AKIA4MTWHTPYXZYRRZHI

4DEEy3xxJUL6HkvsvuefOdD+EAlWr2ifOSeeQcvk

Day 1

Devops:

Github link- [iam-veeramalla (Abhishek Veeramalla) · GitHub](https://github.com/iam-veeramalla)

Basic questions asked in interviews:

*What is devops?*

*Why devops?*

*How to introduce yourself?*

*What are your day to day activities?*

**What is devops?**

Devops is a practice or cultural that improves the organizational to deliver the products/applications effectively. In other word, devops is the process of increasing the ability to deliver our applications. In conclusion, **Devops is the process of improving the application/product/script in fast delivery, by ensuring proper automation, maintaining proper quality, continuous monitoring and continuous testing. At end of the day the aim of devops engineer is there should be very minimal manual process or no manual process.**

**Why devops?**

Developers will write the code for the software and handover to the operation team to do the deployment on the servers for end users. But when Ops team tries to deploy on their machine the code may not work due to some reasons. To avoid such kind situations DevOps came into picture.

Developer writes the code🡪 puts the code in centralized location🡪 system administrator/ engineer🡪 created the server and deploys the application in server🡪 tester will test the application🡪 build and release engineer promotes the app to either staging or prod🡪 application will be used the customer/ end user.

To improve the process of delivery/ automate all of the above steps devops was introduced.

**How to introduce yourself?**

What were we doing before switching to devops field? What was our roles and responsibilities?

**Day-2**

**Software development life cycle (SDLC):**

Its process followed by an industry to design, develop, deploy and test to deliver a high quality product to an end user/customer/client.

Planning (gathers the requirement)🡪 defining (software requirement specification document)🡪 designing (high level design and low level design)🡪 building🡪 testing🡪 deploy

During building, testing and deploy phases the devops comes to picture by making all these 3 phases faster, automated and quality efficiently.

**Building phase/ building the software**: it’s the development phase. In this phase, developer will write the code and move to source code repository GIT after getting the proper approvals.

**Testing phase**: during this phase the application will be deployed into server and QE (quality assurance team) team will be testing the application

**Deployment** **phase**: The application will be moved to prod environment for the end user.

***Software****: takes the input from the user and act according to the instruction.*

***Software development****: gathers the inputs and requirements from the user/ clients and develops the software.*

***Software life cycle: 1. Requirement*** *(the need and purpose) 🡪2.* ***Design*** *(how it should be and which tool to be used to develop the software) 🡪3.* ***Implementation*** *(actual development of software by writing the code) 🡪 4.* ***Verification*** *(checks whether the developed application works as per the client’s requirement) 🡪 5.* ***Maintain*** *(Maintain and monitoring the software)*

***Software development life cycle models:***

***Waterfall Model life cycle:*** *linear sequential model i.e until first phase completes we cannot start the next phase. There is no going back, once done is done.*

***Steps: Feasibility check*** *(same as requirement analysis) 🡪* ***Analysis*** *(Technical people/ architect will analyze the details which has been decided in the requirement phase whether it is feasible or not) 🡪* ***Design*** *(how the software going to be, what tools can be used and etc.)* ***🡪 Testing*** *(unit testing, system testing) 🡪 Maintenance (maintain and Supporting the software in case of any issues)*

***Advantages:*** *clear objectives, well understood milestone and the documentation will be very clear, each and everything will be captured in the document.*

***Disadvantages****: software will not be available till the final stage is completed. The requirement keep changes in over time of period. High risk and uncertainty*

***Agile****- To overcome the disadvantages of waterfall model agile have been introduced in 2000. Developing the software feature by feature. Creating the shorter development life cycle. Will be taking the customer feedback frequently in the process of developing the software.*

***Steps****: Process request 🡪 design 🡪 coding 🡪 Test and debugging 🡪 Release and Deployment*

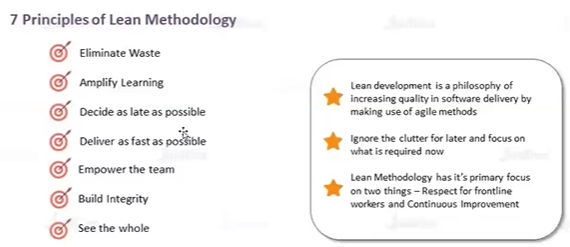
***Advantages:*** *customer satisfaction is high. Requirements can be dynamic in nature, less planning is required.*

***Disadvantages****: KT can be difficult as there is no documentation, not suitable for complex dependencies project.*

***Lean****: Same as Agile model. The only difference is eliminating waste. So the process will be faster than agile.*

***Advantages****: same advantage as agile. Creates positive working environment, limiting wastage saves the time and money.*

***Disadvantages:*** *Strong skill set team is required. No room for proper meetings and etc.*

**

**Day 3**

**Virtual Machines:**

Servers are used to deploy our applications for the end users.

