**KUBERNETES INSTALLATION**

**Launch 3 instance**

Master

Worker 1

Worker 2

After launching the 3 instance under security group by default is ssh edit inbound& outbound to Alltraffic  , anywhere & save it.

Docker installation

**Step -1**

In master

**# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add**

repeats this command in another two nodes

step -2

**# sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"**

repeats this command in another two nodes

**step -3**

# sudo apt-get install -y docker-ce=18.06.1~ce~3-0~ubuntu

 repeats this command in another two nodes

install kubernetes binaries

step-4

adding gpg key

# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add

**step-5**

cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

> deb https:apt.kubernetes.io/ kubernetes-xenial main

> EOF  --enter

**Step-6**

# Sudo apt-get update

 # sudo apt-get install -y kubelet=1.12.2-00 kubeadm=1.12.2-00 kubectl=1.12.2-00

**step -7**

# sudo apt-mark hold kubelet

# sudo apt-mark hold kubeadm

# sudo apt-mark hold kubectl

so we can hold them

**step -8  step (only on master)**

initialing the master node

# sudo kubeadm init --pod-network-cidr=10.244.0.0/16

from this ,we will get result like this

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudochown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node

as root:

(kubeadm join 10.0.1.101:6443 --token vyjohn.ob1j7h339dtcffqb --discovery-token-ca-cert-hash sha256:a1036671f52aef0748717deb1a828921ed55e8451aa5c7e23e885cdecb999e74)

Need to copy the topline when u run this command on server sudo kubeadm init --pod-network-cidr=10.244.0.0/16 system will generate new everytime whenu run this command copy the top line and run it in both nodes.

Can create manually by

kubeadm token create --print-join-command

**step -9 (only on master)**

from above result, we have to run those on master ,

don’t run these commands in worker nodes ,u have to initialize only one master

# mkdir -p $HOME/.kube

# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

# sudo chown $(id -u):$(id -g) $HOME/.kube/config

**step -10 (only nodes)**

# sudo kubeadm join 10.0.1.101:6443 --token vyjohn.ob1j7h339dtcffqb --discovery-token-ca-cert-hash sha256:a1036671f52aef0748717deb1a828921ed55e8451aa5c7e23e885cdecb999e74

run this only on worker nodes

result: Run 'kubectl get nodes' on the master to see this node join the cluster.

**Step -11 (master & nodes)**

Check status by running kubectl get nodes

# echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf

net.bridge.bridge-nf-call-iptables=1

**sudo sysctl -p**

net.bridge.bridge-nf-call-iptables = 1

**step -12only on master**

kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/bc79dd1505b0c8681ece4de4c0d86c5cd2643275/Documentation/kube-flannel.yml>

finally check status

**# kubectl get nodes**

**Nodes don’t get redy until DNS is ready**

run this command on master its displays list of all nodes ready

**https://www.cloudtechnologyexperts.com/kubeadm-on-aws/**

**What is kubernetes?**

Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

 Kubernetes isa google product develoed by google

Advantages of kubernetes:

1. High availability

2. Load balance

**passwordless login from master to workernode**

Run on master)

Be on /root home dir – rum on masternode

# ssh <nodeipaddress> --- diaplays password denied

Then generate the password to exchange the key to create the passwordless login

# ssh-key –t rsa -- enter + enter+enter – by default diplays the key image and will save on

/root home directory (gen.key & creates .ssh under hom dir)

# ls –la – displays the hidden file

# cd .ssh

#ls –l or ll   will displays following lines

drwx------ 2 root root 4096 Feb 27 11:03 ./

drwx------ 5 root root 4096 Feb 25 23:38 ../

-rw------- 1 root root  544 Feb 25 22:17 authorized\_keys

-rw------- 1 root root 1679 Feb 27 11:00 id\_rsa  (private key)

**-rw-r--r-- 1 root root  401 Feb 27 11:00 id\_rsa.pub  (public key)**

-rw-r--r-- 1 root root  222 Feb 27 11:03 known\_hosts

# cat id\_rsa.pub    or  vi id\_rsa.pub- ---will display the keys copy the key and oaste in node

  ( Run the authorized\_keys in workernode

#  vi authorized\_keys   + esc press o and paste the key on workernode +save & exit

Try to check  login to workernode

# ssh <nodeipaddress> --- (logs into workernode and can run all the workernode command on master or any error  needs to change the password authentication  yes on worker node  follow the steps in node configuration )

# exit --- exiting from master/ node

**(Run on workernodes)**

# cd .ssh

#ls –l or ll   will displays following lines

drwx------ 2 root root 4096 Feb 27 11:03 ./

drwx------ 5 root root 4096 Feb 25 23:38 ../

**-rw------- 1 root root  544 Feb 25 22:17 authorized\_keys**

-rw------- 1 root root 1679 Feb 27 11:00 id\_rsa  (private key)

-rw-r--r-- 1 root root  401 Feb 27 11:00 id\_rsa.pub  (public key)

-rw-r--r-- 1 root root  222 Feb 27 11:03 known\_hosts

# vi authorized\_keys  or  cat >> authorized\_keys

Copy the key from master node from # cat id\_rsa and paste it in authorized\_keys file

Esc + o + and paste the key   +save & exit.

# vi /etc/ssh/sshd\_config

Press   /password it will search the word in file content  go to

Password authentication line  by default it no enable  it to yes with insert mode

**Password authentication no**

**Change it to**

**Password authentication yes**

Save & exit

Restart the ssh services and then check the status by following commands

#  systemctl restart ssh

#  systemctl status ssh

ssh-keygen –t rsa  -- (gen.key & creates .ssh under hom dir)

cd .ssh +ll +open  vi id\_rsa.pub and copy the complete key.

Login to workernode

Cd .ssh + vi authorized\_keys (paste the key) & save it

Open ssh configuration file

Enble password authentication yes

Vi  /etc/sss/sshd\_config

Systemctl restart ssh

Check it master to login workernode ipaddress

Ssh <nodeipaddress>

------------------------------------------------

What is Pod?

Is a environment where container will be created and share the same ip.

