K21Academy
Learn Cloud from Experts

docker + kubernetes

kubernetes

INTERVIEW QUESTIONS BASED ON KUBERNETES

1. Basic Interview Questions
2. Architecture Based Questions
3. Scenario-Based Questions
4. Technical Questions

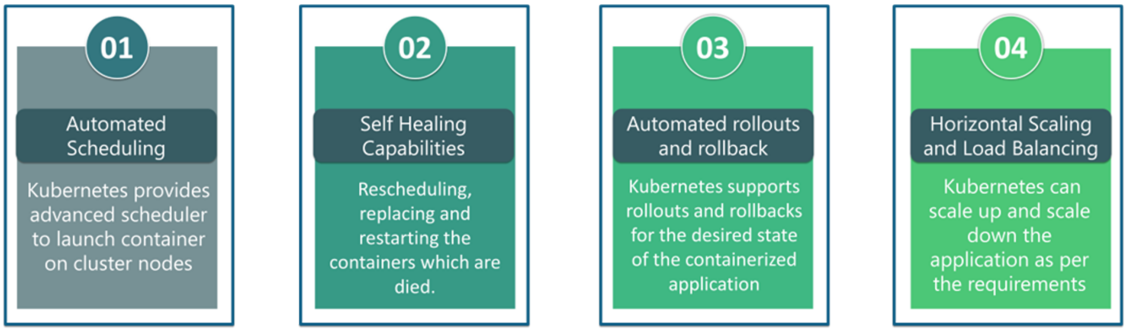
[EDITION 01]

Atul Kumar
Author & Cloud Expert

1. BASIC INTERVIEW QUESTIONS

1. What are the features of Kubernetes?

The features of Kubernetes, are as follows:



2. How is Kubernetes different from Docker Swarm?

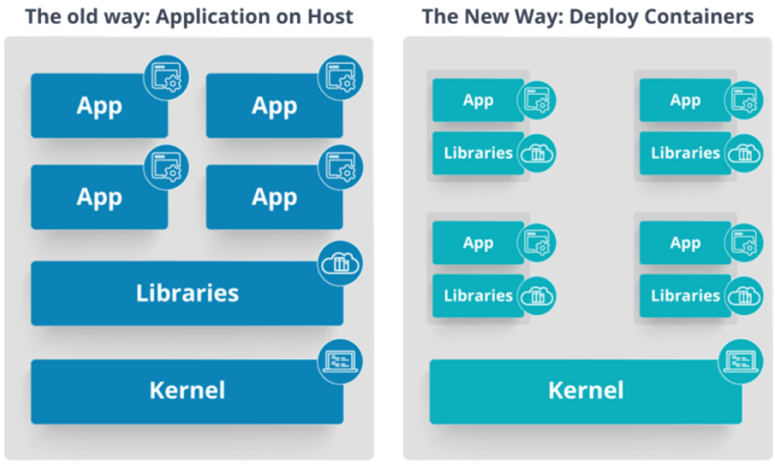
Docker Swarm is Docker's native, open-source container orchestration platform that is used to cluster and schedule Docker containers. Swarm differs from Kubernetes in the following ways:

- *Docker Swarm is more convenient to set up but doesn't have a robust cluster, while Kubernetes is more complicated to set up but the benefit of having the assurance of a robust cluster*
- *Docker Swarm can't do auto-scaling (as can Kubernetes); however, Docker scaling is five times faster than Kubernetes*
- *Docker Swarm doesn't have a GUI; Kubernetes has a GUI in the form of a dash-board*
- *Docker Swarm does automatic load balancing of traffic between containers in a cluster, while Kubernetes requires manual intervention for load balancing such traffic*
- *Docker requires third-party tools like ELK stack for logging and monitoring, while Kubernetes has integrated tools for the same*
- *Docker Swarm can share storage volumes with any container easily, while Kuber-netes can only share storage volumes with containers in the same pod*
- *Docker can deploy rolling updates but can't deploy automatic rollbacks; Kuberne-tes can deploy rolling updates as well as automatic rollbacks*

3. How are Kubernetes & Docker related?

Docker is an open-source platform used to handle software development. Its main benefit is that it packages the settings and dependencies that the software/application needs to run into a container, which allows for portability and several other advantages. Kubernetes allows for the manual linking and orchestration of several containers, running on multiple hosts that have been created using Docker.

4. What is the difference between deploying applications on hosts & containers?



Refer to the above diagram. The left side architecture represents deploying applications on hosts. So, this kind of architecture will have an operating system and then the operating system will have a kernel which will have various libraries installed on the operating system needed for the application. So, in this kind of framework you can have n number of applications and all the applications will share the libraries present in that operating system whereas while deploying applications in containers the architecture is a little different.

This kind of architecture will have a kernel and that is the only thing that's going to be the only thing common between all the applications. So, if there's a particular application which needs Java then that particular application we'll get access to Java and if there's another application which needs Python then only that particular application will have access to Python.

