DEVOPS Q&A

MANAS KUMAR JHA

Link to Join Our WhatsApp group

DevOps

1. What is Source Code Management?

It is a process through which we can store and manage any code. Developers write code, Testers write test cases and DevOps engineers write scripts. This code, we can store and manage in Source Code Management. Different teams can store code simultaneously. It saves all changes separately. We can retrieve this code at any point of time.

2. What are the Advantages of Source Code Management?

Helps in Achieving teamwork. Can work on different features simultaneously. Acts like pipeline b/w offshore & onshore teams. Track changes (Minute level). Different people from the same team, as well as different teams, can store code simultaneously (Save all changes separately)

2.1. Available Source Code Management tools in the market?

There are so many Source Code Management tools available in the market. Those are

. Git. SVN. Perforce. Clear case

Out of all these tools, Git is the most advanced tool in the market where we are getting so many advantages compared to other Source Code Management tools.

3. What is Git?

Git is one of the Source Code Management tools where we can store any type of code. Git is the most advanced tool in the market now. We also call Git is version control system because every update stored as a new version. At any point of time, we can get any previous version. We can go back to previous versions. Every version will have a unique number. That number we call commit-ID. By using this commit ID, we can track each change i.e. who did what at what time. For every version, it takes incremental backup instead of taking the whole backup. That's why Git occupies less space. Since it is occupying less space, it is very fast.

4. What are the advantages of Git?

. Speed: - Git stores every update in the form of versions. For every version, it takes incremental backup instead of taking the whole backup. Since it is taking less space, Git is very fast. That incremental backup we call "Snapshot"

. Parallel branching: - We can create any number of branches as per our requirement. No need to take prior permission from any one, unlike other Source Code Management tools. Branching is for parallel development. Git branches allow us to work simultaneously on multiple features.

. Fully Distributed: - A backup copy is available in multiple locations in each and everyone's server instead of keeping in one central location, unlike other Source Code Management tools. So even if we lose data from one server, we can recover it easily. That's why we call GIT as DVCS (Distributed Version Control System)

5. What are the stages in Git?

There are total of 4 stages in Git

1. Workspace: - It is the place where we can create files physically and modify. Being a Git user, we work in this work space.

2. Staging area/Indexing area: - In this area, Git takes a snapshot for every version. It is a buffer zone between workspace and local repository. We can't see this region because it is virtual.

3. Local repository: - It is the place where Git stores all commit locally. It is a hidden directory so that no one can delete it accidentally. Every commit will have unique commit ID.
4. Central repository: - It is the place where Git stores all commit centrally. It belongs to everyone who is working in your project. Git Hub is one of the central repositories. Used for storing the code and sharing the code to others in the team.

6. What is the common branching strategy in Git?
• Product is the same, so one repo. But different features.
• Each feature has one separate branch
• Finally, merge (code) all branches
• For Parallel development
• Can create any no of branches
• Can create one branch on the basis of another branch
• Changes are personal to that particular branch
• Can put files only in branches (not in repo directly)
• The default branch is "Master"
• Files created in a workspace will be visible in any of the branch workspaces until you commit. Once you commit, then that file belongs to that particular branch.

7. How many types of repositories available in Git?
There are two types of repositories available in Git
Bare Repositories (Central) These repositories are only for Storing & Sharing the code All central repositories are bare repositories
Non – Bare Repositories (Local) In these repositories, we can modify the files All local /user repositories are Bare Repositories

8. Can you elaborate commit in Git?
• Storing file permanently in the local repository we call commit.
• For every commit, we get one commit ID
• It contains 40 long Alpha-numeric characters
• It uses the concept "Check some" (It's a tool in Linux, generates binary value equal to the data present in file)
• Even if you change one dot, Commit-ID will get changed
• Helps in tracking the changes

9. What do you mean by "Snapshot" in Git?
• It is a backup copy for each version git stores in a repository.
• Snapshot is an incremental backup copy (only backup for new changes)
• Snapshot represents some data of particular time so that, we can get data of particular time by taking that particular snapshot
• This snapshot will be taken in Staging area in Git which is present between Git workspace and Git local repository.

10. What is GitHub?
Git hub is central git repository where we can store code centrally. Git hub belongs to Microsoft Company. We can create any number of repositories in Git hub. All public repositories are free and can be accessible by everyone. Private repositories are not free and can restrict public access for

security. We can copy the repository from one account to other accounts also. This process we call as “Fork”. In this repository also we can create branches. The default branch is “Master”

11. What is Git merge?

By default, we get one branch in git local repository called “Master”. We can create any no of branches for parallel development. We write code for each feature in each branch so that development happens separately. Finally, we merge code off all branches in to Master and push to central repository. We can merge code to any other branch as well. But merging code into master is standard practice that being followed widely. Sometimes, while merging, conflict occurs. When same file is in different branches with different code, when try to merge those branches, conflict occurs. We need to resolve that conflict manually by rearranging the code.

12. What is Git stash?

We create multiple branches to work simultaneously on multiple features. But to work on multiple tasks simultaneously in one branch (i.e. on one feature), we use git stash. Stash is a temporary repository where we can store our content and bring it back whenever we want to continue with our work with that stored content. It removes content inside file from working directory and puts in stashing store and gives clean working directory so that we can start new work freshly. Later on you can bring back that stashed items to working directory and can resume your work on that file. Git stash applicable to modified files. Not new files. Once we finish our work, we can remove all stashed items form stash repository.