**Kubernetse commands:**

# kubectl get nodes—display the lsit of nodes commected to master with ready when DNS SERVER is

                                        Ready

# kubectl get pods --- displays the list of pods created

#  kubectl ger pods –o wide ---- displays list of pods with full fledge

# kubectl describe pods ----displays list of pods when they are created, schedule, started & status  or  in

                                                Other  word group of resources

Which apiVersion should I use**?**

<https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-apiversion-definition-guide.html>

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

<https://kubernetes.io/docs/tasks/run-application/run-stateless-application-deployment/#creating-and-exploring-an-nginx-deployment>

<https://kubernetes.io/docs/concepts/services-networking/connect-applications-service/#creating-a-service>

docker compose

|  |  |
| --- | --- |
| **Compose file format** | **Docker Engine** |
| 1 | 1.9.0+ |
| 2.0 | 1.10.0+ |
| 2.1 | 1.12.0+ |
| 2.2, 3.0, 3.1, 3.2 | 1.13.0+ |
| 2.3, 3.3, 3.4, 3.5 | 17.06.0+ |
| 2.4 | 17.12.0+ |
| 3.6 | 18.02.0+ |
| 3.7 | 18.06.0+ |

**Compose file format compatibility matrix**

**Changes**

https://www.instana.com/blog/stans-robot-shop-sample-microservice-application/

<https://github.com/linuxacademy/robot-shop>

**Dockerfile**

FROM python:3.4-alpine  
ADD . /code  
WORKDIR /code  
RUN pip install -r requirements.txt  
CMD ["python", "app.py"]

**Kubernetes interview  questions**

**1.What is kubernetes and why is it popular?**

This is a basic interview question asked by the interviewer to check candidate’s knowledge around Kubernetes. The interviewer expects the candidate to be aware of why Kubernetes is HOT in the market and what problem does Kubernetes solve for all of us. One can begin the answer with -

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery.  
Kubernetes is a HUGE open source project with a lot of code and functionalities. The primary responsibility of Kubernetes is container orchestration. That means making sure that all the containers that execute various workloads are scheduled to run physical or virtual machines. The containers must be packed efficiently following the constraints of the deployment environment and the cluster configuration. In addition, Kubernetes must keep an eye on all running containers and replace dead, unresponsive, or otherwise unhealthy containers.    
  
  
Said that Kubernetes is rightly a platform for managing application containers across multiple hosts. It provides lots of management features for container-oriented applications, such as  rolling deployment, resource, and volume management. Same as the nature of containers, it's designed to run anywhere, so we're able to run it on a bare metal, in our data center, on the public cloud, or even hybrid cloud.



Kubernetes considers most of the operational needs for application containers. The Top 10 Reasons why Kubernetes is so popular are as follow:

* Largest Open Source project in the world
* Great Community Support
* Robust Container deployment
* Effective Persistent storage
* Multi-Cloud Support(Hybrid Cloud)
* Container health monitoring
* Compute resource management
* Auto-scaling Feature Support
* Real-world Use cases Available
* High availability by cluster federation

**2.Why should i use kubernetes?**

his is a bit tricky question. The intention of the interviewer is trying to understand candidate’s understanding of why should one head towards Kubernetes adoption inside the company or on the cloud.

With the advent of microservice architecture, users to individually scale key functions of an application and handle millions of customers. On top of this, technologies like Docker containers emerged in the enterprise, creating a consistent, portable, and easy way for users to quickly build these microservices. While Docker continued to thrive, managing these microservices & containers became a paramount requirement. All you need is a robust orchestration platform which can manage those containers which host your entire application. Kubernetes comes to a rescue.

Kubernetes is a robust orchestration platform which brings a number of features and which can be thought of as:

* As a container platform
* As a microservices platform
* As a portable cloud platform and a lot more.

Kubernetes provides a container-centric management environment. It orchestrates computing, networking, and storage infrastructure on behalf of user workloads. This provides much of the simplicity of Platform as a Service (PaaS) with the flexibility of Infrastructure as a Service (IaaS), and enables portability across infrastructure providers. Below are the list of features which Kubernetes provides -

* Service Discovery and load balancing: Kubernetes has a feature which assigns the containers with their own IP addresses and a unique DNS name, which can used to balance the load on them.
* Planning: Placement of the containers on the node is a crucial feature on which makes the decision based on the resources it requires and other restrictions.
* Auto Scaling: Based on the CPU usage, vertical scaling of applications is automatically triggered using the command line.
* Self Repair: This is an unique feature in the Kubernetes which will restart the container automatically when it fails. If the Node dies, then containers are replaced or re-planned on the other Nodes. You can stop the containers, if they don't respond for the health checks.
* Storage Orchestration: This feature of Kubernetes enables the user to mount the network storage system as a local file system.
* Batch execution: Kubernetes manages both batch and CI workloads along with replacing containers that fail.
* Deployments and Automatic Rollbacks: During the configuration changes for the application hosted on the Kubernetes, progressively monitors the health to ensure that it does not terminate all the instances at once, it makes an automatic rollback only in the case of failure.
* Configuration Management and Secrets: All classifies information like keys and passwords are stored under module called Secrets in Kubernetes. These Secrets are used specially while configuring the application without having to reconstruct the image.

3.how kubernetes is related to docker?

This is one of the most important question ever asked in an interview. Though we compare Docker Vs Kubernetes, it is an apple-to-orange comparison. Reason - They are both fundamentally different technologies but they work very well together, and both facilitate the management and deployment of containers in a distributed architecture.

**Let me elaborate -**

Docker started as a GITHUB project back in 2013(which is almost 5+ years from now). Slowly it grew massively with HUGE contributors across the world. Today it is a platform which is shipped as both - an open source as well as a commercial product. The orchestration is just a mere feature of Docker Enterprise Edition.

But if we really want to study how K8s is related to Docker, then the most preferred answer would be -

Docker CLI provides the mechanism for managing the life cycle of the containers. Where as the docker image defines the build time framework of runtime containers. CLI commands are there to start, stop, restart and perform lifecycle operations on these containers. Containers can be orchestrated and can be made to run on multiple hosts. The questions that need to be answered are how these containers are coordinated and scheduled? And how will the application running in these containers will communicate each other?

 Kubernetes is the answer. Today, Kubernetes mostly uses Docker to package, instantiate, and run containerized applications. Said that there are various another container runtime available but Docker is the most popular runtime binary used by Kubernetes.

Both Kubernetes and Docker build a comprehensive standard for managing the containerized applications intelligently along with providing powerful capabilities.Docker provides a platform for building running and distributing Docker containers. Docker brings up its own clustering tool which can be used for orchestration. But Kubernetes is a orchestration platform for Docker containers which is more extensive than the Docker clustering tool, and has capacity to scale to the production level. Kubernetes is a container orchestration system for Docker containers that is more extensive than Docker Swarm and is meant to coordinate clusters of nodes at scale in production in an efficient manner.  Kubernetes is a plug and play architecture for the container orchestration which provides features like high availability among the distributed nodes

4. how kubernetes simplify the containerized application deployment process?