The individual blocks that you can see on the right side of the diagram are basically containerized and these are isolated from other applications. So, the applications have the necessary libraries and binaries isolated from the rest of the system, and cannot be encroached by any other application.

5. What is Container Orchestration?

Consider a scenario where you have 5-6 microservices for an application. Now, these microservices are put in individual containers, but won't be able to communicate without container orchestration. So, as orchestration means the amalgamation of all instruments playing together in harmony in music, similarly container orchestration means all the services in individual containers working together to fulfill the needs of a single server.

6. What is the Google Container Engine?

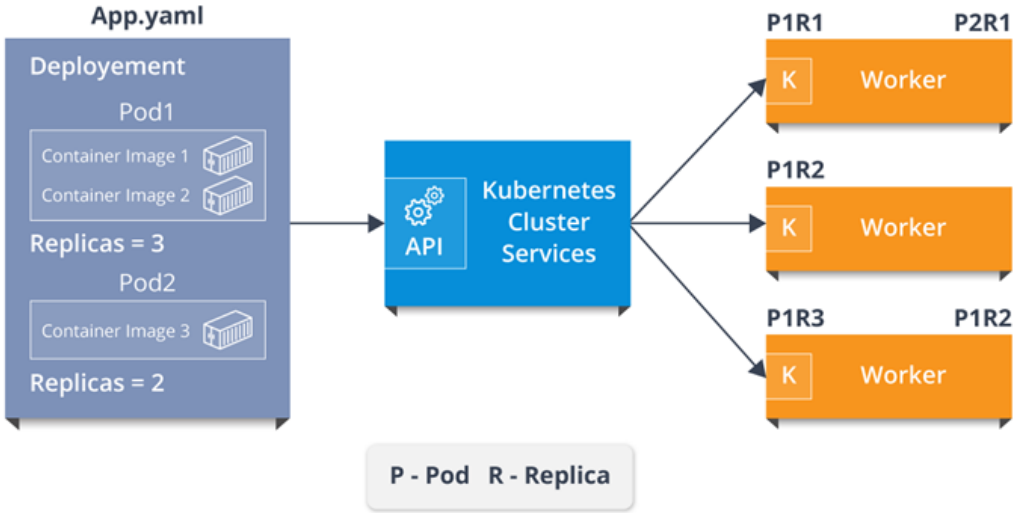
Google Container Engine (GKE) is an open source management platform for Docker containers and the clusters. This Kubernetes based engine supports only those clusters which run within the Google's public cloud services.

7. How does Kubernetes simplify containerized Deployment?

As a typical application would have a cluster of containers running across multiple hosts, all these containers would need to talk to each other. So, to do this you need something big that would load balance, scale & monitor the containers. Since Kubernetes is cloud-agnostic and can run on any public/private providers it must be your choice simplify containerized deployment.

8. What do you know about clusters in Kubernetes?

The fundamental behind Kubernetes is that we can enforce the desired state management, by which I mean that we can feed the cluster services of a specific configuration, and it will be up to the cluster services to go out and run that configuration in the infrastructure.



So, as you can see in the above diagram, the deployment file will have all the configurations required to be fed into the cluster services. Now, the deployment file will be fed to the API and then it will be up to the cluster services to figure out how to schedule these pods in the environment and make sure that the right number of pods are running.
So, the API which sits in front of services, the worker nodes & the Kubelet process that the nodes run, all together make up the Kubernetes Cluster.

9. What is Heapster?

Heapster is a cluster-wide aggregator of data provided by Kubelet running on each node. This container management tool is supported natively on Kubernetes cluster and runs as a pod, just like any other pod in the cluster. So, it basically discovers all nodes in the cluster and queries usage information from the Kubernetes nodes in the cluster, via on-machine Kubernetes agent.

10. What is Kubectl?

Kubectl is the platform using which you can pass commands to the cluster. So, it basically provides the CLI to run commands against the Kubernetes cluster with various ways to create and manage the Kubernetes component.

