GIT:-

- Git is a technology and a free and open source distributed version control system to handle everything from small to very large projects with speed and designed efficiency.
- Github- Online service for hosting Git repositories.

- It was created by Linus Torvalds in 2005 to develop Linux Kernel.
- Version Control is the management of changes to documents, computer programs, large websites and other collection of information.
- **what is diff b/w git and github?**
  Ans: Git is a version tool
  github is online repository manager.

- Jenkins will clone that code. Once the code is downloaded other tool will commit.
- And using maven and gradle we can create the jar and war file.
- We require git to clone code from github to jenkins.
- We require git tool on jenkins server so that we can clone the code.
- We require git on dev, developer and jenkins server.
- All the instructions will be written in Jenkins.
- We should be good with the tool configuration.
- Not with the coding language.

- Ideally we use master branch For Dev ,QA and Pre-prod Env release,
- For final prod deployment , we should use release branch.
- How many branches are there in your organization.
- Mainly we work with 5 branches.
- Master,release,deveops,developer branch and there are some feature branches.
- Also depend on requirement we create number of branches.
- Depend on how many features we are using.

Git-version: - is tool that tracks changes to a file over time so that you can recall any specific version (you can point to any specific version)

- Git has the functionality, performance, security and flexibility that most teams and individual developers need.
- Latest version of git-2.29.2

Features of GITS

1. Free -and- open-source
2. Speed
3. Scalable

4. Reliable
5. Secure
6. Economical
7. Supports non-linear development
8. Easy Branching:
9. Distributed development
10. Compatibility with existing systems or protocol:
11. Lightweight
12. Tracks history
13. Creates backup

==========
==========

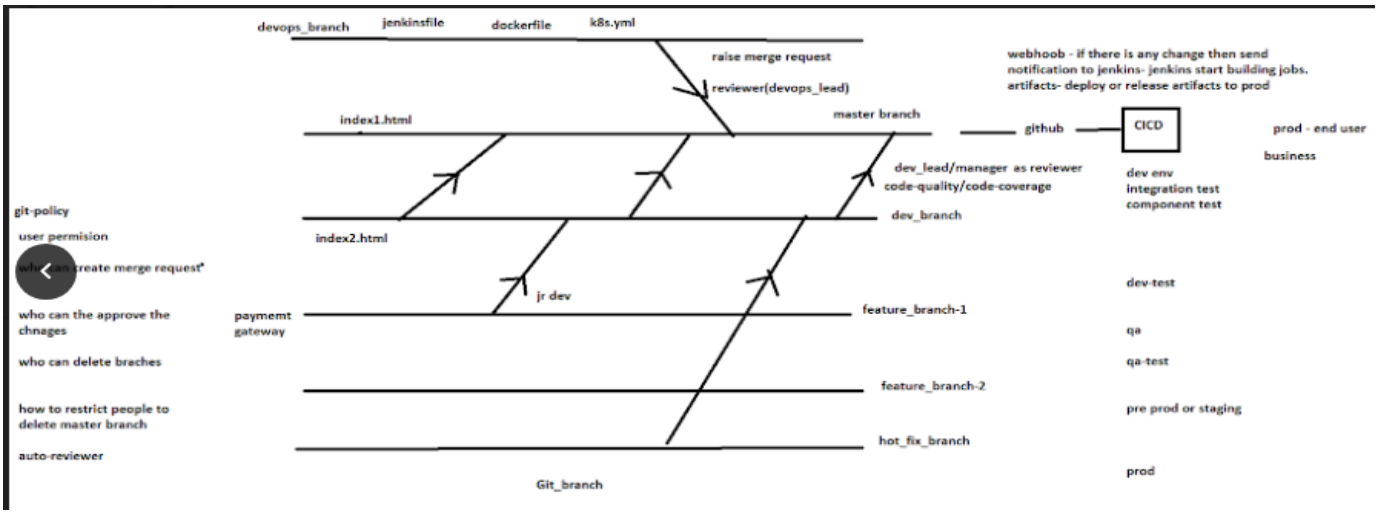
We can create CICD pipeline only on master branch… as it is reviewed or approved code.
Multiple developers on multiple branch like junior, senior hence we have multiple branch.
We have one more branch for devops team. for devops related activity.
New branch will be replica of original branch.
Junior developer can create new merge request. Approval is required from senior developer issue branch also can be created and later you can merge.
Bug fix branch .Release branch we create for prod environment. Master branch for dev/QA environment.