**Virtualization**: in Virtualization hypervisor will be installed on the physical server.is a software that can install virtual machines on the bare metal or physical server. Means logical isolation of the server. So that all the resources will be used efficiently. Ex of hypervisors- VMware, xen. Different teams can use the same server in the form of virtual machines. Each has their own CPU, memory and hardware.

**Day 4**

**Creation of VM in AWS and Azure:**

AWS has created AWS EC2 API, AWS S3 API and etc… so whenever we want to automate something we will write the scripts and inside the script we will just call the API accordingly. Then the API will check the request and sends the response if the request is valid, authenticated and authorized.

The script which we write can be AWS CLI, AWS CDK(cloud development Kit), AWS API instance, AWS CFT(cloud formation Template), terraform. Using any of these option we can communicate with AWS API.

AWS CDK is mainly used to automate AWS resources.

We use terraform for automation when one company follows hybrid cloud.

**Day 5**

We can connect to EC2 in multiple ways.

1. In EC2 dashboard select the instance which is running, and click on connect.
2. We can connect using terminals like CMD prompt, putty, mobaxterm, nomachine and etc…

* Open the terminal which is comfortable for us.
* Copy the public address and enter SSH ubuntu@ipaddress
* SSH -i and paste the path where our key pair is saved.
* Chmod 600 – this will modifies the location of the key pair. Once done again type the above command.

**Automated methods to create EC2 machines:**

**AWS CLI guide:** AWS command line interface used to directly interact with our AWS services. To use this we need to download aws cli on our machine.

Once the CLI is installed, we need to login to our aws account. For that go to our profile in aws console, click on the username, click on the security credentials, under access keys- create one key, copy that access key.

In the terminal enter aws configure- then it will ask for the aws access key ID and secret access key. Here we need to copy paste from our aws cli. Then enter the region which we want.

[Launch, list, and terminate Amazon EC2 instances - AWS Command Line Interface](https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-instances.html) – this link will be used to get the template for creating any script for automation.

Use the below link for reference: [aws — AWS CLI 1.32.66 Command Reference (amazon.com)](https://docs.aws.amazon.com/cli/latest/reference/)

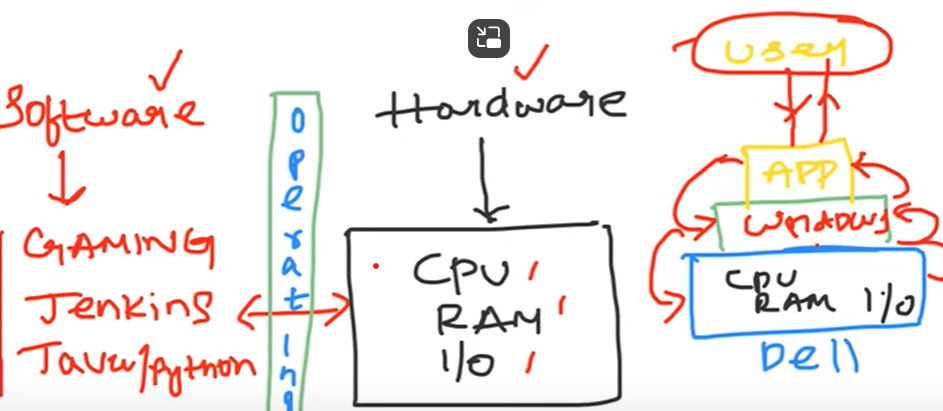
**AWS cloud formation Template:** it’s another option to interact with our aws console. In the aws console, go to CFT source select either one option template is ready or use a sample template.

We can automate aws using shell script, and also by programming languages like python. Need to install boto3 for the same. We can use boto3 document for writing the scripts.

