INSTALASI
G I T

A. What is Git?

By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments).

Having a distributed architecture, Git is an example of a DVCS (hence Distributed Version Control System). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like CVS or Subversion (also known as SVN), in Git, every developer's working copy of the code is also a repository that can contain the full history of all changes.

In addition to being distributed, Git has been designed with performance, security and flexibility in mind.

B. Performance

The raw performance characteristics of Git are very strong when compared to many alternatives. Committing new changes, branching, merging and comparing past versions are all optimized for performance. The algorithms implemented inside Git take advantage of deep knowledge about common attributes of real source code file trees, how they are usually modified over time and what the access patterns are.

Unlike some version control software, Git is not fooled by the names of the files when determining what the storage and version history of the file tree should be, instead, Git focuses on the file content itself. After all, source code files are frequently renamed, split, and rearranged. The object format of Git's repository files uses a combination of delta encoding (storing content differences), compression and explicitly stores directory contents and version metadata objects.

Being distributed enables significant performance benefits as well.

For example, say a developer, Alice, makes changes to source code, adding a feature for the upcoming 2.0 release, then commits those changes with descriptive messages. She then works on a second feature and commits those changes too. Naturally these are stored as separate pieces of work in the version history. Alice then switches to the version 1.3 branch of the same software to fix a bug that affects only that older version. The purpose of this is to enable Alice's team to ship a bug fix release, version 1.3.1, before version 2.0 is ready. Alice can then return to the 2.0 branch to continue working on new features for 2.0 and all of this can occur without any network access and is therefore fast and reliable. She could even do it on an airplane. When she is ready to send all of the individually committed changes to the remote repository, Alice can "push" them in one command.

C. Security

Git has been designed with the integrity of managed source code as a top priority. The content of the files as well as the true relationships between files and directories, versions, tags and commits, all of these objects in the Git repository are secured with a cryptographically secure hashing algorithm called SHA1. This protects the code and the change history against both accidental and malicious change and ensures that the history is fully traceable.

With Git, you can be sure you have an authentic content history of your source code.

Some other version control systems have no protections against secret alteration at a later date. This can be a serious information security vulnerability for any organization that relies on software development.

D. Flexibility

One of Git's key design objectives is flexibility. Git is flexible in several respects: in support for various kinds of nonlinear development workflows, in its efficiency in both small and large projects and in its compatibility with many existing systems and protocols.

Git has been designed to support branching and tagging as first-class citizens (unlike SVN) and operations that affect branches and tags (such as merging or reverting) are also stored as part of the change history. Not all version control systems feature this level of tracking.

E. Criticism of Git

One common criticism of Git is that it can be difficult to learn. Some of the terminology in Git will be novel to newcomers and for users of other systems, the Git terminology may be different, for example, revert in Git has a different meaning than in SVN or CVS. Nevertheless, Git is very capable and provides a lot of power to its users. Learning to use that power can take some time, however once it has been learned, that power can be used by the team to increase their development speed.

For those teams coming from a non-distributed VCS, having a central repository may seem like a good thing that they don't want to lose. However, while Git has been designed as a distributed version control system (DVCS), with Git, you can still have an official, canonical repository where all changes to the software must be stored. With Git, because each developer's repository is complete, their work doesn't need to be constrained by the availability and performance of the "central" server. During outages or while offline, developers can still consult the full project history. Because Git is flexible as well as being distributed, you can work the way you are accustomed to but gain the additional benefits of Git, some of which you may not even realise you're missing.

Now that you understand what version control is, what Git is and why software teams should use it, read on to discover the benefits Git can provide across the whole organization.

F. Installing Git

- **Installing from Source**

  Some people may instead find it useful to install Git from source, because you'll get the most recent version. The binary installers tend to be a bit behind, though as Git has matured in recent years, this has made less of a difference.

  If you do want to install Git from source, you need to have the following libraries that Git depends on: autotools, curl, zlib, openssl, expat, and libiconv. For example, if you're on a system that has dnf (such as Fedora) or apt-get (such as a Debian-based system), you can use one of these commands to install the minimal dependencies for compiling and installing the Git binaries:

```
$ sudo dnf install dh-autoreconf curl-devel expat-devel gettext-devel
  openssl-devel perl-devel zlib-devel
$ sudo apt-get install dh-autoreconf libcurl4-gnutls-dev libexpat1-dev
  gettext libz-dev libssl-dev
```

In order to be able to add the documentation in various formats (doc, html, info), these additional dependencies are required:

```
$ sudo dnf install asciidoc xmlto docbook2X
$ sudo apt-get install asciidoc xmlto docbook2x
```

If you're using a Debian-based distribution (Debian/Ubuntu/Ubuntu-derivatives), you also need the install-info package:

```
$ sudo apt-get install install-info
```

When you have all the necessary dependencies, you can go ahead and grab the latest tagged release tarball from several places. You can get it via the kernel.org site, at https://www.kernel.org/pub/software/scm/git, or the mirror on the GitHub website, at https://github.com/git/git/tags. It's generally a little clearer what the latest version is on the GitHub page, but the kernel.org page also has release signatures if you want to verify your download. Then, compile and install:

```
$ tar -zxf git-2.8.0.tar.gz
$ cd git-2.8.0
$ make configure
$ ./configure --prefix=/usr/local
$ make all doc info
$ sudo make install install-doc install-html install-info
```

- **Installing From Packages**

The option of installing with default packages is best if you want to get up and running quickly with Git, if you prefer a widely-used stable version, or if you are not looking for the newest available functionalities. If you are looking for the most recent release, you should jump to the section on installing from source.

Git is likely already installed in your Ubuntu 20.04 server. You can confirm this is the case on your server with the following command:

```
$ git --version
```

If you receive output similar to the following, then Git is already installed.

```
Output
git version 2.25.1
```

If this is the case for you, then you can move onto setting up Git, or you can read the next section on how to install from source if you need a more up-to-

date version. However, if you did not get output of a Git version number, you can install it with the Ubuntu default package manager APT. First, use the apt package management tools to update your local package index.

```
$ sudo apt update
```

With the update complete, you can install Git:

```
$ sudo apt install git
```

You can confirm that you have installed Git correctly by running the following command and checking that you receive relevant output.

```
$ git --version
```

```
Output
$ git version 2.25.1
```

- **Setting Up Git**

After you are satisfied with your Git version, you should configure Git so that the generated commit messages you make will contain your correct information and support you as you build your software project.

Configuration can be achieved by using the git config command. Specifically, we need to provide our name and email address because Git embeds this information into each commit we do. We can go ahead and add this information by typing:

```
$ git config --global user.name "Your Name"
$ git config --global user.email "youremail@domain.com"
```

We can display all of the configuration items that have been set by typing:

```
$ git config --list
```

The information you enter is stored in your Git configuration file, which you can optionally edit by hand with a text editor of your choice like this (we'll use vim or nano):

```
$ vim ~/.gitconfig
```

Reference

- https://www.atlassian.com/git/tutorials/what-is-git
- https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu-20-04
- https://www.digitalocean.com/community/tutorials/how-to-install-git-from-source-on-ubuntu-20-04-quickstart
- https://git-scm.com/book/en/v2/Getting-Started-Installing-Git