Self intro:

Hi this is XXXXX working as DevOps Engineer for XXXX company for last XX Years.
Currently I have been working for XXXX client where our team is managing2 applications.
As a DevOps engineer, I make sure the environment is clean and running. I have worked on various DevOps Tools like Git, GitHub, Jenkins, maven, Ansible, Docker, Terraform and Kubernetes for orchestration, Aws cloud and physical Data Centres.
In AWS I have worked on various services like ec2, S3, route53, application load balancer, VPC, cloud watch, CloudTrail, elastic container service and elastic container registry.

Roles and Responsibilities:

We worked along with development teams for next releases, where our team is responsible for UAT and PROD releases.
We have weekly release and also flex release which will happen for every 2 or 3 months.
If we have any monitoring alerts then i will be working on that alerts. Managing Jenkins pipelines and other infrastructure.
Working on patching activities along with Linux team for every quarter to make sure the kernels are updated, our team is responsible for pre patching and post patching activities.
We provide 24/7 support means Pager Duty if high priority tickets, then will get a call on mobile and will look into that issue.

Prepatching and post patching activities (Answer only if asked):

Patching activities.
Tell that we support patching activities.
We do pre patching and post patching activities.
Pre patching means bringing down the traffic by disabling the health check, brining down all the services running in the servers.
We have ansible playbooks to do this job.
Post patching means brining up all the services running in the servers and enabling the traffic back.

Team (Answer only if asked)

We have a team of 6 members from offshore and we work 24/7.From onshore we have our manager and scrum master.

Managerial Questions:

1)Tell me something which is not mentioned in your resume?

My family is not mentioned in my resume, basically my father is Worker, he has been very hard and supported our family and he is my role model as well. I Have learned many things like love what we do and be dedicated toward your work. My mother is housemaker and I have learnt time management and the way she supported family.
If you have sister/brothers just tell their name and profession.

2)What is INCIDENT and PROBLEM management?

Any issue/error can be marked as incident, group of similar Incidents can be marked as a Problem.

3)Where would you like to see yourself after 2-3 years?

I would like to see myself at a position handling a team/project.

4)What if your project has not done on sprint duration, how will you manage the client?

First thing is based on priority of work, I will try to complete that on time, but if still I was unable to complete then I will explain my manager/client about the process to finish the task, and the benefits of the particular process in improving the quality.
If I got blocked then will try to reach out team and get the enough information to clear the issue.

5)What is client has changed the requirement for the sprint?

If the Client changed the requirement, then I will try to put extra time and finish my work.
and also, when I get a requirement from client then will ask to share the email with requirement, so that there will be no sudden surprises in changing the requirement

Interview Questions:

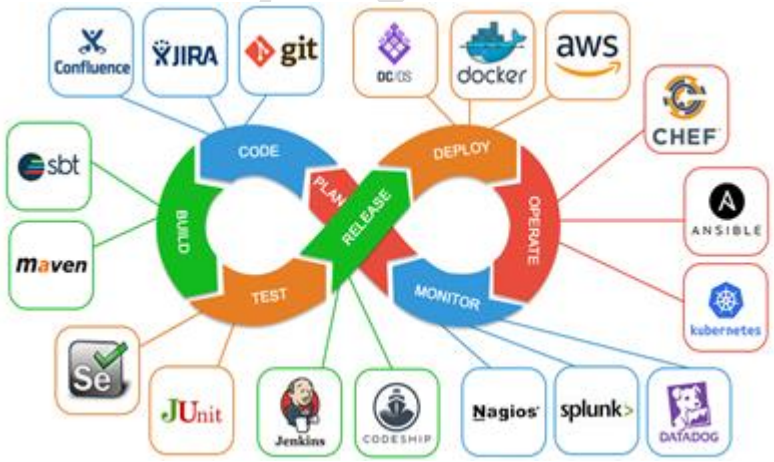
DevOps:

1.What is DevOps?

Devops is a methodology which help to remove the gap between development and operations team. Implementation of Devops will help us to track the bugs at initial stage and also release the application faster.

2.What are the different stages of DevOps?

- Plan
- Code
- Build
- Test
- Release
- Operate
- monitor and Feedback.

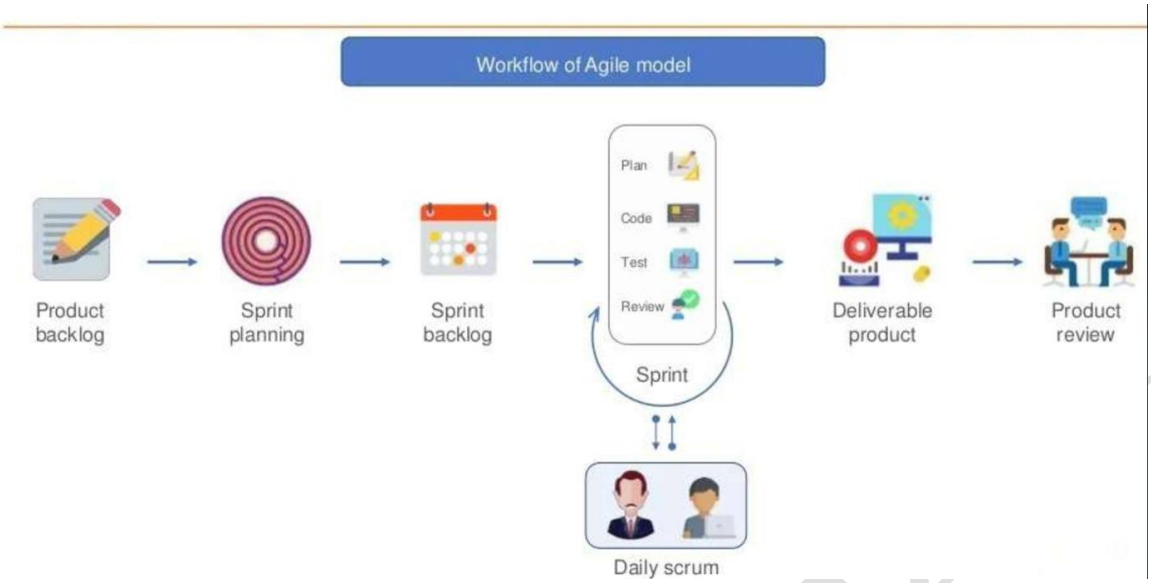


3.What are the methodologies before DevOps?

Waterfall model and agile process.

4.Have you worked on Agile Process?

Yes, where we have scrum master who will be assigning set of tasks for sprint.



5.What is the Sprint duration?

2 weeks.

6.Difference between IAAS, PAAS and SAAS?

SaaS (Software as a service)

platforms involve software that is available via third-party over the Internet.
Examples of popular SaaS providers include:
BigCommerce.
Google Workspace,
Salesforce. Dropbox.
MailChimp. Zendesk.
DocuSign. Slack.
HubSpot.

PaaS (Platform as a service)

focuses primarily on hardware and software tools available over the internet.
Examples of popular PaaS providers include:
AWS Elastic Beanstalk. Heroku.
Windows Azure (mainly used as PaaS). Force.com. Google App Engine.
OpenShift. Apache Stratos.
Adobe Magento Commerce Cloud.

IaaS (Infrastructure as a service)

works primarily with cloud-based and pay-as-you-go services such as storage, networking and virtualization.
Examples of popular IaaS providers include: **AWS EC2. Rackspace.**
Google Compute Engine (GCE). Digital Ocean. Microsoft Azure.
Magento 1 Enterprise Edition.

Linux:

1. Tell 10 Linux commands?

Netstat -na -- To check the running port numbers
Ps -ef -- TO check the running services
find / -name filename -- find a file
grep keyword filename -- Find a keyword from a file
mkdir -- to create the directory
top -- CPU utilization
sed -- to replace a text from file at run time
kill -9 -- to kill the process
touch filename -- to create zero-byte size file
tar -xvf filename.tar -- TO bundle a directory

2. Command to check for the running ports?

netstat -na | grep port_number

3. Command to check the processes running?

ps -ef

4. Command to replace a word from file?

sed -i's/old-text/new-text/g' input.txt
Examples for SED and AWK commands?
awk example:
https://www.geeksforgeeks.org/awk-command-unixlinux-examples/

1. AWK Operations:

(a) Scans a file line by line
(b) Splits each input line into fields
(c) Compares input line/fields to pattern
(d) Performs action(s) on matched lines
2. Useful For:
(a) Transform data files
(b) Produce formatted reports
3. Programming Constructs:
(a) Format output lines
(b) Arithmetic and string operations
(c) Conditionals and loops

4.Command to replace a word from file?

sed -i 's/old-text/new-text/g' input.txt Examples for SED and AWK commands?
awk example: https://www.geeksforgeeks.org/awk-command-unixlinux-examples/
Sed Example : https://www.cyberciti.biz/faq/how-to-use-sed-to-find-and-replace- text-in-files-in-Linux-Unix-shell/

5.What is swap memory?

When the actual memory of RAM is full then swap memory is used to process the request.

6.Command to kill a process?

kill -9 PID

7.Process to free disk space?

When we see memory alert then follow the below steps:

1) Login to the ec2 using ssh

2) **df -h** -- shows the file systems memory and check which file system is occupied with space example /var

cd /var -- Change to var directory

du -sk -- TO check each directory size in var

Delete the directory / clear the space which has consumed more space..

8.What is Inode?

An Inode is an index node. It serves as a unique identifier for a specific piece of metadata on a given filesystem.

Each piece of metadata describes what we think of as a file.

9.We have disk space available but still unable to create a file, what might be the reason?

Basically, an Inode is used for each file on the filesystem.

So, running out of Inodes generally means we are unable to create file even if we have filesystem.

10.Command to find a file and delete the file using the same command?

find / -name "FILE-TO-FIND" -exec rm -rf {} \;

Bash Scripting:

1.#! in bash scripting means = **Shebang**

2.Difference between bash and shell script?

Shell scripting is a method to automate tasks as a collection of commands. Shell script can be executed on any shell.

Example of shell scripting:

#!/bin/sh myString="GeeksforGeeks"echo "myString: $myString"Bash: Bourne after shell.

Bash scripting is a subset of shell scripting.Example of bash scripting:

#!/bin/bash myString="GeeksforGeeks"echo "myString: $myString"

3.Write a bash script to check a directory in a location, and should display the message if the directory is available?

#!/bin/bash

if [-d "/path/to/dir"]then

echo "Directory /path/to/dir exists."else

echo "Error: Directory /path/to/dir does not exists." Fi

4.Write a bash script to check if a service is running or not, if not running then script should start the service?

```
#!/bin/bash service=replace_me_with_a_valid_service
if (( $(ps -ef | grep -v grep | grep $service | wc -l) > 0 ))then
echo "$service is running!!!"else
/etc/init.d/$service startfi
```

5.How to know whether the previous command has been executed successfully or not?

With the help of exit status.
If exit status is 0 then it has been successfully executed.
If Exit status is 1 then it has not been successfully executed.

6. $? = is the exit status of the most recently-executed command.

7. $# = Total number of command line arguments passed.

8. $* = It's a space separated string of all arguments. For example, if $1 is "hello"and $2 is "world", then $* is "hello world".

9. $@ = Stores all the arguments that were entered on the command line.

10. $_ = special variable.

11.How to run bash script in the background?

Use nohup to run the script in background.
Example: **sudo nohup ./hello_world.sh**

12.Which version of Linux are you using?

We can say **RHEL 7.9** or **amazon Linux**
Note: The versions will get changed as per time.

What is kernel:

kernel is a interface between your hardware and software.
User --> Application --> shell --> kernel --> hardware.
Hardware: CPU, Memory
Kernel: Programs
Shell: GUI, bash
Application: Browser, calendar other software's.
User: Users.
Shell and Kernel together built in one package called O.S.
Application and shell are called as Software.
If we execute any command then shell will talk to kernel and kernel will communicate with Hardware and execute the operations.
Return Successful Exist Status

What is Shell:

Interface between users and Kernel/OS
CLI is a shell
GUI is a shell.
Find your shell:
echo $0 --> WHich shell you are using
cat /etc/shells --> Available shell
cat /etc/passwd --> To know which shell user is assigned with.

Types of shells:

Gnome: graphical environment in linux.(Desktop GUI)
KDE: Another Graphical environment in linux.
sh: Shell (Bourne Shell,developed by unix by stepehn bourne.)
Bash: New feature of shell with enhancement. (Bourne Again shell)
csh anmd tcsh : c shell and Tenex c shell advance version on cshell
ksh : Korn shell

Starting a shell:

By default, every user will have a shell configured.
Cat /etc/passwd we can see the username with shell configured.
if want to switch to other shell then execute sh and do echo $0
same for other shells as well.

How to run a script:

Absolute Path: /home/sabair/scripts./xyz-script
Relative Path: cd /home/sabair/scripts./xyz-script
Bash script_name./script_name

Script naming conventions:

1) Always create a directory to store scripts
2) Script name should identify the function
3) Script should end with. shell (If multiple shells are used) ex: .sh .bash

Script File Permissions:

All scripts to be executed should have proper executable file permissions.
eg: rwxr-xr-x
To change permissions
chmod 755 file_name

Basic Administration Tasks:

top
df -h
free -m
uptime

iostat
top get stuck because that will continuously monitor.
use top | head -10.

Basic Shell Scripts:

Output to screen

```
#!/bin/bash
# Simple output script
echo "Hello World"
```

Defining Tasks

```
#!/bin/bash
# Define small tasks
whoami
echo
pwd
echo
hostname
echo
ls -ltr
echo
```

Defining variables

```
#!/bin/bash
# Example of defining variables
a=Imran
b=Afzal
c='Linux class'
echo "My first name is $a"
echo "My surname is $b"
echo 'My surname is $c'
```

Read Input

```
#!/bin/bash
# Read user input
echo "What is your first name?"
read
echo
echo "What is your last name?"
read b
echo
echo Hello $a $b
```

Scripts to run commands within

```
#!/bin/bash
# Script to run commands within
clear
echo "Hello `whoami`"
echo
echo "Today is `date`"
echo
```

```
echo "Number of user login: `who | wc -l `"
echo
```

Read input and perform a task

```
#!/bin/bash
# This script will rename a file
echo Enter the file name to be renamed
read oldfilename
echo Enter the new file name
read new filename
mv $oldfilename $newfilename
echo The file has been renamed as $newfilename
```

for loop Scripts:

Simple for loop output

```
#!/bin/bash
for i in 1 2 3 4 5
do
echo "Welcome $i times"
done
```

Simple for loop output

```
#!/bin/bash
for i in eat run jump play
do
echo See Imran $i
done
```

for loop to create 5 files named 1-5

```
#!/bin/bash
for i in {1..5}
do
  touch $i
done
```

for loop to delete 5 files named 1-5

```
#!/bin/bash
for i in {1..5}
do
  rm $i
done
```

Specify days in for loop

```
#!/bin/bash
i=1
for day in Mon Tue Wed Thu Fri
do
 echo "Weekday $((i++)) : $day"
done
```

List all users one by one from /etc/passwd file

```
#!/bin/bash
```

```
i=1
for username in `awk -F: '{print $1}' /etc/passwd`
do
 echo "Username $((i++)) : $username"
done
```

case Scripts:

```
#!/bin/bash
echo
echo Please chose one of the options below
echo
echo 'a = Display Date and Time'
echo 'b = List file and directories'
echo 'c = List users logged in'
echo 'd = Check System uptime'
echo
      read choices
      case $choices in
a) date;;
b) ls;;
c) who;;
d) uptime;;
*) echo Invalid choice - Bye.
           esac
```

This script will look at your current day and tell you the state of the backup

```
#!/bin/bash
NOW=$(date +"%a")
case $NOW in
          Mon)
                    echo "Full backup";;
          Tue|Wed|Thu|Fri)
                    echo "Partial backup";;
          Sat|Sun)
                    echo "No backup";;
          *) ;;
esac
```

do-while Script

Script to run for a number of times

```
#!/bin/bash
c=1
while [ $c -le 5 ]
do
      echo "Welcone $c times"
      (( c++ ))
done
```

Script to run for a number of seconds

```
#!/bin/bash
```

```
count=0
num=10
while [ $count -lt 10 ]
do
      echo
      echo $num seconds left to stop this process $1
      echo
      sleep 1
num=`expr $num - 1`
count=`expr $count + 1`
done
echo
echo $1 process is stopped!!!
echo
```

If-then Scripts:

Check the variable

```
#!/bin/bash
count=100
if [ $count -eq 100 ]
then
  echo Count is 100
else
  echo Count is not 100
fi
```

Check if a file error.txt exist

```
#!/bin/bash
clear
if [ -e /home/iafzal/error.txt ]
      then
      echo "File exist"
      else
      echo "File does not exist"
fi
```

Check if a variable value is met

```
#!/bin/bash
a=`date | awk '{print $1}'`
if [ "$a" == Mon ]
      then
      echo Today is $a
      else
      echo Today is not Monday
fi
```

Check the response and then output

```
#!/bin/bash
clear
echo
```

```
echo "What is your name?"
echo
read a
echo
echo Hello $a sir
echo
echo "Do you like working in IT? (y/n)"
read Like
echo
if [ "$Like" == y ]
then
echo You are cool
elif [ "$Like" == n ]
then
echo You should try IT, it's a good field
echo
fi
```

Other If statements

If the output is either Monday or Tuesday
if ["$a" = Monday] || ["$a" = Tuesday]
Test if the error.txt file exist and its size is greater than zero
if test -s error.txt

if [$? -eq 0]	If input is equal to zero (0)
if [-e /export/home/filename]	If file is there
if ["$a" != ""]	If variable does not match
if [error_code != "0"]	If file not equal to zero (0)

Comparisons:

-eq	equal to for numbers
==	equal to for letters
-ne	not equal to
!==	not equal to for letters
-lt	less than
-le	less than or equal to
-gt	greater than
-ge	greater than or equal to

File Operations:

-s	file exists and is not empty
-f	file exists and is not a directory
-d	directory exists
-x	file is executable
-w	file is writable
-r	file is readable

Shell

Whenever you login to a Linux system you are placed in a shell program. The shell's prompt is usually visible at the cursor's position on your screen. To get your work done, you enter commands at this prompt.

The shell is a command interpreter; it takes each command and passes it to the operating system kernel to be acted upon. It then displays the results of this operation on your screen.

Several shells are usually available on any UNIX system, each with its own strengths and weaknesses.

Different users may use different shells. Initially, your system administrator will supply a default shell, which can be overridden or changed. The most commonly available shells are:

Bourne shell (sh)
•C shell (csh)
•Korn shell (ksh)
•TC Shell (tcsh)
•Bourne Again Shell (bash)

Each shell also includes its own programming language. Command files, called "shell scripts" are used to accomplish a series of tasks.

Shell Script Execution

Once a shell script is created, there are several ways to execute it. However, before any Korn shell script be can executed, it must be assigned the proper permissions. The chmod command changes permissions on individual files, in this case giving execute permission to the simple file:

```
$ chmod +x simple
```

After that, you can execute the script by specifying the filename as an argument to the bash command:

```
$ bash simple
```

You can also execute scripts by just typing its name alone. However, for that method to work, the directory containing the script must be defined in your PATH variable. When looking at your .profile earlier in the course, you may have noticed that the PATH=$PATH:$HOME definition was already in place. This enables you to run scripts located in your home directory ($HOME) without using the ksh command. For instance, because of that pre-defined PATH variable, the simple script can be run from the command line like this:

```
$ simple
```

(For the purposes of this course, we'll simplify things by running all scripts by their script name only, not as an argument to the ksh command.)

You can also invoke the script from your current shell by opening a background subprocess - or subshell - where the actual command processing will occur. You won't see it running, but it will free up your existing shell so you can continue working. This is really only necessary when running long, processing-intensive scripts that would otherwise take over your current shell until they complete.

To run the script you created in the background, invoke it this way:

```
$ simple &
```

When the script completes, you'll see output similar to this in the current shell:

```
[1] - Done (127) simple
```

It is important to understand that Korn shell scripts run in a somewhat different way than they would in other shells. Specifically, variables defined in the Korn shell aren't understood outside of the defining - or

parent - shell. They must be explicitly exported from the parent shell to work in a subsequent script or subshell. If you use the export or typeset -x commands to make the variable known outside the parent shell, any subshell will automatically inherit the values you defined for the parent shell.For example, here's a script named lookmeup that does nothing more than print a line to
standard output using the myaddress (defined as 123 Anystreet USA) variable:

```
$ cat lookmeup
print "I live at $myaddress"
```

If you open a new shell (using the ksh command) from the parent shell and run the script, you see that myaddress is undefined:

```
$ ksh
$ lookmeup
I live at
$
```

However, if you export myaddress from the parent shell:

```
$ exit
$ export myaddress
```

and then open a new shell and run the lookmeup script again, the variable is now defined:

```
$ ksh
$ lookmeup
I live at 123 Anystreet USA
```

To illustrate further how the parent shell takes processing precedence, let's change the value of myaddress in the subshell:

```
$ myaddress='Houston, Texas'
$ print $myaddress
Houston, Texas
```

Now, if you exit the new shell and go back to the parent shell and type the same command:

```
$ exit
$ print $myaddress
123 Anystreet USA
```

you see that the original value in the parent shell was not affected by what you did in the subshell.

ERROR CODE:

200, 301, 404, & OTHER NUMBERS: HTTP ERROR CODES
You've probably seen it before: 404 Page Not Found.

404? What's that?

It's an HTTP status code.
There are a surprising number of HTTP status codes aside from the two or three you may have run into in the wild. They exist to represent the status of requests that browsers make to load websites, and they're split into five groups.
1XX — Informational Responses
2XX — Success Responses
3XX — Redirection Responses
4XX — Client Error Responses
5XX — Server Error Responses
You can see these codes in action if by using developer tools. With Chrome, just press F12 (on Windows) or ⌥⌘I (on macOS), then select network at the top, and reload the page you're on.
We'll be highlighting some well-known and a few not-so-well-known codes.

1XX Codes

An informational response lets the browser know that the request to load a website or document was received and understood. Codes like these are issued while the request processing continues on the server, and it lets the browser know that it should wait for a final response.
100 — Continue
A status code of 100 indicates that (usually the first) part of a request has been received without any problems, and that the rest of the request should now be sent.

2XX Codes

A success response lets the browser know that the request was received, understood, and accepted — that's what separates them from the 1XX codes.
200 — OK
This is the code that browsers receive when every has gone according to plan.201 — Created
This code indicates that a request was successful and as a result, a resource hasbeen created (for example a new page).
204 — No Content
The 204-status code means that the request was received and understood, but that there is no need to send any data back.
205 — Reset Content
This code is a request from the server to the client to reset the document from which the original request was sent. For example, if a user fills out a form, and submits it, a status code of 205 means the server is asking the browser to clear the form.
206 — Partial Content
This is a response to a request for part of a document. This is used by advanced caching tools, when a browser requests only a small part of a page, and just that section is returned.

3XX Codes

These status codes tell the browser that it must take additional action to complete the request. Many of these status codes are used in URL redirection.

300 — Multiple Choices

The 300-status code indicates that a page or document has moved. The response will also include a list of new locations so the browser can pick a place to redirect to.

301 — Moved Permanently

This tells a browser that the resource it asked for has permanently moved to a new location. The response should also include the location. It also tells the browser which URL to use the next time it wants to fetch it.

304 — Not Modified

The 304-status code is sent in response to a request (for a document) that asked for the document only if it was newer than the one the client already had. Normally, when a document is cached, the date it was cached is stored. The next time the document is viewed, the client asks the server if the document has changed. If not, the client just reloads the document from the cache.

307 — Temporary Redirect

307 is the status code that is sent when a document is temporarily available at a different URL, which is also returned. There is very little difference between a 302-status code and a 307-status code. 307 was created as another, less ambiguous, version of the 302-status code.

4XX Codes

This type of status code is intended for situations in which an error seems to have been caused by the browser or user, like the infamous 404 error.

400 — Bad Request

A status code of 400 indicates that the server did not understand the request due to bad syntax.

401 — Unauthorized

A 401-status code indicates that before a resource can be accessed, the client must be authorized by the server.

402 — Payment Required

The 402-status code is not currently in use, being listed as "reserved for future use". It's interesting to think about how this will be used in the future, especially now that Chrome natively blocks some intrusive ads.

403 — Forbidden

A 403-status code indicates that the client cannot access the requested resource. That might mean that the wrong username and password were sent in the request, or that the permissions on the server do not allow what was being asked.

404 — Not Found

The best known of them all, the 404-status code indicates that the requested resource was not found at the URL given, and the server has no idea how long for.

408 — Request Timeout

A 408-status code means that the client did not produce a request quickly enough. A server is set to only wait a certain amount of time for responses from clients, and a 408-status code indicates that time has passed.

410 — Gone

A 410-status code is the 404's lesser-known cousin. It indicates that a resource has permanently gone (a 404-status code gives no indication if a resource has gone permanently or temporarily), and no new address is known for it.

415 — Unsupported Media Type

A 415-status code is returned by a server to indicate that part of the request was in an unsupported format.

5XX Codes

Simply put, these codes are sent when the server failed to fulfil a request. 500 - Internal Server Error

A 500-status code (which developers see more often that they want) indicates that the server encountered something it didn't expect and was unable to complete the request.

503 — Service Unavailable

A 503-status code is most often seen on extremely busy servers, and it indicates that the server was unable to complete the request due to a server overload.

503 Isn't So Bad

If you ever see one of these errors on your own website, and you don't know what to do, take a look at this list. With this, you'll be able to let us (if we host your website) know what's actually going on when your website looks like it's broken.

Git & GitHub:

1.What will git init do?
This will help us to initialize **. git** folder in local repository.

2.What does. git contains?
.git file contains the configuration of the repository, like **hooks, config** etc.

3.Why do we use git?
Git is a version control tool which is used to **track the changes of the project**.

4.Which branching strategy are you using?
We use **GITFLOW strategy.**
Release branch is used for **PRODUCTION Environment**.
Develop branch is used for **development/staging Environment**.
Feature branch is used for **next updates**.
Hotfixes is used for **fix the bugs of Production server**.
Master is used as a copy of **RELEASE branch**.

5.Git cherry-pick?
Cherry-pick is used to pick a specific commit of a branch to another branch.

6.Git stash?
For example, if I'm working on a specific branch and I don't want to save or commit the changes, in the meantime I got the request to work on other file from same branch then I can use gitstash.
This will help me to push the files to temporary stash memory.

7.Git rebase?
This is used to Merge two different branches.

8.Git merge?
This is used to merge two different branches.

9.Git fork?
This is used to copy a repo from other account to our account on GitHub.

10.Difference between git clone and git pull?
Git clone is used to download the complete repository from central to local.

Git pull is to pull the latest changes of code from central to local.

11.Difference Between Git merge and Git Rebase?
Git merge and rebase are used to merge the two branches but git rebase will keep the commit history more clear than git merge.

12.Difference between Git pull and git fetch?
Git pull is to pull the latest changes of code from central to local.Git fetch is used to pull the metadata configuration to local like branches, configuration, HEAD, Ref etc.

13.Is Git a distributed repository system or central repository system?
Git is a central repository system.

14.Difference between Distributed and Central Repository?
The main difference between centralized and distributed version control is that, in centralized version control, the versions are saved in the remote repository, while in distributed version control, versions can be saved in the remote repository as wellas in local repositories of the local machines.

15.Which version of git are you using?
git version 2.34.1. windows.1

16.What are merge conflicts?
If we have two similar files in different branches with similar content,
when we try to merge these branches then there is a chance to see merge conflicts. This happens because git will not apply the auto-merge strategy if the file is in same region.

17.How to resolve the merge conflicts?
By default, git will Auto-Merge many changes if we see merge conflicts then git will leave that to us to resolve.
In this case there is no automation way to resolve the conflicts.
we need to check the differences in both the commit and manually make changes and push them back to repo.
This will help us to resolve the merge conflicts.
Note: Mostly Merge conflicts will be taken care by Development Team as they are aware of the code changes.

18.How to merge multiple commits into a single commit.
We can use squash to merge multiple commits into single commit.

Apache Tomcat:

1.Default port number of Tomcat?
Tomcat runs on **8080** port number.

2.Default Deployment location of Tomcat?
Webapps will be the **default deployment location**.

3.Can we run multiple tomcat servers, If yes then please explain the process?
Yes, we can run multiple tomcat servers by changing the startup and shutdown port numbers of one server.

4.How to create Users in tomcat?
We can create users in **tomcat-users.xml file**.

5.Tomcat is Developed in which language?
Java language.

6.What are the dependencies for tomcat?
Java is the only prerequisite for tomcat.

7.Apache Tomcat is which server?
Application Server.

8.Difference between app and web servers?
App server is mainly used for to serve dynamic content.Web servers are used to serve static content.

9.list of app servers available in the market?
Tomcat, Jboss, Weblogic

10.Which version of tomcat are you using?
Tomcat-9.0.65

11.File name of Apache Tomcat Logs?
catalina.out is the log file for tomcat.

12.How to change the port number of Tomcat?
Go to tomcat>**conf folder** Edit **server.xml** Search **"Connector port"**
Replace "8080" by your port number Restart tomcat server.

Nginx and Apache:

1.Default port for apache?
Default for Apache is **80.**

2.Default port for nginx?
Default port for Apache is **80.**

3.Difference between apache and nginx?
Apache and Nginx both are web servers.
Nginx can be used as load balancer and we dont need any extra configurations.
Apache can be used as load balander but we need to do extra configurations.
Nginx is pretty fast than apache in serving static content.

4.Default deployment location for apache and nginx?
Apache:/var/www/html/ Nginx: /usr/share/nginx/html

5.Log File names of apache and nginx?
Apache:/var/log/httpd/access_log & /var/log/httpd/error_log
Nginx: /var/log/nginx/access_log & /var/log/nginx/error_log

6.How to change the default port of apache and nginx?
Apache: /etc/httpd/conf/httpd.confNginx: /etc/nginx/nginx.conf

HA PROXY:

1.What is HA Proxy?
HA proxy is the high availability, open-source load balancer, reverse based proxy which is used for maintaining the website to work smoothly without causing any overload.

2.What is reverse Proxy?
Reverse proxy server will be at the front of the servers such that when the users sent the request the reverse proxy server acts as load balancer and distributes the traffic to the servers. Due to this the users can't directly interact with the servers.

3.Configuration file name of HA PROXY?
haproxy.cfg

4.How to configure HA Proxy?
We can configure the haproxy at **etc/haproxy/haproxy.cfg**

AWS:

1.List of aws services you have worked on?
I have worked on **IAM, ec2, s3, applciation loadbalancer, Route53, VPC, CLoud watch, cloud trial, ecr and ecs.**

2.What is region and availability zone in aws?
A region is the collection of **avaialbility zones/data centres.** Availabilty zone is an independent data centre.

3.what is edge location?
Edge locations are used for delivering the content with low latency and hightransfer rate.

4.Name the global services of aws?
S3, IAM, cloudfront and route53

5.What is aws access key and secret key?
AWS access key and secret key are generate for connecting to the AWS Account through the CLI.

Ec2:

1.Ec2 stands for?
Elastic Cloud Compute.

2.How many types of ec2 are available?
On Demand, Reserved and Spot Instances.

3.Different families of ec2?
General purpose, storage optimized, compute optimized, memory optimized, accelerated computing.

4.What is the type of ec2 you are using?
On Demand (t2 large -- 2 core cpu and 8 gb memory)

5.Can we attach multiple ec2 to one ebs volume?
No We can't attached multiple ec2 to same ebs volumes, but we can attach multiple ebs to single ec2.

6.What if the pem key of ec2 is lost?
If the pem key is lost then we need to take the sanpshot of the ec2 and launch anew ec2 using the snapshot using a new PEM key.

7.Can we connect to ec2 if the pem is lost,if yes then explain the process?
No we can't connect to ec2 without the pem key.
Only way is possible to take snapshot and launch using new pem key or if we have copied the ssh keygen to the ec2 earlier then we can login without pem key.

8.How to take a snapshot of ec2?
Select the ec2,go to settings and click on snapshot.

9.What is a snapshot?
Snapshots are a point of time copy of the ebs.
Example i have launched one amazon linux ec2 and installed Jenkins and took snapshot then the same snapshot can we used to launch new ec2 and by default it will be having Jenkins in it.

10. A snapshot of t2.micro has been taken, can we change the type of instance when using the snapshot to launch new ec2?
A snapshot of t2.micro can be changed to t2.large.

