**1. What is GIT?**

GIT is the most popular and widely used distributed version control system (DVCS) and source code management (SCM) tool with lightning speed, data integrity and efficiency. Git snapshots each and every commit you do, it becomes easy to track the version you saved, and in case you can revert to whichever version you need.

**2. What is a Repository in GIT?**

Git repository is nothing but a location where you can store all files related to your project, when you commit your code. Normally a repository contains a directory named ".git" along with your project files, where git keeps all of its metadata of the repository. The content in the ".git" is private to git. For ‘bare-repository’ ".git" directory will not be there but git’s metadata store in the same directory where you store your projects files.

**3. What are the advantages of using GIT?**

a) Git is Open source  
b) Better branching system  
b) High availability, very reliable  
c) Only one .git directory per repository  
d) Superior disk utilization and network performance  
f) Light weight and Lighting speed  
e) Collaboration friendly and compatible with previous version control system  
f) Any sort of projects can use GIT.

**4. What is the command to write a commit message?**

Command to write a commit message is "**git commit –m < commit message >**". The "–m" on the command line instructs git to commit the new content of all files that have been modified or created. You can use "git add " before git commit –m if new files need to be committed for the first time. To commit only modified files, use this command "**git commit –am < commit message >**".

**5. What language is used in GIT?**

Its ‘C’ language, GIT is fast as ‘C’ language makes this possible by reducing the overhead of runtimes associated with high level languages.

**6. What is the difference between GIT and SVN?**

Difference between GIT and SVN are

| GIT | SVN |
| --- | --- |
| Git is a Distributed Version Control tool | SVN is not a Distributed Version Control tool |
| Git Commits are unchangeable | A tag can be treated as a branch and multiple versions can be committed |
| Clients can clone entire repositories on their local systems | Can't Clone Version history |
| Commits are possible even if offline | Offline Commits are not possible |
| Push/pull operations are faster | Push/pull operations are slower |
| Git is less preferred in case of extremely large files or frequently changing binary files | SVN allows to multiple projects with in a same repository |
| Works are shared automatically by a commit | Nothing is shared automatically |

**7. Why is GIT better than Subversion?**

Git and SVN are each viable workflow and version control systems, Git will allow you to store versions of a project, and it has stage area where you can store your uncommitted changes and you can also do patch staging in case you and your co-developer are working on same file then you can patch your work to the file by using ‘**-p**’ flag while **git add**. And Git has a better branching system and git stash too.

**8. In GIT, What is "Index" or "Staging Area"?**

Before committing your changes to git you can add changes to git using **git add < file\_name >**, it is reviewed as an intermediate area known as ‘Staging Area’ or ‘Index’.

**9. Is GIT better than Subversion?**

Can't say yes or no to this. It’s up to you, Git and SVN are each viable workflow and VCS. There are lots of grounds where both differ with each other, of course git is latest to SVN includes advanced features.

**10. What is the function of ‘GIT PUSH’ in GIT?**

‘GIT PUSH’ publishes local repository changes into remote repositories so that others can take those changes.

##### 11. What is GIT stash?

If you are in the middle of a job and you need to jump to another job, use the '**git stash**' command. git stash records the current state of your working directory & index, and gives you a clean working directory which matches the HEAD. So you can jump to another job, by this you won't lose your changes/edits. And you can un-stash to get your previous changes back to the working directory. To see what changes you stashed use '**git stash list**' this will list you the changes and '**git stash pop**' will get back your changes to the working directory.

##### 12. What is GIT stash drop?

When you don't need the stashed changes and want to remove it from the stashed list, run the git '**git stash drop**' command. By default it will remove the last added stash item, and to remove a specific item you include it as an argument '**git stash drop < stash\_name >**'.

##### 13. How will you know if the Git branch has merged to master or not?

This command '**git branch --merged master**' will list the branches that have been merged into the master. Use the '**-a**' flag to show local and remote branches and '**-r**' to view only remote branches.

##### 14. What is a Git Clone?

The git clone creates a copy of an existing remote repository into a local, command to clone a git repository is "**git clone < remote\_repo\_url >**".

##### 15. What is the function of Git Config?

As we know git is a version control tool it keep track of all commits done by each and every user, so git should have user data like name and email, git never allow user to commit without this details, so user need to configure git using this command,  
**git config --global user.name "your\_name"**  
**git config --global user.email "your\_email"**

##### 16. What does the commit object contain?

A 'commit' points to the specific version of the project looked like at a certain point of time.  
   a) A set of files/directories, representing the state of a project at a given point of time.  
   b) Reference to parent commit(s) objects  
   c) Time stamp.  
With all these info git commit has a unique id known as SHA id or SHA value.

##### 17. How to create a local repository in Git?

Create a directory for the project if it does not exist, and go into the directory then run the command ‘**git init**’. A directory by name **.git** will be created in the project directory.

##### 18. What is ‘HEAD’ in git and how many heads can be created in a repository?

HEAD is simply a reference to the last commit in the current working branch. In every repository, there will be a default head known as "Master". A repository can contain any number of heads. When you checkout to any branch the HEAD revision changes to the latest commit of the new branch. We directly will not create head but when we commit, git creates a reference to that commit known as head.

##### 19. What is the purpose of branching in GIT?

The purpose of branching in GIT is that you can create any no.of branches and jump between those branches. It will allow you to go to your previous work in one branch keeping your recent work intact in another branch. For example you can create a branch for main release and a branch for bug fix after completion of job in one branch. You can merge the code to master by this. You can work parallely with multiple jobs and you will have a safe code in master.

##### 20. What is the common branching pattern in GIT?

A common way to create a branch in GIT is to maintain one as "Main" branch and create another branch to develop new features. Generally used branches are Feature branch, Release branch and Hotfix branch. This pattern is useful for multiple developers working on the same project.

##### 1. How can you bring a new feature to the main branch?

To bring a new feature from any branch to the main branch, you can use the command "git merge" or "git pull command" to get new features from the same branch.

##### 22. What is a ‘conflict’ in git?

Conflicts come into picture when a user has some changes in local and he tries to pull the latest code from remote, if the remote varies in content of the same file then conflict arises, and other thing when user has some changes in local branch and he tries to merge another branch which has different content in it.

##### 23. How can GIT conflicts be resolved?

To resolve the conflicts in git, edit the conflicting files, discuss with your team and keep whatever required and remove what is not needed. Then git add, git commit and push. Git knows that you are in the middle of a merger, so it sets the parents of the commit correctly.

##### 24. To delete a branch what is the command that is used?

Once you merge your development branch into the main branch, you don’t need a development branch. To delete a branch use the command to delete local branch "**git branch -d < localBranchName >**" to delete remote branch "**git push origin --delete < remoteBranchName >**".

##### 25. What is another option for merging in git?

**"Rebasing**" is an alternative to merging in git.

##### 26. What is the syntax for "Rebasing" in Git?

Syntax used to rebase the branch is '**git rebase**'

##### 27. What are the differences in ‘git remote’ & ‘git clone’?

‘**git remote add origin URL**’ just adds remote repository's origin to your git config that specifies a name for a particular URL. While, ‘**git clone URL**’ copies the entire git repository located at the URL to the local.

##### 28. What is GIT version control?

Version control allows you to track the commit history of files and to revert the changes. Each version captures a snapshot of the file system at a certain point of time. And it has all the files and their histories in a repository.

##### 29. Mention some of the best graphical GIT clients for LINUX?

Here are Some best GIT client for LINUX:  
a) Git Cola  
b) Git-g  
c) Smart git  
d) Giggle  
e) Git GUI  
f) qGit

##### 30. What is Subgit? Why use Subgit?

Subgit is a tool to migrate from SVN to Git. Some of its features are:  
a) It supports all git and all sub-version features  
b) There is no requirement to change the company’s infrastructure  
c) It is much better than git-svn  
d) Provides genuine & smooth migration experience.

##### 31. What is the function of ‘git diff’ in git?

A **$ git diff** command lists you the changes between the commits.  
For example: $ **git diff < commitId1 > < commitId2 >**

##### 32. What is ‘git status’ used for?

‘**Git Status**’ shows your changes status in the working directory and the index area, it is helpful in understanding a git more comprehensively.

##### 33. What is the difference between the ‘git diff ’and ‘git status’?

‘**git diff**’ is similar to ‘git status’, but "git diff" shows the differences between various commits, and "git status" shows the difference between local repo and the working directory and index.

##### 34. What is the function of ‘git checkout’ in git?

A ‘git checkout’ updates your directories or specific files of your working tree. And to switch from one branch to another.  
To switch among branches "**git checkout**"  
To jump to specific commit "**git checkout**"

##### 35. What is the function of ‘git rm’?

To remove the file/s from the staging area and also from the working directory ‘**git rm**’ is used.

##### 36. What is the function of ‘git stash apply’?

It is possible to start your work form where you have left, ‘**git stash apply**’ command is used to bring back the saved changes to the working directory.

##### 37. What is the use of ‘git log’?

GIt Log command shows you the commit history. To get specific histories of your project, there are some commands listed below:  
$git log //shows complete history of your repository files.  
$git log --oneline //shows all history but in one line.  
$git log -n5 //shows last 5 commits history.  
$git log --follow //shows complete history of that file even filename has been changed.  
$git log --author //shows complete history committed by that user

##### 38. What is ‘git add’ used for?

‘**git add**’ adds file/s from the working directory to your index area.

##### 39. Is it to create an additional commit than to amend an existing commit?

Of Course creating a new commit is suggested then to edit the existing commit because, Amend operation will destroy the state of existing commit. If it is just a commit message changed then that is not a problem. But if the contents are changed then chances of eliminating something important remains more.

##### 40. What is the function of ‘git reset’?

The command ‘**git Reset**’ is used to reset your index as well as the working directory to the state of your last commit. By doing this you left with zero changes in your working directory.

##### 41. What is git Is-tree?

The command ‘**git Is-tree**’ is used to represent a tree object including the mode, the name of each item and the SHA-1 value of the blob or the tree.

##### 42. How is git instaweb used?

‘**Git Instaweb**’ browse your local repository to gitweb, which is a GUI.

##### 43. What does ‘hooks’ consist of in git?

Git hooks are nothing but scripts which triggers before or after running the Git commands. For example, a post-commit script will execute after you run a git commit command.

##### 44. Explain what is a commit message?

Commit message actually has the reason for that particular commit. Which appears when you commit a change. Git provides a text editor where you can enter the reason for your commits.

##### 45. How can you fix a broken commit?

To fix any broken commit, you will use the command "git commit --amend  
. By running this command, you can fix the broken commit message in the editor then save and quit.

##### 46. What is a ‘Bare Repository’ in GIT?

To coordinate with the distributed development and developers team, especially when you are working on a project from multiple computers, 'Bare Repository’ is used. A bare repository comprises a version of the history of your code.

##### 47. Name a few Git repository hosting services?

Pika Code, Visual Studio, GitHub, Git Enterprise, SourceForge.net

##### 48. Difference between Git & Github?

**Git**: it is a distributed version control system (DVCS) used to track your source code changes during software development. Git is Lightning speed, Light weight, Open Source and supports distributed, non-linear workflows.  
**GitHub**: It is a cloud service to provide a Web-based GUI to browse & manage git repositories.

##### 49. What are the benefits of using the Version Control System?

Each and every commit you made is logged with a commit message.  
All the team members can work on any file at any time and can commit their changes to the version control system (VCS).  
Each change you commit is saved separately and can be reverted to whichever commit you require.  
Stores details information in the commit object regarding commit like, TIme stamp, content, commit message, and user details.

##### 50. What is the difference between git pull and git fetch?

Git Pull "**git pull URL**" pulls new changes/commits from your Remote repository to your local Repository.  
Git Fetch "**git fetch**" pulls all new changes/commits histories but not content from remote repository to local repository. If you want those changes in your local repository, you need to "**git merge**". Then only your local repo will get updated.  
**Git pull = git fetch + git merge.**

##### 51. What work is restored when the deleted branch is recovered?

The files which were stashed and saved to index and committed will be recovered back. Any untracked files will be lost.

##### 52. How do you find a list of files that have changed in a particular commit?

**git diff-tree --no-commit-id --name-only -r**  
Here –no-commit-id will hide the commit ids from listing in the output, and –name-only will only list the file names, instead of their full paths.

##### 53. How to undo or revert the last commit?

View commit history using the "**git log**" command. Now you are at HEAD (latest commit).  
You need to check out the previous working commit. For this you have to "**git checkout**".  
Now you should be at detached HEAD and you can undo the last commit using "**git revert**".  
Then make sure that your head should always be at latest, so do "**git checkout**". That's it.

##### 54. Tell me the difference between HEAD, working tree and index, in Git?

The HEAD always points to the last commit in your current branch.  
The working tree is the workspace or working directory, where you write/add/edit the code.  
The index is also known as staging area. which stores your works of working tree in it with the user details, SHA Id, timestamps, and the file name. It is not another directory and does not contain a copy of files in it. Now if you commit the files/directories which are in the staging area are moved to your local repository.

##### 55. What is an Ansible?

Ansible is an open-source software, powerful and agentless automation platform. It is used while deploying an application using ssh with zero downtime. It is also used in Configuration management, Cloud Provisioning, Application Deployment, Intra-Service Orchestration easily. It runs on many Unix-like systems, and can manage configurations of both Unix-like systems as well as Windows systems. And YAML is its declarative script. It is developed in Ruby, Python and PowerShell languages by Michael DeHaan and acquired by Red Hat in 2015.

##### 56. What is configuration management?

Managing configurations of the project like in IT-Infrastructure identifying the configuration, controlling configuration and configuration audit. So this is like a practice of managing and automating all the configurations required for the applications, so that the application can run seamlessly on readily tuned infrastructure. And this reduces the product release time as well.

##### 57. What is an Ansible Task?

Ansible Tasks are small blocks of code in the playbook that can be used to execute any job. For example, if you want to install a package or update a software, you can follow the below code notation, it is a single task to install git software:  
  
tasks:  
    - name: Installing Git Application  
      yum: git  
      state: present  
  
And these tasks are reusable, and named as ansible roles and shared in Ansible Galaxy.

##### 58. What are the advantages of using Ansible?

The main three advantages of Ansible are:

* **Agentless**: Ansible is fast and performs all functions over SSH and doesn't require agent installation. As long as the machine has ssh and python installed.
* **Very Simple**: Has very simple architecture, simple installation and easily manageable.
* **Idempotent**: Architecture of Ansible is structured around the concept of idempotency. Means the things which you do on a daily basis regularly, can be automated.
* **Declarative**: No procedural approach, Ansible is declarative, needs to be defined at a high level in Yaml and Ansible makes the things done.
* **Ansible Modules**: Ansible has so many built in modules, which gives Ansible more power to execute various variety of tasks.
* **Ansible Galaxy**: A website where Ansible users share their customized roles to make it reusable.

##### 59. What is ansible-playbook?

The Ansible Playbooks contains details of remote hosts, user variables, tasks, handlers in it. Playbook may have one or more tasks and these tasks are executed by Ansible. Usually playbooks are .yaml files. Here is a sample playbook:  
---  
  - name: Install Git  
    hosts: linux  
    gather\_facts: false  
    become: true  
    vars:  
         version: 2.19.1  
    tasks:  
       - name: Install yum package  
         yum: git  
         state: present

##### 60. Ansible Playbooks vs Roles?

| Roles | Playbooks |
| --- | --- |
| Roles are reusable subsets of a Play. | Playbooks may contain one or more Plays. |
| A Role is a set of tasks to be done. | Defines hosts, where roles should be executed. As it knows inventory. |
| AExample: general, git. | Example: site.yml, myplay.yml. |

##### 61. What is inventory or host-file in ansible?

Inventories are the host files where information of target servers are written. Inventory also known as host-file. By default this file is located in "/etc/ansible/hosts". It contains the group of the servers, IP’s, and connection type etc.

