Project: OS based
Topics Covered: Kubernetes, Microservices, Linux, SQL, Python, Robot Framework, Git, Selenium , Test & Defect Tools
L1 Q: 1hr

1. **Briefly introduce yourself**

K8S, Microservices

2. **Explain the Kubernetes architecture?**

Kubernetes has Master (Control Plane) and Worker Nodes(minion).

Control Plane components:
API Server: Entry point for all kubectl operations
Controller Manager: Ensures desired state
Scheduler: Assigns pods to nodes
ETCD: Key-value store for cluster state

Worker node components:
Kubelet: Communicates with control plane
Kube-proxy: Handles networking
Container Runtime: Docker/Containerd
workernode contains pods
(Each pod runs one or more containers, pods scheduled by master onto worker node)

3. **What are Kubernetes namespaces?**

Namespaces logically divide a Kubernetes cluster into multiple virtual clusters.
They help in:
Isolating environments (dev, QA, prod)
Applying role-based access
Organizing microservices
Avoiding naming conflicts
***List all namespaces** kubectl get ns (or) kubectl get namespaces
Create a namespace kubectl create ns <namespace-name>
Delete a namespace kubectl delete ns <namespace-name>
Describe a namespace kubectl describe ns <namespace-name>
List all resources in a namespace kubectl get all -n <namespace-name>
Set a default namespace in current context

```
kubectl config set-context --current --namespace=<namespace-name>
```

Check the current namespace kubectl config view --minify | grep namespace:
Switch namespace (temporary, per command) kubectl get pods -n <namespace>
YAML-based namespace creation

```
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
```

```
kubectl apply -f namespace.yaml
```

4. **How can you list all microservices under a specific namespace?**

```
kubectl get pods -n <namespace>
kubectl get svc -n <namespace>
```

5. **logs-application, console**

Application kubectl logs <pod-name>
Console print() in Python , console.log() in Node.js, System.out.println() in Java

6. **login/connect command**

"There is no kubectl login command.
We access a Kubernetes cluster by using kubeconfig, cloud provider CLIs (AWS/GCP/Azure), or service account tokens. Login is handled outside kubectl, and kubectl simply uses the credentials stored in kubeconfig to talk to the cluster."

Login using Kubeconfig file kubectl --kubeconfig=/path/to/kubeconfig get pods
    export KUBECONFIG=/path/to/kubeconfig

Check current cluster / user (to confirm login)
    kubectl config current-context
    kubectl config get-contexts

Switch between clusters (contexts) kubectl config use-context <context-name>

Login to Kubernetes (Cloud Providers)

AWS EKS aws eks update-kubeconfig --region <region> --name <cluster-name>

Azure AKS
    az login
    az aks get-credentials --resource-group <rg-name> --name <cluster-name>

Google GKE gcloud auth login

Get cluster credentials:
    gcloud container clusters get-credentials <cluster-name> --zone <zone>

If cluster uses Token-based authentication kubectl --token=<token-value> get pods

If cluster uses certificate-based authentication kubectl --client-certificate=client.crt --client-key=client.key get pods

7. **execute command**

    kubectl exec is used to execute commands inside running containers.
    The -i flag keeps input open, and -t allocates a terminal (interactive mode).

Execute a command inside a pod kubectl exec <pod-name> -- <command>
    kubectl exec my-pod -- ls

Open a shell inside a pod kubectl exec -it <pod-name> -- /bin/bash

Execute a command in a specific namespace
    kubectl exec -it <pod-name> -n <namespace> -- /bin/bash

Execute a command inside a specific container (multi-container pod)
    kubectl exec -it <pod-name> -c <container-name> -- /bin/bash

Run a single command without opening a shell
    kubectl exec <pod-name> -- cat /etc/os-release

Copy files from pod to local machine
    kubectl cp <namespace>/<pod-name>:/path/in/pod /local/path

Copy files from local machine to pod
    kubectl cp /local/path <namespace>/<pod-name>:/path/in/pod

8. **How do you delete or restart a pod in Kubernetes?**

restart(by deleting it) kubectl delete pod <pod-name>
delete kubectl delete pod <pod-name> -n <namespace>
Or if deployment-based

kubectl rollout restart deployment <deployment-name> -n <namespace>

9. What are microservices?

Microservices are small, independent, loosely coupled services that work together to build an application. Each service:

* Has its own database (sometimes)
* Has its own deployment
* Can scale independently
* Communicates via REST APIs, message queues, etc.