11.What is the 2/2 system checks in ec2?
1)System connectivity Checks
2)Instance Status Checks

12.What are the possible reasons for system 2/2 checks?
A) System connectivity Checks:
Loss of network connectivity, Loss of system power
Software issues on the physical host Hardware issues on the physical host

B) Instance Status Checks:
Mis-configured networking or startup configuration Exhausted memory Corrupted file system Incompatible kernel

13.What can be done if we see system checks as failure?

General scenario we can restart the instance and we can see the system checks will be normal as the instance will get hosted/launched on another hardware stack.

14.Difference between SG and NACL?

SG AND NACL both are firewalls which helps to restrict unwanted traffic.

SG works at instance level and by default everything is denied in SG. These are stateful, which means any changes which are applied to an incoming rule is automatically applied to a rule which is outgoing.

NACL works at subnet level and we can add rules to allow and deny the source. NACL works on rule number.

These are stateless, meaning any change applied to an incoming rule isn't automatically applied to an outgoing rule.

I have to ec2 instance in eu region and us region how you redirect the traffic from one region to another region?

To redirect traffic from one EC2 instance in one region to another EC2 instance in a different region, you can use several methods depending on your requirements and setup. Here are a few options:

DNS-Based Routing:

Use Amazon Route 53's latency-based routing or geolocation routing. This way, Route 53 will direct users to the instance in the region with the lowest latency or based on their geographic location.

You can configure a Route 53 health check to monitor the instances and route traffic to the healthy instance.

Load Balancers:

Set up an Elastic Load Balancer (ELB) in front of your EC2 instances. ELB can distribute incoming traffic across multiple instances in different regions based on various algorithms like round-robin or least connections.

Amazon's Application Load Balancer (ALB) supports cross-region load balancing and can route traffic to different regions based on URL paths, hostnames, or containerized applications.

Application-Level Redirection:

Modify your application to perform redirection based on the user's location or preferences. For example, you can use HTTP redirection codes (e.g., 301, 302) to redirect users to the appropriate region's URL.

Global Accelerator:

Amazon Global Accelerator helps improve the availability and performance of your applications by routing traffic to optimal endpoints over the AWS global network. It can intelligently route traffic to the closest healthy instance in any AWS region.

Custom Solutions:

Implement a custom solution using AWS Lambda and API Gateway to handle traffic routing based on specific criteria.

Use Amazon S3 for static content and redirect users to different S3 buckets based on their region.

Each of these methods has its advantages and considerations, so choose the one that best fits your use case in terms of performance, scalability, and cost-effectiveness.

AMI:

1.What does AMI stand for?
AMI stands for Amazon machine images

2.Difference between AMI and snapshot?
Snapshots are use to take the copy of certain time of ebs volume, whereas AMI is used for ec2.

3.What is AMI?
AMI are a template that has the configurations which can be used to provision ec2.

4.What is AMI id?
Thats the unique identifier number for each AMI.

IAM:

1.IAM stands for?
Identity access management.

2.How to create a user using IAM?
Got to IAM service, select create user and provide the details.

3.What are groups in IAM?
Groups can be used to keep multiple users in the same cluster.
Example if we have 5 users under development, then we can create a group with dev and add users in that group.

Apply set of rules/policies to that group and
each user within that group will have the same level of access.

4.What is role in IAM?
IAM role is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person.

5.What is Policy in IAM?
Policy in IAM define permissions for an action regardless the method use toper form. A policy is something that will be assigned to role/user/group.

6.Difference between role and policy?
IAM Roles manage who has access to your AWS resources, whereas IAM policies control their permissions.
A Role with no Policy attached to it won't have to access any AWS resources.

7.What are the different levels of access we can provide to users using IAM?
We can provide two types of access. Console/GUI access Programmatic/CLI Access.

8.What do you mean by identity in IAM?
Identity can be user, group, roles.

9.Types of Roles in IAM?
1)Aws service Role:
A service role is an IAM role that a service assumes to perform actions on your behalf.
2)AWS account:
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
3)Web identity:

Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

4)SAML 2.0 federation (Security Assertion Markup Language):

Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

5)Custom trust policy:

Create a custom trust policy to enable others to perform actions in this account.

10.Types of Policies?

1)Identity Based policies:

InLine Policy:

An inline policy is a policy created for a single IAM identity (a user, group, or role). Inline policies maintain a strict one-to-one relationship between a policy and an identity.
They are deleted when you delete the identity.

Managed Policies:

1)Aws Managed Policy
2)Customer Managed Policy

AWS managed policies – Managed policies that are created and managed by AWS.
Customer managed policies – Managed policies that you create and manage in your AWS account. Customer managed policies provide more precise control over your policies than AWS managed policies.

2)Resource-based policies:

Are attached directly to resources and specify permissions for specific actions on the resource by some principals.

3)IAM permissions boundaries:

Define the maximum permissions for an IAM entity and are used as safeguards.

4)Access control lists (ACLs):

Are attached to resources and control cross-account permissions for principals from other accounts.

5)Organizations Service Control Policies (SCPs):

Specify the maximum level of permissions for an organization's accounts. These policies are used to limit the permissions that can be assigned within member accounts.

6)Session policies:

Are advanced policies used during temporary sessions for roles or federated users.

EBS:

1.EBS stands for?

Elastic Block storage

2.Can we attach multiple ebs to one ec2?
Yes we can attach multiple ebs to one ec2.
Once ebs is attached then we need to follow below steps.
lsblk -- to check the list of filesystems
then we need to create a mount point to that file
systemdf -h -- to check the filesystem.

3.What is an EBS snapshot?
EBS snapshot is a copy of certain time/backup of ebs volume.

4.Different types of EBS volumes available?
General Purpose SSD volumes. Provisioned IOPS SSD volumes.
Throughput Optimized HDD and Cold HDD volumes. Previous generation Magnetic volumes.

VPC:

1.VPC stands for?
Virtual Private Cloud

2.How to create VPC?
Got to the VPC service click on create VPC. Give the name and IPV4 CIDR. Click on create VPC.
Then create subnets in different availability zones. Create a internet gateway and attach it to the VPC.
A default route table is created when the VPC is created and edit the routes and subnet associations.

3.How many VPC can be created in one aws region?
Five – 5

4.What is CIDR?
Classless inter domain range

5.How many Ips will be associated to 192.168.0.0/24?
256

6.How many ips will be used internally by aws for communication?
five

7.What are subnets?
Subnets are the rang of ipv4 address in the VPC.

8.What is the difference between public and private subnet?
Public subnet is associated with the internet gateway in the route table.
In private subnet it is not associated with the internet gateway in the route table.

9.What is IGW?
Internet gateway is attached to VPC, which helps for communication between VPC.

10.What is RT?
Route Table is used to route the traffic to targeted subnets.

11.What is NAT gateway/ NAT instance?
NAT gateway is a aws service which is used to connect with instances in private subnet.
NAT gateway service will be managed by aws.
Nat instances is like similar ec2 and also used connect with instances in private subnets.
NAT instance maintenance part will be on our head.

12.Nat will be deployed in which subnet?
Public

13.What is VPC Peering?

VPC peering is used to establish the connection between two VPC in the same region or different region or different aws account.

14.Explain the process of VPC peering?

Create two VPC in different regions with different Ip address. Create two instances in the both regions. Create a peering connection and accept the connection in the other region. Modify the route table by adding the private address in the routes.

15.What is VPC Transit Gateway?

VPC Transit Gateway service is used to connect with cloud and on-premise networks.

16.What is the difference between Internet Gateway and NAT gateway?

IGW allows both inbound and outbound access to the internet, NAT gateway will only allow outbound access.

17.What is vpc flow log service?

VPC flog log service is used to store the logs related to VPC.

18.Where can we store vpc logs using flowlogs service?

We can store logs in two ways.

1)Using Amazon s3.

2)Cloud watch log stream.

19.What is vpc endpoint and what is the difference between endpoint and nat gate way

A VPC endpoint is a connection between your Virtual Private Cloud (VPC) and another AWS service without requiring internet access or a public IP address. It enables you to privately access AWS services from within your VPC.

There are two types of VPC endpoints:

Interface Endpoints: These are elastic network interfaces with private IPs that serve as entry points for traffic destined to supported AWS services. Interface endpoints are powered by Private Link, allowing you to connect to services like Amazon S3, DynamoDB, and AWS Systems Manager without internet traffic.

Gateway Endpoints: These endpoints are used for accessing specific AWS services (such as S3 and DynamoDB) over a direct connection from your VPC to the AWS network. Gateway endpoints are used for services that support this type of endpoint and provide a route for the traffic directly to the service.

Now, let's discuss the difference between VPC endpoints and NAT Gateways:

VPC Endpoints:

Purpose: VPC endpoints are used to privately access AWS services without going through the internet.

Traffic Handling: They handle traffic destined for specific AWS services (depending on the type of endpoint) within the AWS network.

Connectivity: They establish a private connection between your VPC and the AWS service, ensuring secure and efficient communication.

Use Cases: Commonly used for secure data transfer to and from AWS services like S3, DynamoDB, and Systems Manager.

NAT Gateways:

Purpose: NAT Gateways are used for instances within your private subnets to access the internet or other AWS services that require internet access.

Traffic Handling: They handle outbound traffic from instances in private subnets, performing Network Address Translation (NAT) to hide the private IP addresses of the instances.
Connectivity: They provide internet connectivity to private instances while blocking inbound traffic initiated from the internet.
Use Cases: Typically used for instances that need to access external resources such as software updates, package downloads, or API calls to external services.
In summary, VPC endpoints facilitate private access to specific AWS services within your VPC, while NAT Gateways handle outbound internet-bound traffic from instances in private subnets, allowing them to access resources outside the VPC or connect to the internet securely.

Load Balancer:

1.Full form of ELB?
Elastic Load balancer.

2.How many types of load balancer are available in aws?
We have **4 types** of load balancer
1)Classic Load Balancer
2)Application Load balancer
3)Network Load balancer
4)gateway Load Balancer (Available in specific regions only)

3.How does Classic Load Balancer work?
Classic load balancer works on the **HTTP, HTTPS, TCP.**
If we have equal numbers of instances in both AZ then classic can be used.

4.How does Application Load Balancer work?
Application Load balancer works on **HTTP/HTTPS** protocol It works on **application layer** that is **layer 7.**
We can configure **different listeners.**
It will work based on **path prefix** and **rules configure**. **Target group** will be attached to the load balancer.

5.How does Network Load Balancer work?
Network Load balancer works on **TCP/UDP** protocol It works on Transport layer that is **layer 4.**
We can configure different listeners.
It will work based on path prefix and rules configure. Target group will be attached to the load balancer.
we can assign a **elastic ip** to NLB.

6.When can we use ALB and NLB?
ALB can be used if we are aware of the traffic hitting our application.
NLB will be used if we are not sure of the traffic hitting our application and if we are expecting sudden spikes in traffic then NLB
can easily handle such situations.

7. What is the difference between NLB AND ALB
Here's the difference between an NLB (Network Load Balancer) and an ALB (Application Load Balancer):

Traffic Handling:

NLB: Handles both TCP and UDP traffic, making it suitable for high-throughput applications and protocols that require low latency. It operates at the transport layer (Layer 4) of the OSI model.

ALB: Focuses on HTTP and HTTPS traffic, performing advanced routing based on content, such as URL paths and headers. It operates at the application layer (Layer 7) of the OSI model.

Target Types:

NLB: Supports IP addresses as targets, making it ideal for routing traffic to services that require direct access to IP addresses (e.g., TCP-based services, UDP services).

ALB: Supports various target types, including EC2 instances, ECS containers, Lambda functions, and IP addresses. It's well-suited for HTTP/HTTPS applications and microservices architectures.

Load Balancing Algorithms:

NLB: Uses flow-based load balancing, distributing traffic based on the flow of packets. It's designed for handling millions of requests per second with minimal latency.

ALB: Offers multiple load balancing algorithms, including round robin, least outstanding requests, and least connections. It's optimized for web applications and supports features like path-based routing and host-based routing.

Health Checks:

NLB: Supports TCP health checks, ensuring that target instances are healthy by checking TCP connectivity.

ALB: Provides both HTTP and HTTPS health checks, verifying the health of targets based on their responses to HTTP requests.

Features:

NLB: Focuses on high-performance and low-latency traffic routing, suitable for applications requiring efficient handling of TCP/UDP traffic.

ALB: Offers advanced features like content-based routing, WebSocket support, SSL termination, integration with AWS services (such as AWS WAF for web application firewall), and support for containerized applications.

In essence, NLB is best suited for applications that require high throughput and low latency at the transport layer, while ALB is tailored for HTTP/HTTPS applications with advanced routing capabilities and features at the application layer. Choosing between NLB and ALB depends on your specific application requirements and traffic patterns.

Autoscaling Groups:

1.ASG stands for?

Autoscaling Groups.

2.Difference between launch template and launch configuration?

Launch Configurations cannot be editable, if we want to update then we need to replace the launch configuration.

launch Template can be edited and can have different versions of template. We can pass parameters in Launch template.

3.Different types of scaling in ASG?

We have **Horizontal** and **vertical scaling.**

4.What is Horizontal Scaling?
Horizontal scaling means adding additional ec2 to our environment.

5.What is Vertical Scaling?
vertical scaling means adding more power to our existing ec2.

6. What are the different scaling policies available in Auto Scaling, and when would you use each one?
Auto Scaling offers three scaling policies:

Target Tracking Scaling: Scales the ASG to maintain a specific target value for a metric, such as CPU utilization or request count per instance. Use this for predictable workloads.

Simple Scaling: Adds or removes a fixed number of instances when a CloudWatch alarm threshold is breached. Suitable for sudden spikes in demand or specific events.

Step Scaling: Adjusts the ASG capacity in steps based on the magnitude of a CloudWatch metric. Useful for workloads with varying demand patterns.

In Auto Scaling Groups (ASGs), the algorithms used depend on the type of scaling policy configured:

Target Tracking Scaling:
Algorithm: Dynamic algorithm that adjusts ASG capacity to maintain a specified target value for a metric (e.g., CPU utilization). The algorithm calculates the desired capacity based on current metric values and scales the ASG accordingly.

Simple Scaling:
Algorithm: Basic algorithm that adds or removes a fixed number of instances when CloudWatch alarms are triggered. Scale-out adds instances, scale-in removes instances.

Step Scaling:
Algorithm: Algorithm that adjusts ASG capacity in steps based on metric thresholds. It evaluates metric values against predefined step adjustments to determine scale-out or scale-in actions.

Scheduled Scaling:
Algorithm: No dynamic algorithm invoking Auto Scaling Groups (ASGs), the algorithms used depend on the type of scaling policy configured:

S3 Buckets:

1.Why do we use s3 Buckets?
S3 buckets are used to **store fixed objects**.

2.What are objects in s3?
Objects can be referred to any file uploaded in s3.

3.Default s3 bucket class?
Standard.

4.Different classes in s3 bucket?
Amazon S3 Standard
Amazon S3 Intelligent-Tiering
Amazon S3 Standard-Infrequent Access
Amazon S3 One Zone-Infrequent Access
Amazon S3 Glacier Instant Retrieval
Amazon S3 Glacier Flexible Retrieval
Amazon S3 Glacier Deep Archive

5.How to deploy static website in s3?

We need to enable the static website hosting option in s3 bucket.
Create one index.html and one error.html file and upload the same to s3 bucket. Attach a policy to make the objects public in s3 bucket.
We can access the website using s3bucket url on browser.

6.What is versioning?

Versioning helps us to store file with multiple version.

7.what is bucket policy?

S3 bucket policies specify what actions are allowed or denied for which principles on the bucket that the bucket policy is attached to.

8.S3 object Encryptions?

Object encryption helps us to encrypt the data before saved to disk and decrypt when downloaded.
Encryption is a method which helps to secure the data.

9.What is object lock?

Object Lock can help prevent objects from being deleted or overwritten for a period of retention time.

10.Explain S3 Cross region replication?

Cross region replication helps to automatically upload the data into destination bucket without manual intervention.
This can be used as a backup for s3 bucket.

To encrypt objects in an existing Amazon S3 bucket after creation, you can do so using server-side encryption (SSE) through the AWS Management Console or AWS CLI. Here are the steps to encrypt objects in an S3 bucket after creation using the console:

Using AWS Management Console:

Log in to the AWS Management Console:
Go to the AWS Management Console (https://aws.amazon.com/console/).
Log in using your AWS account credentials.

Navigate to Amazon S3:

Once logged in, navigate to the "Services" dropdown at the top of the console.
Select "S3" under the "Storage" category. This will open the Amazon S3 dashboard.

Select the Bucket:

From the list of buckets in the Amazon S3 dashboard, click on the name of the bucket containing the objects you want to encrypt.

Select Objects to Encrypt:

In the bucket's content's view, select the objects (files) that you want to encrypt. You can select multiple objects if needed.

Choose Actions:

After selecting the objects, click on the "Actions" dropdown button above the object list.
Enable Default Encryption:
In the "Actions" dropdown, select "Change encryption."
Choose the encryption type you want to apply to the selected objects: **SSE-S3, SSE-KMS, or SSE-C.**
SSE-S3: Amazon S3 manages the keys used for encryption.
SSE-KMS: AWS Key Management Service manages the encryption keys.

SSE-C: You provide your own encryption keys.
Click "Save" to apply the encryption settings to the selected objects.

Using AWS CLI:

If you prefer using the AWS Command Line Interface (CLI) to encrypt objects in an S3 bucket, follow these steps:
Open your terminal or command prompt.
Use the aws s3 cp command to copy objects to the same bucket with encryption enabled:

For SSE-S3 encryption:

aws s3 cp s3://bucket-name/source-file.txt s3://bucket-name/destination-file.txt --sse AES256

For SSE-KMS encryption:

aws s3 cp s3://bucket-name/source-file.txt s3://bucket-name/destination-file.txt --sse aws:kms --sse-kms-key-id kms-key-id
Replace bucket-name, source-file.txt, destination-file.txt, and kms-key-id with your actual bucket name, file names, and KMS key ID.
Execute the command to copy and encrypt the objects in the S3 bucket.

How can you monitor and track Amazon S3 storage usage and performance?

You can monitor Amazon S3 storage usage and performance using AWS CloudWatch metrics, S3 access logs, S3 Storage Class Analysis, S3 Inventory reports, and AWS Trusted Advisor recommendations.

What are the best practices for securing Amazon S3 buckets?

Some best practices for securing Amazon S3 buckets include enabling bucket policies and ACLs to control access, using IAM roles and policies for fine-grained access control, enabling versioning and MFA delete to protect against accidental deletions, enabling logging and monitoring, encrypting data at rest and in transit, and regularly auditing bucket permissions and configurations.

Explain the concept of S3 object tagging.

S3 object tagging allows you to assign custom metadata (tags) to S3 objects, providing additional information for organizing, categorizing, and managing objects. Tags can be used for cost allocation, access control, and lifecycle management of objects.

What is the difference between S3 object storage and EBS block storage?

S3 object storage is designed for storing and retrieving unstructured data (e.g., files, images, videos) over the web, whereas EBS (Elastic Block Store) block storage is designed for persistent block-level storage used with EC2 instances. EBS volumes are attached to EC2 instances and provide low-latency access for applications.

Route53:

1.What is Route53?
Route53 is a DOMAIN NAME SERVICE in aws.

2.Is Route53 a global or regional service?
Global Service.

3.What is Alias in Route53?
Alias records help us to route the traffic to selected aws resources.

4.What are nameservers in Route53?
Amazon Route 53 automatically creates a name server (NS) record that has the same name as your hosted zone.
It lists the four name servers that are the authoritative name servers for your hosted zone.

5.What are different records available in Route53? A (address record)
AAAA (IPv6 address record)
CNAME (canonical name record)
CAA (certification authority authorization)
MX (mail exchange record)
NAPTR (name authority pointer record)
NS (name server record)
PTR (pointer record)

6.Different hostings policies available in Route53 and explain them?
Simple Routing Policy.
==========
Simple routing policy is a simple round-robin policy and can be applied when there is a single resource doing the function for the domain e.g. web server that serves content for the website.

Weighted Routing Policy.
Weighted routing policy helps route traffic to different resources in specified proportions (weights) e.g., 75% to one server and 25% to the other during a pilot release

Latency-based Routing (LBR) Policy.
==========
Latency-based Routing Policy helps respond to the DNS query based on which data center gives the user the lowest network latency.

Failover Routing Policy.
==========
Failover routing policy allows active-passive failover configuration, in which one resource (primary) takes all traffic when it's healthy and the other resource (secondary) takes all traffic when the first isn't healthy.

Geolocation Routing Policy.
==========
Geolocation routing policy helps respond to DNS queries based on the geographic location of the users i.e. location from which the DNS queries originate.

Geoproximity Routing Policy.
==========
Geoproximity routing helps route traffic to the resources based on the geographic location of the users and the resources.

Multivalue Routing Policy.

==========

Multivalve routing helps return multiple values, e.g. IP addresses for the webservers, in response to DNS queries.

Route 53 Traffic Flow.

==========

Route 53 Traffic Flow helps easily manage traffic globally through a variety of routing types combined with DNS Failover
in order to enable a variety of low-latency, fault-tolerant architectures.

Cloud Trial:

1.What is Cloudtrial?

Cloudtrail is a service that enables governance, compliance, Operational auditing and risk auditing of our aws account.
Cloudtrial is a service which helps us to track the activities in aws account. It records all the activities done by user, roles and aws services.

These recorded sessions are called as EVENTS.
CloudTrail is enabled on your AWS account when you create it. View events in Event History, where you can view, search,
and download the past 90 days of activity in your AWS account.

2.What is the different type of Log events?

Management Events

Capture management operations performed on aws resources. Free of cost for one management event by aws.
Management Events will be created by default. Ex: who signed in at what time and other api calls execute on resources.

Data Events:

Log the resource operations performed on or within a resource. Ex: when user uploaded, deleted and downloaded files in s3.

Insight Events:

Identify unusual activity, errors, or user behaviour in your account. Ex: Unauthorized API calls indicate someone tried to perform an action in your AWS account that they did not have permission to carry out.

What is the difference between CloudTrail trails and CloudWatch Logs in AWS?

CloudTrail trails capture and log API activity and management events in your AWS account, while CloudWatch Logs collect and store log data from various AWS services and custom applications. CloudTrail is focused on audit and governance, while CloudWatch Logs are for log monitoring and analysis.
How does AWS CloudTrail help in security monitoring and incident response?

AWS CloudTrail helps in security monitoring by providing detailed logs of API activity and changes to AWS resources, allowing security teams to detect unauthorized actions, analyse security incidents, and respond quickly to potential threats or breaches

Cloudwatch:

CloudWatch is a service used to monitor your AWS resources and applications that you run on AWS in real time. CloudWatch is used to collect and track metrics that measure your resources and applications.
It displays the metrics automatically about every AWS service that you choose.
You can create the dashboard to display the metrics about your custom application
and also display the metrics of custom collections that you choose.
You can also create an alarm to watch metrics.
For example, you can monitor CPU usage, disk read and disk writes of Amazon EC2 instance to determine whether the additional EC2 instances are required to handle the load or not.
It can also be used to stop the instance to save money.

What is CloudWatch Agent?

Cloud watch agents are responsible to push the metrics to CloudWatch dashboard.
Agents needs to be up and running on the ec2.

What is Metric?

Metrics are a measurement at a point in time for the system.

Types of Metrics:

Cpu utilization
DiskReadOps
DiskWriteOps
DiskReadBytes
DiskWriteBytes
NetworkIn and NetworkOut
NetworkPacketsIn and NetworkPacketsOut
MetadataNoToken

What is Alarm?

Cloudwatch alarm is used to monitor a single cloud watch metric or the result of Match expression using cloud watch metrics. Also, it sends out a notification based on the threshold we set for each service in the cloud watch alarm.

Type of Alarms:

There are **3** alarm states:
OK – Within Threshold.
ALARM – Crossed Threshold.
INSUFFICIENT_DATA – Metric not available/ Missing data (Good, Bad, Ignore, Missing).

What is cloudwatch log group?

A log stream is a sequence of log events that share the same source.
Each separate source of logs in CloudWatch Logs makes up a separate log stream.

What is cloudwatch Events?

CloudWatch Events help you to respond to state changes to your AWS resources.

1.What is cloudwatch?
CloudWatch is a monitoring tool used to monitor our aws resources, application.

2.What are default metrics and custom metrics?
Default metrics are cpu utilization, diskreads, diskwrites etc.

Custom metrics can be any metric which can be pushed with bash scripting, CLI or API.

3.How to configure a cloudwacth alert to a specific custom service?
We need to create a alarm and configure **SNS** to that alarm.

4.What is SNS?
Simple notification service which will helps the user to notify based onemail, SMS.

4.What are the different metrics we can configure on cloudwatch?
CPU Utilization

Service running or not Metric Disk write and not metric etc..

5.How to configure dashboard in cloudwatch?
Goto CloudWatch and click on dashboard, select the metric to which you want to create dashboard.

Select the type of widget you want to configure and save.

6.What is cloud watch agent used for?
Cloud watch agents are responsible to push the metrics to cloudwatch dashboard. Agents needs to be up and running on the ec2.

7.What is Metric?
Metrics are a measurement at a point in time for the system.

8.What is Alarm?
Cloudwatch alarm is used to monitor a single cloud watch
metric or the result of Match expression using cloud watch metrics. Also, it sends out a notification based on the threshold we set for each service in the cloud watch alarm.

9.What is CloudWatch log group?
A log stream is a sequence of log events that share the same source.

Each separate source of logs in CloudWatch Logs makes up a separate log stream.

10.What is CloudWatch Events?
CloudWatch Events help you to respond to state changes to your AWS resources.

CloudFront

What is Amazon CloudFront?
Amazon CloudFront is a content delivery network (CDN) service provided by Amazon Web Services (AWS) that helps speed up the delivery of content (such as web pages, videos, images) to users by caching content at edge locations closer to them.

How does CloudFront improve website performance?
CloudFront improves website performance by caching content at edge locations worldwide, reducing latency and speeding up content delivery to users regardless of their geographic location.

What are the key benefits of using Amazon CloudFront?
The key benefits of using Amazon CloudFront include faster content delivery, improved website performance, reduced latency, increased scalability, and global reach with edge locations in multiple countries.

What types of content can be delivered using CloudFront?
CloudFront can deliver various types of content, including static and dynamic web pages, streaming media (videos, audio), software downloads, APIs, and other digital assets.

How does CloudFront handle user requests for content?
When a user requests content, CloudFront routes the request to the nearest edge location that has the content cached. If the content is not cached at that edge location, CloudFront retrieves it from the origin server (e.g., S3 bucket, EC2 instance) and caches it for future requests.

What is the origin server in CloudFront?
The origin server in CloudFront is the original source of the content, such as an Amazon S3 bucket, an EC2 instance, an Elastic Load Balancer (ELB), or a custom origin server outside of AWS.

How can CloudFront improve the security of content delivery?
CloudFront supports HTTPS (SSL/TLS) encryption for secure content delivery, protects against DDoS attacks with AWS Shield, and offers access control features such as signed URLs and signed cookies for content protection.

What is the purpose of CloudFront edge locations?
CloudFront edge locations are distributed data centres located around the world that cache content closer to users. They help reduce latency and improve performance by delivering content from the nearest edge location.

Can CloudFront cache dynamic content?
Yes, CloudFront can cache dynamic content by configuring caching behaviors and setting appropriate cache control headers. It supports caching of both static and dynamic content to improve performance.

How does CloudFront handle content updates and cache invalidation?
CloudFront automatically updates cached content based on cache expiration settings (Time to Live or TTL). You can also manually invalidate cached objects or use versioning to ensure timely content updates.

What is the difference between CloudFront and S3 for content delivery?
CloudFront is a content delivery network (CDN) that caches content at edge locations for faster delivery to users, while Amazon S3 is an object storage service for storing and serving static files.

Can CloudFront cache content with query strings or cookies?
Yes, CloudFront can cache content with query strings or cookies by configuring cache behaviours and whitelisting query string parameters or cookies for caching.

How does CloudFront integrate with other AWS services?
CloudFront integrates seamlessly with other AWS services such as Amazon S3 for content storage, AWS Lambda for serverless functions, AWS WAF for web application firewall, and AWS Certificate Manager for SSL/TLS certificates.

What is the purpose of CloudFront distributions?
CloudFront distributions define how content is delivered and cached, including origin settings, cache behaviours, security settings, and distribution configurations.

How can CloudFront help in reducing bandwidth costs?
CloudFront can help reduce bandwidth costs by caching and serving content from edge locations, reducing the load on origin servers and minimizing data transfer costs.

What is CloudFront Access Logs, and how can they be used?
CloudFront Access Logs are logs that capture detailed information about requests made to CloudFront distributions. They can be analysed for traffic patterns, usage insights, and troubleshooting.

Can CloudFront be used for live streaming and video on demand (VOD)?
Yes, CloudFront supports both live streaming and video on demand (VOD) delivery through integrations with AWS Elemental Media Live and AWS Elemental Media Package.

How does CloudFront handle caching of content with different TTLs?
CloudFront handles caching of content with different Time to Live (TTL) values by respecting the shortest TTL for each cached object, ensuring that updated content is delivered promptly.

What is the role of CloudFront Signed URLs and Signed Cookies?
CloudFront Signed URLs and Signed Cookies are used for controlling access to private content by generating temporary URLs or cookies with restricted access permissions.

How can CloudFront be used for serving dynamic content from EC2 instances?
CloudFront can be used with Amazon EC2 instances by configuring a custom origin and cache behaviours to cache dynamic content responses, reducing the load on EC2 servers and improving scalability.

AWS Lambda

What is AWS Lambda?
AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS) that allows you to run code without managing servers. You upload your code, and Lambda automatically scales and executes it in response to events.

How does AWS Lambda work?
You upload your code (e.g., Node.js, Python, Java) to Lambda, define the triggers (events) that should invoke your code (e.g., API Gateway requests, S3 bucket uploads), and Lambda handles the execution, scaling, and management of your code in response to those events.

What are the benefits of using AWS Lambda?
The benefits of using AWS Lambda include reduced operational overhead (no server management), automatic scaling based on workload, cost savings (pay only for the compute time used), and fast time-to-market for deploying code.

What types of applications can you build with AWS Lambda?
You can build various types of applications with Lambda, including web applications, backend services, data processing pipelines, IoT (Internet of Things) applications, and serverless architectures.

What are Lambda functions?
Lambda functions are units of code that you deploy to AWS Lambda. They are small, single-purpose functions that can be triggered by events and execute tasks such as processing data, responding to API requests, or performing backend operations.

What programming languages are supported by AWS Lambda?
AWS Lambda supports multiple programming languages, including Node.js, Python, Java, Ruby, Go, and .NET Core. You can choose the language that best fits your application's requirements.

How does AWS Lambda pricing work?
AWS Lambda pricing is based on the number of requests (invocations) and the duration of execution (compute time). You pay only for the compute time used, rounded to the nearest 100ms, and there is no charge when your code is not running.

What is a Lambda trigger?
A Lambda trigger is an event that invokes (triggers) a Lambda function to execute. Triggers can include API Gateway requests, S3 bucket events (e.g., object creation), DynamoDB table updates, CloudWatch events, and more.

Can Lambda functions interact with other AWS services?
Yes, Lambda functions can interact with other AWS services through SDKs and APIs. For example, you can read/write data to S3, query DynamoDB tables, send/receive messages from SQS or SNS, and invoke other Lambda functions.

How does AWS Lambda handle concurrency and scaling?
AWS Lambda automatically scales your functions in response to incoming requests. It manages concurrency by creating multiple instances of your function (containers) to handle simultaneous requests, and scales up or down based on demand.

What is the maximum execution time for a Lambda function?
The maximum execution time for a Lambda function is 15 minutes (900 seconds). If a function runs longer than this limit, it will be terminated by Lambda.

Can Lambda functions access external resources or databases?
Yes, Lambda functions can access external resources and databases over the internet or within your VPC (Virtual Private Cloud). You can configure permissions and security settings to control access to these resources.

What is AWS SAM (Serverless Application Model), and how does it relate to Lambda?
AWS SAM is a framework for building serverless applications on AWS. It simplifies the deployment and management of Lambda functions, API Gateway resources, DynamoDB tables, and other serverless components.

How can you monitor and debug Lambda functions?
You can monitor Lambda functions using AWS CloudWatch, which provides logs, metrics, and alarms for monitoring function performance. You can also use X-Ray for tracing and debugging function invocations.