==========

Q. diff b/w git and SCM (both are version control tool)

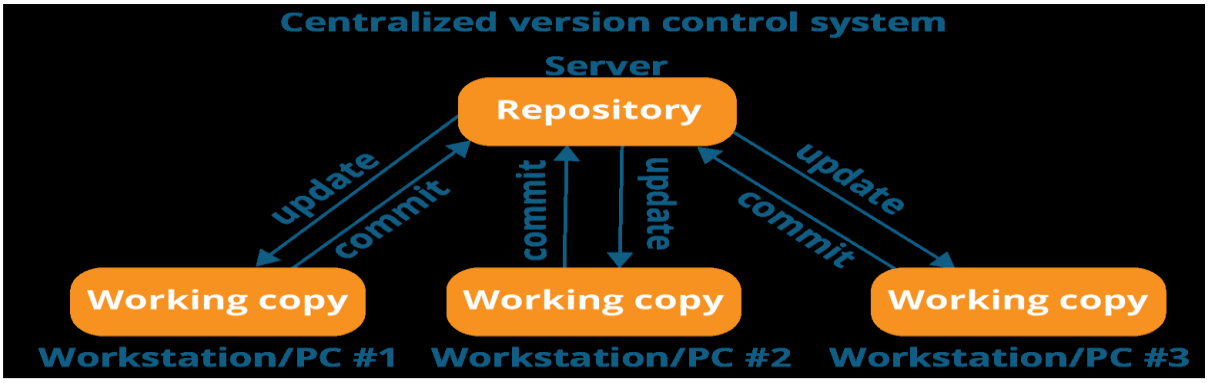
Ans : git- follows distributed vesion control
we can commit locally without having internet connectity
SCM : follows centralized version
without internet connection , we cant commit

==========



There are two types of VCS:
Centralized Version Control System(CVCS)
Distributed Version Control System(DVCS)

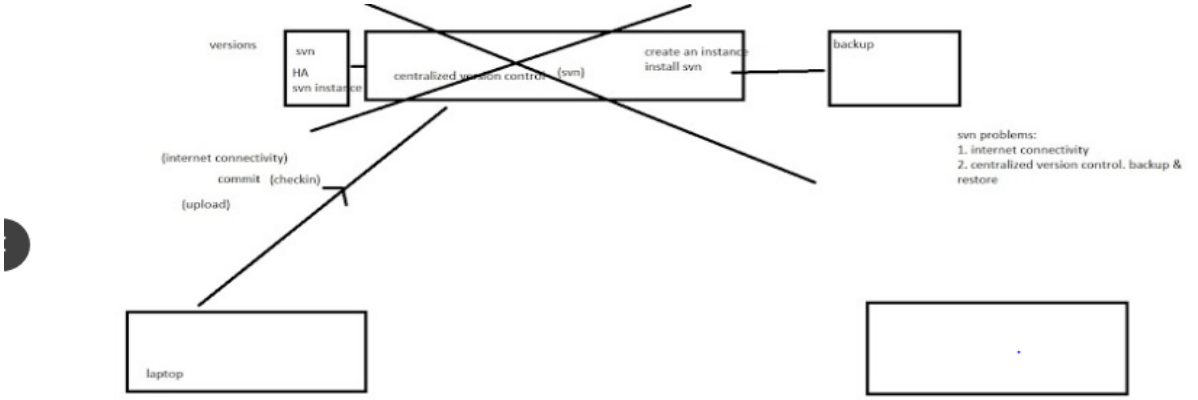
Centralized VCS
Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration. It works on a single repository to which users can directly access a central server.
Please refer to the diagram below to get a better idea of CVCS:



The repository in the above diagram indicates a central server that could be local or remote which is directly connected to each of the programmer's workstation.
Every programmer can extract or update their workstations with the data present in the repository or can make changes to the data or commit in the repository. Every operation is performed directly on the repository.

