KUBERNETES INTERVIEW QUESTIONS FOR BEGINNER

By QuickTechie.com

Question-1: What do you mean by Monolithic applications?

Answer: Monolithic application are those one which runs in a single process (e.g. in single JVM app) or small number of processes spread across the few servers. At the end of every release cycle, developers package up the whole system and hand it over to the ops team and they deploys and monitors it. And in case of any hardware failure, the ops team migrates it to the remaining healthy servers.

Question-2: What all are the tasks are simplified using the Kubernetes?

Answer: Kubernetes is to simplify the following tasks

- Building
- Deploying and
- Maintaining Distributed applications.

Question-3: What is a Kubernetes engine?

Answer: Kubernetes is an open source orchestrator for deploying containerized applications. It provides the software which are required to successfully build and deploy reliable, scalable distributed systems. You can also sometime refer it as a Kubernetes API.

Question-4: What do you mean by immutable infrastructure changes?

Answer: In Software world you were initially working with the Mutable software. Means you download the new component of a software and install it on existing software and that is known as mutable software infrastructure. However, in case of immutable infrastructure you download new image of a container and replace the existing container with the newly downloaded container. And there would not be any incremental changes.

Question-5: What is the advantage you see for creating a new image rather than upgrading the existing image?

Answer: When you create a new image rather than upgrading the existing image it gives you an advantage that old image still available and you can quickly rollback if it is required. If you upgrade your existing binary then to rollback you have to remove newly installed binary from the existing one.

Question-6: What do you mean by infrastructure as a code?

Answer: In a very simple term I can say that it represent how declarative way you can create any required infrastructure and commit all these declaration files in the source control system like defining a template to create new infrastructure and whenever you want to create new infrastructure specially in the cloud env, checkout the template and use it create the env. You can maintain all version of templates and any time if it is required rollback, use the previous version of the templates and rollback the system.

Question-7: Can you give an example of Kubernetes self-healing behavior?

Answer: Imaging you need 5 containers and one of the developers manually start the sixth container then Kubernetes will check and make sure one container is destroyed. Because it needs only 5

containers. Similarly, if one container is destroyed then Kubernetes would make sure, it starts a new container.

Question-8: What are the common abstractions you find in Kubernetes?

Answer: Following are the common abstractions we can found in Kubernetes.

- **PODs**: This is also known as group of containers. And a single team can work on group of containers (pods) and make it single deployable unit.
- **Isolation**: Kubernetes Services provides load balancing, naming and discovery features and we can isolate one microservice to another microservice.
- **Namespaces**: Using namespaces we can have isolation and access control. Hence, each microservice can control the interaction between services.
- **Ingress**: With the Ingress we can have single frontend which can combine multiple microservices into a single centralized API layer.

Question-9: What is KaaS?

Answer: Almost every public cloud provider (Azure, AWS, Google cloud etc.) are providing managed Kubernetes-as-a-service solution. However, cloud provider operator has their own limitation for using this Kubernetes clusters. Like any new feature of Kubernetes may not be available immediately on KaaS, as they have to maintain cluster for multiple clients and it can create issue for having new features as soon as it is available, they want to test before providing on KaaS.

Question-10: If I have to make Kubernetes application fully portable across cloud provider, what should I take care?

Answer: You have to make sure your Kubernetes applications/containers are not depending on any of the cloud specific services like in DB AWS DynamoDB. Keep it completely independent from cloud provider specific services.

Question-11: Is Kubernetes for distributed applications?

Answer: Yes, Kubernetes is a platform for creating, deploying, and managing distributed applications. However, it also solves many other problems like Shared Library Dependencies. You no more need to have shared library which your application requires different version and other application on same machine require another version. You can ship the library version which you want with your application as part of container and no need to share with other application on same machine.

Question-12: Which is the container runtime env. you know are available?

Answer: Docker and Podman I know as of now.

Question-13: What do you mean by registry?

Answer: Registry is the place where once your image is built or image building file e.g Dockerfile, you can keep it and you can share with others as well. And whenever and whoever required these images can pull it back and use it. And also helps you to manage and deploy private-images as well. And you can also use image-builder services to provide integration with continuous delivery system.

Question-14: What is the container image?