A application deployment requires , web tier , application tier and database tier . All these requirements will spawn multiple containers and these containers should communicate among each other . Kubernetes cluster will take care of the whole system and orchestrates the container needs .

Let us look at a quick WordPress application example. WordPress application consists of frontend(WordPress running on PHP and Apache) and backend(MySQL). The below YAML file can help you specify everything you will need to bring WordPress Application in a single shot:

apiVersion: v1

kind: Service

metadata:

 name: wordpress

 labels:

   app: wordpress

spec:

 ports:

   - port: 80

 selector:

   app: wordpress

   tier: frontend

 type: LoadBalancer

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

 name: wp-pv-claim

 labels:

   app: wordpress

spec:

 storageClassName: manual

 accessModes:

   - ReadWriteOnce

 resources:

   requests:

     storage: 2Gi

---

apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2

kind: Deployment

metadata:

 name: wordpress

 labels:

   app: wordpress

spec:

 storageClassName: manual

 selector:

   matchLabels:

     app: wordpress

     tier: frontend

 strategy:

   type: Recreate

 template:

   metadata:

     labels:

       app: wordpress

       tier: frontend

   spec:

     containers:

  - image: wordpress:4.8-apache

       name: wordpress

       env:

       - name: WORDPRESS\_DB\_HOST

         value: wordpress-mysql

       - name: WORDPRESS\_DB\_PASSWORD

         valueFrom:

           secretKeyRef:

             name: mysql-pass

             key: password

       ports:

       - containerPort: 80

         name: wordpress

       volumeMounts:

       - name: wordpress-persistent-storage

         mountPath: /var/www/html

     volumes:

     - name: wordpress-persistent-storage

       persistentVolumeClaim:

         claimName: wp-pv-clai

I assume that you have n-node Kubernetes cluster running in your infrastructure. All you need is to run the below command:

kubectl create -f wordpress-deployment.yaml

           That’s it. Browse to http://<IP>:80 port to open to see WordPress App up and running. Hence, we saw that how Kubernetes simplifies the application deployment.

5.how to install  kubernetes?

Install below packages on all of your machines:

* kubeadm: the command to bootstrap the cluster.
* kubelet: the component that runs on all of the machines in your cluster and does things like starting pods and containers.
* kubectl: the command line util to talk to your cluster.

Note : kubeadm will not install or manage kubelet or kubectl for you, so you will need to ensure they match the version of the Kubernetes control panel you want kubeadm to install for you.

 If you do not, there is a risk of a version skew occurring that can lead to unexpected, buggy behavior.

However, one minor version skew between the kubelet and the control plane is supported, but the kubelet version may never exceed the API server version. For example, kubelet running 1.7.0 should be fully compatible with a 1.8.0 API server, but not vice versa.

Below is the example for installing in Debian or Ubuntu flavours

# apt-get update && apt-get install -y apt-transport-https curl

# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main

EOF

# apt-get update

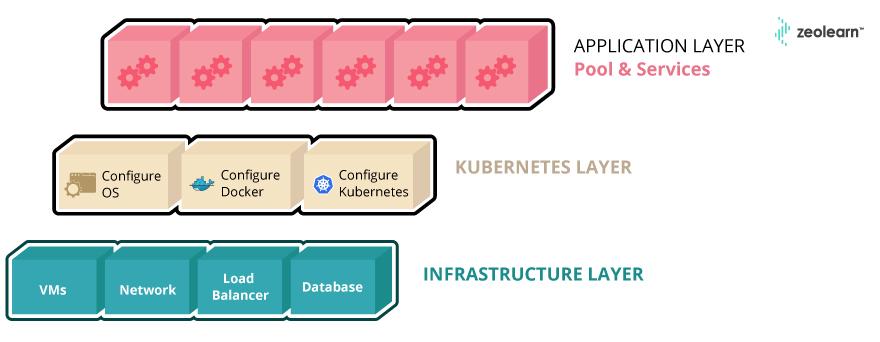
# apt-get install -y kubelet kubeadm kubectl

# apt-mark hold kubelet kubeadm kubectl

Configure cgroup driver used by kubelet on Master Node  
When using Docker, kubeadm will automatically detect the cgroup driver for the kubelet and set it in the /var/lib/kubelet/kubeadm-flags.env file during runtime.  
If you are using a different CRI, you have to modify the file /etc/default/kubelet with your cgroup-driver value, like so:  
KUBELET\_KUBEADM\_EXTRA\_ARGS=--cgroup-driver=<value>  
This file will be used by kubeadm init and kubeadm join to source extra user-defined arguments for the kubelet.  
Please mind, that you only have to do that if the cgroup driver of your CRI is not cgroupfs, because that is the default value in the kubelet already.  
Restarting the kubelet is required:

6. Explain the architecture layers of kubernetes?

Kubernetes can be visualized as a system built in layers, with each higher layer abstracting the complexity found in the lower levels

**Base Layer**

At the base, Kubernetes makes a cluster which is a collection of hosts storage and networking resources that Kubernetes uses to run the various workloads that comprise the system . Cluster groups together a large fleet of machines into a single unit that can be consumed .

**Mid Layer**

The machines in the cluster are each given a role within the Kubernetes ecosystem.

The MASTER is the control plane of Kubernetes having functionalities like Authorization and authentication , RESTful API entry point. Master comprises of components, such as an API server , scheduler, and controller manager. The master is responsible for the global, cluster-level scheduling of pods and handling of events.