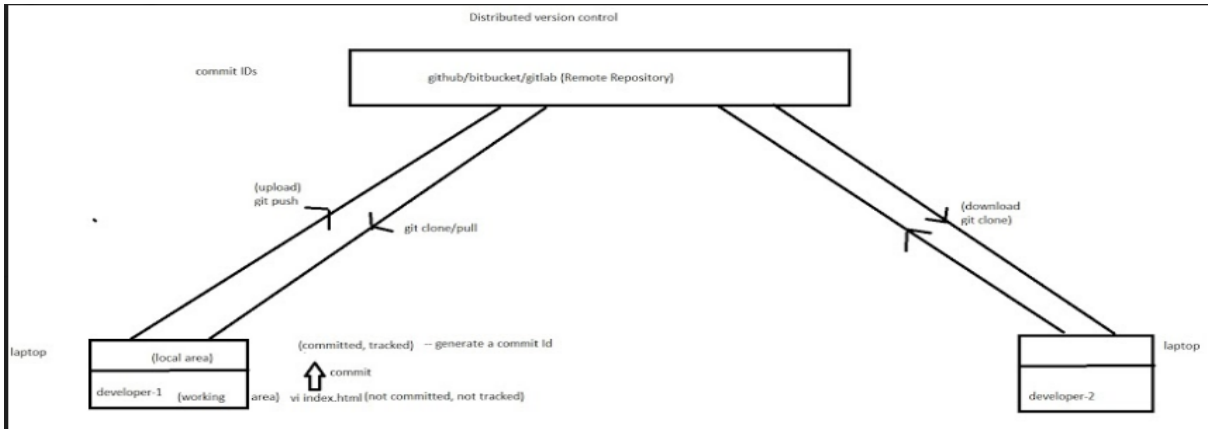
Eventhough it seems pretty convenient to maintain a single repository, it has some major drawbacks .Some of them are:
It is not locally available; meaning you always need to be connected to a network to perform any action.

Since everything is centralized, in any case of the central server getting crashed or corrupted will result in losing the entire data of the project.