13. What is Git Reset?

Git Reset command is used to remove changes form staging area. This is bringing back file form staging area to work directory. We use this command before commit. Often we go with git add accidentally. In this case if we commit, that file will be committed. Once you commit, commit ID will be generated and it will be in the knowledge of everyone. So to avoid this one, we use Git reset. If you add “–hard” flag to git reset command, in one go, file will be removed from staging area as well as working directory. We generally go with this one if we fell that something wrong in the file itself.

15. What is Git Revert?

Git Revert command is used to remove changes from all 3 stages (work directory, staging area and local repository). We use this command after commit. Sometimes, we commit accidentally and later on we realize that we shouldn’t have done that. For this we use Git revert. This operation will generate new commit ID with some meaningful message to ignore previous commit where mistake is there. But, here we can’t completely eliminate the commit where mistake is there. Because Git tracks each and every change.

16. Difference between Git pull and Git clone?

We use these two commands to get changes from central repository. For the first time if you want whole central repository in your local server, we use git clone. It brings entire repository to your local server. Next time onwards you might want only changes instead of whole repository. In this case, we use Git pull. Git clone is to get whole copy of central repository Git pull is to get only new changes from central repository (Incremental data)

17. What is the difference between Git pull and Fetch?

We use Git pull command to get changes from central repository. In this operation, internally two commands will get executed. One is Git fetch and another one is Git merge. Git fetch means, only bringing changes from central repo to local repo. But these changes will not be integrated to local repo which is there in your server. Git merge means, merging changes to your local repository which is there in your server. Then only you can see these changes. So Git pull is the combination of Git pull and Git merge.

18. What is the difference between Git merge and rebase?

We often use these commands to merge code in multiple branches. Both are almost same but few differences. When you run Git merge, one new merge commit will be generated which is having the history of both development branches. It preserves the history of both branches. By seeing this merge commit, everyone will come to know that we merged two branches. If you do Git rebase, commits in new branch will be applied on top of base branch tip. There won't be any merge commit here. It appears that you started working in one single branch form the beginning. This operation will not preserve the history of new branch.

19. What is Git Bisect?

Git Bisect we use to pick bad commit out of all good commits. Often developers do some mistakes. For them it is very difficult to pick that commit where mistake is there. They go with building all commits one by one to pick bad commit. But Git bisect made their lives easy. Git bisect divides all commits equally in to two parts (bisecting equally). Now instead of building each commit, they go with building both parts. Where ever bad commit is there, that part build will be failed. We do operation many times till we get bad commit. So Git bisect allows you to find a bad commit out of good commits. You don't have to trace down the bad commit by hand; git-bisect will do that for you.

20. What is Git squash?

To move multiple commits into its parent so that you end up with one commit. If you repeat this process multiple times, you can reduce "n" number of commits to a single one. Finally we will end up with only one parent commit. We use this operation just to reduce number of commits.

21. What is Git hooks?

We often call this as web hooks as well. By default we get some configuration files when you install git. These files we use to set some permissions and notification purpose. We have different types of hooks (pre commit hooks & post commit hooks)

Pre-commit hooks:- Sometimes you would want every member in your team to follow certain pattern while giving commit message. Then only it should allow them to commit. These type of restrictions we call pre-commit hooks.

Post-commit hooks:- Sometimes, being a manager you would want an email notification regarding every commit occurs in a central repository. This kind of things we call post-commit hooks.

In simple terms, hooks are nothing but scripts to put some restrictions.

22. What is Git cherry-pick?

When you go with git merge, all commits which are there in new development branch will be merged into current branch where you are. But sometimes, requirement will be in such that you would want to get only one commit form development branch instead of merging all commits. In this case we go with git cherry-pick. Git cherry-pick will pick only one commit whatever you select and merges with

commits which are there in your current branch. So picking particular commit and merging into your current branch we call git cherry-pick.

23. What is the difference between Git and SVN?

SVN:- It is centralized version control system (CVCS) where back up copy will be placed in only one central repository. There is no branching strategy in SVN. You can't create branches. So no parallel development. There is no local repository. So can't save anything locally. Every time after writing code you need to push that code to central repository immediately to save changes.

Git:- It is a Distributed version control system where back up copy is available in everyone's machine's local repository as well as a central repository. We can create any no of branches as we want. So we can go in parallel development simultaneously. Every Git repository will have its own local repository. So we can save changes locally. At the end of our work finally, we can push code to a central repository.

24. What is the commit message in Git?

Every time we commit, while committing, we have to give commit message just to identify each commit. We can't remember to commit numbers because they contain 40 long alphanumeric characters. So, to remember commits easily, we give commit message. The format of commit message differs from company to company and individual to individual. We have one more way to identify commits. That is giving "Tags". Tag is a kind of meaningful name to a particular commit. Instead of referring to commit ID, we can refer to tags. Internally tag will refer to respective commit ID. These are the ways to get a particular commit easily.

26. What is Configuration Management?

It is a method through we automate admin tasks. Each and every minute details of a system, we call configuration details. If we do any change here means we are changing the configuration of a machine. That means we are managing the configuration of the machine. System administrators used to manage the configuration of machine through manually. DevOps engineers are managing this configuration through automated way by using some tools which are available in the market. That's why we call these tools as configuration management tools.

27. What is IAC?

IAC means Infrastructure As Code. It is the process through which we automate all admin tasks. Here we write code in Ruby script in chef. When you apply this code, automatically code will be converted into Infrastructure. So here we are getting so many advantages in writing the code. Those are 1. Code is Testable (Testing code is easy compare to Infrastructure) 2. Code is Repeatable (Can re-use the same code again and again)

3. Code is Versionable (Can store in versions so that can get any previous versions at any time)

28. What do you mean by IT Infrastructure??

IT Infrastructure is a composite of the following things

- Software
- Network
- People
- Process

29. What are the problems that system admins used to face earlier when there were no configuration management tools?

1. Managing users & Groups is big hectic thing (create users and groups, delete, edit……) 2. Dealing with packages (Installing, Upgrading & Uninstalling) 3. Taking backups on regular basis manually 4. Deploying all kinds of applications in servers 5. Configure services (Starting, stopping and restarting services) These are some problems that system administrators used to face earlier in their manual process of managing configuration of any machine.