##### 62. What is an Ansible Galaxy?

Ansible Galaxy is a GUI service that lets Ansible users share their roles and modules. The Ansible Galaxy command line tool comes with normal Ansible. It is used to install roles from Galaxy or from a SCM system like GIT. Use this command to get roles from the Galaxy:  
$ ansible-galaxy install username.role\_name

##### 63. Compare Ansible vs Chef vs Puppet?

| Ansible | Chef | Puppet |
| --- | --- | --- |
| Ansible Installation is very easy | Time consuming & complicated because of Chef Workstation | Time consuming due to certificate signing between master and agent |
| Simplest Technology | Complex than Ansible | Complex than Ansible |
| Written in YAML Script | Written in Ruby DSL language | Written in Ruby language |
| Agentless setup | Has an Agent | Has an Agent |
| Configures any machine which has SSH and Python installed. | Configures only those machines which has Chef Agent installed | Configures only those machines which has Puppet Client installed |

##### 64. What is the Use of Ansible?

From the day one Ansible focused on multi-tier deployments. So it is used to manage and deploy applications to remote nodes. It manages how the entire IT-Infrastructure inter relates. You were using commands or scripts to manage infrastructure or some automation scripts which would take a lot of time and effort. Now Using Ansible there are reusable roles and inventories, you just need to write a YAML script and you are done. By this you can automate repetitive tasks.

##### 65. What is Ansible tower?

Ansible Tower is a commercial product(Ansible with support and extra features) from RedHat . It is used in simplifying the job of ansible automation. Ansible is very easy to use in towers as it acts like a hub for all automation tasks. The Ansible tower is free for usage of upto 10 nodes.

##### 66. What are Ansible vaults?

The Ansible vaults are used to keep your sensitive data like passwords or keys in encrypted files, rather than as plaintext in playbooks or roles so that the data can be protected. Not only about protecting data but also makes it access into the playbooks. The files can be encrypted and unencrypted as the Vault is implemented with file-level granularity. These are very user friendly.

##### 67. What about Ansible architecture?

Ansible has a very simple architecture. It has control over the configurations of your IT-Infrastructure. Automates the cloud provisioning, configuration management, infrastructure as a code, application deployment, intra-service orchestration, and many other IT needs. It has no agent, just needs ssh and python installed in target servers. The SSH protocol and python interpreter enables Ansible to copy modules and executes them in target servers. At last it removes copied modules from target serves.  
The main components of Ansible architecture are:

* Ansible Modules
* Plugins & API
* Inventory/hosts file
* Playbooks

##### 68. How does Ansible work? Please explain in detail?

There are so many configuration management tools like Puppet, Chef, CEFengine, Salt, etc. And the most popular tool is Ansible, in this tool infrastructure is categorized into two type:  
1. Ansible Server  
2. Target Servers  
As Ansible is an agentless tool so it doesn’t require any installations on target servers (remote nodes). So setup and managing nodes is very simple. Ansible can handle huge no.of nodes over SSH connection and entire operations can be executed by one single command “ansible”. Playbooks written in YAML contain one or more play, each play has one or more tasks.  
$ ansible-playbook playbook\_name.yml

##### 69. Do we have any Web Interface/ Rest API etc for Ansible?

Yes, Ansible Inc makes a great efficient GUI tool. It is very easy to use.

##### 70. What is the use of –start-at-task in ansible?

start-at-task option, will start executing the play from the task you specify and subsequent tasks are executed. The prior tasks are skipped.

##### 71. Explain Ansible facts?

Ansible Facts are unchangeable information about the remote hosts. Ansible collects almost all the information about the target hosts as it runs a playbook. The task of collecting this remote system information is called Gathering Facts. To generate facts, ansible runs the setup module. And the command is  
$ ansible- m setup hostname  
this will print out a dictionary of all the facts available for that host.  
You can also see all the facts using the below command  
$ ansible all- m setup

##### 72. What is Role in Ansible?

Roles are collections of certain tasks variables and handlers. These Roles are shared over Ansible Galaxy to be reusable/redistributable for other Ansible users as well.

##### 73. What are the different components of Ansible?

Ansible consist of the following components:

* Inventories
* Modules
* Variables
* Plugins & APIs
* Hosts
* Playbooks
* Facts
* Roles
* Vault and
* Handlers

##### 74. How do I handle different machines needing different user accounts or ports to log in with?

Let say, suppose these hosts have different ports, username and connection type as one is linux and other is windows machine.  
Setting variables in the inventory file is the easiest way for this as shown:  
  [web\_servers]  
  abc.example.com   ansible\_port=8000   ansible\_user=user1   ansible\_connection=ssh  
  xyz.example.com   ansible\_port=8001   ansible\_user=user2   ansible\_connection=winrm

##### 75. How to see all the inventory variables that are defined in the host?

To see all the inventory variables execute the following command:  
$ansible -m debug -a var=hostvars[web\_servers] localhost

##### 76. What is that Ansible can do?

Ansible can do the following:

* Configuration management
* Application deployment
* Task automation
* IT orchestration

##### 77. In which language Ansible is written in?

Ansible is written in Python, PowerShell and in Ruby by Michael DeHaan.

##### 78. What is Red Hat Ansible?

Both Ansible and Ansible Tower are Red Hat products, both are complete automation platforms with the following features:

* Cloud Provisioning
* Deploying applications
* Orchestration
* Manage IT Infrastructure
* Configuration Management
* Networking

But Ansible is an open source and Ansible Tower is a commercial enterprise product that comes with RedHat Support and with some extra features like GUI, etc.

##### 79. Is Ansible an open source tool?

Yes, Ansible is an open source tool which is a powerful automation software tool for configuration management from RedHat.

##### 80. What are Ansible server requirements?

Ansible can configure windows systems, but Windows systems can’t be an ansible server without virtualization. Ansible Server should be a Linux machine with ssh and python 2.6+ version installed.

##### 81. What is the best way to make content redistributable?

Create Roles for a set of tasks and share it over Ansible Galaxy, so that it can be reusable/redistributable.

##### 82. How to generate encrypted passwords for the user module?

Using Ansible ad-hoc command you can generate encrypted passwords for modules:  
ansible all -i localhost -m debug -a msg={{ my\_password | password\_hash(sha512, my\_secretsalt) }}

##### 83. How can you connect to other devices within Ansible?

Whatever might be the other device you get the IP of that machine and save it to the inventory file and of course you have a ping module.  
$ ansible - m ping hostname

##### 84. Explain Module in Ansible?

Ansible has a huge number of modules (called the ‘module library’) as it can be written in any language. Actually Ansible works by connecting target servers and copying and executing a small program in those target servers those small programs are called Ansible modules. Playbook is a high level declarative script, you just have to invoke modules and modules does the entire thing happen.

##### 85. Can you build your own modules with Ansible?

Yes, we can create our own Ansible modules. As it can be written in any language it becomes easy for everyone to write their own custom modules. And these are capable of controlling system resources, like services, directories, files, and handling executing system commands also.

##### 86. How can you find information in Ansible?

Ansible is very simple and powerful, to get all the information of all target server just need to execute the following command:  
ansible all -m setup

##### 87. What is ask\_pass in ansible?

This controls whether an ansible playbook should prompt for a password by default or not. Usually, it won’t ask. You can set True or False for it in /etc/ansible/ansible.cfg file. ask\_pass = True/False

##### 88. What is ask\_sudo\_pass?

This controls whether an ansible playbook should prompt for a sudo password by default or not. Usually, it won’t ask. You can set True or False for it in /etc/ansible/ansible.cfg file. ask\_sudo\_pass = True/False

##### 89. What is ask\_vault\_pass?

This controls whether an ansible playbook should prompt for a vault password by default or not. Usually, it won’t ask. You can set True or False for it.

##### 90. Where is the unit testing available in Ansible?

Unit tests are available in test/units. To test any file in ansible follow the command:  
$ ansible-test units --tox

##### 91. Explain in detail about ad-hoc command?

Instead of writing playbooks you can just execute an ad-hoc command. For example, if we want to copy a file to all hosts in a particular group (stageservers). In simple terms, if you have only one task to do, then why to write playbooks. As in mentiond example. For repeated tasks list of actions can be stored in playbooks and can run whenever needed.

##### 92. How Can you submit a change to the Documentation in Ansible?

Ansible documentation is shared over GitHub, there is an option "Edit on GitHub" on the right top corner of docs.ansible.com website. So through GitHub we can submit a change in Ansible Documentation.

##### 93. What is Jenkins?

Jenkins is an open source automation tool written in Java with plugins built for continuous Integration Process. Jenkins is used to build, test and deploy your software projects continuously and it integrates changes to the project continuously. So it is also known as CI/CD tool. There are other similar tools in the market but Jenkins stands out with its vast plugin collection, easily configurable and of course it is free of cost.

##### 94. What is Continuous Integration?

Continuous Integration is one of the best practices followed in Devops. Monitors the new changes coming into VCS like GIT, checkouts the source code, triggers the Build and generates a new artifact (with the new code) without any human intervention. In short, Jenkins continuously integrates the code(new code) to the Build files/artifact. Here Jenkins does Code Testing prior to build.

##### 95. What is meant by Deployment?

Deployment means copying build files (artifacts) to any specific server/environment. It may be a physical machine or cloud.

##### 96. What is Continuous Delivery?

Continuous Delivery is a practice followed in Devops. Build, Test, and Release of projects is automated without human intervention, which is not recommended for crucial/Big projects as there might be chances of having bugs with automated testings. It depends on the project whether to implement a continuous delivery or continuous deployment. In short, continuous delivery is releasing products/updates to the end-user/client automatically (automating deployment to production).

##### 97. What is Continuous Deployment?

Continuous Deployment is one of the best practices followed in Devops. Build, Test, and Deployment of projects is automated without human intervention. Which means qualified changes in code are deployed to the Test, UAT, QA, etc., environments but not to the production as soon as the code is pushed to VCS(Version Control System) without any human intervention. Then here the code is thoroughly tested. In short, continuous deployment is deploying the updates/product to the specific environment (automating deployment to Testing environment).

##### 98. How to Migrate Jenkins from one machine to another?

Install Jenkins in new machines and copy the .jenkins directory into the new machine, and then start Jenkins.

##### 99. What are the commands you use to start Jenkins?

Here are the commands to start, stop and to restart the jenkins:

* Start Jenkins: > jenkins.exe start or $ sudo service jenkins start
* Stop Jenkins: > jenkins.exe stop or $ sudo service jenkins stop
* Restart Jenkins: > jenkins.exe restart or $ sudo service jenkins restart

##### 100. What are the two components that you integrate with Jenkins?

The two important components that I integrated with Jenkins are Version control tool and Build tool, in general GIT and Maven.

##### 101. Mention some important plugins of Jenkins?

Here are some important plugins which are good to use in Jenkins:-  
1 Gits  
2 Maven  
3 Thin backup plugin  
4 Copy Artifact  
5 Shelve project  
6 Green Balls  
7 Amazon EC2  
8 Nexus  
9 Sonarqube and etc..

##### 102. What are the Jenkins Features?

Jenkins comes with the following features…  
1. It is a Free open source.  
2. Easy to install on various operating systems  
3. Supports Build Pipeline  
4. Supports Master/Slave architecture  
5. Supports third party tool integration  
6. It has both GUI and CLI  
7. It has built-in test harness  
8. Easily upgradable  
9. Records build events/logs  
10. Has notifications alerts system (through mails)  
11. Supports configuring Environmental Variables  
12. Has vast collection of plugins

##### 103. What are the advantages of Jenkins? Why do we use Jenkins?

Jenkins is an automation tool. It saves human time, just one click triggers the Code Build, Tests it and Deploys it to the specified environment.  
The Advantages of Jenkins are as follows:  
1. Jenkins is free of cost, easy to install and requires almost zero maintenance (easily upgradable).  
2. Easy to install on various operating systems as it is platform independent.  
3. Execution sequence of jobs can be managed using the build pipeline.  
4. Computing power is not limited, we can add any number of machines to it, as it supports Master/Slave architecture.  
5. Third party tool integration is very easy, GIT, Maven, Docker, Nexus, Sonarqube etc can be easily integrated.  
6. It has both a Graphical User interface and Command line interface.  
7. Jenkins test the code before code build as it has built-in test harness  
8. Records each and every build events/logs whether the build is success of fail, and notifies all users with an email.  
9. Jenkins has more than 1000 plugins used for integrations of hundreds of tools and services. And more than 1500 communities contribute to Jenkins plugins.

##### 104. What is Jenkinsfile?

The Jenkinsfile is a text file written in a groovy script. where all the stages of the pipeline are defined. And it is stored in the root directory of the project and pushed into a source code repository. This file triggers the build of jobs in the pipeline.

##### 105. What is SCM? Which SCM tools are supported in Jenkins?

SCM is a source code management tool that manages the versions of the project code and specifies the path of source code to jenkins (of course to the latest version). And Jenkins supports almost all SCM tools like,  
1 CVS  
2 Git  
3 Perforce  
4 AccuRev  
5 Subversion  
6 Clearcase  
7 RTC  
8 Mercurial.

##### 106. List some CI Tools that you know?

Here are some Popular CI Tools in the market,  
1. Jenkins  
2. Gitlab CI  
3. Travis CI  
4. Circle CI  
5. Codeship  
6. GO CD  
7. Teamcity and  
8. Bamboo

##### 107. How to make sure that Jenkins build does not break before complete?

Here are some major points to make sure that the project build does not break.

* Make sure you have a clean and successful installation of Jenkins.
* Make sure that all changes in config, POM and code are checked into the repository.
* Ensure specifying the correct branch/tag while configuring the job.
* Make sure you configured a correct version of your build tool like maven-3.6.

##### 108. How do you check Jenkins logs?

If you are using Jenkins(jenkins.war file in webapps) in Apache Tomcat then Log files should be at $TOMCAT HOME/logs/catalina.2020-02-02.log.  
  
In Linux:  
Log files should be at this location /var/log/jenkins/jenkins.log, unless customized in  
(for \*.deb) /etc/default/jenkins,  
(for \*/rpm) /etc/sysconfig/jenkins.  
  
In Windows:  
Log files should be at this location %JENKINS\_HOME%/jenkins.out and %JENKINS\_HOME%/jenkins.err, unless customized in %JENKINS\_HOME%/jenkins.xml  
  
In Mac OS X:  
Log files should be at this location /var/log/jenkins/jenkins.log, unless customized in org.jenkins-ci.plist.  
  
In Docker:  
If you run Jenkins inside docker as a detached container, you can use $docker logs < containerId > to view the Jenkins logs.

##### 109. Where does Jenkins store global and job related Configuration?

Jenkins global configurations are stored in "%USER\_HOME%/.jenkins/config.xml"  
And Jenkins Job configurations are stored in "%USER\_HOME%/.jenkins/jobs/%JOB\_NAME%/config.xml".

##### 110. How do you change the port number of Jenkins?

Go to %TOMCAT\_HOME%/conf/server.xml, Change port number 8080 to whatever you want.  
For example : <connector port="xxxx" Protocol ="HTTP/1.1"  
Or  
Go to %USER\_HOME%/.jenkins/config.xml and change the port number as per your need.

##### 111. I want to modify the JDK version from 1.8 to 1.10 in 1000 jobs? How do you do it?

Jenkins stores all configuration data in .jenkins/jobs/job\_name/config.xml file. We have to find the 1.7 version in all config.xml to replace it with 1.8 using the Linux Find and Sed command or with a small script of the same command. The command should be executed in jobs directory, like  
user@machine:~/.jenkins/jobs$**find . -name "config.xml" | xargs sed -i s/jdk1.8/jdk1.10/g**  
Then to load changes you have to click on “Reload config from disk” option in manage jenkins.