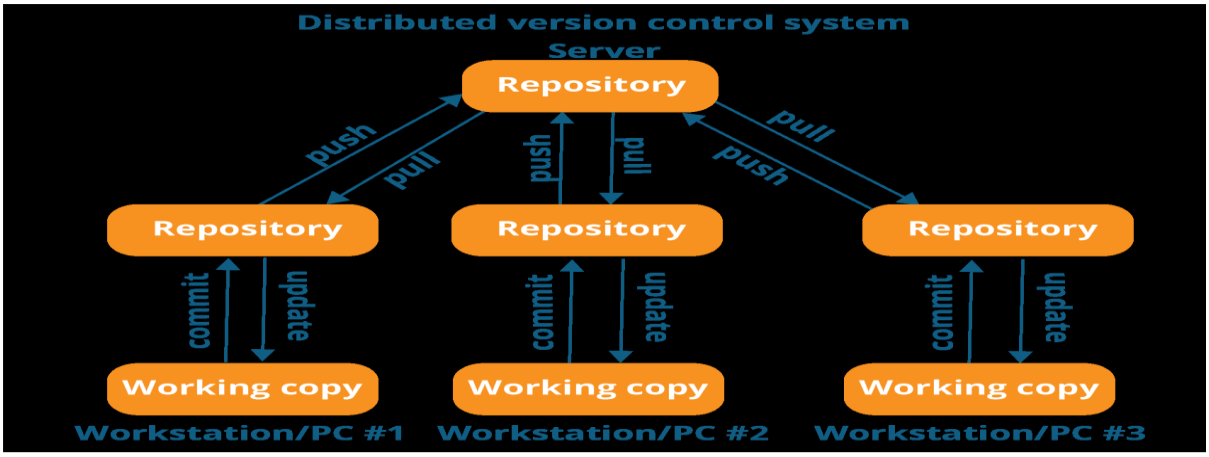


Example –SVNtoolisCentralizeversioncontrolmanagementtool.

Distributed VCS-



- File is getting saved on two locations like git repository and local repository.
- These systems do not necessarily rely on a central server to store all the versions of a project file.
- In Distributed VCS, every contributor has a local copy or“ clone”of the main repository i.e. everyone maintains a local repository of their own which contains all the files and metadata present in the main repository.



4 stages.

- **Working copy**
- **Staging/index**
- **Local repository**
- **Remote repository**

- Every programmer maintains a local repository on its own, which is actually the copy or clone of the central repository on their hard drive. They can commit and update their local repository without any interference.

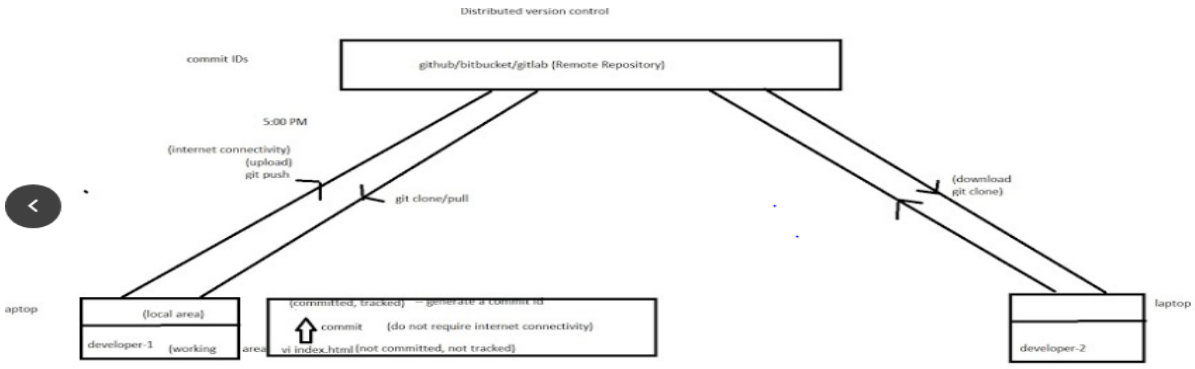
- They can update their local repositories with new data from the central server by an operation called “pull” and affect changes to the main repository by an operation called “push” from their local repository.

- The act of cloning an entire repository into your workstation to get a local repository gives you the following advantages:

- All operations (exceptpush&pull) are very fast because the tool only needs to access the harddrive, not a remote server. Hence, you donot always need an internet connection.

- Committing new change-sets can be done locally without manipulating the data on the main repository. Once you have a group of change-sets ready, you can push the mallat once.

- Since every contributor has a full copy of the project repository, they can share changes with one another if they want to get some feedback before affecting changes in the main repository.



If the central server gets crashed at anypoint of time, the lost data can be easily recovered from any one of the contributor's local repositories.

Features of GITS

Free-and-open-source:

Gitis released under GPL's (General Public License) open source license. You don't need to purchase Git. It is absolutely free.And since it is open source, you can modify the source code as per your requirement.

Speed:

Since you do not have to connect to any network for performing all operations, it completes all the tasks really fast. Fetching version history from a locally stored repository can be one hundred times faster than fetching it from the remote server.The core part of Gitis written in C, which avoids runtime overheads associated with other high level languages.

Scalable

Gitis very scalable. So, if in future , the number of collaborators increase Gitcan easily handle this change.