Answer: Container image is a bundle which include your program, and its dependencies into a single artifact under root file system. This also include some metadata which is used by a container runtime to start a running application instance based on the contents of the container image.

Question-15: Which is the most popular container image format?

Answer: As of this writing Docker image was most popular but RedHat 8.0 onwards they want to reduce the Docker monopoly and stopped supporting Docker and supporting PodMan as a new image format.

Question-16: Can you give some more detail about container image?

Answer: A container image is a binary package that encapsulates all of the files necessary to run a program inside of an OS container. You can create a new container image or download one from the container registry. Once you have the container image you can run it to produce a running application inside OS container.

Question-17: Which is the command used to create Docker image?

Answer: You can use "docker" command to create Docker image. This command can help you in packaging, distributing, and running containers.

Question-18: What do you mean by Docker image can be created using Layers?

Answer: Docker image is made up of a series of filesystem layers. Each layer adds, removes, or modifies files from the preceding layer in the filesystem.

Question-19: Why it is said that container image is a manifest file?

Answer: If you see Docker image it is not a single file but rather a specification for a manifest file which points to the other files. And both manifest and associated files are considered as a single unit. And this particular Docker image is created by a series of filesystem layers, and each layer inherits and modifies the layers before this.

Question-20: Can you give simple example, why do you say containers are layered filesystem?

Answer: Lets Assume you have a Base Container classed ContainerA, which just have base operating system like CentOS. Now you have two applications one runs on Java7 and Other on Java8 runtime env. You would fork the ContainerA and create two separate container one for Java7 and another for Java8 support. Like ContainerB built upon ContainerA by installing Java7 and ContainerC is build upon ContainerA by installing Java8 on it. Now you need to install Oracle12C as well which require Java8. Hence, you would fork ContainerC and create new ContainerD on which you can install Oracle 12C.

ContainerA (Base OS CentOS)

- ContainerB (Java7)
- ContainerC(Java8)
    - ContainerD(Oracle12C)

Hence, we can say each container image is built upon another container. In real-world you would see more complex layers.

Question-21: What is the use of container configuration files?

Answer: Container configuration files are always available with the container and they provides instructions on how to setup the container environment and execute the application entry point. Which has information like

- How to setup Networking
- How to setup namespace isolation
- How to setup resource constraints like (cgroups)
- What should be the syscall restrictions should be placed on a running container instance.

Both Container root filesystem and configuration files are typically bundled using the Docker image format.

Question-22: What are the main categories of the containers?

Answer: Containers fall into two categories as below

- System Containers
- Application Containers

Question-23: What do you mean by System Containers?

Answer: You can assume System Containers are somewhat similar to virtual machines and often run a full boot process. And often includes the system services like ssh, cron, syslog etc. However, it is avoided to create system containers now a days.

Question-24: What is application Containers?

Answer: Application containers are different from System containers and they usually run single programs. And provides the granularity for creating scalable applications.

Question-25: Now can you further add detail, what is Kubernetes?

Answer: Yes, now we can say that Kubernetes are more of building and deploying distributed system which is made up using application containers.

Question-26: What is the use of Dickerfile?

Answer: A Dockerfile can be used to automate the creation or building of a Docker container image.

Question-27: What is .dockerignore file?

Answer: This .dockerigonre defines the set of files that should be ignored when copying files into the image.

Question-28: Which are the two main files when you create a Docker Image?

Answer: There are following two files which is required when you what to create Docker Container

- Dockerfile: You can say this a recipe to create a Docker Container.
- .dockerignore: This would define what all files to be ignored while copying file to container.

Question-29: As you know, every Docker image is built on another Docker image, which attriubute in Dockerfile you use for base container?

Answer: "FROM" attribute. Example as below

FROM node:11

Here, it is specifying that get the preconfigured container from Docker Hub which has Node.js 11 already installed.

Question-30: Can you please explain the few main attributes of Dockerfile?

Answer: Below are the main attributes in the Dockerfile, which you come across many times

- **WORKDIR**: You can specify the directory inside the container image, where all the command would run.
- **COPY**: Copy the files.
- **RUN**: Run the command in the container.
- **CMD**: This specify the default command when you start the container.
- **FROM**: What should be the base image for a container.