**Day 6**

**Linux OS and Basics of shell scripting:**

A software which we install on our laptop/desktop it cannot directly talk to the hardware (CPU,memory, I/O). Here OS will help us to communicate both the software and hardware (Server). *OS acts as bridge between s/w and h/w. OS is the heart of the system/server.*



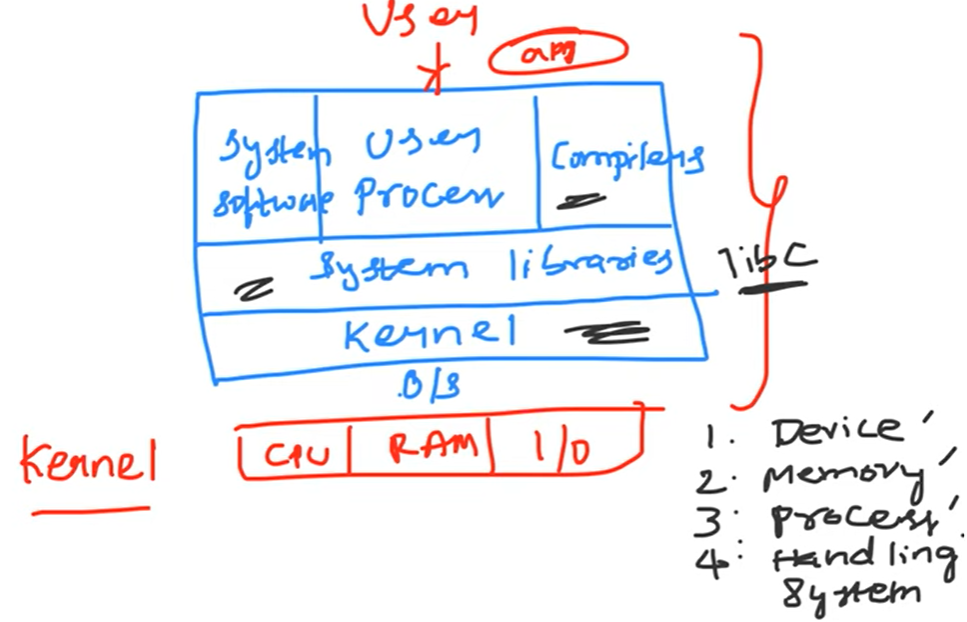
Linux os is very popular because, its free, open source software. More secure as we no need to install any kind of antivirus. Linux is very fast.

Heart of the linux OS is KERNEL. There are some 4 important aspects/responsibility are there for kernel.

* Device magnt
* Memory mgnt
* Process mgnt
* Handling the system related calls

**System Libraries:** responsible for handling the user requests. Each OS has their own system libraries (lipc)

**Architecture of linux OS:**



**Introduction/ fundamentals of Shell scripting:**

Shell is a command language used to communicate with our os. (Why shell? there will not be any GUI or any kind of folder structure inside the server, so we use commands to communicate with our server where we have installed linux OS).

Shell commands are same for different distributions like Ubuntu, fedora, centos.

**To connect our ec2 via our terminal:** ssh -i "C:\Users\aaa\Downloads\test1.1.pem" [ubuntu@3.235.175.186 (public](mailto:ubuntu@3.235.175.186%20(public) ip)

**Common basic commands used in shell:**

1. **ls**- list of the file
2. **pwd**- present working directory (which folder we are in)
3. **cd**- change directory
4. **cd ..**- to come out of one directory
5. **cd ../..** – used to come out of 2 directories
6. **ls –ltr or ls –ltra** – used to list all the files and directories along with timestamp, owner, permissions and few other details. (files which starts with d- directory, r- files)
7. **touch xxxx**– used to create empty file
8. **vi xxxx**- used to create a file and write content inside that
9. **mkdir xxx**-used to create directory
10. **rm**- used to delete the file
11. **rm -r** – used to delete the directory
12. **man (any keyword of shell command like touch,ls,cd,etc**…)- gives the detailed description about the command
13. **echo –** used to print any comments about the script
14. **cat XXX-** used to see the content
15. **sh xxx or ./xxx-** to execute any script
16. **chmod xxx-** to provide permission/ grant access for accessing the file
17. **chmod 777 xxx**- to grant all permissions to everyone
18. **history**- shows all the commands which we had used before
19. **echo**- used to print any description or comments
20. **kill**- to kill any running process

**Commands used to check the performance of a linux machine:**

1. **free –g –** used to check the available memory
2. **nproc-** used to find no of cpus
3. **df –**h – used to check the disk space
4. **top-** to check everything mentioned above we can make use of this single command (to check the node health/performance- combination of df –h, nproc and free -g)

**Shell scripting:**

Shell scripting in linux is used to automate our day to day activities which requires more manual work. How to create a file in shell? 🡪 touch xxxx.sh

Touch and vi/vim command used to create the file. But touch command mostly used in automation. vi/vim command used in creating and opening a file.