Reliable-

Since every contributor has its own local repository, on the event so far system crash, the lost data can be recovered from any of the local repositories. You will always have a backup of all your files.

Secure:
Git uses the SHA1 (SecureHashFunction) to name and identify objects within its repository.The Git history is stored in such a way that the Idofa Once it is published, it is not possible to change the old versions without it being noticed.

Economical:-
In case of CVCS, the central server needs to be powerful enough to server requests of the entire team. For smaller teams, it is not an issue, but as the team size grows, the hard ware limitations of the server can be a performance bottleneck. In case of DVCS, developers don't interact with the server unless the yneed to push or pull changes.All the heavy lifting happens on the client side,so the server hardware can be very simple indeed.

Supports non-linear development:
Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history. A core assumption in Git is that a change will be merged more often than it is written, as it is passed around various reviewers.Branches in Git are very light weight. A branch in Git is only a reference to a single commit.With its parental commits, the full branch structure can be constructed.

Easy Branching:Branch management with Gitis very simple. It takes only few seconds to create, delete, and merge branches. Feature branches provide an isolated environment for every change to your code.
When a developer wants to start working on something, no matter how big or small, they create a new branch. This ensures that the master branch always contains production-quality code.

Distributed development:
Git gives each developer a local copy of the entire development history, and changes are copied from one such repository to another. These changes are imported as additional development branches, and can be merged in the same way as a locally developed branch.

Compatibility with existing systems or protocol:

Repositories can be published via http, ftp or a Git protocol over either a plain socket, or ssh. Apache Sub Version(SVN) and SVK repositories can be used directly with Git-SVN.
Git plays a vital role when it comes to managing the code that the collaborators contribute to the shared repository. This code is then extracted for performing continuous integration to create a build and test it on the test server and eventually deploy it on the production

Commit messages in Git play a very important role in communicating among the team.The bits and pieces that we all deploy lies in the Version Control system like Git.To succeedin DevOps, you need to have all of the communication in Version Control. Hence, Git plays a vital role in succeedin gat DevOps.

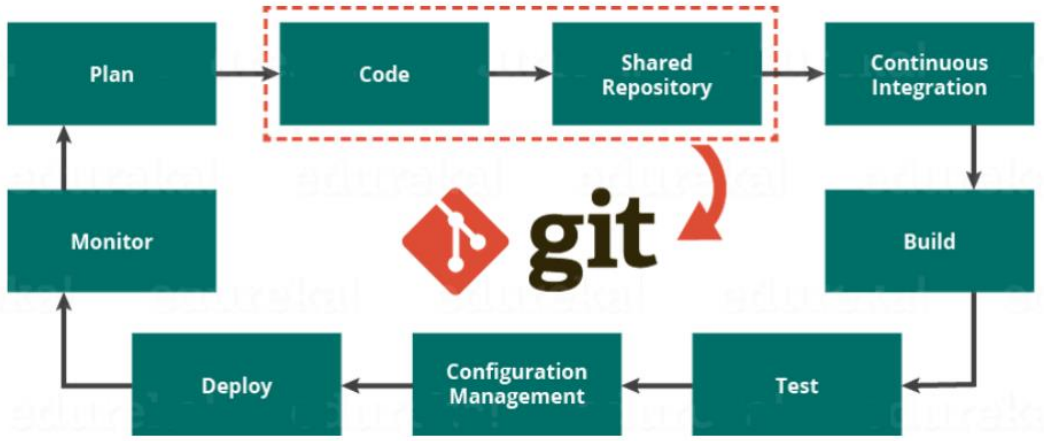
Companies Using Git:

Git has earned way more popularity compared to other version control tool available in the market like Apache Subversion(SVN),Concurrent Version Systems(CVS), Mercurial etc.Some companies that use Git for version control are: Facebook, Yahoo, Zynga, Quora, Twitter,eBay, Salesforce, Microsoft and many more.

Role Of Git In DevOps?