* The other machines in the cluster are designated as nodes:

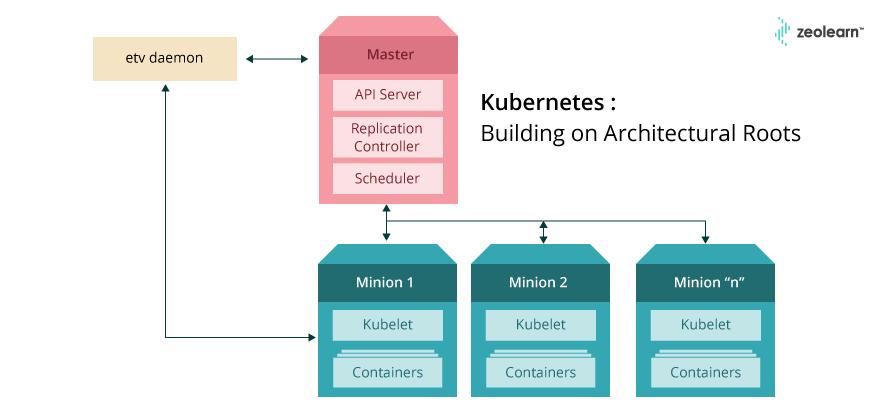
Nodes are managed by a Kubernetes master. The nodes are worker bees of Kubernetes and do all the processing and hardwork . Each node has a container runtime , receives instructions from master node , creates / destroys container as per the workload and enroutes traffic appropriately

**Application Layer/ Final Layer**

Kubernetes itself is a somewhat complicated distributed system which runs on API approach .

* To run an application , a plan is submitted in yaml or json
* The master server runs the plan by examining the requirements and current state of the cluster
* All user interact with the cluster by the help of API ecosystem implemented by control plane of the master server
* Next comes the scheduler and controller-manager components that keeps the cluster functioning correctly
* In the last its the workers that will take the pain and provides the output for the job

**7.List out the master server components of kubernetes?**



Kubernetes is a combination of multiple parts working together to get the container job done and the most vital part of it is Master node . The node acts as brain to the cluster and manages the whole ecosystem .  
Master connects to etcd via HTTP or HTTPS to store the data and also connects flannel to access the container application.  
Worker nodes engage with master via HTTP or HTTPS to get a command and report the status.  
Overlay network makes connections of their container applications. All of it will be discussed below for more in-depth  
Below are the mentioned components :  
etcd

* The heart of any Kubernetes cluster that implements a distributed key value store where all of the objects in a kubernetes cluster are persisted .
* It works on a algorithm which has replication techniques across servers to maintain the data stored in etcd .
* Optimistic concurrency is also used to compare-and-swap data across etcd server , when a user reads and update a value , the system checks that no other component in the system has updated the same value . This technique removes the locking mechanism that increases the server throughput .
* Another technique known as watch protocol , which accounts for changes made in key value pair in etcd directory . Its improves efficiency to the client as it wait for the changes and then react to the change without continuous polling to the server .

kube-apiserver

As the name connects , its a server that provides an HTTP- or HTTPS-based RESTful API that is allowed to have direct access to the Kubernetes cluster .

* Its a connector between all the kubernetes components and mediates all interactions between clients and the API objects stored in etcd .
* Api server database is external to it , so it is a stateless server which is replicated 3 times to implement fault-tolerance
* The APIs are exposed and managed by the server , the characteristics of those API requests must be described so that the client and server know how to communicate .
* Define API pattern where the request is defined like api paths or groups.
* Internal loops are responsible for background operations like CRD (Custom Resource Definitions) which inherently creates new paths for API requests

kube-controller-manager

The controller manager is a general service that has many responsibilities.

* Controller manager is a collection of control loops rolled up into one binary
* Manages Kubernetes nodes
* The control loops needed to implement the functionality like replica sets and deployments are run by Manager
* Creates and updates the Kubernetes internal information
* changes the current status to the desired status

kube-scheduler

* It is a simple algorithm that defines the priority to dispatch and is responsible for scheduling pods into nodes .
* is continuously scanning the API server (with watch protocol) for Pods which don’t have a nodeName and are eligible for scheduling
* Node affinity provide a simple way to guarantee that a Pod lands on a particular node
* Predicates is a concept that helps in making correct resource requirements for the pods
* Data locality

**8.**[**List the Node Server Components of Kubernetes ?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-583)

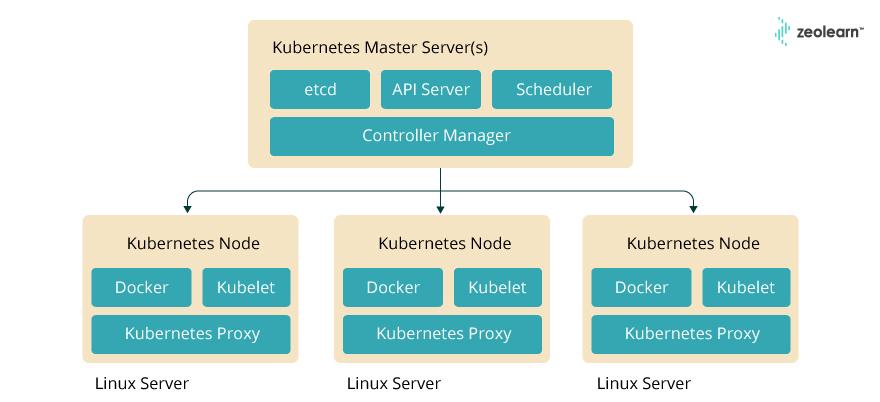
In Kubernetes, servers that perform work by running containers are known as nodes. Execution of jobs and reporting the status back to the master are the primary tasks on Node server .

kubelet

The main process on Kubernetes node that performs major container operations .

* The Kubelet is the node-daemon that communicates with Kubernetes master for all machines that are part of a Kubernetes cluster.
* It periodically access the controller to check and report the status of the cluster
* It merges the available CPU, disk and memory for a node into the large Kubernetes cluster.
* Communicates the state of containers back up to the api-server for control loops to observe the current state of the containers.