What is the cold start issue in AWS Lambda?
The cold start issue refers to the initial latency when a Lambda function is invoked for the first time or after a period of inactivity. Lambda mitigates this by reusing function instances (warm start) and optimizing performance over time.

Can Lambda functions be invoked synchronously or asynchronously?
Yes, Lambda functions can be invoked synchronously (e.g., API Gateway requests, AWS SDK calls) or asynchronously (e.g., S3 events, CloudWatch events, SNS notifications). Synchronous invocations wait for a response, while asynchronous invocations are non-blocking.

What is the AWS Lambda Execution Environment?
The AWS Lambda Execution Environment is the runtime environment where Lambda functions execute. It includes the operating system, language runtime, dependencies, and configurations needed to run your code.

How can you deploy and update Lambda functions?
You can deploy Lambda functions using the AWS Management Console, AWS CLI, SDKs, or AWS SAM. Updates to functions can be done by uploading new code versions or using continuous integration/continuous deployment (CI/CD) pipelines.

Can Lambda functions be used for real-time processing and streaming data?
Yes, Lambda functions can process streaming data in real-time using services like Amazon Kinesis Data Streams, Amazon DynamoDB Streams, or Amazon S3 event notifications. They are commonly used for real-time analytics, data transformations, and event-driven processing.

What is Lambda@Edge, and how does it extend Lambda functionality?

Lambda@Edge is a feature of AWS Lambda that allows you to run Lambda functions at CloudFront edge locations. It extends Lambda functionality to handle global content delivery, edge processing (e.g., CDN customization, security headers), and low-latency application logic at the edge.

ACM

What is AWS Certificate Manager (ACM)?

AWS Certificate Manager (ACM) is a service provided by Amazon Web Services (AWS) that simplifies the process of managing SSL/TLS certificates for your AWS resources, such as Elastic Load Balancers (ELB), API Gateways, and CloudFront distributions.

What is an SSL/TLS certificate?

An SSL/TLS certificate is a digital certificate that encrypts data transmitted between a user's web browser and a website, ensuring secure communication and protecting sensitive information like passwords, credit card details, and personal data.

What types of certificates does ACM support?

ACM supports two types of SSL/TLS certificates: public certificates, which are used for securing publicly accessible websites and applications, and private certificates, which are used for internal communication within your AWS resources.

How does ACM simplify certificate management?

ACM simplifies certificate management by handling certificate provisioning, renewal, and deployment automatically. It also integrates with other AWS services like Elastic Load Balancing (ELB) and Amazon CloudFront for seamless certificate deployment.

Can ACM automatically renew certificates before they expire?

Yes, ACM automatically renews certificates before they expire, eliminating the need for manual renewal and ensuring continuous SSL/TLS protection for your AWS resources.

What is the process of requesting and obtaining a certificate in ACM?

To request a certificate in ACM, you need to provide domain validation by confirming ownership of the domain through email validation or DNS validation. Once validated, ACM issues the certificate for use with your AWS resources.

How are ACM certificates deployed to AWS resources?

ACM certificates can be easily deployed to AWS resources such as Elastic Load Balancers (ELB), Amazon CloudFront distributions, API Gateways, and AWS Elastic Beanstalk environments using the AWS Management Console or API.

What is the difference between ACM-managed certificates and imported certificates?

ACM-managed certificates are SSL/TLS certificates that ACM automatically manages, renews, and deploys to supported AWS resources. Imported certificates are certificates that you bring from external Certificate Authorities (CAs) and import into ACM for use with AWS resources.

Does ACM support wildcard certificates?

Yes, ACM supports wildcard certificates (*.example.com) for securing multiple subdomains under a single domain name. Wildcard certificates simplify certificate management for websites and applications with dynamic subdomains.

Is ACM free to use?

Yes, ACM is a free service for managing SSL/TLS certificates in AWS. You only pay for the AWS resources (e.g., ELB, CloudFront) that use ACM certificates, and there are no additional charges for ACM itself

Cloudformation:

1.Why do we use CloudFormation?
CloudFormation Templates are used to provision infrastructure on aws environment.

2.What is IAC?
Infrastructre As Code.

3.Can you explain what a template is in the context of AWS CloudFormation?
A CF template can be written in JSON or YAML language. A template describes all your resources and their properties.

4.What are stacks?
A stack is a collection of AWS resources that you can manage as a single unit.

5.Sample Cloudformation Template to provision ec2?
AWSTemplateFormatVersion: 2010-09-09
Description: Part 1 - Build a webapp stack with CloudFormation

Resources:
WebAppInstance:
Type: AWS::EC2::InstanceProperties:
ImageId: ami-0d5eff06f840b45e9 # ImageID valid only in us-east-1 region InstanceType: t2.micro

6.How to add parameters in CF template while creating Stack?
While Parameters are technically optional, they are essential to building flexible CloudFormation templates
Example:
Parameters:
BucketNameParam:
Description: Name of the bucketType: String Default: PrimaryUploadBucket

Resources:
MyS3Bucket:
Type: "AWS::S3::Bucket"Properties:
AccessControl: PublicRead BucketName: !Ref BucketNameParam

7.What is nested stack?
Nested stacks are stacks created as part of other stacks. You create a nested stack within another stack by using the AWS::CloudFormation::Stack resource.

8.What is reference?
When you are declaring a resource in a template and you need to specify another template resource by name, you can use the Ref to refer to that other resource.

9.What is output in template?
The optional Outputs section declares output values that you can import into other stacks (to create cross-stack references),

return in response (to describe stack calls), or view on the AWS CloudFormation console.
For example, you can output the S3 bucket name for a stack to make the bucket easier to find.

10.What is mapping?

CloudFormation provides two elements known as Mappings and Conditionals. Mappings allow you to create simple "Key:Value" dictionaries or hashes for use in your resource declarations.
And Conditionals allow you to use some logic-based decisions in your resources to add or modify values.

ECS

What is Amazon ECS, and how does it fit into a DevOps environment?

Amazon ECS is a managed container orchestration service that allows you to run Docker containers on AWS. In a DevOps environment, ECS enables automated deployment, scaling, and management of containerized applications, promoting agility and efficiency.

Can you explain the difference between ECS and AWS EKS (Elastic Kubernetes Service)?

Amazon ECS is a fully managed container service that simplifies container orchestration, while AWS EKS is a managed Kubernetes service that allows you to run Kubernetes clusters on AWS. ECS provides a simpler and more integrated approach, while EKS offers more control and flexibility for Kubernetes users.

How do you define a task definition in Amazon ECS, and what does it include?

A task definition in Amazon ECS is a JSON or YAML file that defines how containers should be run within a task. It includes container definitions, CPU/memory limits, networking settings, volumes, IAM roles, and optional container health checks.

What strategies can you use to scale Amazon ECS services based on traffic or resource utilization?

You can use ECS Auto Scaling to scale services based on CloudWatch metrics like CPU utilization or custom metrics. ECS also supports manual scaling and scheduled scaling based on time-based rules or events.

How does ECS handle container networking, and what are the different networking modes available?

ECS uses AWS VPC (Virtual Private Cloud) networking for containers within a cluster. Networking modes include bridge mode (default for containers), host mode (shares host networking), and aws vpc mode (each container gets its own network interface).

What is an ECS cluster, and how does it relate to tasks and services?

An ECS cluster is a grouping of EC2 instances or Faregate resources where tasks are run. Tasks are instances of task definitions, and services manage the desired state of tasks, ensuring they are running and healthy within the cluster.

How can you deploy a multi-container application using ECS, and what are the benefits of using task definitions?

You can deploy a multi-container application by defining multiple container definitions within a task definition. Task definitions provide a centralized configuration for containers, simplifying deployment, updates, and scaling of complex applications.

What is AWS Fargate, and how does it differ from ECS EC2 launch type?

AWS Fargate is a serverless compute engine for containers that allows you to run containers without managing underlying EC2 instances. ECS EC2 launch type requires you to manage EC2 instances where containers run.

How do you integrate ECS with CI/CD pipelines for automated deployments?

You can integrate ECS with CI/CD pipelines using tools like AWS CodePipeline, AWS CodeBuild, or Jenkins. Pipelines can automate building Docker images, updating task definitions, and deploying new versions of services to ECS clusters.

What is a container image repository, and how does ECS integrate with repositories like Amazon ECR (Elastic Container Registry)?

A container image repository stores Docker images for use in ECS tasks. ECS integrates seamlessly with Amazon ECR, allowing you to store, manage, and deploy container images securely within your AWS environment.

How do you manage secrets and sensitive information in ECS tasks securely?

ECS supports integration with AWS Secrets Manager or AWS Systems Manager Parameter Store for managing secrets like API keys, database credentials, and environment variables securely. You can reference secrets in task definitions without exposing sensitive information.

What are ECS capacity providers, and how do they impact ECS cluster scaling and resource allocation?

ECS capacity providers define the type of resources (EC2 instances or Fargate) available to an ECS cluster for running tasks. They influence how tasks are placed, scaled, and managed within the cluster based on resource availability and capacity settings.

Can you describe the lifecycle of an ECS task, including task placement, scheduling, and termination?

When a task is launched, ECS determines where to place it based on task placement constraints, instance availability, and task definitions. The task scheduler ensures tasks are scheduled and managed within the cluster, and tasks can be terminated manually or automatically based on scaling rules or events.

How do you monitor and troubleshoot ECS clusters and tasks for performance issues or failures?

You can monitor ECS clusters and tasks using Amazon CloudWatch metrics, logs, and alarms. CloudWatch provides insights into CPU/memory utilization, task health, container logs, and application performance metrics for troubleshooting and optimization.

What security best practices do you follow when working with ECS clusters and containerized applications?

Security best practices for ECS include using IAM roles and policies for task execution, enabling VPC security groups and network ACLs, implementing least privilege access controls, encrypting sensitive data at rest and in transit, and regularly patching container images for vulnerabilities.

How do you handle rolling updates and blue-green deployments in ECS to minimize downtime during application updates?

ECS supports rolling updates and blue-green deployments through service update strategies. Rolling updates gradually replace container instances with new versions, while blue-green deployments launch a new set of tasks alongside existing ones, allowing for seamless cutover and rollback if needed.

Can you explain the difference between ECS service discovery and AWS App Mesh for microservices communication?

ECS service discovery allows containers within a cluster to discover and communicate with each other using DNS-based service discovery. AWS App Mesh is a service mesh that provides centralized control and monitoring of microservices communication, including traffic routing, load balancing, and observability.

How do you optimize ECS resource utilization and costs for running containerized workloads?

You can optimize ECS resource utilization and costs by right-sizing EC2 instance types or Fargate resources, using Auto Scaling policies based on workload demand, leveraging spot instances for cost savings, optimizing container CPU/memory limits, and reviewing AWS Cost Explorer for cost analysis.

What are the limitations or challenges you've faced when working with ECS, and how did you address them?

Common challenges with ECS may include managing complex task definitions, handling container networking issues, optimizing resource utilization, and integrating third-party tools or services. Addressing challenges often involves implementing best practices, using automation tools, leveraging AWS documentation and support, and continuous monitoring and optimization.

How do you stay updated with the latest ECS features, best practices, and industry trends in container orchestration and DevOps practices?

I stay updated with the latest ECS features, best practices, and industry trends by regularly reading AWS blogs, attending webinars and conferences, participating in AWS community forums, following industry experts on social media, and hands-on experimentation and training with new ECS capabilities and tools.

ECR

What is Amazon ECR?
Amazon ECR is a fully-managed Docker container registry service provided by AWS. It allows you to store, manage, and deploy Docker container images.

How does Amazon ECR integrate with other AWS services?
Amazon ECR seamlessly integrates with services like Amazon ECS (Elastic Container Service), AWS Fargate, and Kubernetes on AWS (EKS) for container orchestration and deployment.

What are the key benefits of using Amazon ECR?
Some key benefits include secure storage of container images, integration with AWS IAM for access control, scalability, high availability, and low latency image retrieval.

How do you push a Docker image to Amazon ECR?
You can push a Docker image to Amazon ECR using the docker push command after authenticating with your AWS credentials and tagging the image appropriately with the ECR repository URI.

What authentication methods does Amazon ECR support?
Amazon ECR supports AWS IAM authentication and registry-level permissions to control access to repositories.

How do you manage access control in Amazon ECR?
Access control in Amazon ECR is managed through AWS IAM policies that define who can push, pull, or manage images within specific repositories.

Can you automate image builds and uploads to Amazon ECR?
Yes, you can automate image builds and uploads using CI/CD pipelines with tools like AWS CodeBuild, Jenkins, or GitLab CI/CD.

What is the lifecycle policy in Amazon ECR?
A lifecycle policy in Amazon ECR defines rules for automatically cleaning up unused or expired images to save storage space and reduce costs.

How do you monitor Amazon ECR usage and performance?
You can monitor Amazon ECR using AWS CloudWatch metrics, logs, and alarms to track image pull requests, repository size, and performance metrics.

What is the difference between Amazon ECR and Docker Hub?
Amazon ECR is tightly integrated with AWS services and provides private repositories with IAM-based access control, while Docker Hub is a public registry with limited free private repositories.

Can you replicate Docker images between Amazon ECR repositories in different AWS regions?
Yes, Amazon ECR supports cross-region replication of Docker images to ensure availability and reduce latency for deployments in different regions.

How do you secure Docker images stored in Amazon ECR?
You can secure Docker images in Amazon ECR by implementing encryption at rest using AWS KMS keys, enabling image scanning for vulnerabilities, and regularly updating images with security patches.

What happens if a Docker image pull request exceeds Amazon ECR rate limits?
Amazon ECR rate limits can temporarily throttle or deny image pull requests. You can request limit increases or implement caching mechanisms to mitigate this issue.

Can you share Docker images between AWS accounts using Amazon ECR?
Yes, you can share Docker images between AWS accounts by configuring resource-based permissions and granting cross-account access to specific repositories.

How do you manage versioning and tagging of Docker images in Amazon ECR?
You can manage versioning and tagging of Docker images in Amazon ECR by following best practices such as using semantic versioning, tagging images with Git commit IDs or build numbers, and maintaining a versioning strategy.

What are the considerations for optimizing Docker image storage in Amazon ECR?
Considerations include using efficient base images, minimizing image layers, leveraging caching mechanisms, implementing lifecycle policies, and optimizing image sizes for faster deployment and reduced storage costs.

Can you integrate Amazon ECR with container orchestration platforms like Kubernetes?
Yes, Amazon ECR seamlessly integrates with Kubernetes on AWS (EKS) for container image management, deployment, and scaling within Kubernetes clusters.

How do you troubleshoot image push or pull issues in Amazon ECR?
You can troubleshoot image push or pull issues in Amazon ECR by checking AWS CloudTrail logs, examining IAM permissions, verifying network connectivity, and reviewing Docker client logs for errors.

What are the backup and disaster recovery strategies for Amazon ECR?
Backup strategies include regular snapshots of ECR repositories using AWS Data Lifecycle Manager or third-party backup tools. Disaster recovery plans involve replicating repositories across multiple AWS regions and ensuring data integrity.

How do you handle Docker image vulnerabilities in Amazon ECR?
You can handle Docker image vulnerabilities in Amazon ECR by enabling automated image scanning with services like Amazon ECR Public Scan or third-party vulnerability scanning tools. Regularly update base images and dependencies to patch known vulnerabilities.

EKS

What is Amazon EKS (Elastic Kubernetes Service), and why would a company use it?
Amazon EKS is a managed Kubernetes service provided by AWS. Companies use it to deploy, manage, and scale containerized applications using Kubernetes without the need to manage the underlying infrastructure.

How does Amazon EKS handle node provisioning and management in a Kubernetes cluster?
Amazon EKS automates node provisioning by allowing you to create managed node groups or use AWS Fargate for serverless container management. These options handle node scaling, updates, and infrastructure management.

What are the key components of an Amazon EKS cluster?

An Amazon EKS cluster consists of a control plane managed by AWS, worker nodes for running Kubernetes pods, networking components like VPC and subnets, and IAM roles for access control.

Can you explain the difference between Amazon EKS and self-managed Kubernetes clusters?

Amazon EKS is a managed service where AWS handles the control plane, scaling, and maintenance tasks. Self-managed Kubernetes clusters require you to manage the control plane, nodes, updates, and scaling on your own infrastructure.

How does Amazon EKS integrate with AWS services like IAM, VPC, and CloudWatch?

Amazon EKS integrates with IAM for role-based access control, VPC for networking isolation and security, and CloudWatch for monitoring cluster metrics, logs, and alarms.

What deployment strategies can you use with Amazon EKS for rolling updates and blue-green deployments?

Amazon EKS supports rolling updates by gradually replacing old pods with new versions. Blue-green deployments involve launching new pods alongside existing ones, allowing for seamless switching and rollback if needed.

How do you manage secrets and sensitive data in Amazon EKS securely?

Amazon EKS integrates with AWS Secrets Manager or Kubernetes Secrets for managing sensitive data like API keys, passwords, and configuration settings securely within the cluster.

What is Kubernetes RBAC (Role-Based Access Control), and how does Amazon EKS implement it?

Kubernetes RBAC allows granular control over access permissions within a cluster. Amazon EKS implements RBAC through IAM roles mapped to Kubernetes roles and service accounts for fine-grained access control.

How do you monitor and troubleshoot performance issues in Amazon EKS clusters and pods?

You can monitor Amazon EKS clusters using CloudWatch metrics, logs, and alarms. Troubleshooting involves analysing pod logs, checking cluster events, using kubectl commands for debugging, and leveraging AWS X-Ray for distributed tracing.

What are some best practices for optimizing Amazon EKS clusters for performance and cost efficiency?

Best practices include right-sizing worker nodes based on workload requirements, using spot instances for cost savings, optimizing pod resource requests and limits, enabling autoscaling based on metrics, and regularly updating Kubernetes versions for security and features.

Can you describe how Amazon EKS handles node upgrades and maintenance tasks?

Amazon EKS automates node upgrades and maintenance by allowing you to specify maintenance windows for updates, draining nodes gracefully before termination, and providing rolling updates for minimal downtime.

How does Amazon EKS support multi-AZ (Availability Zone) deployments for high availability?

Amazon EKS distributes worker nodes across multiple Availability Zones (AZs) within a region, ensuring redundancy and fault tolerance for applications running in the cluster.

What are pod disruption budgets, and how can they be used in Amazon EKS?

Pod disruption budgets (PDBs) define constraints on how many pods of a specific deployment can be disrupted during updates or maintenance. In Amazon EKS, you can configure PDBs to control pod evictions and maintain application stability.

What is the role of AWS CloudFormation in managing Amazon EKS resources?

AWS CloudFormation allows you to define Amazon EKS resources such as clusters, node groups, IAM roles, networking configurations, and service deployments as code, enabling infrastructure-as-code practices and automation.

How do you ensure data persistence and storage for stateful applications in Amazon EKS clusters?

Amazon EKS supports various storage options like Amazon EBS volumes, Amazon EFS file systems, and third-party storage solutions for persisting data and supporting stateful applications running in Kubernetes pods.

Can you explain the concept of pod security policies (PSPs) and how they enhance security in Amazon EKS?

Pod security policies (PSPs) define security controls and restrictions on pod behaviours such as privilege escalation, host networking, and filesystem access. In Amazon EKS, PSPs can be enforced using third-party solutions or Kubernetes-native policies.

How do you manage and deploy Helm charts in Amazon EKS for packaging Kubernetes applications?

Helm is a package manager for Kubernetes that simplifies application deployment and management. In Amazon EKS, you can use Helm to package applications into charts and deploy them using Helm CLI or AWS services like CodePipeline for automated deployments.

What are the considerations for networking and load balancing in Amazon EKS clusters?

Amazon EKS integrates with AWS networking services like Elastic Load Balancing (ELB) for distributing traffic to pods, Network Load Balancer (NLB) for external access, and AWS App Mesh for advanced service mesh capabilities.

How does Amazon EKS support CI/CD (Continuous Integration/Continuous Deployment) pipelines for automated application deployments?

Amazon EKS integrates with CI/CD tools like AWS CodePipeline, Jenkins, GitLab, and others for automating build, test, and deployment workflows. You can use Kubernetes manifests, Helm charts, or Docker images in CI/CD pipelines for deploying applications to EKS clusters.

What are the backup and disaster recovery strategies for Amazon EKS clusters and applications?

Backup and disaster recovery strategies for Amazon EKS include taking snapshots of critical data, using AWS backup services for EKS resources, replicating data across regions for redundancy, and implementing failover mechanisms for high availability and data resilience.

AWS RDS:

1.What is RDS service?

RDS is a relational database service, which helps us to setup and maintain database in cloud.

2.Types of Databases and examples?

Relational database stored information in tables.

Often, these tables have shared information between them, causing a relationship to form between tables.

This is where a relational database gets its name from. Example: MySQL, Oracle, MariaDB, Postgres and MS-SQL

Non-Relational Database, sometimes called NoSQL (Not Only SQL), is any kind of database that doesn't use the
tables, fields, and columns structured data concept from relational databases. Example: MongoDB, Bigtable, Redis, Raven DB, Cassandra, and CouchDB.

3.How can we migrate db from ec2 to RDS?

Migrating DB from EC2 to RDS:

==========

1)Get the dump of your existing DB on EC2

mysqldump -u root -p database_name> file_name.sql

2)Migrate the DB dump that you have taken in step 1 to RDS

mysql -h <replace-rds-end-point-here> -P 3306 -u <user_name> -p database_name< ec2db.sql

3)Connect to your RDS DB instance

mysql -h <replace-rds-end-point-here> -P 3306 -u rdsuser -p

4)**Switch to the database and verify the details**.

USE rdsdb SELECT * FROM table1;

4.What is multi-AZ RDS?

Multi Az option can be enabled on rds instance and it will create a replica/standby dB of original db in anaother availability zone.

The ebs volume attached for standby db will be in the created in the same region. RDS access only via database CNAME. so we cannot access standy db for any reason via RDS.

Standby replica cannot be used as extra capacity.

synchronus replication means if some is written in the db then the same will be replicated in standby db.

This is helpful if for some reasons primary is not working as expected then aws will detect
and it will take 1-2 mins to create a new instance. (Fail Over)

Fail over means it will have some downtime. Fault tolerance means there will be no downtime.

Points to remember:

==========

1)Multi AZ feature is not free and we need to pay twice the price.

2)Standy replica cannot be accessed directly unless a failure occurs.

3)Failover is highly available, not faulty tolerant.

4)Backups taken from standby (Removes performance impacts)

5)Multi AZ will be created in same region.

5.What is RPO?

1)Time between the last backup and whenever the failure occurred. Ex: We have taken backup at 6 a.m and failure happened at 7 a.m then RPO us for 1 hour and we will use 1 hr of data.

2)Amount of maximum data loss.
3)Business provides an RPO value.
4)influence technical solution and close.

6.What is RTO?

1)Time between the DR event and full recovery.
Ex: Disaster happened at 6 a.m and you took 30 mins to restore back the DB. Then 30 mins we can say as RTO.
2)Influenced by process, staff, tech and documentation.

7.How to take backup of RDS?

We can create snapshot of the RDS instance.
if we have Multi Az then backup will be taken from standby db.

8.What is retention period in db?

Retention period means after taking backup it will be kept for 7 days if retention period is selected as 7.

9.What is Read replica and when in which scenario it will be used?

Read Replicas is a copy of the primary instance.
Use case: Let's assume we have a primary RDS instance that serves both read and write traffic.
Due to the size of the instance and the amount of read-intensive traffic being directed to the database for queries. the performance of the instance is taking a hit. To help resolve this, you can create a read replica.
A snapshot will be taken of your database, and if you are using multi-AZ, then this snapshot will be taken of your secondary database instance to ensure that there are no performance impacts during
the process. Once the snapshot is completed, a read replica instance is created from this data.

10.Difference between multi az and read replicas?

Multi Az is configured for fail over and read replicas are used to improve the performance of the db.

11.What is the difference between Database Cluster and Database Instances?

Database Cluster
It's the name of the cluster that holds instances. Database Instances Its the name of each instance in the cluster.

12.Difference between RDS service and DB in ec2/vm?

RDS service is provided by aws where we can spin up rds instance and start using db.
DB in ec2/vm is standalone DB where we need to install and do the maintenance as well.

13.How can we connect to MS SQL RDS?

We can install sql workbench and enter the dbhost url and port number and click on test connection.

14.List Backup types supported by Amazon RDS?

There are two types of backups supported by Amazon RDS such as **automated backups** and **database snapshots.**
The automated backup enables point-in-time recovery of your DB instance automatically.
A DB snapshot is a manual process to back up the DB instance. It can be done as frequently as you wish.

15.What is RDS backup retention period?

That will be period to take backup of RDS.
If retention period is of 7 days then it will take backup after 7 days.

Points to Remember:

RDS Read-Replicas:

Read Replicas is a copy of the primary instance.

Use case: Let's assume we have a primary RDS instance that serves both read and write traffic.
Due to the size of the instance and the amount of read-intensive traffic being directed to the database for queries. the performance of the instance is taking a hit.
To help resolve this, you can create a read replica. A snapshot will be taken of your database, and if you are using multi-AZ, then this snapshot will be taken of your secondary database instance to ensure that there are no performance impacts during the process. Once the snapshot is completed, a read replica instance is created from this data.
1) Asynchronous replication.
2) It is written fully to the primary instance.
once its stored-on disk, it is then push to the replica db.
This means there could be small log.
3) These can be created in the same region or different region,it is a cross region replication.

why we need Read Replicas:

1) We can create 5 read-replicas per DB instance.
2) Each of these provides an additional instance of Read performance.
3) This allows you to scale out read operations for an instance.
4) Can provide global performance improvements.

What is SQL, and what is its role in database management?

SQL (Structured Query Language) is a programming language used for managing and manipulating relational databases. It allows users to perform tasks like querying data, inserting, updating, and deleting records, and managing database structures.

What are the basic components of an SQL statement?

An SQL statement typically consists of keywords like SELECT, INSERT, UPDATE, DELETE, WHERE clause for filtering data, JOINs for combining tables, and functions or expressions for calculations and transformations.

Can you explain the difference between SQL's DDL, DML, and DCL?

DDL (Data Definition Language) is used to define and manage database objects like tables, indexes, and constraints. DML (Data Manipulation Language) is used to manipulate data within tables (e.g., INSERT, UPDATE, DELETE). DCL (Data Control Language) is used to control access and permissions (e.g., GRANT, REVOKE).

What is a primary key in SQL, and why is it important?

A primary key is a unique identifier for each record in a table. It ensures data integrity by enforcing uniqueness and allows for efficient data retrieval and indexing.

How do you retrieve specific data from a database using SQL?

You can use the SELECT statement to retrieve specific columns or all columns from a table based on conditions specified in the WHERE clause. For example, SELECT * FROM customers WHERE city='New York'; retrieves all columns for customers from New York.

What is an SQL join, and can you explain different types of joins?

An SQL join combines data from two or more tables based on a related column. Common types of joins include INNER JOIN (returns rows where there is a match in both tables), LEFT JOIN (returns all rows from the left table and matching rows from the right table), and RIGHT JOIN (vice versa of LEFT JOIN).

How do you insert new records into a table using SQL?

You can use the INSERT INTO statement to add new records to a table. For example, INSERT INTO employees (name, salary) VALUES ('John Doe', 50000); inserts a new employee record into the employees' table.

What is an SQL transaction, and why is it important in database operations?

An SQL transaction is a series of database operations treated as a single unit that must either complete entirely or fail entirely (ACID properties). It ensures data consistency and integrity by allowing multiple operations to be grouped together and rolled back if an error occurs.

How do you update existing records in a table using SQL?

You can use the UPDATE statement to modify existing records in a table based on specified conditions. For example, UPDATE products SET price=price*1.1 WHERE category='Electronics'; increases the price of electronics products by 10%.

What is an SQL view, and how is it different from a table?

An SQL view is a virtual table derived from one or more existing tables. It stores a query's results but does not store actual data. Views provide a way to simplify complex queries, restrict data access, and present data in a different format without modifying the underlying tables.

Docker:

1.What is Docker?

Docker is for containerization where we can deploy different applications on different containers.
Docker containers are independent of Operating system and the boot up time for docker is very fast comparing to VM.
Docker containers can be easily ship from one env to another env and the image scan be stored in central Repo (Docker Hub).

2.What is the difference between containerization and virtualization?

Virtualization enables you to run multiple operating systems on the hardware of a single physical server, while containerization enables you to deploy multiple applications using the same operating system on a single virtual machine or server.

3.Explain the Architecture of Docker?

Docker follows Client-Server architecture, which includes the three main components that are Docker Client, Docker Host, and Docker Registry.

1.Docker Client

Docker client uses commands and REST APIs to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.
Note: Docker Client has an ability to communicate with more than one docker daemon.
Docker Client uses Command Line Interface (CLI) to run the following commands-
docker build docker pull docker run

2.Docker Host

Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

3.Docker Registry

Docker Registry manages and stores the Docker images. There are two types of registries in the Docker - Pubic Registry - Public Registry is also called as Docker hub.
Private Registry - It is used to share images within the enterprise.

4.What is Docker file?

Docker File is a template which contains set of instructions which helps to build docker image.

5.What is Docker image?

A Docker image is a Template used to execute code in a Docker container.

6.What is docker container?

A docker container is a running state of image, which execute the code of docker image and help to run application.

7.Create sample docker file use case?

FROM alpine:3.12
MAINTAINER <Your_Name>
RUN mkdir /usr/local/tomcat/
WORKDIR /usr/local/tomcat
RUN apk --no-cache add curl && \apk add --update curl && \
curl -O https://mirrors.sonic.net/apache/tomcat/tomcat-8/v8.5.61/bin/apache-tomcat- 8.5.61.tar.gz
RUN tar xvfz apache*.tar.gz
RUN mv apache-tomcat-8.5.61/* /usr/local/tomcat/.

RUN rm -rf apache-*
COPY SampleWebApp.war /usr/local/tomcat/webapps
RUN apk update && apk add openjdk8
WORKDIR /usr/local/tomcatEXPOSE 8080
CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]

8.How to build the docker file?

docker build -t . (-t means tag '.' represent current location of docker file) OR
docker build -t /pathofDockerFile

9.Difference between CMD and Entry Point?

CMD and Entry Point both are executables which will help to execute the commands at the runtime of container.
We can have multiple CMD but only the last CMD will get execute.CMD are Overridable and Entry point is not overridable.
CMD and ENTRY both can we used is same Docker File but that's not the best practice.

EXAMPLE:
FROM ubuntu RUN apt-get update ENTRYPOINT ["echo","hello"]CMD ["WORLD"]
When running the container it will show "Hello,WORLD"
Suppose we are passign argument like docker run -it -d <image_name> Sabair Output: "hello Sabair"

10.Difference between COPY and ADD?

COPY is used to copy file from docker host to docker image.
ADD can be used to download file from internet and also to extract the tar file.

11.Explain different modules of Docker file?

FROM: where to download the base image. -
MAINTAINER: Non executable instruction used to indicate the author of Docker File.
ADD: It copies the file from source to destination and also extracts the file.
CMD: It specifies the intended command for the image.
ENTRYPOINT: when the container is up, entry point is the starting of execution.
ENV: This instruction can be used to set the env variables in the container.
EXPOSE: expose a specified port.
RUN: This instruction is used to execute a command on top of an existing layer.
USER: This is used to set the username.
VOLUME: Volume instruction is used to enable access to a location.
WORKDIR: To change the directory.
ONBUILD: It will add a trigger instruction.

12.Multistage Dockefile?

Multi stage Docker file is used to reduce the size of image, better security and to spin out containers much faster.

13.Example for single stage and multi stage docker file?

Example for single stage dockerfile.
Clone the below repo and add the DockerFile. https://github.com/betawins/multi- stage-example.git

Single stage Dockerfile:

==========

FROM openjdk:8-jdk-alpine

RUN mkdir -p /app/source

COPY . /app/source

WORKDIR /app/source

RUN./mvnwlean packageEXPOSE 8080 ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom", "-jar", "/app/source/target/multi-stage-example-0.0.1-SNAPSHOT.jar"]

Multi stage-Dockerfile:

==========

#Build Image

FROM openjdk:8-jdk-alpine as builderRUN mkdir -p /app/source

COPY. /app/source WORKDIR /app/source RUN ./mvnw clean package

#Run image

FROM openjdk:8-jdk-alpineWORKDIR /app

COPY --from=builder /app/source/target/*.jar /app/app.jarEXPOSE 8080 ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom", "-jar", "/app/app.jar"]

14.Docker scan?

