DevOps UNIT-3

Introduction to Project Management

⇒ The need for Source Code Control
⇒ the history of Source Code management
⇒ Roles and Code
⇒ Source code management System and migrations
⇒ Shared authentication
⇒ Hosted Git Servers
⇒ Different Git Server implementations
⇒ Docker intermission
⇒ Gerrit Git Lab
⇒ the pull request model.

Need of Source Code Control

Source Code Control (also Known as Version Control) is a System that helps you to manage and Store your files, particularly the Code you write.

⇒ Git and Github are primary examples of tools used for Source Code Control.

Git :- Git is a Software tool that Can be used locally on your Computer. you Can Create a local repository on your Computer where you Store and manage your Code.

Git hub :- Github is a web based platform that allows you to Store your Git repositories in the cloud. multiple people Can work on the Same project, Submit changes, review Code, and manage projects.

⇒ An American author, Terence McKenna, once said that everything is Code. Whether you believe that or not, in DevOps, almost many parts of our work Can be described, managed and automated using Code. this includes:

* the application we Create
* the infra Structure that Supports our applications (like Servers & networks)
* the documentation that explains our products
* Even the Setup for the hardware that runs our applications.

Why we need Source Code Control :-

① **Central Storage :-**
All Code, Scripts and documents are kept in One place, making them easy to find and manage.

② **Teamwork :-**
Developers Can work together on the Same Code. the Source Code Control System keeps track of changes made by different people.

③ **Tracking changes :-**
Every change to the Code is Saved. If there is a problem, we Can go back to an earlier version of the Code.

④ **Backup :-**
the Source Code repository acts as a backup. If a developers Computer breaks, the Code is Still in the repository.

⑤ **History :-**
we Can See who made changes and when. this helps us understand why changes were made and find bugs.

⑥ **Consistency :-**
when everyone uses the Same repository, the Code Stays the Same for everyone.

(7) Deployment :-

Automated Systems can take code from the repository and deploy it to Servers. this makes testing and releasing new Versions easier.

⇒ Source Code Control is essential in DevOps because it Organizes and protects Our code. It helps with teamwork, keeps track of changes, ensures Consistency, without a Source code repository, managing code would be difficult and risky. Having a good Source Code Control System is Crucial for Success in DevOps.

The History of Source code Management

Understanding the history of Source Code Control can help us see why it is important today. Here are the main points:

Storing old Versions of Code :-

At first, people kept old versions of their code in separate files or archives. this method was simple and useful.

Centralized Source Code management with check-in and check out :-

⇒ In some systems, a developer could lock a file to work on it alone, so no one else could change it at the same time. Each file was managed separately.

⇒ Examples of these tools are RCS (Revision Control System) and SCCS (Source Code Control System). these tools are mostly outdated now.

Centralized Store with merging Before Committing :-

Tools like Concurrent Versions System (CVS) and Subversion (SVN) requires you to merge changes before you commit them.

Sub Version is still used a lot because it works well for many companies that have Centralized work flows.

Decentralized Source Code Management :-

This method allows for more flexibility and faster workflows. the most popular tool in this Category is Git, but there are others like Bazaar and Mercurial.

⇒ Decentralized tools are very powerful, but they can also be risky if not used carefully.

Key points :- * The most popular decentralized tool today is Git.

* Decentralized tools helps people work in better way and collaborate better.
* with more power and flexibility, there are also more chances to make mistakes.

⇒ Knowing the history of Source code management helps us understand why we need certain features today.

From Simple archives and folders to powerful decentralized tools, the evolution of these Systems shows how technology have changed overtime.

Roles and Code

In DevOps, Source Code management tools are super important because they help different team members work together smoothly.

Here's how various roles use these tools:

Developers:-

they use Source Code management every day to do their coding work.

Operations team:-

Operations team also depends on Source Code management. they use it to handle things like: Setting up networks and managing Software on Servers Using Code and Scripts.

Quality Assurance (QA):-

QA teams store their automated tests in the Source Code repository using tools like Selenium and JUnit.

Project management:-

Project managers might find it a bit tricky to use Source Code management tools Compared to technical teams.

Documentation challenges:

Sometimes, its hard to document manual steps for tasks.

Sometimes it's tough to write down the steps for doing tasks. Some companies use wikis Software for this, but lots still keep documents in word files or emails. this makes it hard for everyone to find and use them.