11. What is Kubelet?

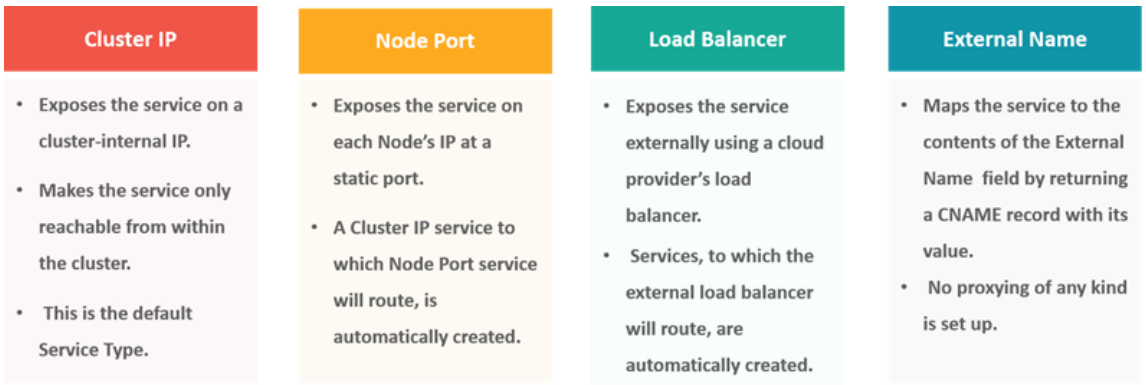
This is an agent service which runs on each node and enables the slave to communicate with the master. So, Kubelet works on the description of containers provided to it in the PodSpec and makes sure that the containers described in the PodSpec are healthy and running.

12. What is etcd?

Etcd is written in Go programming language and is a distributed key-value store used for coordinating between distributed work. So, Etcd stores the configuration data of the Kubernetes cluster, representing the state of the cluster at any given point in time.

13. What are the different services within Kubernetes?

The following are the different types of services used:



14. What is the LoadBalancer in Kubernetes?

A load balancer is one of the most common and standard ways of exposing service. There are two types of load balancer used based on the working environment i.e. either the Internal Load Balancer or the External Load Balancer. The Internal Load Balancer automatically balances load and allocates the pods with the required configuration whereas the External Load Balancer directs the traffic from the external load to the backend pods.

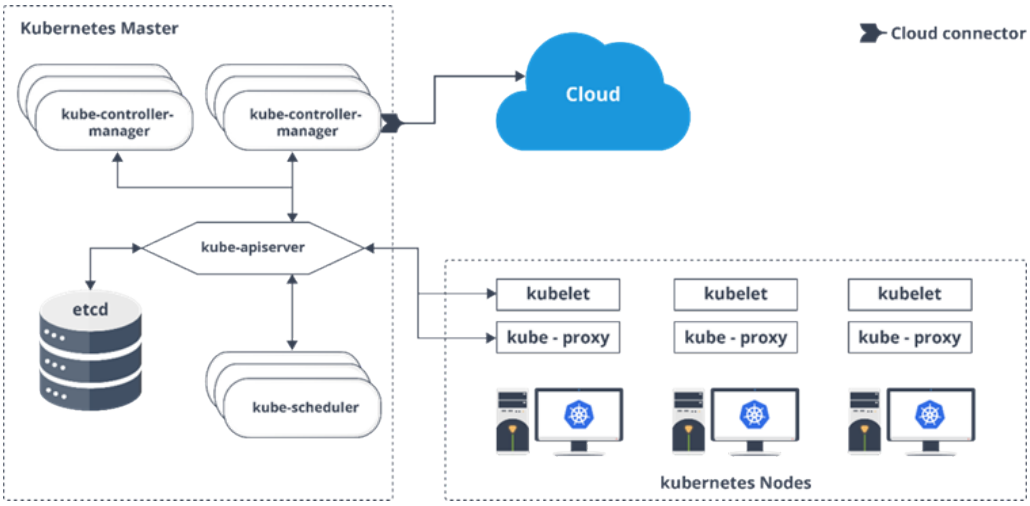
15. What is a headless service?

A headless service is used to interface with service discovery mechanisms without being tied to a ClusterIP, therefore allowing you to directly reach pods without having to access them through a proxy. It is useful when neither load balancing nor a single Service IP is required.

2. ARCHITECTURE BASED QUESTIONS

1. What are the different components of Kubernetes Architecture?

The Kubernetes Architecture has mainly 2 components – the master node and the worker node. As you can see in the below diagram, the master and the worker nodes have many inbuilt components within them. The master node has the kube-controller-manager, kube-apiserver, kube-scheduler, etcd. Whereas the worker node has kubelet and kube-proxy running on each node



2. What is Kube-proxy?

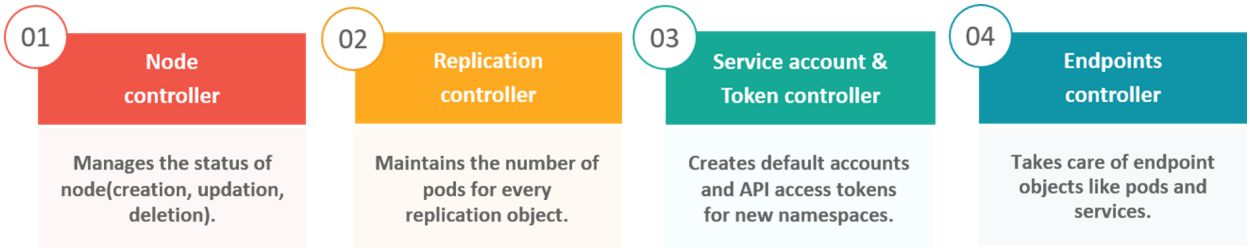
Kube-proxy is an implementation of a load balancer and network proxy used to support service abstraction with other networking operation. Kube-proxy is responsible for directing traffic to the right container based on IP and the port number of incoming requests.