30. Why should we go with Configuration Management Tool?

1. By using the Configuration Management Tool, we can automate almost each and every admin task. 2. We can increase uptime so that can provide maximum user satisfaction. 3. Improve the performance of systems. 4. Ensure compliance 5. Prevent errors as tools won't do any errors 6. Reduce cost (Buy tool once and use 24/7)

31. How this Configuration Management Tool works?

Whatever system admins (Linux/windows) used to do manually, now we are automating all those tasks by using any Configuration Management Tool. We can use this tool whether your servers are in on-premises or in the cloud. It turns your code into infrastructure. So your code is versionable, repeatable and testable. You only need to tell what the desired configuration should be, not how to achieve it. Through automation, we get our desired state of server. This is unique feature of Configuration Management Tool.

32. What is the architecture of Chef?

Chef is an administration tool. In this we have total 3 stages. 1. Chef Workstation (It is the place where we write code) 2. Chef Server (It is the place where we store code) 3. Chef Node (It is the place where we apply code) We need to establish communication among workstation, server and nodes. You can have any no of nodes. There is no limit. Chef can manage any no of nodes effectively.

33. Components of Chef?

Chef Workstation: Where you write the code
Chef Server: Where you upload the code
Chef Node: Where you apply the code
Knife: Tool to establish communication among workstation, server & node.
Chef-client: Tool runs on every chef node to pull code from chef server
Ohai: Maintains current state information of chef node (System Discovery Tool)
Idempotency: Tracking the state of system resources to ensure that the changes should not re-apply repeatedly.
Chef Supermarket: Where you get custom code

34. How does Chef Works?

We need to install chef package in workstation, server and nodes. We create cookbook in workstation. Inside cookbook, there will be a default recipe where you write code in ruby script. You can create any no of recipes. There is no limit. After writing code in recipe, we upload whole cookbook to chef server. Chef server acts as central hub storing code. Then, we need to add this cookbook's recipe to nodes run-list. Chef-client tool will be there in each and every chef node. It runs frequently. Chef-client comes to chef server and take that code and applies that code in node. This is how code will be converted into infrastructure.

35. What is Idempotency?
It is unique feature in all configuration management tools. It ensures that changes should not re-apply repeatedly. Once chef-client converted code into Infrastructure, then even chef-client runs again, it will not take any action. It won't do the same task again and again. If any new changes are there in that code, then only chef-client is going to take action. So it doesn't make any difference ever if you run chef-client any no of times. So tracking the system details to not to reapply changes again and again, we call Idempotency.

36. What is Ohai and how does it works??
Ohai we call "System Discovery Tool". It stores system information. It captures each and every minute details of system and updates it then and there if any new changes are there. Whenever chef-client converts code in infrastructure in node, immediately Ohai store will be updated. Next time onwards, before chef-client runs, it verifies in Ohai store to know about current state of information. So chef-client will come to know the current state of server. Then chef-client acts accordingly. If new changes are there, then only it will take action. If there are no new changes, then it won't take any action. Ohai tool helps in achieving this.

37. How many types of chef server?
Total there are 3 ways through which we can manage chef server. 1. Directly we can take chef server from Chef Company itself. In this case, everything will be managed by Chef Company. You will get support from chef. This type of server we call Managed/Hosted chef. This is completely Graphical User Interface (GUI). It's not free. We need to pay to Chef Company after exceeding free tier limit. 2. We can launch one server and we need to install chef server package. It is completely free package. It's GUI. 3. We can launch one server and we need to install chef server package. It is completely free package. It's CLI (Command Line Interface).

38. What is there inside cookbook??
Below mentioned files and folders will be there inside cookbook when you first create it
Chefignore: like .gitignore (to ignore files and folders)
Kitchen.yml: for testing of cookbook
Metadata.rb: name, author, version…. etc of cookbook Readme.md: information about usage of cookbook Recipe: It is a file where you write code **Spec:** for unit test **Test:** for integration test

39. What is Attributes concept in chef?
Sometimes we might need to deploy web applications to in nodes and for that we need to know some host specific details of each server like IP Address, Host name ….. etc. Because we need to mention that in configuration files of each server. These files we call as Configuration files. This information will be vary from system to system. These host specific details that we mention in Configuration files,we call "Attributes". Chef-client tool gathers these Attributes from Ohai store and puts in configuration files. Instead of hard coding these attributes, we mention as variables so that every time, file will be updated with latest details of their respective nodes.

40. What is Run-list in Chef?
This is an ordered list of recipes that we are going to apply to nodes. We mention all recipes in cookbook and then we upload that cookbook to chef server. Then, we attach all recipes to nodes run-list in sequence order. When chef-client runs, it applies all recipes to nodes in the same order

whatever the order you mention in run-list. Because sometimes order is important especially when we deal with dependent recipes.

41. What is bootstrap?

It is the process of adding chef node to chef server or we can call, bringing any machine into chef environment. In this bootstrapping process total three action will be performed automatically. 1. Node gets connected to chef server. 2. Chef server will install chef package in chef node. 3. Cookbooks will be applied to chef node.

It is only one time effort. As and when we purchase any new machine in company, immediately we add that server to chef server. At a time, we can bootstrap one machine. We can't bootstrap multiple machines at a time.