As soon as we are opening a file, we need to specify **#!/bin/(sh/bash/ksh/dash)** indentation/syntax which is called as seabang(#!). (Every shell scripts will be having this syntax at the beginning of the script)

**Why we are writing this syntax?**

**Bash/sh/ksh/dash🡪** all these are executables. We need to mention to inform linux/kernel which we want to use for executing the shell scripts.

***Note: bash is very important learn.***

**Diff between #!/bin/sh and #!/bin/bash:** In earlier days both were same.Even though we mention #!/bin/sh using linking concept it will be redirected to #!/bin/bash. But nowadays both become different as some os decided to use **dash** by default.So /sh is not getting converted into bash.

Its recommended to use always **#!/bash.**

Whenever we are creating any file in linux we need to grant permission to access the same. In linux giving permission is not directly possible. **chmod xxx** is the command which is used to grant permission for the file. ch means change the file.

Using chmod command we can mention what are the permissions for root user/admin, what are the permissions a group can have and what are the permissions a user can have.

If we want to grant access to everyone use chmod 777 xxx. 7 means 4+2+1. So if we mention 7 it will grant all the access read, write and execute permission.

We give/ grant the access number terminology. In the above line, three 7’s are there. One 7 for giving access to the user who created the filre, second 7 gives access for the group, another 7 gives access to everyone.

Linux is using one formula for read, write and execute which is 421 (4- read,2- write, 1- execute).

For ex if we are giving chmod 444 xxx- it will grant read access for the user, group and everyone.

**First shell program:**

Whenever we are creating any script, its always recommended to write few lines about whats the script is about, so that anyone who sees the script will be able to understand why it was written.

#!/bin/bash

#creating the folder

mkdir latha

#going inside the folder

cd latha

#creating the file

touch firstfile secondfile

Once the file creation is done we need to give the permission using chmod command.

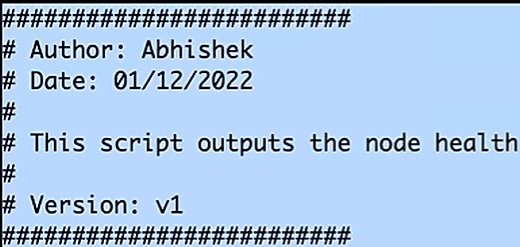
**Role/purpose of shell script in DevOps:**

As a devops engineer we will be having different activities like infra maintenance, maintain the code (using git (linux)), and configuration management. For all of these purposes we use shell script in our day to day activities.

For ex: automated the monitoring performance, CPU(**nproc**), RAM(**free**) of 10000 vms using shell script.

**Advanced shell script:**

**Metadata:** Whenever we write any shell script its better to write along with the metadata which is like some introduction about the author and the script



**echo –** used to print any comments about the script

**set –x –** used to make the shell script in debug mode (when we use this command it will show us the command along with the output) (in other words it will show us which command is used for what purpose)

**ps –ef –** used to print all the information about the processes which is running on our machine.

**Ps –ef | grep “amazon”-** used to fetch/filter only the required processes using the keyword mentioned in the quotation. ( here amazon is just an example). | 🡪 sends the output of first command to the second command

Interview question: when we write date | echo “xxxxxxxx” it will print only xxxxxxx as date is system default command and it will send the output to stdin our | pipe parameter cannot receive the output from first command. Read about awk

**Ps –ef | grep “amazon” | awk –F” “ ‘{print $2}’**- when we want to get only particular column information

Whenever we are using pipeline(|) in our script it’s important to mention 2 things.

***set –e🡪*** *exits the script when there is an error in our script. But this will not error out or exit when there is pipeline. To avoid this we use the below command*

***set –o pipefile***

***set –exo pipefail🡪*** *combination of set –e and set –o pipefail -but this is not recommended as it may lead unknown error or deletion.*

**Logfile- to find the errors in case of any issues:**

**Curl link** where the logfile is available **–** used to retrieve the logfile from any external storage along with grep command we can search for the errors.

**Wget link** where the logfile is available – same as the curl command where curl will give us the direct content / output but wget will save the logfile in our machine.