##### 112. How did you set up, build and deploy for your Jenkins Job?

It's very simple in Jenkins just we have to do the following,

* Configure GIT URL.
* Integrate Maven and specify the build command "mvn install"
* Go to the execute bash section and write a script to deploy. Ensure the following while writing the script,
  + Before copying the war file checks for disk space of the target server.
  + Copy war files to only those servers which are specified to do so.

##### 113. How to install Jenkins?

**Method:1**  
1. Download & Install JDK  
2. Download Tomcat  
3. Download jenkins.war and copy to $TOMCAT\_HOME/webapps [deploying jenkins to tomcat]  
4. To start Tomcat server use $ sh TOMCAT\_HOME/bin/startup.sh  
5. Launch Jenkins using this URL http://localhost:8080/jenkins  
6. command to shutdown tomcat: $TOMCAT\_HOME/bin/shutdown.sh  
**Method:2**  
1. Download & Install JDK  
2. Download Jenkins.war  
3. Jenkins.war comes with a light weight server called "jetty", the command runs jenkins jetty server is $ java –jar jenkins.war  
4. You can launch Jenkins by using below URL: http://localhost:8080  
**Method:3**  
1. $sudo apt update  
2. $sudo apt install jenkins  
3. $sudo service jenkins start  
4. Launch Jenkins using this URL: http://localhost:8080

##### 114. How will you define the post in Jenkins?

The **Post** section has one or more additional steps to execute after the completion of the pipeline. The execution of the **Post** section depends upon the specified condition. The post-conditions are as follows, Changed, Success, always, failure, regression, unstable and aborted.

##### 115. What is Label?

Label is a virtual name for one or more slave nodes using which we can tie a particular Jenkins job to always run on a particular machine.

##### 116. What kind of Problems you faced with your Jenkins system ?

Very rarely I came across some issues in the Jenkins are as follows:  
1. In case of running more jobs, UI becomes slow as it polls remote repo regularly.  
2. Connection to slave timeout.  
3. Git checkout timeouts.  
4. Confusion in selecting the right plugin as it has 3-5 plugins with similar names and less descriptive.

##### 117. Suddenly my Jenkins instance becomes slow, what steps I need to follow to improve the Performance?

1. Clean up the old jobs, for doing this Shelve plugin is recommended.  
2. Make sure your jenkins have a master/slave distributed concept.  
3. Make sure your master doesn’t run any job, just keep it for servicing jenkins traffic and schedule all your builds in slave nodes.  
4. Check your computing power (CPU utilization) and Memory usage.

##### 118. How do you upgrade Jenkins?

Before upgrading jenkins on the Production server you have to cross check the things by doing the same process on the Test environment. So do as follows,  
1. Take a test machine.  
2. Install the same old version of jenkins into the test machine.  
3. Copy .jenkins from old jenkins to test machine and turn on the Jenkins server in Test Machine.  
4. Deploy the new war file to the test machine.  
5. Test the newly deployed features and old features randomly in the test machine and make sure that everything is working fine as in old Jenkins.  
Now, Finally you can do the same thing on your production server.

##### 119. What is Continuous Testing?

The process of automating Code Testing as a part of SDLC is Continuous Testing. Jenkins will do automated testing as a part of Continuous Integration. And also Jenkins can be integrated with automation testing tools, so that it can automate the process of continuous testing.

##### 120. How do you set-up the crontab?

$ crontab –e  
Crontab is a simple text file that has to be written in a specific format. Like each field should be separated by spaces or tabs. And should have a single value or a set of single values. A single cron job should take only one line, but the line can be a long (more than 80 characters).

##### 121. How do I add delay in Jenkins?

There is a "quiet period" option in jenkins. You can set a value of 120(it’s in seconds) to delay the job by 2 minutes in starting.

##### 122. What is the difference between Web server & Application server?

Web server mainly deals with the http requests and serves static content like static html web pages, images , javascript, etc. Caching static content, load balancing and clustering can be done in the web server.  
Application server receives requests from web-server to serve dynamic content like Search results, business logic of the application. Application server is the one who interacts with databases and provides content to the web users.

##### 123. How do you deploy an application in Tomcat?

1. Firstly we should build a war-file/artifact.  
2. And Jenkins will trigger a shell script for deployment.  
3. It checks if the tomcat/target machine is running & has enough disk space.  
4. Jenkins will copy the war files into tomcat’s webapps location.

##### 124. How to clone a GIT Repository via Jenkins?

Create a Jenkins Job to clone you a Git repository, go to configure and set git repository’s URL, Save and Build the Job, that clones you a Git repository.

##### 125. How to set up a Jenkins Job?

1. Log in to Jenkins.
2. Click "New Item."
3. Enter a job name and select the job type (e.g., Freestyle, Pipeline).
4. Configure job settings like source code management and build steps.
5. Save the job configuration.
6. Trigger a build to test your setup.

##### 126. Explain how to move or copy Jenkins jobs from one server to another?

It is very simple to move or copy jenkins jobs from one machine to another, just go through the following steps:  
1. To copy a Jenkins job, copy an existing job and paste it in another machine’s Jenkins jobs location.  
2. You can move a job from one Jenkins server to another by simply moving the particular job directory and paste it in Jenkins Jobs location of another machine.

##### 127. What is a parameterized job in Jenkins?

For a single job/project you can have multiple configurations and while triggering a job/project jenkins will ask you to provide any configuration to follow. For example you have created a job and configured it to deploy in all the environments, and you made it parameterized so that Jenkins will ask you to select a specific environment (like QA, UAT, Staging, Pre-Prod, etc..) before starting a Parameterized build.

##### 128. How to Create a backup and copy files in Jenkins?

To create a backup of Jenkins Job, just copy %Jobs\_name% directory, you can also copy a Jobs directory to all your jobs. Here you may miss some configuration files. So the best way is to copy .jenkins directory.  
To create a Jenkins backup, all you need to do is to periodically copy your %JENKINS\_HOME% (.jenkins) directory. It contains all of your builds, jobs configurations, your build history and everything.

##### 129. Explain how you deploy a custom build of a core plugin?

The hpi package is the custom format Jenkins uses to distribute plugins. Here are the steps to deploy custom build of a core plugin:  
1. Stop jenkins.  
2. Copy your custom HPI package to $JENKINS\_HOME/plugins.  
3. Remove the previous expanded plugin directory.  
4. Make an empty file called .hpi. And start jenkins.

##### 130. What are the various ways to schedule builds in jenkins?

Yes, Jenkins has an options to schedule a builds:  
  1 By Source code management commits using Poll SCM.  
  2 After completion of other builds using post build condition.  
  3 Can be scheduled to run at a specified time using the build periodically option.  
  4 And there is one more option called Delay.

##### 131. Let us say you have a Jobs in the pipeline. The 1st job was successful, but the 2nd Failed, what to do next?

You should restart the pipeline from the second job where it failed, by doing "restart from Stage".

##### 132. What syntax does jenkins use to schedule build jobs or SCM polling?

The Cron Syntax, it is represented using five asterisks (\* \* \* \* \*) each separated by a space. The Syntax is as follows [Minute] [Hours] [Day Of the Month] [Month] [Day Of the Week].  
**For Example:**  
If you want to set up a cron for every Monday at 06 59 AM it would be 59 06 \*\* 1. Here 59 is a minute and 06 is hour and \* represents all days of the month and another star represents all months and 1 denotes monday.  
**For your clear understanding of Cron Syntax:**  
MINUTE - Minutes within the hour (0–59)  
HOUR - The hour of the day (0–23)  
DOM - The day of the month (1–31)  
MONTH - The month (1–12)  
DOW - The day of the week (0–7) where 0 and 7 both are Sunday.

##### 133. What is Devops ? In which stage the jenkins come into picture?

Devops is a culture followed by the company to collaborate, communicate, automate and integrate jobs of developers (dev team) and the IT operations. By making the whole product life cycle simpler. In order to improve speed and quality of release, Jenkins plays a Crucial role in DevOps, it automates code integration, automates builds, Tests and automates deployments.

##### 134. Name a Jenkins Environment you have used in shell script Environment or a batch file?

Here are some Jenkins Environment variables used in shell script,  
$JOB\_NAME  
$NODE\_NAME  
$WORKSPACE  
$BUILD\_URL  
$JOB\_URL  
$GIT\_URL.

##### 135. How to create a Multibranch pipeline in jenkins?

Multibranch pipeline creates a separate Jenkinsfile for each branch in a project. It means it will automatically create a pipeline for each branch and stores in SCM.

##### 136. What are the types of Projects or jobs available in Jenkins?

Different types of jobs/Projects are listed while creating a new job in Jenkins, and those are:

* Freestyle Project
* Maven Project
* Pipeline Project
* Multibranch pipeline
* External project
* Multiconfigurational project
* GitHub project.

##### 137. What are the System Requirements for installation of Jenkins?

The minimum hardware requirements:

* 256 MB of RAM
* 1 GB of drive space

And recommended hardware configuration for a small team:

* 1 GB+ of RAM
* 50 GB+ of drive space.

##### 138. What is the Jenkins Pipeline? What is the CI/CD pipeline?

Pipeline is a group of jobs that are chained and integrated in sequence to execute. The automation of Pipeline jobs (to build, test, release) is CI/CD pipeline. The automated pipeline saves a lot of time in the builds.

##### 139. What are the differences between Jenkins, Bamboo and TeamCity?

| Jenkins | Bamboo | TeamCity |
| --- | --- | --- |
| Jenkins is Open source. | Bamboo is very expensive. | TeamCity is a little expensive. |
| Very easy to set up and use. | Little complex to set up and use. | Setting up and using is easy but not as jenkins. |
| Has less built in features but there are lots of plugins to integrate. | Has more built in features than Jenkins but very less plugins to integrate. | Has advanced built in features than jenkins and bamboo, but has less plugins. |
| Supports cloud hosting. | Runs on local machine/ Bitbucket as cloud. | Runs on local machine. |

##### 140. What is the trigger? How the repository is being polled when a new commit is detected?

Triggers are defined as when and how pipelines should be executed. Jenkins polls SCM periodically at configured time, if it detects the new commit then triggers a build.  
Ex: You can create jobs that polls the SCM and triggers when a change is pushed.

##### 141. What is an Agent?

Agent is a physical or cloud environment that is controlled by Jenkins master and assigns jobs to execute. This is a master-slave architecture, where a slave is an agent.

##### 142. How do I schedule a Period in Jenkins?

Here are the steps to schedule Build Periodically jobs in Jenkins  
1. Go into the Job and click on the "Configure" option in the left panel.  
2. Scroll down to "Build Triggers" and click on the checkbox of "Build Periodically" and set a value in cron syntax.

##### 143. What is Groovy in Jenkins?

Groovy is the default scripting language that is used in writing pipeline scripts.

##### 144. How many builds you store in your jenkins?

Jenkins GUI keeps the last 30 builds of each job. But actually Jenkins stores all builds and logs in disk. You can access those using the build number in url. It is suggested to remove old builds to save disk space. You can set "how many days the build should be kept in jenkins" and "how many builds to keep". But some companies suggest keeping 60 builds of each job in jenkins.

##### 145. How do you rotate logs with your jenkins?

Jenkins GUI keeps the last 30 builds of each job. But actually Jenkins stores all builds and logs in disk. Jenkins has a feature to remove old builds, in order to save disk space. We can set a number of builds to keep in the General setting of Job. So log rotation depends on your configuration.

##### 146. What is Maven?

Maven is a Build Automation tool mainly used for java projects and also compatible for other projects such as project written in C# and Ruby etc. Maven downloads project dependent libraries from the central repository Maven2 and keeps in ".M2" Directory in the "$userhome". By this you came to know Maven manages project builds and its dependencies as well. Maven keeps all key information about the project in the POM.xml file.

##### 147. Do you know the maven life cycle?

Yes, Maven works as a build tool for mainly java code. And these are the phases involved in the maven life cycle.  
**Validate** - It is the initial step where it validates the project and all necessary information is available or not.  
**Compile** - In this step, the Project source code is Compiled.  
**Test** - During this step compiled source code is tested using a suitable builtin unit test framework.  
**Package** - In this step the Compiled code packaged in its distributable format such as JAR/WAR/EAR.  
**Integration Test** - Deploys the package into an environment, where integration tests can be run.  
**Verify** - Verifies the result package of the integration test is valid, it has to meet quality criteria.  
**Install** - It will install the package into local repository, for use as a dependency in other projects locally.  
**Deploy** - In this Step it copies the final package to the remote repository for sharing with other developers and Projects.

##### 148. What is POM?

POM (Project Object Model), is the fundamental unit in maven, it is written in XML which has all the key information about the project’s Configuration, which is needed to build the project. And the pom.xml file will be in the project's source directory.

##### 149. What is the use of Maven Plugins?

Maven Plugins are used for:

* Creation of jar files.
* Creation of War files.
* Code Compilation.
* Unit testing of code.
* Documentation.

##### 150. Tell me some differences between Ant & Maven?

| ANT | Maven |
| --- | --- |
| Ant is a toolBox | Maven is a Framework. |
| Ant is flexible with configurations | Maven has conventions. |
| Ant is having Procedural Approach, it takes a long xml build script. | Maven is Declarative, enough declaring the requirements in pom.xml. |
| Ant does not have any Life Cycle. | Maven has a Life Cycle. |
| Ant Scripts are not reusable | Maven scripts are reusable with little modifications. |

##### 151. What are the minimum requirements for POM?

The minimum. requirements for the POM file is Project root , Model Version, group ID, artifact ID and Version.  
For example:  
   <project>  
          <modelVersion>1.0.0</modelVersion>  
          <groupId>com.wiculty.app</groupId>  
          <artifactId>wiculty-Learning-app</artifactId>  
          <version>1</version>  
   </project>

##### 152. What are the features of Maven?

Here are some features listed:  
**Simple to Use**: As The Maven is Conventional, it provides a project setting that is based on genuine Projects.  
**Fast**: Create any project/module within a few Seconds.  
**Easy to Use**: Maven commands are pretty straight forward and they are easy to understand.  
**Multiple Projects**: Can manage more than one project at the same time.  
**Huge Library**: As the Maven is a growing repository of libraries and metadata to use out of box. Can easily write plugins in Java or scripting languages.  
**Dependency Management**: Automatic updating dependencies and transitive dependencies.

##### 153. How to install Maven?

Maven is a open source build tool, it can download & install in any platform like ubuntu, windows, The steps are described below

* Download & Extract Maven
* Add JAVA\_HOME and MAVEN\_HOME to the list of Environment variables.
* Add the Environment Path in Maven Variable. (If OS is windows)
* Verify the maven installation by checking its version.

##### 154. What does the mvn clean do?

The "mvn clean" command removes/cleans the target directory with all the previous build data before starting the new build.

##### 155. Define Repository? What are the types of repositories?

Maven Repository is nothing but a directory where all the Jars and POM.XML files are stored. There are 3 types of repositories:  
**Local Repository**: It is a local repository that will be created by maven, when the maven commands are executed. When you run maven command maven looks for dependencies first in the local repository. Which is your .M2 directory located in your $userhome  
**Central Repository**: Maven community manages one repository on the web which is named as Central Repository. It contains a large number of libraries. When you run maven command maven looks for dependencies first in the local repository, if it is not there then by default maven looks in Central Repository.  
Maven’s Central Repository url is "repo.maven.apache.org/maven2/"  
**Remote Repository**: If needed dependency is not available in Manven’s Central Repository then it may be available in any other Remote location managed by some other organizations those repositories are called Remote repository. So manually defining libraries in pom is suggested because some of the Libraries may not be available in the Maven’s central repository.

##### 156. Name the three Build cycles of Maven?

Here are the Three build cycles of maven listed below:  
**Clean**:Cleans the artifacts created by prior builds  
**Default**:Deals with the complete deployment of the application  
**Site**:Generates site document(java document) for the project.

##### 157. When does the maven use the External dependency concept?

Maven dependency management search for dependencies in local, central and remote in sequence, if there is no dependency present in any of these, then in this case it uses External dependency concept.