⇒ Source Code management tools are super important for many roles in DevOps. Developers, Operations team, and testing teams Depends on them for Smooth work. But project managers might find them a bit hard. its really important to make documentation better and put it all in one place So everyone can find it easily. Knowing how different jobs use Source Code tools helps Companies help ther teams and make work better

Source Code Management System and Migrations

There are many Source Code management (SCM) Systems, and they are essential for Software development. Among them, Git is the most popular tool today.

⇒ Git has an interesting history. It was created by Linus Torvalds because the Linux developers used a tool called BitKeeper to manage their code. BitKeeper was a good tool, but its license changed, which causes problem for the Linux developers. they couldn't use BitKeeper anymore without paying for it or agreeing to new restrictions. So, Linus Torvalds decided to create a new tool called Git. Git was designed to handle the complex work for developing main program of Linux and is now used by many organizations.

⇒ the main advantage of Git over older Systems is that it's a distributed version control System (DVCS). while there are other DVCS available, Git is the most widely used.

DVCS like Git have Several benefits:-

⇒ you can work offline

⇒ you don't need to connect to a server for every operations, making it faster.

⇒ you can work on your own changes privately until they are ready to share.

⇒ you can save multiple copies of your project in different locations. if one location fails, you still have other copies available. This reduces the risk of losing your data.

⇒ Git is open source and free to use, it has a large community, so resources are easy to find. Many popular code hosting services like Github, Gitlab and Bit bucket supports Git.

⇒ Other Distributed Version Control Systems (DVCS) includes Bazaar, Mercurial. Although git can be complex, it's very fast and efficient. if it's hard to understand, you can use graphical frontends to simplify its use.

Source Code management Systems migrations:-

⇒ After working with many source management systems, you may experience many transitions from one type of system to another

⇒ moving Data from one Source Code management System to another can be complicated or simple, depending on Systems (like Git, Subversion, Bazaar etc..)

⇒ For many projects keeping the history is not necessary, ~~these~~ older Systems can still be accessed if needed.

⇒ SCM Systems Examples: Git, Subversion (SVN), mercurial, preforce, Bazaar

⇒ Easy migration from Subversion to git. history remains accurate.

Shared Authentication

⇒ In many Organizations, there is a Central Server that handles authentication, which Verifies and allows access for users, One Common type of Central authentication Server is an LDAP (Lightweight Directory Access protocol) Server.

⇒ if your Organization already uses an LDAP Server to manage user authentication, that's great! if not, Setting up a test LDAP Server in your System for experimenting and testing is easy. you Can use Software like 389 Server and web application Called phpLDAPadmin to manage this test LDAP Server. This Setup allows you to know how LDAP Server works without affecting your main System.

⇒ Having a test LDAP Server is useful because if you have a practice Version of authentication Server. you Can See how authentication works, without effecting your main System. we Can use the Same for all the different Servers that we are going to navigate