Now that you know what is Git, you should know Git is an integral part of DevOps. DevOps is the practice of bringing agility to the process of development and operations. It's an entirely new ideology that has swept IT organizations worldwide, boosting project life-cycles and in turn increasing profits. DevOps promotes communication between development engineers and operations, participating together in the entire service life-cycle, from design through the development process to production support.

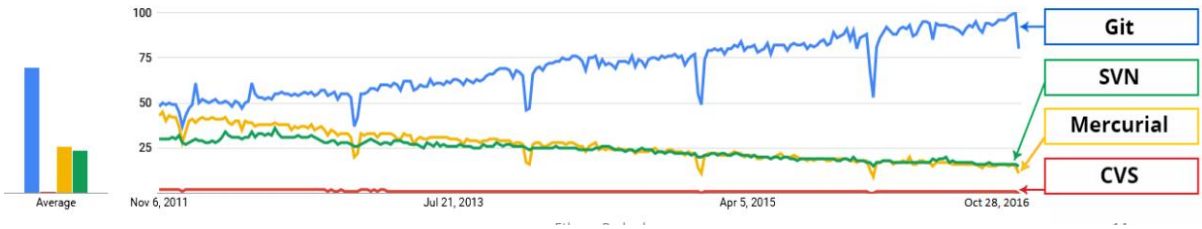
The diagram below depicts the Devops life cycle and displays how Git fits in Devops.



Git plays a vital role when it comes to managing the code that the collaborators contribute to the shared repository. This code is then extracted for performing continuous integration to create a build and test it on the test server and eventually deploy it on the production.

Tools like Git enable communication between the development and the operations team. When you are developing a large project with a huge number of collaborators, it is very important to have communication between the collaborators while making changes in the project. Commit messages in Git play a very important role in communicating among the team. The bits and pieces that we all deploy lies in the Version Control system like Git. To succeed in DevOps, you need to have all of the communication in Version Control. Hence, Git plays a vital role in succeeding at DevOps.

Companies Using Git:
Git has earned way more popularity compared to other version control tools available in the market like Apache Subversion(SVN), Concurrent Version Systems(CVS), Mercurial etc.



You can compare the interest of Git by time with other version control tools with the graph collected from Google Trends below: Ethans-Prakash 14 In large companies, products are generally developed by developers located all around the world. To enable communication among them, Git is the solution. Some companies that use Git for version control are: Facebook, Yahoo, Zynga, Quora, Twitter, eBay, Salesforce, Microsoft and many more. Lately, all of Microsoft's new development work has been in Git features. Microsoft is migrating .NET and many of its open source projects on GitHub which are managed by Git. One of such projects is the LightGBM. It is a fast, distributed, high performance gradient boosting framework based on decision tree algorithms which is used for ranking, classification and many other machine learning tasks. Here, Git plays an important role in managing this distributed version of LightGBM by providing speed and accuracy. Ethans-Prakash 15 git init This command turns a directory into an empty.
==========
==========
==========

How to set up permission when you go to organization

- Go to git hub and want to create a project.
- Then on right top corner, we have new organization
- Then for test go for free…for organisation we go for enterprise.
- In project we create group.
- Then join free type and
- Set up your organization name
- And give email id and my personal account
- Now we can create repository.
- Under organisation we can create repositories for our project and applications.
- Provide what type of project there and submit
- Like devops repository.
- We can think of permission here.
- Add email id there who is going to be work or part of organisation.
- Invite teams or people-----put email id-
- So we have read permission also-so that he can view the code.
- Contributor role is for senior developer role.
- Likewise we set up the permission and later on we can change the permission.
- Admin can change the permission.
- Here we have given user level permission.