42. What is the workflow of Chef?

We connect chef workstation, chef server and chef node with each other. After that, we create cookbook in chef workstation and inside that cookbook, we write code in recipe w.r.t. the infrastructure to be created. Then we upload entire cookbook to chef server and attach that cookbook's recipe to nodes run-list. Now we automate chef-client which will be there in all chef nodes. Chef-client runs frequently towards chef server for new code. So chef-client will get that code from server and finally applies to chef node. This is how, code is converted into infrastructure. If no changes are there in code, even if chef-client runs any no of time, it won't take any action until it finds some changes in code. This is what we call Idempotency.

43. How does we connect Chef Workstation to Chef Server?

First we download started kit from chef server. This will be downloaded in the form of zip file. If we extract this zip file, we will get chef-repo folder. This chef-repo folder we need to place in chef workstation. Inside chef-repo folder, we can see total three folders. They are .chef, cookbooks and roles. Out of these three, .chef folder is responsible to establish communication between chef server and chef workstation. Because, inside .chef folder, we can see two files. They are knife.rb and organization.pem. Inside kinfe.rb, there will be the url (address) of chef server. Because of this url, communication will be established between chef server and chef workstation. This is how we connect Chef Workstation to Chef Server.

44. How does the chef-client runs automatically?

By default, chef-client runs manually. So we need to automate this manually. For this, we use "cron tool" which is the default tool in all Linux machines use to schedule tasks to be executed automatically at frequent intervals. So in this "crontab" file, we give chef-client command and we need to set the timing as per our requirement. Then onwards chef-client runs automatically after every frequent intervals. It is only one time effort. When we purchase any new server in company, along with bootstrap, we automate chef-client then and there.

45. What is chef supermarket?

Chef supermarket is the place where we get custom cookbooks. Every time we need not to create cookbooks and need not to write code from scratch. We can go with custom cookbooks which are available in chef supermarket being provided by chef organization and community. We can download these cookbooks and modify as per our needs. We get almost each and every cookbook from chef supermarket. They are safe to use.

46. What is wrapper cookbook?

Either we can download those chef supermarket cookbooks or without downloading, we can call these supermarket cookbooks during run time so that every time we get updates automatically for that cookbook if any new updates are there. Here, we use our own cookbook to call chef supermarket cookbook. This process of calling cookbook by using another cookbook, we call wrapper cookbook. Especially, we use this concept to automate chef-client.

47. What is "roles" in chef?

Roles are nothing but a Custom run-list. We create role & upload to chef server & assign them to nodes. If we have so many nodes, need to add cookbook to run-list of all those nodes, it is very difficult to attach to all nodes run-list. So, we create role & attach that role to all those nodes once. Next time onwards, add cookbook to that role. Automatically, that cookbook will be attached to all those nodes. So role is one time effort. Instead of adding cookbooks to each & every node's run-list always, just create a role & attach that role to nodes. When we add cookbook to that role, it will be automatically applied to all nodes those assigned with that role.

48. What is include_recipe in chef?

By default, we can call one recipe at a time in one cookbook. But if you want to call multiple recipes from same cookbook, we use include_recipe concept. Here, we take default recipe and we mention all recipes to be called in this default recipe in an order. If we call default recipe, automatically default recipe will call all other recipes which are there inside default recipe. By using one recipe, we can call any no of recipes. This process of calling one recipe by using other recipe, we call as include_recipe. Here condition is we can call recipes from same cookbook, but not from different cookbooks.

49. How to deploy a web server by using chef?

package 'httpd' do action :install end
file '/var/www/html/index.html' do content 'Hello Dear Students!!' action :create end
service 'httpd' do action [:enable, :start] end

50. How to write ruby code to create file, directory?

file '/myfile' do content 'This is my second file' action :create owner 'root' group 'root' end
directory '/mydir' do action :create owner 'root' group 'root' end

51. How to write ruby code to create user, group and install package?

user 'user1' do action: create end
group 'group1' do action :create members 'user1' append true end
package 'httpd' do action: install end

52. What is container?

The container is like a virtual machine in which we can deploy any type of applications, soft wares and libraries. It's a light weight virtual machine which uses OS in the form of image, which is having less in size compare to traditional VMware and oracle virtual box OS images. Container word has been taken from shipping containers. It has everything to run an application.

53. What is virtualization?

Logically dividing big machine into multiple virtual machines so that each virtual machine acts as new server and we can deploy any kind of applications in it. For this first we install any virtualization software on top of base OS. This virtualization software will divide base machine resources in to logical components. In a simple terms, logically dividing one machine into multiple machines we call virtualization.

54. What is Docker?

Docker is a tool by using which, we create containers in less time. Docker uses light weight OS in the form of docker images that we will get from docker hub. Docker is open source now. It became so popular because of its unique virtualization concept called "Containerization" which is not there in other tools. We can use docker in both windows and Linux machines.

55. What do you mean by docker image?

Docker image is light weight OS provided by docker company. We can get any type of docker image form docker hub. We use these docker images to create docker containers. This docker images may contain only OS or OS + other soft wares as well. Each software in docker image, will be stored in the form of layer. Advantage of using docker images is, we can replicate the same environment any no of times.

56. What are the ways through which we can create docker images?

There are three ways through which we can create docker images. 1. We can take any type of docker image directly from docker hub being provided by docker company and docker community. 2. We can create our own docker images form our own docker containers. I.e. first we create container form base docker image taken form docker hub and then by going inside container, we install all required soft wares and then create docker image from our own docker container. 3. We can create docker image form docker file. It is the most preferred way of creating docker images.