##### 158. What is SNAPSHOT in Maven?

The **SNAPSHOT** is nothing but a current development copy of the project. In short, the working version of the project is a snapshot. For example, prior to version "1.0-release" there will be "1.0-Snapshot" version.

##### 159. What is a transitive dependency in Maven?

Transitive dependency is like indirect dependency, suppose A depends on B and B depends on C. Then, A depends on both B & C but A is transitively dependent on C. And the good thing is Maven downloads transitive dependencies automatically.

##### 160. What is all the information POM contains?

The POM contains the following Configuration Details like:

* Project Dependencies
* Plugins
* Goals
* Build Profiles
* Project Version
* Developers info
* Mailing List

##### 161. Explain how you can exclude dependency?

By Using an Exclusion Element in pom.xml dependencies can be excluded. For example, you have to add this in your project's pom.xml file.  
<exclusions>  
    <exclusion>  
          <groupId>sample.Project</groupId>  
          <artifactId>Project</artifactId>  
          <version>1.2.3</version>  
     </exclusion>  
</exclusions>

##### 162. Explain how to run the test classes in Maven?

The **surefire plugin** is used to run the test classes in Maven & check and configure your settings in setting.xml and pom.xml for a property named 'test'. Here is an example of adding surefire plugin to Pom.xml.  
  
<plugins>  
     <plugin>  
            <groupId>org.apache.maven.plugins</groupId>  
            <artifactId>maven-surefire-plugin</artifactId>  
            <version>2.22.2</version>  
     </plugin>  
</plugins>  
  
And then you have to execute this command  
$ mvn test   //for single test  
$ mvn -Dtest=UsersServiceImplTest test   //for all tests.

##### 163. List out the build, source and test directory in Maven?

Build directory is your "Target" directory, Source is your "src/main/java" and test is "src/main/test" directory all these will be under "src" directory.

##### 164. What build tool does?

Build tool does the following:

* Generates artifacts/binaries for the source code will be generated.
* From source code it generates documentation.
* Source code will be compiled.
* Packs compiled code into a JAR, WAR or ZIP File.
* Copies packed code into the local repository along with their build versions in "com" directory.

##### 165. What is meant by MOJO in Maven?

**MOJO** (**M**aven Plain **O**ld **J**ava **O**bject) is the smallest executable unit that maven recognizes. MOJO has its own goal, a plugin has one or more goals to execute as it is a collection of one or more MOJOs. It is the smallest unit of maven plugin. Or the smallest plugin.

##### 166. Where MAV coordinates/GAV Parameters come into picture?

Decide Artifact type and its storage path in local/central/remote repositories. And decides Jar/War names Identify a particular version of JAR file to be downloaded during dependency resolution. And all this GAV parameters are used in POM.xml file.

##### 167. List out the Maven's order of inheritance?

1. Parent POM or Super POM
2. Project POM
3. Settings
4. CLI Parameters.

##### 168. How do you produce error messages in maven?

There are various commands to produce debug messages, For instance, more popular is -x flag or --debug. There are various commands to produce error messages ,For instance ,more popular is -e flag or --error.

##### 169. What do you Mean by Super POM?

Default POM of Maven is the Parent POM or Super POM. All POMs are derived from super pom.

##### 170. What is a Maven Artifact?

An Artifact is a file, usually it is a JAR/WAR file that gets deployed into a Maven Repository. A Maven build creates one or more artifacts. This file includes all the project.

##### 171. What is the Goal in Maven Terminology?

A Goal in Maven is a smallest task to be executed. Maven builds and manages a project, may be built to Zero goal or more build phases(has more goals).

##### 172. Where do you find Class Files when you Compile a Maven Project?

You will find the class files in Project’s base directory/target/classes/.

##### 173. What is the Project's fully qualified Artifact name?

Fully qualified artifacts name has these properties group ID, artifact ID and the version string. These three together identify the artifact. "<groupId> <artifactId> <version>".

##### 174. What is Build Profile?

A **Build Profile** is nothing but the set of Configuration values which are used to overwrite default values of Maven build. Using this profile you can customise build for development and production.

##### 175. What are the different types of build profiles?

Here are the different types of build profiles listed:  
**Per Project**: It is defined in Project POM file(pom.xml)  
**Per User**: It is defined in Maven settings.xml file($USERHOME/.m2/settings.xml)  
**Global**: It is defined in global setting.xml file($USERHOME/.m2/conf/settings.xml)  
**Profile descriptor**: It is defined in the project’s base dir (profiles.xml)

##### 176. What is Maven Archetype? What are all the different types of Project types?

Archetype plugin of maven a templating tool kit to create a variety of Java Project structures, like webapp, site-simple, mojo, plugin, quickstart, etc. Which generates jar, war, zip files and dll files.

##### 177. What are the things that you must define for External dependency?

External dependencies are also configured in the POM.XML file as in the same way normal dependencies are configured. Even the same GAV parameters to be defined so that maven can include that dependency.

##### 178. What are the benefits of storing JARS/external dependencies in the local repository instead of remote repository?

As the local repository is quickly accessible for the project and there will be no need of versioning of JAR files.

##### 179. What is the use of Optional Dependency?

You can mark any dependency as optional using "Optional" element For Example  
<dependencies>  
      <dependency>  
            <groupId>sample.Project</groupId>  
            <artifactId>Project</artifactId>  
            <version>1.2.3</version>  
            <optional>true</optional>  
      </dependency>  
</dependencies>

##### 180. How to run the clean plugin automatically during the build?

For this you need to include the clean plugin inside the execution tag in the POM.xml.

##### 181. What is a maven Assembly?

The Assembly is to concatenate the project’s output along with its dependencies, modules, site documentation and files/directory into a single distributable archive. Like jar, war, zip, tar, tar.gz, tar.xz etc.

##### 182. What are the types of Maven Plugins?

There are mainly two types of Plugins listed below:  
Build Plugin:- This one is used during build and should be configured in element of pom.xml  
Reporting plugin:- This get executed during site generation and they should be configured in the element of pom.xml

##### 183. What are the Configurations Possible in setting.xml file?

The possible configurations in setting.xml are:

* User can configure proxy
* User can configure local repository
* User can configure central repository
* User can configure remote repositories

##### 184. If you fail to define information in pom.xml, then from where does pom inherit that information?

From Supre POM, all poms are inherited from this super pom some call it as parent pom or base pom.

##### 185. How can you activate the profiles?

Here are some ways to activate/trigger Maven build profiles:

* Explicitly using commands
* Through the maven settings
* Through the Environment variables
* From OS Settings
* Missing Files/Present Files.

##### 186. How Maven handles and determines which dependency version to be used when found multiple versions?

Maven has a concept of Dependency Mediation. Maven chooses the first declared dependency, if it got two separate versions of dependency declarations in the same level of dependency tree.

##### 187. Explain what would the "jar: jar" goal do?

Jar: Jar will just create JAR from target/classes directory, considering that compilation and test already done. You can call it a jar plugin, more specifically a jar mojo.

##### 188. What is a force maven update?

$mvn clean install -u.  
-u flag, forcefully updates snapshot dependencies. It is not good to update release version dependencies in this way.

##### 189. How to override default name of WAR file?

By using the final name element <finalName></finalName> in the pom.xml file, overwrites the default name of the artifact.

##### 190. Is there a sequence in which Maven searches for dependencies?

Yes, Maven has the hierarchy to search dependencies, firstly search for dependency in local repository if not found it will look in central repository then in the remote repository only if remote repository is mentioned else maven quits searching.

##### 191. What aspects are managed by maven?

Maven manages the following aspects:

* Builds
* Release
* Distribution
* Documentation
* Reporting
* SCMs

##### 192. What are the tenets of Maven?

Tenets of Maven are:

* Project Oriented
* Dependency Management
* Reusable Central Repositories
* Conventions over Configuration

##### 193. How do you stop the inheritance of plugins to other Poms?

Set inherited elements to false in pom.xml in order to stop inheritance. For example:  
<plugins>  
      <plugin>  
            <groupId>org.apache.maven.plugins</groupId>  
            <artifactId>maven-antrun-plugin</artifactId>  
            <version>1.2</version>  
            <inherited>false</inherited>  
      </plugin>  
</plugins>

##### 194. How can you build your project offline?

$mvn dependency:go-offline.  
this will resolve all the project dependencies, transitive dependencies, reports and everything else required to build a project. And then by executing the following command you will see the project build without any network.  
$mvn -o package //can use clean command as well($mvn -o clean package).

##### 195. What is the command to build the project quickly on the maven site?

$mvn site //this command will build the project code quickly.

##### 196. What does the command do mvn clean dependency:copy-dependencies package?

$mvn clean dependency:copy-dependencies package //This command will clean the project, copy the dependencies and package the project (executing all phases up to the package).

##### 197. What are the phases of a clean life cycle?

The Clean life cycle consists of following phases:

* Pre-Clean
* clean
* Post-Clean.

##### 198. Mention how profiles are specified in Maven?

Profiles are specified in the POM file itself. For example,  
<settings>  
    <activeProfiles>  
          <activeProfile>profile-1</activeProfile>  
    </activeProfiles>  
    <activeProfiles>  
          <activeProfile>profile-2</activeProfile>  
    </activeProfiles>  
</settings>

##### 199. What is the need for DevOps?

Companies are trying to release updates as soon as they are developed. Releasing small features like this has many advantages over full one time Big-Release like quick feedback from clients, better quality etc. To achieve this, companies need to:

* Increases the deployment frequency
* Very fast Deployments of new features
* Less complexity in SDLC process
* Lower failure rate of new releases
* Shortened lead time between fixes
* Improved communication and collaboration
* Will get more time to develop new features
* Rolling back a new update will not be as complex as in Big-Release.

DevOps fulfills all these and helps in achieving a seamless software delivery cycle.

##### 200. If you vaguely remember a command, without googling how will you recollect it?

This command lists all Docker commands. If you need help with one specific command, you can use the following syntax:  
$ **docker < command > --help** (or) simply $ **docker --help**

##### 201. What is Hyper-V or Hypervisor?

Hypervisor is a software that makes virtualization possible. It divides the host system resources and allocates the resources to each divided virtual environment. The hypervisor allows you to create a virtual environment on your system so that guest virtual machines can be created and operated. It has control over resource allocation to multiple virtual machines.  
In short, hypervisor allows you to create multiple virtual servers on a single physical server. Some hypervisor software are Hyper-V, Xen, VMware Player, Virtual-PC, Oracle-Box, KVM, etc.

##### 202. What is Docker?

Docker is an open-source lightweight OS-level containerization technology. It bundles the whole application and its dependencies together and forms a container so that your application works hassle free in any environment (Development, Test or Production). Containers are lightweight and portable and they can communicate with each other. Containerization technology like Docker will share the same base machine’s operating system kernel, and due to this it is extremely fast. With the help of containers any app can Build, Ship and Run.

##### 203. What is virtualization?

Virtualization is the logical division of your physical machine into multiple machines(compute storage, servers, application, etc.). However, this allows multiple OS to run simultaneously on a single system. A software called Hypervisor makes this kind of splitting possible. The virtual environment created by the hypervisor is called Virtual Machine.

##### 204. What is containerization?

Usually, code developed on one machine might not work perfectly fine on any other machine because of the dependencies. So an application is bundled with all its configuration files and dependencies so that it could work on any machine. To make you understand this bundle is called a container. Most popular containerization tool after rocker is Docker.

##### 205. What is a Docker Container?

Container is a process, or Docker containers are runtime instances of the Docker image. one container is an instance of one image. Docker containers have the application and its dependencies in it. It shares the kernel of the host operating system. Docker containers are not limited to any specific IT infrastructure, they can run on any computer, infrastructure, and in any cloud. Docker containers are created using Docker images and then run it, or we can use the images that are already created in the Docker Hub.

##### 206. What are Registries and types?

Docker's registry is nothing but a Docker Hub, which allows you to store images publicly/privately. Millions of images/registries can be stored to docker hub.  
There are two types of registry is  
\* Public Registry  
\* Private Registry  
Public registry can be accessed by any one, and private has restrictions to be defined.

##### 207. What is a Docker Hub?

There has to be a centralized registry to store docker images. Docker Hub is one among those cloud registry which store repositories(images). We can also store or push images to Docker Hub, So that the images can be deployed to the host. Users can pull images from Docker Hub and use them to create customized images and containers.

##### 208. What are Docker Images?

To make you understand, just assume Docker image is a kind of mini OS of Docker container. But keep in mind that containers do not have their own OS or a separate kernel, containers share the OS of the base machine. Docker images are used to create containers. Every docker image can be stored as the Docker registry in Docker Hub. You can create your own images by using an existing container or by writing Dockerfile.

##### 209. How to create a docker container from an image?

Get required images from docker repository and run it to create a container. Use the following command:  
$ docker run -it < image\_name > /bin/bash  
-t (tty) flag for a terminal which is bin bash here.  
-i flag for interactive mode, means we will get terminal in interactive mode.  
This is a basic command to create a container. This command is enough to create a container.  
But we can create a container by adding extra flags as per need, as follows  
$ docker run -itd < image\_name > /bin/bash  
-d flag to create a container in the detached mode.  
$ docker run -itd < image\_name > --name < container\_name > /bin/bash  
--name option to give a name to your container.  
$ docker run -itd < image\_name > --name < container\_name > -h < host-name > /bin/bash  
-h flag to provide a host name to your container.  
$ docker run -itd < image\_name > --name < container\_name > -h < host-name > -v < source-path >:< destination-path > /bin/bash  
-v flag to allocate volume to your container.  
$ docker run -itd < image\_name > --name < container\_name > -h < host-name > -v < source-path >:< destination-path > -p8080:8080 /bin/bash  
-p flag to expose the base container’s 8080 port on the 8080 port of the base machine.

##### 210. What are docker volumes?

Docker volumes are like virtual disks to store data and can be shared between containers, in simple docker volumes are just like normal directories which contain sub directories and files stored in host machines, but these directories are mounted to containers so that the data can be accessed in containers. We can create volumes by the following command:  
$ docker volume create < volume-name >  
Then mount this volume to the container using -v flag, or you can mount the existing directory to containers as volume.

##### 211. Where the docker volumes are stored?

In a linux machine Docker volumes are stored at this location:  
/var/lib/docker/volumes

##### 212. What are the states of Docker containers?

Important states of Docker container are as listed below:  
1. Running  
2. Paused  
3. Restarting  
4. Exit.

##### 213. What is a Docker Swarm?

Docker Swarm is an orchestration tool for Docker containers. Docker Swarm is used to scale up/down nodes in a cluster. Even clusters can be created by Docker Swarm. It uses a network for internal communication among containers. Docker Swarm is native clustering for Docker. It turns a container pool of Docker hosts into a single Docker host. Docker Swarm manages the Docker API as well.

##### 214. Do Docker support for IPV6?

Yes, but only if you modify /etc/docker/daemon.json and set the ipv6 key to "true". IPv6 networking is supported only on Docker daemons run on Linux hosts.

##### 215. How does communication happen between Docker clients and the Docker Daemon?

Communication between Docker client and Docker Daemon happens with the combination of Rest API, socket.IO, and TCP.

##### 216. What is a Dockerfile?

Docker can build images by reading the instructions given in a file called Dockerfile. A Dockerfile is a text document that contains all the commands in a sequence. This "docker build -t < Dockerfile/path > .", command will execute several instructions written in Dockerfile in a sequence to create Docker images. The common commands used in Dockerfile are: FROM, MAINTENANCE, COPY, RUN, and CMD.

##### 217. How do you run multiple containers using a single service?

By using an orchestration tool like docker-compose, you can run multiple containers using a single service. All docker-compose files use yaml language.

##### 218. What is Docker Compose?

Docker Compose contains information about the services, networks, and volumes for setting up the Docker application. So, you can use Docker Compose to create separate containers, and host them and get them to communicate with each other. Make sure that each container should expose a port for communicating with others. Docker Compose is written in YAML format.