3. What is the role of kube-apiserver and kube-scheduler?

The kube – apiserver follows the scale-out architecture and, is the front-end of the master node control panel. This exposes all the APIs of the Kubernetes Master node components and is responsible for establishing communication between Kubernetes Node and the Kubernetes master components.
The kube-scheduler is responsible for distribution and management of workload on the worker nodes. So, it selects the most suitable node to run the unscheduled pod based on resource requirement and keeps a track of resource utilization. It makes sure that the workload is not scheduled on nodes which are already full.

4. Can you brief about the Kubernetes controller manager?

Multiple controller processes run on the master node but are compiled together to run as a single process which is the Kubernetes Controller Manager. So, Controller Manager is a daemon that embeds controllers and does namespace creation and garbage collection. It owns the responsibility and communicates with the API server to manage the end-points.
So, the different types of controller manager running on the master node are :



5. What do you understand by load balancer in Kubernetes?

A load balancer is one of the most common and standard ways of exposing service. There are two types of load balancer used based on the working environment i.e. either the Internal Load Balancer or the External Load Balancer. The Internal Load Balancer automatically balances load and allocates the pods with the required configuration whereas the External Load Balancer directs the traffic from the external load to the backend pods.

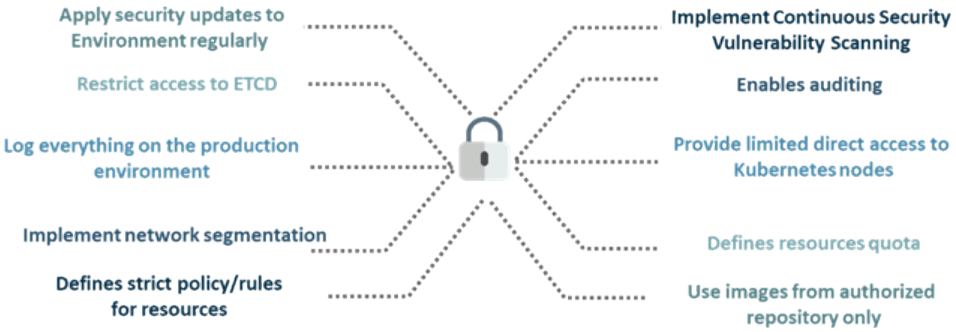
6. What is the difference between a replica set and replication controller?

Replica Set and Replication Controller do almost the same thing. Both of them ensure that a specified number of pod replicas are running at any given time. The difference comes with the usage of selectors to replicate pods. Replica Set use Set-Based selectors while replication controllers use Equity-Based selectors.

Equity-Based Selectors: This type of selector allows filtering by label key and values. So, in layman terms, the equity-based selector will only look for the pods which will have the exact same phrase as that of the label. Example: Suppose your label key says app=nginx, then, with this selector, you can only look for those pods with label app equal to nginx.
Selector-Based Selectors: This type of selector allows filtering keys according to a set of values. So, in other words, the selector based selector will look for pods whose label has been mentioned in the set.

7. What are the best security measures that you can take while using Kubernetes?

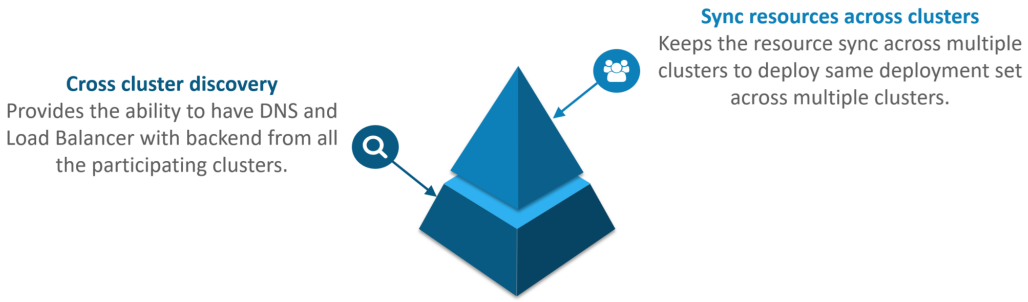
The following are the best security measures that you can follow while using Kubernetes:



8. What are federated clusters?

Multiple Kubernetes clusters can be managed as a single cluster with the help of federated clusters. So, you can create multiple Kubernetes clusters within a data center/cloud and use federation to control/manage them all at one place.

The federated clusters can achieve this by doing the following two things. Refer to the below diagram.



9. What is Container resource monitoring?

As for users, it is really important to understand the performance of the application and resource utilization at all the different abstraction layer, Kubernetes factored the management of the cluster by creating abstraction at different levels like container, pods, services and whole cluster. Now, each level can be monitored and this is nothing but Container resource monitoring.