57. What is docker file and why do we use it?

It is a just normal text file with instructions in it to build docker image. It is the automated way of creating docker images. Once you build docker image, automatically docker file will be created. In this file, we mention required OS image and all required soft wares in the form of instructions. Once we build docker file, back end, docker container will be created and then docker image will be crated from that container and that container will be destroyed automatically.

58. Difference between docker and VM Ware?

VM Ware uses complete OS which contains GBs in size. But docker image size is MBs only. So it takes less size. That's why it takes less base machine resources. This docker image is compressed version of OS. The second advantage of docker is, there is no pre-allocation of RAM. During run time, it takes RAM as pre-requirement from base machine and one's job is done, it releases RAM. But in VM Ware, pre-allocation of RAM is there and it blocked whether it uses or not. So, need more RAM for base machine if you want to use VM Ware unlike Docker.

59. What is OS-Lever Virtualization?

It is the unique feature of Docker which is not available in other virtualization soft wares. Docker takes most of UNIX features form host machine OS and it only takes extra layers of required OS in the form of docker image. So docker image contains only extra layers of required OS. For core UNIX kernel, it depends upon host OS, why because UNIX kernel is same in any of the UNIX and Linux flavours. In a simple term, docker takes host OS virtually. That's why we call this concept as OS-Lever Virtualization.

60. What is Layered file system/Union file system?

Inside docker container, wheat ever we do, that forms as a new layer. For instance, creating files, directories, installing packages etc. This is what we call as layered file system. Each layer takes less space. We can create docker image form this container. In that docker image also we get all these layers and forms unity. That's why we also call Union File System. If we create container out of docker image, you can able to see all those files, directories and packages. This is what replication of same environment.

61. What are the benefits of Docker?

- Containerization (OS level virtualization) (No need guest OS)
- No pre-allocation of RAM
- Can replicate same environment
- Less cost
- Less weight (MB's in size)
- Fast to fire up
- Can run on physical/virtual/cloud
- Can re-use (same image)
- Can create containers in less time

62. List of Docker components?

Docker image: – Contains OS (very small) (almost negligible) + soft wares

Docker Container: – Container like a machine which is created from Docker image.

Docker file: – Describes steps to create a docker image.

Docker hub/registry: – Stores all docker images publicly.

Docker daemon: – Docker service runs at back end Above five components we call as Docker components

63. What is Docker workflow?

First we create Docker file by mentioning instructions to build docker image. Form this Docker image, we are going to create Docker container. This Docker image we can push to docker hub as well. This image can be pulled by others to create docker containers. We can create docker images from docker containers. Like this we can create Docker images form either docker file or docker containers. We can create docker containers from docker images. This is the work flow of docker.

64. Sample Docker file instructions?
FROM ubuntu WORKDIR /tmp RUN echo “Hello” > /tmp/testfile ENV myname user1 COPY testfile1 /tmp ADD test.tar.gz /tmp

65. What is the importance of volumes in Docker?
• Volume is a directory inside your container
• First declare directory as a volume and then share volume
• Even if we stop container, still we can access volume
• Volume will be created in one container
• You can share one volume across any no of containers
• Volume will not be included when you update an image
• Map volumes in two ways
• Share host – container
• Share container – container

66. What do you mean by port mapping in Docker?
Suppose if you want to make any container as web server by installing web package in it, you need to provide containers IP address to public in order to access website which is running inside docker container. But Docker containers don’t have an IP address. So, to address this issue, we have a concept called
Docker port mapping. We map host port with container port and customers use public IP of host machine. Then their request will be routed from host port to container’s port and will be loaded web page which is running inside docker container. This is how we can access website which is running inside container through port mapping.

67. What is Registry server in Docker?
Registry server is our own docker hub created to store private docker images instead of storing in public Docker hub. Registry server is one of the docker containers. We create this Registry server from “registry” image, especially provided by docker to create private docker hub. We can store any no of private docker images in this Registry server. We can give access to others, so that, they also can store their docker images whomever you provide access. Whenever we want, we can pull these images and can create containers out of these images.

68. Important docker commands?
1. Docker ps (to see list of running containers)
2. Docker ps -a (to see list of all containers)
3. Docker images (to see list of all images)
4. Docker run (to create docker container)
5. Docker attach (to go inside container)
6. Docker stop (to stop container)
7. Docker start (to start container)
8. Docker commit (to create image out of docker file)
9. Docker rm (to delete container)
10. Docker rmi (to delete image)

69. What is Ansible?

Ansible is one of the configuration Management Tools. It is a method through we automate system admin tasks. Configuration refers to each and every minute details of a system. If we do any changes in system means we are changing the configuration of a machine. That means we are changing the configuration of the machine. All windows/Linux system administrators manage the configuration of a machine manually. All DevOps engineers are managing this configuration automatic way by using some tools which are available in the market. One such tool is Ansible. That's why we call Ansible as configuration management tool.

70. Working process of Ansible?

Here we crate file called playbook and inside playbook we write script in YAML format to create infrastructure. Once we execute this playbook, automatically code will be converted into Infrastructure. We call this process as IAC (Infrastructure as Code). We have open source and enterprise editions of Ansible. Enterprise edition we call Ansible Tower.

71. The architecture of Ansible?

We create Ansible server by installing Ansible package in it. Python is pre-requisite to install ansible. We need not to install ansible package in nodes. Because, communication establishes from server to node through "ssh" client. By default all Linux machine will have "ssh" client. Server is going to push the code to nodes that we write in playbooks. So Ansible follows pushing mechanism. 72. Ansible components?

Server: – It is the place where we create playbooks and write code in YML format
Node: – It is the place where we apply code to create infrastructure. Server pushes code to nodes.
Ssh: – It is an agent through ansible server pushes code to nodes.
Setup: – It is a module in ansible which gathers nodes information.
Inventory file:- In this file we keep IP/DNS of nodes.