##### 219. Is it possible to use JSON instead of YAML in Docker Compose?

Yes, JSON can be used instead of YAML for a Docker Compose file. we have to specify the filename with the command:  
"**docker-compose -f docker-compose.json up**".

##### 220. What is a Docker Namespace?

The namespace is one of the features in linux and an important concept of containers. Namespace in Docker is a technique which offers isolated workspaces in Containers. In order to remain portable Docker provides various namespaces and does not affect the host system. Few Namespace types supported by Docker are as: – PID, IPC, NET, MNT, UTS, User accounts, Mount, Process Tree, Network etc.

##### 221. What is the lifecycle of a Docker Container?

A complete lifecycle of Docker containers:  
\*. Create a container  
\*. Run the container  
\*. Start the container  
\*. Stop the container  
\*. Pause the container  
\*. Unpause the container  
\*. Restart the container  
\*. Kill the container  
\*. Destroy the container

##### 222. What is Docker Machine?

You can use Docker Machine to install Docker Engine, and Docker Swarm on one or more virtual systems. These hosts can be managed by using the docker-machine commands.

##### 223. How to check versions of Docker Client and Docker Server?

This is the command used to know about Docker Client and Server versions:  
**$ sudo docker --version**

##### 224. How to get the number of containers running, paused and stopped?

You can use the following command to get detailed information about the docker installed on your system.  
$ docker info (or)  
$ docker ps -a (or)  
$ docker container ls -a  
You can get the number of containers running, paused, stopped, the number of images and a lot more.

##### 225. How to build a Dockerfile?

Once you’ve written a Dockerfile, you need to build it to create an image with those specifications. Use the following command to build a Dockerfile:  
**$** "**docker build -t < username/imageName > .**"  
Here at the end of the above command one space and period(.) includes a command to locate the path of Dockerfile. And instead of period(.) use the entire path if it is not stored in the same directory.

##### 226. How to login into docker repository or registry?

You can use the following command to login into hub.docker.com:  
$ docker login  
You’ll be prompted for your username and password, insert those, you’re logged in.

##### 227. How do you modify a base image or personalize it?

Pull an image from docker hub to your local system with one simple command:  
$ docker pull < image\_name >  
Then write a docker file as per your requirement, Then build the image using that Dockerfile  
$ " docker build -t < username/image-name:tag > . ". Now you have your own customised image.

##### 228. How do you list all the running containers?

This is the command to lists down all the running containers:  
$ docker ps (or)  
$ docker container ls

##### 229. How do you access a container among 50 running containers? And how do you execute a command in it?

This is the command to get access to a running container and to run a commands in it:  
$ docker exec -it < container-id > bash-name  
For example $ docker exec -it my\_container /bin/bash  
The exec command lets you execute commands inside a container.  
Or  
$ docker attach < container-id >  
But, the first method is preferable and recommended.

##### 230. How to Start, Stop and Kill a container?

To start a docker container this is the command :  
$ docker start < container\_id >and for stopping a running container:  
$ docker stop < container\_id >  
And to kill a container:  
$ docker kill < container\_id >

##### 231. How do you edit and update a Docker Container? Also, how do you make it new and store it on the local system?

Yes, you can edit and update a container. Go to your container, update it with whatever you want and then, execute the following command in the docker environment but not in a container.  
$ docker commit < container-id > < username/image-name >

##### 232. How do you push an image to docker hub?

$ docker push < username/image-name > (this will take tag "latest" as default) to tag your own versions you should mention tags as  
$ docker push

##### 233. How to delete a stopped container?

Use the following command to delete a stopped container:  
$ docker rm < container-id >

##### 234. How to delete a running container?

Use the following command to delete a running container:  
$ docker rm -f < container-id >  
-f flag stands for force.

##### 235. Will you lose your data, when a docker container exits?

No, you don’t lose any data when you exit Docker container. Data that you save in the container gets preserved on the disk until you explicitly delete the data or container.

##### 236. How to delete an image from the local storage system?

To delete an image form the local Docker machine is as follows:  
$ docker rmi image\_id

##### 237. Why is docker prune used for?

$ docker system prune  
This command is used to remove all the stopped containers, all the networks that are not used, all dangling images and all build caches. It’s one of the most useful docker commands.

##### 238. What is a client in Docker?

A command line interface (CLI) is known as a client of docker (the docker command). Docker has CLI to interact with Docker daemon.

##### 239. How is Docker different from other containerization methods?

Any type of applications can be made running on the same hardware using docker containers, it is easy for engineers to create containerized applications and it makes managing and deploying much more easier. Even it is very easy to deploy docker containers in any cloud based environment. You can even share containers with your applications.

##### 240. Can you use JSON instead of YAML for ‘mycomposefile.yml’ in Docker?

You can use JSON instead of YAML for your compose file, to use JSON file with compose, specify the JSON filename to use, for eg:  
**$ docker-compose -f docker-compose.json up.**

##### 241. How far do Docker containers scale? Are there any requirements for the same?

Almost all MNCs run container technology. Containers can be scaled to hundreds of thousands folds and can run parallely. For scaling docker needs the memory at all times and a way to use this memory efficiently when scaled up.

##### 242. What platforms does docker run on?

Docker can run on almost all environments a linux based physical machine or cloud environment and in windows using Docker Toolbox.  
various Linux versions are:  
Ubuntu 16.04, 18.04+  
Fedora 19/20+  
RHEL 7+  
CentOS 7+  
openSUSE 12.3+  
various Cloud platforms are:  
Amazon EC2  
Amazon ECS  
Google Compute Engine  
Microsoft Azure and etc..

##### 243. How do you know the status of a Docker container?

Use the following command that displays the list running containers  
$ docker ps  
To display all containers, use the following command:  
$ docker ps -a  
It shows the status of the docker container.

##### 244. Can you remove a paused container from Docker?

No, You cannot remove a paused container. The container should be stopped before removing it. But to remove a paused container you need to apply force, by using this command  
$ docker rm -f

##### 245. Can a container restart by itself?

By default No, it’s not possible for a container to restart by itself as the flag "--restart" is set to false by default.

##### 246. Is it better to directly remove the container using the rm command or stop the container before removing?

It is always better to stop the container and then remove it using the remove command.  
$ docker stop then  
$ docker rm  
Stopping the container and then removing it will allow sending SIG\_HUP signal to recipients. This will ensure that all the containers have enough time to clean up their tasks. This method is considered a good practice, avoiding unwanted errors.

##### 247. Will cloud overtake the use of Containerization?

My or may not be, it is a business, whoever provides a good service they may gain popularity. Of Course no service is permanent one comes after another. Docker containers are gaining popularity but at the same time, Cloud services are a good competitor.

##### 248. What are the main drawbacks of Docker?

Some notable drawbacks of Docker as follows:  
\* Doesn't provide a proper storage option  
\* Offer a poor monitoring option (docker stat or docker event)  
\* Complicated automatic horizontal scaling set up  
\* No automatic rescheduling of inactive Nodes

##### 249. How many containers can run per host?

You can run as many containers as you wish per host until your machine memory is full. Docker does not put any restrictions on it. But each container needs storage space, CPU and memory which the hardware needs to support. Containers are considered to be lightweight but very dependent on the host OS.

##### 250. What about load balancing in Docker containers and hosts? How does it work?

Suppose if you have your application running on multiple containers, it is good to maintain a Load Balancer because it diverts the user request equally to all containers. So that no container goes down due to heavy traffic. If any container goes down the traffic of that container will be shared equally to remaining containers. The same is in case of hosts.

##### 251. Do you run Docker compose in production?

Of course yes, used docker compose in production. It’s good to define applications with compose files in production, so that the same can be used in various stages like staging, testing, pre-prod etc.

##### 252. What is Jenkins X?

Jenkins X is an open-source project that extends the capabilities of Jenkins, the popular automation server, specifically tailored for Kubernetes-based development and continuous integration/continuous delivery (CI/CD) pipelines. It aims to streamline and automate the process of building, deploying, and managing cloud-native applications on Kubernetes.

##### 253. What is Terraform?

Terraform is a tool to create and control infrastructure(computer networks, storage, software applications, and other digital services) using simple instructions. It can manage these systems across various cloud providers like AWS, Azure, and GCP, all using just one tool.  
These simple instructions are known as code. If we have code, then we can create infrastructure. So, we have infrastructure in the form of code, meaning Infrastructure as Code (IaC).  
  
Terraform is open-source, written in Golang, and created by HashiCorp, offering features like idempotency and scalability for managing infrastructure deployments efficiently.

##### 254. What is IaC?

IaC stands for "Infrastructure as Code". It's a way to use code to set up and manage computer systems, like servers and networks, instead of doing it manually. By writing code, they can automate these tasks, which makes managing infrastructure easier, faster, and more reliable. It also helps ensure consistency and allows for easier tracking of changes.

##### 255. What is IaS?

IaC stands for Infrastructure as a Service, is a cloud computing model where users can rent virtualized computing resources like servers and storage over the internet. Instead of owning and maintaining physical hardware, organizations can access and manage these resources as needed. IaaS providers handle hardware maintenance, offering scalability, and flexibility. This enables businesses to focus on their applications without the hassle of managing physical infrastructure.

##### 256. What purpose does Terraform serve in the realm of infrastructure as code (IaC)?

Terraform serves as a fundamental tool for automating the provisioning, deployment, and management of infrastructure resources. Its primary purpose is to streamline and standardize the process of infrastructure configuration through code. Here's how Terraform serves in the realm of IaC:

* Automation of Infrastructure Deployment
* Scalability and Consistency
* Version Control and Auditing
* Multi-Cloud and Hybrid Cloud Support
* Dependency Management and Execution Plans
* Modularity and Reusability
* Infrastructure as Code Best Practices

##### 257. Explain the difference between declarative and imperative programming?

|  |  |
| --- | --- |
| **Declarative** | **Imperative** |
| Focuses on what the program should achieve without detailing the process. | Explicitly states how the program should achieve the desired result. |
| Expresses the logic of a computation without revealing its control flow or state management. | Involves writing step-by-step instructions for the computer to follow. |
| Defines the desired result and lets the system determine the steps to achieve it. | Closer to the machine's way of thinking, requiring a clear understanding of system internals. |
| Examples include SQL for database management and HTML/CSS for front-end UI design. | Commonly used in languages like C, Java, and Python. |

##### 258. Does Terraform fits into declarative or imperative paradigm?

Terraform primarily fits into the declarative paradigm of programming. While it has some imperative-like features such as for loops, dynamic blocks, and complex data structures, Terraform's core approach is declarative. In declarative programming, the focus is on defining the desired outcome without specifying the exact steps to achieve it, allowing the system to determine how to reach that state efficiently.

##### 259. What are providers?

Terraform, providers are plugins that interact with APIs and manage resources within specific cloud or service providers. These providers enable Terraform to provision and manage infrastructure resources across various platforms like AWS, Azure, Google Cloud, and more.

##### 260. Why AWS-CLI is used?

AWS CLI (Command Line Interface) is used to interact with Amazon Web Services from the command line. It allows users to perform various tasks such as managing EC2 instances, S3 buckets, IAM roles, and more, directly from the terminal. AWS CLI provides a convenient and efficient way to automate tasks, script operations, and integrate AWS services into workflows without needing to use the AWS Management Console.

##### 261. How terraform contribute to managing resources in different cloud platforms or infrastructure providers?

Terraform simplifies managing resources across different cloud platforms or infrastructure providers by providing a unified way to define infrastructure configurations. With Terraform, users can use a single set of configuration files to provision and manage resources on AWS, Azure, Google Cloud, and other providers. This allows for consistency, scalability, and ease of management, as changes can be applied uniformly across diverse environments without the need for platform-specific tools or scripts.

##### 262. What is the best way to store cloud providers (ex: AWS, GCP or Azure) Access Key and Secrect Key?

The best way to store cloud provider access keys and secret keys is to use a secure and centralized secrets management service provided by the respective cloud platform. For example, AWS provides AWS Secrets Manager, Google Cloud offers Secret Manager, and Azure offers Azure Key Vault. These services encrypt and store sensitive credentials, allowing for secure access and management while ensuring compliance with security best practices.

##### 263. Describe the basic structure of a Terraform configuration file?

Terraform configuration file define the infrastructure resources that Terraform will manage.  
Example:  
resource "aws\_instance" "example" {  
    ami = "ami-0c55b159cbfafe1f0"  
    instance\_type = "t2.micro"  
}

##### 264. What language is used to write terraform configurations file?

Terraform configuration files are primarily written in HashiCorp Configuration Language (HCL).

##### 265. What is 'state' in terraform?

In Terraform, "state" refers to a snapshot of your infrastructure's current status and configuration. It's stored in a file named 'terraform.tfstate' by default and helps Terraform to keep track of what resources are created and how they're configured.

##### 266. How does Terraform maintain state?

Terraform maintains state by keeping track of the current state of your infrastructure in a state file. This file is usually stored locally or remotely (e.g., in cloud storage) and records the mapping between the resources in your configuration files and the real-world infrastructure. This state file allows Terraform to understand what changes need to be made to your infrastructure when you run commands like terraform apply or terraform destroy.

##### 267. How will you manage state and how it impacts on managing infrastructure?

Managing state in Terraform involves storing information about your infrastructure in a file (commonly terraform.tfstate) and ensuring it's handled securely and consistently. The suggested way to manage Terraform state files is by using Terraform's built-in support for remote backends, such as Amazon S3, Azure Blob Storage, or Terraform Cloud. This approach provides improved collaboration, increased security, and better data protection for your infrastructure resources.  
  
This impacts infrastructure management by providing a reliable record of the deployed resources, enabling Terraform to accurately plan and execute changes, and facilitating collaboration among team members.

##### 268. List the purpose of some terraform commands.

* **init:** Initializes Terraform in a directory.
* **plan:** Shows changes Terraform will make.
* **apply:** Applies changes to infrastructure.
* **destroy:** Destroys Terraform-managed infrastructure.
* **validate:** Validates Terraform configuration files.
* **output:** Shows output values from Terraform configuration.

##### 269. What is the purpose of the Terraform plan, and how is it generated?

Terraform generates the plan by comparing the desired state defined in your configuration files with the current state of your infrastructure. It analyzes the differences and presents them in a human-readable format, detailing what resources will be created, modified, or deleted.

##### 270. Explain the significance of the "terraform apply" command and the steps involved in applying changes to the infrastructure.

* **Planning:** Terraform generates a plan to show what changes will be made.
* **Confirmation:** You confirm to proceed with the changes.
* **Execution:** Terraform applies the changes, creating, modifying, or deleting resources as necessary.
* **Completion:** Terraform provides a summary of the actions taken and any outputs generated from the changes.

##### 271. How does Terraform handle dependencies between resources?

Terraform handles dependencies between resources automatically. It figures out the order in which resources need to be created or modified based on their interdependencies specified in the configuration files. This ensures that resources are created or modified in the correct sequence to avoid any conflicts or errors.

##### 272. What are terraform variables?

Terraform variables are placeholders used to customize your infrastructure configurations. They allow you to parameterize your Terraform code, making it more flexible and reusable. Instead of hardcoding values directly into your configuration files, you can use variables to define values that can be easily changed or passed in from external sources.

##### 273. What is the use of terraform output concept?

The "terraform output" concept is used to display information about your infrastructure after it's been created or modified by Terraform. It allows you to define specific values or resources that you want to extract from your Terraform configuration and make them easily accessible for use elsewhere, such as scripts or documentation.

##### 274. What is the role of variables in Terraform, and how are they defined and used?

Terraform allow to parameterize your Terraform code, enabling customization without modifying the configuration itself.