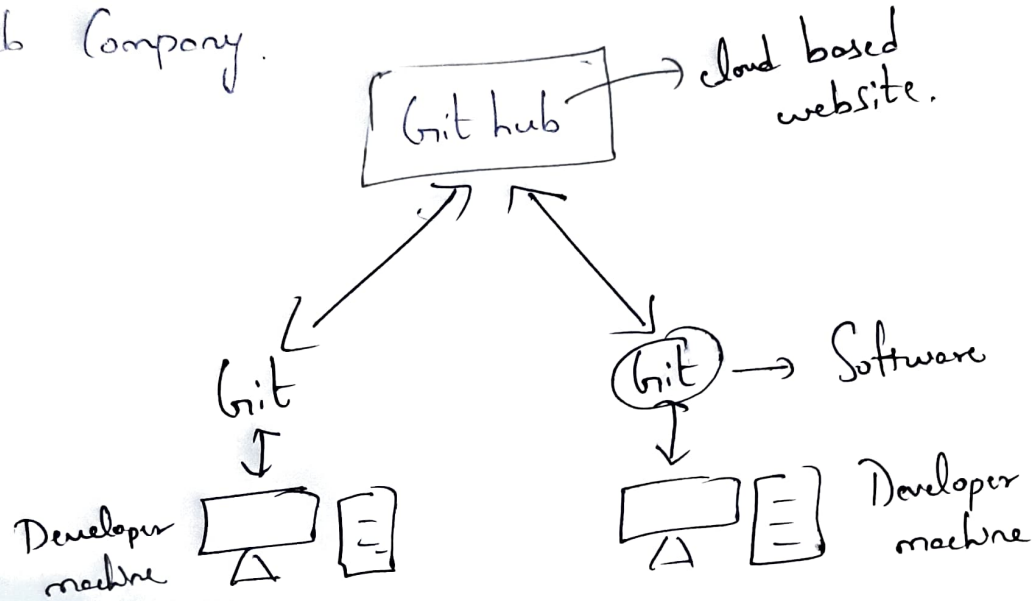
Hosted Git Servers

⇒ Many Organizations can't use hosted Git Services like Github or Gitlab. This is often because of legal reasons or concerns about Security. For example, government organizations or companies dealing with money, like banks and insurance companies, often need to keep their critical code within their own systems

⇒ However, if you don't have these concerns, using hosted Services like Github or Gitlab can be very convenient. Both offer free accounts so you can try them out and see what they offer. Github and Gitlab provides many useful features, such as:

- ⇒ web interfaces
- ⇒ Documentation wiki
- ⇒ issue tracker
- ⇒ Commit & branch visualization
- ⇒ pull request workflow

⇒ Git Software is maintained by Linus, GitHub is the product of microSoft and Gitlab is a product of Gitlab Company.



⇒ Git was Orginally designed for text files, not for large binary files like images, audio, video files. Although you can store these files in Git, it can make Git operations slow, this reduces Git's performance. So you can handle large binary files using Git hub and Git lab.

⇒ Git LFS and Git Annex. these two tools can handle large binary files.

Git LFS (Large File Storage) → Supported by Github

Git Annex → Supported by Gitlab (only in enterprise edition)

Gitlab:

⇒ Gitlab is a platform that helps Software teams manage their projects from Start to finish. It's like a tool box that includes everything that developers need:

(1) **Git Repositories**: Gitlab Stores and manages your project Code using Git, which tracks changes and multiple people work on Same Code without Conflicts

(2) **Continuous integration & Continuous Deployment (CI/CD)**:
Gitlab automates the process of testing and deployment your Code.

(3) **Code Reviews**:
Before new Code gets added to the main project, Gitlab allows ~~teams~~ members to review that Code.

(4) **issue tracking**:
GitLab helps keep track of tasks, bugs, features that are very important. everyone on the team Can See and work on important tasks.

(5) **wiki and Documentation**:
it provides a place to write and share project documentation.

(6) **Security**:- Gitlab includes features to make Sure your Code is Safe ~~and meets~~

(7) **integration with other tools**: Gitlab works well with other Software tools that developers use. like chat Systems, monitoring tools etc..

Different Git Server Implementations

⇒ Git is a System that allows you to work with multiple Copies of your project. these Copies Can be on different Servers like Github, Gitlab, bit bucket etc. to See which one works best for you. Each Server might have different features or benfits.

⇒ No matter which Server you choose, the way you use Git on your Computer will be the Same. you need to use Same Git Commands and tools

⇒ you Can use more than one Git Server at the Same time. for example, you Could have Some projects on Github and others on Gitlab. By using Git we Can Connect to different Servers easily.

Docker intermission :-

⇒ Docker is a tool that makes easy to package and run applications. Docker helps us test different Git Server implementations like Git hub, Git lab, bit bucket etc. without much Struggle.

⇒ As Docker packages your application into a Container. So your application will run Same way on different Systems like laptops, desktops, Virtual machines, physical Servers

⇒ Docker Containers run independently, so you can test different setups without interfering with others

installing Docker :-

to install docker in Red hat based systems use

```
dnf install docker-io
```

to enable and start Docker, use:

```
Systemctl enable docker
Systemctl start docker
```

⇒ we need another tool called "Docker Compose". to start multiple docker applications together.

Gerrit :-

⇒ Gerrit is a tool for reviewing and approving code changes before they are merged into the main codebase.

⇒ whenever any developer make changes in code, that code changes are reviewed by experienced developers before they are merged into the main codebase.

⇒ Gerrit is a platform where developers can submit their code changes for review. This ensures that only well reviewed and high quality code modifications are integrated into the project.