**Sudo su -** 🡪 using this command we can switch to the root user/ any other user

**Sudo find /-name xxx(filename which we want)-** used to find any file available in the virtual machine

**If/ if else/ for loops:**

**If syntax:**

if [condition]

then

echo “print statement”

else

echo “print statement”

**trap command: used to traping signals**

[Unix / Linux - Signals and Traps (tutorialspoint.com)](https://www.tutorialspoint.com/unix/unix-signals-traps.htm) read important interview question

Trap “echo don’t use the ctrl+C “ SIGINT

Most commonly asked interview questions about shell scripting:

1. List some of the commonly used shell commands

**Ans**: ls- listing the files, cd, mkdir,touch, vim, grep, find, top, df, kill and tell the purposes of those commands

1. Write a simple shell script to list all the processes

**Ans:** ps –ef | awk –F” ” ‘{print $2}’

1. Write a script to print only error from the remote log

**Ans:** curl remote path/link | grep “error” (curl+|+grep)

1. Write a script to print numbers divided by 3, 5 but not 15

**Ans:** for $i in {1..100}; do

if [ ‘exp $i % 3’ == 0]; || if [‘exp $i % 5’==0] && if [‘exp $i % 15’!=0]

then

echo $i

fi;

done

1. Write a script to print no. of “S” in mississipi

**Ans**: a=Singapore

grep -o "s" <<<"$a" | wc –l

1. How you will debug your shell script?

**Ans**:Set –x will make our script in debug mode

1. What is crontab in linux? Example of usage

**Ans**:Crontab is like an alarm in linux, when we need to send some kind of report daily by running the shell script we can make use of crontab. When we set crontab at 6pm , our linux will run/execute the script and sends the report automatically.

1. How do open a script in read-only mode?

**Ans** : Vim –r test.sh (r is for read only)

1. Difference between soft and hard link

**Ans:** **Hard link** is used to create copy of any file. When the original file gets deleted the copy of the file will not be deleted. Means we can reuse the file.

**Soft link** is like creating alias for some application. In soft link when the original file/copy is getting deleted everything will get deleted automatically. It can’t be reused.

1. Difference between break and continue

**Ans** : Break- Breaking the execution when the condition is failed or met.

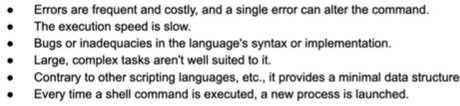
Ex: we need to find whether the given number is there or not in particular range. The moment we get the required number we can use break and come out of the execution, no need to search further.

Continue- skip and continue the execution when the condition is failed

Ex: in continue, even the number is found it will check further to find if there is anything else

break is used for immediate termination of loop. On the other hand, ‘continue’ terminate the current iteration and resumes the control to the next iteration of the loop.

1. Disadvantages of shell scripting:



1. Different types of loops and when to use in shell script

Refer Google

1. Is bash dynamic or statically typed and why?

Bash is dynamically typed as we no need to mention any variable type when we are initializing value to any parameter. Like a=5. Static means we need to mention the data type of the variable before initializing value.

1. Explain about a network troubleshooting utility

**Traceroute** is the best tool for network debugging. When we want to find out why our netwoirk is slow just write traceroute google.com lap🡪router🡪isp🡪…. 🡪 google

There is another one tracepath is also one tool for troubleshoot.

1. Sort the list of names in the file

Linux is having a default command for sorting the order, so we can make use of that.

1. How you will manage logs of a system that generates huge logfile every day?

We can use the logrotate command along some conditions like once in a day just create the compressed zip folder of that log file, then the files which is older than 30 days just delete those files.

Day -6

**Write a shell script to report the usage of AWS in your project**

There are 2 main reasons why someone will move to cloud

1. Management
2. Cost effective

As an AWS resource our primary goal is to maintain the cost effectiveness by tracking the resource usage. There are multiple ways to do this. Shell scripting is one of them

**Cronjob** is a concept in linux used to run/execute any script automatically at particular time.

Mention this in resume:

Shell scripting projects github link: [GitHub - iam-veeramalla/shell-scripting-projects: Projects on shell scripting. Goal is to simulate real time projects as much as posible.](https://github.com/iam-veeramalla/shell-scripting-projects)

Scenario: there is a manager in one organization to whom we need to send report at a particular time, about the usage of EC2, lambda, s3, IAM and etc…. Ideally all these reports will be sent to reporting dashboard. For that we will write a shell script, then integrated with cropnjob and using cronjob to run the script at particular time every day.

**Pre- requisites:** in windows AWS CLI installation should be done. Inside the linux machine we need to use these commands.

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

sudo apt install unzip

unzip awscliv2.zip

sudo ./aws/install

Once the aws cli is installed we need to do the aws configure. It will ask for the token id and the key value along with region and file format.