Linux

10. Have you worked on Linux? If yes, explain your experience.

11. listing

***list all file in dir**

ls
ls -a (hidden)
ls -l (details)
ls -la/al (detailslist+hidden)
ls -lh (human readable sizes)
ls -R (list everything recursively including subfolders)
ls -lt (sorted by time), ls -lS(sorted by size),

12. List latest

***list recursively latest** - ls -ltR
***latest files** - ls -lt | head -n 1
latest top5 - ls -lt | head -n 5

13. Example existing files Abc.txt, ABC.txt, abc.txt, xyz.txt

*__list xyz file__ Exact match (case-sensitive): ls -l xyz.txt
***list all abc files** Case-insensitive search: ls -l | grep -i "^xyz.txt$"
Case-insensitive wildcard: ls -l | grep -i "abc"

14. Rename

***rename file** - mv oldname.txt newname.txt
Rename all abc* files to abc_*.txt for f in abc*; do mv "$f" "new_$f"; done

15. Number of lines

*__no. of lines__ - wc -l filename (asked to try with cat aswell)
Find how many lines contain a word grep -c "error" logfile.log

16. Ps commands

list background process - command &, ps aux
***list all running process - ps -ef, ps(current terminal),**

17. Kill commands

***kill process** kill pid
force kill kill -9 pid

SQL

18. Have you worked with SQL

19. What are joins in SQL? Explain different types

Employee table

Empl_ID	Firstname	LastName
1	ABC	D
2	DEF	G
3	HIJ	K
4	AGG	L

Employee_Details table

Empl_ID	Age	Salary
1	24	123
2	35	456
3	28	350
5	30	650

INNER JOIN Common rows between tables

example

```
SELECT e.Firstname, e.LastName,
d.Age, d.Salary
FROM Employee e
INNER JOIN Employee_Details d
ON e.Empl_ID = d.Empl_ID;
```

output:

```
| Firstname | LastName | Age |
Salary |
| --------- | -------- | --- | ------ |
| ABC      | D       | 24  | 123    |
| DEF      | G       | 35  | 456    |
| HIJ      | K       | 28  | 350    |
```

LEFT JOIN (LEFT OUTER JOIN)

Returns all records from the left table, and matching from right. NULL if no match.

```
SELECT e.Firstname, e.LastName,
d.Age, d.Salary
FROM Employee e
LEFT JOIN Employee_Details d
ON e.Empl_ID = d.Empl_ID;
```

```
| Firstname | LastName | Age  |
Salary |
| --------- | -------- | ---- | ------ |
| ABC      | D       | 24   | 123    |
| DEF      | G       | 35   | 456    |
| HIJ      | K       | 28   | 350    |
| AGG      | L       | NULL | NULL   |
```

RIGHT JOIN (RIGHT OUTER JOIN) Returns all records from the right table, and matching from left.

```
SELECT e.Firstname, e.LastName,
d.Age, d.Salary
FROM Employee e
RIGHT JOIN Employee_Details d
ON e.Empl_ID = d.Empl_ID;
```

```
| Firstname | LastName | Age |
Salary |
| --------- | -------- | --- | ------ |
| ABC      | D       | 24  | 123    |
| DEF      | G       | 35  | 456    |
| HIJ      | K       | 28  | 350    |
| NULL     | NULL    | 30  | 650
```

FULL OUTER JOIN Returns all records from both tables, matching where possible.

```
SELECT e.Firstname, e.LastName,
d.Age, d.Salary
FROM Employee e
FULL OUTER JOIN
Employee_Details d
ON e.Empl_ID = d.Empl_ID;
```

```
| Firstname | LastName | Age  |
Salary |
| --------- | -------- | ---- | ------ |
| ABC      | D       | 24   | 123    |
| DEF      | G       | 35   | 456    |
| HIJ      | K       | 28   | 350    |
| AGG      | L       | NULL | NULL   |
| NULL     | NULL    | 30   | 650    |
```