The various container resource monitoring tools are as follows:

Heapster
Gathers data and events from the containers and pods within the cluster.

InfluxDB
Used along with Heapster for visualizing data within the Kubernetes environment.

Grafana
A time-series database to store the data captured by all Heapster pods.

CAdvisor
A built-in tool in a kubelet that automatically discovers all the active containers and monitors them.

Prometheus
A project of CNCF which provides a powerful querying, alerting and visualization capabilities.

3. SCENARIO-BASED QUESTIONS

1. **Suppose a company built on monolithic architecture handles numerous products. Now, as the company expands in today's scaling industry, their monolithic architecture started causing problems.**

   How do you think the company shifted from monolithic to microservices and deploy their services containers?

   As the company's goal is to shift from their monolithic application to microservices, they can end up building piece by piece, in parallel and just switch configurations in the background. Then they can put each of these built-in microservices on the Kubernetes platform. So, they can start by migrating their services once or twice and monitor them to make sure everything is running stable. Once they feel everything is going good, then they can migrate the rest of the application into their Kubernetes cluster.

2. **Consider a multinational company with a very much distributed system, with a large number of data centers, virtual machines, and many employees working on various tasks.**

   How do you think can such a company manage all the tasks in a consistent way with Kubernetes?

   As all of us know that I.T. departments launch thousands of containers, with tasks running across a numerous number of nodes across the world in a distributed system.
   In such a situation the company can use something that offers them agility, scale-out capability, and DevOps practice to the cloud-based applications.
   So, the company can, therefore, use Kubernetes to customize their scheduling architecture and support multiple container formats. This makes it possible for the affinity between container tasks that gives greater efficiency with an extensive support for various container networking solutions and container storage.

3. **Consider a situation, where a company wants to increase its efficiency and the speed of its technical operations by maintaining minimal costs.**

   How do you think the company will try to achieve this?

   The company can implement the DevOps methodology, by building a CI/CD pipeline, but one problem that may occur here is the configurations may take time to go up and running. So, after implementing the CI/CD pipeline the company's next step should be to work in the cloud environment. Once they start working on the cloud environment, they can schedule containers on a cluster and can orchestrate with the help of Kubernetes. This kind of approach will help the company reduce their deployment time, and also get faster across various environments.

4. **Suppose a company wants to revise it's deployment methods and wants to build a platform which is much more scalable and responsive.**

   How do you think this company can achieve this to satisfy their customers?

   In order to give millions of clients the digital experience they would expect, the company needs a platform that is scalable, and responsive, so that they could quickly get data to the client website. Now, to do this the company should move from their private data centers (if they are using any) to any cloud environment such as AWS. Not only this, but they should also implement the microservice architecture so that they can start using Docker containers. Once they have the base framework ready, then they can start using the best orchestration platform available i.e. Kubernetes. This would enable the teams to be autonomous in building applications and delivering them very quickly.

5. **Consider a multinational company with a very much distributed system, looking forward to solving the monolithic code base problem.**

   How do you think the company can solve their problem?

   Well, to solve the problem, they can shift their monolithic code base to a microservice design and then each and every microservices can be considered as a container. So, all these containers can be deployed and orchestrated with the help of Kubernetes.

6. **All of us know that the shift from monolithic to microservices solves the problem from the development side, but increases the problem at the deployment side.**

   How can the company solve the problem on the deployment side?

   The team can experiment with container orchestration platforms, such as Kubernetes and run it in data centers. So, with this, the company can generate a templated application, deploy it within five minutes, and have actual instances containerized in the staging environment at that point. This kind of Kubernetes project will have dozens of microservices running in parallel to improve the production rate as even if a node goes down, then it can be rescheduled immediately without performance impact.

7. **Suppose a company wants to optimize the distribution of its workloads, by adopting new technologies.**

   How can the company achieve this distribution of resources efficiently?

   The solution to this problem is none other than Kubernetes. Kubernetes makes sure that the resources are optimized efficiently, and only those resources are used which are needed by that particular application. So, with the usage of the best container orchestration tool, the company can achieve the distribution of resources efficiently.

8. **Consider a carpooling company wants to increase their number of servers by simultaneously scaling their platform.**

   How do you think will the company deal with the servers and their installation?

   The company can adopt the concept of containerization. Once they deploy all their application into containers, they can use Kubernetes for orchestration and use container monitoring tools like Prometheus to monitor the actions in containers. So, with such usage of containers, giving them better capacity planning in the data center because they will now have fewer constraints due to this abstraction between the services and the hardware they run on.

9. **Consider a scenario where a company wants to provide all the required hand-outs to its customers having various environments.**

   How do you think they can achieve this critical target in a dynamic manner?