* **Definition:** Variables are defined in Terraform configuration files using the variable block.  
  variable "region" {  
    description = "The AWS region to deploy resources"  
    default = "us-east-1"  
  }
* **Usage:** You can use variables throughout your Terraform configuration by referencing them using interpolation syntax, ${var.variable\_name}.  
  resource "aws\_instance" "example" {  
    ami = "ami-12345678"  
    instance\_type = "t2.micro"  
    availability\_zone = "${var.region}a"  
  }

##### 275. What is the difference between Terraform input variables and output variables?

Input variables are values provided to a Terraform configuration to customize its behavior, while output variables are values extracted from the Terraform state after applying the configuration.

##### 276. How can you handle sensitive information, such as API keys, tokens or passwords, in Terraform configurations?

* **Use Terraform input variables:** Prompt users to input sensitive values during terraform apply.
* **Utilize environment variables:** Store sensitive data in environment variables and reference them in your Terraform code.
* **Leverage secure storage:** Use external secret management systems like HashiCorp Vault or AWS Secrets Manager to store and retrieve sensitive information securely.
* **Avoid hardcoding:** Refrain from hardcoding sensitive data directly into Terraform configuration files.

##### 277. Explain the concept of remote backends in Terraform and why they are recommended for production environments.

You can handle sensitive information in Terraform configurations by using environment variables, external secret management systems, or by prompting users for input during terraform apply.

##### 278. How does Terraform handle the destruction of resources, and what precautions should be taken when destroying infrastructure?

Terraform handles the destruction of resources using the "terraform destroy" command, which removes all resources defined in the Terraform configuration. Precautions include reviewing the execution plan, backing up important data, and communicating with stakeholders before proceeding with the destruction.

##### 279. What is the purpose of Terraform modules, and how do they promote reusability and maintainability?

Terraform modules are reusable components that encapsulate infrastructure resources and configurations. They promote reusability and maintainability by allowing you to define and manage infrastructure in a modular way. Modules abstract away complexity, making it easier to reuse code across different projects and environments. This approach reduces duplication, improves consistency, and simplifies maintenance of infrastructure code.

##### 280. Describe a use case where Terraform's "remote-exec" provisioner might be employed.

A use case for Terraform's "remote-exec" provisioner is when you need to execute commands or scripts on a provisioned resource, such as installing software or configuring settings, that cannot be accomplished through built-in Terraform resource properties or configuration. For example, you might use the "remote-exec" provisioner to set up software dependencies or perform custom configurations on an EC2 instance after it's been provisioned.

##### 281. Explain the concept of Terraform workspaces and how they can be used to manage multiple environments (e.g., development, staging, production).

Terraform workspaces are a feature that lets you manage different environments, like development, staging, and production, within the same Terraform configuration. Each workspace maintains its own state, allowing you to deploy and manage infrastructure configurations independently for each environment. This helps organize and isolate changes, making it easier to manage multiple environments efficiently.

##### 282. How can you customize or extend providers to support additional resources or functionalities?

You can customize or extend Terraform providers by creating custom plugins using the provider's SDK. These plugins add new resources, data sources, or functionalities tailored to your needs.

##### 283. Describe the difference between local and remote provisioners in Terraform. Provide an example use case for each.

Local provisioners execute commands on the machine where Terraform is run, typically used for local tasks like installing software on a machine. Remote provisioners execute commands on resources after creation, useful for configuring settings or installing software on a provisioned resource like a server.

##### 284. How can you manage secrets and sensitive information more securely in Terraform, especially in a collaborative environment?

You can manage secrets and sensitive information securely in Terraform by using external secret management systems, storing secrets as environment variables, restricting access to sensitive data, and avoiding committing secrets to version control.

##### 285. What is the purpose of Terraform state locking, and how does it prevent conflicts when multiple users are making changes to the infrastructure concurrently?

Terraform state locking prevents conflicts by ensuring that only one user can modify the state file at a time. This prevents concurrent modifications and maintains the integrity of the state, preventing data corruption and conflicts when multiple users are working on the same infrastructure concurrently.

##### 286. Explain the significance of the count and for\_each meta-arguments in Terraform, and provide scenarios where each is more appropriate to use.

**Count meta-argument:** Allows you to create a fixed number of resource instances based on a numeric value. It's suitable when you know the exact number of instances needed beforehand, like creating a specific number of virtual machines in a cluster.  
**For\_each meta-argument:** Enables you to create multiple resource instances based on a map or set of strings. It's useful when the number of instances is dynamic or based on variable input, such as creating DNS records for different domain names specified in a map.

##### 287. How can you handle dynamic configuration or varying resource counts in Terraform, especially when dealing with auto-scaling groups or variable workloads?

To handle dynamic configuration or varying resource counts in Terraform, especially with auto-scaling groups or variable workloads, you can use Terraform's count meta-argument to dynamically create multiple instances of a resource based on a numeric value or for\_each meta-argument to create instances based on a map or set of strings. These features allow you to scale resources up or down automatically based on workload demands or changing conditions.

##### 288. Describe the differences between Terraform provisioners and modules. When would you choose one over the other?

|  |  |
| --- | --- |
| **Terraform provisioners** execute scripts or commands on remote resources during or after resource creation. They're used for tasks like bootstrapping or configuration management. | **Terraform modules** are reusable collections of Terraform configurations representing a set of related resources. They promote modularity and reusability. |

##### 289. What is the purpose of the terraform validate command, and how does it contribute to the development workflow in Terraform?

The purpose of the terraform validate command is to check the syntax and validity of Terraform configuration files. It ensures that the configuration files adhere to the correct Terraform syntax and structure. This command is crucial in the development workflow as it helps catch syntax errors early in the development process, reducing the likelihood of errors during deployment.

##### 290. Explain the role of remote backends in Terraform, and discuss the pros and cons of using different types of remote backends (e.g., AWS S3, Azure Storage, HashiCorp Consul).

Terraform serve as storage mechanisms for the Terraform state file, allowing it to be stored remotely rather than locally on your machine.

* **AWS S3:** Benefits include scalability, durability, and security provided by AWS. It integrates well with other AWS services. However, it may incur costs based on usage.
* **Azure Storage:** Similar to S3, Azure Storage offers scalability, durability, and security within the Azure ecosystem. It integrates seamlessly with other Azure services. However, it may also incur costs based on usage.
* **HashiCorp Consul:** Provides a distributed key-value store with built-in support for locking and consistency. It's suitable for large-scale, distributed environments. However, it requires additional infrastructure and operational overhead compared to cloud storage options.

##### 291. How can you promote code reuse and modularity in Terraform configurations using variables, locals, and data sources?

* **Variables:** Define input variables to make configurations more flexible and reusable across different environments or use cases.
* **Locals:** Use locals to define reusable expressions or values within a Terraform configuration, reducing duplication and improving readability.
* **Data sources:** Utilize data sources to fetch dynamic information from external systems, such as cloud provider APIs, enabling configurations to adapt to varying environments or requirements without hardcoding values.AWS S3: Benefits include scalability, durability, and security provided by AWS. It integrates well with other AWS services. However, it may incur costs based on usage.

##### 292. What is the purpose of the terraform fmt command, and how does it help maintain a consistent and readable code style across Terraform configurations?

The purpose of the terraform fmt command is to format Terraform configuration files according to a consistent style defined by Terraform's conventions. It helps maintain a consistent and readable code style across Terraform configurations by automatically applying indentation, line breaks, and other formatting rules.

##### 293. How can you achieve idempotence in Terraform configurations, and why is it important in the context of infrastructure management?

Idempotence in Terraform configurations means that running the same configuration multiple times results in the same outcome. It's important in infrastructure management because it ensures consistency and predictability, preventing unintended changes and maintaining the desired state of the infrastructure.

##### 294. Explain the difference between Terraform's taint and destroy commands. When and why might you use each of them?

* **Terraform's taint** command marks a resource for recreation during the next terraform apply. It essentially flags the resource as needing to be recreated, which can be useful if the resource is in a problematic state or needs to be reconfigured.
* **Terraform's destroy command** removes all resources defined in the Terraform configuration, effectively tearing down the infrastructure. It's used when you want to completely remove all resources, such as when decommissioning an environment or cleaning up after testing.You might use the taint command when you want to force the recreation of a specific resource without destroying all resources. This can be helpful for troubleshooting or when a resource is stuck in an undesired state.You would use the destroy command when you want to remove all resources defined in the Terraform configuration, such as when decommissioning an environment or cleaning up after testing.

##### 295. Describe a scenario where Terraform's count or for\_each would be used to create multiple instances of the same resource type, and how you would manage them dynamically.

**Scenario:**  
You need to provision multiple AWS S3 buckets to store files for different departments in your organization. Each department requires its own dedicated S3 bucket.  
**Solution:**  
You can use Terraform's for\_each meta-argument to create multiple instances of the same S3 bucket resource type. For example, you could define a map variable where each key represents a department name and the corresponding value represents the configuration options for the S3 bucket. Then, you can use for\_each to dynamically create S3 buckets based on the map variable. This allows you to easily manage the S3 buckets for each department dynamically, adjusting the configurations as needed without modifying the Terraform configuration each time.

##### 296. What is Kubernetes?

Kubernetes is a container orchestration platform that automates the deployment, scaling, and management of containerized applications. It helps in managing and coordinating multiple containers across a cluster of machines.

##### 297. What problem does K8s solve in the context of containerized applications?

Kubernetes (K8s) simplifies the management of containerized applications by automating tasks like deployment, scaling, and monitoring, making them easier to run and maintain.

##### 298. Can you explain the architecture of Kubernetes and its key components?

**Master Node:**

* **API Server:** Handles requests and controls the cluster.
* **Controller Manager:** Manages cluster functions like scaling and updates.
* **Scheduler:** Assigns workloads to nodes based on resource availability.

**Worker Nodes:**

* **Kubelet:** Manages containers on the node.
* **Container Runtime:** Software to run containers (e.g., Docker).
* **Kube Proxy:** Handles network communication for pods.
* **Pods:** Basic units that hold containers.
* **Controllers:** Manage workload lifecycle (e.g., Deployment for scaling).
* **Services:** Networking for accessing applications in the cluster.

##### 299. What is a Pod in Kubernetes and how does it relate to containers?

A Pod in Kubernetes is a group of one or more containers that share resources and are deployed together. It's like a tiny ecosystem for your containers, providing them with a common network and storage space. Pods make it easier to manage and deploy related containers that need to work together.

##### 300. How does Kubernetes handle scaling applications horizontally and vertically?

|  |  |
| --- | --- |
| **Horizontal Scaling** | **Vertical Scaling** |
| It adds or removes more instances (pods) of an application to match changing demand, ensuring consistent performance and resource usage across the cluster. | It adjusts the resources (CPU and memory) allocated to individual pods based on their workload requirements, optimizing performance for each pod as needed. |

##### 301. What is a Kubernetes Service and how does it enable communication between different parts of an application?

A Kubernetes Service is an abstraction that defines a logical set of pods and a policy by which to access them. It acts as a stable endpoint to access your application, regardless of the underlying pod instances.  
  
Services enable communication between different parts of an application by providing a consistent way to access pods, even as they scale up or down or move across nodes. They use labels and selectors to group pods and route traffic to them, making it easier for components within an application to communicate reliably.

##### 302. How does Kubernetes ensure high availability and fault tolerance for applications?

Kubernetes ensures high availability and fault tolerance for applications by automatically managing multiple replicas of pods, monitoring their health, and restarting or rescheduling failed pods on healthy nodes. It also provides load balancing and supports rolling updates and rollbacks for seamless application management.

##### 303. What role does the Kubernetes Master play in a Kubernetes cluster?

* **API Server:** Acts as the front-end for the Kubernetes control plane, receiving and processing API requests from users, controllers, and other components.
* **Scheduler:** Assigns pods to nodes based on resource requirements, policies, and constraints specified by users or controllers.
* **Controller Manager:** Manages various controllers responsible for maintaining the desired state of the cluster, such as node controller, replication controller, endpoint controller, and more.
* **etcd:** Stores the cluster's configuration data and the current state of the cluster, serving as the cluster's database.

##### 304. What are Kubernetes Nodes and what components do they consist of?

Kubernetes Nodes are individual machines (physical or virtual) in a Kubernetes cluster where containers are deployed and run. They consist of the following components.

* **Kubelet:** Manages containers on the node and communicates with the Kubernetes Master.
* **Container Runtime:** Software (e.g., Docker, containerd) that runs containers.
* **Kube Proxy:** Handles network communication for pods running on the node.

##### 305. How does Kubernetes manage storage for applications running within its cluster?

Kubernetes manages storage for applications running within its cluster through Persistent Volumes (PVs) and Persistent Volume Claims (PVCs). PVs represent storage resources, while PVCs are requests for storage by applications. Kubernetes can dynamically provision PVs based on PVCs, ensuring efficient and flexible storage management.

##### 306. What is a Deployment in Kubernetes and how does it help in managing application updates and rollbacks?

A Deployment in Kubernetes manages the rollout and scaling of application pods. It helps in managing updates by allowing for rolling updates, which replace old pods with new ones gradually to avoid downtime. Deployments also enable rollbacks to previous versions if there are issues with the update, ensuring a smooth and reliable application deployment process.

##### 307. How does Kubernetes handle networking between different Pods and Services?

|  |  |
| --- | --- |
| Pod Networking | Each Pod has its own IP address, allowing direct communication between Pods within the cluster. |
| Service Networking | Kubernetes Services provide a stable endpoint for accessing Pods, using selectors for routing traffic. |

##### 308. What is a Namespace in Kubernetes and how is it used to organize resources?

A Namespace in Kubernetes is a way to logically divide a single Kubernetes cluster into multiple virtual clusters. It's used to organize resources by providing isolation and separation between different applications or environments within the same cluster. This helps in managing and maintaining large-scale deployments more efficiently by grouping related resources together and applying resource quotas and access controls specific to each Namespace.

##### 309. How does Kubernetes handle security and access control within a cluster?

Kubernetes uses Role-Based Access Control (RBAC), Service Accounts, Network Policies, Secrets Management, and Pod Security Policies to manage security and access control within a cluster. These mechanisms ensure that only authorized users and components can access resources and communicate with each other, enhancing the overall security of the cluster.

##### 310. Can you explain the concept of Labels and Selectors in Kubernetes and their significance?

Labels in Kubernetes are like sticky notes you attach to objects (like pods, services, and deployments) to categorize them. Selectors are tools that help you find and group objects based on these labels.

* **Organize:** Labels group similar things together, making it easy to manage and find them later.
* **Select:** Selectors help you pick out objects with specific labels, which is handy for tasks like scaling or applying updates.
* **Automate:** Labels and selectors are crucial for automation, allowing tools to work with objects based on their labels, streamlining tasks, and ensuring consistency.

##### 311. How does Kubernetes handle resource allocation and scheduling of workloads within a cluster?

Kubernetes manages resource allocation by letting pods specify their resource needs and limits, and uses schedulers to place them on nodes based on available resources. It also supports automatic scaling to adjust resources as workload demands change, ensuring efficient allocation and scheduling within the cluster.

##### 312. What are the key components of a Kubernetes cluster?

Master Node, Worker Nodes, Kubelet, Kube Proxy, etcd, Controller Manager, Scheduler and API Server.

##### 313. How does Kubernetes manage containerized applications?

|  |  |
| --- | --- |
| Deploying Containers | It schedules containers to run on nodes in the cluster. |
| Scaling | It can automatically scale applications based on resource usage. |
| Load Balancing | It balances traffic between containers for high availability. |
| Health Monitoring | It checks the health of containers and restarts failed ones. |
| Rolling Updates | It updates applications without downtime using rolling updates. |
| Resource Allocation | It allocates resources like CPU and memory to containers based on requirements. |

##### 314. What is a Pod in Kubernetes, and how does it relate to containers?

A Pod in Kubernetes is a group of one or more containers that share resources and are deployed together. It's the basic unit for running containers in Kubernetes, allowing them to work together closely.

##### 315. Explain the role of a Deployment in Kubernetes.

A Deployment in Kubernetes manages the deployment and scaling of applications. It ensures that a specified number of identical pods are running and handles updates and rollbacks, making application management easier.