Question-31: Can you please tell which command you can use to build new Docker container?

Answer: You can use command "docker build" . For example

docker run -t hadoopexam-image

It would create a new Docker image named "hadoopexam-image"

Question-32: What is the command you can use to run Docket image?

Answer: You can use "docker run" command to run the Docker Image.

Question-33: What happens when you remove files from existing container?

Answer: When you remove files from the container then they remain in the image. But your new layer would not have this and you cannon access this. If you are experimenting then it can create large image. For example

LayerA: HE1File.txt

LayerB: Remove HE1File.txt

LayerC: Add new file HE2File.txt

You think that HE1File.txt is no longer present on your Docker Image. But that is not true. And when you run the image this file is not accessible. However, the file added at LayerA is still there. And whenever

you download this image this file always come with Image and this is not good as you don't want this file but unnecessarily taking more space and consuming more network bandwidth whenever downloaded.

Question-34: Can you explain why it is said that you should order layers for image which are least likely to change?

Answer: to understand let's take two example of the container layer

In Example-1 we have three layers as below

- LayerA:BaseOS
- LayerB:add file HadoopExam.java
- LayerC: Install java11

In Example-2 we have three layers as below

- LayerA:BaseOS
- LayerB: Install java11
- LayerC:add file HadoopExam.java

Now, you want to update the HadoopExam.java file. In first case you need both layer HadoopExam.java and Java11 needs to be pulled and pushed, because Java11 layer is depend on the HadoopExam.java layer. If you see Example-2 in this case you have to change only HadoopExam.java layer and need to pull and push. That's why it is important you need to layer your image which is least likely to change. This is a simple example but what if we have many layers in image.

Question-35: Is it ok if containers lower layers are having password in it?

Answer: Not at all, you should never have password stored in any layer of the container. This is not a good practice. This can be easily cracked with the tools.

Question-36: What do you mean by Docker Multistage builds?

Answer: Using multistage builds, your Dockerfile can produce multiple images and you can consider each image as a stage. And artifacts from preceding image can be copied to the current image.

Question-37: What is the difference between public and private repository?

Answer: With public repository your submitted Docker image can be downloaded by anyone. If you are using private repository then it requires authentication and authorization for somebody to download the image created by you.

Question-38: If you want your image should be publicly available, does not require authentication to add in registry?

Answer: Even, you want your Docker Image should be available to public. It is requiring that you yourself authenticate before submitting the image to public registry. And for that you can use "docker login" command. Docker Hub is the place where you want to start first.

Question-39: Which command you can use to publish your image in registry?

Answer: You can use "docker push" command.

Question-40: What does it mean, you have to use container specific API to set up container runtime?

Answer: You can use API from the Kubernetes to describe the way application should be deployed. However, this relies on the container runtime to set-up an application container and that can be done using container specific API which is native to target OS. For Linux it mean you have to configure cgroups and namespaces.

Question-41: What is CRI (Container Runtime Interface)?

Answer: You want interface to the container using API and that is defined by the Container Runtime Interface standard. And this CRI API can be implemented by a number of different programs for example

- **By Docker:** containerd-cri
- **By RedHat:** cri-o

Question-42: In Kubernetes how each container would be launched?

Answer: In Kubernetes usually containers are launched by a daemon on each node called the Kubelet. However, you can use Docker Command Line utility as well to deploy the containers. For example below command

```
docker run -d --name HE –publish 8080:8080 registryURL/HEImage
```

In above command, it starts an HEImage container which maps the ports 8080 of your local machine to 8080 in container. Here, -d option specifics that run this in background as daemon process. Here, --name is giving a name to your container (e.g. HE) which you are getting from registry.

Question-43: What port forwarding is required where container is started?

Answer: When you start a container a port forwarding is required because each container gets its own IP, so listening on localhost inside the container does not cause you to listen on your machine. And without port forwarding, connections will not be accessible to your machine.

Question-44: How you can limit the resources use by Docker container application?

Answer: You can limit the resources used by a Docker Container application by exposing the underlying cgroup technology provided by the Linux Kernel. And same is used by the Kubernetes to limit the resources used by each pod.