   The company can use Docker environments, to put together a cross-sectional team to build a web application using Kubernetes. This kind of framework will help the company achieve the goal of getting the required things into production within the shortest time frame. So, with such a machine running, the company can give the hands-outs to all the customers having various environments.

10. **Suppose a company wants to run various workloads on different cloud infrastructure from bare metal to a public cloud.**

    How will the company achieve this in the presence of different interfaces?

    The company can decompose its infrastructure into microservices and then adopt Kubernetes. This will let the company run various workloads on different cloud infrastructures.

4. TECHNICAL QUESTIONS

4.1 Intermediate Level

1. What is the difference between config map and secret? (Differentiate the answers as with examples)

Config maps ideally stores application configuration in a plain text format whereas Secrets store sensitive data like password in an encrypted format. Both config maps and secrets can be used as volume and mounted inside a pod through a pod definition file.

Config map:

```
kubectl create configmap myconfigmap
 --from-literal=env=dev
```

Secret:

```
echo -n 'admin' > ./username.txt
echo -n 'abcd1234' ./password.txt
kubectl create secret generic mysecret --from-file=./username.txt
--from-file=./password.txt
```

2. If a node is tainted, is there a way to still schedule the pods to that node?

When a node is tainted, the pods don't get scheduled by default, however, if we have to still schedule a pod to a tainted node we can start applying tolerations to the pod spec.

Apply a taint to a node:

```
kubectl taint nodes node1 key=value:NoSchedul
```

Apply toleration to a pod:

```
spec:
tolerations:
- key: "key"
operator: "Equal"
value: "value"
effect: "NoSchedule"
```

3. Can we use many claims out of a persistent volume? Explain?

The mapping between persistentVolume and persistentVolumeClaim is always one to one. Even When you delete the claim, PersistentVolume still remains as we set persistentVolumeReclaimPolicy is set to Retain and It will not be reused by any other claims. Below is the spec to create the Persistent Volume.

```
apiVersion: v1
kind: PersistentVolume
metadata:
name: mypv
spec:
capacity:
storage: 5Gi
volumeMode: Filesystem
accessModes:
- ReadWriteOnce
persistentVolumeReclaimPolicy: Retain
```

4. What kind of object do you create, when your dashboard like application, queries the Kubernetes API to get some data?

You should be creating serviceAccount. A service account creates a token and tokens are stored inside a secret object. By default Kubernetes automatically mounts the default service account. However, we can disable this property by setting automountServiceAccountToken: false in our spec. Also, note each namespace will have a service account

```
apiVersion: v1
kind: ServiceAccount
metadata:
name: my-sa
automountServiceAccountToken: false
```

5. What is the difference between a Pod and a Job? Differentiate the answers as with examples)

A Pod always ensure that a container is running whereas the Job ensures that the pods run to its completion. Job is to do a finite task.

Examples:

```
kubectl run mypod1 --image=nginx --restart=Never
kubectl run mypod2 --image=nginx --restart=onFailure
○ → kubectl get pods
NAME            READY STATUS    RESTARTS AGE
mypod1          1/1 Running   0 59s
○ → kubectl get job
NAME      DESIRED SUCCESSFUL   AGE
mypod1    1 0             19s
```

6. How do you deploy a feature with zero downtime in Kubernetes?

By default Deployment in Kubernetes using RollingUpdate as a strategy. Let's say we have an example that creates a deployment in Kubernetes

```
kubectl run nginx --image=nginx # creates a deployment
○ → kubectl get deploy
NAME    DESIRED  CURRENT UP-TO-DATE   AVAILABLE AGE
nginx   1  1 1            0 7s
```

Now let's assume we are going to update the nginx image

```
kubectl set image deployment nginx nginx=nginx:1.15 # updates the
image
```

Now when we check the replica sets

```
kubectl get replicasets # get replica sets
```

```
NAME                DESIRED CURRENT READY   AGE
nginx-65899c769f    0 0 0       7m
nginx-6c9655f5bb    1 1 1       13s
```

From the above, we can notice that one more replica set was added and then the other replica set was brought down

```
kubectl rollout status deployment nginx
```

 # check the status of a deployment rollout

```
kubectl rollout history deployment nginx
```

 # check the revisions in a deployment

```
o → kubectl rollout history deployment nginx
deployment.extensions/nginx
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

7. How to monitor that a Pod is always running?

We can introduce probes. A liveness probe with a Pod is ideal in this scenario.
A liveness probe always checks if an application in a pod is running, if this check fails the container gets restarted. This is ideal in many scenarios where the container is running but somehow the application inside a container crashes.