73. Disadvantages in other SCM (Source Code Management) tools?

- Huge overhead of Infrastructure setup
- Complicated setup
- Pull mechanism
- Lot of learning required

74. Advantages of Ansible over other SCM (Source Code Management) tools?

- Agentless
- Relies on "ssh"
- Uses python
- Push mechanism

75. How does Ansible work?

We give nodes IP addresses in hosts file by creating any group in ansible server why because, ansible doesn't recognize individual IP addresses of nodes. We create playbook and write code in YAML script. The group name we have to mention in a playbook and then we execute the playbook. By default, playbook will be executed in all those nodes which are under this group. This is how ansible converts code into infrastructure.

76. What do you mean by Ad-Hoc commands in Ansible?
These are simple one liner Linux commands we use to meet temporary requirements without actually saving for later. Here we don't use ansible modules. So there, Idempotency will not work with Ad-Hoc commands. If at all we don't get required YAML module to write to create infrastructure, then we go for it. Without using playbooks, we can use these Ad-Hoc commands for temporary purpose.

77. Differences between Chef and Ansible?
- Ansible chef
- Playbook – Recipe
- Module – Resource
- Host – Node
- Setup – Ohai
- Ssh – Knife
- Push-Pull

78. What is Playbook in Ansible?
Playbook is a file where we write YAML script to create infrastructure in nodes. Here, we use modules to create infrastructure. We create so many sections in playbook. We mention all modules in task section. You can create any no of playbooks. There is no limit. Each playbook defines one scenario. All sections begin with "-" & its attributes & parameters beneath it.

79. Mention some list of sections that we mention in Playbook?
1. Target section 2. Task section 3. Variable section 4. Handler section

80. What is Target section in Ansible playbook?
This is one of the important sections in Playbook. In this section, we mention the group name which contains either IP addresses or Hostnames of nodes. When we execute playbook, then code will be pushed too all nodes which are there in the group that we mention in Target section. We use "all" key word to refer all groups.

81. What is Task section in Ansible playbook?
This is second most important section in playbook after target section. In this section, we are going to mention list of all modules. All tasks we mention in this task section. We can mention any no of modules in one playbook. There is no limit. If there is only one task, then instead of going with big playbook, simply we can go with arbitrary command where we can use one module at a time. If more than one module, then there is no option except going with big playbook.

82. What is Variable section?
In this section we are going to mention variables. Instead of hard coding, we can mention as variables so that during runtime it pulls the actual value in place of key. We have this concept in each and every programming language and scripting language. We use "vars" key word to use variables.

83. What is Handler section?
All tasks we mention in tasks section. But some tasks where dependency is there, we should not mention in tasks section. That is not good practice. For example, installing package is one task and starting service is one more task. But there is dependency between them. I.e. after installing package only, we have to start service. Otherwise it throws error. These kind of tasks, we mention in handler

section. In above example, package task we mention in task section and service task we mention in handler section so that after installing task only service will be started.

84. What is Dry run in playbook?
Dry run is to test playbook. Before executing playbook in nodes, we can test whether the code in playbook is written properly or not. Dry run won't actually executes playbook, but it shows output as if it executed playbook. Then by seeing the output, we can come to know whether the playbook is written properly or not. It checks whether the playbook is formatted correctly or not. It tests how the playbook is going to behave without running the tasks.

85. Why are we using loops concept in Ansible?
Sometimes we might need to deal with multiple tasks. For instance, Installing multiple packages, Creating many users, creation many groups. Etc. In this case, mentioning module for every task is complex process. So, to address this issue, we have a concept of loops. We have to use variables in combination with loops.

86. Where do we use conditionals in Playbooks?
Sometimes, your nodes could be mixture of different flavours of Linux OS. Linux commands vary in different Linux operating systems. In this case, we can't execute common set of commands in all machines, at the same time, we can't execute different commands in each node separately. To address this issue, we have conditionals concept where commands will be executed based up on certain condition that we give.

87. What is Ansible vault?
Sometimes, we use sensitive information in playbooks like passwords, keys …etc. So any one can open these playbooks and get to know about this sensitive information. So we have to protect our playbooks from being read by others. So by using Ansible vault, we encrypt playbooks so that, those who ever is having password, only those can read this information. It is the way of protecting playbooks by encrypting them.

88. What do you mean by Roles in Ansible?
Adding more & more functionality to the playbooks will make it difficult to maintain in a single file. To address this issue, we organize playbooks into a directory structure called "roles". We create separate file to each section and we just mention the names of those sections in playbook instead of mentioning all modules in main playbook. When you call main playbook, main playbook will call all sections files respectively in the order whatever order you mention in playbook. So, by using this Roles, we can maintain small playbook without any complexity.

89. Write a sample playbook to install any package?
— # My First YAML playbook – hosts: demo user: ansible become: yes connection: ssh tasks: – name: Install HTTPD on centos 7 action: yum name=httpd state=installed

90. Write a sample playbook by mentioning variables instead of hard coding?
— # My First YAML playbook – hosts: demo user: ansible become: yes connection: ssh vars: pkgname: httpd tasks: – name: Install HTTPD server on centos 7 action: yum name='{{pkgname}}' state=installed

91. What is CI & CD?

CI means Continues Integration and CD means Continues Delivery/Deploy. Whenever developers write code, we integrate all that code of all developers at that point of time and we build, test and deliver/deploy to the client. This process we call CI & CD. Jenkins helps in achieving this. So instead of doing night builds, build as and when commit occurs by integrating all code in SCM tool, build, test and checking the quality of that code is what we call Continues Integration.

