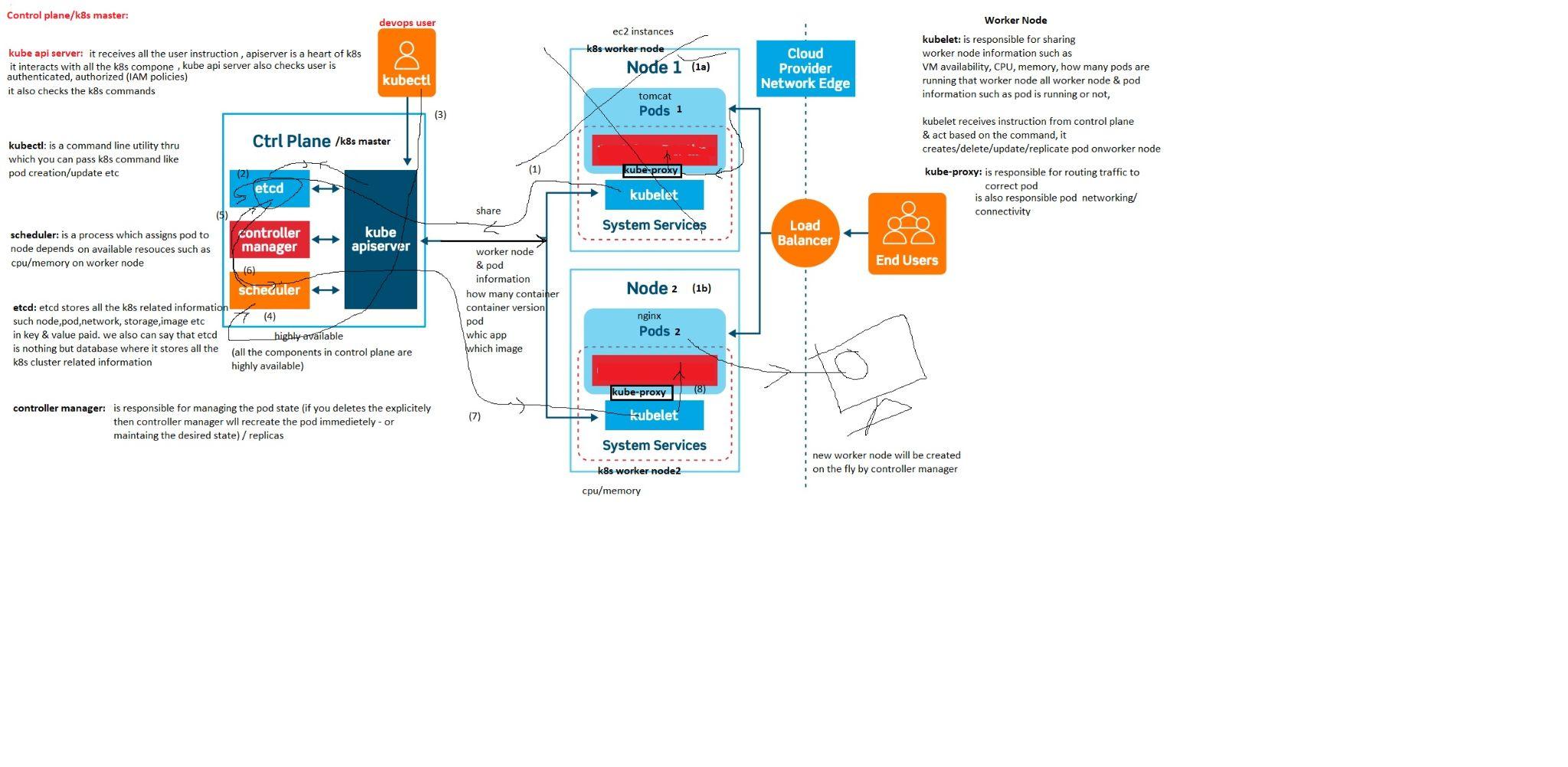
**Pod**: pod is the smallest unit that can be deployed in kubernetes, it is combination of containers (one or more) only {you don’t to deploy any container directly, simply create pod, pod will create container automatically in the backend}

Note: never run web and database containers in the same pod.

We can only helping container {to process script/data/to pass a variable}, helping containers are temporary container, k8s creates helping container during execution only



=================**Kubernetes Architecture:**



**Control Plane components:**

**Kubectl:** It is a command line utility, you can pass instructions to Kubernetes from this command line utility.

**API Server:** APi server receives instructions from kubectl, it checks for user authentication, user authorization, manifest command instructions. It acts as a bridge between all the Kubernetes components.