```
spec:
containers:
- name: liveness
image: k8s.gcr.io/liveness
args:
```

Now when we check the replica sets

```
- /server
livenessProbe:
      httpGet:
        path: /healthz
```

8. Is there a way to make a pod to automatically come up when the host restarts?

Yes using replication controller but it may reschedule to another host if you have multiple nodes in the cluster
A replication controller is a supervisor for long-running pods. An RC will launch a specified number of pods called replicas and makes sure that they keep running. Replication Controller only supports the simple map-style `label: value` selectors. Also, Replication Controller and ReplicaSet aren't very different. You could think of ReplicaSet as Replication Controller. The only thing that is different today is the selector format. If pods are managed by a replication controller or replication set you can kill the pods and they'll be restarted automatically. The yaml definition is as given below:

apiVersion: v1
kind: ReplicationController
metadata:
name: test
spec:
replicas: 3
selector:
app: test
template:
metadata:
name: test
labels:
app: test
spec:
containers:
name: test
image: image/test
ports:
containerPort: 80

9. What is the difference between replication controllers and replica sets?

The only difference between replication controllers and replica sets is the selectors. Replication controllers don't have selectors in their spec and also note that replication controllers are obsolete now in the latest version of Kubernetes.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
name: frontend
labels:
app: guestbook
tier: frontend
spec:
```

 # modify replicas according to your case

```
 replicas: 3
  selector:
    matchLabels:
      tier: frontend
template:
metadata:
labels:
tier: frontend
spec:
containers:
- name: php-redis
image: gcr.io/google_samples/gb-frontend:v3
```

Reference:
https://Kubernetes.io/docs/concepts/workloads/controllers/replicaset/

10. How do you tie service to a pod or to a set of pods?

By declaring pods with the label(s) and by having a selector in the service which acts as a glue to stick the service to the pods.

kind: Service

```
apiVersion: v1
metadata:
name: my-service
spec:
  selector:
    app: MyApp
ports:
- protocol: TCP
port: 80
```

Let's say if we have a set of Pods that carry a label "app=MyApp" the service will start routing to those pods.

4.2 Advanced Level

1. Let's say a Kubernetes job should finish in 40 seconds, however on a rare occasion it takes 5 minutes, How can I make sure to stop the application if it exceeds more than 40 seconds?

When we create a job spec, we can give --activeDeadlineSeconds flag to the command, this flag relates to the duration of the job, once the job reaches the threshold specified by the flag, the job will be terminated.

```
kind: CronJob
apiVersion: batch/v1beta1
metadata:
  name: mycronjob
spec:
  schedule: "*/1 * * * *"
activeDeadlineSeconds: 200
```

```
  jobTemplate:
    metadata:
      name: google-check-job
    spec:
      template:
        metadata:
          name: mypod
        spec:
          restartPolicy: OnFailure
          containers:
            - name: mycontainer
              image: alpine
              command: ["/bin/sh"]
              args: ["-c", "ping -w 1 google.com"]
```

2. How do you test a manifest without actually executing it?

use --dry-run flag to test the manifest. This is really useful not only to ensure if the yaml syntax is right for a particular Kubernetes object but also to ensure that a spec has required key-value pairs.
kubectl create -f <test.yaml> --dry-run
Let us now look at an example Pod spec that will launch an nginx pod

```
o → cat example_pod.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: my-nginx
  namespace: mynamespace
spec:
  containers:
    - name: my-nginx
      image: nginx
o → kubectl create -f example_pod.yaml --dry-run
pod/my-nginx created (dry run)
```

3. How do you initiate a rollback for an application?

Rollback and rolling updates are a feature of Deployment object in the Kubernetes. We do the Rollback to an earlier Deployment revision if the current state of the Deployment is not stable due to the application code or the configuration. Each rollback updates the revision of the Deployment

```
o → kubectl get deploy
NAME    DESIRED  CURRENT UP-TO-DATE   AVAILABLE AGE
nginx   1   1 1            1 15h
```

```
o → kubectl rollout history deploy nginx
deployment.extensions/nginx
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

```
kubectl undo deploy <deploymentname>
```

```
o → kubectl rollout undo deploy nginx
deployment.extensions/nginx
```

```
o → kubectl rollout history deploy nginx
deployment.extensions/nginx
REVISION  CHANGE-CAUSE
2         <none>
3         <none>
```

We can also check the history of the changes by the below command

```
kubectl rollout history deploy <deploymentname>
```

4. How do you package Kubernetes applications?

Helm is a package manager which allows users to package, configure, and deploy applications and services to the Kubernetes cluster.
helm init # when you execute this command client is going to create a deployment in the cluster and that deployment will install the tiller, the server side of Helm
The packages we install through client are called charts. They are bundles of templatized manifests. All the templating work is done by the Tiller

```
helm search redis # searches for a specific application

helm install stable/redis # installs the application

helm ls # list the applications
```

5. What are init containers?

Generally, in Kubenetes, a pod can have many containers. Init container gets executed before any other containers run in the pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
  annotations:
    pod.beta.Kubernetes.io/init-containers: '[
        {
            "name": "init-myservice",
            "image": "busybox",
            "command": ["sh", "-c", "until nslookup myser-
vice; do echo waiting for myservice; sleep 2; done;"]
        }
    ]'
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo The app is running! &&
sleep 3600']
```

6. What is node affinity and pod affinity?

Node Affinity ensures that pods are hosted on particular nodes.
Pod Affinity ensures two pods to be co-located in a single node.