Docker scan is used to scan the docker images for vulnerabilities.

By default, we have Docker scout available in Docker hub registry, will helps us to scan every image which is pushed to Docker hub.

15.How to list the images in docker?

docker image ps

16.How to delete the images in docker?

docker image rm <image_name>

17.How to get the details of docker image?

docker inspect <Docker_image>

18.What is dangling images?

Dangling images are layers that have no relationship to any tagged images. They no longer serve a purpose and consume disk space.

19.What is tag in docker images?

Tag in docker images specify the version of the image.

20.What is docker registry?

Docker registry is centralized place to store our images similar like GitHub.

21.How to push docker image to docker registry?

We need to login to docker registry first by using docker login

We need to tag image name with docker repo name once login is succeeded, use docker push <image_name>

22.How to run a container?

docker run -itd -p 8080:8080 -name test <image_name>

(-IT= Interactive Terminal -d= detach mode/foreground -p= port exposing -name=

name of image).

23.command to check the running containers?

docker container ps

24.Command to Get the details of containers?

docker container inspect <Container_id>

25.Can we delete a pause container?

No, we need to stop the pause container and then delete the container by using

docker container stop <Container_ID>

26.How to check the list of containers running and exited?

docker container ps -a

27.What is -it and -d in docker run commands?

-IT = interactive Terminal

-D = Detach mode /foreground/background

28.What is port mapping in docker containers?

Port mapping is used to expose a port to public.

29.How to connect to the running container?

docker container exec -it <Container_id> /bin/bash

30.How to kill a running container?

docker kill my_container

31.How to check the logs of container?

docker container logs <container_id> --follow

32.How to commit a running container?

docker commit <container_id>

33.What is docker volumes and how many types of volumes are available?

Docker volumes are used to store docker container data. This is used to store persistent data.

Even if our container is killed or deleted then the data will be stored in volume.

We have 3 types of volumes

1)Volume

2)tmpfs mount

3)BindMount

34.Command to run docker container with a volume attached?

docker run -d --name <container_name> -v <Volume_name>:/app <Image_name>

35.What is docker networks?

Docker networking is primarily used to establish communication between Docker containers and the outside world via the host machine where the Docker daemon is running.

36.Different types of networks and their usage?

The Bridge Driver:

This is the default. Whenever you start Docker, a bridge network gets created and all newly started containers will connect automatically to the default bridge network.

The Host Driver: You can use the host network if you don't want to rely on Docker's networking but instead rely on the host machine networking.

The Overlay Driver:

The Overlay driver is for multi-host network communication, as with DockerSwarm or Kubernetes. Overlay network allows us to communicate with containers running on different node machines.

37.What is docker compose?

Docker compose is a different tool used to provision multiple containers at the same time.
If we have dependencies on two containers then we can easily manage them withthe help of docker Compose.

38.Sample-two-tier: docker-compose file

version: '3'-> Depends upon docker-compose version
services:-> All the services should be in this
sectiondb:-> Name of service (name can be anything)
image: mysql:5.7 -> Images used to run
container volumes:-> volumes to store contaienr data
-db_data:/var/lib/mysql-> mount pointrestart:
always environment: -> Environmental variables
-MYSQL_ROOT_PASSWORD=somewordpress
-MYSQL_DATABASE=wordpress
-MYSQL_USER=wordpress
-MYSQL_PASSWORD=wordpresswordpress: depends_on: -> It will wait for db service to get started.
-db
image: wordpress:latest
ports: -> port to expose to outerworld- "8000:80"
restart: alwaysenvironment:
-WORDPRESS_DB_HOST=db:3306
-WORDPRESS_DB_USER=wordpress
-WORDPRESS_DB_PASSWORD=wordpress
-WORDPRESS_DB_NAME=wordpress
volumes: db_data: { }

39.Docker-compose commands?

1)**docker-compose up ->** To run the docker-compose file
2)**docker-compose up -d ->** To run the docker-compose file in detach mode
3)**docker-compose down** -> To stop & delete the docker-compose services
4)**docker-compose restart** -> To restart docker-compose
5)**docker-compose stop** -> To stop docker-compose.
6)**docker-compose start** -> To start docker-compose
7)**docker-compose ps** -> list of containers
8)**docker-compose pause** -> pause the docker-compose
9)**docker-compose unpause** -> unpause the docker-compose
10)**docker-compose top** -> to view top performances
11)**docker-compose up -d** --scale db=5 ->To run 5 db container (db is nothing but service name)
Note: we can't scale the services using port number because of port conflicts.
12)**docker-compose -f docker-compose2.yml up -d** -> To use other docker- compose file
docker-compose create -> It will create container with only default network
13)**docker-compose images** -> List of imags used in docker compose
14)**docker- compose kill** -> kill the containers
15)**docker-compose logs ->** To check the logs of containers
16)**docker-compose port webapp 80** -> To check if port 80 is bind with webapp service or not
17)**docker-compose exec webapp ls** -> Execute the command in webapp container

40.Docker Commands?

docker run hello-world -- run container of hello-world image
docker ps -- to check the list of running containers docker ps -a --list of running and exited containers
docker images -- list of images
docker search tomcat -- search for tomcat image locally
docker pull tomcat -- pull tomcat image from docker registry
docker run -it -p 1234:8080 tomcat -- RUN container using tomcat
imagedocker build -t sabair . -- Build docker image with name sabair
docker tag sabair_ubuntu sabair_ubuntu-PROD -- tag the image sabair_ubuntu with sabair_ubuntu-PROD
docker rm <Container_ID> -- delete container
docker rmi <Image_ID> -- delete image
docker exec -it 5267e21d140 /bin/bash -- Connect with image 5267 (container_id)
docker commit 5267e21d140 sabair_v2:latest -- save the changes made to container sabair_v2

41.Export/Import Docker Image to file?

docker save image:tag > arch_name.tar docker load -i arch_name.tar

42.Import/Export Docker Image to AWS ECR?

docker build -t sabair:v1 .
aws ecr get-login --no-include-email --region=ca-central-1docker tag sabair:v1 12345678.dkr.ecr.ca-central- 1.amazonaws.com/myrepo:latest
docker push 12345678.dkr.ecr.ca-central-1.amazonaws.com/myrepo:lastestdocker pull 12345678.dkr.ecr.ca-central-1.amazonaws.com/myrepo:latest

43.Kill and Delete Containers and Images?

docker rm -f $(docker ps -aq) # Delete all Containers
docker rmi -f $(docker images -q) # Delete all Images

#What is Docker Hub Registry

Docker Hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts. It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

Dockerfile for a Java application using Maven

 # Base image with Maven and OpenJDK
FROM maven:3.8-openjdk-11 AS build
WORKDIR /app
 # Copy and build the Maven project
COPY pom.xml .
COPY src ./src
RUN mvn clean package
 # Stage 2: Create a lightweight Java image
FROM openjdk:11-jre-slim
WORKDIR /app
 # Copy the built JAR file from the previous stage

COPY --from=build /app/target/myapp.jar ./app.jar
 # Expose port 8080 (assuming your Java application listens on that port)
EXPOSE 8080
 # Command to run the Java application when the container starts
CMD ["java", "-jar", "app.jar"]

multi-stage Dockerfile for building an Nginx server

 # Stage 1: Build Nginx with custom configuration
FROM nginx:alpine AS build
WORKDIR /usr/src/nginx
COPY nginx.conf .
RUN apk add --no-cache openssl-dev && \
openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 \
-subj "/C=US/ST=State/L=City/O=Organization/CN=example.com" \
-keyout /etc/ssl/private/nginx-selfsigned.key \ -out /etc/ssl/certs/nginx-selfsigned.crt && \
cp nginx.conf /etc/nginx/nginx.conf

 # Stage 2: Final Nginx image

FROM nginx:alpine
COPY --from=build /etc/nginx /etc/nginx
EXPOSE 80 443
CMD ["nginx", "-g", "daemon off;"]

Explanation:

- The first stage (`build`) starts with the Nginx Alpine image and sets up a custom Nginx configuration (`nginx.conf`) along with a self-signed SSL certificate generated using OpenSSL.
- The second stage uses the official Nginx Alpine image again and copies the configuration files (`/etc/nginx`) from the `build` stage.
- Port 80 (HTTP) and 443 (HTTPS) are exposed, and the `CMD` instruction starts Nginx in the foreground with the `daemon off;` directive to run as the main process.

You can place this Dockerfile in a directory along with your custom `nginx.conf` file and build the Docker image using the `docker build` command:

Copy code

```
docker build -t mynginx .
```

Base image with Node.js pre-installed

FROM node:14-alpine
 # Set the working directory inside the container
WORKDIR /app
 # Copy package.json and package-lock.json to the working directory
COPY package*.json ./
 # Install dependencies using npm
RUN npm install
 # Copy the rest of the application files to the working directory
COPY . .

 # Expose port 3000 (assuming your application runs on port 3000)
EXPOSE 3000
 # Command to start the application when the container starts
CMD ["npm", "start"]

Image with Python
FROM python:3.9-slim
 # Set the working directory inside the container
WORKDIR /app
 # Copy and install requirements
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt
 # Copy the rest of the application files to the working directory
COPY . .
 # Expose port 5000 (assuming your Flask app runs on port 5000)
EXPOSE 5000
 # Command to start the Flask application when the container starts
CMD ["python", "app.py"]

Ansible:

1.What is ansible used for?
Ansible is used for configuration management, if suppose we have 10 servers and need to make changes in the 10 servers then ansible is something which can help us to do the changes in a single click.
Ansible is agentless architecture which works on port number 22 (SSH).

We need to configure inventory/hosts which will have the Ip or dns of servers. With ansible adhoc commands or playbooks we can execute the job and make our work done.
Ansible is an example of push-based configuration management tool.

2.Explain the architecture of ansible?
Playbooks Modules Inventory Plugins

3.Is ansible push based or pull based?
Push based.

4.What is inventory in ansible?
Inventory can be also called as hosts file, which will contain the Ip address of other servers.

5.How many types of inventories available in ansible and explain them?
Static Inventory - If we have 10 servers then 10 ips will be configured in inventory.
Dynamic Inventory - If we are provisioning new ec2 servers and in this case dynamic inventory will help us to capture the newly provisioned server ips.
For this we need to have python script which will execute and capture the ip of newly provisioned servers.

6.What is playbook in ansible?
Playbook will consist of different task/plays which will get executed on servers.

7.ansible is written in which language?
Python

8.Ansible playbooks can be written in which language?
YAML Language, YML Language

9.Tell 15 modules in ansible?
Yum, Service, Shell, lineinfile, dir, copy, handlers, notifiers, apt service
Debug, registerfile, git, archive.

10.What is adhoc command in ansible?
Adhoc commands are single line commands to execute the task and server/node machines.
Example:
ansible all -m ping -- To check the connectivity between slaves

11.What are ansible roles?
Ansible roles are used to make small parts of a playbook.
meaning instead of having all the tasks within single playbook and make it clumsy, we create roles with small set of tasks.
Roles are also reusable, roles an be called with any playbook and start using them.

12.Direcotry structure of ansible roles?
```
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
└── main.yml
```

13.What is ansible vault?
Ansible vault is used to store the files with encryption.
if we have password then we use the vault to encrypt the file.

14.What is handlers in ansible?
handlers are used if we have dependency on different plays/task.
Example task2 should get execute only if task1 is successfully executed then
in task 1 we configure "Notifier" which helps to send the alert to handler in task2that the task is completed.
Note: Notifier name and Handler name should be same.

15.What is an Ansible galaxy?
Ansible Galaxy is essentially a large public repository of Ansible roles. Roles ship with READMEs detailing the
role's use and available variables. Galaxy contains a large number of roles that are

constantly evolving and increasing.

16.what is ansible tower?

Ansible Tower is more for enterprise edition, where we can use GUI and easily manage and execute our playbooks
and inventories.

17.Write 5 Sample Ansible playbooks?

https://github.com/betawins/Ansible-playbook-NcodeIT

ssh-authentication-less for ansible

1) Login to ansible master
2) create a ssh keygen
ssh-keygen -t rsa
3)copy public key
cat id_rsa.pub
4) Login to ansible worker node
5) create .ssh directory
mkdir .ssh
6) Chnage the ownership to user which is used to login
chmod ec2-user:ec2-user .ssh
7) Change permission to 600
chmod 600 .ssh
8) create a authorized_keys files
touch authorized_keys
9) edit the file and paste the ansible master public key
vi authorized_keys
10) change permissions
chmod 600 authorized_keys
11) DOne !!

 # To check the connectivity between slaves

ansible all -m ping
ansible all -m ping -i ansible_hosts

 # How to gather facts of slave machine

ansible all -m setup
Tips: -i is used if we are using more than one inventory file

 # What if you do not have SSH key-based? How to pass username and password?

ansible all -m ping --user=ansadm --ask-pass

 # TO checkthe uptime of a slave machine

ansible all -a uptime
Tips: -m is the module and -a should contain the command it should run which goes as an argument to command and shell.

 # check the free memory or memory usage of host

ansible all -a "free -m"

 # Execute a command as root user (sudo) on host
ansible all -m shell -a "cat /etc/passwd|grep -i ansadm" -s --ask-sudo-pass

 # Execute a command using become module
ansible all -m shell -a "cat /etc/passwd|grep -i ansadm" -b -K

 # Tips: -b is the option for become and by default it will become root user
–K is to tell ansible to ask for SUDO password

#Execute a command as a different user (sudo su)
ansible all -m file -a "path=/home/ansadm/test state=directory mode=0755" -b --become-user=ansadm

 # Create a Linux user group
ansible all -s -m group -a "name=test state=present" -b -K

 # Create a file with 755 permissions
ansible all -m file -a "path=/home/asnadm/testfile state=touch mode=0755"

 # Change ownership of a file
ansible all -m file -a "path=/home/ansadm group=weblogic owner=weblogic" -b

 # Install a package using yum command
ansible all -m yum -a "name=httpd state=installed"

 # Start or stop the service
To Start
ansible all -m service -a "name=httpd state=started enabled=yes"
To Stop
ansible all -m service -a "name=httpd state=stop enabled=yes"

Jenkins:

1.What is Jenkins?
Jenkins is a Opensource CICD tool, this is used to automate the jobs and easily deploy on environments.

2.What is CICD?
Continuous Integration, Continuous Deployment/Continuous Delivery.

3.What is continuous integration, continuous Delivery and continuous Deployment?
Continuous Integration:
When a commit occurs on GitHub repository then with the help of webhooks and based upon the event occur Jenkins job will get triggered, which will execute different stages and build a war package, this package will be push to antifactory location.
This part comes under Continuous Integration.
Continuous Deployment Means whenever a package is created then it will automatically deploy on the end environment.
Continuous Delivery makes sure that the environment is ready for deployment but it will actually not deploy the package.

4.Jenkins is written in which language?
Jenkins is written in **JAVA.**

5.Jenkins run on which port number?
8080

6.What is Jenkins file?

Jenkins File is a text file which consists of different stages. Each stage will have a set of tasks to be executed.

7.What is Jenkins pipeline?

Jenkins Pipeline will be executed based on **Jenkins File**, where different stages will get executed.
We have **two types** of pipelines,
1)Declarative
2)Scripted

8.Types of jenkins job/projects? Freestyle project.

Maven project.
Pipeline. Multibranch pipeline. External Job.
Multi-configuration project.
GitHub organization.

9.Name some jenkins plugins?

Git, Java, Mavn
role based access plugin Docker
tomcat Sonarqube
Nexus Artifactory & Slack

10.What is workspace in Jenkins?

Workspace is a place where all the Jenkins jobs data will be stored.

11.Different stages of Jenkins pipeline?

Git clone -- Helps to clone the source code. SonarQube -- To check the quality of code.
Maven -- TO build the source code and create package out of it.
Nexus -- TO push the artifacts to nexus repository. Slack -- To send the build out put to slack channel.
Email -- To send the build output to email.

12.How many types of pipelines we have in jenkins?

We have two types of pipelines,
1)Declarative
2)Scripted

13.Difference between declarative and scripted pipeline?

Declarative Pipeline is configured within the Jenkins file and stored in Central

Repository (GitHub)
Declarative Pipeline will start with pipelines and if there is any issue in pipeline then the moment pipeline is triggered it will throw the error.
Example syntax:

```
pipeline {
agent any

stages {
stage('Build') {
steps {
echo 'Building..'
}
}
```

```
stage('Test') {
steps {
echo 'Testing..'
}
}
stage('Deploy') {
steps {
echo 'Deploying           '
}
}
}
}
```

Scripted pipelines are configured inside the job.
Scripted pipelines will start with nodes and if there is any error in 100 the line then it will execute all the 99 lines and throw the error at the 100 line. Example:

```
node {
agent any

stages {
stage('Build') {
steps {
echo 'Building..'
}
}
stage('Test') {
steps {

echo 'Testing..'
}
}
stage('Deploy') {
steps {
echo 'Deploying     '
}
}
}
}
```

14.How to segregate user roles in Jenkins?

We can use role-based strategy plugin and add the user read, view and other access.

15.How to take backup of Jenkins server?

Jenkins backup can be taken in two ways either by using thin backup Plugin or by using bash script to take the backup of workspace and dump that in some other location.

16.What is master and slave configuration Jenkins?

Master and salve configuration is implemented for highly availability of Jenkins server.
We can offload the work from master to slave and master can be used only to pass order on slave nodes.

Slave node are also responsible to execute some specific job.
For example, if we have java and python applications then we can configure two slaves for JAVA and Python applications.
java slave is responsible to execute java related jobs and python slave for python related jobs.

17.What is label in slave?

Label are names to define our slave machines.
labels play very important role when dedicating a job to the particular slave.

18.What is executors in Jenkins?

If we have 3 executors then means we can run 3 jobs at the same time.

19.Write sample Jenkins Pipeline?

```
pipeline {
agent {
label "master"
}
tools {

// Note: this should match with the tool name configured in your jenkins instance (JENKINS_URL/configureTools/)
maven "MVN_HOME"

}
environment {
// This can be nexus3 or nexus2 NEXUS_VERSION = "nexus3"
// This can be http or https NEXUS_PROTOCOL = "http"
// Where your Nexus is running NEXUS_URL = "18.216.151.197:8081/"
// Repository where we will upload the artifactNEXUS_REPOSITORY = "soanrqube"
// Jenkins credential id to authenticate to Nexus OSSNEXUS_CREDENTIAL_ID = "nexus_keygen"
}
stages {
stage("clone code") {steps { script {
// Let's clone the source
git 'https://github.com/betawins/sabear_simplecutomerapp.git';
}
}
}
stage("mvn build") {steps { script {
// If you are using Windows then you should use "bat" step
// Since unit testing is out of the scope we skip themsh 'mvn - Dmaven.test.failure.ignore=true install'
}
}
}
stage("publish to nexus") {steps { script {
// Read POM xml file using 'readMavenPom' step , this step 'readMavenPom' is included in: https://plugins.jenkins.io/pipeline-utility-steps

pom = readMavenPom file: "pom.xml";
```

```
// Find built artifact under target folder
filesByGlob = findFiles(glob: "target/*.${pom.packaging}");
// Print some info from the artifact found
echo "${filesByGlob[0].name} ${filesByGlob[0].path}
${filesByGlob[0].directory} ${filesByGlob[0].length}
${filesByGlob[0].lastModified}"
// Extract the path from the File foundartifactPath = filesByGlob[0].path;
// Assign to a boolean response verifying If the artifact name existsartifactExists = fileExists artifactPath;
if(artifactExists) {
echo "*** File: ${artifactPath}, group: ${pom.groupId}, packaging:
${pom.packaging}, version ${pom.version}"; nexusArtifactUploader( nexusVersion:
NEXUS_VERSION,protocol:
NEXUS_PROTOCOL, nexusUrl: NEXUS_URL,
groupId: pom.groupId,version: pom.version,
repository: NEXUS_REPOSITORY, credentialsId: NEXUS_CREDENTIAL_ID, artifacts: [
// Artifact generated such as .jar, .ear and .war files.[artifactId: pom.artifactId, classifier: '',
file: artifactPath, type: pom.packaging],
// Lets upload the pom.xml file for additional information for Transitive dependencies
[artifactId: pom.artifactId,classifier: '', file: "pom.xml",
type: "pom"]
]
);
} else {

error "*** File: ${artifactPath}, could not be found";
}
}
}
}
}
}
```

20.What is Jenkins parameterized job?

Parameterized jobs will help us to pass the parameters in Jenkins job.

For example, if we want to deploy the same job in different environments then instead of creating different jobs

we can configure a parameter with environment and the same will be reflected in job and segregate environments.

21.What are the global variables in jenkins?

Global environment variables are the variables that can be used in any and every Pipeline or Job built on Jenkins.

The global variables are set via the Jenkins console and via the groovy script of a pipeline.

Example:

BUILD_NUMBER - The current build number. For example "153"

BUILD_ID - The current build id. For example "2018-08-22_23-59-59" BUILD_DISPLAY_NAME - The name of the current build. For example "#153". JOB_NAME - Name of the project of this build. For example "foo"

BUILD_TAG - String of "jenkins-${JOB_NAME}-${BUILD_NUMBER}".

EXECUTOR_NUMBER - The unique number that identifies the current executor.
NODE_NAME - Name of the "slave" or "master". For example "linux".
NODE_LABELS - Whitespace-separated list of labels that the node is assigned.
WORKSPACE - Absolute path of the build as a workspace.
JENKINS_HOME - Absolute path on the master node for Jenkins to store data. JENKINS_URL - URL of Jenkins. For example http://server:port/jenkins/ BUILD_URL - Full URL of this build. For example http://server:port/jenkins/job/foo/15/
JOB_URL - Full URL of this job. For example http://server:port/jenkins/job/foo/

22.What is Maven?
Maven is build tool used to build java applications.
With maven we can compile the source code and build the packages.

23.Maven lifecycles?
prepare-resources --Resource copying can be customized in this phase.
validate -- Validates if the project is correct and if all necessary information is available.
compile -- code compilation is done in this phase.
Test -- Tests the compiled source code suitable for testing framework.
package -- This phase creates the JAR/WAR package as mentioned in thepackaging in **POM.xml.**
install -- This phase installs the package in local/remote maven repository.
Deploy -- Copies the final package to the remote repository.

24.What is Pom.xml?
PROJECT OBJECT MODEL.
POM contains artifact id, versions, packaging, plugins and dependencies.

25.Maven Repositories?
Local Central and Remote Repositories.

26.What is SonarQube?
SonarQube is a tool to check the quality of code.

27.What are sonar scanners?
Sonar Scanner will help to run project analyses and send the results to SonarQube server.

28.What is quality gate and quality profile in sonarqube?
Quality Profiles are a core component of SonarQube where you define sets of Rules that, when violated, raise issues on your codebase.
Quality Gates are the set of conditions a project must meet before it should be pushed to further environments.
Quality Gates considers all of the quality metrics for a project and assigns a passed or failed designation for that project.

29.Sonarqube runs on which port number?
9000

30.What is nexus?
Nexus is a central repository to store the artifacts (packages)

31.Nexus runs on which port number?
8081

33.What is snapshots and releases in nexus?

Snapshots are not the final version of artifact and can be replaced. Release are final artifact version and this cannot be replaced.

Jenkins

1) **Default port :** 8080
2) **Jenkins home directory** : /var/lib/jenkins
3) **Jenkins port config file** : /etc/sysconfig/jenkins
4) **Jenkins secrets key :** /var/lib/jenkins/secrets/initialAdminPassword
5) **Jenkins Task logs** : /var/lib/jenkins/logs
6) **Jenkins app logs:** /var/log/jenkins/jenkins.log
7) **Jenkins jobs**: /var/lib/jenkins/jobs
8) **Jenkins workspace**: /var/lib/jenkins/workspaces

Terraform:

1. Basic terraform commands?

Common commands:

apply	Builds or changes infrastructure
console	Interactive console for Terraform interpolations
destroy	Destroy Terraform-managed infrastructure
fmt	Rewrites config files to canonical format
get	Download and install modules for the configuration
graph	Create a visual graph of Terraform resources
import	Import existing infrastructure into Terraform
init	Initialize a new or existing Terraform configuration
output	Read an output from a state file
plan	Generate and show an execution plan
providers	Prints a tree of the providers used in the configuration
push	Upload this Terraform module to Terraform Enterprise to run
refresh	Update local state file against real resources
show	Inspect Terraform state or plan
taint	Manually mark a resource for recreation
untaint	Manually unmark a resource as tainted
validate	Validates the Terraform files
version	Prints the Terraform version
workspace	Workspace management

All other commands:

debug	Debug output management (experimental)
force-unlock	Manually unlock the terraform state

2.What is a Provisioner Terraform?

Provisioners are used to execute scripts on a local or remote machine as part of resource creation or destruction.

3.What are terraform providers?

A provider is a Terraform plugin that allows users to manage an external API.
Provider plugins like the AWS provider or the cloud-init
provider act as a translation layer that allows Terraform to communicate with many different cloud providers, databases, and services.

4.What happens if a resource is removed from terraform state file?

Items removed from the Terraform state are only no longer managed by Terraform. For example, if you remove an AWS instance from the state, the AWS instance will continue running, but terraform plan will no longer see that instance.

5.What is State File Locking?

State file locking is a Terraform mechanism that prevents operations on a specific state file from being performed by multiple
users at the same time. Once the lock from one user is released, any other user who has taken a lock on that state file can operate on it.
This aids in the prevention of state file corruption. The acquiring of a lock on a state file in the backend is a backend operation.
If acquiring a lock on the state file takes longer than expected, you will receive a status message as an output.

6.What is a Remote Backend in Terraform?

Terraform remote backend is used to store Terraforms state and can also run operations in Terraform Cloud.
Multiple terraform commands such as init, plan, apply, destroy (terraform version
>= v0.11.12), get, output,
providers, state (sub-commands: list, mv, pull, push, rm, show), taint, untaint, validate, and many more are
available via remote backend. It is compatible with a single remote Terraform cloud workspace or multiple workspaces.
You can use terraform cloud's run environment to run remote operations such as terraform plan or terraform apply.

7.What is terraform taint and tainted resource?

The terraform taint command informs Terraform that a particular object has become degraded or damaged.
Terraform represents this by marking the object as "tainted" in the Terraform state, and Terraform will propose to replace it in the next plan you create.

8.What are components of terraform?

Terraform has two important components: Terraform Core and Terraform Plugins.

9.Modules in terraform?

A Terraform module allows you to create logical abstraction on the top of some resource set.
In other words, a module allows you to group resources together and reuse this group later, possibly many times.

10.What is Import in Terraform?

Terraform is able to import existing infrastructure. This allows us take resources we've created by some other
means (i.e. via console) and bring it under Terraform management.
The terraform import command is used to import existing infrastructure.

11.What is Workspace in Terraform?

Workspace can be used if we have multiple project/environments using the same configuration.

What is Terraform?

Terraform is very popular open-source tool used for infrastructure provisioning. It is developed by hashicorp(HCL). It will allow us to build, destroy, modify infrastructure in minutes. it supports lot of cloud and physical platforms. terraform supports many infrastructures platform based on providers. Providers are nothing but api calls to communicate with the cloud, other infrastructure.

Terraform uses HCL language.

All infrastructure resources can be define in .tf extension file.

HCL language uses Declarative and can be maintain in version control system.

Every object terraform provisions is called as Resource, a resource can be ec2, s3, iam etc..

#Steps To Install Terraform on Windows 10 or 8 or 7

1) Download Terraform

https://www.terraform.io/downloads.html

2) Unzip the terraform package

Extract the downloaded zip file, after extracting the zip file you can see terraform.exe file.

Extracted path can be Example "C:\Users\devops\Downloads\terraform_0.12.23_windows_amd64"

3) Configure environment variables for terraform

This PC(MyComputer)—>properties —>advanced system settings–>environment variables—>system variables—>path–edit–>new

paste the path "C:\Users\devops\Downloads\terraform_0.12.23_windows_amd64"

Click on Ok.

4) Verify terraform version

Open gitbash and enter

> terraform version

Terraform v0.12.23

Terraform .tf file can be created using any texteditor,notepad++,vi,vim or visual studio etc..

What is a Resource:

Resource is a object that terraform manages.

it can be any file in local machine.ec2.iam,s2,on cloud, database many more things.

HCL Basics:

==========

HCL files consists of block and arguments.

Example:

=======

<block> <parameters> {

key1= value1

key2 = value2

}

Create a file called local.tf the configuration file to create a file in local

resource "local_file" "pet" {

filename = "/root/pets.txt"

content = "we love pets!"

}

resource --> Block name

local -->provider
file -->resource type (What needs to be created)
pet --> it is a logical name to identify by terraform and can be named anything.
filename and contents --> arguments for the local_file resource type
Once we have the terraform resource file then we need to perform below commands
terraform init --> to initliaze the repository and download the dependencies
terraform plan --> dry run to check how it will be create the respource (It will not actually create them)
terraform apply --> to execute and create the infra based on the configuration
terraform destroy --> to remove/delete the existing infra.
terraform show --> to show/get the details of the resources created
Terraform Provider--> it is api call which is used to communicate with the resources, it can be aws, gcp, azure or local.

Providers are in three categories

1) **Official -->** provided by Hashicorp
2) **Partner** --> third party vendors
3) **Community** --> individual

When we execute terraoform init it will create a .terraform directory in the working directory
Configuration Directory:
==========
Main.tf -->main configuration file containing resource definition
variables.tf --> contains varibles declaration
output.tf --> contains outputs from resources
provider.tf --> contains the provider definition
Multiple providers:
==========
resource "local_file" "pet" {
filename = "/root/pets.txt"
content = "we love pets!"
}
resource "random_pet" "mypet" {
prefix = "MR"
separator = "."
length = "1"
}

variables:

==========
Instead of hardcoding all the variables in the same configuration file we can use variables so that it can be reuse again and again by only changing the variables file.
We need to create a file called variables.tf in the same directory
variables.tf
==========
variable "filename" {
default =""root/pets.txt"
}
variable "content" {
default ="we love pets!"
}

```
variable "prefix" {
default ="MR"
}
variable "separator" {
default ="."
}
variable "length" {
default ="1"
}
```

main.tf

```
======
resource "local_file" "pet" {
filename = "var.filename"
content = "var.content"
}
resource "random_pet" "mypet" {
prefix = "var.prefix"
separator = "var.separator"
length = "var.length"
}
```

variable block:

```
==============
variable "filename" {
default =""root/pets.txt"   --> Default value
type =  --> type of variable
description =   --> optional but can define why the varible is used for
}
```

Type	Example
String	I love pets
Number	1
bool	true/false
any	default value
list	["cat","dog"]
map	pet1= cat pet2=dog
object	complex data structure
tuple	complex data structure

Example for type list:

Varibales.tf:

```
variable "prefix" {
default = ["mr","mrs","miss"]
type = list
}
```

main.tf

```
reource "random_pet" "my=pet" {
prefix = var.prefix[0]
}
```

Example for type MAP:
Varibales.tf:
variable "file-content" {
type = map
default = {
"statement1" = "we love pets"
"statement2" = "We love animals"
}
}
main.tf
resource "local_file" "pet" {
filename = "/root/pets.txt"
content = "var.file-content[statement2]"
}

Example of type Object:

we can see various variables type in objects.
Ex:
Varibale "Bella" {
type = object({
name = string
color = string
age = number
food = list(string)
favorite_pet = bool
})
}
We can pass the same variables in the variables.tf
default = {
name = "bella"
color = "brown"
age = 7
food = ["fish","chicken"]
favorite_pet = true
}
Example of type tuple:
tuple is similar to list
list will use variable type of same
tuple will use different variable types
Ex:
variable kitty {
type = tuple([string,number,tuple])
default = ["cat", 7,true]
}

Using of variables:

==========
1) By using varibles.tf file
2) By using interactive mode (This will get activated if we dont pass defa;ult value in variable.tf file)

3) Command line flags
--> terraform apply - var "filename=/root/pets.txt" -var "prefix=MR"
4) Environment variables
--> export TF_VAR_filename="/root.pets.txt"
--> export TF_VAR_prefix= "MR"
--> Set-Item -Path env:TF_VAR_filename -Value 'wild.txt'
terraform apply
5) varibale definition file (Should be end with terraform.tfvars/terraform.tfvars.json)
--> for automatically loaded file name *.auto.tfvars/*.auto.tfvars.json
--> if we are saving the file with other name like varible.tfvars then we need to pass this in CLI
--> terraform apply -var-file varibale.tfvars

Variable definition precedence:

==========
If we use multiple ways to define variables for the same file then terraform uses variable definition precedence
Example:
--> main.tf
resource local_file pet {
filename = var.filename
}
--> variable.tf
variable filename {
type = string
}
--> export TF_VAR_filename="/root/cat.txt"
--> Set-Item -Path env:TF_VAR_filename -Value 'wild.txt'
--> terraform.tfvars
filename = "/root/pets.txt"
--> varible.auto.tfvars
filename = "/root/mypet.txt"
--> terraforma apply -var "filename=/root/best-pet.txt"

Precedence order:

==========
in the above example we have passed all the possible varibles,which will terraform laod first and which will override ?