kube-proxy

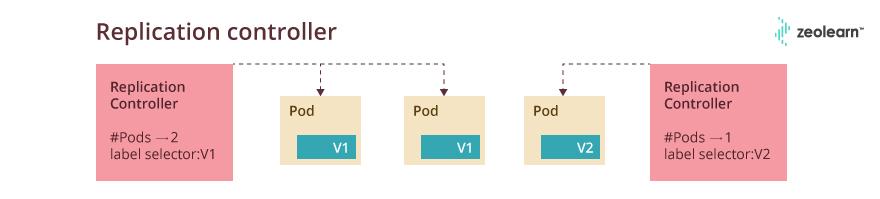
* The kube proxy implements load-balancer networking model on each node.
* It makes the Kubernetes services locally and can do TCP and UDP forwarding.
* The kube-proxy programs the network on its node, so that network requests to the virtual IP address of a service, are in-fact routed to the endpoints which implement this service
* It finds cluster IPs via environment variables or DNS.
* Routes traffic from Pods on the machine to Pods, anywhere in the cluster  
  [9.List out Kubernetes Objects and Workloads?](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-584)
* kubernetes object model provides set of features to manage containers and interact with instances.
* Here are the few listed different types of objects used to define workloads.
* Pods : Pod is the basic unit in the Kubernetes Object Model. In Kubernetes containers are not attached to hosts directly, instead one or more containers are tightly coupled form an encapsulated object called pod.
* Replication Controllers and Replication Sets : know as replication of pods. These are created from pod templates and can be horizontally scaled by controllers known as replication controllers and replication sets.
* Deployments: Deployments under Kubernetes refers to one or multiple identical pods with no
* distinctive identities.It runs multiple copies/replicas of application and is intelligent to automatically replace any failed instances It ensure that one or more copies of instances of application are always available to serve user requests. Deployments are managed by Deployment controller.
* Stateful sets: It is a controller in Kubernetes which provides a distinctive identity to its pods and responsible for managing the deployment and scaling of a set of Pods. This controller intend to be used with stateful applications(like database) and distributed systems.
* Daemon Sets: It ensure that all the cluster nodes run a copy of a Pod.Whenever you add any nodes to Kubernetes cluster, DaemonSets ensures that Pods get automatically added to the new nodes as needed. Hence, it takes responsibility to manage multitude of replicated Pods.
* Jobs & Cron Jobs: A Job under Kubernetes is responsible to create one or multiple pods and ensure that a specific number of those pods successfully terminate. Job automatically tracks the successful completions.

[10.Explain the Replication Controllers and Replication Sets in Kubernetes ?](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-585)

ReplicationController makes sure that a specified number of pod replicas are running at any point of time. Specifically ReplicationController ensures that a pod or set of pods are homogeneous in nature and are always up and running.

ReplicationController always maintains desired number of pods, if the number exceeds then it will terminate extra pods, or if the number decreases extra pods will be created. The pods maintained by a ReplicationController are automatically replaced if they fail, are deleted, or are terminated. Replication controllers can also perform rolling updates to roll over a set of pods to a new version one by one, minimizing the impact on application availability.

Replication sets are an iteration on the replication controller design with greater flexibility in how the controller identifies the pods it is meant to manage. Replication sets are much more advanced than ReplicationController as they have greater replica selection capabilities, but they dont have the rolling updates capabilities.



Creating a replication controller  
To create replication controllers, use the subcommand run after kubectl.

//  kubectl run <REPLICATION CONTROLLER NAME> --images=<IMAGE NAME> [OPTIONAL\_FLAGS]

# kubectl run my-first-rc --image=nginx

CONTROLLER    CONTAINER(S) IMAGE(S)   SELECTOR REPLICAS

my-first-rc   my-first-rc nginx      run=my-first-rc 1

Above command is creating a replication controller by image nginx .The name, my-first-rc, must be unique in all replication controllers.

Without specified number of replicas, the system will only build one pod as its default value

[**11.Explain Pods in Kubernetes Context**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-586)

Pod is a single or bunch of containers that is controlled as a single application

* Containers inside the Pod operate closely together and share a common life cycle, but has to be scheduled on the same node.
* Pods are managed as a unit and share common environment wrt volume and IP address space.
* Every Pods consists of master container that satisfies of balancing the workload among the other containers that facilitate to orchestrate other related tasks.
* For example, a pod may have one container running the primary application server and a helper container pulling down files to the shared file system when changes are detected in an external repository.
* Users are recommended not to manage pods themselves, because they might miss few features specifically needed in applications.
* Users are advised to operate with the objects that use pod templates as base components and add additional functionality to them.

Use Secrets in Pods

To use Secrets inside Pods, choose to expose pods in environment variables or mount the Secrets as volumes.

In terms of accessing Secrets inside a Pod, add env section inside the container spec

// using access-token Secret inside a Pod

# cat 2-7-2\_env.yaml

apiVersion: v1

kind: Pod

metadata:

   name: secret-pod-env

spec:

  containers:

  - name: ubuntu

    image: ubuntu

    command: ["/bin/sh", "-c", "while : ;do echo $ACCESS\_TOKEN; sleep 10; done"]

    env:

        - name: ACCESS\_TOKEN

          valueFrom:

            secretKeyRef:

              name: access-token

              key: 2-7-1\_access-token

// create a pod

# kubectl create -f 2-7-2\_env.yaml

pod "secret-pod-env" created

Below example, expose 2-7-1\_access-token key in access-token Secret as ACCESS\_TOKEN environment variable, and print it out through a while infinite loop .

// check stdout logs

# kubectl logs -f secret-pod-env

9S!g0U616456r

[**12.Explain stateful sets**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-587)

During regular deployment process, all the instances of a pod are identical,and these stateless applications can be easily scaled up and down. In a PetSet, each pod is unique and is been assigned with unique identifier that needs to be maintained. This is technique is generally used for more stateful applications.

Example creating Stateful Set  
Use the following command to get to start the creation of this StatefulSet

$ kubectl create -f abc-statefulset.yaml

Use get subcommand to see stateful sets:

$ kubectl get statefulsets

$ kubectl get pods

Get the volumes the set has created and claim for each pod :

$ kubectl get pv

[**13.Explain Daemon sets**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-588)

A DaemonSet is a set of pods that is run only once on a host. It's used for host-layer features, for instance a network, host monitoring or storage plugin or other things which you would never want to run more than once on a host.

**14.**[**Explain Master**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-589)

Master is the central control point that provides a unified view of the cluster. There is a single master node that control multiple minions.

Master servers work together to accept user requests, determine the best ways to schedule workload containers, authenticate clients and nodes, adjust cluster-wide networking, and manage scaling and health checking responsibilities

[**15.Explain Minions**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-590)

A node is a worker machine in Kubernetes, previously known as a minion. A node may be a VM or physical machine, depending on the cluster. Each node contains the services necessary to run pods and is managed by the master components. The services on a node include the container runtime, kubelet and kube-proxy.

[**16.ExplainVolumes and Persistent Volumes in kubernetes**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-591)

A Kubernetes volume, on the other hand, the same as the Pod that encloses it. Consequently, a volume outlives any Containers that run within the Pod, and data is preserved across Container restarts. Of course, when a Pod ceases to exist, the volume will cease to exist, too. Perhaps more importantly than this, Kubernetes supports many types of volumes, and a Pod can use any number of them simultaneously.