##### 316. What is a Service in Kubernetes, and why is it important for networking?

A Service in Kubernetes is a way to access and communicate with pods. It's important for networking because it provides a stable endpoint, enables load balancing, and facilitates service discovery, making applications more reliable and easier to manage.

##### 317. What are Kubernetes Namespaces, and why are they used?

Kubernetes Namespaces are virtual clusters that help organize and isolate resources within a Kubernetes cluster. They are used to separate applications, teams, or projects, making it easier to manage and secure resources independently.

##### 318. Describe the concept of Persistent Volumes in Kubernetes.

Persistent Volumes (PVs) in Kubernetes are storage resources that provide persistent storage for applications. They are managed independently of pods and can be dynamically provisioned and controlled by the cluster administrator. PVs simplify storage management by decoupling storage from application lifecycles.

##### 319. What is the purpose of a ConfigMap in Kubernetes, and how is it used?

A ConfigMap in Kubernetes is used to decouple configuration data from containerized applications. Its purpose is to store key-value pairs, environment variables, or configuration files that can be injected into pods at runtime. ConfigMaps help in maintaining configuration consistency across different environments and make it easier to update configuration settings without changing the application code.

##### 320. Explain the difference between a StatefulSet and a Deployment in Kubernetes.

|  |  |
| --- | --- |
| **StatefulSet** | **Deployment** |
| Manages stateful applications where each pod has a unique identity and persists data. | Manages stateless applications where each pod is interchangeable and does not maintain state. |
| Pods managed by StatefulSet have stable network identifiers and persistent storage. | Pods managed by Deployment do not have stable identities or persistent storage. |
| Pods are created and terminated in a predictable, ordered manner, suitable for databases or clustered applications. | Pods can be scaled up or down, updated, and rolled back easily. |

##### 321. How does Kubernetes handle 'rolling updates' and rollbacks of application deployments?

Kubernetes handles rolling updates by creating a new version of the application while gradually shifting traffic to it, ensuring minimal downtime. If issues occur, Kubernetes can roll back to the previous version by reverting to the old configuration.

##### 322. What are Labels and Selectors in Kubernetes, and why are they significant?

Labels in Kubernetes are key-value pairs attached to objects like pods or services for identification and categorization. Selectors are used to filter and target objects based on their labels. They are significant because they enable flexible management, grouping of resources, and support various Kubernetes features like service discovery and deployment strategies.

##### 323. Describe the role of a Service Mesh like Istio in a Kubernetes environment.

A Service Mesh like Istio in a Kubernetes environment provides advanced networking, security, and observability features for managing communication between services. It enhances control over traffic routing, enforces security policies, and offers monitoring tools for better management of microservices.

##### 324. How can you monitor and troubleshoot applications running in a Kubernetes cluster?

* Using monitoring tools like Prometheus and Grafana to track metrics.
* Utilizing centralized logging with tools like Elasticsearch and Kibana.
* Implementing distributed tracing with tools like Jaeger.
* Configuring health checks for applications.
* Using debugging tools like kubectl exec and kubectl logs.
* Setting up alerting rules for critical issues.
* Monitoring resource usage and performance.

##### 325. What is the different between node selector and node affinity?

|  |  |
| --- | --- |
| **Node Selector** | **Node Affinity** |
| Basic method for node selection based on labels. | Advanced method for node selection based on label rules and expressions. |
| Directly specified in pod configurations using nodeSelector field. | Allows more complex criteria for pod placement. |
| Matches exact label values. | Specified in pod specifications' affinity field. |
| Limited to straightforward node selection based on predefined labels. | Includes options like required rules and preferred preferences for node selection, offering finer control over scheduling. |

##### 326. What is Terraform?

Terraform is a tool to create and control infrastructure(computer networks, storage, software applications, and other digital services) using simple instructions. It can manage these systems across various cloud providers like AWS, Azure, and GCP, all using just one tool.  
These simple instructions are known as code. If we have code, then we can create infrastructure. So, we have infrastructure in the form of code, meaning Infrastructure as Code (IaC).  
  
Terraform is open-source, written in Golang, and created by HashiCorp, offering features like idempotency and scalability for managing infrastructure deployments efficiently.

##### 327. What is IaC?

IaC stands for "Infrastructure as Code". It's a way to use code to set up and manage computer systems, like servers and networks, instead of doing it manually. By writing code, they can automate these tasks, which makes managing infrastructure easier, faster, and more reliable. It also helps ensure consistency and allows for easier tracking of changes.

##### 328. What is IaS?

IaC stands for Infrastructure as a Service, is a cloud computing model where users can rent virtualized computing resources like servers and storage over the internet. Instead of owning and maintaining physical hardware, organizations can access and manage these resources as needed. IaaS providers handle hardware maintenance, offering scalability, and flexibility. This enables businesses to focus on their applications without the hassle of managing physical infrastructure.

##### 329. What purpose does Terraform serve in the realm of infrastructure as code (IaC)?

Terraform serves as a fundamental tool for automating the provisioning, deployment, and management of infrastructure resources. Its primary purpose is to streamline and standardize the process of infrastructure configuration through code. Here's how Terraform serves in the realm of IaC:

* Automation of Infrastructure Deployment
* Scalability and Consistency
* Version Control and Auditing
* Multi-Cloud and Hybrid Cloud Support
* Dependency Management and Execution Plans
* Modularity and Reusability
* Infrastructure as Code Best Practices

##### 330. Explain the difference between declarative and imperative programming?

|  |  |
| --- | --- |
| **Declarative** | **Imperative** |
| Focuses on what the program should achieve without detailing the process. | Explicitly states how the program should achieve the desired result. |
| Expresses the logic of a computation without revealing its control flow or state management. | Involves writing step-by-step instructions for the computer to follow. |
| Defines the desired result and lets the system determine the steps to achieve it. | Closer to the machine's way of thinking, requiring a clear understanding of system internals. |
| Examples include SQL for database management and HTML/CSS for front-end UI design. | Commonly used in languages like C, Java, and Python. |

##### 331. Does Terraform fits into declarative or imperative paradigm?

Terraform primarily fits into the declarative paradigm of programming. While it has some imperative-like features such as for loops, dynamic blocks, and complex data structures, Terraform's core approach is declarative. In declarative programming, the focus is on defining the desired outcome without specifying the exact steps to achieve it, allowing the system to determine how to reach that state efficiently.

##### 332. What are providers?

Terraform, providers are plugins that interact with APIs and manage resources within specific cloud or service providers. These providers enable Terraform to provision and manage infrastructure resources across various platforms like AWS, Azure, Google Cloud, and more.

##### 333. Why AWS-CLI is used?

AWS CLI (Command Line Interface) is used to interact with Amazon Web Services from the command line. It allows users to perform various tasks such as managing EC2 instances, S3 buckets, IAM roles, and more, directly from the terminal. AWS CLI provides a convenient and efficient way to automate tasks, script operations, and integrate AWS services into workflows without needing to use the AWS Management Console.

##### 334. How terraform contribute to managing resources in different cloud platforms or infrastructure providers?

Terraform simplifies managing resources across different cloud platforms or infrastructure providers by providing a unified way to define infrastructure configurations. With Terraform, users can use a single set of configuration files to provision and manage resources on AWS, Azure, Google Cloud, and other providers. This allows for consistency, scalability, and ease of management, as changes can be applied uniformly across diverse environments without the need for platform-specific tools or scripts.

##### 335. What is the best way to store cloud providers (ex: AWS, GCP or Azure) Access Key and Secrect Key?

The best way to store cloud provider access keys and secret keys is to use a secure and centralized secrets management service provided by the respective cloud platform. For example, AWS provides AWS Secrets Manager, Google Cloud offers Secret Manager, and Azure offers Azure Key Vault. These services encrypt and store sensitive credentials, allowing for secure access and management while ensuring compliance with security best practices.

##### 336. Describe the basic structure of a Terraform configuration file?

Terraform configuration file define the infrastructure resources that Terraform will manage.  
Example:  
resource "aws\_instance" "example" {  
    ami = "ami-0c55b159cbfafe1f0"  
    instance\_type = "t2.micro"  
}

##### 337. What language is used to write terraform configurations file?

Terraform configuration files are primarily written in HashiCorp Configuration Language (HCL).

##### 338. What is 'state' in terraform?

In Terraform, "state" refers to a snapshot of your infrastructure's current status and configuration. It's stored in a file named 'terraform.tfstate' by default and helps Terraform to keep track of what resources are created and how they're configured.

##### 339. How does Terraform maintain state?

Terraform maintains state by keeping track of the current state of your infrastructure in a state file. This file is usually stored locally or remotely (e.g., in cloud storage) and records the mapping between the resources in your configuration files and the real-world infrastructure. This state file allows Terraform to understand what changes need to be made to your infrastructure when you run commands like terraform apply or terraform destroy.

##### 340. How will you manage state and how it impacts on managing infrastructure?

Managing state in Terraform involves storing information about your infrastructure in a file (commonly terraform.tfstate) and ensuring it's handled securely and consistently. The suggested way to manage Terraform state files is by using Terraform's built-in support for remote backends, such as Amazon S3, Azure Blob Storage, or Terraform Cloud. This approach provides improved collaboration, increased security, and better data protection for your infrastructure resources.  
  
This impacts infrastructure management by providing a reliable record of the deployed resources, enabling Terraform to accurately plan and execute changes, and facilitating collaboration among team members.

##### 341. List the purpose of some terraform commands.

* **init:** Initializes Terraform in a directory.
* **plan:** Shows changes Terraform will make.
* **apply:** Applies changes to infrastructure.
* **destroy:** Destroys Terraform-managed infrastructure.
* **validate:** Validates Terraform configuration files.
* **output:** Shows output values from Terraform configuration.

##### 342. What is the purpose of the Terraform plan, and how is it generated?

Terraform generates the plan by comparing the desired state defined in your configuration files with the current state of your infrastructure. It analyzes the differences and presents them in a human-readable format, detailing what resources will be created, modified, or deleted.

##### 343. Explain the significance of the "terraform apply" command and the steps involved in applying changes to the infrastructure.

* **Planning:** Terraform generates a plan to show what changes will be made.
* **Confirmation:** You confirm to proceed with the changes.
* **Execution:** Terraform applies the changes, creating, modifying, or deleting resources as necessary.
* **Completion:** Terraform provides a summary of the actions taken and any outputs generated from the changes.

##### 344. How does Terraform handle dependencies between resources?

Terraform handles dependencies between resources automatically. It figures out the order in which resources need to be created or modified based on their interdependencies specified in the configuration files. This ensures that resources are created or modified in the correct sequence to avoid any conflicts or errors.

##### 345. What are terraform variables?

Terraform variables are placeholders used to customize your infrastructure configurations. They allow you to parameterize your Terraform code, making it more flexible and reusable. Instead of hardcoding values directly into your configuration files, you can use variables to define values that can be easily changed or passed in from external sources.

##### 346. What is the use of terraform output concept?

The "terraform output" concept is used to display information about your infrastructure after it's been created or modified by Terraform. It allows you to define specific values or resources that you want to extract from your Terraform configuration and make them easily accessible for use elsewhere, such as scripts or documentation.

##### 347. What is the role of variables in Terraform, and how are they defined and used?

Terraform allow to parameterize your Terraform code, enabling customization without modifying the configuration itself.

* **Definition:** Variables are defined in Terraform configuration files using the variable block.  
  variable "region" {  
    description = "The AWS region to deploy resources"  
    default = "us-east-1"  
  }
* **Usage:** You can use variables throughout your Terraform configuration by referencing them using interpolation syntax, ${var.variable\_name}.  
  resource "aws\_instance" "example" {  
    ami = "ami-12345678"  
    instance\_type = "t2.micro"  
    availability\_zone = "${var.region}a"  
  }

##### 348. What is the difference between Terraform input variables and output variables?

Input variables are values provided to a Terraform configuration to customize its behavior, while output variables are values extracted from the Terraform state after applying the configuration.

##### 349. How can you handle sensitive information, such as API keys, tokens or passwords, in Terraform configurations?

* **Use Terraform input variables:** Prompt users to input sensitive values during terraform apply.
* **Utilize environment variables:** Store sensitive data in environment variables and reference them in your Terraform code.
* **Leverage secure storage:** Use external secret management systems like HashiCorp Vault or AWS Secrets Manager to store and retrieve sensitive information securely.
* **Avoid hardcoding:** Refrain from hardcoding sensitive data directly into Terraform configuration files.

##### 350. Explain the concept of remote backends in Terraform and why they are recommended for production environments.

You can handle sensitive information in Terraform configurations by using environment variables, external secret management systems, or by prompting users for input during terraform apply.

##### 351. How does Terraform handle the destruction of resources, and what precautions should be taken when destroying infrastructure?

Terraform handles the destruction of resources using the "terraform destroy" command, which removes all resources defined in the Terraform configuration. Precautions include reviewing the execution plan, backing up important data, and communicating with stakeholders before proceeding with the destruction.

##### 352. What is the purpose of Terraform modules, and how do they promote reusability and maintainability?

Terraform modules are reusable components that encapsulate infrastructure resources and configurations. They promote reusability and maintainability by allowing you to define and manage infrastructure in a modular way. Modules abstract away complexity, making it easier to reuse code across different projects and environments. This approach reduces duplication, improves consistency, and simplifies maintenance of infrastructure code.

##### 353. Describe a use case where Terraform's "remote-exec" provisioner might be employed.

A use case for Terraform's "remote-exec" provisioner is when you need to execute commands or scripts on a provisioned resource, such as installing software or configuring settings, that cannot be accomplished through built-in Terraform resource properties or configuration. For example, you might use the "remote-exec" provisioner to set up software dependencies or perform custom configurations on an EC2 instance after it's been provisioned.

##### 354. Explain the concept of Terraform workspaces and how they can be used to manage multiple environments (e.g., development, staging, production).

Terraform workspaces are a feature that lets you manage different environments, like development, staging, and production, within the same Terraform configuration. Each workspace maintains its own state, allowing you to deploy and manage infrastructure configurations independently for each environment. This helps organize and isolate changes, making it easier to manage multiple environments efficiently.

##### 355. How can you customize or extend providers to support additional resources or functionalities?

You can customize or extend Terraform providers by creating custom plugins using the provider's SDK. These plugins add new resources, data sources, or functionalities tailored to your needs.

##### 356. Describe the difference between local and remote provisioners in Terraform. Provide an example use case for each.

Local provisioners execute commands on the machine where Terraform is run, typically used for local tasks like installing software on a machine. Remote provisioners execute commands on resources after creation, useful for configuring settings or installing software on a provisioned resource like a server.

##### 357. How can you manage secrets and sensitive information more securely in Terraform, especially in a collaborative environment?

You can manage secrets and sensitive information securely in Terraform by using external secret management systems, storing secrets as environment variables, restricting access to sensitive data, and avoiding committing secrets to version control.

##### 358. What is the purpose of Terraform state locking, and how does it prevent conflicts when multiple users are making changes to the infrastructure concurrently?

Terraform state locking prevents conflicts by ensuring that only one user can modify the state file at a time. This prevents concurrent modifications and maintains the integrity of the state, preventing data corruption and conflicts when multiple users are working on the same infrastructure concurrently.

##### 359. Explain the significance of the count and for\_each meta-arguments in Terraform, and provide scenarios where each is more appropriate to use.

**Count meta-argument:** Allows you to create a fixed number of resource instances based on a numeric value. It's suitable when you know the exact number of instances needed beforehand, like creating a specific number of virtual machines in a cluster.  
**For\_each meta-argument:** Enables you to create multiple resource instances based on a map or set of strings. It's useful when the number of instances is dynamic or based on variable input, such as creating DNS records for different domain names specified in a map.

##### 360. How can you handle dynamic configuration or varying resource counts in Terraform, especially when dealing with auto-scaling groups or variable workloads?