Order	Option
1	Environment variables
2	Terraform.tfvars
3	*.auto.tfvars(alphabetical order)
4	-var or -var-file (Command line flags)

Resource Attribute reference:

==========
If i want to link two resources together by using resource attributes.
main.tf
======

```
resource "local_file" "pet" {
filename = "/root/pets.txt"
content = "My cat is MR.Cat"
}
resource "random_pet" "mypet" {
prefix = "MR"
separator = "."
length = "1"
}
```

When we execute terraform apply it will create random id with pet name,
now i want to add this pet name in my content file (using output of one resource as input for another resource).
main.tf
======

```
resource "local_file" "pet" {
filename = "/root/pets.txt"
content = "My cat is ${random_pet.mypet.id}"     (random_pet = resource type,mypet = resource name, id = attribute)
}
resource "random_pet" "mypet" {
prefix = "MR"
separator = "."
length = "1"
}
```

Resource Dependencies:

==========

From the above main.tf we have local_file which is dependent on random_pet resource.
We are not mentioning that anywhere but still terraform will figure that out, we call them as implicit dependency.
Still we can define the dependency explicit by ourself using "depends_on" module

```
resource "local_file" "pet" {
filename = "/root/pets.txt"
content = "My cat is MR.CAT"
depends_on = [
random_pet.mypet
]
}
resource "random_pet" "mypet" {
prefix = "MR"
separator = "."
length = "1"
}
```

Output variables:

==========

These are used to display the output of the resources.
Ex:

```
resource "random_pet" "mypet" {
```

prefix = "MR"
separator = "."
length = "1"
}
output my-pet {
value = random_pet.my-pet.id
description = optional name
}
when we use terraform apply, we can see the id as output.
we can use terraform output command to see the output of the resource.

Terraform state:

==========

Terraform state file will have the complete record of the infra created by terraform.
State file is considered as a blue print of all the resources terraform manages.
terraform. tfstate will be the name of the file and this will created only after using terraform apply command.
When we execute terraform apply then terraform will check for the state file config and main.tf configuration and make the changes.
If both the files are in sync and we are again trying to execute terraform apply then terraform will not make the changes but show
"Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed."
Each resource created by terraform will have the unique ID.
State files also capture the Metadata of the configuration file.
State file will help for better performance because of the cache of the data.
state file benefits in collaborating with different team members.
State files should be shared in the remote backend place so that team can access the state file.
State files also store the sensitive data so not recommended to store in public repo's like github, gitlab.
Terraform state is a json format file, never try to edit the state file manually.

Terraform Commands:

==========

Terraform init --> to initialize terraform and download the dependencies for the resources.
Terraform plan --> To dry run or show how terraform will be executing the resource file.
Terraform apply --> To execute and make changes of the resource file.
Terraform output --> to print the output present in the resource file
Terraform validate --> to check the syntax is correct or not
Terraform fmt (format) --> to correct the format of the configuration file
Terraform show --> to display the current state of the infrastructure.
Terraform providers --> to see the list of providers used in configuration file
Terraform refresh --> used to sync terraform with real world infrastructure
Terraform graph --> to create visual representation of dependency

Terraform mutable vs immutable infrastructure:

==========

Terraform as a IAC tool uses immutable infrastructure strategy.
Immutable means deleting the older infra and creating a newer one with new update.
Mutable means using the existing infra and updating the system with newer version.

Lifecycle rules:

==========

We can configure lifecycle rules to our resource file.
Terraform will destroy the file before creating a new file by default.
We can create a file before destroy by using the lifecycle rules.
resource "local_file" "pet&q..

KUBERNETES:

1.What is Kubernetes?

Kubernetes is also called as k8s.
k8s is an open-source popular orchestration tool developed and used by google in prod env.
We have different orchestration technologies like docker swarm, Kubernetes and mesos.

2.What are the different components in k8s?

We have two Kind of machines in k8s.
1)Master
2)Worker node
Again, we have different components on master and worker nodes.

Master Components:

==========

1)Kube api server
2)etcd
3)Controller
4)Scheduler
worker node components:

==========

1)Docker container runtime
2)Kubelet agent
3)Kubeproxy
Note: kubectl is a command line tool to execute k8s commands.

Api server: Acts as frontend of k8s, authenticate user, authorization and Api server is the only gateway to communicate with our k8s cluster.
Etcd: Cluster brain, which stores the information related to cluster, node in the form of key value pair.
Scheduler: Responsible to schedule pods on the node machine.
Controller manager: controller is the brain behind orchestration, if nodes, container, endpoint goes down then controller will work behind to bring them up.
Container runtime: any container service to start, stop container (Docker).
kubelet: Kubelet run on all the machines, kubelet make sure that the containers are running as expected on node machines.
KubeProxy: It maintains network rules on your nodes and enables network communication to your Pods.

3.What is Pod?

pod is the smallest unit in k8s.
pod will have at least one container.

4.What is Helper Container?

we can have multiple containers in one pod.

one container with application and another container will be helper container. Helper container: this will be deployed along with our application; helper container will help our application with any processing/functionality.

5.List out some Basic commands of Pods?

kubectl run nginx --image=nginx --> to run a pod with name nginx and image nginx

kubectl get pods --> to display the list of pods

kubectl describe pod nginx --> Detail information of the pod created.

kubectl get pods -o wide --> additional information of pod like node,ip etc

kubectl explain pods -->Detailed description of pods

kubectl get pods -w --> Continously watch the status of POD

kubectl delete pod pod_name --> To delete the pod

kubectl delete resourcetype resourcename

kubectl run nginx --dry-run=client --image=nginx --> To check how the commandwill execute.

kubectl delete pods --all --> Delete all pods

6.What is Labels?

Labels are key-value pairs which are attached to pods, replication controller and services. They are used as identifying attributes for objects such as pods and replication controller.

They can be added to an object at creation time and can be added or modified at the run time.

7.Basic commands to check/attach/delete labels?

kubectl describe pod <pod_name> --> to check the label attached

kubectl label pod firstpod env=test --> To attach label env=test

kubectl label --overwrite pod firstpod env=prod --> To update the label env

kubectl label pod firtpod env- --> To delete the label env

Kubectl label pods -all status=xyz --> To update the label to all pods

kubectl get pods --show-labels --> to check labels

8.Define the mandatory fields in YAML?

k8s definition filed contains four top level fields.Api version kind metadata spec

==========

kind version

==========

POD v1

service v1 ReplicaSet apps/v1 Deployment apps/v1

==========

9.How to run pod using YAML?

kubectl create -f <file_name>.yml --dry-run --> to check the outputkubectl create -f <file_name>.yml

10.Sample YAML file to create pod?

apiVersion: v1

kind: Pod

metadata: name:

firstpodlabels:

env: prodspec: containers:

- name: nginximage: nginx

10.What is Namespaces in k8s?

These are used to group your applications.
They can be helpful when different teams or projects share a Kubernetes cluster.

11.Difference between create and apply?

kubectl supports three types of object management.
1)Imperative commands --> Which are not created from yaml
2)Imperative object configuration --> Kubectl create is example for imperative.
3)Declarative object configuration --> Kubectl apply is example for declarative.
Imperative:
You have to manage different resources like pods, service, replica sets, etc by your own.
Imperative object configuration will helps us to modify the objects and these changes are not stored in yaml.
The kubectl create command creates a resource from a file or from stdin. JSON and YAML formats are accepted.
If the resource already exists, kubectl create will error.
Declarative:
K8 will take care of all the resources, all you need have to specify what is your actual requirement.
Declarative object configuration will help to modify the yaml file.
The kubectl apply command applies a configuration to a resource by file name or stdin. The resource name must be specified.
This resource will be created if it doesn't exist yet. If the resource already exists, this command will not error. JSON and YAML formats are accepted.

12.What is init container in pods?

Init containers are used if you want any container to execute before the app container starts.
Init containers are exactly like regular containers, except: Init containers always run to completion.
Each init container must complete successfully before the next one starts.

13.How to create init container?

apiVersion: v1kind: Pod metadata:
name: firstpodlabels:
env: prod name: sabair spec: containers:
-**name: firstcontainerimage:** nginx env:
-**name: mynamevalue:** sabair
-**name: secondcontainerimage**: nginx initContainers:
-**name: initcontainerimage:** nginx env:
-name: mynamevalue: sabair
-**name: City value:** Hyderabad args: ["sleep", "30"]

14.Services in k8s?

Services are used to expose our pods to outside world.
For example, we have deployed an Apache webserver and want to access from our computer/outside network then it is possible with services only.
services are also a object like nodes, pods in k8s.Default service will be Cluster Ip in k8s.
if we don't specify type in spec then it will assume as Cluster Ip only.

15.How many types of services are available in k8s?

Three types of services are available on k8s.
1)ClusterIp

2)Nodeport
3)Load balancer
ClusterIp:
Cluster ip is used to communicate within the cluster.
we may have some pods running frontend, backend and database.
And in micro services frontend will be talking to backend and databases.
we can create a service for backend and database and group the pods related to Microservice.
Each service will have a Ip and we call it as ClusterIp which help to communicate with other services.
NodePort:
NodePort is used to access for pod outside the world, means via browser. Port can be assigned between (30000 - 32767).
LoadBalancer:
LoadBalancer service can be used if we are using any cloud platform elb.

16.How services will work?
Services will check the request which is sent on port and redirect that based on label.

17.Replication Controller in k8s?
Replication controller or RC will ensure that these many pods are running on the cluster.
For any reason if the pod is down then RC will monitor that and spin up a new pod for us.
RC will get attached to the pods based on labels and selectors.

18.Why do we need Labels and selectors?
If suppose we have 100 of containers running then labels and selectors will help RS to filter the containers and apply monitoring based on labels and selectors.

19.Replica set in k8s?
RS and RC both purpose is same but they are not same. RC is older technology and k8s recommend to use RS. RS is just the updated version of RP.

20.Replication controller vs Replica set Difference?
RC and RS functionality is almost similar.
RC will work based on equality-based selector. Ex: env = prod
RC selects all resources with key equal to env and value equal to prod.RS will work based on equality-based selector and set based selector. Ex: env = (prod, test)
RS selects all resources with key equal to env and value equal to prod or test.

21.What is Deployments in k8s?
Deployments are used to deploy, upgrade, undo the changes, pausing and again resuming the changes.
Deployments use rolling update means the new version will be slowing updated with the older version and the users will be able to access our application without downtime. Hierarchy for deployment:
pods --> replicasets --> Deployment

22.How many types of Deployment strategies are used in k8s?
There are two types of deployment strategy in k8s.
1)Recreate
This will destroy the existing pods and create new. We can see downtime in recreate deployment strategy
2)Rolling update
This is the default deployment strategy. No downtime for rolling update strategy. For example, if we have 5 pods then 1 will be destroyed and a newer version will be created and 4 will be severing the traffic.

In this way it will destroy all the 5 and create new 5 pods slowly. we can see the difference when we use kubectl describe deployment deployment_name command In recreate it will scale down the RS to 0 and then Scale up to 5.
In Rolling it will scale down to 4 and and bring one new up and then vice versa.

23)what is Rollout,Rollback and versioning?

when we create deployment then a rollout is created and a Revision is created. When we again update the deployment then again, a rollout will be carried and New Revision is created.
These revisions are helpful when we want to rollback to previous version. Note: Rollout will get triggered only when there are change in the container
configurations only.

24)What is Maxunavailable and MaxSurge in Deployments?

Max unavailable means the number of unavailable pods at the time of update.
If we configure Maxunavailable as 1 then it will create one new and delete one old. If 5 as value then create 5 and delete 5.
This will help us to complete the deployment faster.

Max Surge: This will imply the number of pods that will be there on top of total replicas being mentioned. Example if the replicas in the deployment is mentioned as 3, when rolling update will kick in this property will define how many extra pods will be created at that point of time.

25)What will be the Maxunavailable, MaxSurge, MinReady Seconds and strategy if not mentioned in deployment?

Strategy: RollingUpdateMinReady Seconds: 0 Max surge and Max unavailable: 25%

26)How to check the history of deployment?

kubectl rollout history deployment <Deployment_name>

27)what is rollback in k8s and how to do the rollback?

Rollback will helps us to get back to old version of deployment.
How to do the rollback:
kubectl rollout undo deployment <deployment_name>This will roll back to previous version.

28)what is Resource Request in k8s?

When we schedule a pod on any node without any resource request then it will automatically take the resource
available on the node.
Resource request will be RAM and CPU request.

29)Why do we use resource request?

If suppose our pod needs 200MB of RAM then scheduler will check on which node the RAM is available and then it will schedule the pod on that node only.

30)what is Resource Limits in k8s?

Resource limit will ensure that container is not using more than the specified RAM and CPU.

30)Namespaces in k8s?

In Kubernetes, namespaces provide a way to divide a cluster into virtual partitions or segments.
They act as a logical boundary that isolates and separates resources within a cluster. In Kubernetes, A cluster can be divided into multiple namespaces, and each namespace can have its own set of resources. This helps organize and manage applications and services running in the cluster. By Default, the pod what we have created will be created in "Default" namespace.

31)Some important commands related to namespaces?

kubectl get ns --> List of name spaces
kubectl create ns <name> --> To create namespace
kubectl apply -f po.yml --namespace test --> Create pod in ns
kubectl get pods -n test --> to list pods in sn test.
kubectl get pods --all-namespaces --> To view the pods running in all namespaces.
kubectl delete pod <pod_name> -n test --> To delete pod from namespace test.
kubectl api-resources --> To check what all can be created inside namespace.
kubectl delete ns <namesapce> --> To delete the name space
kubectl config set-context --current --namespace=test --> To make test as default

32)How many namespaces are available in k8s by default and what is there use?

when we create k8s cluster then by default we can see 4 name spaces.

1)default

This ns will get created when we try to create any object without mentioning namespace

2)kube-node-lease

node lease is a mechanism for workers nodes to master about there health status and they are ready to take workloads.

lease should be sent for every 60 secs, if lease is not renewed then master may be considered nodes as unhealthy or unresponsive.

3)kube-public

serves as a central location for storing public resources that need to be accessed by all users and service accounts within a Kubernetes cluster.

4)kube-system

All the pods, management related stuff to create clusters are stored in kube-system

33)What is Service DNS?

In k8s service DNS, also known as the service discovery DNS,
is a built-in mechanism that allows communication between services using their names instead of their IP addresses. Each service deployed in a Kubernetes clusters assigned a DNS name that can be used by other services to access it.

34)what is Resource Quota in Namespace:

Resource quota in k8s helps us to enforce resource limits on namespace. We can restrict out namespace with certain CPU, memory and storage.
We have two types of resource quota.
1**)Resource based Quota.**
2)Compute based quota. Resource/object Based Quota:
==========
In resource-based quota we are going to set how many objects can be created in namespaces.
Ex: pods, services, PVC, RS etc. which are supported by namespaces.
Compute based Quota:
==========
In compute-based quota we can restrict the namespace with certain CPU limits.

35)what is LimitRange in k8s?

The Limit Range resource in Kubernetes allows you to define default and maximum resource limits for containers running within a namespace.
It helps ensure resource fairness and prevent containers from consuming excessive resources.

36)What is max and min in Limit range?

Min is used to allocate min amount of CPU and memory to a container. max is used to allocate max amount of CPU and memory to a container.
Min should be always lesser or equal to default request.
Max should be always greater or equal to default limit.

37)What is Max Limit/Request Ratio?

This will helps us to fix a ratio between max limit and max request.
If suppose the Max Limit/Request ratio is set as 2 and our max limit = 1000 and max request = 100 then 1000/100 = 10 : value we have set is MAX 2 then it will not create the pod.

38)What is Config Maps?

ConfigMaps in Kubernetes are used to store non-sensitive configuration data thatcan be consumed by pods or other Kubernetes objects. They provide a way to decouple configuration from application code, making it easier to manage and update configurations without
redeploying the application

ConfigMaps can be created in different ways

--from-literal
--from-file
--from-env-file andfrom directory.

39)How to create config maps?

kubectl create cm <Config_map_name> --from-literal=database_ip=192.168.0.1

40)What are Secrets in k8s?

Secrets in k8s is used to store small amount of secure data like passwords. Small amount of data can be up to 1MB of size.

Whenever we try to create any secret then it will be converted into base64 and stored.

Secrets Will be again of 3 types:

1)Docker-registry
2)Generic
3)tls

Most commonly used will be generic.

Secrets can be created in different ways same as configmaps like
--from-literal
--from-file
--from-env-file and from directory.

40)Taint and Tolerations in k8s?

Taint and Tolerations are advance k8s pod scheduling technique. Taint is attached to Node.
Toleration is attached to Pods.

Taint in general terms means a filter.
Tolerations with matching taint will be scheduled on tainted node. Tolerations allow the scheduler to schedule pods with matching taints.

41)Node Selector in k8s?

Node selector helps us to select node where the pod can be scheduled. It will work based on the label attached to node.

42)Node Affinity in k8s?

Node affinity is a set of rules used by the scheduler to determine where a pod canbe placed.

Two types of Node Affinity is available:

1)Preferred During Scheduling Ignored During Execution (Soft Scheduling):

Means we are asking scheduler to check if the values/label is matching then schedule pod on the node and if node with same label is not available then it will schedule on any available node. Ignored During Execution means if the pod is already scheduled on node and later
on, we have removed the label then the pod will be ignored and continue running on the same node.

2)Required During Scheduling Ignored During Execution (Hard Scheduling):

Means we are asking scheduler to check if the values/label is matching then schedule pod on the node. If not matched then pod will not be scheduled in other nodes.
Ignored During Execution means if the pod is already scheduled on node and later on we have removed the label then
the pod will be ignored and continue running on the same node.

43)K8s Volumes?

Volumes in k8s are used to store persistent data.
We have two types of pods one is stateful, other can be stateless.
Stateful: In simple terms stateful applications can be storing any data.
Stateless: In simple terms stateless applications will not be storing any data.

44)How many types of volumes are available?

1)Empty Dir:

==========

We create volume inside a pod to store data related to container.
If the container is killed for any reason, a new container will be created in the same pod and
the same volume will be attached to container.

2)Hostpath:

==========

We create volume on hostpath, means volume will be created outside pod.
If pod get deleted and a new pod is created then it can access the volume available on the host.

3)Amazon Elastic Block Storage:

==========

If we have multi node k8s cluster, then in this case we need to keep our volume outside the cluster.
If our pod is created on other node then the volume should also be moved to that node.

4)Amazon Elastic File Storage:

==========

If we have multinode k8s cluster, then in this case we need to keep our volume outside the cluster.
If our pod is created on other node then the volume should also be moved to that node.

45)Daemon Set in k8s?

Daemon Set is a type of workload that ensures that a specific pod is running on each node in a cluster.
It is useful for scenarios where you need to run a single instance of a pod on every node, such as log collection or monitoring agents.

46)Liveness Probe in k8s?

Liveness can use with pod configuration to check the health status of pod. If pod in not healthy for any reason, then Liveness will restart the pod.

47)Readiness Probe in k8s?

Readiness can use with pod configuration to check the if the application running in the pod is ready to serve traffic.

If pod in not healthy for any reason, then Readiness will remove the pod from endpoints and it will be in "not Ready" status.

48)What is ingress resource and ingress controller in k8s?

ingress resource:

An Ingress resource in Kubernetes is an API object that manages external access to services within a Kubernetes cluster. It acts as a traffic controller, routing incoming traffic to different services based on defined rules.

In simpler terms, it provides a way to expose HTTP and HTTPS routes from outside the cluster to services within the cluster. This can be useful for scenarios like load balancing, SSL termination, and virtual hosting.

The Ingress resource defines how traffic should be directed, including which services should receive the traffic and how that traffic should be handled (e.g., load balancing, SSL termination). These rules are typically specified using annotations or rules within the Ingress resource definition.

ingress controller:

Ingress controllers, such as NGINX Ingress Controller or Traefik, are responsible for implementing the rules defined in the Ingress resource and managing the routing of traffic accordingly.

An Ingress controller is a component of the Kubernetes ecosystem that manages inbound traffic to your Kubernetes services. It acts as an entry point for external users accessing your services running within the Kubernetes cluster.

When you deploy applications on Kubernetes, each service typically gets its own ClusterIP, which is accessible only within the cluster. However, if you want to expose these services to users outside the cluster, you need a way to route traffic from outside the cluster to the appropriate services inside. This is where Ingress controllers come in.

Ingress controllers typically work by reading Ingress resources defined within Kubernetes. An Ingress resource defines rules for routing external HTTP and HTTPS traffic to internal services based on hostnames or URL paths. The Ingress controller then configures the necessary load balancers, routers, or other

networking resources to implement these rules.

There are several implementations of Ingress controllers available, each with its own features and capabilities. Some popular examples include NGINX Ingress Controller, Traefik, and HAProxy Ingress. These controllers can provide features like SSL termination, path-based routing, load balancing, and more, making it easier to manage external access to your Kubernetes services.

49)horizonal pod autoscaling in k8s?

Horizontal Pod Autoscaling (HPA) in Kubernetes is a feature that automatically scales the number of pod replicas in a deployment, replica set, or stateful set based on observed metrics like CPU utilization or

custom metrics. HPA ensures that the desired number of pods are available to handle varying workloads efficiently.

Here's how HPA works in Kubernetes:

1.Metrics Collection: HPA collects metrics from the pods in the specified deployment, replica set, or stateful set. By default, HPA uses CPU utilization metrics, but it can also be configured to use custom metrics, such as request per second (RPS), memory usage, or any other relevant metric.
2.Metric Evaluation: HPA continuously evaluates the collected metrics against predefined thresholds. These thresholds define when scaling actions should be triggered. For example, you might set a threshold to trigger scaling when CPU utilization exceeds 50%.
3.Scaling Decisions: When the observed metrics exceed or fall below the configured thresholds, HPA makes scaling decisions. If the metrics indicate that more resources are needed, HPA will scale up by increasing the number of pod replicas. If the metrics indicate that fewer resources are needed, HPA will scale down by decreasing the number of pod replicas.
4.Adjustment: After making a scaling decision, HPA updates the desired number of replicas for the specified deployment, replica set, or stateful set. Kubernetes then takes care of creating or deleting pod replicas accordingly.
5.Feedback Loop: HPA continuously monitors the workload and adjusts the number of replicas based on changing demand. This creates a feedback loop that ensures the deployment can dynamically scale in response to fluctuations in traffic or resource usage.

HPA can be configured using either kubectl autoscale the command or by defining an resource in YAML or JSON format. Here's an example of an HPA

How do you troubleshoot issues in Kubernetes clusters?

Troubleshooting issues in Kubernetes clusters involves a systematic approach to identify and resolve problems efficiently. Here's a general guide on how to troubleshoot Kubernetes cluster issues

Understand the Problem:
Start by gathering information about the issue. Identify what symptoms are being observed, when the problem started, and any recent changes or deployments that might have caused the issue.

Check Cluster Health:
Use Kubernetes commands to check the overall health of the cluster:
kubectl cluster-info: Check the cluster's overall status and components.
kubectl get nodes: Verify the status of individual nodes.
kubectl get pods --all-namespaces: List all pods in all namespaces to check for any pod failures or issues.
kubectl get events --all-namespaces: View recent events in the cluster to identify any abnormalities or errors.

Inspect Pod Status:
If specific pods are experiencing issues, inspect their status and logs:
kubectl describe pod <pod-name>: Get detailed information about a specific pod, including its current status, events, and conditions.
kubectl logs <pod-name>: View the logs of a pod to check for errors, crashes, or unexpected behavior.

Check Node Resources:
Ensure that nodes have enough resources (CPU, memory, storage) to run pods:

kubectl describe node <node-name>: Check node capacity, allocatable resources, and system conditions.
Monitor resource usage using tools like Prometheus or Kubernetes Dashboard to detect resource bottlenecks or overutilization.

Examine Network Connectivity:

Verify network connectivity within the cluster and to external services:
kubectl exec -it <pod-name> -- /bin/sh: Access a pod's shell to test network connectivity within the cluster.
Use ping, curl, or traceroute commands to test connectivity to external services or endpoints.
Check Kubernetes network policies, service endpoints, and DNS resolution.

Review Configuration:

Check Kubernetes object configurations for errors or misconfigurations:
kubectl describe <resource-type> <resource-name>: Review configuration details of specific resources (e.g., deployments, services, ingresses).
Ensure that labels, selectors, ports, volumes, and environment variables are correctly configured.
Inspect Persistent Volumes (PVs) and Persistent Volume Claims (PVCs):
If using persistent storage, verify PVs and PVCs:
kubectl get pv: List persistent volumes and check their status, capacity, and access modes.
kubectl get pvc --all-namespaces: List persistent volume claims and verify their binding status and conditions.

Monitor Cluster Events and Logs:

Utilize Kubernetes monitoring and logging tools to track cluster events and logs:
Set up centralized logging with tools like Elasticsearch, Fluentd, and Kibana (EFK stack) or Prometheus and Grafana.
Use Kubernetes Dashboard or monitoring solutions to monitor metrics, alerts, and cluster health in real-time.
Review Application Code and Dependencies:
If the issue persists, review application code, dependencies, and container images:
Ensure that application code is compatible with Kubernetes and adheres to best practices.
Check container images for vulnerabilities, compatibility issues, or missing dependencies.
Consult Documentation and Community Resources:
Refer to Kubernetes documentation, forums, GitHub issues, and community resources for troubleshooting guides, known issues, and solutions.
Engage with Kubernetes user groups, forums, or support channels for assistance from the community or Kubernetes experts.

what are different types of status for pods & types of errors for pods

In Kubernetes, pods can have different statuses depending on their current state of execution. Here are the different types of pod statuses and common types of errors that pods may encounter:

Pod Statuses:

Pending:

The pod has been accepted by the Kubernetes system, but its containers are not yet created.

Reasons for pending status:

Insufficient resources (CPU, memory, storage) on the node.
ImagePullBackOff: Unable to pull the container image from the registry.
Pod scheduling constraints (affinity, node Selector) not satisfied.

Running:

All containers in the pod are created and running.
The pod's entry point command has started executing.
Succeeded:
All containers in the pod have terminated successfully.
The container's exit code is 0.
Failed:
One or more containers in the pod have terminated with a non-zero exit code.
Kubernetes automatically restarts failed containers unless specified otherwise.
Unknown:
The status of the pod cannot be determined, typically due to communication issues with the Kubernetes API server.

Common Types of Pod Errors:

CrashLoopBackOff:

The pod's container keeps crashing and restarting in a loop.
Possible causes: misconfigured startup command, missing dependencies, or application crashes.

ImagePullBackOff:

The pod is unable to pull the specified container image from the container registry.
Causes: incorrect image name, authentication issues with the registry, or network connectivity problems.

OOMKilled (Out Of Memory Killed):

One or more containers in the pod exceeded the allocated memory limit and were killed by the system.
This can happen if the application consumes more memory than allocated or if memory limits are not set appropriately.

ErrImagePull:

Kubernetes is unable to pull the container image specified in the pod's definition.
Reasons: incorrect image name or tag, image not available in the specified repository, or authentication issues.

ErrImageNeverPull:

The pod's image pull policy is set to Never, but the specified image is not present on the node.
This error occurs when the pod's imagePullPolicy does not match the actual availability of the image.

CreateContainerConfigError:

Kubernetes encountered an error while creating the container configuration for the pod.
Causes: misconfigured container settings, invalid volume mounts, or incompatible container runtime settings.

ContainerCreating:

The pod is in the process of creating its containers but has not yet completed the process.
This status is transient and should transition to Running or another appropriate status.

NodeAffinity/NodeSelector errors:

Pods may remain in a Pending state if their scheduling constraints (node affinity, node selector) cannot be satisfied due to node constraints or labels.

These are some common statuses and errors encountered by pods in Kubernetes. Understanding these statuses and errors can help in diagnosing and resolving issues effectively within Kubernetes clusters.

what is crash loopback error why it occurs and how you resolve it?

The CrashLoopBackOff error in Kubernetes occurs when a pod repeatedly crashes immediately after starting, causing Kubernetes to restart the pod in a loop. This error indicates that the container within the pod is not able to start successfully, leading to continuous restarts.

Causes of CrashLoopBackOff Error:

Misconfigured Startup Command:

The container's entry point command or startup script may be misconfigured, causing the container to exit immediately after starting.

Missing Dependencies:

The application or container may be missing necessary dependencies or libraries required for proper execution, leading to crashes.

Application Crashes:

Issues within the application code or runtime environment may cause the application to crash upon startup, triggering the CrashLoopBackOff error.

How to Resolve CrashLoopBackOff Error:

Check Container Logs:

Use kubectl logs <pod-name> to view the logs of the container that is crashing. Look for error messages, stack traces, or any indications of why the container is failing to start.

Review Container Configuration:

Verify the container's configuration, including command, arguments, environment variables, volume mounts, and resource limits.

Ensure that the startup command or entry point script is correctly specified and that all required dependencies are available within the container.

Check Image and Tags:

Confirm that the container image specified in the pod's definition exists in the container registry.

Check the image tags and versions to ensure that the correct and compatible image is being pulled.

Increase Resource Limits:

If the container is crashing due to resource constraints (CPU, memory), consider increasing the resource limits allocated to the pod.

Update the pod's resource requests and limits in the pod's YAML definition file.

Verify Network Dependencies:

If the application relies on network dependencies (e.g., database connection), ensure that network connectivity is established and configurations are correct.

Test Locally:

Run the container image locally or in a development environment to test its startup behavior and identify any issues early on.

Update Application Code:

If the CrashLoopBackOff error is caused by application code issues, update and debug the application code to fix bugs or errors.

Use Liveness and Readiness Probes:

Implement liveness and readiness probes in the pod's definition to monitor the health of the container and avoid continuous restarts.

Configure probes to check specific endpoints or conditions within the container to determine its health status.

Monitor and Analyze Logs:

Use Kubernetes monitoring tools, logging solutions, or external logging services to monitor pod behavior, analyze logs, and track container restarts.

Helm charts

Helm is a package manager for Kubernetes that simplifies the deployment and management of applications on Kubernetes clusters. Helm uses charts, which are packages of pre-configured Kubernetes resources, to define, install, and upgrade Kubernetes applications.

Here's an overview of Helm charts and how they are used:

Chart Structure:

Chart Repository:

Helm charts are typically stored in chart repositories, which can be public or private repositories accessible via URLs.

Public repositories like the official Helm Hub (https://hub.helm.sh/) host a wide range of charts for popular applications and services.

Organizations often set up private repositories to store proprietary or customized charts securely.

Chart Commands:

helm create <chart-name>: Generates a new Helm chart with default directory structure and files.

helm install <chart-name> <release-name>: Installs a Helm chart into the Kubernetes cluster, creating a new release with a specified name.

helm upgrade <release-name> <chart-name>: Upgrades an existing Helm release to a new version of the chart.

helm uninstall <release-name>: Uninstalls a Helm release, removing the deployed resources from the Kubernetes cluster.

Values Overrides:

During deployment, you can override default values in the values.yaml file using the --set flag or by providing a custom values file with --values flag.

For example, helm install myapp ./myapp-chart --set image.tag=v1.2.3 sets the image tag to v1.2.3 during installation.

Chart Versioning and Updates:

Helm charts follow semantic versioning, allowing for versioned releases and upgrades.

Chart authors release new versions with bug fixes, feature enhancements, or security updates, which can be applied using helm upgrade.

Helm Plugins:

Helm supports plugins that extend its functionality, such as linting charts (helm lint), packaging charts (helm package), and managing repositories (helm repo).

Overall, Helm charts simplify the process of packaging, deploying, and managing Kubernetes applications by providing a standardized and reusable way to define application configurations and dependencies. They promote infrastructure as code practices and streamline application lifecycle management on Kubernetes clusters.

Prometheus and Grafana

What is Prometheus?
Prometheus is an open-source monitoring and alerting toolkit designed for collecting and querying time-series data.