.**CROSS JOIN** Returns all possible combinations between two tables.

```
SELECT e.Firstname, e.LastName, d.Age, d.Salary
FROM Employee e
CROSS JOIN Employee_Details d;
```

If Employee has 4 rows and Employee_Details has 4 rows → 16 rows in output

20. **Require o/p**

Firstname	LastName	Age	Salary
ABC	D	24	123
DEF	G	35	456
HIJ	K	28	350

```
SELECT e.Firstname, e.LastName,
d.Age, d.Salary
FROM Employee e
INNER JOIN Employee_Details d
ON e.Empl_ID = d.Empl_ID;
```

21. **salry desc or asc**

Firstname	LastName	Age	Salary	Department
ABC	D	24	123	IT
DEF	G	35	456	IT
HIJ	K	28	350	FN

```
SELECT Firstname, LastName, Age,
Salary, Department
FROM Employee e
INNER JOIN Employee_Details d
ON e.Empl_ID = d.Empl_ID
ORDER BY Salary ASC/DESC
```

Python

22. **Explain the concept of data structures in Python?**
23. **What are Python dictionaries, and how are they different from lists and tuples?**

Data structures organize and store data efficiently.
Common Python data structures:
List: ordered, mutable
List: ordered collection, mutable
Tuple: ordered collection, immutable
Dictionary: key-value pairs, unordered, mutable

24. **Have you used the extend() method? Explain with an example**.**?**

```
a = [1, 2, 3]
b = [4, 5]
a.extend(b) (or)
a.extend([4, 5])
print(a)  # [1, 2, 3, 4, 5]
```

25. **Python dictionary example?**

```
my_dict = {"name": "John", "age": 30, "city": "New York"}
my_dict["salary"] = 50000
```

robot framework

26. **How do you use the Robot Framework? Explain workflow.**
27. **What are import libraries in Robot Framework?**

Write test cases in `.robot` files
Import libraries (BuiltIn, Selenium, Requests, custom Python libs)
Run tests using `robot` command
Generate logs and HTML reports

28. **What are test cases in Robot Framework?** HOW DO YOU WRITE AND EXECUTE THEM

A test case includes:
* Keywords
* Steps
* Expected results

Example:

```
*** Test Cases ***
Validate API Response GET
https://api.example.com/users
Should Be Equal As Numbers    ${status}
200
```

29. **Which tool or IDE do you use for Robot Framework?**

GIT

30. **all commands from clone to push**

```
>cd path
>git clone <repourl>
>cd project
>update changes
>git status
> git diff
> git add .
> git  commit -m "msg"
> git pull
resovle
> git push
```

31. **if only selected files to push - git add file1.txt file2.txt**

Selenium

32. **implicit & explicit waits**
33. **What is automation testing? How do you perform it?**

Automation testing uses tools/scripts to run tests automatically.I perform it by:
* Identifying test cases to automate
* Writing framework scripts
* Running tests on CI/CD
* Reporting results

Tools

34. **name test & defect management tools used**

L2:Q
1. Explain your experience in Automation Testing.
2. Have you worked with Microservices? How do you troubleshoot issues?
Check pod logs (kubectl logs)
Inspect API gateway logs
Use Postman for API testing
Check service health using /health endpoints
Validate Kafka/message queue events
Verify database connectivity
Trace calls using distributed tracing tools (Jaeger/Zipkin—if available)
3. Experience with deploying applications to Kubernetes.
My role includes:
Applying YAML files
Checking deployments, services, pods
Debugging failed deployments
Verifying configmaps/secrets
Restarting services during testing
4. What cloud platforms have you worked with?
AWS / GCP / Azure (select based on your experience).
You can mention: EC2, S3, EKS, CloudWatch, IAM, etc.
5. Experience with Linux and shell scripting.
6. Databases you have worked on?
MySQL / PostgreSQL
Writing queries
Creating joins
Validating application data
7. Experience with REST APIs and messaging services.
GET, POST, PUT, DELETE
Status codes
Response time
Authentication (JWT, OAuth, Basic)
Messaging services:
Kafka / RabbitMQ (based on your experience)
Validate consumed/published messages
8. Experience in Agile and tools like JIRA.
9. Experience with performance testing.
If yes → Mention JMeter or Locust.
If no → "I have basic understanding but haven't done full-scale performance testing. I can learn quickly."