- If we have 100 people … so adding user and managing permission is difficult
- So we have team option there so create and user can be added to the team.
- And then assign permissions to it.
- **This is how we maintain permissions.**
- **Owner role** (dev-ops team- owner of this repo)
  (Can perform any activity such as branch creation/deletion/repository creation/repo deletion etc)

- **Maintainer role** (all the leads- dev lead/Qa lead/ release manager/ sr folks / higher management---> can create branches/delete ,
  can't create new repository)

  Guest role (only read access)
- **Developer role** (jr folk)
- **Reporter role** (higher management)
- **Owner role** (devops team- owner of this repo)

==========

- Migration from SVN to Git:
- we need to:-
- maintain all the log
- maintain user permission
- maintain reference url
- Ans: subgit tool

- SVN migrations to Git can vary in complexity, depending on how old the repository is and how many branches were created and merged, and whether you're using regular SVN or close relative like SVK.
- It could be simple if:
- You have a new repository
- You have a standard setup of a trunk, branches, and tags directory
- It's likely going to be complex if:
- Your team has performed many branching and merging operations
- Your repository follows a non-standard directory setup
- Your directory setup has changed over time
- There are several ways to migrate from SVN to Git. The approach outlined in this article is based on using git-svn, a Git extension, which can be used to check out a Subversion repository to a local Git repository and then push changes from the local Git repository back to the Subversion repository. These steps give a detailed overview of the process for migrating from SVN to Git in a Windows environment, without synchronizing back to the original SVN repository. The result will be a bare Git repository for sharing with the rest of your team.
- The high-level workflow for migrating from SVN to Git is as follows:
- Prepare a migration environment
- Convert the source SVN repository to a local Git repository
- (Optional) Synchronize the local Git repository with any changes from SVN repository while developers continue using SVN

- Push the local Git repository to a remote Git repository hosted on Azure Repos
- Lock SVN repository, synchronize any remaining changes from SVN repository to local Git repository and push final changes to the remote Git repository on Azure Repos
- Developers switch to Git as main source control system

----------**Interview questions**----------

Difference Between Git and AWS S3?

Features

In Git, we can smoothly work with Git like we can go back to previous version easily, we can maintain the version, we can do the changes.

In S3, we cannot go back to previous version easily, we can't do the changes easily.

.

How to push code from all the local branches?

Ans. git push –all

How to secure git repository? git best practices?

- Master branch should be protected from deletion.
- Require pull request reviews before merging.
- disable force push for all the people
- Require signed commits

How to migrate code from svn to git

Ans: subgit tool

- maintain all the log
- maintain user permission
- maintain reference url

https://docs.microsoft.com/en-us/azure/devops/repos/git/perform-migration-from-svn-to-git?view=azure-devops

https://docs.gitlab.com/ee/user/project/import/svn.htmls

```
Q.8. Role in Git???? https://docs.gitlab.com/ee/user/permissions.html
Ans: Owner role (devops team- owner of this repo)
(can perform any activiy such as branch creation/deletion/repository
creation/repo deletion etc)

     maintainer role (all the leads- dev lead/qa lead/ release manager/
sr folks / higher management---> can create branches/delete ,
can't create new repository)

     guest role (only read access)
     developer role (jr folk)
     Reporter role (higher management)
```

Diff b/w git clone and fork?

- when you fork a repository, you create a copy of the original repository (it creates a copy in user's github account) but repository remains in github.

- when you clone a repository the repository is copied on to your local system . (download code from remote repo to local repo)
- **A**s a devops, we need to push the files like… Jenkinsfile,dockerfile,kubernet,ansible playbook(conf mgmt),terraform (provisioning),tomcat
-
- Source code related files.
- Developer related files are-server and webapp.
- Q.5. How to protect master branch? (no one can delete the protected branches)
- Q 6. Define you git branch strategy?
- Q 7. How to secure git repository? best practices?
- Q. 1 diff b/w distributed & centralized version control?
- Q. 2 how to migrate code from svn to git? how to maintain history??
- Q.5. How to protect master branch? (noone can delete the protected branches)
- Q 6. Define you git branch strategy?
- Q 7. How to secure git repository? best practices?