The PersistentVolume subsystem provides an API for users and administrators that abstracts details of how storage is provided from how it is consumed. To do this we introduce two new API resources:PersistentVolume and PersistentVolumeClaim.

A PersistentVolume (PV) is a storage in the cluster that has to be provisioned by an administrator and it is a cluster resource. PVs are volume plugins like Volumes, but have a life cycle independent of any individual pod that uses the PV.

This API object captures the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system.

 A PersistentVolumeClaim (PVC) is a request for storage by a user. It is similar to a pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific levels of resources (CPU and Memory). Claims can request specific size and access modes

[**17.Explain Labels and Annotations in kubernetes**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-592)

Label in Kubernetes is meaningful tag word that can be attached to Kubernetes objects to make them as a part of a group. Labels can be used for working on different instances for management or routing purposes.

For example, the controller-based objects use labels to mark the pods that they should operate on. Micro Services use labels to understand the structure of backend pods they should route requests to.

Labels are key-value pairs. Each unit can have more than one label, but each unit can only have one entry for each key. Key is used as an identifier, but additionally can classify objects by other criteria based on  development stage, public accessibility, application version, etc activities.

Annotations attach arbitrary key-value information to an Kubernetes object. On the other hand labels should be used for meaningful information to match a pod with selection criteria, annotations contain less structured data. Annotations are a way of adding more metadata to an object that is not helpful for selection purposes.

[**18.What is a service role in kubernetes components**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-593)

A service is an abstraction for pods. It provides a  virtual IP (VIP) address. It allow clients to reliably connect to the containers running in the pods, using the Virtual IP address. It is bascially a component that groups together collection of pods.

To display services under kubernetes, you can run the below command:

$kubectl get services

[**19.Explain Kubelet**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-594)

Each node runs services to run containers and be managed from the master. In addition to Docker, Kubelet is another key service installed there. It reads container manifests as YAML files that describes a pod. Kubelet ensures that the containers defined in the pods are started and continue running.

[**20.What Is The Gke?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-595)

Google Container Engine (GKE) is a management and orchestration system for Docker container and container clusters that run within Google’s public cloud services. Google Container Engine is based on Kubernetes, Google’s open source container management system.

[**21.What happens when a master fails? What happens when a worker fails?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-596)

Whenever master node under kubernetes fails, the cluster still remain in an operational mode. It doesn’t affect pod creation or service member changes. If worker node fails, master stop receiving updates from worker node.

[**22.How does DNS work in Kubernetes?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-597)

DNS is a built-in service in Kubernetes. It gets  launched automatically when Kubernetes is setup for the first time. Kubernetes Domain Name Server schedules a DNS Pod and Service on the cluster, and setup the kubelets to inform individual containers to use the DNS Service’s IP to resolve DNS names. Every Service which gets defined in the Kubernetes cluster (including the DNS server itself) is assigned with a DNS name. By default, a client Pod’s DNS search list will include the Pod’s own namespace and the cluster’s default domain. For E.g. if we have a Service named serve1 in the Kubernetes namespace ns1. A Pod running in namespace ns1 can look up this service by simply doing a DNS query for serve1. A Pod running in namespace **collab** can look up this service by doing a DNS query for serve1.ns1.

[**23.How do you build a High Availability (HA) cluster?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-623)

Add nodes in a HA cluster in kubernetes

Once the masters are ready, nodes can be added into the system. The node should be finished with the prerequisite configuration as a worker node in the kubeadm cluster.

Need to start kublet

$ sudo systemctl enable kubelet && sudo systemctl start kubelet

Run  the join command as below . However, please change the master IP to the load balancer one:

// join command

$ sudo kubeadm join --token <CUSTOM\_TOKEN> <LOAD\_BALANCER\_IP>:6443 --discovery-token-ca-cert-hash sha256:<HEX\_STRING>

Then go to the first master or second master to check the nodes' status:

// see the node is added

$ kubectl get nodes

NAME       STATUS ROLES     AGE VERSION

master01   Ready master    4h v1.10.2

master02   Ready master    3h v1.10.2

node01     Ready <none>    22s v1.10.2

[**24.How do I determine the status of a Deployment?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-624)

 use kubectl get deployment <deployment>. If the DESIRED, CURRENT, UP-TO-DATE are all equal, then the Deployment has completed.

[**25.How do I update all my pods if the image changed but the tag is the same?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-625)

Make sure your imagePullPolicy is set to Always(this is the default). That means when a pod is deleted, a new pod will ensure it has the current version of the image. Then refresh all your pods.

The simplest way to refresh all your pods is to just delete them and they will be recreated with the latest image. This immediately destroys all your pods which will cause a service outage. Do this with kubectl delete pod -l <name>=<value> where name and value are the label selectors your deployment uses.

A better way is to edit your deployment and modify the deployment pod spec to add or change any annotation. This will cause all your pods to be deleted and rescheduled, but this method will also obey your rollingUpdate strategy, meaning no downtime assuming your rollingUpdate strategy already behaves properly. Setting a timestamp or a version number is convenient, but any change to pod annotations will cause a rolling update. For a deployment named nginx, this can be done with:

PATCH='{"spec":{"template":{"metadata":{"annotations":{"timestamp":"'$(date)'"}}}}}'

kubectl patch deployment nginx -p "$PATCH"

It is considered bad practice to rely on the :latest docker image tag in your deployments, because using :latest there is no way to rollback or specify what version of your image to use. It's better to update the deployment with an exact version of the image and use --record so that you can use kubectl rollout undo deployment <deployment> or other commands to manage rollouts.

[**26.How do I debug a Pending pod?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-626)

Pending pod cannot be scheduled onto a node.

Performing command  kubectl describe pod <pod\_name> will help you undestand the problem.

kubectl logs <pod> can also be helpful.

Common reasons for pods getting stuck in Pending State are:

1) When the pod requesting more resources than are available, for example a pod has set a request for an amount of CPU or  that is not available

anywhere on any node. eg. requesting a 8 CPU cores when all your nodes only have 4 CPU cores.

Doing a kubectl describe node <node> on each node will also show already requested resources.

2) There are taints that prevent a pod from scheduling on your nodes.

3) The nodes have been marked unschedulable with kubectl cordon

4) There are no Ready nodes. kubectl get nodes will display the status of all nodes.

$ kubectl get po requests-pod-3

NAME             READY STATUS   RESTARTS AGE

requests-pod-3   0/1 Pending   0 4m

Examining why a pod is stuck at Pending with kubectl describe pod

$ kubectl describe po requests-pod-3

Name:       requests-pod-3

Namespace:  default

Node:       /

...