Question-45: How can you specify that container application is limited to 512MB memory and 2GB as swap space?

Answer: For that you can use --memory and --memory-swap flags when you are using “docker run” command as below

```
docker run -d --name HE \
--publish 8080:8080 \
--memory 512m \
--memory-swap 2G \
registryURL\HEImage
```

If your program is trying to use much more memory then it can be terminated. Similarly the way JVM crashes on native OS, when it tries to occupy more memory than allocated.

Question-46: How can you delete the Docker Image?

Answer: You can delete the docker image by using command “docker rmi <imageId or image tag>”

Question-47: Why it is mandatory to explicitly delete the image if not needed?

Answer: Please note that unless you explicitly delete an image it will live on your machine forever, even you build a new image with the same name as previous. Because building new image simply moves the tag to the new image and it does not delete or replace the old image.

Question-48: How can you check all the available images on your machine?

Answer: You can use command “docker images”. Which list all the images on your machine and you have to delete the image which you don’t want and save the space on your machine.

Question-49: What is the use of “system prune” command?

Answer: Using “docker system prune” command you can do general cleanup. This will remove all the stopped containers, all untagged images and all unused image layers cached as part of the build process. However, you should use this command very carefully. Because you may not be able to recover the images which are deleted.

Question-50: What is the command to garbage collect Docker Images?

Answer: You can use “docker-gc” tool to garbage collect the Docker image. Usually, professionals schedule this tool using cron to regularly cleanup or garbage collect images which are not in use.

Question-51: What is the benefits of having applications in the container?

Answer: Application container provide a clean abstraction for applications, and when packaged in the Docker Image format they can be easily build, deploy and distributed. And being in a container they also provide the isolation between applications running on the same hardware and helps in avoiding dependency conflicts.

Question-52: What is Kubernetes?

Answer: Kubernetes is a service using this you can create a distributed system with the container. And you can create a Kubernetes cluster on the bare metal servers or you can use the cloud provided Kubernetes service.

Question-53: What is the minikube?

Answer: Using minikube tool you can create local Kubernetes cluster which can run un a VM on your local laptop or desktop. However, minikube helps you to create a single node cluster as a starting point you can use it, but not that much powerful.

Question-54: What is “Docker-in-Docker” tool?

Answer: This is a new tool which can help you create a multi-node cluster on a single machine.

Question-55: Which service is provided by Google Cloud for Kubernetes?

Answer: Google Cloud provides as hosted service named as “KaaS”, Kubernetes as a service and known as GKE (Google Kubernetes Engine)

Question-56: What is the name of the Azure provided Kubernetes solution?

Answer: Microsoft Azur provides Azure Container Service.

Question-57: Which service is provided by the Amazon for Kubernetes cluster?

Answer: Amazon web service provides this using “Elastic Kubernetes Service”.

Question-58: What is the Docker Desktop?

Answer: Docker Desktop you can install on your Laptop or Desktop which comes bundled with single-machine installation of Kubernetes. However, it is not a real multi-node cluster rather a single node cluster for education purpose.

Question-59: Is there any requirement for installing minikube on local machine?

Answer: If you want to install minikube then it needs the Hypervisor on your machine. For Linux and MacOS you can use VirtualBox or VMware and for Windows Hyper-V. And you have to install it before installing minikube.

Question-60: What is a KIND project?

Answer: Kind stands for Kubernetes in Docker. (Usually, you use Kubernetes to manage container). In this project it uses the Docker Containers to simulate multiple Kubernetes nodes. And this is mostly used for building Kubernetes for fast and easy testing.

Question-61: Which is a popular Kubernetes client?

Answer: “kubectl” is an official Kubernetes client. This a command line tool for interacting with the Kubernetes API. Using this you can manage Kubernetes objects like Pods, ReplicaSets and Services. Managing, checking and monitoring the Kubernetes cluster health.

Question-62: How can you check version of Kubernetes client as well as Kubernetes API server?

Answer: You can use “kubectl version” command which would tell you the version for the this client tool itself as well as for Kubernetes API server.

Question-63: How can you get the overall cluster health?

Answer: For that you can use “kubectl get componentstatuses” command.

Question-64: What is the use of Controller Manager?