Then we need to create one shell script to execute this. Use the link to write the script

[aws — AWS CLI 1.32.76 Command Reference (amazon.com)](https://docs.aws.amazon.com/cli/latest/reference/)

#Date: 04/03/2022

#version: 0.1

#This script is to create report about the resource usage in aws

#AWS S3

#AWS lambda

#AWS EC2 instances

#AWS IAM users

##################################

#list of S3 buckets

echo "list of S3 buckets"

aws s3 ls

#list of S3 buckets

echo "list of EC2 instances"

aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'

#list of lambda functions

echo "list lambda functions"

aws lambda list-functions

#list of IAM user

echo "list of IAM users"

aws iam list-users

Homework: run the above script, save the output in one file, also use cronjob

**Day: 8**

**Shell Scripting Project Used In Real Time | GitHub API Integration: Advanced devops project using shell script:**

Github provides below document which is very useful when we try to do some activities through the CLI. [GitHub REST API documentation - GitHub Docs](https://docs.github.com/en/rest?apiVersion=2022-11-28)

As a devops engineer, in our day to day we will be supporting multiple teams. For each team we will be maintaining one repository in the github with the proper access. We can automate this process with automation using shell script. This is very common task of a devops engineer.

To check who and all having access to the particular repository in our organization we execute the command along with organization name and repository name. for that we should be having access to that repo first.

**Ex: ./list-users.sh(script name) devops-by-example(org name) Python (repo name)**

If we are not having access to that repo will be getting error like “jq: error (at <stdin>:4): Cannot index string with string "permissions"”

#!/bin/bash

Helper()

# GitHub API URL

API\_URL="https://api.github.com"

# GitHub username and personal access token

USERNAME=$username

TOKEN=$token

# User and Repository information

REPO\_OWNER=$1

REPO\_NAME=$2

# Function to make a GET request to the GitHub API

function github\_api\_get {

local endpoint="$1"

local url="${API\_URL}/${endpoint}"

# Send a GET request to the GitHub API with authentication

curl -s -u "${USERNAME}:${TOKEN}" "$url"

}

# Function to list users with read access to the repository

function list\_users\_with\_read\_access {

local endpoint="repos/${REPO\_OWNER}/${REPO\_NAME}/collaborators"

# Fetch the list of collaborators on the repository

collaborators="$(github\_api\_get "$endpoint" | jq -r '.[] | select(.permissions.pull == true) | .login')"

# Display the list of collaborators with read access

if [[ -z "$collaborators" ]]; then

echo "No users with read access found for ${REPO\_OWNER}/${REPO\_NAME}."

else

echo "Users with read access to ${REPO\_OWNER}/${REPO\_NAME}:"

echo "$collaborators"

fi

}

Function helper {

Expected\_cmd\_args=2

If [$# -ne $expected\_cmd\_args]; then

Echo “please execute the script with required cmd parameters”

Echo “asd”

# Main script

echo "Listing users with read access to ${REPO\_OWNER}/${REPO\_NAME}..."

list\_users\_with\_read\_access

**Day 9: Git and Git Hub:**

**What is version controlling system:** the core concept of git and git hub is vcs. The primary of vcs is to address 2 major things which is sharing of our code and versioning.

**What is sharing a code**? There are 2 developers who is working on one project, both are working in different functionalities and end of the days both of their codes needs to be at one place for the project to work. In real life scenarios it’s difficult to share which can be resolved by version control.

**What is versioning?** We have one application where the clients wants to add some more features in it, then developers start implementing one by one. When anyone from the team wants to see the initial version of the code it’s difficult to go back by removing all changes we did for implementing the features said by the customers. This issue can be addressed using versioning.

There are more popular tools available for version control system.

* Cvs
* Svn
* Git

**Why git became very popular?** The cvs and svn is centralized version control system. Git is distributed VCS.

**What is centralized vcs:** in CVCS the developers used to communicate with each other using central server which is svn. Here if the central server goes down there will not any communication between the developers. Always need to connect with central server.

**What is Distribute VCS?** Here developers can create multiple copies from distributed version. Can share his changes to the copied file. Even the original goes down, we can use the copies. Instead of one common place developers can communicate multiple places.

What is fork? Creating entire copy of the original source/code. Cloning the repo from one github account to another github account.

In simple words GIT is a version control tool, to track who has done what.

**Difference between Git and GitHub**

Git is dcvs open source where we can download and use for storing our code. Github is same as Git along with better usability, resolving issues, commenting, reviewing and project management etc….

**Git Commands:**

**Git init-** to initialize the git. When we do git init git repository .git will be created. To check whether git is initialized or not we can use **ls –ltra.** If .git is deleted nothing will be tracked by git.

Inside .git, we can find Head, config, hooks, objects and refs. Whatever file we create everything will be tracked as objects and refs.

Hooks will be used to prevent unwanted commits.