**The Scheduler:** Scheduler is responsible for the scheduling of containers across the nodes in the cluster; it takes various constraints into account, such as resource limitations or guarantees, and affinity and anti-affinity specifications.

**Controller Manager:** Controller manager is a daemon that runs the core control loops, watches the state of the cluster, and makes changes to drive status toward the desired state. The Cloud Controller Manager integrates into each public cloud for optimal support of availability zones, VM instances, storage services, and network services for DNS, routing and load balancing.

**Etcd**: **etcd** is where **Kubernetes** stores all of the information about a cluster's state such as worker node OS version, networking, storage, pod details, available cpu and memory on worker node etc.

**Worker Node Components:**

**Kubelet**: Kubelet is responsible for following activity

1. It passes all the resources info (CPU, memory, number of running pod, network etc) to control panel
2. Once it receives the instruction from the control panel then creates a pod on the worker node, also responsible for managing the pod states.

**Kube Proxy**: It is responsible for communication between pods, it also connects user requests to correct pods

**============There are 2 method to deploy Kuberentes resources:-**

1. **Imperative method (command line method)**
2. **Declarative method (YAMl/Manifest file)**

**Lets focus on imperative method (this is only good for learning not for production, we prefer declarative approach for prod release )**

**Kubernetes is an open-source container orchestration system/tool for automating software deployment, scaling, and management**

**k8s is a container orchestration tool/management tool**

**Q. What is Pod in kubernetes?**

Ans: pod is the smallest unit that can be deployed in kubernetes, Pod contains containers only, In a pod you may have one or more containers. pod contains primarily or App container and helping container (just to provide some data to primary container or run some script)

**=========deploy your first pod**

**kubectl run --image=<docker image> <pod name>**

**ex: kubectl run --image=nginx mynginx**

kubectl: command line utility

run: create pod

**kubectl get pod** -->list running pod

**kubectl describe pod <pod name>** -->inspect your pod (similar to docker inspect)

**================How to access from browser**

we can access deployed application using Service

Service: we can expose deployed application to outside the kubernetes env (internet) using Service concept,

we need DNS name (Elastic Load Balancer)/Public IP (worker Node Public IP) /Custom Domain (www.ethans.co.in/devops)

**there are 3 types of Services:**

**a. ClusterIP:** It is responsible for internal communication only, we can access from outside

**b. NodePort:** There is Public port range 30000-32768 (total available Public port 2768)

**how to access URL:**

**http://<public IP of worker Node>:<Node Port>**

**ex: http://3.121.86.50:30001**

**what are the problems with NodePort:**

- every time you have remember the NodePort number

- human error, same NodePort port can be assigned to different Applications (conflict)

- though it has limitation, it means we can expose max 2768 Applications only using NodePort

- Ddos attack

**syntax: kubectl expose pod <pod name> --port=<default App port> --target-port=<pod port> --type=<Service Type>**

**--target-port= Default App Port**

**-Service Type= NodePort**

**ex: kubectl expose pod mynginx --port=80 --target-port=80 --type=NodePort**

**kubectl get service**

**or**

**kubectl get svc**

**[cloudshell-user@ip-10-2-90-12 ~]$ kubectl get service**

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 112m

mynginx NodePort 10.100.208.38 <none> 80:30166/TCP 51s

In this case K8s has exposed mynginx pod to NodePort 30166 (k8s will allocate public port from this range: 30000-32767)

how to access URL:

http://<public IP of worker Node>:<Node Port>

ex: http://3.121.86.50:30166

if you are not able to reach to the Application then please check your ec2 (worker Node)security group

sgr-0e36db7a733bbbf49 IPv4 Custom TCP TCP **30000 - 32767** **0.0.0.0/0** –

sgr-05c587c24325ad7ff IPv4 HTTP TCP 80 0.0.0.0/0 –

sgr-06485c820d9d98b36 IPv4 HTTPS TCP 443 0.0.0.0/0 –

sgr-068d7f93d90caff9d IPv4 SSH TCP 22 0.0.0.0/0 –

sgr-01a22a1c8db077d2a IPv4 All traffic All All

**c. LoadBalancer:** We can create AWS ELB or AWS NLB , it provides us public DNS names. (ELB is default)

We can use AWS managed ELBs to expose Application from external world

**Create Jenkins Pod:**

**kubectl run --image=jenkins/jenkins:lts myjenkins1**

**syntax: kubectl expose pod <pod name> --port=<default App port> --target-port=<pod port> --type=<Service Type>**

**ex: kubectl expose pod myjenkins1 --port=8080 --target-port=8080 --type=LoadBalancer**

[cloudshell-user@ip-10-2-90-12 ~]$ kubectl get service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

myjenkins1 LoadBalancer 10.100.137.170 **a4680fc6977194f2cabdaf223a020d68-1138947925.eu-central-1.elb.amazonaws.com** 8080:30447/TCP 15s