What is Grafana?
Grafana is a multi-platform open-source analytics and visualization platform used to create rich dashboards for monitoring and analyzing metrics.

How does Prometheus collect data?
Prometheus uses a pull-based model to scrape metrics from configured targets (e.g., applications, servers) at regular intervals.

What is PromQL?
PromQL is the query language used in Prometheus for querying and analyzing collected metrics.

How does Grafana integrate with Prometheus?
Grafana integrates with Prometheus as a data source, allowing users to visualize Prometheus metrics in Grafana dashboards.

What are some key features of Grafana dashboards?
Key features include customizable panels, data source integration, alerting, templating, plugins/extensions, and role-based access control.

What types of visualizations can Grafana dashboards include?
Grafana dashboards can include graphs, gauges, tables, histograms, heatmaps, logs, and more.

How does Prometheus handle alerting?
Prometheus supports alerting based on predefined rules and thresholds, sending alerts to external alerting platforms or receivers.

What is the purpose of Grafana alerting?
Grafana alerting allows users to define alert rules based on metric queries and thresholds, sending notifications via email, Slack, etc.

What is the difference between Prometheus and Grafana?
Prometheus is a monitoring and alerting toolkit for data collection, storage, and querying, while Grafana is a visualization platform for creating dashboards.

How can you create a new dashboard in Grafana?
In Grafana, go to the dashboard menu, click on "New Dashboard," and then add panels to the dashboard using the panel editor.

What is a data source in Grafana?
A data source in Grafana represents a backend database or service from which metrics or data are fetched and displayed in dashboards.

Can Grafana visualize data from sources other than Prometheus?
Yes, Grafana supports integration with various data sources such as InfluxDB, Elasticsearch, MySQL, and more.

What are the benefits of using Grafana with Prometheus?
Grafana provides advanced visualization capabilities, dashboard customization, alerting, and integration with multiple data sources, enhancing Prometheus's monitoring capabilities.

How can you create a custom dashboard panel in Grafana?
To create a custom panel, select the panel type (e.g., Graph, Table), configure data source and queries, customize visualization settings, and add panel options.

What is the role of templates in Grafana dashboards?
Templates in Grafana allow users to create dynamic dashboards by using variables for filtering, grouping, and aggregating data based on user input.

How does Grafana handle authentication and access control?
Grafana supports role-based access control (RBAC) to manage user permissions, access levels, and dashboard sharing within organizations.

Can Grafana visualize logs in addition to metrics?
Yes, Grafana can visualize logs by integrating with log aggregation platforms like Loki, Elasticsearch, or Fluentd.

How can you share Grafana dashboards with team members?
Grafana allows dashboard sharing through export/import of JSON configurations or using built-in sharing features to share links or embed dashboards.

What are some common use cases for using Grafana and Prometheus together?
Common use cases include monitoring infrastructure metrics, application performance, service health, system resources, and cloud-native environments.

Q: How do you configure Prometheus to scrape metrics from a target?
A: In Prometheus, you define scrape configurations in the prometheus.yml file. Each job specifies the target's endpoint, metrics path, and scraping interval.

Q: What is the purpose of retention policies in Prometheus?
A: Retention policies in Prometheus define how long metrics data should be retained. They help manage storage resources and data retention periods.

Q: How can you set up alerting in Prometheus for a specific metric exceeding a threshold?
A: You define alerting rules in Prometheus using the alert.rules file. For example, you can create a rule to alert when CPU usage exceeds a certain threshold for a duration.

Q: Explain how to integrate Grafana with Prometheus as a data source.
A: In Grafana, navigate to Configuration > Data Sources > Add data source. Choose Prometheus and enter the Prometheus server URL. Configure authentication if needed and save the data source.

Q: What is a Grafana dashboard variable, and how can it be useful?
A: A Grafana dashboard variable is a dynamic placeholder that allows users to change data displayed in panels. It can be used for filtering, grouping, or switching between different data sets.

Q: How do you create an alert notification channel in Grafana for receiving alerts from Prometheus?
A: In Grafana, go to Configuration > Notification channels > Add channel. Choose the notification type (e.g., Email, Slack), configure the details, and test the notification to ensure it works.

Q: What are the benefits of using Grafana annotations in dashboards?
A: Grafana annotations provide contextual information about events or changes in metrics data. They can be used to mark deployments, incidents, or other significant events on dashboards.

Q: Explain how to visualize a custom metric from Prometheus in a Grafana panel.
A: In Grafana, create a new panel and select Prometheus as the data source. Write a PromQL query to fetch the custom metric, configure visualization options (e.g., graph type, axis labels), and save the panel.

Q: How can you scale Grafana deployments for high availability and performance?
A: To scale Grafana, deploy multiple instances behind a load balancer for redundancy. Use shared storage for dashboard configurations and ensure database performance for large-scale deployments.

Q: What is the role of dashboards in Grafana, and how do they help in monitoring and analysis?

A: Dashboards in Grafana are visual representations of metrics data from data sources like Prometheus. They help users monitor system health, analyze performance trends, detect anomalies, and make data-driven decisions.

WAF

Q: What is AWS WAF, and what does it protect against?

A: AWS WAF is a web application firewall that protects web applications from common web exploits such as SQL injection, cross-site scripting (XSS), and other OWASP top 10 vulnerabilities.

Q: How do you deploy AWS WAF for a web application?

A: To deploy AWS WAF, you create a web ACL (Access Control List) in the AWS WAF console, define rules to filter and block malicious traffic, and associate the ACL with your application's CloudFront distribution or Application Load Balancer.

Q: What are the benefits of using AWS WAF with AWS CloudFront?

A: AWS WAF integrated with AWS CloudFront provides scalable and distributed protection against DDoS attacks and helps secure web applications deployed on AWS infrastructure.

Q: Can AWS WAF protect against bot traffic and automated attacks?

A: Yes, AWS WAF includes bot mitigation features to detect and block malicious bot traffic, such as web scrapers, content scrapers, and automated attacks targeting web applications.

Q: How does AWS WAF handle rate-based rules for protection against rate-based attacks?

A: AWS WAF's rate-based rules allow you to set thresholds for incoming requests per IP address or per resource. If the threshold is exceeded, AWS WAF can block or rate-limit the requests.

Q: What is the difference between AWS WAF's rule groups and web ACLs?

A: Rule groups in AWS WAF contain sets of rules that define conditions for allowing or blocking traffic. Web ACLs are collections of rule groups and rule statements that define the overall security policies for web applications.

Q: How can you monitor and analyze AWS WAF logs for security insights?

A: AWS WAF logs can be sent to Amazon CloudWatch Logs or Amazon Kinesis Data Firehose for real-time monitoring and analysis. You can create custom metrics, alerts, and dashboards based on WAF logs.

Q: What are managed rule sets in AWS WAF, and how do they enhance security?

A: Managed rule sets in AWS WAF are pre-configured rule sets provided by AWS or third-party vendors. They offer ready-to-use protection against common threats and vulnerabilities, reducing the time to deploy and manage custom rules.

Q: Can AWS WAF be integrated with AWS Lambda for automated response to security incidents?

A: Yes, AWS WAF can trigger AWS Lambda functions based on rule matches or security events. This allows for automated response actions such as blocking IP addresses, logging events, or sending notifications.

Q: How does AWS WAF handle geographic-based blocking or allowlisting of traffic?

A: AWS WAF supports geographic match conditions that allow you to block or allow traffic based on the geographic location of the request origin, using IP address ranges associated with specific countries or regions.

Q: What are the best practices for managing AWS WAF rules and policies?
A: Best practices include regularly reviewing and updating rule sets, monitoring traffic patterns and rule effectiveness, leveraging managed rule sets, implementing rate-based rules for DDoS protection, and integrating with logging and monitoring services.

Q: Can AWS WAF be used in a multi-tiered application architecture?
A: Yes, AWS WAF can be deployed in front of multiple tiers of web applications, APIs, or microservices to provide consistent security policies and protection across the application stack.

Q: How does AWS WAF handle HTTPS traffic and SSL/TLS termination?
A: AWS WAF can inspect HTTPS traffic by integrating with AWS CloudFront or Application Load Balancers that terminate SSL/TLS connections. It can analyze HTTP headers, request payloads, and other HTTPS attributes for security enforcement.

Q: What options are available for customizing AWS WAF rules for specific application requirements?
A: AWS WAF allows for the creation of custom rule statements using regular expressions, string matching, IP match conditions, and advanced filtering criteria based on HTTP request attributes.

Q: How does AWS WAF handle IP reputation lists and threat intelligence feeds?
A: AWS WAF can integrate with IP reputation lists and threat intelligence feeds from third-party providers to automatically block traffic from known malicious IP addresses, botnets, or blacklisted sources.

Q: How can AWS WAF be integrated with AWS Shield for comprehensive DDoS protection?
A: AWS WAF integrates seamlessly with AWS Shield Standard or AWS Shield Advanced for additional DDoS protection at the network and application layers. AWS Shield Advanced includes enhanced protection, 24/7 support, and DDoS response services.

Q: What are the common use cases for AWS WAF in cloud-based applications?
A: Common use cases include protecting web applications, APIs, e-commerce platforms, content management systems, public-facing websites, and microservices architectures from web-based attacks and vulnerabilities.

Q: How can you automate AWS WAF deployments and rule updates using AWS CloudFormation or AWS CLI?
A: AWS CloudFormation templates and AWS CLI commands can be used to automate the provisioning of AWS WAF resources, including web ACLs, rule groups, IP sets, and rule statements, allowing for consistent and repeatable deployments.

Q: What are the options for integrating AWS WAF with AWS services such as Amazon S3, Amazon API Gateway, or AWS Lambda?
A: AWS WAF can be integrated with Amazon S3 for protecting static websites, Amazon API Gateway for API endpoint security, and AWS Lambda for serverless application security. Integration options include AWS Managed Rules, custom rules, and Lambda@Edge for edge-based processing.

Recent issues:

1)We were able to see lot of tickets from development teams that they are not able to see the logs of production server on Splunk centralized server.
We checked if the Splunk forwarders are running or not on that servers and found that these forwarders were down for some reason.

So we have created a bash script to check the status of Splunk forwarder for every5 mins and alert the team if these forwarders are down, so that we can forecast the issue and make sure Splunk forwarders are up and running always
2)In Jenkins we can able to see lot of issues with connectivity, this will happen when there is a change in the systems, I mean bcz of patching activities...

And also pipelines will get failed bcz of any missing dependencies or any kind of missing artifacts.

Recent challenge/achievment:

1)Used s3 bucket to store terraform statefiles and also used dymanod db locking to protect state files which not be used by multiple users at the same time.

Top 127 scenario-based interview questions

1. What are all the types of applications you have deployed?

Answer: Web applications (e.g., Java Spring Boot, Node.js), microservices architectures, databases (e.g., MySQL, MongoDB), message brokers (e.g., Apache Kafka), monitoring tools (e.g., Prometheus, Grafana), CI/CD pipelines (e.g., Jenkins), and containerized applications using Docker and Kubernetes.

2. How have you injected the secrets in ConfigMaps?

Answer: Secrets should not be injected in ConfigMaps as ConfigMaps are not designed for sensitive data. Instead, Kubernetes Secrets should be used. Secrets can be injected into pods via environment variables or mounted as files.

3. How do you find which pod is taking more system resources across nodes using kubectl?

Answer: Use **kubectl top pod --all-namespaces** to list resource usage by pods. Combine it with **kubectl describe pod <pod-name>** to get detailed resource usage.

4. How do you know which worker node is consuming more resources across the clusters using kubectl?

Answer: Use kubectl top nodes to see resource consumption across nodes. This will show CPU and memory usage on each node.

5. What are the steps for configuring Prometheus and Grafana for monitoring Kubernetes clusters?

Answer: 1. Deploy Prometheus using Helm or a custom YAML configuration.
2. Set up Kubernetes service discovery for Prometheus.
3. Deploy Grafana and configure it to use Prometheus as a data source.
4. Import Kubernetes monitoring dashboards in Grafana.
5. Set up alerting rules in Prometheus as needed.

6. If 20 pods are running, how do you visualize the metrics of these pods in Grafana?

Answer: Configure Grafana to use Prometheus as the data source. Use existing Kubernetes dashboards or create custom dashboards to visualize metrics for all pods.

7. What is Apache Kafka?

Answer: Apache Kafka is a distributed event streaming platform used for building real-time data pipelines and streaming applications. It's designed to handle large volumes of data in a distributed, fault-tolerant manner.

8. How do you set up a Docker Hub private registry and integrate it with a CI/CD pipeline? What is the procedure?

Answer: 1. Create a private repository on Docker Hub.

2. Configure CI/CD pipeline to authenticate with Docker Hub using credentials.

3. Build Docker images in CI pipeline and push them to the private repository.

4. Pull images from the private registry during the CD process.

9. What is the difference between a hard link and a soft link?

Answer: ○

Hard Link: A direct reference to the file's data on the disk. Deleting the original file does not affect the hard link.

○ **Soft Link (Symbolic Link):** A pointer to the original file's path. If the original file is deleted, the soft link becomes broken.

10. What is the use of the break command in shell scripting? In what scenarios have you used it?

Answer: The `break` command is used to exit a loop prematurely. It is commonly used when a specific condition is met, and you no longer need to continue the loop.

11. How do you count the number of "devops" words in 15 HTML files?

To count the number of times the word "devops" appears across 15 HTML files, you can use the following shell command:

grep -o -i "devops" *.html | wc -l

Explanation:

grep -o -i "devops" *.html: Searches for the word "devops" (case-insensitive due to -i) in all HTML files and outputs each occurrence on a new line (-o).

wc -l: Counts the number of lines output by grep, which corresponds to the number of occurrences of "devops".

12. What is the terraform taint command?

The terraform taint command is used to manually mark a specific resource for recreation during the next terraform apply. When a resource is tainted, Terraform will destroy it and create a new instance of it on the next apply. This is useful when you know a resource needs to be replaced but Terraform hasn't automatically determined that it should be.

Syntax: terraform taint <resource_address>

Example: terraform taint aws_instance.example

13. What are the possible ways to secure a state file in Terraform?

1. **Encrypt the State File**: Store the state file in an encrypted format using tools like AWS S3 bucket encryption, Azure Blob encryption, or Google Cloud Storage encryption.

2. **Use Remote Backends with Security Controls**: Use a remote backend like AWS S3 with IAM roles and policies, HashiCorp Consul with ACLs, or Terraform Cloud, which manages access control and encryption.

3. **Enable Versioning:** Use versioning on the state file storage to recover from any unintended changes or deletions.

4**. Access Control:** Restrict access to the state file using IAM policies, RBAC (Role-Based Access Control), or similar mechanisms.
5. **Use Backend Locking:** Use state locking mechanisms provided by backends like S3 with DynamoDB or Consul to prevent concurrent operations from corrupting the state file.

14. If you provision 100 servers and someone deletes 50 VMs manually, what happens if you apply the terraform apply command?

When you run terraform apply, Terraform will compare the state file with the actual infrastructure. It will notice that 50 VMs are missing and will attempt to recreate those missing VMs to match the state defined in your configuration. The end result will be 100 VMs again.

15. What is the syntax for for_each in Terraform?

The for_each meta-argument in Terraform allows you to create multiple instances of a resource or module based on the items in a map or set.

The syntax is as follows: resource

```
"aws_instance" "example" {
 for_each = var.instances
 ami = each.value["ami"]
 instance_type = each.value["instance_type"]

 tags = {
   Name = each.key
  }
   }
```

In this example, var. instances is a map, and each. key refers to the current key in the map, while each. value refers to the associated value.

16. What are the advantages and disadvantages of multi-stage builds in Docker?

Advantages:
Smaller Image Size: By separating the build environment from the runtime environment, only the necessary components are included in the final image, leading to smaller image sizes.
Improved Security: Reduces the attack surface by excluding build tools and other unnecessary components from the final image.
Better Caching: Allows for more efficient use of Docker's caching mechanism, potentially speeding up the build process.
Separation of Concerns: Different stages can focus on specific tasks, making the Docker file more organized and easier to maintain.

Disadvantages:
Complexity: Multi-stage Docker files can be more complex and harder to understand, especially for those new to Docker.
Longer Build Times: In some cases, the use of multiple stages may lead to longer build times due to additional steps and transitions between stages.

Troubleshooting: Debugging and troubleshooting can be more challenging because intermediate stages are not preserved by default.

17. How do you deploy containers on different hosts, not the same host, within a Docker cluster?

To deploy containers on different hosts within a Docker cluster, you can use Docker Swarm or Kubernetes:

Docker Swarm: Use Docker Swarm's built-in orchestration capabilities. When deploying a service, Swarm will automatically distribute the containers across different nodes in the cluster. You can influence this behavior using placement constraints.

Example:

```
docker service create --name myservice --replicas 2 --constraint 'node.role == worker' myimage
```

Kubernetes: Use Kubernetes, which will schedule pods on different nodes based on the available resources and any specified node selectors or affinities.

18. If you have a Docker Compose setup, how do you deploy the web container on one host and the DB container on another host?

To deploy the web container on one host and the DB container on another using Docker Compose, you can:

1. Use Docker Swarm: Convert the Compose file into a stack file, and deploy it with docker stack deploy. Docker Swarm will distribute services across nodes.
2. Manual Placement: Use placement constraints in your docker-compose.yml file to specify which containers should run on which nodes.

```
version: '3.7'
services:
web:
image: my-web-image
deploy:
placement:
constraints: [node.hostname == web-node]
db:
image: my-db-image
deploy:
placement:
constraints: [node.hostname == db-node]
```

19. What is the difference between bridge networking and host networking in Docker?

Bridge Networking: The default Docker network driver. Containers connected to the same bridge network can communicate with each other. Each container gets its own IP address and is isolated from the host network. You can expose ports to the host using the -p option.

Host Networking: The container shares the host's network stack, meaning it doesn't get its own IP address, and the container's network is the same as the host's network. This can lead to performance improvements but reduces isolation between the host and the container.

20. How do you resolve merge conflicts?

To resolve merge conflicts, follow these steps:

1.Identify the Conflict: Git will mark the conflicting areas in the files with conflict markers (<<<<<<<, =======, and >>>>>>>).
2. Manually Resolve: Open the conflicting file and decide how to combine the changes. Remove the conflict markers and modify the content to resolve the conflict.
3. Stage the Resolved Files: Once resolved, stage the files using git add.
4. Commit the Changes: Commit the resolved conflicts with git commit. If you were in the middle of a merge, this will complete the merge process.
5. Test: Run tests to ensure that the merge didn't introduce any issues.

Example:
git add <resolved-files>
git commit -m "Resolved merge conflicts in X, Y, Z files"

21. What command do you use to change the existing commit message?
To change the most recent commit message, you can use the following Git command:

git commit --amend -m "New commit message"

If you have already pushed the commit to a remote repository, you will need to force push the changes:

git push --force

22.What is session affinity?
Session affinity, also known as sticky sessions, is a concept in load balancing where requests from a particular user are consistently directed to the same server (or pod) in a multi-server environment. This ensures that the user's session data, which might be stored locally on the server, remains accessible throughout the session.

23.What is pod affinity and its use case?
Pod affinity is a feature in Kubernetes that allows you to specify rules for scheduling pods to run on nodes that have other specified pods running on them. This can be useful when you want certain pods to be located together due to factors like data locality, network latency, or shared resources.
Use Case: An application where the frontend and backend services communicate frequently might use pod affinity to ensure that both are scheduled on the same node to reduce network latency.

24. What is the difference between pod affinity and pod anti-affinity?
Pod Affinity: Ensures that pods are scheduled on the same node or in proximity to each other.
Pod Anti-Affinity: Ensures that pods are not scheduled on the same node or are placed far apart from each other.
Example Use Case:
Pod Affinity: Running frontend and backend on the same node for low-latency communication.
Pod Anti-Affinity: Ensuring replicas of the same application are spread across different nodes to increase availability and fault tolerance.

25. What are readiness and liveness probes?

Readiness Probe: Used to determine when a pod is ready to start accepting traffic. If the readiness probe fails, the pod will be removed from the service endpoints, ensuring it does not receive traffic until it's ready.

Liveness Probe: Used to determine if a pod is still running. If the liveness probe fails, Kubernetes will restart the pod, assuming it's in a failed state.

26. Write a simple Groovy pipeline for a Java Spring Boot application that waits for user input for approval to move to the next stage, with stages for checkout, build, push, and deploy.

```
pipeline {
agent any
stages {
stage('Checkout') {
steps {
git 'https://github.com/your-repo/java-spring-boot-app.git'
}
}
stage('Build') {
steps {
sh './mvnw clean package'
}
}
stage('Push') {
steps {
sh 'docker build -t your-docker-repo/java-spring-boot-app .'
sh 'docker push your-docker-repo/java-spring-boot-app'
}
}
stage('Approval') {
steps {
input 'Do you want to deploy to production?'
}
}
stage('Deploy') {
steps {
sh 'kubectl apply -f deployment.yaml'
}
}
}
}
```

27. How do you export test reports in Jenkins?

To export test reports in Jenkins:

1. Ensure that your tests generate reports in a standard format like JUnit XML or HTML.
2. Use the Publish JUnit test result report post-build action to archive test reports.
3. You can also use the Archive the artifacts post-build action to store other types of reports.
4. The test reports can then be accessed and downloaded from the Jenkins build page.

28. If 5 pods are running, how do you scale the number of pods to 10 using the command line in Kubernetes?

To scale the number of pods from **5 to 10,** use the following command:

kubectl scale --replicas=10 deployment/<your-deployment-name>

29. Can you explain the usage of the terraform import command?

The terraform import command is used to import existing infrastructure resources into Terraform's state file, allowing Terraform to manage them. This is particularly useful when you want to bring existing infrastructure under Terraform management without having to recreate resources.

terraform import <resource_type>.<resource_name> <resource_id>

Example:
terraform import aws_instance.my_instance i-1234567890abcdef0

30. In AWS, where do you store the state file, and how do you manage it?

In AWS, Terraform state files are typically stored in an S3 bucket, and the state can be managed using a combination of S3 and DynamoDB for locking.

Example Configuration: terraform {
backend "s3" {
bucket = "my-terraform-state-bucket"
key = "path/to/my/terraform.tfstate"
region = "us-west-2"
dynamodb_table = "terraform-lock-table"
encrypt = true
}
}

Management:
S3: Stores the state file securely, supports versioning, and can be encrypted.
DynamoDB: Provides locking to prevent multiple simultaneous operations on the same state file, avoiding conflicts.

This setup ensures that your Terraform state is stored securely and is protected from simultaneous access issues.

31. What is the biggest issue you have faced with Terraform, and how did you resolve it?

One significant issue I faced was managing Terraform state when multiple teams were working on the same infrastructure. This was resolved by organizing the infrastructure into separate modules and using remote state management with proper locking mechanisms.

32. What are the types of storage accounts in AWS S3?

In AWS S3, the different storage classes include:
- **S3 Standard**

- **S3 Intelligent-Tiering**
- **S3 Standard-IA (Infrequent Access)**
- **S3 One Zone-IA**
- **S3 Glacier**
- **S3 Glacier Deep Archive**

33. Are you familiar with lifecycle management in S3 buckets? How do you set up lifecycle policies?

Yes, lifecycle management in S3 allows you to define rules to transition objects between different storage classes or delete them after a certain period. Lifecycle policies can be set up using the S3 Management Console, AWS CLI, or Terraform by specifying the transitions and expiration actions in a JSON configuration file.

34. What are the differences between load balancers, and why do we need them?

Load balancers distribute incoming network traffic across multiple servers.

The main types are:

- **Application Load Balancer (ALB):** Operates at the application layer (Layer 7), and is used for HTTP/HTTPS traffic with advanced routing capabilities.
- **Network Load Balancer (NLB):** Operates at the transport layer (Layer 4) and is used for ultra-low latency TCP/UDP traffic.
- **Classic Load Balancer (CLB): Supports** both Layer 4 and Layer 7, but is now mostly deprecated in favor of ALB and NLB.

Load balancers improve fault tolerance, scalability, and ensure high availability.

35. Have you worked with Auto Scaling Groups (ASG)?

Yes, I have worked with ASGs to automatically scale the number of instances in response to demand. ASGs are configured with policies that adjust the desired capacity based on metrics such as CPU utilization, helping to maintain application performance and optimize costs.

