Basic Git & GitHub for DevOps Engineers

Day 8 : 90Days of DevOps Challenge

SS Shivraj Salunkhe · Mar 31, 2023 · 5 min read

What is Git?

1. **Git** is a **distributed version control system** that allows multiple developers to work on the **same codebase** without conflicting changes. It enables users to create branches, commit changes, and merge code back into the **main branch**.
2. **With Git**, developers can create multiple branches of their codebase, work on different features or fixes in parallel, and merge changes back into the main branch when they are ready. Git also provides features for resolving conflicts that may arise when multiple developers make changes to the same file.
3. One of the key advantages of Git is that it is a **distributed version control system**, meaning that each developer has a complete copy of the **repository** on their own machine. This allows for offline work and easier collaboration between geographically distributed teams.
4. Git is widely used in software development, and many hosting services such as **GitHub**, **GitLab**, and **Bitbucket** provide Git hosting as a **service**.

What is GitHub?

1. **GitHub** is a web-based hosting service for **Git repositories** that provides additional features such as issue tracking, pull requests, and **continuous integration** and **deployment (CI/CD) pipelines**. It is a popular platform for **open-source** projects and collaboration between developers.
2. At its core, **GitHub** is a Git repository **hosting service**, meaning that it allows users to create and store Git repositories in the cloud. This makes it easy to share and **collaborate** on code with other developers, regardless of where they are located. GitHub provides a **web-based interface** for managing Git repositories, allowing users to view and compare changes, manage branches, and perform other Git-related tasks.
3. One of the key features of GitHub is **pull requests**, which enable developers to propose changes to a codebase and **solicit feedback** and reviews from other team members before merging the changes into the main branch. This helps ensure that code changes are thoroughly reviewed and tested before they are merged into the codebase.
4. GitHub also provides **integration** with various **third-party tools** and **services**, such as **CI/CD pipelines**, code analysis tools, and chat applications, making it easy to automate many aspects of the development process. Overall, GitHub is a powerful platform for software development and collaboration that has become an essential tool for many **developers** and **teams**.

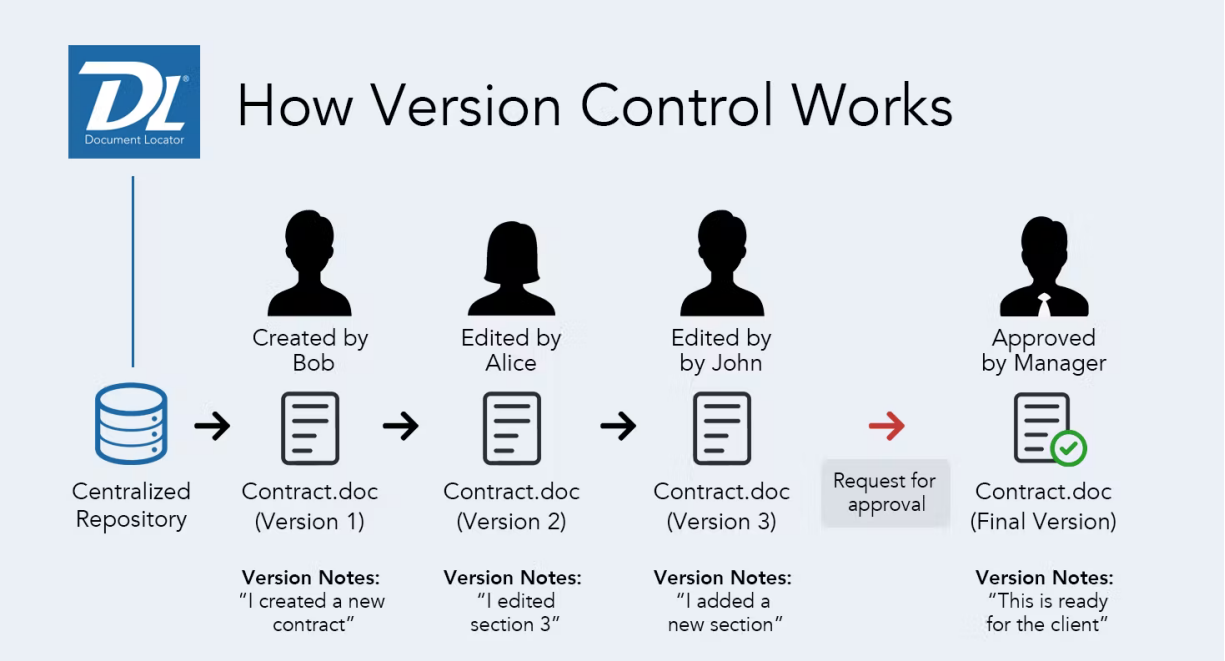
What is Version Control? How many types of version controls we have?

Version control is a software tool used to track changes to code and other files over time. It allows **developers** to keep track of different versions of their **code**, **collaborate** with others on the same codebase, and easily **revert** changes if necessary.