**It will create an ELB, copy ELB DNS url and paste in the browser**

http://a4680fc6977194f2cabdaf223a020d68-1138947925.eu-central-1.elb.amazonaws.com:8080

**How to get inside the pod? how to pod's terminal??**

**syntax: kubectl exec -it <pod name> /bin/bash**

**ex: kubectl exec -it myjenkins1 /bin/bash**

cloudshell-user@ip-10-2-90-12 ~]$ kubectl exec -it myjenkins1 /bin/bash

jenkins@myjenkins1:/$

jenkins@myjenkins1:/$ cat /var/jenkins\_home/secrets/initialAdminPassword

28155be104d94c66983eec787cee27ee

paste password and install suggested plugin. that's it

**How to delete pod?**

**kubectl delete pod <pod name>**

**How to delete service?**

**kubectl delete service <service Name>**

**Delete worker Node -> Delete EKS cluster (All are chargeable services)**

[**https://aws.amazon.com/eks/pricing/**](https://aws.amazon.com/eks/pricing/)

**=======**

**Kubernetes is an open-source container orchestration system/tool for automating software deployment, scaling, and management**

**k8s is a container orchestration tool/management tool**

**Q. what is Pod in kubernetes?**

**=========deploy your first pod**

**kubectl run --image=<docker image> <pod name>**

**ex: kubectl run --image=nginx mynginx**

**kubectl: command line utility**

**run: create pod**

**kubectl get pod -->list running pod**

**kubectl describe pod <pod name> -->inspect your pod (similar docker inpsect)**

**================How to access from browser**

**we can access deployed application using Service**

**Service: we can expose deployed application to outside the kubernetes env (internet) using Service concept,**

**we need DNS name (Elastic Load Balancer)/Public IP (worker Node Public IP) /Custom Domain (www.ethans.co.in/devops)**

**there are 3 types of Services:**

**a. ClusterIP: It is responsible for internal communication only, we can access from outside**

**b. NodePort: There is Public port range 30000-32768 (total available Public port 2768) ->**

**how to access URL:**

**http://<public IP of worker Node>:<Node Port>**

**ex: http://3.121.86.50:30001**

**wht is the problem: every time you have remember the NodePort number**

**: human error, same NodePort port can be assigned to different Applications (conflict)**

**: though it has limitation, it means we can expose max 2768 Applications only using NodePort**

**syntax: kubectl expose pod <pod name> --port=<default App port> --target-port=<pod port> --type=<Service Type>**

**--target-port= Default App Port**

**-Service Type= NodePort**

**ex: kubectl expose pod mynginx --port=80 --target-port=80 --type=NodePort**

**kubectl get service**

**or**

**kubectl get svc**

**[cloudshell-user@ip-10-2-90-12 ~]$ kubectl get service**

**NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE**

**kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 112m**

**mynginx NodePort 10.100.208.38 <none> 80:30166/TCP 51s**

**In this case K8s has exposed mynginx pod to NodePort 30166 (k8s will allocate public port from this range: 30000-32767)**