36. Can you write a simple Dockerfile?

Yes, here is a simple example of a Dockerfile for a Node.js application: Dockerfile

```
FROM node:14
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

37. If you want to expose your application to the public internet or access your application within a cluster, how would you do that in Kubernetes?

To expose your application to the public internet, you can use a Kubernetes Service of type LoadBalancer or NodePort. For internal access within the cluster, you can use a
ClusterIP service. Additionally, you might use an Ingress controller for more advanced routing.

38. Why do we need a ConfigMap in Kubernetes?

A ConfigMap is used to store non-confidential configuration data in key-value pairs. It allows you to decouple configuration artifacts from image content, enabling you to modify application settings without rebuilding your container images.

39. Which AWS services do you consider when setting up a CI/CD pipeline for a microservices application?

For a CI/CD pipeline, I would consider using:

- **AWS CodeCommit** for source control.
- **AWS CodeBuild** for building and testing.
- **AWS CodeDeploy** or Amazon EKS for deployment.
- **AWS CodePipeline** to orchestrate the CI/CD process.
- **Amazon S3** for artifact storage.
- **AWS Lambda** for any custom automation tasks.

40. On a day with unusually high traffic for an e-commerce application, how would you, as a cloud engineer, manage the current setup to handle the load smoothly?

I would ensure that Auto Scaling Groups are properly configured to handle the increased demand by automatically adding more instances. I'd also verify that the load balancers are evenly distributing traffic and consider enabling caching (e.g., using Amazon CloudFront) to reduce the load on backend servers. Monitoring tools like CloudWatch would be used to track performance metrics and adjust resources in real-time.

Strategies:

- Auto Scaling: Scale up instances automatically to handle the increased load.
- Caching: Use a caching layer to reduce the load on your application servers.
- Load Balancing: Distribute traffic evenly across available instances.
- Database Optimization: Ensure your database is properly configured and optimized for performance.
- Monitoring: Closely monitor system metrics to identify bottlenecks and adjust resources accordingly.

41. If traffic is currently handled on a single instance, how would you upgrade for high availability in AWS?

To upgrade for high availability, I would:

- Deploy multiple instances across different Availability Zones (AZs) using an Auto Scaling Group.
- Set up a Load Balancer (ALB or NLB) to distribute traffic across these instances.
- Configure health checks to ensure traffic is only routed to healthy instances.
- Use Multi-AZ deployments for databases like RDS to ensure data availability.

42. When auto-scaling instances, how do you manage the backend RDS database?

To manage the backend RDS database during auto-scaling:

- Enable Multi-AZ for high availability and automatic failover.
- Use RDS Read Replicas to handle read-heavy traffic, reducing the load on the primary database.
- Scale RDS vertically (instance size) or horizontally (read replicas) based on the database workload.
- Monitor performance using Amazon CloudWatch and adjust as necessary.

43. Have you ever set up cross-account access for S3? For example, if the QA team needs access to the production database.

Yes, I've set up cross-account access by:

- Creating an IAM role in the production account with the necessary S3 permissions.
- Establishing a trust relationship to allow the QA account to assume that role.
- Using S3 bucket policies to grant access from the QA account.
- QA team members can then assume the role using AWS STS (Security Token Service) to access the production S3 bucket.

44. How can an S3 account in Account A access an S3 account in Account B?

Account A can access Account B's S3 bucket by:

- Setting up a bucket policy in Account B that grants **Account A the necessary permissions.**
- Creating an IAM role in Account B with permissions for S3 and allowing Account A to assume that role via a trust policy.
- Using AWS STS to assume the role from Account A and access the S3 bucket in Account B.

45. Can you differentiate between IAM policies and IAM roles?

IAM Policies: These are sets of permissions attached to users, groups, or roles, defining what actions are allowed or denied.

IAM Roles: These are identities with specific permissions that can be assumed by entities like users, applications, or AWS services. Roles are often used for temporary access or cross-account access.

46. Can you explain the STS assume role policy?

The STS (Security Token Service) AssumeRole policy allows a user or service to assume a role in a different account or within the same account. This provides temporary security credentials with the permissions associated with the assumed role, enabling cross-account access or delegation of permissions.

47. Have you experienced any challenging issues or incidents in your project? How did you and your team identify and resolve them?

Yes, one challenge was a sudden traffic spike causing performance degradation. We identified the issue using CloudWatch metrics and logs, pinpointing the bottleneck in the database. The resolution involved scaling the database vertically and adding read replicas to distribute the load, along with optimizing slow-running queries.

48. What is the difference between CMD and ENTRYPOINT in Docker?

CMD: Provides default arguments for the entrypoint or the command to run if no other command is provided.

ENTRYPOINT: Defines the executable that will always run, with CMD as its default arguments. ENTRYPOINT is useful when you want your container to behave like a specific executable.

Example: ENTRYPOINT ["python", "app.py"] ensures app.py is always executed, while CMD allows passing different arguments.

49. Have you ever managed an application single-handedly?

Yes, I have managed applications single-handedly, handling tasks such as deployment, monitoring, troubleshooting, and scaling. This involved setting up the infrastructure, CI/CD pipelines, and ensuring high availability and security.

50. What are the benefits of Infrastructure as Code (IaC)?

Consistency: Ensures consistent environments by automating the provisioning process.
Version Control: Infrastructure can be versioned and tracked, enabling rollbacks and audits.
Automation: Reduces manual intervention, minimizing errors and speeding up deployments.
Scalability: Allows easy scaling and management of resources through scripts.
Collaboration: Teams can collaborate more effectively using code reviews and version control systems.

51. What are the different ways to create infrastructure as code?

Terraform: A popular open-source tool that allows you to define infrastructure as code using a declarative **configuration language**. It works with various cloud providers.
AWS CloudFormation: A service provided by AWS that enables you to define AWS resources using JSON or YAML templates.
Azure Resource Manager (ARM) Templates: Azure's solution for infrastructure as code, allowing you to define resources in JSON.
Google Cloud Deployment Manager: Google's infrastructure as code tool that uses YAML to define resources.
Ansible: Although primarily a configuration management tool, it can also be used to provision infrastructure using playbooks written in YAML.

Pulumi: A modern infrastructure as code tool that supports multiple programming languages like Python, JavaScript, and Go.

52. What is the difference between public and private networking?

Public Networking: Refers to networks that are accessible from the internet. Public IP addresses are used, and resources are exposed to external access.
Private Networking: Refers to networks that are isolated from the public internet. Resources within a private network communicate with each other securely using private IP addresses, and access from outside is typically restricted.

53. What is a Docker registry and why do we need it?

Docker Registry: A storage and distribution system for Docker images. It allows you to store, share, and manage Docker container images. Docker Hub is a popular public registry, but you can also set up private registries.
Why We Need It: Docker registries allow teams to version, share, and deploy Docker images easily. They support CI/CD pipelines by enabling automated builds and deployments.

54. What is a secrets manager?

Secrets Manager: A tool or service that securely stores and manages sensitive information such as API keys, passwords, certificates, and tokens. Examples include AWS Secrets Manager, HashiCorp Vault, and Azure Key Vault.

Purpose: To securely store and access secrets without hardcoding them into application code or configuration files.

55. What is the secure way to manage sensitive information?

Use a **Secrets Manager:** Store secrets in a dedicated service like AWS Secrets Manager, Azure Key Vault, or HashiCorp Vault.

Environment Variables: Use environment variables to inject secrets at runtime rather than storing them in code.

Encrypted Storage: Store sensitive data in encrypted databases or files, ensuring that encryption keys are managed securely.

Access Control: Implement strict access controls and auditing to ensure that only authorized personnel and applications can access sensitive information.

56. Have you worked with Kubernetes (K8s)?

If you have experience with Kubernetes, you might discuss:

Deploying and managing containerized applications using Kubernetes.

Configuring Kubernetes clusters and using tools like kubectl.

Managing services, ingress, and networking in Kubernetes.

Using Helm charts for packaging and deploying applications.

57. What is the difference between Docker and Kubernetes?

Docker: A platform for developing, shipping, and running applications inside containers. It simplifies the process of managing application dependencies and environment consistency.

Kubernetes: An open-source orchestration system for automating the deployment, scaling, and management of containerized applications. Kubernetes manages multiple containers across a cluster, providing features like load balancing, scaling, and self-healing.

58. Can you explain an end-to-end deployment for an application?

An end-to-end deployment process might involve the following steps:

1. **Code Commit:** Developers push code changes to a version control system like Git.
2. **CI/CD Pipeline:** A continuous integration pipeline builds the code, runs tests, and creates a Docker image.
3. **Image Storage:** The Docker image is pushed to a Docker registry.
4. **Deployment:** The image is pulled from the registry by Kubernetes, Docker Swarm, or another orchestration tool, and deployed to a staging or production environment.
5. **Monitoring & Logging:** The application is monitored for performance and errors, with logs collected and analyzed.
6. **Scaling & Updates:** The application is scaled based on demand, and updates are rolled out using a strategy like blue-green or canary deployment.

59. If you want to use Kubernetes instead of EC2 instances, how would you do it? Have you used Helm charts or other CD tools? How would you handle a project with multiple microservices on Kubernetes?

Using Kubernetes Instead of EC2: You would deploy your applications on a Kubernetes cluster rather than directly on EC2 instances. This involves setting up an EKS (Elastic Kubernetes Service) cluster in AWS or using another managed Kubernetes service.

Helm Charts: Helm is a package manager for Kubernetes that helps you manage Kubernetes applications. You can use Helm charts to deploy and manage multiple microservices in a consistent and repeatable manner.

Handling Multiple Microservices: Use Kubernetes namespaces to isolate microservices, and manage their deployment using Helm charts or a CI/CD tool like Jenkins, ArgoCD, or GitLab CI/CD. Implement service discovery, networking, and security policies to ensure seamless communication between microservices.

60. How do you connect a bastion host to a private network? Can you explain VPC and VPC peering?

Connecting a Bastion Host: A bastion host is typically set up in a public subnet of a Virtual Private Cloud (VPC) with access to the private network. Users connect to the bastion host via SSH, and from there, they can access resources in the private subnet.

VPC (Virtual Private Cloud): A logically isolated section of a cloud provider's network where you can launch and manage resources. It allows you to define IP ranges, subnets, route tables, and network gateways.

VPC Peering: A network connection between two VPCs that allows traffic to be routed between them using private IP addresses. This is useful for connecting resources across different VPCs without going over the public internet.

61. Have you configured a system where code is automatically merged and published upon a developer completing a ticket in Jira? What exactly have you managed?

Yes, I have set up a CI/CD pipeline where code is automatically merged and published once a developer completes a ticket in Jira. This process typically involves:

Integration with Jira: Configuring Jira to trigger CI/CD pipelines when a ticket is marked as "Done" or moved to a specific workflow stage.

Code Merging: Using tools like GitLab CI, GitHub Actions, or Jenkins to automatically merge feature branches into the main branch after passing tests.

Automated Testing: Running unit, integration, and end-to-end tests to ensure the quality of the code before merging.

Deployment: Using deployment tools like Kubernetes, Helm, or Docker Swarm to automatically deploy the merged code to staging or production environments.

62. How do you set up Nginx on a server?

Setting up Nginx on a server involves:

1. **Installation:** ○ For Ubuntu/Debian: sudo apt-get update && sudo apt-get install nginx
○ For CentOS/RHEL: sudo yum install nginx

2. **Configuration:** ○ Edit the configuration file located at /etc/nginx/nginx.conf or individual site configurations in /etc/nginx/sites-available/.

○ Define server blocks (virtual hosts) to specify different sites and their root directories.
○ Configure reverse proxy, load balancing, SSL/TLS certificates, and caching as needed.

3. **Start and Enable Nginx:** ○ Start the service: sudo systemctl start nginx
○ Enable it to start on boot: sudo systemctl enable nginx

4. **Testing:** ○ Test the configuration: sudo nginx -t
○ Ensure Nginx is running and properly serving content.

63. What is a load balancer and its benefits? What is Cloud NAT?

Load Balancer: A load balancer distributes incoming network traffic across multiple servers or services to ensure reliability, scalability, and high availability. Benefits include:
○ **Increased Fault Tolerance:** Distributes traffic to prevent overload on a single server.
○ **Scalability:** Easily manage increased traffic by adding more servers.
○ **Improved Performance:** Balances load based on performance metrics, reducing latency.

Cloud NAT: Network Address Translation (NAT) service in cloud environments like Google Cloud. It allows instances in private subnets to connect to the internet without exposing them to inbound internet traffic, maintaining security while enabling outbound connectivity.

64. What is the difference between a load balancer and a Cloud NAT gateway?

Load Balancer: ○ Distributes incoming traffic across multiple servers or services.
○ Primarily used for **load distribution, redundancy, and high availability.**
○ Works at various layers **(L4 for TCP, L7 for HTTP/HTTPS).**

Cloud NAT Gateway:
○ Provides outbound internet access for instances in a private network without exposing them to inbound traffic.
○ Used for secure, private instances that need internet access without being directly accessible from the internet.

65. How do you see yourself fitting into this particular role?

I see myself fitting into this role by leveraging my technical expertise in infrastructure management, automation, and cloud technologies. My experience in setting up CI/CD pipelines, managing deployments, and optimizing resource allocation aligns with the responsibilities of this role. I also bring problem-solving skills and a proactive approach to ensuring system reliability, which will contribute to the success of the team and organization.

66. Can you share an instance where you provided a solution for cost optimization while managing resource allocation?

In a previous project, I noticed that our cloud infrastructure was over-provisioned, leading to unnecessary costs. I implemented auto-scaling based on actual usage metrics and utilized spot instances for non-critical workloads. Additionally, I restructured the storage solution by moving infrequently accessed data to lower-cost storage classes. These changes resulted in a significant reduction in our monthly cloud expenses without compromising performance.

67. Describe a situation where the entire production instance crashed, and you had to fix it quickly. Have you experienced such a scenario?

Yes, I have experienced such a scenario. In one instance, our production server crashed due to a memory leak in the application. I quickly identified the issue using monitoring tools like Prometheus and logs from ELK Stack. To resolve it, I restarted the affected services and temporarily scaled up the infrastructure to handle the load. I then worked with the development team to identify and fix the memory leak, ensuring it didn't happen again.

68. What is blue-green deployment and why is it needed?

Blue-Green Deployment: A deployment strategy where two identical environments (Blue and Green) are maintained. The Blue environment is the active production environment, while the Green is the idle one. During deployment, the new version is deployed to the **Green environment.** After testing, traffic is switched to Green, making it the new production environment.

Why Needed:

- **Minimal Downtime:** Reduces downtime as the switch between environments is instantaneous.
- **Easy Rollback:** If issues arise, switching back to the Blue environment is straightforward.
- **Improved Reliability:** Reduces the risk of deployment failures affecting users.

69. What other deployment strategies do you know?

Canary Deployment: Gradually rolling out the new version to a small subset of users before a full deployment.

Rolling Deployment: Incrementally updating instances or servers with the new version, ensuring at least some instances are always running the old version.

A/B Testing: Similar to blue-green, but used for comparing different versions/features with live user traffic to determine which performs better.

Feature Toggles: Allows features to be turned on/off dynamically, enabling deployment of incomplete features without impacting the user.

70. What advanced AWS resource types have you worked with and utilized?

I have worked with several advanced AWS resource types, including:

AWS Lambda: Serverless computing for running code in response to events without managing servers.

AWS Fargate: Serverless compute engine for containers, allowing the deployment of containerized applications without managing the underlying infrastructure.

Amazon RDS: Managed database service for relational databases, including features like automated backups, scaling, and multi-AZ deployments.

AWS CloudFormation: Infrastructure as Code (IaC) tool for automating resource provisioning and management.

Amazon VPC Peering and Transit Gateway: For creating complex network architectures across multiple VPCs and accounts.

AWS Step Functions: Orchestration service for combining AWS Lambda functions and other services into serverless workflows.

71. How are hosted modules (like AI/ML) deployed, customized, and scaled as per different frontend/backend requirements in AWS with the help of a DevOps engineer?

Deploying, Customizing, and Scaling Hosted Modules in AWS

Deployment: DevOps engineers use tools like AWS CodePipeline to automate the deployment of hosted **AI/ML modules. This involves:** Containerization: Packaging the modules into Docker containers for portability.
Infrastructure Provisioning: Using IaC (e.g., CloudFormation) to set up the necessary AWS resources (EC2 instances, S3 buckets, etc.).
Deployment Automation: Using tools like AWS CodeDeploy to deploy the containers to the provisioned infrastructure.
Customization: Customization often involves: Configuration Management: Using tools like Ansible or Puppet to configure the modules based on specific requirements.
Environment Variables: Using environment variables to inject different configurations for different environments (development, testing, production).
Scaling: DevOps engineers leverage AWS services like: Auto Scaling Groups: Automatically adjust the number of instances based on load.
Elastic Load Balancing: Distribute traffic across multiple instances for high availability.
Serverless Computing: Using services like AWS Lambda to scale AI/ML workloads dynamically.

72. Can you describe a technology you had not heard of before but managed to learn and use on your own?

Learning a New Technology
Example: I recently learned about Apache Kafka, a distributed streaming platform. I was initially unfamiliar with its concepts but was intrigued by its potential for real-time data processing.
Learning Process: I started by reading documentation, watching tutorials, and experimenting with Kafka on my own. I also joined online communities and forums to connect with other users and learn from their experiences.
Application: I successfully implemented Kafka in a project to handle high-volume event streams, improving data processing efficiency and real-time insights.

73. What challenges have you faced as a DevOps engineer?

DevOps Engineer Challenges
Complexity: Managing complex cloud environments with multiple services and dependencies.
Automation: Finding the right balance between automation and manual intervention.
Security: Ensuring the security of cloud infrastructure and applications.
Collaboration: Working effectively with developers, operations teams, and other stakeholders.
Continuous Learning: Keeping up with the rapid pace of change in the cloud computing landscape.

74.Can you share real-life incidents where you solved an error after working for two or three days?

Real-Life Incident Resolution
Incident: A recent incident involved a production application experiencing slow performance due to a database query bottleneck.
Resolution: I spent two days analyzing logs, profiling the database, and identifying the inefficient query. I then worked with the development team to optimize the query, resulting in a significant performance improvement.

75.Have you managed large-scale databases and real-time backups or replication?

Large-Scale Databases and Backups

Experience: Yes, I have managed large-scale databases like MySQL and PostgreSQL, including real-time backups and replication.

Tools: I've used tools like Percona XtraBackup for MySQL backups and pg_dump for PostgreSQL backups. I've also implemented replication using tools like MySQL replication and PostgreSQL streaming replication.

76. During data loss, what strategy do you use to ensure no data loss, especially in critical applications like banking?

Data Loss Prevention Strategy

Strategy: For critical applications like banking, a multi-layered approach is essential: Redundancy: Use multiple data centers or cloud regions for replication and failover.

Backups: Implement frequent and automated backups to multiple locations.

Version Control: Track changes to data and maintain historical versions.

Monitoring: Monitor database health and performance to detect potential issues early.

Disaster Recovery Plan: Develop a comprehensive disaster recovery plan to restore data and services in case of an outage.

77.Have you faced any cyberattacks on systems you built and implemented? What precautions do you take?

Cyberattacks and Precautions

Experience: I've encountered security incidents like brute-force attacks and attempts to exploit vulnerabilities.

Precautions: Security Best Practices: Implement strong passwords, multi-factor authentication, and least privilege access.

Vulnerability Scanning: Regularly scan systems for vulnerabilities and patch them promptly.

Security Monitoring: Use security information and event management (SIEM) tools to monitor for suspicious activity.

Incident Response Plan: Develop a plan to respond to security incidents effectively.

78. What are the networking setup rules you follow?

Networking Setup Rules

Rules: Security Groups: Use security groups to control inbound and outbound traffic to instances.

Network Segmentation: Divide the network into smaller segments to isolate resources and limit the impact of security breaches.

Firewall Rules: Implement firewall rules to block unauthorized access.

VPN and Tunneling: Use VPNs and tunnels to secure communication between networks.

Network Monitoring: Monitor network traffic and performance to identify potential issues.

79. What are your daily responsibilities as a DevOps engineer?

Daily Responsibilities

Monitoring: Monitor system health, performance, and security.

Automation: Automate tasks like deployments, infrastructure provisioning, and backups.

Troubleshooting: Resolve issues and incidents.

Collaboration: Work with developers, operations teams, and other stakeholders.
Continuous Improvement: Identify areas for improvement and implement changes to enhance efficiency and reliability.

80. Which DevOps tools are you proficient with?

DevOps Tools Proficiency
Infrastructure as Code: Terraform, CloudFormation, Ansible, Puppet.
Containerization: Docker, Kubernetes.
CI/CD: Jenkins, GitLab CI/CD, AWS CodePipeline.
Monitoring and Logging: Prometheus, Grafana, ELK Stack.
Configuration Management: Ansible, Puppet, Chef.
Version Control: Git.

81. Can you describe the CI/CD workflow in your project?

CI/CD Workflow Description
Example: In a recent project, our CI/CD workflow involved:
1. **Code Commit:** Developers commit code changes to a Git repository.
2. **Build and Test**: A CI server (e.g., Jenkins, GitLab CI) automatically builds the application, runs unit tests, and performs code quality checks.
3. **Artifact Storage:** Successful builds are stored as artifacts in a repository (e.g., S3).
4. **Deployment:** The CD server (e.g., AWS CodeDeploy) deploys the artifact to the target environment (development, testing, production).
5. **Monitoring: Continuous monitoring tools** (e.g., Prometheus, Grafana) track application health and performance.

82. How do you handle the continuous delivery (CD) aspect in your projects?

Continuous Delivery (CD) Handling
Methods:
Automated Deployments: Use tools like AWS CodeDeploy to automate deployments to different environments.
Blue-Green Deployments: Deploy new versions of the application alongside the existing version, allowing for seamless switchover.
Canary Deployments: Gradually roll out new versions to a small subset of users, monitoring for issues before full deployment.
Feature Flags: Use feature flags to enable or disable features in the application without code changes, allowing for controlled releases.

83. What methods do you use to check for code vulnerabilities?

Code Vulnerability Checks
Methods:
Static Code Analysis: Use tools like SonarQube or Snyk to analyze code for vulnerabilities and security issues.
Dynamic Code Analysis: Use tools like Burp Suite or ZAP to test the application in runtime for vulnerabilities.
Security Scanning: Use tools like AWS Inspector or Qualys to scan infrastructure and applications for vulnerabilities.

84. What AWS services are you proficient in?

AWS Service Proficiency

Services: Compute: EC2, Lambda, ECS, EKS.
Storage: S3, EBS, EFS.
Networking: VPC, Route 53, Load Balancers.
Database: RDS, DynamoDB, Redshift.
CI/CD: CodePipeline, CodeBuild, CodeDeploy.
Security: IAM, Security Groups, KMS.
Monitoring: CloudWatch, CloudTrail.

85. How would you access data in an S3 bucket from Account A when your application is running on an EC2 instance in Account B?

Accessing S3 from Account B

Method: Use IAM roles and cross-account permissions:

1. **Create Role:** In Account A, create an IAM role with permissions to access the S3 bucket.
2. **Assume Role:** In Account B, configure the EC2 instance to assume the role created in Account A.
3. **Access S3:** The EC2 instance can now access the S3 bucket using the assumed role's credentials.

86. How do you provide access to an S3 bucket, and what permissions need to be set on the bucket side?

S3 Bucket Access and Permissions

Access: You can provide access to an S3 bucket using:
IAM Policies: Attach policies to users, groups, or roles to grant specific permissions.
Bucket Policies: Define access control rules directly on the bucket.
Permissions: Common permissions include:
Read: Allows users to read objects from the bucket.
Write: Allows users to write objects to the bucket.
Delete: Allows users to delete objects from the bucket.
List: Allows users to list objects in the bucket.

87. How can Instance 2, with a static IP, communicate with Instance 1, which is in a private subnet and mapped to a multi-AZ load balancer?

Instance Communication with Private Subnet

Method:

Use a **NAT gateway** or a **bastion host:**

NAT Gateway: A managed service that provides internet access for instances in a private subnet.
Bastion Host: A secure server in a public subnet that allows access to private instances.

88. For an EC2 instance in a private subnet, how can it verify and download required packages from the internet without using a NAT gateway or bastion host? Are there any other AWS services that can facilitate this?

EC2 Instance in Private Subnet Accessing Internet

Method: Use a private DNS hostname:

1. **Private DNS:** Configure a private DNS zone within your VPC.
2. **Private Hostname:** Assign a private hostname to the EC2 instance.

3. **DNS Resolution:** The EC2 instance can resolve the private hostname to access internet resources.

89. What is the typical latency for a load balancer, and if you encounter high latency, what monitoring steps would you take?

Load Balancer Latency and Monitoring

Typical Latency: Load balancer latency typically ranges from a few milliseconds to a few hundred milliseconds, depending on factors like network conditions and load.

Monitoring Steps: CloudWatch Metrics: Monitor load balancer latency using CloudWatch metrics.

Network Tracing: Use tools like AWS X-Ray to trace requests through the load balancer and identify bottlenecks.

Performance Testing: Run load tests to simulate real-world traffic and identify performance issues.

90. If your application is hosted in S3 and users are in different geographic locations, how can you reduce latency?

Reducing Latency for S3 Hosted Application

Methods:

Edge Locations: Use AWS CloudFront to cache content at edge locations closer to users, reducing latency.

Regional Buckets: Store data in S3 buckets in the same region as the users, minimizing network hops.

Content Delivery Networks (CDNs): Use a CDN to distribute content across multiple locations, reducing latency for users worldwide.

91. Which services can be integrated with a CDN (Content Delivery Network)?

CDN Integration

Services: CDNs can integrate with various services, including: Web Servers: Apache, Nginx, IIS.

Content Management Systems (CMS): WordPress, Drupal, Joomla.

Cloud Storage: AWS S3, Google Cloud Storage, Azure Blob Storage.

Streaming Services: Netflix, YouTube, Twitch.

API Gateways: AWS API Gateway, Google Cloud Endpoints.

92.How do you dynamically retrieve VPC details from AWS to create an EC2 instance using IaC?

. Dynamically Retrieving VPC Details

Method: Use Terraform's data sources:

1. **aws_vpc Data Source:** Retrieve details of a specific VPC by its ID or name.
2. **aws_subnet Data Source:** Retrieve details of subnets within a VPC.
3. **aws_security_group Data Source:** Retrieve details of security groups associated with a VPC.
4. **aws_instance Data Source**: Retrieve details of existing EC2 instances within a VPC.

93. Managing Unmanaged Resources in Terraform

Approach: Use Terraform import command to bring existing unmanaged resources under Terraform's control. This allows you to manage them alongside your IaC code.

94. Passing Arguments to VPC During Import

Method: Use the --var flag with the terraform import command to pass arguments:

terraform import aws_vpc.example "vpc-1234567890abcdef0" --var="cidr_block=10.0.0.0/16"

95. What is the master-slave architecture in Jenkins?

A master-slave architecture in Jenkins allows you to distribute build tasks across multiple nodes (slaves).
This provides:
Parallel Execution: Run builds concurrently on multiple nodes, reducing build times.
Resource Optimization: Utilize different hardware configurations for different build tasks.
Scalability: Scale your Jenkins infrastructure by adding more slave nodes.

96. How do you integrate LDAP with AWS and Jenkins?

A: You can integrate LDAP with AWS and Jenkins by:
1. **Configuring LDAP:** Set up an LDAP server and configure it to authenticate users.
2. **AWS IAM**: Create an IAM role with permissions to access the LDAP server.
3. **Jenkins Configuration:** Configure Jenkins to use the LDAP server for authentication.

97. What are some key features of GitHub?

A: Key features of GitHub include:
Version Control: Track changes to code over time.
Collaboration: Facilitate collaboration among developers.
Pull Requests: Enable code reviews and approvals.
Issues: Track bugs, feature requests, and other tasks.
Projects: Organize and manage work items.

98. What are some key features of Jenkins?

A: Key features of Jenkins include:
Continuous Integration (CI): Automate build, test, and deployment processes.
Continuous Delivery (CD): Deploy applications to different environments.
Pipeline as Code: Define pipelines using code (Jenkinsfile).
Plugins: Extend Jenkins functionality with a wide range of plugins.
Master-Slave Architecture: Distribute build tasks across multiple nodes.

99. What are the benefits and uses of CI/CD?

A: Benefits of CI/CD:
Faster Delivery: Reduce the time it takes to deliver new features and updates.
Improved Quality: Catch bugs and errors early in the development process.
Increased Efficiency: Automate repetitive tasks, freeing up developers to focus on innovation.
Reduced Risk: Deploy changes more frequently and with less risk.
Uses of CI/CD:
Software Development: Automate build, test, and deployment processes.
Infrastructure Management: Provision and configure infrastructure automatically.
Data Pipelines: Automate data processing and analysis tasks.

100. What is a GitHub workflow, and how is it used?
A: A GitHub workflow is a set of automated tasks that are triggered by events in a GitHub repository.
You use workflows to:
Build and Test Code: Automate build and test processes.
Deploy Applications: Deploy applications to different environments.
Run Code Analysis: Perform code quality checks and security scans.

101. How do you handle merge conflicts in Git?
A: You handle merge conflicts in Git by:
1. **Identifying Conflicts:** Git will identify conflicts when merging branches.
2. **Resolving Conflicts**: Manually resolve conflicts by editing the affected files.
3. **Staging Changes:** Stage the resolved files using git add.
4. **Committing Changes:** Commit the changes with a descriptive message using git commit.

102. What steps do you take when a build fails in Jenkins?
A: When a build fails in Jenkins, you should:
1. **Analyze Logs:** Review the build logs to identify the cause of the failure.
2. **Troubleshoot:** Investigate the issue and try to resolve it.
3. **Fix Code:** If the failure is due to a code error, fix the code and rebuild.
4. **Update Configuration:** If the failure is due to a configuration issue, update the Jenkins configuration.
5. **Rollback:** If necessary, roll back to a previous working version of the application.

103. How do you execute jobs in AWS?
A: You can execute jobs in AWS using:
AWS Batch: A fully managed batch computing service for running large-scale, compute-intensive jobs.
AWS Lambda: A serverless computing service for running code in response to events.
AWS ECS: A container orchestration service for deploying and managing containerized applications.

104. What are Ansible roles, and how do you use them?
A: Ansible roles are a way to organize and reuse Ansible playbooks. They encapsulate tasks, variables, and dependencies related to a specific component or service. You use roles to:
Modularize Playbooks: Break down complex playbooks into smaller, reusable units.
Simplify Deployment: Deploy multiple components or services with a single role.
Improve Maintainability: Make it easier to manage and update Ansible configurations.

105. How do you ensure data persistence with Docker volumes?
A: Use Docker volumes to persist data outside the container:
Named Volumes: Create named volumes that can be shared between containers.
Data Volumes: Mount data volumes from the host machine into the container.
Bind Mounts: Mount directories from the host machine into the container.

106. What are the key differences between Docker and Kubernetes?
A: Key differences:
Scope: Docker focuses on containerization, while Kubernetes focuses on container orchestration.
Management: Docker manages individual containers, while Kubernetes manages clusters of containers.
Scalability: Kubernetes provides advanced features for scaling and managing large-scale deployments.

Networking: Kubernetes offers more sophisticated networking capabilities for container communication.

107. How do you securely store credentials in GitHub?
A: You can securely store credentials in GitHub using:
GitHub Secrets: Use GitHub Secrets to store sensitive information securely.
Environment Variables: Set environment variables in your GitHub workflow to access credentials.
GitHub Actions: Use GitHub Actions to manage credentials and access them within your workflow.

108. Where is the Jenkinsfile typically stored?
A: The Jenkinsfile is typically stored in the root directory of your Git repository.

109. How is pull request approval managed in GitHub?
A: GitHub provides features for managing pull request approvals:
Required Approvers: Specify the number of reviewers required to approve a pull request.
Review Policies: Define policies for code reviews, such as requiring specific reviewers or checks.
Approval Flow: Control the approval process, such as requiring approvals from specific teams or roles.

110. How do you execute a shell script within a Python script?
Use the subprocess module:
import subprocess
subprocess.run(["/path/to/script.sh"])

111. How do you ensure data persistence with Docker volumes?
A: Use Docker volumes to persist data outside the container:
Named Volumes: Create named volumes that can be shared between containers.
Data Volumes: Mount data volumes from the host machine into the container.
Bind Mounts: Mount directories from the host machine into the container.

112. What are the key differences between Docker and Kubernetes?
A: Key differences:

Scope: Docker focuses on containerization, while Kubernetes focuses on container orchestration.
Management: Docker manages individual containers, while Kubernetes manages clusters of containers.
Scalability: Kubernetes provides advanced features for scaling and managing large-scale deployments.
Networking: Kubernetes offers more sophisticated networking capabilities for container communication.

113. How do you securely store credentials in GitHub?

A: You can securely store credentials in GitHub using:
GitHub Secrets: Use GitHub Secrets to store sensitive information securely.
Environment Variables: Set environment variables in your GitHub workflow to access credentials.
GitHub Actions: Use GitHub Actions to manage credentials and access them within your workflow.

114. Where is the Jenkinsfile typically stored?
A: The Jenkinsfile is typically stored in the root directory of your Git repository.

115. How is pull request approval managed in GitHub?

A: GitHub provides features for managing pull request approvals:
Required Approvers: Specify the number of reviewers required to approve a pull request.
Review Policies: Define policies for code reviews, such as requiring specific reviewers or checks.
Approval Flow: Control the approval process, such as requiring approvals from specific teams or roles.

116. Have you upgraded any Kubernetes clusters?

A: (This is a yes/no question, followed by a description of your experience if you have.)

117. How do you deploy an application in a Kubernetes cluster?

A: You can deploy applications in a Kubernetes cluster using:
kubectl: Use kubectl commands to deploy applications using YAML or JSON manifests.
Helm: Use Helm charts to package and deploy applications, simplifying the process.
Knative: Use Knative for serverless deployments on Kubernetes.

118. How do you communicate with a Jenkins server and a Kubernetes cluster?

A: You can communicate with a Jenkins server and a Kubernetes cluster using:
Jenkins Plugins: Use Jenkins plugins like the Kubernetes plugin to interact with a Kubernetes cluster.
API Calls: Use the Kubernetes API to interact with the cluster programmatically.
kubectl: Use kubectl commands from within Jenkins to manage Kubernetes resources.

119. How do you generate Kubernetes cluster credentials?

A: You can generate Kubernetes cluster credentials using:
Service Accounts: Create service accounts in Kubernetes to provide access to specific resources.
kubeconfig: Generate a kubeconfig file that contains authentication and connection details for the cluster.

120. Do you only update Docker images in Kubernetes, or do you also update replicas, storage levels, and CPU allocation?

A: You can update various Kubernetes resources, including:
Docker Images: Update the container image used by a deployment.
Replicas: Scale up or down the number of replicas for a deployment.
Storage Levels: Adjust storage capacity for persistent volumes.
CPU Allocation: Modify CPU and memory resource requests and limits for containers.

121. What types of pipelines are there in Jenkins?

A: Jenkins offers several pipeline types:
Pipeline: A flexible and powerful way to define complex workflows.
Freestyle: A simpler option for basic build and deployment tasks.
Multibranch Pipeline: Automatically creates pipelines for different branches in a Git repository.

122. Can you define environment variables inside your Jenkins pipeline?

A: Yes, you can define environment variables inside your Jenkins pipeline using:
Pipeline Script: Use the environment directive within your Jenkinsfile to define environment variables.
Global Variables: Configure global environment variables in Jenkins settings.

Credentials: Store sensitive information as credentials and access them within the pipeline

123. What is the role of artifacts in Jenkins, and why do we need to push them to Nexus instead of building and storing them locally?

A: Artifacts are the output of a build process, such as compiled code, container images, or test reports.
Pushing artifacts to Nexus (a repository manager) provides:
Version Control: Track different versions of artifacts.
Dependency Management: Manage dependencies between projects.
Security: Control access to artifacts and ensure their integrity.

124. If you're developing a Python-based application, how do you separate the packages needed for your local deployment to avoid interfering with globally installed packages?

A: Use virtual environments:
1. **Create Virtual Environment:** Use **python -m venv <env_name>** to create a virtual environment.
2. **Activate Environment:** Activate the environment using
   source <env_name>/bin/activate.
2.**Install Packages:** Install packages specific to your project using
pip install <package_name>.

124. What are the prerequisites before importing a VPC in Terraform?

A: Before importing a VPC, you need:
Terraform Configuration: A Terraform configuration file defining the VPC resource you want to import.
Resource ID: The unique identifier (ID) of the VPC in AWS.
Resource Type: The correct Terraform resource type (e.g., aws_vpc).

125. If an S3 bucket was created through Terraform but someone manually added a policy to it, how do you handle this situation using IaC?

A: You can use Terraform's terraform plan command to detect differences between the current state of the S3 bucket and your IaC configuration. This will highlight the manually added policy.
You can then:
Update IaC: Modify your Terraform configuration to include the manually added policy, ensuring consistency.
Ignore: If the policy is acceptable, use the ignore_changes option in Terraform to exclude it from future plans.

126. How do you handle credentials for a PHP application accessing MySQL or any other secrets in Docker?

A: You can manage credentials securely in Docker using:
Environment Variables: Set environment variables within the Docker container to store credentials.
Secrets Management: Use a secrets manager like AWS Secrets Manager or HashiCorp Vault to store and retrieve credentials securely.
Docker Secrets: Use Docker secrets to store sensitive information separately from the container image.

127. What is the command for running container logs?

A: The command for running container logs is:

docker logs containerid or container name

Revision-1:

1.How many Environments?

Development
Staging/UAT
Production

2.When will you deploy on the environments?

Whenever we see any communication email from developer's team to deploy
Specific version on specific environment, then will start our deployment process. We raise a change **request (CR)in Service Now(SNOW).**
We provide the version number to deploy, environment, time and then rollback version, and also add sanity checks and submit the CR request.
CR request will go for couple of approvals, once we received the approvals then we will deploy the application based on timing.

Development -->UAT-->Production

3.How many jobs do we have?

For each application we can say as **3.**

4)Can you explain you CICD process/jenkins pipeline?

1st job:

We have 6 stages for our continuous Integration.

1)**Git clone** to checkout the source code.
2)**Sonarqube** Integration to check the quality of the code.
3)**Maven** Compilation to create a war package for us.
4)**Nexu**s Artifactory to push the artifact to the nexu repository Post build actions
5)**Slack** integration to send the build output/notification to the channels
6)**Email** integration to send the email alerts.

2nd job:

1) To build the **docker image** and **push to registry**

3rd job:

1)Deploy to **Tomcat**
2)Deploy to **ECS.**
3)Deploy to **kubernetes cluster**

5)Have you ever worked on production server issues?

Yes

6)What type of tickets you work on?

I generally work on **P1, P2 and P3** tickets.

7)What is the SLA(Service level agreement) for these tickets?

Priority 1 (P1) –

A complete business down situation or single critical system down with high financial impact. The client is unable to operate.

Priority 2 (P2) –

A major component of the clients' ability to operate is affected. Some aspects of the business can continue but its a major problem.

Priority 3 (P3) –

The clients' core business is unaffected but the issue is affecting efficient operation by one or more people.

Priority 4 (P4) –

The issue is an inconvenience or annoying but there are clear workarounds or alternates.

Priority 5 (P5) –

The issue is a background or planned task and will be addressed when time permits or on the planned date.

Standard SLA

Priority1	2 hours to respond
Priority2	8 hours to respond
Priority3	16 hours to respond
Priority4	10 days to respond

8) Can you explain your recent issue?

1) We were able to see lot of tickets from development teams that they are not able to see the logs of production server on splunk centralized server.

We checked if the splunk forwarders are running or not on that servers and found that these forwarders were down for some reason..

So we have created a bash script to check the status of splunk forwarder for every 5 mins and alert the team if these forwarders are down ,

so that we can forecast the issue and make sure splunk forwarders are up and running always

2) **In Jenkins** we can able to see lot of issues with connectivity, this will happen when there is a change in the systems,i mean bcz of patching activities..

And also pipelines will get failed bcz of any missing dependencies or any kind of missing artifacts.

Recent challenge/achievment:

1)**Used s3** bucket to store terraform statefiles and also used dymanod db locking to protect statefiles which not be used by multiple users at the same time.

9) What types of incidents you commonly see?

We commonly see incidents related to pipeline failures,
400 errors,500 errors and microservices are taking much time for response.
War package in s3, trigger war package using lambda and deploy on ec2.

Revision-2:

Linux:

1)Have you worked on Linux?

yes

2)How many years of experience?

3 years or 4 years depending upon experience

3)How much you can rate on linux out of 5 ?

Iam pretty comfortable on linux platform. you can rate upto **3 out of 5.**

4)Which version/flavours of linux you have worked on ?

Amazon linux 2
ubuntu 22
RHEL - 7.9 version

5)Running port number?

netstat -na | grep 80

6)How to list the running services?

ps -ef

7)How to know the name of service which is running in port 8080?

lsof -i tcp:8080

8)Important commands?

kill -9 PID TOP
df -h
free -m

9)What are the three different parameters in top command output?

load avaerage,
number of tasks,
cpu utilization,

zombie process,
memory and swap memory.

10)What is load avaerage in top command?
load average: 0.29, 0.53, 0.42
https://serverpilot.io/docs/how-to-use-the-top-command/

11)Swap memory?
Actual amount of RAM is full then swap memory is used to execute the process.

12)How swap memory is assigned?
Hard disk

13)I have 1 gb of space in my filesystem and when trying to create one file, it is throwing error as space not available?
Inodes are occupied.
Inode is nothing but index node and unique identification.

14)Clean up some files in linux?
Login to ec2 df -h
du -sk

15)How to create password less authentication?
Create ssh-keygen
copy public_key in authorized key in another server and then we can cannot with any password.

16)How to attach a new file system in linux?
Talk about **lsblk** command mount command

17)LVM in linux?
LVM, or Logical Volume Manager, is a storage management technology for Linux and other Unix-like operating systems.
LVM provides a flexible and scalable way to manage disk storage, making it easier to allocate, resize, and manage storage volumes.

18)Hard link and soft link in linux?
https://www.geeksforgeeks.org/difference-between-hard-link-and-soft-link/

19)Cronjob in linux?
will help us to execute the script without any manual intervention based upon the timing.
*** * * * * /var/lib/test.bash**

20)Script to be executed when machine is rebooted by using cron job?
@reboot /var/lib/test.bash

21)Different boot levels in linux? Runlevel 0 (Shutdown):

Runlevel 1 (Single User Mode or Rescue Mode):
Runlevel 2 (Multi-User Mode, No Networking):
Runlevel 3 (Multi-User Mode with Networking):
Runlevel 4 (Unused):
Runlevel 5 (Graphical User Interface Mode):
Runlevel 6 (Reboot):

22)AWK and SED?

Used for Data processing at run time.
sed can be used to replace any text/string from one file awk can be used to cut a specific output.

Git and Github:

1)Branching Strategy?

Git flow strategy.
Release -> Is used for production env
Development -> Dev and staging related code.
Hotfix -> Bugs fixing, pulled from release branch.
Feature branch -> for new updates, pulled from development branch.
Master --> Copy of release.

2)Pick a specific commit from one branch to another branch?

git cherry-pick <commit id>

3)Undo the previous commint?

git revert <commit_id>

4)Git stash?

Moving file to temporary stash location.

5)Difference between git pull and git fetch?

git pull to pull the latest commits from central to local. git fetch is to pull the metadata configuration

6)Difference between rebase and merge?

Git merge and rebase is used to merge the two branches. git rebase will help us with more clear commit history.

7)Have you faced merge conflicts?

yes, we can resolve manually by checking the difference of both the files. make the changes manually and commit them back and push.

8)Explain/what is PR process?

pull request (PR) is raised to merge two branches,

when raising PR we need to select two reviewers, who will check the change and approve. once approved the code will be merged.

Revision-3:

Bash Scripting:

1)Have you worked on bash/shell scripting?

Yes.

2)What was the task or why you have used bash scripting or the use case?

1)We have used bash script to monitor the custom metric.
Status of splunk/tomcat or any other service and send the metric to cloudwatch dashboard.
if status is 1 it means service is running, if status is 0 it means service is not running.
2)Take backup of jenkins server.

3)Can you explain your script of write down the script or how is your script checking the status?