Answer: “controller-manager” component is responsible for running various controllers that regulate the behavior in the cluster. For instance, it make sure that all of the replicas of a service are available and healthy.

Question-65: Which all are the components which can make the Kubernetes cluster?

Answer: Following are the components for the Kubernetes cluster?

- controller-manager
- scheduler
- etcd server

Question-66: What is the use of scheduler and etcd server in Kubernetes cluster?

Answer: Scheduler is responsible for placing different Pods onto different nodes in the cluster. And etcd server is the storage for the cluster where all of the API objects are stored.

Question-67: What are the master and worker nodes for the Kubernetes cluster?

Answer: Kubernetes has two types of nodes as below

- master
- worker

Master node contain the containers like API server, scheduler etc, And this can be used to manage the cluser while worker nodes where your containers would run. And Kubernetes avoid scheduling work on the master nodes to ensure that application workloads does not impact the overall operation of the cluster.

Question-68: Can you please let me know, few of the Kubernetes components?

Answer: Following are the Kubernetes Cluster components and all are deployed by the Kubernetes itself. And all these components run in the kube-system namespace.

- Kubernetes Proxy
- Kubernetes DNS
- Kubernetes UI

You can think namespace as folder in a filesystem.

Question-69: What is the use of Kubernetes Proxy?

Answer: The Kubernetes proxy would route network traffic to load-balanced services in the Kubernetes cluster. And for this proxy must be present on each node of the cluster. And mostly this is done using API object named DaemonSet. Hence, it is required kube-proxy is running on every node in the cluster.

Question-70: What is the Kubernetes DNS and what is the use?

Answer: Every Kubernetes cluster runs the DNS server, which provides naming discovery for the services that are defined in the distributed cluster. And this DNS server runs as a replicated service on the cluster. And based on the size of the cluster number of DNS server would run.

Question-71: What is the use of Kubernetes UI?

Answer: You can use Kubernetes UI to explore the Kubernetes Distributed cluster as web UI you can also create new containers from UI. You may see some provider does not provide such UI. Hence, you have to be comfortable with command line utility as well.

Question-72: Which is the command line utility available to interact with Kubernetes API?

Answer: "kubectl" is the utility which can be used to interact with the Kubernetes API. You can use this to create Kubernetes objects as well.

Question-73: What is "Namespaces" in Kubernetes?

Answer: Using namespaces Kubernetes can organize the objects in the Cluster. Imagine each individual namespace as Unix Directory or Windows folder. "default" namespace is a default namespace with "kubectl" interacts.

Question-74: How can you interact with different namespace using "kubectl"?

Answer: To interact with different namespace like "he_default" you can use "kubectl --namespace=he_default". And similarly, if you want to interact with all the namespaces then you can use "kubectl --all-namespace" flag.

Question-75: What is the "context"?

Answer: You can use "context" to manage the clusters. And some other activities like

Adding new namespace as default namespace. You can use below command to do that

```
kubectl config set-context he-context --namespace=he_default
```

In above command it is creating new context named "he-context" and also set default namespace in it. However, to start using this newly created context you have to run below command

```
kubectl config use-context he-context
```

You can also use context to authenticate users to cluster.

Question-76: How are resources referred in Kubernetes?

Answer: Everything in Kubernetes is referred by a RESTful resource and you can say these are Kubernetes objects. And each Kubernetes object exists on unique http path.

Question-77: How kubectl command line utility interacts with Kubernetes objects?

Answer: Each kubectl command interact with RESTful resource and makes an http request to the RESTful URLs and interact with Kubernetes objects.

Question-78: Can you please explain the "kubectl" get command?

Answer: get is one the most commonly used command in kubectl. If you re a command like

"kubectl get resource-name"

It will return you the listing of all resources in the current namespace. And if we need to access any specific object then we have to use command as

"kubectl get resource-name object-name"

Question-79: Which query kubectl uses?

Answer: If you want to extract specific path from object than you can use JSONPath query language. And this will help you to select specific fields from an object. See example below

"kubectl get pods he-pod -o jsonpath ==template={.status.podIP}"

In above example it would extract the IP address for "he=pod" and print it.

Question-80: How are the objects represent by Kubernetes API?