**how to access URL:**

**http://<public IP of worker Node>:<Node Port>**

**ex: http://3.121.86.50:30166**

**if you are not able to reach to the Application then please check your ec2 (worker Node)security group**

**sgr-0e36db7a733bbbf49 IPv4 Custom TCP TCP 30000 - 32767 0.0.0.0/0 –**

**sgr-05c587c24325ad7ff IPv4 HTTP TCP 80 0.0.0.0/0 –**

**sgr-06485c820d9d98b36 IPv4 HTTPS TCP 443 0.0.0.0/0 –**

**sgr-068d7f93d90caff9d IPv4 SSH TCP 22 0.0.0.0/0 –**

**sgr-01a22a1c8db077d2a IPv4 All traffic All All**

**c. LoadBalancer: We can create AWS ELB or AWS NLB , it provides us public DNS names. (ELB is default)**

**We can use AWS managed ELBs to expose Application from external world**

**Create Jenkins Pod:**

**kubectl run --image=jenkins/jenkins:lts myjenkins1**

**syntax: kubectl expose pod <pod name> --port=<default App port> --target-port=<pod port> --type=<Service Type>**

**ex: kubectl expose pod myjenkins1 --port=8080 --target-port=8080 --type=LoadBalancer**

**[cloudshell-user@ip-10-2-90-12 ~]$ kubectl get service**

**NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE**

**kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 134m**

**myjenkins1 LoadBalancer 10.100.137.170 a4680fc6977194f2cabdaf223a020d68-1138947925.eu-central-1.elb.amazonaws.com 8080:30447/TCP 15s**

**mynginx NodePort 10.100.208.38 <none> 80:30166/TCP 22m**

**It will create ELB, copy ELB DNS url and paste in the browser**

**http://a4680fc6977194f2cabdaf223a020d68-1138947925.eu-central-1.elb.amazonaws.com:8080**

**How to get inside the pod? how to pod's terminal??**

**syntax: kubectl exec -it <pod name> /bin/bash**

**ex: kubectl exec -it myjenkins1 /bin/bash**

**cloudshell-user@ip-10-2-90-12 ~]$ kubectl exec -it myjenkins1 /bin/bash**

**jenkins@myjenkins1:/$**

**jenkins@myjenkins1:/$ cat /var/jenkins\_home/secrets/initialAdminPassword**

**28155be104d94c66983eec787cee27ee**

**paste password and install suggested plugin. that's it**

**How to delete pods?**

**kubectl delete pod <pod name>**

**How to delete service?**

**kubectl delete service <service Name>**

**Delete worker Node -> Delete EKS cluster (All are chargeable services)**

**4-2-2-23**

**=========There are 2 method to deploy Kuberentes resources:-**

**(resources: Pod/Service/Deployment/ReplicaSets/ReplicationController/Job)**

**a) Imperative method (command line method):** you can create k8s resources using commands

kubectl run --image=<docker image> <pod name>

\*\*\* Imperative method is not good for production deployment, due to limitation

**b) Declarative method (YAMl/Manifest file) ->** yet another markup language

**vi my-vistara-nginx-pod.yaml** or my-vistara-pod.yml -> create a file & upload to github along with source code

**apiVersion: v1**

**kind: Pod**

**metadata:**

**name: mynginx**

**labels:**

**env: prod**

**app: webui**

**spec:**

**containers:**

**- name: mynginx**

**image: nginx:1.14.2**

**ports:**

**- containerPort: 80**

**----vi new-apache-pod.yaml**

**apiVersion: v1**

**kind: Pod**

**metadata:**

**name: myapache1**

**labels:**

**env: prod**

**spec: #specification, you can provide kind related resources**

**containers:**

**- name: myache1 #container name**

**image: httpd #docker image**

**ports: #default application port**

**- containerPort: 80**

**How to create resources from Manifest File**

kubectl create -f <file name> ----> create new resource

kubectl apply -f <file name> -----> create new resource & apply changes to existing resource

you can validate yaml by installing yaml extension in VS code or <https://www.yamllint.com/>

you can **create a service & map to a pod using Labels/Tags**

**vi mynginx-service.yaml** (ClusterIP, NodePort, LoadBalancer)

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: my-nginx-service**

**spec:**

**selector: #pod Label and Service Selector should be same**

**env: prod**

**app: webui**

**ports:**

**- protocol: TCP**

**port: 80**

**targetPort: 80**

kubectl apply -f mynginx-service.yaml

kubectl get service

kubectl describe service <service name>

---->check the endpoint (it should point to pod) and Selector

kubectl get pod -o wide ----> get pod with details info where -o=option , wide=wider output

**### if type is not defined then k8s create service as ClusterIP**

In the above example there no Service type defined so it should service as Cluster IP, and you won’t be able to access from browser

**So let’s add type as NodePort and update existing service**

**vi mynginx-service.yaml**  (ClusterIP, NodePort, LoadBalancer)

apiVersion: v1 #apiVersion has fixed values

kind: Service #what type of resource you want to provision "Service"

metadata: #data about the data,

name: my-nginx-service

spec:

selector: #pod Label and Service Selector should be same

env: prod

app: webui

ports:

- protocol: TCP

port: 80

targetPort: 80

**type: NodePort** ### create service as NodePort

Check k8s link for reference: <https://kubernetes.io/docs/concepts/services-networking/service/>

kubectl apply -f mynginx-service.yaml

kubectl get service

kubectl describe service <service name> ---->check the endpoint(it should point to pod)

kubectl get pod -o wide --> get pod with details info where -o=option , wide=wider output

Now you can access from the browser

http://localhost:<NodePort value>

**===========apiVersion:**

if kind is Service -> v1

if kind is Pod -> v1

if kind is Deployment -> apps/v1

if kind is ReplicaSet ->apps/v1

if kind is ReplicationController -> v1

Reference link**:** [**https://kubernetes.io/docs/reference/using-api/**](https://kubernetes.io/docs/reference/using-api/)

How do we release products in the market?

Early Access program (EAP) or alpha release -> for limited user : just to test the product

Beta release or preview release -> more stable as compared to alpha release -> wider range

final production deployment -> for all

**----------How to write yaml file**

**key: value**

ex:

trainer: prakash

course: devops

batch: 567

kind: Pod

**kind is a key and pod is a value**

**Rule1: add a space after :**

**Rule2: add a space after -**

**Array/List: list of element or where key is common, Array/List is ordered collection, it maintains the sequence**

**this is a valid example**

**key: value1**

**key: value2**

**key: value3**

**what is a problem with this example: key is common for all the values**

**key:**

**- value1**

**- value2**

**- value3**

**course: devops**

**course: aws**

**course: azure**

**course:**

**- devops**

**- aws**

**- azure**

**Dictionary: you can define properties of value**

**fruit:**

**- mango: (you can define the properties of mango)**

**weight: 500g**

**colour: yellow**

**origin: india**

**season: summer**

**export: usa**

**- banana:**

**colour: yellow**

**origin: kerla**

**export: uae**

**weight: 400g**

You can validate your yaml online**:** [**https://www.yamllint.com/**](https://www.yamllint.com/)

**============5-2-2023**

**write a yaml file for pune based institutes Ethans, abc, and prq. add courses and course brief description (trainer name, duration, batchtime)**

**pune-institutes:**

**- ethans: #2s**

**courses: #6**

**- devops: #8**

**trainer-info:**

**name: prakash #10**

**address: pune**

**company: abc.org**

**calling-no: 12345**

**batchtime: morning**

**mode: online**

**- aws:**

**trainer-info: jatin**

**batchtime: noon**

**mode: online**

**- azure:**

**trainer-info: jatin**

**mode: online**

**batchtime: noon**

**- abc:**

**courses:**

**- devops**

**- aws**

**- azure**

**- pqr:**

**courses:**

**- devops**

**- aws**

**- azure**

**======================**

**array in above example:**

**- ethans**

**- abc**

**- pqr**

**- devops**

**- aws**

**- azure**

**Dictionary in above example:**

**trainer-info:**

**batchtime:**

**mode:**

**name: prakash #10**

**address: pune**

**company: abc.org**

**calling-no: 12345**

**Dictionaries in Dictionary**

**- devops: #8**

**trainer-info:**

**name: prakash #10**

**address: pune**

**company: abc.org**

**calling-no: 12345**

**batchtime: morning**

**mode: online**

**-------------write a yaml file for colleges in pune. describe courses, fee , number of students, duration, faculty, faculty info such name, address, phone no.**

**========Service manifest file (can we have services and pods in the same yaml file?)**

ans: --- (using three hyphen we can separate the manifest files)

**vi mywebapp-pod-service.yaml**

apiVersion: v1

kind: Pod

metadata:

name: mywebapp

labels:

env: prod

app: mywebapp

spec:

containers:

- name: mywebapp

image: pkw0301/mavenwebapp:01

ports:

- containerPort: 8080 #application port

--- #we can separate using three hyphen

apiVersion: v1

kind: Service

metadata:

name: mywebapp-service

spec:

selector: #should match with Pod's Label

env: prod

app: mywebapp

ports:

- name: http

protocol: TCP

port: 8080

targetPort: 8080

type: NodePort

**--------------------------------Problems with Pod**

a: missing rollback (go back to the previous version) and rollout (release new version)feature

b. it doesn't support replica features (it can create max 1 pod only, it doesn't support the scaling feature. this is not good for app where customer count is unpredictable)

it doesn't provide desired number

**Kubernetes proposed other deployment strategies:**

1. **ReplicationController:** it is also one of the deployment units. ultimately it creates pod (we are not deploying pod directly)

a. it has replica concept/feature

b. it maintains the desired number (if you try to delete/something goes wrong with pod, RC always maintains the desired number of Replicas)

if one pod goes down, it will create a new pod automatically.

c. it has rollback & rollout features.