Config is used to configure some certificate and other things.

**Git status:** shows the tracking and untracked files.

**Git add:** used to move the untracked file to staging area. **Git add filename**

**Git diff:** used to find the changes made in the existing file.

**Git commit:** used to commit the file in git. **Git commit –m “message about the commit” filename**

**Error 1:** while doing the first commit we may face below error

Author identity unknown,\*\*\* Please tell me who you are. Run git config --global user.email "you@example.com" git config --global user.name "Your Name" to set your account's default identity. Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'root@agent.(none)')

**To resolve this just follow below steps:**

git config --global user.email [hellogirllara7@gmail.com](mailto:hellogirllara7@gmail.com)

git config --local user.name "hellogirllara"

git config --list (to check whether the user name and email go updated or not

**git log**- to track the changes of the file, to find the commit id

**git reset:** if we want to go back to any particular version of the code we can use **git reset --hard commitid.** This is one of the main advantage of git. Using git we can switch to any versioning.

**git show “commit id”:** To see/ know about one commit

**git restore –staged filename** : To move back from commit and staging are to working area

Whatever we had implemented, or changed in our local machine needs to share with other team members. Here our distributed version control github will come into picture.

**What is git branching strategies?**

The main goal of any organization is to ensures customers satisfaction and delivers the releases on time. For that we need to have a proper branching strategies or efficient branching.

Now we are going to learn about kubernetes branching strategies as they are having 3300 contributors which is huge.

**Master branch: (Important for interview) - why do we need to create a branch?**

The branch which is created first becomes the master branch. In the master branch, live code will be available which is running successfully and used by the end users. When a developer or a product owner wants to do some enhancement, we can’t go ahead and edit the master branch as the live code is running without any issues. So in such case we create a new branch from the master branch. As we creating new branch from the master all files will be inherited to the newly created branch. Ex: UBER app. The master branch will always up to date.

**What is feature branch?**

When a developer or a product owner wants to do some enhancement, we can’t go ahead and edit the master branch as the live code is running without any issues. So in such case we create a new branch from the master branch. That new branch is called as feature branch.

**What is Release branch?**

Once all the features are implemented and tested we will create a release branch and move our code to it. Instead of providing the main branch, we will build the code from the release branch and we will deliver to customers. Why we are not sharing main means, there will be a user’s who is actively using the application so first release branch will be release to the customers for testing, once everything is fine this changes will be moved to main/ master branch.

**What is hot-fix branches?**

When there is an issue in the production we need to fix that immediately we will be creating the hot-fix branches. Once we fix the changes will be moved to release as well as master.

**Day-11:**

**Git commands and interview questions:**

**How to create or initialize git repository?**

**Ans:** using **git init** command we can create/initiate git. When we initialize hidden folder .git will be created inside the directory.

**How to prevent developers pushing the password into git?**

**Ans:** using git/hooks/pre-commit we can prevent.

**Hot to push the file into github?**

**Ans:** git push url of our repo master

**What is the git workflow followed in an organization?**

**Ans:** git add & git commit –m “message” & git push url of repo

**How to add the remote repo to our local?**

**Ans:** Git remote add url of the repo

**Git cloning or how to pull the code from github?**

**Ans:** cloning is getting code from remote repo to our local. **Git clone url of the repo master.** We can clone using either https or ssh. If we are using https we just need to copy the url. If we are using ssh we need to use the publickey. We can create the publickey using **ssh –keygen –t rsa .T**he public key will be created in our local. Once the file is created we can copy and paste it in our github account under settings.

**Difference between clone and fork:**

Clone is used to download any particular repository from the github whereas fork is used to create the copy or replica of any repository.

* Forking is done on the GitHub Account while Cloning is done using Git.
* When you fork a repository, you create a copy of the original repository (upstream repository) but the repository remains on your GitHub account. Whereas, when you clone a repository, the repository is copied on to your local machine with the help of Git.
* Forking a repository allows developers to create independent copies, while cloning a repository creates a local copy of the project.

**Lifecycle of GIT: git init🡪 git add🡪 git commit**

**Working directory** (is the place where we have initialized the git –untracked files) ** Staging area** (tracked files and ready to commit)  **commit** (we can’t make any changes to the file once the commit is done).

Creating a branch: -**Git branch branchname**

Creating and switching to that branch: **git checkout –b branchname.** This will be helpful for development activities. Using branching we can isolate the different codes.

Once the changes are done the code will be moved to the main/master branch. For that we can use 3 things which is git merge, git rebase or git cherry-pick.