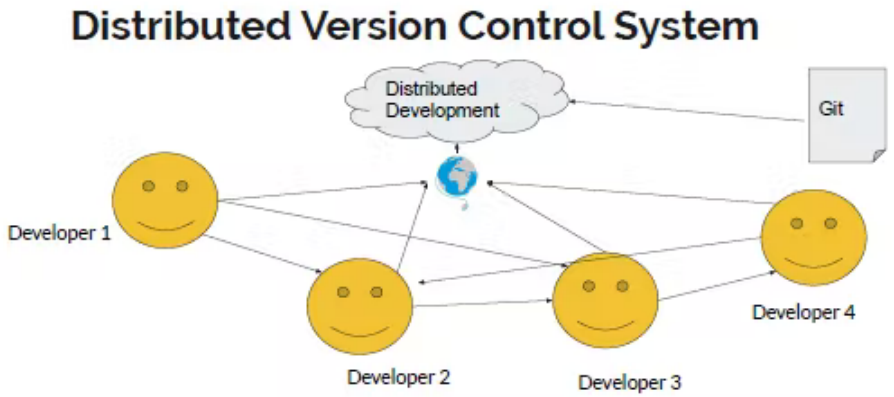
There are two main types of version control systems: centralized and distributed.

1. **Centralized version control systems (CVCS)** rely on a central repository where developers check out and check in code. Examples of CVCS include Apache

Subversion and Microsoft Team Foundation Server.



2. **Distributed version control systems (DVCS)** are based on a peer-to-peer model, where each developer has a copy of the repository on their local machine. Changes are made locally and can be synchronized with other developers' repositories. Examples of DVCS include Git and Mercurial.



In addition to these main types, there are also other version control systems with different **features** and **capabilities**. For example, some systems focus on tracking changes to binary files or managing large datasets, while others are designed for specific **programming languages** or **platforms**.

Regardless of the type of version control system used, the key benefits of version control include **better collaboration**, more efficient **development workflows**, and easier **tracking** of changes to code over time.

Why we use distributed version control over centralized version control?

Distributed version control systems (DVCS) like Git have become more popular than **centralized version control systems (CVCS)** like Subversion in recent years because of several advantages they offer:

1. **Offline work:** With DVCS, developers can work offline and commit changes to their local repository without the need for a centralized server. This allows them to work in environments where internet connectivity is limited or nonexistent.
2. **Branching and merging:** DVCS allows developers to easily create multiple branches of their codebase, make changes in parallel, and merge them back together. This makes it easier to manage complex codebases and work collaboratively with other developers.
3. **Faster performance:** DVCS tends to be faster than CVCS because most operations are done locally on the developer's machine. This eliminates the need for constant communication with a central server, which can slow down performance.
4. **Better security:** With DVCS, each developer has a complete copy of the repository, so there's no single point of failure or vulnerability. This makes it harder for malicious actors to compromise the codebase.

Overall, DVCS is more flexible, efficient, and secure than CVCS, which is why it's become the preferred choice for many developers and organizations.

Install Git on your computer :

steps to install Git on your computer:

1. Go to the Git website at git-scm.com/downloads.

2. Click the download link for your operating system (e.g. Windows, macOS, or Linux).
3. Follow the installation instructions provided for your operating system.
4. Once the installation is complete, open a terminal or command prompt window and run the command **"git --version"** to verify that Git is installed and working properly. This should display the version number of Git that you installed.

That's it! You should now be able to use Git on your computer.

-> Create a new repository on GitHub and clone it to your local machine.

- Click on **New** to create a new repository.

- Copy the repository URL by clicking on the green **"code"** button on the repository you want to clone. Copy the **URL.**

- Once the repository is created you can **clone** it to your local machine.

```
git clone repository URL
```

-> Make some changes to a file in the repository and commit them to the repository using Git.

- Make some changes to a file using a text editor.
- Run the below command to stage the changes.

```
git add <file-name>
```

COPY

Replace the **<file-name>** with the name of the file you changed.

- **Commit** the changes by running the command

```
git commit -m "Commit message"
```

COPY

Replace **"Commit message"** with a brief description of the change you made.

-> Push the changes back to the repository on GitHub.

- **Push** the back to the repository on GitHub by running the following command.

```
git push
```

COPY

These commands will stage your changes, **Commit** them to your local repository and **push** them to the **remote repository** on GitHub.

Thank You for Reading the Blog.. 👍

connect with me : linkedin.com/in/shivraj-salunkhe-5881141a4

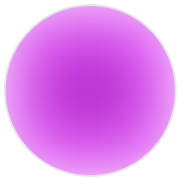
follow my blog channel : **shivrajofficial.hashnode.dev**

Subscribe to my newsletter

Read articles from directly inside your inbox.
Subscribe to the newsletter, and don't miss out.

Enter your email address SUBSCRIBE

Devops GitHub Git version control Linux



WRITTEN BY

Shivraj Salunkhe

Follow

Dedicated and hardworking undergraduate student pursuing a degree in Information Technology with a passion for learning, leadership experience, and real-world skills through internships and part-time jobs in IT sector. Actively looking for new Opportunities and committed to do personal and professional growth.

MORE ARTICLES

Shivraj Salunkhe

Understanding package manager and systemctl

What is a package manager in Linux? In Linux, a package manager is a software tool that helps users...

Shivraj Salunkhe

File Permissions and Access Control Lists

In Linux, file permissions refer to the settings that determine who can read, write, or execute a fi...

Shivraj Salunkhe

Advanced Linux Shell Scripting for DevOps Engineers with User management

Advanced Linux Shell Scripting for DevOps Engineers covers a wide range of topics, including : Auto...