```
ps -ef | grep service_name
#!/bin/bash
#set -x
INSTANCE_ID=$(/opt/aws/bin/ec2-metadata -i | cut -d " " -f2)
checkTomcatStatus(){ counter=0
ps x | grep /opt/apache-tomcat-9.0.65/ | grep -v grep | #Need to change tomcat version while read -r LINE
do
read PROCESS_ID <<< $LINE
counter=$((counter+1))
echo $counter done
}
i=$(checkTomcatStatus) #echo $i
if [ "$i" == "" ] then
aws --region ap-south-1 cloudwatch put-metric-data --metric-name tomcat --value 0 -- namespace tomcat --dimensions InstanceId=$INSTANCE_ID
else
aws --region ap-south-1 cloudwatch put-metric-data --metric-name tomcat --value 1 -- namespace tomcat --dimensions InstanceId=$INSTANCE_ID
fi
```

4)What does $? means?

Exit status

5)What does $# means?

Total number of arguments passed in script.

6)Bash Script to check if the directory is available or not?

#!/bin/bash
if [-d "/opt"] then
echo "Directory is available" else
echo "Directory not available" fi

7)Bash script to build docker image,push the image to registry and deploy on k8s?

#!/bin/bash
Docker_image="nginx:latest"
reposrity_name="infosys_repo_test" docker_hub_user_name="xxxx" dockerhub_user_pwd="xxxx"
docker build -t <tagname> .
docker login -u $docker_hub_user_name -p $dockerhub_user_pwd docker tag $reposrity_name
docker push $repostiroy_name kubectl apply -f file_name.yml

8)How to run the bash script in background?

nohup
set -X in bash script

9)How to run bash script?

./script.shell bash script.sh

10)Bash script to read the content of a file line by line?

#!/bin/bash
 # Specify the path to the file
file_path="path/to/your/file.txt"
 # Check if the file exists
if [-e "$file_path"]; then
 # Read the file line by line
while IFS= read -r line; do
 # Process each line (replace this with your desired action)
echo "Processing line: $line" done < "$file_path"
else
echo "File not found: $file_path" fi

11)Cron job 5 stars?

* * * * * /pathofscript
first * --> Minutes
second * --> hours
third * --> days of month
fourth * --> month
fifth * --> Day of week
Cron job to execute script at 2:00 p.m every sunday
* 14 0 * *

Ansible:

1)Have you worked on ansible?
Yes

2)have you developed playbooks from scratch?
Yes

3)Use case in your project where you have used ansible?
For pre patching and post patching activities we have majorly used ansible playbooks.

4)Write any sample ansible playbook?

- name: Install Apache hosts: all
tasks:
-name: Install apache
yum: apache2
state: latest
notify:- restart apache2
-name: Copy index.html
copy: src= index.html dest= /var/www/html
- handlers: restart apache2 service: apache2 state:started

5)What is inventory?
Inventory/hosts is the place or file where we configure the ip address of other servers.

6)How many types of inventories are there?
1)**Static inventory**
2)**Dynamic inventory** --> we should have a python program which will capture the new ip of servers.

7)Have you worked on ansible roles, if yes then why do we use roles?
Instead of writing all tasks in one playbook we can segregate these tasks in different roles. With the help of roles we can reuse the same piece of code again and again for different playbooks.

8)Directory structure of roles?
files
handlers vars default
templates metadata tasks

9)What is ansible-galaxy?
Galaxy is a registry or public repository with predefined roles from community.

10)How to store/pass secrets in ansible?
By using ansible vault, we can store any secret details like passwords.

11)have you worked on Ansible Tower?
No, but we have used ansible awx which is open source and similar to tower.
Ansible tower is for enterprise edition and is for GUI.

12)What is ad-hoc command?
Ad-hoc commands is single line command to install or do any task for us.

13)List some moduels you have used in modules?
yum, apt, service, copy, register, debug, handlers, notifiers, dir, linein, become, shell

14)Difference between debug and register module?
Register module is to capture the output.
Debug module is used to print the message.

15)What is handlers and notifiers?
Handlers will help us to execute the specific task only if the dependant task has been successfully executed.

16)why do we use line in module?
This is used to add/modify the content of a file.

17)Why do we use dir module?
We use dir to create directory.

18)Ansible playbook to create ec2 instance?

-hosts: localhost vars_file:
-secret.yml
name: Provision ec2 amazon_aws_ec2:
ami_id: ami-xxxxx
key_name: test instance_type: t2.medium region: us-east-1
ansible vault and create one file for access key and secret key
ansible-playbook playbook.yml --ask-vault-password

Revision 4:

AWS EC2:

1)Which type of ec2 you have used in project?
t2.large

2)t2.large configurations?
8 gb of ram and 4 core of cpu's

3)If ec2 pem key is lost then how will you retrieve your pem key ?
If we lost the pem key then we cannot retrieve the pem key.
We need to take the snapshot, create ami and then launch instance using the ami and create one new pemkey to the ec2.

4)How to move one ec2 from us-east-1 to us-east2?
Take the snapshot of the current ec2, create ami to the target region and then we can launch new ec2 instance.

5)If you see the ec2 system checks is 1/2, then how to resolve?
system checks might be failed because of
connectivity hardware.
We need to restart our ec2, so that it will be launched in new host.

6)You have launched one ec2 and you are unable to ssh? what steps will you perform to resolve?
1)Checking security groups.
2)Pem key is correct or not.
3)No internet connection, IGW.
4)No space on hardisk.
5)Check the system logs of ec2.

7)Different families of ec2:
t2, t3, c,m

8)Type of instances?
1**)Ondemand**
2)**Reserve**d -70% cheaper then on demand
3)**Spot instances** - 90% cheaper then on demand

9)How to make our ec2 to communicate with s3?
we need to create on IAM role with s3 options IAM role should be attached to ec2.

S3:

1)What do you store in s3 in your project?
State file related to terraform we are storing in s3.
VPC flow logs, access logs for load balancer and we are also storing images related to application.
We do have static website in our s3.

2)How to replciate objects of one s3 bucket to another s3 bucket?
we can enable cross region replication
where we create bucket policy and add details like source bucket, destination bucket.

3)What is versioning in s3?
It will help us to keep multiple files with different version.

4)Different lifecycle of objects in s3?
1)Standard
2)Infrequent access
3)Glacier
4)Deep archive Glacier.

5)Access control list in s3?
ACL is used to configure different level of access to objects in s3.

6)You have on s3 bucket and this should be accessible to specific user?
We need to create one bucket policy with the user details.
We need to create iam policy to the user with the bucket details.

7)I want to access my s3 bucket in another aws account. Bucket policy with target account details.
IAM role with bucket details in the target account.

8)if you have file of 1gb of size then how can we upload file?
1)aws s3 commands with --recursive
2)Multipart upload.

Virtual Private Cloud:

1)What is VPC?
Virtual private cloud, do create our own network.

2)What is CIDR in VPC?
Classless inter domain range is used to assign the number of ip's to vpc.

3)192.168.0.0/24, how many ip address?
32-24 = 256 ip's.

4)How many types of subnets are there?
Public and Private

5)What is the difference between public and private?
Public will be having internet connection and private will not have internet connection.

6)How to connect to ec2, which is in private subnet?
We can use NAT gateway and Nat should be deployed on public subnet. And Nat details should be added in private subnet RT.

7)If you have multiple vpc then how can we communicate?
we can use vpc peering or transist gateway.

8)Difference between vpc peering and transist gateway?
VPC peering can be used to communicate between multiple vpc in same region, another region or another account.
This can be used for small network, means 5-10 vpc
Trasist gateway: Is used as router-hub configuration and it is suitable for large network.

9)How many vpc's can be created in one region?
5 vpc

10)Subnet range is 192.168.0.0/28 then how many ip's are available?
16 will be associated but **11** will be available.

11)What is elastic ip?
Fixed ip for our ec2 and we need to purchase these ip from aws.

12)Difference between public and private ip?
Public ip can be used to connect from the outside world, assigned by aws and will be changed if we shutdown and start our ec2.
private ip is used to communication inside the network and this will be assigned from CIDR.

Load balancer:

1)How many types of load balancers are available in aws?
1)Classic load balancer
2)Application load balancer
3)Network Load balancer
4)Gateway load balancer

2)Difference between application and network load balancer?
Application load balancer works on protocol HTTP and HTTPS. it works on layer7 i.e., application layer
We can configure multiple listeners and multiple target groups and work on path prefix.
Network load balancer works on TCP and UDP protocol. it works on layer4 i.e., transport layer.
We can configure multiple listeners and multiple target groups and work on path prefix. We can configure elastic ip to network load balancer.

3)When to use which load balancer?
ALB can be used if we are aware of the traffic and if our application on http and https protocol. NLB can be used to server sudden spike in traffic.

Ex: Live streaming,video processing.

4)What is sticky session in load balancer?
Sticky session will helps to hold the user session state.

IAM User and roles

1)What is IAM?
IAM is identity access management to provide different level of access to different users.

2)Difference between role and policy?
Policies --> Set of permissions.
Role --> Used to attach to the aws resources.
Role --> policy.

3)If we have multiple aws account then how can we access services from different account.
Cross Account iam role.

4)Different type of policies?
Inline policy --> Dedicated to user and it will be deleted once the user is deleted.
Aws managed policy --> Provided by aws customer managed policy --> Created by user.

Revision_Autoscaling Groups:

1)Have you used asg in your project?
we have used asg in previous project.

2)How many type of ASG are available?
Two types:
1)Horizontal scaling
2)vertical scaling.

3)Difference between horizontal and vertical scaling?
horizontal scaling is used to add new resources to the environment. Vertical scaling is used to add more power to the existing ec2 instance.

4)How will ASG work?
It will scale up and scale down based on different metrics.
Ex: Cpu utilization,disk I/O, number of users at load balancer level. We need to configure launch template or launch configuration.

Where we configure the details like
ami id, size, region, vpc, type if instance, desired number, min number and max number.

5)Difference between launch template and launch configuration?
Launch template is a template with all details.
We can have multiple versions of Launch template.
launch configuration cannot be edited and we need to create new launch configuration. launch Template is recommended by aws.

6)Example asg has triggred 4 new instances and total is 10 now.
For example if your cpu utilization is back to normal which instances will be deleted? It will try to delete the older instances.

Cloudfront:

1)What is cloudfront?
cloudfront will create a distributed network to improve the performance and reduce the latency.
For the first time it will connect to original server and then it will try to cache in edge locations.

2)Why have you used cloudfront in your project?
we have used cloudfront to map with s3 static website and cloudfront dns is used to configure in the R53.

Cloudwatch:

1)What is cloudwatch?
Cloudwatch is used to monitor our aws resources.

2)What are the metrics you have collected using cloudwatch?
cpu utilization
disk space disk i/o
network bytes
custom metric to monitor status of service.
400 Error from load balancer 500 error from load balancer.

3)How cloudwatch is able to collect the metrics from server?
we need to install cloudwatch agent in the ec2 instances of all servers
and this agent is responsible to collect the metrics and store on cloduwatch dashboard.

4)Have you created dashboard from scratch?
Yes

5)Can you explain the process of creating dashboard?
We need to go to cloudwatch service, select dashboard, select widgets. then select the metric which you want to add to the dashboard.
We can configure alarms and we can set the SNS to send the notification to email based on metrics.

ECS and ECR:

1)What is the difference between ecs and ecr?

ECR is used to store docker images in aws.
ECS is orchestration platform provided by aws.

2)Difference between ecs, dockerswarm and k8s?

ECS, dockerswarm and k8s are used for container orchestration. ECS is limited to aws only.
Dockerswarm will help us with orchestration.
k8s is a single platform with all the features related to orchestration. Like load balancer, autohealing feature.

3)What is task in ecs?

task = container

4)What is task definition?

Template with all instruction to create our cluster like image, soft limit, harlimit,
port numbers.

5)How can we scale up and scale down task in ecs?

Using services, we can scale up and scale down our task.

6)Which one have you used ecs ec2 or ecs fargate?

ecs ec2.

7)Difference between ecs ec2 and ecs fargate?

ecs ec2 will use regular ec2 instance and we need to manage that ec2. ecs is serverless service where aws will manage our cluster.

Route53:

1)Explain the different records in r53.

A record is for ip
AAAA - ipv6
CName: colonical name
NS: Name servers
SOA: State of authority
Alias - Attach to another resource.

2)Routing policies in r53?

Simple routing policy geolocation routing policy failover routing policy
Geoproximity routing policy

3)If we have users in us-east-1 and us-east-2 which routing policy will be used?
Geolcoation routing policy

RDS:

1)have you used RDS service?
Yes, we have used **Rds** and **mysql** image.

2)What is Multi-AZ in RDS?
high availability rds instance to improve the performance. one will be primary and another will be standby.
backup will be done by standby rds instance.

3)Can we connect to standby rds instance?
No, AWS will automatically switch if something goes wrong with primary.

4)How data will be replicated from primary db to standbydb?
Aws uses synchronous replication strategy to copy data between databases.

5)What is Read Replicas?
Read replicas is used for read operations to improve the performance of db.

6)How to take backup of RDS?
We have one option called maintenance window, where we can configure the time and aws will take the backup of RDS instance.

7)Have you ever worked on migrating db to rds?
Yes

8)How to migrate your application from physical data center to aws?
1)Capacity planning.
2)Dependencies.
3)Networking
4)Api calls.
Based upon the above details will try to check for the relevant services in aws and try to move my application.

9)How to migrate database to RDS?
1)Take dump
2)mysqldump command to dump my data from db to rds.
3)Try to create tables, columns and try to copy the data.

10)What is RPO and RTO?

RPO: Recovery point objective

1)Time between the last backup and whenever the failure occured.

Ex: We have taken backup at 6 a.m and failure happened at 7 a.m then RPO us for 1 hour and we will use 1 hr of data.

2)Amount of maximum data loss.

3)Business provides an RPO value.

4)influence technical solution and close.

RTO: Recovery time objective.

1)Time between the DR event and full recovery.

Ex: Disaster happened at 6 a.m and you took 30 mins to restore back the DB. Then 30 mins we can say as RTO.

2)Influenced by process, staff, tech and documentation

Cost optimization:

1)Have you been involved in cost optimization?

Yes i have been part of cost optimization meetings.

2)We have 7 ec2 which is running 24/7,365 days and we can expect traffic only few days? What steps can be implemented to reduce the cost and make sure our applciation is HA?

1)By implementing **ASG** we can reduce the number of instances.

2)By using Reserved instances instead of on demand instances.

How to secure aws platform?

1)By using **MFA, IAM Roles.**

2)By using **cloudtrail** to monitor the **activity of aws.**

3)By configuring firewalls like **security group, NACL, Web application firewall**.

4)By using **Aws shield** to prevent DDOS attack.

5)By using **Aws Guard duty** to prevent from vulnerabilities or malicious attacks.

Chaos Engineering - aws fault injection service.

Revision_5_Terraform:

1)Terraform version?
Latest is 1.6.6 or 1.5

2)What resources you have created using terraform?
Ec2, VPc, s3 bucket, dynamodb, RT, ecs, ecr, IGW, subnets.

3)Where are you storing you state file?
Remote backend and the service is **aws s3.**

4)How to prevent multi user execution in terraform?
We use **dynamodb locking** to prevent multi user execution.

5)What will happen if multiple users execute the main.tf at same time?
There will be configuration drift.

6)My state file has been deleted, how to recover it or backup it?
We can use terraform import command to get the state file,
because already we have the complete configuration related to infrastructure in main.tf

7)What is terraform state file?
Blueprint of Infrastructure managed by Terraform.

8)Have you integrated terraform with CICD tool?
yes we can do that by using terraform plugin in jenkins but this is not recommend.
We will face issue with plugin so we have manually executed main.tf from visual studio code.

9)What are providers in terraform?
Providers are nothing but plugin which helps us to communicate with the resources. It will have all the api calls required to communicate with the resource.
Ex: Aws, Azure, GCP, Docker, k8s.

10)What is Resource in terraform?
Resource will be the infrastructure which needs to be created. **Ex: Ec2, S3, VPc**

11)How many types of provisioners are there?
Two types
1)**Local** (If we want to execute any command in local machine)
2)**Remote** (If we want to execute any command in Remote Server)
Example:
Local: To execute any command using localfile resource.
Remote: While provisioning ec2 and want to update any configuration then i can use remote provisioners.

12)What are the challenges or issues while working on terraform?

1)versions of providers not supporting.

2)If we have different environments then it might be difficult to manage or increase in the resources while creating.

(Modules or workspaces will be used to handle this)

13)What are modules in terraform?

By using modules, we can reuse the same piece of code again and again for different purposes.

14)What is taint in terraform?

Taint will help us to recreate our resource again when we execute terraform apply. Taint we can see if resource creation is not successful or we can mark the resource as tain manually.

15)List few terraform commands?

Terraform init --> It will initialize and download all the dependencies.
Terraform plan --> It is a dry run to check how infrastructure will be created.
Terraform validate --> To check for the code.
Terraform apply --> To create resources based on main.tf
Terraform refresh **-->** Use to sync terraform with the actual infrastructure.

16)How to read the details of existing infrastructure using terraform?

We can use data source to read the details of existing infrastructure.

17)Lifecycle rules of terraform?

1)Create_before_destroy
2)Ignore_changes
3)prevent_destory

18)I want to provision multiple ec2 instances using one resource block,how to do that?

We can use meta-arguments like count and for-each.

19)Example Terraform template?

resource "aws_instance" "test-instance"{
ami_id = ami-xxxx
instance_type = t2.micro key_name = test
tags {
name = ""test-server
}
}

20)Terraform has provisioned one ec2 t2.micro and it has been updated to t2.large. what will happen if someone try to execute terraform apply?

No changes will be made because terraform will validate the state file and in state file it is shown as **t2.micro only.**

21)We have different enviroments to be managed by terraform,how can we do this?
Terraform workspaces.
Isolated Group.

22)Terraform uses which langauge?
HCL - hashi corp

23)Terraform is developed using which language?
Go langauge.

24)Which deployment strategy will be used by terraform?
Terraform by default will use recreate strategy.

25)How to store secret credentials in terraform?
1)By using hashicorp vault
2)By using environmental variables
3)By using aws secret manager

Docker:

1)Difference between virtualization and Containerization?
Virtual machines are dependent on operating systems.
Virtual machines will take longer time to start and shutdown.
Virtual machines are higher in size and we cannot move them from one environment to another environment.

Containerization are independent of operating systems.
Containers will start within fractions of seconds.
Containers can be moved from one environment to another environment easily.

2)Have you created dockerfile?
yes.

3)What is a Dockerfile?
Docker file will have set of instructions to create docker image.

4)Different modules used in Dockerfile?
FROM -- To download the base image
MAINTAINER - To provide the author name
RUN - To install any software/dependecies.
COPY - To copy file from docker engine to docker image

ADD - To download file from internet and also to extract the tar file.
EXPOSE - Expose a specific port number
ENV - To set the environemtnal variables
ENTRYPOINT - To start or execute the command at the runtime of container
CMD - To execute the command at the runtime of container.
WORKDIR - To change the path when we login inside the container

5)Difference between ADD and COPY?

COPY will copy the file from docker host to docker image.
ADD will download the file from internet and also to extract the tar packages.

6)Difference between CMD and ENTRYPOINT?

Both are executable at the run time of container.
CMD is overridable and **entry point** is not overridable.
We can have multiple CMD's but the last one will get executed.

7)How to reduce the size of docker image?

1)By using Multistage dockerfile.
2)By using base images as alpine.
3)We can avoid unwanted layers in dockerfile.

8)Command to create docker image?

docker build -t "tag_name".

9)Difference between single stage Dockerfile and Multi stage Dockerfile?

Single stage dockerfile will have only one stage with installing, downloading source code and running the package.
Multi stage docker file will have different stages,

1)Downloading source code
2)Compile the source code
3)copy the source code.

10)Can you explain your Dockerfile?

We have used alpine as the base image and customized with our own tomcat.

11)Why you have used custom tomcat image?

Because the official image of tomcat is about **750 Mb** and by customizing with alpine has reduced the size to **150MB** and also we have removed unwanted layers.

12)Can you write Dockerfile used in your project?

FROM alpine:3.10
MAINTAINER Techie_Horizon
RUN mkdir /usr/local/tomcat/
WORKDIR /usr/local/tomcat
RUN apk --no-cache add curl && \ apk add --update curl && \

curl -O https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.73/bin/apache-tomcat-9.0.73.tar.gz
RUN tar -xvf apache*.tar.gz
RUN mv apache-tomcat-9.0.73/* /usr/local/tomcat/.
RUN rm -rf apache-*
COPY sample.war /usr/local/tomcat/webapps
RUN apk update && apk add openjdk8
WORKDIR /usr/local/tomcat
EXPOSE 8080
CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]

13)How to run a docker container?
Docker container run -itd -p 8080:80 <image_name>

14)How to login to the container?
docker exec -it <container_id> /bin/bash

15)How to delete multiple containers at the same time?
docker container prune

16)How to kill a running container?
First, we need to stop the container "**docker container stop <container_id>"** then we can kill the container.

17)How to delete docker image?
Docker image rmi <image_name>

18)How to get the detail information of image or container?
docker inspect <image_id>; docker inspect <Container_id>

19)What is the use of docker compose?
To run multiple containers at the same point of time.

20)Difference between docker and docker-compose?
Docker is used to create containers.
Docker-compose is used to run multiple containers.

21)Have you worked on Docker-compose?
No.

Common Errors:

1. HTTP Error Codes
400 Bad Request: The server cannot process the request due to a client error (e.g., malformed request syntax).
401 Unauthorized: Authentication is required and has failed or has not been provided.

403 Forbidden: The server understands the request but refuses to authorize it.
404 Not Found: The requested resource could not be found.
500 Internal Server Error: A generic error occurred on the server.
502 Bad Gateway: The server received an invalid response from an upstream server.
503 Service Unavailable: The server is currently unable to handle the request due to temporary overload or maintenance.

2. Git Error Codes

error: src refspec branch_name does not match any: The specified branch does not exist.
fatal: Authentication failed: Credentials are incorrect or missing.
fatal: unable to connect to remote repository: Network issues or incorrect repository URL.
error: Your local changes to the following files would be overwritten by merge: Local changes are preventing a merge.

3. Jenkins Error Codes

ERROR: Unable to resolve class: Jenkins cannot find a class in the pipeline script or plugin.
java.lang.OutOfMemoryError: Jenkins has run out of memory.
java.lang.NullPointerException: A null object reference was used in the pipeline or job.
Failed to connect to repository: Issues with SCM integration or credentials.

4. Docker Error Codes

Error response from daemon: pull access denied: The image does not exist or you lack permission to access it.
docker: Error response from daemon: driver failed programming external connectivity: Networking issues with container ports.
Error: No such container: The specified container does not exist.

5. Kubernetes Error Codes

CrashLoopBackOff: A container is repeatedly crashing.
ImagePullBackOff: Kubernetes cannot pull the container image from the registry.
Pending: A pod cannot be scheduled due to insufficient resources or other issues.
ErrImagePull: The image specified in the pod specification cannot be pulled from the registry.

6. Maven Error Codes

[ERROR] Failed to execute goal: Issues with Maven goals (e.g., compile, test, package).
[ERROR] Could not find artifact: Dependencies are missing or cannot be resolved.
[ERROR] Invalid version range: Version specified is not valid or not available.

7. Ansible Error Codes

fatal: [hostname]: UNREACHABLE! Ansible cannot connect to the target host.
failed: Task failed due to an issue in the playbook or role.
could not find a suitable distribution: Issues with package management on the target host.

Kubernetes:

1. Difference Between Deployment and Job
Deployment: Manages continuously running applications, ideal for apps that need updates and scaling.
Job: Runs pods for tasks with a specific duration, suitable for short-lived processes.

2.Running a Pod at a Specific Time
Use a CronJob to schedule pods to run at specific times, like a Linux cron job.

3.Securing a Kubernetes Cluster
RBAC (Role-Based Access Control): Defines access permissions for different users.
Service Accounts: Assign permissions to users and applications.
Network Policies: Control which pods can communicate with each other.
Secrets: Securely store sensitive data, like passwords.
Monitoring: Continuously check the health of pods and the cluster.

4.Headless Service
A Headless Service allows direct communication with a pod rather than using a stable IP address.

5.Alpha and Beta API Versions
Alpha: Initial testing phase for new features.
Beta: More stable version, closer to a final release.

6.Difference Between Deployment and StatefulSet
Deployment:
For stateless apps that don't need to store data.
Pods are created in any order, without a fixed IP address.
StatefulSet:
For stateful apps that need persistent data storage.
Pods are created in sequence and retain unique identities.

7. Difference Between PV and PVC
PV (Persistent Volume): Provides storage for data.
PVC (Persistent Volume Claim): Requests specific storage resources for a pod.

8. Types of Volumes
Examples include **emptyDir, hostPath, NFS, configMap, secret, and persistentVolume**.

9. Replica
Defines the desired number of pod instances running in a deployment.

10. DaemonSet
Ensures one instance of a pod runs on all (or selected) worker nodes.

11. HPA (Horizontal Pod Autoscaler)
Adjusts the number of pods based on resource usage, like CPU utilization.

12. Node Autoscaler
Automatically scales the number of nodes up or down based on resource needs. Note: You mentioned you haven't worked with Node Autoscaling.

13. Probes
Liveness Probe: Checks if a pod is functioning. If not, Kubernetes restarts it.
Readiness Probe: Ensures a pod is ready to receive traffic.

14. Init Container
Performs setup tasks before the main application container starts.

15. Sidecar Container
Adds extra functionality to the main container, like logging or monitoring.

16. Types of Services
ClusterIP: Only accessible from within the cluster.
NodePort: Exposes the service on a specific port on each node's IP.
LoadBalancer: Creates an external load balancer for public access.

17. Ingress Resource and Controller
Ingress Resource: Manages external access to services with routing rules.
Ingress Controller: Implements load balancing and traffic management (e.g., NGINX).

18. Metrics in Kubernetes
Use labels and selectors to collect and monitor resource usage metrics.

19. Namespaces
Divides the cluster into isolated environments to organize resources.

20. Node Affinity
Node Affinity: Defines rules for which nodes a pod can be scheduled on, based on labels.
Types of Affinity:
Preferred During Scheduling: Tries to schedule on preferred nodes.
Require During Scheduling: Ensures pods run only on specific nodes.

21. Resource Limits and Quotas for Pods
Resource limits control the maximum CPU and memory a pod can use.
Resource quotas set limits on total resource usage within a namespace.

22. Stopping a Deployment
You can pause, scale down to zero replicas, or delete a deployment to stop it.
kubectl rollout pause deployment <deployment-name>
kubectl scale deployment <deployment-name> --replicas=0
kubectl delete deployment <deployment-name>

23. Max Surge and Max Unavailability
Max Surge: Allows extra pods during updates.
Max Unavailability: Permits a specific number of pods to be offline during updates.

24. Access Control with Namespaces
Use RoleBindings and Cluster Role Bindings to grant user access to specific namespaces.

25. Pod in Kubernetes
A Pod is the smallest deployable unit in Kubernetes, containing one or more containers.

26. Kubernetes Architecture
Consists of a control plane (handles scheduling) and worker nodes (run the workloads).

Master Components:
1) Kube api server
2) etcd
3) Controller
4) Scheduler

worker node components:
1) Docker container runtime
2) Kubelet agent
3) Kubeproxy

Note: kubectl is a command line tool to execute k8s commands.

Api server: Acts as frontend of k8s, authenticate user, authorization and api server is the only gateway to communicate with our k8s cluster.

Etcd: Cluster brain, which stores the information related to cluster, node in the form of key value pair.

Scheduler: Responsible to schedule pods on the node machine.

Controller manager: controller is the brain behind orchestration, if nodes, container, endpoint goes down then controller will work behind to bring them up.
Container runtime: any container service to start, stop container (Docker).

kubelet: Kubelet run on all the machines, kubelet make sure that the containers are running as expected on node machines.

KubeProxy: It maintains network rules on your nodes and enables network communication to your Pods.

27. Handling an etcd Failure
If etcd (cluster data store) is down, the control plane won't function, causing the cluster to fail.

28. Monitoring with Prometheus and Grafana
Prometheus gathers metrics, and Grafana visualizes them for monitoring.

29. Troubleshooting Failing Pods
Check logs, events, probes, and resource usage to identify issues.

30. Pod Disruption Budgets (PDBs)
PDBs limit how many pods can be disrupted at once to ensure application availability.

31. Secrets vs. ConfigMaps
Secrets: Store sensitive information like credentials.
ConfigMaps: Store non-sensitive configuration data.

32. Taints and Tolerations
Taints prevent pods from running on specific nodes.
Tolerations allow pods to run on tainted nodes if needed.

33. Node Selector
Node Selector ensures that pods are scheduled only on specific nodes with matching labels.