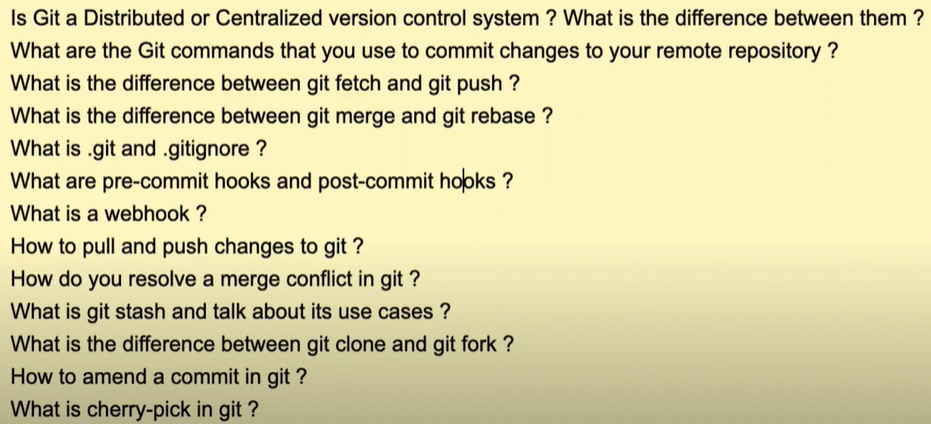
Among these cherry-pick is simplest one when there are one or two commits. **Git cherry-pick commitid.** But when we have n number of commits its not practically possible to check the logs and pick the commit id and do the cherry-pick. So we can use either merge or rebase.

Ideally git merge and rebase does the same thing by keeping master up to date. The only difference is in merge commit we cannot find the commits in linear/ sequential pattern. But when we do rebase the commits will be in sequential order**.**

**Git conflicts:**

When we are not taking the updated version of code and makes changes we face conflict when we try to commit our changes. We need to check with the respective people to check what is needed which needs to be removed. Once the conflict is resolved we need to commit the file again.

**Interview questions:**

****

1. Centralized vcs is a tradition method hardly or nowhere people are using this system. Ex svn and cbs. It follows client server architecture. So all the copies of code will be available only in the server.

**Git is a distributed vcs,** where every developer has the copy of the code up to date by cloning the repository.

1. Git init, git add, git commit –m “message” file name, git push url of the repo master
2. Diff between git fetch and git pull?

When we using git fetch it will show us the latest changes done on the particular repository, but it will not merge those changes to our local so we need to use two commands while using git fetch. **Fetch Command**: is used to get file from github gui to our local cli- **git fetch url of repo branch name + git merge url of repo branchname(from where we are pulling file)**

When we git pull it will show latest changes and merge those changes to our local.

1. Git merge and git rebase

Both merge and rebase does the same to keep all the branches up to date, except how they are projecting the history. Git rebase will give linear code history where git merge will not.

1. .git and .gitignore

.git folder will get created when we initiate the git in our directory. It contains objects, hooks and other folders. Used to track our files. .gitignore we use when we don’t want to keep a track of any file.

1. What are pre-commit and post-commit hooks?

The action that are taken by git before commit- pre commit hooks

The action that are taken by git after commit- post commit hooks

1. What is webhook?

Refer google

1. How to pull and push changes to git?

git pull url of repo && git push url of remote repo branch name

1. Stash / **UnCooked file:**

In dev there is one file created in dev but it is not yet committed, meanwhile if we switch to any other branch we can see the uncommitted file which available in dev. That particular file is called as uncooked file. In this scenario we cant commit any new changes until that file gets committed.

To avoid this issue, first we need to switch to the branch where we have uncooked or uncommitted file. Then we can use stash command- **git stash –u** (u- untracked)

If we want to work on the uncommitted file, to bring back that file- **git stash pop**

1. Fork and clone

Fork getting repo from one account to another account

Clone- getting copy of our remote to our local

1. Cherry-pick: getting any specific commit directly from one branch to another branch if there are no much commits
2. How to amend a commit in a git 🡪 git commit –amend
3. How to resolve merge conflict? Check with developer which is correct what needs to be present.

**Day 13:**

**Top 15 AWS services every devops engineer should know:**

1. EC2
2. VPC (security)
3. EBS (volume)
4. S3 (Storage)
5. IAM (access)
6. Cloud watch (monitoring)
7. LAMBDA (serverless)
8. Cloud build services- like Jenkins, aws provides 3 services to build and deploy any application. 1. AWS code pipeline, 2. AWS code build and 3. AWS code deploy