92. Key terminology that we use in Jenkins?

Integrate: Combine all code written by developers till some point of time.
Build: Compile the code and make a small executable package.
Test: Test in all environments whether application is working properly or not.
Archived: Stored in an antifactory so that in future we may use/deliver again.
Deliver: Handing the product to Client
Deploy: Installing product in client's machines.

93. What is Jenkins Workflow?

We attach Git, Maven, Selenium & Artifactory plug-ins to Jenkins. Once Developers put the code in Git, Jenkins pulls that code and send to Maven for build. Once build is done, Jenkins pulls that built code and send to selenium for testing. Once testing is done, then Jenkins will pull that code and send to Artifactory as per requirement and finally we can deliver the end product to client we call Continues delivery. We can also deploy with Jenkins into clients machine directly as per the requirement. This is what Jenkins work flow.

94. What are the ways through which we can do Continues Integration?

are total three ways through which we can do Continues Integration
1. **Manually: –** Manually write code, then do build manually and then test manually by writing test cases and deploy manually into clients machine.
2. **Scripts: –** Can do above process by writing scripts so that these scripts do CI&CD automatically. But here complexity is, writing script is not so easy.
3. **Tool: –** Using tools like Jenkins is very handy. Everything is preconfigured in these type of tools. So less manual intervention. This is the most preferred way.

95. Benefits of CI?

1. Detects bugs as soon as possible, so that bug will be rectified fast and development happens fast. 2. Complete automation. No need manual intervention. 3. We can intervene manually whenever we want. I.e. we can stop any stage at any point of time so have better control. 4. Can establish complete and continues work flow.

96. Why only Jenkins?

- It has so many plug-ins.
- You can write your own plug-in
- You can use community plug-ins
- Jenkins is not just a tool. It is a framework. I.e. you can do what ever you want. All you need is plug-ins.
- We can attach slaves (nodes) to Jenkins master. It instructs others (slaves) to do Job. If slaves are not available,
- Jenkins itself does the job.

• Jenkins also acts as crone server replacement. I.e. can do repeated tasks automatically
• Running some scripts regularly
• E.g.: Automatic daily alarm.
• Can create Labels (Group of slaves) (Can restrict where the project has to run)

97. What is Jenkins Architecture?
Jenkins architecture is Client-Server model. Where ever, we install Jenkins, we call that server is **Jenkins master.** We can create slaves also in Jenkins, so that, server load will be distributed to slaves. Jenkins master randomly assigns tasks to slaves. But if you want to restrict any job to run in particular slave, then we can do it so that, that particular job will be executed in that slave only. We can group some slaves by using "Label"

98. How to install Jenkins?
• You can install Jenkins in any OS. All OSs supports Jenkins. We access Jenkins through web page only. That's why it doesn't make any difference whether you install Jenkins in Windows or Linux.
• Choose Long Term Support release version, so that you will get support from Jenkins community. If you are using Jenkins for testing purpose, you can choose weekly release. But for production environments, we prefer Long Term Support release version.
• Need to install JAVA. Java is pre-requisite to install Jenkins.
• Need to install web package. Because, we are going to access Jenkins through web page only.

99. Does Jenkins open source?
There are two editions in Jenkins 1. Open source 2. Enterprise edition Open source edition we call Jenkins. Here we get support from community if we need it. Enterprise edition we call Hudson. Here Jenkins company will provide support.

100. How many types of configurations in Jenkins?
There are total 3 types of configurations in Jenkins.
1. **Global: –** Here, whatever configuration changes we do, applicable to whole Jenkins including jobs as well as nodes. This configuration has high priority.
2. **Job: –** These configurations applicable to only Jobs. Jobs also we call as projects or items in Jenkins.
3. **Node: –** These configurations applicable to only nodes. Also we call Slaves. These are kind of helpers to Jenkins master to distribute the excessive load.

101. What do you mean by workspace in Jenkins?
The workspace is the location on your computer where Jenkins places all files related to the Jenkins project. By default each project or job is assigned a workspace location and it contains Jenkins-specific project metadata, temporary files like logs and any build artefacts, including transient build files. Jenkins web page acts like a window through which we are actually doing work in workspace.

102. List of Jenkins services?
• localhost:8080/restart (to restart Jenkins)
• localhost:8080/stop (to stop Jenkins)
• localhost:8080/start (to start Jenkins)

103. How to create a free style project in Jenkins?
• Create project by giving any name
• Select Free style project
• Click on build
• Select execute windows batch command
• Give any command (echo “Hello Dear Students!!”)
• Select Save
• Click on Build now
• Finally can see Console output

104. What do you mean by Plugins in Jenkins?
• With Jenkins, nearly everything is a plugin and that nearly all functionality is provided by plugins. You can think of Jenkins as little more than an executor of plugins.
• Plugins are small libraries that add new abilities to Jenkins and can provide integration points to other tools.
• Since nearly everything Jenkins does is because of a plugin, Jenkins ships with a small set of default plugins, some of which can be upgraded independently of Jenkins

105. How to create Maven Project?
• Select new item
• Copy the git hub maven project link and paste in git section in Jenkins
• Select build
• Click on clean package
• Select save
• Click on Build now
• Verify workspace contents with GitHub side See console output

106. How can we Schedule projects?
Sometimes, we might need some jobs to be executed after frequent intervals. To schedule a job,
• Click on any project
• Click on Configure
• Select on Build triggers
• Click on Build periodically
• Give timing (* * * * *)
• Select Save
• Can see automatic builds every 1 min
• You can manually trigger build as well if you want

107. What do you mean by Upstream and Downstream projects?
We can also call them as linked projects. These are the ways through which, we connect jobs one with other. In Upstream jobs, first job will trigger second job after build is over. In Downstream jobs, second job will wait till first job finishes its build. As and when first job finishes its work, then second job will be triggered automatically. In Upstream, first job will be active. In Downstream jobs, second job will be active. We can use any one type to link multiple jobs.