Node Affinity

```
apiVersion: v1
kind: Pod
metadata:
name: with-node-affinity
spec:
affinity:
nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: Kubernetes.io/e2e-az-name
            operator: In
            values:
            - e2e-az1
```

Pod Affinity

```
apiVersion: v1
kind: Pod
metadata:
name: with-pod-affinity
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: security
            operator: In
            values:
            - S1
```

The pod affinity rule says that the pod can be scheduled to a node only if that node is in the same zone as at least one already-running pod that has a label with key “security” and value “S1”

Reference:
https://Kubernetes.io/docs/concepts/configuration/assign-pod-node/

7. **How do you drain the traffic from a Pod during maintenance?**

When we take the node for maintenance, pods inside the nodes also take a hit. However, we can avoid it by using the below command

```
kubectl drain <nodename>
```

When we run the above command it marks the node unschedulable for newer pods then the existing pods are evicted if the API Server supports eviction else it deletes the pods
Once the node is up and running and you want to add it in rotation we can run the below command

```
kubectl uncordon <nodename>
```

Note: If you prefer not to use kubectl drain (such as to avoid calling to an external command, or to get finer control over the pod eviction process), you can also programmatically cause evictions using the eviction API.

More info:
https://Kubernetes.io/docs/tasks/administer-cluster/safely-drain-node/

8. **I have one POD and inside 2 containers are running one is Nginx and another one is wordpress So, how can access these 2 containers from the Browser with IP address?**

Just do port forward kubectl port-forward [nginx-pod-name] 80:80 kubectl port-forward [wordpress-pod-name] drupal-port:wordpress-port.

To make it permanent, you need to expose those through nodeports whenever you do kubectl port forward it adds a rule to the firewall to allow that traffic across nodes but by default that isn't allowed since flannel or firewall probably blocks it.proxy tries to connect over the network of the apiserver host as you correctly found, port-forward on the other hand is a mechanism that the node kubelet exposes over its own API

9. **If I have multiple containers running inside a pod, and I want to wait for a specific container to start before starting another one.**

One way is Init Containers are for one-shot tasks that start, run, end; all before the next init container or the main container start, but if a client in one container wants to consume some resources exposed by some server provided by another container or If the server ever crashes or is restarted, the client will need to retry connections. So the client can retry always, even if the server isn't up yet. The best way is sidecar pattern_ are where one container is the Main one, and other containers expose metrics or logs or encrypted tunnel or somesuch. In these cases, the other containers can be killed when the Main one is done/crashed/evicted.

10. What is the impact of upgrading kubelet if we leave the pods on the worker node - will it break running pods? why?

Restarting kubelet, which has to happen for an upgrade will cause all the Pods on the node to stop and be started again. It's generally better to drain a node because that way Pods can be gracefully migrated, and things like Disruption Budgets can be honored. The problem is that `kubectl` keeps up with the state of all running pods, so when it goes away the containers don't necessarily die, but as soon as it comes back up, they are all killed so `kubectl` can create a clean slate. As kubelet communicates with the apiserver, so if something happens in between of upgrade process, rescheduling of pods may take place and health checks may fail in between the process. During the restart, the kubelet will stop querying the API, so it won't start/stop containers, and Heapster won't be able to fetch system metrics from cAdvisor. Just make sure it's not down for too long or the node will be removed from the cluster!

Reference

https://www.simplilearn.com/tutorials/kubernetes-tutorial/kubernetes-interview-questions

https://www.edureka.co/blog/interview-questions/kubernetes-interview-questions/

https://www.knowledgehut.com/interview-questions/kubernetes

FREE CLASS

Register for our FREE Class To know about what is the difference between Kubernetes vs Docker and Demo on Deploying & Running web server on Container (Docker), why you should learn Docker and Kubernetes, Job opportunities for Kubernetes administrator in the market, and what to study Including Hands-On labs you must perform to clear Certified Kubernetes Administrator (CKA) so that you can stay ahead in your career and earn a lot more at https://k21academy.com/kubernetes02

K21Academy
Learn Cloud from Experts

"Docker & Kubernetes Interview Question" is important to any administrator or architect, who is planning to give an Interview for the Kubernetes Admin role.

This guide is prepared by
Oracle ACE, Author & Cloud Expert **Atul Kumar** from K21Academy.

ABOUT AUTHOR

Atul Kumar is an Oracle ACE, Author & Oracle Certified Cloud Architect with 20+ Years experience.

He is helping his customer to learn & become an expert in Docker & Certified Kubernetes Administrator (CKA).

/oracleappsdba /k21academy /k21academy /k21academy /k21academy

contact@k21academy.com