**limitation of RC:**

1. it has equality based selector (less feature as compared to RS & Deployment)

**example:** [**https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/**](https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/)

**vi my-rc-yaml**

apiVersion: v1

kind: ReplicationController

metadata:

name: nginx

spec:

replicas: 3 #desired number of pod

selector:

app: nginx #equality based selector

template:

metadata:

name: nginx

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

kubectl get rc

PS C:\Users\zy5ddw\Desktop\k8s-example> kubectl get rc

NAME DESIRED CURRENT READY AGE

nginx 3  **3 3 69s**

**Where:**

**DESIRED: from yaml file**

**current: how many pod are currently present in k8s cluster**

**READY: how many pod are actually running**

kubectl describe rc <name of the rc>

kubectl get pod

kubectl delete pod <rc pod>

(automatically it will create new container from template)

**2. ReplicaSet (RS):** [**https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/**](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/)

It is also one of the deployment units. ultimately it creates pod (we are not deploying pod directly)

a. it has replica concept/feature

b. it maintains the desired number (if you try to delete/something goes wrong with pod, RC always maintains the desired number of Replicas)

if one pod goes down, it will create a new pod automatically.

c. it has set based selector (it supports some extra features as compared to RC)

**limitation of RS:**

a. it is difficult to rollout in RS.

**3. Deployment (99.9 % we prefer only Deployment type of manifest file only)**

**it has all the feature of pod, RS and RC**

It is also one of the deployment units (highest unit). ultimately it creates pod (we are not deploying pod directly)

a. it has replica concept/feature

b. it maintains the desired number (if you try to delete/something goes wrong with pod, RC always maintains the desired number of Replicas)

if one pod goes down, it will create a new pod automatically.

c. it has set based selector (it supports some extra features as compared to RC)

d. it supports Rollout & RollBack both

[**https://kubernetes.io/docs/concepts/workloads/controllers/deployment/**](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)

**vi deployment-my.yaml**

apiVersion: **apps/v1**

kind: **Deployment**

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

**matchLabels**:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

kubectl apply -f deployment-my.yaml

kubectl get deployment

kubectl describe deployment

kubectl get pod

**you can keep deployment and service in same yaml file**

**vi mydeployment-service.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

author: prakash

template:

metadata:

labels:  **#labels should match with matchLabels**

app: nginx

author: prakash

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

**---**

apiVersion: v1

kind: Service

metadata:

name: mynginxapp-service

spec:

selector:  **#should match with Deployment's template label**

app: nginx

author: prakash

ports:

- name: http

protocol: TCP

port: 80

targetPort: 80

type: NodePort

[**https://www.geeksforgeeks.org/kuberneters-difference-between-replicaset-and-replication-controller/**](https://www.geeksforgeeks.org/kuberneters-difference-between-replicaset-and-replication-controller/)

**—--11-2-2023**

**Rolling update/Rollout new features & Rollback in case of disaster/or issues in your deployment**

when to consider - whenever you have new feature(dev team has added new functionality ->

build the code & generate new artifact-> just copy new artifact in Docker image-> as devops person you need build new Docker image)

using rollout command we can update existing Deployment with newer Docker image

**vi Dockerfile**

FROM nginx:1.14.2

LABEL author="prakash"

LABEL feature="feature2"

COPY index.html /usr/share/nginx/html/

**vi index.html**

hey, this is feature1

**then create new docker image:**

docker build -t pkw0301/nginx:12 .

**once your new image is ready then update existing deployment with newer image:**

kubectl set image deployment/nginx-deployment nginx=pkw0301/nginx:12

**where nginx : is the container name , pkw0301/nginx:12 is newer image name**

kubectl describe deployment nginx-deployment

**see the rollout status, run:**

kubectl rollout status deployment/nginx-deployment

**to Rollback to any previous version:**

kubectl set image deployment/nginx-deployment nginx=nginx:1.14.2 **----> this is an option to go back to the previous version**

**or**

kubectl rollout undo deployment/nginx-deployment  **---> this is also an option to go back to the previous version**