Answer: Kubernetes API represent all objects in either JSON or YAML files. These files are either returned by the server in response to query or posted to the server as part of an API request. And using YAML or JSOM you can either create, update, or delete the Kubernetes objects on the cluster.

Question-81: How can you create an object specified in "he_Object.yaml"?

Answer: You can use apply command as below

"kubectl apply -f he_object.yaml"

And while specifying, you don't have to provide the resource type of the objects. This can be obtained from object itself. And same command you can use to update the object. If object already exists in the cluster it will simply exit successfully without making any changes.

Question-82: How can you check before apply command, what it can do if you run it?

Answer: In that case you have to use "--dry-run" option while running apply command.

Question-83: How does "kubectl edit" command is different?

Answer: When you use "edit" command, it will download the latest object state and then launch an editor that contains the definition. And once you save the file it will automatically uploaded back to the Kubernetes cluster.

Question-84: Which are the commands which are helpful for checking history of apply command?

Answer: You can use following options to check the entire history of the apply command

- edit-last-applied
- set-last-applied
- view-last-applied

For example below command would show you the last applied command to the object

"kubectl apply -f heobj.yaml view-last-applied"

Question-85: How can you delete the object in Kubernetes?

Answer: If you want to delete the object in Kubernetes cluster you can use delete command as below

"kubectl delete resource-name obj-name"

Question-86: How do you define the tags for Kubernetes objects?

Answer: You can define the tags using Labels and annotation to a Kubernetes object. You can use below command for applying/updating tag

"kubectl label pods hepod author=hadoopexam"

In above command it would add the label "author=hadoopexam" to a pod which has name as "hepod"

Question-87: How can you run the bash command in a running container?

Answer: You can use exec command to execute command in a running container as below

"kubectl exec -it hepod –bash

Above command would give you an interactive shell inside the running container, which would help you to interact with container.

Question-88: How can you send input to running process from standard input?

Answer: You can use "attach" option. Using this you can send input to the running process and if your running process can read data from standard input then it will read this.

Question-89: What you have to do, when you want to access your pod via network?

Answer: Then you have to use port-forward command and this will forward network traffic from the local machine to the Pod. And this is the best way to enable secure traffic which is not exposed to public network. You have to use following command

"kubectl port-forward hePOD 8080:80"

This will open connection and forward traffic from local machine on port 8080 to the remote container 80.

Question-90: What is pod?

Answer: Kubernetes groups multiple containers and create a single unit that is known as Pod. Pod is a collection of application containers and volumes running in the same execution environment. And you should know that pod is a smallest unit which is deployed as a smallest unit on a Kubernetes cluster. And now you can think that all the containers of the same pod are deployed on the same machine.

Question-91: How cgroup and Linux namespace sharing done for a pod and its container?

Answer: Each container within a pod runs in its own cgroup, but they share the Linux Namespaces.

Question-92: How the application running in same pod use the network resources?

Answer: All the application running in the same pod

- Share the same IP address
- Share the port space
- Have same hostname
- Can communicate using IPC (interprocess communication channel)

Question-93: How does application in different pods have network?

Answer: Application running in different pods are

- Isolated from each other.
- Have different IP address
- Have Different hostnames.

Question-94: Can you give an example what you can put inside the pod?

Answer: If all your applications can talk to each other over the network (e.g. Webservice, Messaging Queues etc), then we highly recommend to put them all in different pod. Because then you can scale each individual pod based on the requirement. If you think all your application cannot talk to each other over network and required interprocess communication then you have to put all of them in single POD. However, it means your applications are not well designed.

Question-95: What is declarative configuration?

Answer: You might have seen technology world is moving towards the declarative configurations. It means you write down what all configuration you want in a configuration file and then submit that configuration to a service that takes actions to ensure desired state becomes the actual state. And this kind of declarative configuration you can check in your source code repository.

Question-96: What is Pod Manifest file?

Answer: This Manifest file describe the pod. It is a text file and Kubernetes API object and you can think of it as a declarative configuration. The Kubernetes API server accepts and process Pod manifests before storing them in persistent storage that is etcd.

Question-97: How scheduler works with pods?