108. What is view in Jenkins?

We can customize view as per our needs. We can modify Jenkins home page. We can segregate jobs as per the type of jobs like free style jobs and maven jobs and so on. To create custom view

- Select List of Related Projects
- Select Default views
- Click on All
- Click on + and select Freestyle
- Select List Views
- Select Job filter
- Select required jobs to be segregated
- Now, you can see different view

109. What is User Administration in Jenkins?

In Jenkins, we can create users, groups and can assign limited privileges to them so that, we can have better control on Jenkins. Users will not install Jenkins in their machines. They access Jenkins as a user. Here we can't assign permissions directly to users. Instead we create "Roles" and assign permissions to those roles. These roles we attach to users so that users get the permissions whatever we assign to those roles.

110. What is Global tool configuration in Jenkins?

We install Java, Maven, Git and many other tools in our server. Whenever Jenkins need those tools, by default Jenkins will install them automatically every time. But it's not a good practice. That's why we give installed path of all these tools in Jenkins so that whenever Jenkins need them, automatically Jenkins pull them form local machine instead of downloading every time. This way of giving path of these tools in Jenkins we call "Global tool configuration"

111. What is Build?

Build means, Compile the source code, assembling of all class files and finally creating deliverable
Compile: – Convert Source code into machine-readable format
Assembly (Linking): – Grouping all class files
Deliverable: – .war, .jar The above process is same for any type of code. This process we call Build.

112. What is Maven?

Maven is one of the Build tools. It is the most advance build tool in the market. In this, everything is already pre-configured. Maven belongs to Apache Company. We use maven to build Java code only. We can't build other codes by using Maven. By default, we get so many plugins with Maven. You can write your own plug-in as well. Maven's local repository is ".M2" where we can get required compilers and dependencies. Maven's main configuration file is "pom.xml" where we keep all instructions to build.

113. Advantages of Maven?

- Automated tasks (Mention all in pom.xml)
- Multiple Tasks at a time
- Quality product
- Minimize bad builds
- Keep history
- Save time – Save money

• Gives set of standards
• Gives define project life cycle (Goals)
• Manage all dependencies
• Uniformity in all projects
• Re-usability

114. List of Build tools available in Market?
• C and C++: Make file
• .Net: Visual studio
• Java: Ant, Maven

115. What is the architecture of Maven?
Main configuration file is pom.xml. For one project, there will be one workspace and one pom.xml Requirements for build: –
• Source code (Will be pulled from Git hub)
• Compiler (Pulls from remote repo and then put them in local repo, from there, maven pulls into Workspace)
• Dependencies (Pulls from remote repo and then put them in local repo, from there, maven pulls into Workspace)

116. What is Maven's Build Life Cycle?
In maven, we have different goals. These are
• Generate resources (Dependencies)
• Compile code
• Unit test
• Package (Build)
• Install (in to local repo & artifactory)
• Deploy (to servers)
• Clean (delete all run time files)

117. What does POM.XML contains?
POM.XML is maven's main configuration file where we keep all details related to project. It contains
• Metadata about that project
• Dependencies required to build the project
• The kind of project
• Kind of output you want (.jar, .war)
• Description about that project

118. What is Multi-Module Project in Maven?
• Dividing big project into small modules, we call Multi Module Project.
• Each module must have its own SRC folder & pom.xml so that build will happen separately
• To build all modules with one command, there should be a parent pom.xml file. This calls all child pom.xml files automatically
• In parent pom.xml file, need to mention the child pom.xml files in an order.

119. What is Nagios?

Nagios is one of the monitoring tools. By using Nagios we can monitor and give alerts. Where ever you install Nagios that becomes Nagios server. Monitoring is important, because we need to make sure that our servers should never go down. If at all in some exceptional cases server goes down, immediately we need alert in the form of intimation so that we can take required action to bring the server up immediately. So for this purpose, we use Nagios.

120. Why do we have to use Nagios?

There are many advantages in using Nagios

- It is oldest & Latest (every now and then, it is getting upgraded as per current market requirements)
- Stable (we have been using this since so many years and it is performing well)
- By default, we get so many Plug-ins
- It is having its own Database
- Nagios is both Monitoring & Alerting tool.

121. How does Nagios works?

- We mention all details in configuration files what data to be collected from which machine
- Nagios daemon reads those details about what data to be collected
- Daemon use NRPE (Nagios Remote Plug-in Executer) plug-in to collect data form nodes and stores in its own database
- Finally displays in Nagios dashboard

122. What is the Directory structure of Nagios?

/usr/local/nagios/bin – binary files /usr/local/nagios/sbin – CGI files (to get web page) /usr/local/nagios/libexec – plugins /usr/local/nagios/share – PHP Files /usr/local/nagios/etc – configuration files /usr/local/nagios/var – logs /usr/local/nagios/var/status.dat(file) – database

123. What are the Important Configuration files in Nagios?

Nagios main configuration file is /usr/local/nagios/etc/nagios.cfg
/usr/local/nagios/etc/objects/localhost.cfg (where we keep hosts information)
/usr/local/nagios/etc/objects/contacts.cfg (whom to be informed (emails))
/usr/local/nagios/etc/objects/timeperiods.cfg (at what time to monitor)
/usr/local/nagios/etc/objects/commands.cfg (plugins to use)
/usr/local/nagios/etc/objects/templates.cfg (sample templates)