Conditions:

  Type           Status

  PodScheduled   False

...

Events:

... Warning  FailedScheduling    No nodes are available

                                                        that match all of the

                                                         following predicates::

                                                          Insufficient cpu (1).

[**27.What is KUBE proxy?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-627)

The Kubernetes network proxy runs on each node. Service cluster ips and ports are currently found through Docker-links-compatible environment variables specifying ports opened by the service proxy. There is an optional addon that provides cluster DNS for these cluster IPs.

[**28.How do I debug a Pending pod?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-628)

A Pending pod is one that cannot be scheduled onto a node. Doing a kubectl describe pod <pod> will usually tell you why. kubectl logs <pod> can also be helpful. There are several common reasons for pods stuck in Pending:

\*\* The pod is requesting more resources than are available, a pod has set a request for an amount of CPU or memory that is not available anywhere on any node. eg. requesting a 8 CPU cores when all your nodes only have 4 CPU cores. Doing a kubectl describe node <node> on each node will also show already requested resources. \*\* There are taints that prevent a pod from scheduling on your nodes. \*\* The nodes have been marked unschedulable with kubectl cordon \*\* There are no Ready nodes. kubectl get nodes will display the status of all nodes.

[**29.How do I rollback the Deployment?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-629)

Applying changes to a Deployment process with the --record flag then Kubernetes by default saves the previous Deployment activities in its history.

The below command will display all the prior Deployments,

***kubectl rollout history deployment <deployment>***

The last Deployment can be restored with the command,

***kubectl rollout undo deployment <deployment>***

The Deployments which are in progress can also be paused and resumed.

The moment new Deployment is applied, during this process a new ReplicaSet object is created which is slowly scaled up while the old ReplicaSet is scaled down.

We can get the ReplicaSet that has been rolled out with command

***kubectl get replicaset***

 Each ReplicaSet is named with the format -,

***kubectl describe replicaset <replicaset>a***

[**30.What is an Ingress Controller?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-beginner-630)

An Ingress Controller is a pod that can act as an inbound traffic handler.  Ingress Controller implemented as a HTTP reverse proxy. Prominent features are HTTP path and service based routing and SSL termination.

**31.**[**How do I expose a service to a host outside the cluster?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-631)

There are two ways by which we can expose a service to host outside the cluster:

1) Service type should be set to NodePort. Every node in the cluster will be made to listen on the specified NodePort, then all the traffic from any node will be forwarded from that NodePort to a random pod in the service.

2) Service type should be set to Load Balancer mode. Nodeport will be provisioned as mentioned in the above step, and then additional step which will automatically provision the load balancer in the cloud infrastructure.

**3**[**2.How does a Load Balancer service work?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-632)

Cloud Provider normally sets up LoadBalancer as a TCP Load Balancer. The Kubernetes controller manager helps to provision a load balancer in the cloud and configures all the Kubernetes nodes into the load balancer network.

As each node in the pod is assumed to be running kube-proxy and listening to the specific NodePort where it can forward incoming requests to a pod that is available for the service

Because each node is assumed to be running kube-proxy it should be listening on the appropriate NodePort and then it can forward incoming requests to a pod that is available for the service. As LoadBalancer is set to TCP by default, higher level features of HTTP LoadBalancer are not available.

**3**[**3.Is it possible to force the pod to run on a specific node?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-633)

By default, anti-affinity is not attempted by Kubernetes as its not a hard requirement,If its enforced, will schedule multiple pods on the same node

Pod can be constrained to only be able to run on particular nodes or to prefer to run on particular nodes. Several ways to achieve this, and all use label selectors to make the selection. Such constraints make scheduler to automatically do a reasonable placement, but in some circumstances we need more control on a node where a pod lands,

from two different services that communicate a lot into the same availability zone.

There are several ways to do this, and they all use label selectors to make the selection.

Generally such constraints are unnecessary, as the scheduler will automatically do a reasonable placement

**3**[**4.How do I get all the pods on a node?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-634)

You can use the following command to get all the pods on a node in kubernetes Cluster -

$ kubectl get po --all-namespaces  -o jsonpath='{range .items[?(@.spec.nodeName =="nodename")]}{.metadata.name}{"\n"}{end}'

**3**[**5.Can pods mount NFS volumes?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-635)

Yes, there's an example here of both an NFS client and server running within pods in the cluster:

Example:

Configuring NFS Server

Define NFS server pod and NFS service:

$ kubectl create -f nfs-server-pod.yaml

$ kubectl create -f nfs-server-service.yaml

The server exports /mnt/data directory, which contains dummy index.html. Wait until the pod is running!

Configuring NFS Client

See WEB server pod, which runs a simple web server serving data from the NFS. The pod assumes your DNS is set up and the NFS service is reachable as nfs-server.default.kube.local. Edit the yaml file to supply another name or directly its IP address (use kubectl get services to get it).

Finally, define the pod:

$ kubectl create -f web-pod.yaml

Now the pod serves index.html from the NFS server:

$ curl http://<the container IP address>/

Hello World!

**3**[**6.Is it possible to route traffic from outside the Kubernetes cluster directly to pods?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-636)

Yes, it is possible to route traffic to any or all of the K8 minions as well as Pods. Though there are multiple ways like Ingress, Load-Balancer & NodePort, but the Ingress seems to be a recommended and convenient solution. If you choose the NodePort service, you might want to implement a load balancer in front of your cluster as well.

**3**[**7.Why is my pod showing “pending” status?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-637)

It may be due to resource limitation. The cluser is unable to find a nod which has the available CPU and memory requests to satisfy the scheduler.

**3**[**8.What monitoring tools are popularly used in Kubernetes?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-638)

Tools like Heapster, Prometheus etc. is quite popular for monitoring Kubernetes. For measuring CPU and memory, Heapster is quite popular. It can be queries via REST API.

**3**[**9.Is it possible for containers within a pod to communicate with each other?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-639)

Yes, it is possible for containers to communicate within a pod. They reach other on localhost network. For E.g, if you have two containers within a pod, a MySQL container running on port 3306, and a PHP container running on port 80, the PHP container could access the MySQL one through localhost:3306.