Answer: Scheduler looks for the pods which are not yet placed on any host and for that it uses the Kubernetes API. Once it find the Pods places those Pods on the nodes depending on the resource requirement and any constraint define in the Manifest file. So you can think of many Pods can be placed on the same machine, if that machine has enough resources.

Question-98: Is it a good idea to schedule multiple replicas on same machine?

Answer: No, not at all. It should not schedule multiple replicas on the same machine because it is not good for reliability, because any machine can fail any time and if multiple replica's are on the same machine then they all would be down when host machine is down. Hence, Kubernetes scheduler tries to ensure that Pods from same application are distributed onto different machines for reliability.

Question-99: Scheduler automatically keep moving pods from one machine to another machine?

Answer: No, Once Pod are scheduled to a node, it does not move them, automatically. You must have to explicitly destroy them and reschedule it.

Question-100: What is the best way to deploy multiple replicas of a Pod?

Answer: ReplicaSets is a best way to deploy multiple instances of a Pod.

Question-101: Which command you can use to create a new pod?

Answer: You can use “kubectl run” command to create a new Pod.

Question-102: In what all languages you can write Pod Manifest file?

Answer: You can use either YAML or JSON language. However, YAML is more preferred because it is more human readable and better support of writing comments.

Question-103: What would below command do?

“kubectl apply -f he-pod.yaml”

Answer: This will submit manifest “he-pod.yaml” file to Kubernetes API server. And then its Kubernetes server responsibility to schedule that Pod to run on any available health node (machine/host) in the cluster. And once this pod is deployed it would be monitored by the “Kubelet” daemon process.

Question-104: Can you delete a pod using Manifest file?

Answer: Yes, you can use command as below.

“kubectl delete -f he-pod.yaml”

Question-105: What happen when you delete a Pod?

Answer: When you delete a Pod, it would not delete it immediately. But it would go in terminating state. And this should have some grace period, be default it is 30 seconds and in this Terminating state, pod would not accept any new request and try to finish all the request which are already submitted and then terminate it.

Question-106: How does data affected when you delete a Pod?

Answer: When you delete a Pod, it would delete any data which is stored in the containers as well. So, if you want to persist data across multiple instances of a Pod, you have to use PersistentVolumes.

Question-107: What below command do?

“kubectl pot-forward hePod 8080:8080”

Answer: Above command create a secure tunnel from your local machine, through Kubernetes master, to the instance of the Pod running on one of the Worker Nodes. And you can access your Pod using web interface http://localhost:8080 . However, you can access until above command is running.

Question-108: How can you get the logs from previously running container?

Answer: When you use “logs” command it would download the logs from container and show you. If you use -f flag the it will continuously stream the logs. And if you use the --previous flag, it would give you the logs from a previous container.

Question-109: Why you should avoid copying file into a container?

Answer: In Kubernetes world we always want that containers are immutable. But in some cases you may want to copy some configuration file then you can use “cp” option to copy the file in container. But this is not a good practice and should be avoided.

Question-110: How Kubernetes make sure your process is running in Kubernetes?

Answer: When we run your application as a container in Kubernetes and that would be kept alive using a process called “health check” . And this “health check” process make sure that the main process of your application is always running. And if it is found as part of health check that this process is not running then Kubernetes would restart it.

Question-111: What is “liveness” health check?

Answer: As we have “health check” process is used by the Kubernetes itself to check your process is live or not. But it is possible that your process is up but not running properly then in that case you have to use application specific logic. Like website page is available or not. In that case you can use “liveness” health check and that would be used to run application specific checks. This you have to define explicitly in your Pod manifest file. This “Liveness Probe” is defined per container, which means each container inside the Pod is having process running.

Question-112: What are the various parameters you can define for liveness Probe in manifest file?

Answer: Following are the parameters you can use but these are mainly used

- **initialDelaySeconds:** As soon as container deployed, after these many seconds (e.g. 10 seconds) this check would be called.
- **timeoutSeconds:** Probe should return in these many seconds (e.g. 5 seconds) to be considered successful.
- **periodSeconds:** It will try these many seconds (e.g. 10 seconds).
- **failureThresold:** If more than these many numbers (e.g. 3 times) if probe fails, then it would be considered container fail and restart it.