**First, check the revisions of this Deployment:**

kubectl rollout history deployment/nginx-deployment

**======================namespace:**

**namespace is a virtual cluster in physical kubernetes cluster: or Logical division of physical kubernetes cluster:**

**you can create a namespace for Teams, or Application for logical isolation.**

[**https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/**](https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/)

[**https://kubernetes.io/docs/tasks/administer-cluster/namespaces-walkthrough**](https://kubernetes.io/docs/tasks/administer-cluster/namespaces-walkthrough)

**kubectl get ns**

**or**

**kubectl get namespaces**

**kube-system:** this is reverse for k8s, bydefault k8s creates all the k8s related resources here ex: etcd, scheduler, controller manager etcd

**kubectl get pod -n kube-syetm**

**default:** this is reverse for user, by default k8s provision resources in the default namespaces only

kubectl get pod

or

kubectl get pod -n default

**kube-node-lease:** this is reserve for workernode. k8s maintains the workernode history, (this is not for us)

kubectl get pod -n kube-node-lease

**kube-public:** this is for read only purpose, if you want to share some data/basic info in read only then provision pods here. (rarely we use kube-public pod)

**default:** here you can see the resources/pods created by the user, bydefault k8s creates resources in default namespaces only.

or you have not defined the "namespace" keyword in manifest then k8s provisions resources in "default" namespace.

"default" namespace is reserve for user:

kubectl get pod

kubectl get deployment

kubectl get svc

simply describe one of resource:

kubectl describe service <service name>

kubectl get pod -n kube-system

kubectl create ns dev

kubectl create ns teama

kubectl create ns teamb

kubectl get ns

**How to deploy pod/deployment/service in custom namespace "dev"**

vi mydeployment-service-dev-ns.yaml

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: nginx-deployment**

**labels:**

**app: nginx**

**namespace: dev**

**spec:**

**replicas: 3**

**selector:**

**matchLabels:**

**app: nginx**

**author: prakash**

**template:**

**metadata:**

**labels: #labels should match with matchLabels**

**app: nginx**

**author: prakash**

**spec:**

**containers:**

**- name: nginx**

**image: nginx:1.14.2**

**ports:**

**- containerPort: 80**

**---**

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: mynginxapp-service**

**namespace: dev #first create the namespace**

**spec:**

**selector: #should match with Deployment's template label**

**app: nginx**

**author: prakash**

**ports:**

**- name: http**

**protocol: TCP**

**port: 80**

**targetPort: 80**

**type: NodePort**

kubectl apply -f mydeployment-service-dev-ns.yaml

kubectl get pod -n dev

kubectl get service -n dev

kubectl get deployment -n dev

kubectl delete ns <namespace>

kubectl delete ns teama

kubectl get pod (without -n : this command list resources of default namespace by default)

**How to provide/restrict access on namespace:**

[**https://aws.amazon.com/premiumsupport/knowledge-center/eks-iam-permissions-namespaces/**](https://aws.amazon.com/premiumsupport/knowledge-center/eks-iam-permissions-namespaces/)

**============Helm chart:**

[**https://opensource.com/article/20/5/helm-charts**](https://opensource.com/article/20/5/helm-charts)

[**https://empathy.co/blog/hat-ci-cd-for-deploying-cloud-native-applications/**](https://empathy.co/blog/hat-ci-cd-for-deploying-cloud-native-applications/)

**Helm is a package manager in kubernetes,**

**Q. How do you deploy in kubernetes? -> do you write k8s manifest at your own, who manages the connectivity between resources, rollout (trigger cicd pipeline )**

**and rollback (trigger cicd pipeline to rollback but we will trigger the pipeline when needed)**

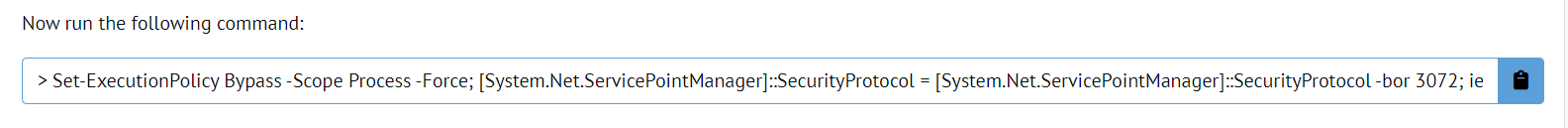
ans: using helm chart

**Install Helm:**

**1. Chocolatey is a package manager for windows OS: install choco**

[**https://chocolatey.org/install**](https://chocolatey.org/install)

**Navigate to download link**

****

**open powershell as admin: (better copy from above link)**

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

**2. install helm using choco**

choco install kubernetes-helm

**3. Create helm chart**

helm create myfirstchart

**image Pull policy:** [**https://kubernetes.io/docs/concepts/containers/images/**](https://kubernetes.io/docs/concepts/containers/images/)

**IfNotPresent**: the image is pulled only if it is not already present locally.

**Always** :every time the kubelet launches a container, the kubelet queries the container image registry to resolve the name to an image digest. If the kubelet has a container image with that exact digest cached locally, the kubelet uses its cached image; otherwise, the kubelet pulls the image with the resolved digest, and uses that image to launch the container.