**4**[**0.Is it possible to configure credentials to download images from a private docker registry?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-640)

Private registries may require keys to read images from them. Credentials can be provided in several ways:

* Using Google Container Registry
  + Per-cluster
  + automatically configured on Google Compute Engine or Google Kubernetes Engine
  + all pods can read the project’s private registry
* Using AWS EC2 Container Registry (ECR)
  + use IAM roles and policies to control access to ECR repositories
  + automatically refreshes ECR login credentials
* Using Azure Container Registry (ACR)
* Configuring Nodes to Authenticate to a Private Registry
  + all pods can read any configured private registries
  + requires node configuration by cluster administrator
* Pre-pulling Images
  + all pods can use any images cached on a node
  + requires root access to all nodes to setup
* Specifying ImagePullSecrets on a Pod
  + only pods which provide own keys can access the private registry.

[**41.Can you brief about Taints and Tolerations**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-641)

This is an interesting feature of Kubernetes. This feature help users to mark a node (its like tainting the node) so that no pods can be scheduled to it, unless a pod explicitly tolerates the taint. With this feature, one can create nodes that are reserved for specific pods.

[**42.Explain the role of Secrets in Kubernetes**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-642)

A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. Such information might otherwise be put in a Pod specification or in an image; putting it in a Secret object allows for more control over how it is used, and reduces the risk of accidental exposure.

Objects of type secret are intended to hold sensitive information, such as passwords, OAuth tokens, and ssh keys. Putting this information in a secret is safer and more flexible than putting it verbatim in a pod definition or in a docker image.

Users can create secrets, and the system also creates some secrets.

To use a secret, a pod needs to reference the secret. A secret can be used with a pod in two ways: as files in a volume mounted on one or more of its containers, or used by kubelet when pulling images for the pod.

[**43.How can we priortize pods and preempt them.**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-643)

A PriorityClass is a non-namespaced object that defines a mapping from a priority class name to the integer value of the priority. The name is specified in the name field of the PriorityClass object’s metadata. The value is specified in the required value field. The higher the value, the higher the priority.

A PriorityClass object can have any 32-bit integer value smaller than or equal to 1 billion. Larger numbers are reserved for critical system Pods that should not normally be preempted or evicted. A cluster admin should create one PriorityClass object for each such mapping that they want.

PriorityClass also has two optional fields: globalDefault and description. The globalDefault field indicates that the value of this PriorityClass should be used for Pods without a priorityClassName. Only one PriorityClass with globalDefault set to true can exist in the system. If there is no PriorityClass with globalDefault set, the priority of Pods with no priorityClassName is zero.

[**44.What is Kubectl and what is it used for?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-644)

Kubectl is CLI tool which when installed on Linux or Windows system helps you to change or modify Kubernetes cluster. It is a simple tool with rich set of features. It provides  short commands for operations like listing the cluster nodes, getting services deployment details, scaling app instances and getting deployments info.

[**45.Explain the concept of Storage Classes**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-645)

A StorageClass provides a way for administrators to describe the “classes” of storage they offer. Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators. Kubernetes itself is unopinionated about what classes represent. This concept is sometimes called “profiles” in other storage systems.

Each StorageClass contains the fields provisioner, parameters, and reclaimPolicy, which are used when a PersistentVolume belonging to the class needs to be dynamically provisioned.

[**46.What are the volumes per node limits for each Cloud providers**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-646)

The Kubernetes scheduler has default limits on the number of volumes that can be attached to a node:

|  |  |
| --- | --- |
| **Cloud service** | **Maximum volumes per node** |
| Amazon Elastic Block Store (EBS) | 39 |
| Google Persistent Disk | 16 |
| Microsoft Azure Disk Storage | 16 |

[**47.How many types of Container Hooks are available, and explain each one of them**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-647)

The hooks enable Containers to be aware of events in their management lifecycle and run code implemented in a handler when the corresponding lifecycle hook is executed.

There are two hooks that are exposed to Containers:

**PostStart**

This hook executes immediately after a container is created. However, there is no guarantee that the hook will execute before the container ENTRYPOINT. No parameters are passed to the handler.

**PreStop**

This hook is called immediately before a container is terminated. It is blocking, meaning it is synchronous, so it must complete before the call to delete the container can be sent. No parameters are passed to the handler.

[**48.What is a Resource Quota and Why we need them.**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-648)

A resource quota, defined by a ResourceQuota object, provides constraints that limit aggregate resource consumption per namespace. It can limit the quantity of objects that can be created in a namespace by type, as well as the total amount of compute resources that may be consumed by resources in that project.

Resource quotas work like this:

* Different teams work in different namespaces. Currently this is voluntary, but support for making this mandatory via ACLs is planned.
* The administrator creates one or more ResourceQuotas for each namespace.
* Users create resources (pods, services, etc.) in the namespace, and the quota system tracks usage to ensure it does not exceed hard resource limits defined in a ResourceQuota.
* If creating or updating a resource violates a quota constraint, the request will fail with HTTP status code 403 FORBIDDEN with a message explaining the constraint that would have been violated.
* If quota is enabled in a namespace for compute resources like cpu and memory, users must specify requests or limits for those values; otherwise, the quota system may reject pod creation. Hint: Use the LimitRanger admission controller to force defaults for pods that make no compute resource requirements. See the walkthrough for an example of how to avoid this problem.

[**49.What is a Pod Security Policy and why is it important?**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-649)

n Kubernetes, Pod Security Policies consists of settings & strategies that control the security features a pod has access to. It is a type of cluster-level resource which helps in controlling security aspects of Pod.

Since the pod security policy API (policy/v1beta1/podsecuritypolicy) is enabled independently of the admission controller, for existing clusters it is recommended that policies are added and authorized before enabling the admission controller.

[**50.Explain the Service Catalog along with its Architecture**](https://www.zeolearn.com/interview-questions/kubernetes#collapse-advanced-650)

Service Catalog is an extension API that enables applications running in Kubernetes clusters to easily use external managed software offerings, such as a datastore service offered by a cloud provider.

It provides a way to list, provision, and bind with external Managed Services from Service Brokers without needing detailed knowledge about how those services are created or managed.

Using Service Catalog, a cluster operator can browse the list of managed services offered by a service broker, provision an instance of a managed service, and bind with it to make it available to an application in the Kubernetes cluster.

Service Catalog uses the Open service broker API to communicate with service brokers, acting as an intermediary for the Kubernetes API Server to negotiate the initial provisioning and retrieve the credentials necessary for the application to use a managed service.

It is implemented as an extension API server and a controller, using etcd for storage. It also uses the aggregation layer available in Kubernetes 1.7+ to present its API.