To handle dynamic configuration or varying resource counts in Terraform, especially with auto-scaling groups or variable workloads, you can use Terraform's count meta-argument to dynamically create multiple instances of a resource based on a numeric value or for\_each meta-argument to create instances based on a map or set of strings. These features allow you to scale resources up or down automatically based on workload demands or changing conditions.

##### 361. Describe the differences between Terraform provisioners and modules. When would you choose one over the other?

|  |  |
| --- | --- |
| **Terraform provisioners** execute scripts or commands on remote resources during or after resource creation. They're used for tasks like bootstrapping or configuration management. | **Terraform modules** are reusable collections of Terraform configurations representing a set of related resources. They promote modularity and reusability. |

##### 362. What is the purpose of the terraform validate command, and how does it contribute to the development workflow in Terraform?

The purpose of the terraform validate command is to check the syntax and validity of Terraform configuration files. It ensures that the configuration files adhere to the correct Terraform syntax and structure. This command is crucial in the development workflow as it helps catch syntax errors early in the development process, reducing the likelihood of errors during deployment.

##### 363. Explain the role of remote backends in Terraform, and discuss the pros and cons of using different types of remote backends (e.g., AWS S3, Azure Storage, HashiCorp Consul).

Terraform serve as storage mechanisms for the Terraform state file, allowing it to be stored remotely rather than locally on your machine.

* **AWS S3:** Benefits include scalability, durability, and security provided by AWS. It integrates well with other AWS services. However, it may incur costs based on usage.
* **Azure Storage:** Similar to S3, Azure Storage offers scalability, durability, and security within the Azure ecosystem. It integrates seamlessly with other Azure services. However, it may also incur costs based on usage.
* **HashiCorp Consul:** Provides a distributed key-value store with built-in support for locking and consistency. It's suitable for large-scale, distributed environments. However, it requires additional infrastructure and operational overhead compared to cloud storage options.

##### 364. How can you promote code reuse and modularity in Terraform configurations using variables, locals, and data sources?

* **Variables:** Define input variables to make configurations more flexible and reusable across different environments or use cases.
* **Locals:** Use locals to define reusable expressions or values within a Terraform configuration, reducing duplication and improving readability.
* **Data sources:** Utilize data sources to fetch dynamic information from external systems, such as cloud provider APIs, enabling configurations to adapt to varying environments or requirements without hardcoding values.AWS S3: Benefits include scalability, durability, and security provided by AWS. It integrates well with other AWS services. However, it may incur costs based on usage.

##### 365. What is the purpose of the terraform fmt command, and how does it help maintain a consistent and readable code style across Terraform configurations?

The purpose of the terraform fmt command is to format Terraform configuration files according to a consistent style defined by Terraform's conventions. It helps maintain a consistent and readable code style across Terraform configurations by automatically applying indentation, line breaks, and other formatting rules.

##### 366. How can you achieve idempotence in Terraform configurations, and why is it important in the context of infrastructure management?

Idempotence in Terraform configurations means that running the same configuration multiple times results in the same outcome. It's important in infrastructure management because it ensures consistency and predictability, preventing unintended changes and maintaining the desired state of the infrastructure.

##### 367. Explain the difference between Terraform's taint and destroy commands. When and why might you use each of them?

* **Terraform's taint** command marks a resource for recreation during the next terraform apply. It essentially flags the resource as needing to be recreated, which can be useful if the resource is in a problematic state or needs to be reconfigured.
* **Terraform's destroy command** removes all resources defined in the Terraform configuration, effectively tearing down the infrastructure. It's used when you want to completely remove all resources, such as when decommissioning an environment or cleaning up after testing.
* ou would use the destroy command when you want to remove all resources defined in the Terraform configuration, such as when decommissioning an environment or cleaning up after testing.

##### 368. Describe a scenario where Terraform's count or for\_each would be used to create multiple instances of the same resource type, and how you would manage them dynamically.

* **Scenario:**  
  You need to provision multiple AWS S3 buckets to store files for different departments in your organization. Each department requires its own dedicated S3 bucket.  
  **Solution:**  
  You can use Terraform's for\_each meta-argument to create multiple instances of the same S3 bucket resource type. For example, you could define a map variable where each key represents a department name and the corresponding value represents the configuration options for the S3 bucket. Then, you can use for\_each to dynamically create S3 buckets based on the map variable. This allows you to easily manage the S3 buckets for each department dynamically, adjusting the configurations as needed without modifying the Terraform configuration each time.

##### 369. What is Kubernetes?

* Kubernetes is a container orchestration platform that automates the deployment, scaling, and management of containerized applications. It helps in managing and coordinating multiple containers across a cluster of machines.

##### 370. What problem does K8s solve in the context of containerized applications?

* Kubernetes (K8s) simplifies the management of containerized applications by automating tasks like deployment, scaling, and monitoring, making them easier to run and maintain.

##### 371. Can you explain the architecture of Kubernetes and its key components?

**Master Node:**

* **API Server:** Handles requests and controls the cluster.
* **Controller Manager:** Manages cluster functions like scaling and updates.
* **Scheduler:** Assigns workloads to nodes based on resource availability.

**Worker Nodes:**

* **Kubelet:** Manages containers on the node.
* **Container Runtime:** Software to run containers (e.g., Docker).
* **Kube Proxy:** Handles network communication for pods.
* **Pods:** Basic units that hold containers.
* **Controllers:** Manage workload lifecycle (e.g., Deployment for scaling).
* **Services:** Networking for accessing applications in the cluster.

##### 372. What is a Pod in Kubernetes and how does it relate to containers?

A Pod in Kubernetes is a group of one or more containers that share resources and are deployed together. It's like a tiny ecosystem for your containers, providing them with a common network and storage space. Pods make it easier to manage and deploy related containers that need to work together.

##### 373. How does Kubernetes handle scaling applications horizontally and vertically?

|  |  |
| --- | --- |
| **Horizontal Scaling** | **Vertical Scaling** |
| It adds or removes more instances (pods) of an application to match changing demand, ensuring consistent performance and resource usage across the cluster. | It adjusts the resources (CPU and memory) allocated to individual pods based on their workload requirements, optimizing performance for each pod as needed. |

##### 374. What is a Kubernetes Service and how does it enable communication between different parts of an application?

A Kubernetes Service is an abstraction that defines a logical set of pods and a policy by which to access them. It acts as a stable endpoint to access your application, regardless of the underlying pod instances.  
  
Services enable communication between different parts of an application by providing a consistent way to access pods, even as they scale up or down or move across nodes. They use labels and selectors to group pods and route traffic to them, making it easier for components within an application to communicate reliably.

##### 375. How does Kubernetes ensure high availability and fault tolerance for applications?

Kubernetes ensures high availability and fault tolerance for applications by automatically managing multiple replicas of pods, monitoring their health, and restarting or rescheduling failed pods on healthy nodes. It also provides load balancing and supports rolling updates and rollbacks for seamless application management.

##### 376. What role does the Kubernetes Master play in a Kubernetes cluster?

* **API Server:** Acts as the front-end for the Kubernetes control plane, receiving and processing API requests from users, controllers, and other components.
* **Scheduler:** Assigns pods to nodes based on resource requirements, policies, and constraints specified by users or controllers.
* **Controller Manager:** Manages various controllers responsible for maintaining the desired state of the cluster, such as node controller, replication controller, endpoint controller, and more.
* **etcd:** Stores the cluster's configuration data and the current state of the cluster, serving as the cluster's database.

##### 377. What are Kubernetes Nodes and what components do they consist of?

Kubernetes Nodes are individual machines (physical or virtual) in a Kubernetes cluster where containers are deployed and run. They consist of the following components.

* **Kubelet:** Manages containers on the node and communicates with the Kubernetes Master.
* **Container Runtime:** Software (e.g., Docker, containerd) that runs containers.
* **Kube Proxy:** Handles network communication for pods running on the node.

##### 378. How does Kubernetes manage storage for applications running within its cluster?

Kubernetes manages storage for applications running within its cluster through Persistent Volumes (PVs) and Persistent Volume Claims (PVCs). PVs represent storage resources, while PVCs are requests for storage by applications. Kubernetes can dynamically provision PVs based on PVCs, ensuring efficient and flexible storage management.

##### 379. What is a Deployment in Kubernetes and how does it help in managing application updates and rollbacks?

A Deployment in Kubernetes manages the rollout and scaling of application pods. It helps in managing updates by allowing for rolling updates, which replace old pods with new ones gradually to avoid downtime. Deployments also enable rollbacks to previous versions if there are issues with the update, ensuring a smooth and reliable application deployment process.

##### 380. How does Kubernetes handle networking between different Pods and Services?

|  |  |
| --- | --- |
| Pod Networking | Each Pod has its own IP address, allowing direct communication between Pods within the cluster. |
| Service Networking | Kubernetes Services provide a stable endpoint for accessing Pods, using selectors for routing traffic. |

##### 381. What is a Namespace in Kubernetes and how is it used to organize resources?

A Namespace in Kubernetes is a way to logically divide a single Kubernetes cluster into multiple virtual clusters. It's used to organize resources by providing isolation and separation between different applications or environments within the same cluster. This helps in managing and maintaining large-scale deployments more efficiently by grouping related resources together and applying resource quotas and access controls specific to each Namespace.

##### 382. How does Kubernetes handle security and access control within a cluster?

Kubernetes uses Role-Based Access Control (RBAC), Service Accounts, Network Policies, Secrets Management, and Pod Security Policies to manage security and access control within a cluster. These mechanisms ensure that only authorized users and components can access resources and communicate with each other, enhancing the overall security of the cluster.

##### 383. Can you explain the concept of Labels and Selectors in Kubernetes and their significance?

Labels in Kubernetes are like sticky notes you attach to objects (like pods, services, and deployments) to categorize them. Selectors are tools that help you find and group objects based on these labels.

* **Organize:** Labels group similar things together, making it easy to manage and find them later.
* **Select:** Selectors help you pick out objects with specific labels, which is handy for tasks like scaling or applying updates.
* **Automate:** Labels and selectors are crucial for automation, allowing tools to work with objects based on their labels, streamlining tasks, and ensuring consistency.

##### 384. How does Kubernetes handle resource allocation and scheduling of workloads within a cluster?

Kubernetes manages resource allocation by letting pods specify their resource needs and limits, and uses schedulers to place them on nodes based on available resources. It also supports automatic scaling to adjust resources as workload demands change, ensuring efficient allocation and scheduling within the cluster.

##### 385. What are the key components of a Kubernetes cluster?

Master Node, Worker Nodes, Kubelet, Kube Proxy, etcd, Controller Manager, Scheduler and API Server.

##### 386. How does Kubernetes manage containerized applications?

|  |  |
| --- | --- |
| Deploying Containers | It schedules containers to run on nodes in the cluster. |
| Scaling | It can automatically scale applications based on resource usage. |
| Load Balancing | It balances traffic between containers for high availability. |
| Health Monitoring | It checks the health of containers and restarts failed ones. |
| Rolling Updates | It updates applications without downtime using rolling updates. |
| Resource Allocation | It allocates resources like CPU and memory to containers based on requirements. |

##### 387. What is a Pod in Kubernetes, and how does it relate to containers?

A Pod in Kubernetes is a group of one or more containers that share resources and are deployed together. It's the basic unit for running containers in Kubernetes, allowing them to work together closely.

##### 388. Explain the role of a Deployment in Kubernetes.

A Deployment in Kubernetes manages the deployment and scaling of applications. It ensures that a specified number of identical pods are running and handles updates and rollbacks, making application management easier.

##### 389. What is a Service in Kubernetes, and why is it important for networking?

A Service in Kubernetes is a way to access and communicate with pods. It's important for networking because it provides a stable endpoint, enables load balancing, and facilitates service discovery, making applications more reliable and easier to manage.

##### 390. What are Kubernetes Namespaces, and why are they used?

Kubernetes Namespaces are virtual clusters that help organize and isolate resources within a Kubernetes cluster. They are used to separate applications, teams, or projects, making it easier to manage and secure resources independently.

##### 391. Describe the concept of Persistent Volumes in Kubernetes.

Persistent Volumes (PVs) in Kubernetes are storage resources that provide persistent storage for applications. They are managed independently of pods and can be dynamically provisioned and controlled by the cluster administrator. PVs simplify storage management by decoupling storage from application lifecycles.

##### 392. What is the purpose of a ConfigMap in Kubernetes, and how is it used?

A ConfigMap in Kubernetes is used to decouple configuration data from containerized applications. Its purpose is to store key-value pairs, environment variables, or configuration files that can be injected into pods at runtime. ConfigMaps help in maintaining configuration consistency across different environments and make it easier to update configuration settings without changing the application code.

##### 393. Explain the difference between a StatefulSet and a Deployment in Kubernetes.

|  |  |
| --- | --- |
| **StatefulSet** | **Deployment** |
| Manages stateful applications where each pod has a unique identity and persists data. | Manages stateless applications where each pod is interchangeable and does not maintain state. |
| Pods managed by StatefulSet have stable network identifiers and persistent storage. | Pods managed by Deployment do not have stable identities or persistent storage. |
| Pods are created and terminated in a predictable, ordered manner, suitable for databases or clustered applications. | Pods can be scaled up or down, updated, and rolled back easily. |

##### 394. How does Kubernetes handle 'rolling updates' and rollbacks of application deployments?

Kubernetes handles rolling updates by creating a new version of the application while gradually shifting traffic to it, ensuring minimal downtime. If issues occur, Kubernetes can roll back to the previous version by reverting to the old configuration.

##### 395. What are Labels and Selectors in Kubernetes, and why are they significant?

Labels in Kubernetes are key-value pairs attached to objects like pods or services for identification and categorization. Selectors are used to filter and target objects based on their labels. They are significant because they enable flexible management, grouping of resources, and support various Kubernetes features like service discovery and deployment strategies.

##### 396. Describe the role of a Service Mesh like Istio in a Kubernetes environment.

A Service Mesh like Istio in a Kubernetes environment provides advanced networking, security, and observability features for managing communication between services. It enhances control over traffic routing, enforces security policies, and offers monitoring tools for better management of microservices.

##### 397. How can you monitor and troubleshoot applications running in a Kubernetes cluster?

* Using monitoring tools like Prometheus and Grafana to track metrics.
* Utilizing centralized logging with tools like Elasticsearch and Kibana.
* Implementing distributed tracing with tools like Jaeger.
* Configuring health checks for applications.
* Using debugging tools like kubectl exec and kubectl logs.
* Setting up alerting rules for critical issues.
* Monitoring resource usage and performance.

##### 398. What is the different between node selector and node affinity?

|  |  |
| --- | --- |
| **Node Selector** | **Node Affinity** |
| Basic method for node selection based on labels. | Advanced method for node selection based on label rules and expressions. |
| Directly specified in pod configurations using nodeSelector field. | Allows more complex criteria for pod placement. |
| Matches exact label values. | Specified in pod specifications' affinity field. |
| Limited to straightforward node selection based on predefined labels. | Includes options like required rules and preferred preferences for node selection, offering finer control over scheduling. |

##### 399. What is Terraform?

Terraform is a tool to create and control infrastructure(computer networks, storage, software applications, and other digital services) using simple instructions. It can manage these systems across various cloud providers like AWS, Azure, and GCP, all using just one tool.  
These simple instructions are known as code. If we have code, then we can create infrastructure. So, we have infrastructure in the form of code, meaning Infrastructure as Code (IaC).  
  
Terraform is open-source, written in Golang, and created by HashiCorp, offering features like idempotency and scalability for managing infrastructure deployments efficiently.

##### 400. What is IaC?

IaC stands for "Infrastructure as Code". It's a way to use code to set up and manage computer systems, like servers and networks, instead of doing it manually. By writing code, they can automate these tasks, which makes managing infrastructure easier, faster, and more reliable. It also helps ensure consistency and allows for easier tracking of changes.