Benefits of Gerrit:

=) Control who can review and approve changes

=) Ensures that only reviewed and approved changes are merged

Running Gerrit with Docker:

Use following docker command to start Gerrit

```
docker run -d -p 8080:8080 -p 29418:29418 <gerrit image name>
```

- docker run: Starting the new container to pull image that contains Gerrit application
- -d: detached mode that runs container in background
- -p 8080:8080 -p 29418:29418: port no
- <gerrit image name>: image name that contains Gerrit application

Access Gerrit:-

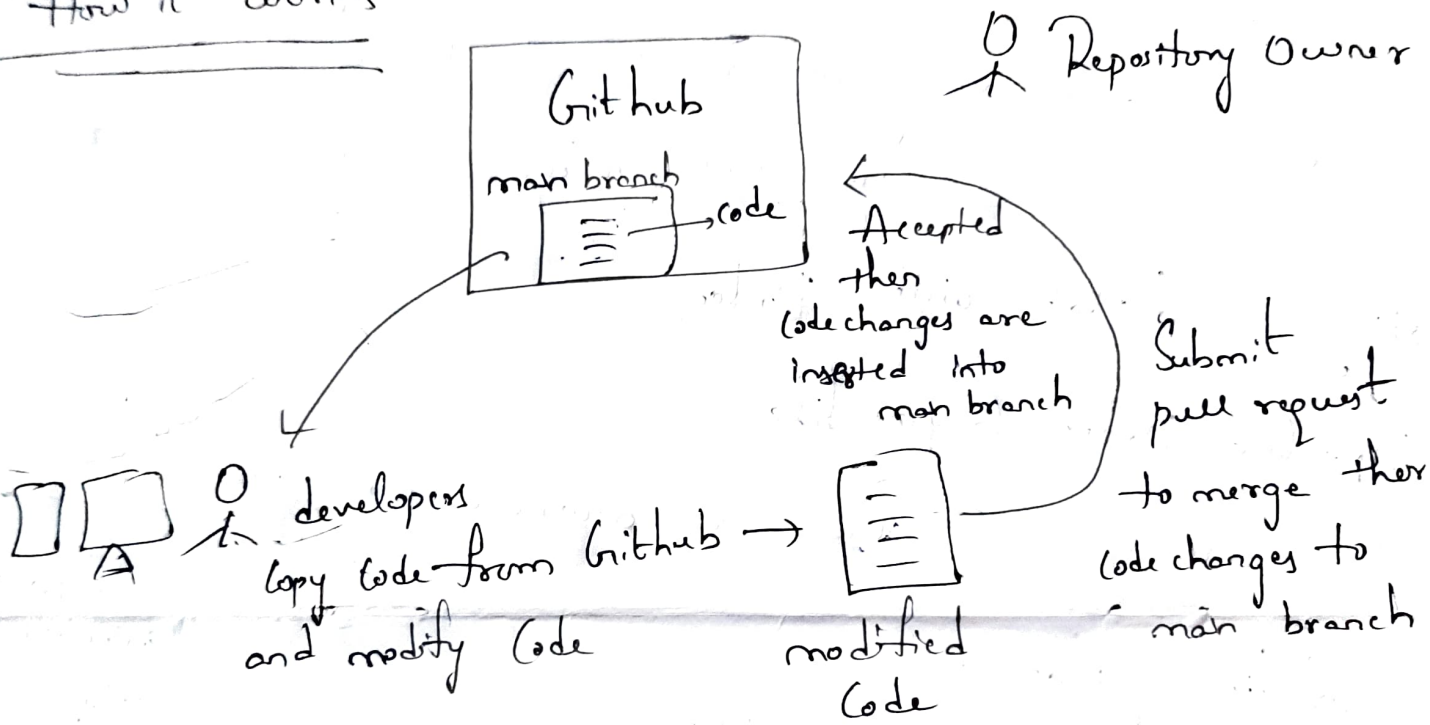
Open your browser and go to "http://<docker host url>:8080"

Pull Request model:

The pull request model is a way of managing code changes where developers make changes in their own copies of the repository and then request to merge those changes into the main repository.

=> it is simple process that is widely used, especially in Open-Source projects. Repository Owners can review and approve changes before they are merged.

How it works



Benefits of the pull Request model:

=> Encourages collaboration by allowing multiple people to work on the same project

=> Improves code Quality as changes are reviewed before being integrated.