**Never**: the kubelet does not try fetching the image. If the image is somehow already present locally, the kubelet attempts to start the container; otherwise, startup fails. See pre-pulled images for more details

Once your helm chart is created-> go to the values.yaml & update image repository, Service type, Port, Replica count and depending on your requirement.

so values will be replaced automatically during helm deploy (during pod/service/deployment/ingress/hpa etc creation)

**Once your values.yaml file is updated, just deploy the helm chart , it will create all the resources for us.**

cd myfirstchart ----> go inside the chart

helm install myfirstchart **.** ----> to create chart

helm ls

kubectl get pod

kubectl get deployment

**helm --help**

**Available Commands:**

**completion generate autocompletion scripts for the specified shell**

**create create a new chart with the given name**

**dependency manage a chart's dependencies**

**env helm client environment information**

**get download extended information of a named release**

**help Help about any command**

**history fetch release history**

**install install a chart**

**lint examine a chart for possible issues**

**list list releases**

**package package a chart directory into a chart archive**

**plugin install, list, or uninstall Helm plugins**

**pull download a chart from a repository and (optionally) unpack it in local directory**

**push push a chart to remote**

**registry login to or logout from a registry**

**repo add, list, remove, update, and index chart repositories**

**rollback roll back a release to a previous revision**

**search search for a keyword in charts**

**show show information of a chart**

**status display the status of the named release**

**template locally render templates**

**test run tests for a release**

**uninstall uninstall a release**

**upgrade upgrade a release**

**verify verify that a chart at the given path has been signed and is valid**

**version print the client version information**

**just update values.yaml (ex. image or port or replica) then deploy newer chart version**

helm upgrade <chart name> .

**you should be able to see the change.**

kubectl get deployment

helm ls ----> you will get revision 2 (deploy helm chart two times)

helm rollback <chartname> 1 ---> 1 rollback to revision 1

**once rollback done , you can check the rollback status**

helm rollback <chartName> --->rollback will point to revision 3

**try to upload a helm chart to ECR/GitHub/Company repository ???**

**helm history <chart name> ---> list all the history**

**delete helm chart**

**helm delete <chat name>**

**=========18-2-2023**

**Prometheus, Grafana**

Q. What do you understand by monitoring the AWS infrastructure (AWS services such ec2, vpc, storage, database, EKS,EBS etc)

Q. What do you understand about Prometheus and Grafana??

Q. Difference between Cloud Watch (AWS native tool) & Prometheus Grafana (Open Source tool)

Q. Explain Prometheus Grafana working model or Flow Diagram ?

Q. What is the port number of Prometheus Grafana ?

Ans: grafana 3000, prometheus 9090

Q. How do you troubleshoot in AWS

* How to kill process
* How to clean temp data
* How to check cpu & memory usage (linux command)
* How to check disk usage (linux command)
* How to clean storage
* What do u understand by swap memory
* How to check cpu and memory logs (command linux)

**CloudWatch**: You can only monitor AWS services, you can not monitor On-premises, or other cloud services

(hybrid model -> few services on Cloud and few service on On-premise

Multi Cloud: service will run on multiple cloud provider)

You can not monitor memory in cloud watch

**Prometheus Grafana**: Monitoring tool, You can monitor AWS resources, Azure, GCP, On premises

Memory can be monitored.

You can monitor kubernetes pods, worker node, EKS cluster, ec2, RDS, EBS etc

(there are other alternative of cloudwatch data dog, nagios, ELK)

**What is Grafana:** Grafana is a tool whose purpose is to compile and visualize data through dashboards from the data sources available throughout an organization. From these dashboards it handles a basic alerting functionality that generates visual alarms.

**Prometheus:** Prometheus is an open-source event monitoring and alerting tool.

it collects the data (it monitors resources) from different different sources & passes to Grafana