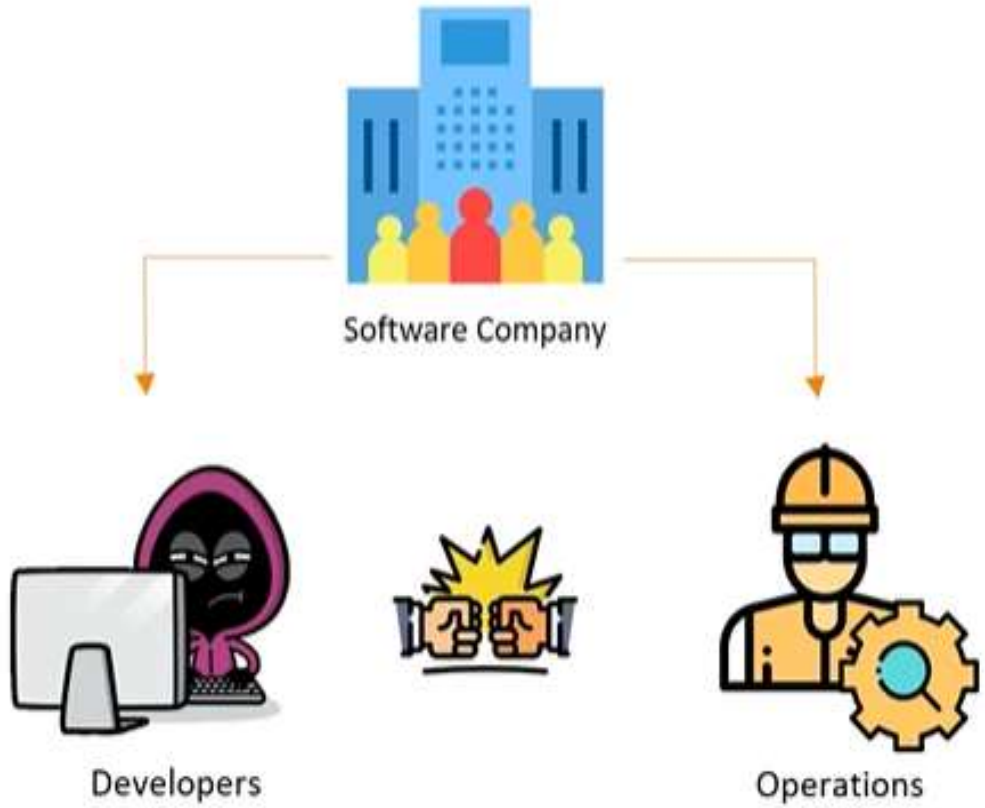
DEVOPS LECTURE

Waterfall vs Agile vs Lean

Summarizing

Problem with Waterfall Model was, the development lifecycle took a lot of time to complete. Therefore, by the time finished product was delivered, the customer requirements were no longer the same.

Why DevOps?



Although, the software quality was improved. We still had a lack of efficiency among the development team. A typical software development team consists of Developers and Operations employees. Let us understand their job roles

Why DevOps?

A developer's job is to develop applications and pass his code to the operations team

Developer

The operations team job is to test the code, and provide feedback to developers in case of bugs. If all goes well, the operations team uploads the code to the build servers

Operations

Why DevOps?

Developer

But, the code runs fine on the developer's system and hence he says "It is not my fault!"

Operations

The operations then marked this code as faulty, and used to forward this feedback to the developer

Developer

Operations

This led to a lot of back and forth between the developer and the operations team, hence impacted efficiency.

Why DevOps?

Developer

Operations

This problem was solved using Devops!

Traditional IT vs DevOps

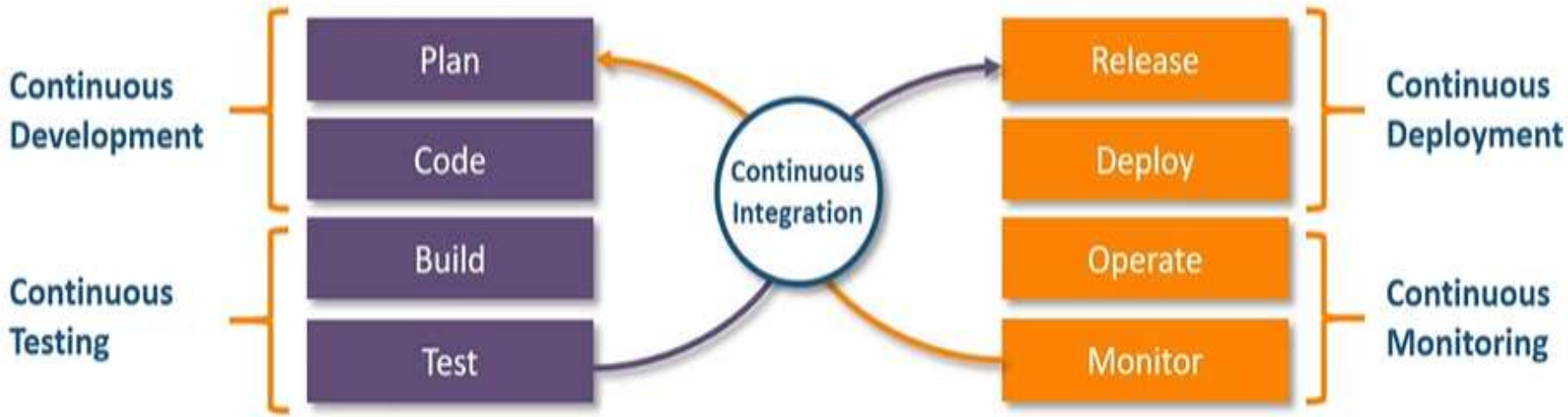
Traditional IT	Devops
Less Productive	More Productive
Skill Centric Team	Team is divided into specialized silos
More Time invested in planning	Smaller and Frequent releases lead to easy scheduling and less time in planning
Difficult to achieve target or goal	Frequent releases, with continuous feedback makes achieving targets easy

What is DevOps?

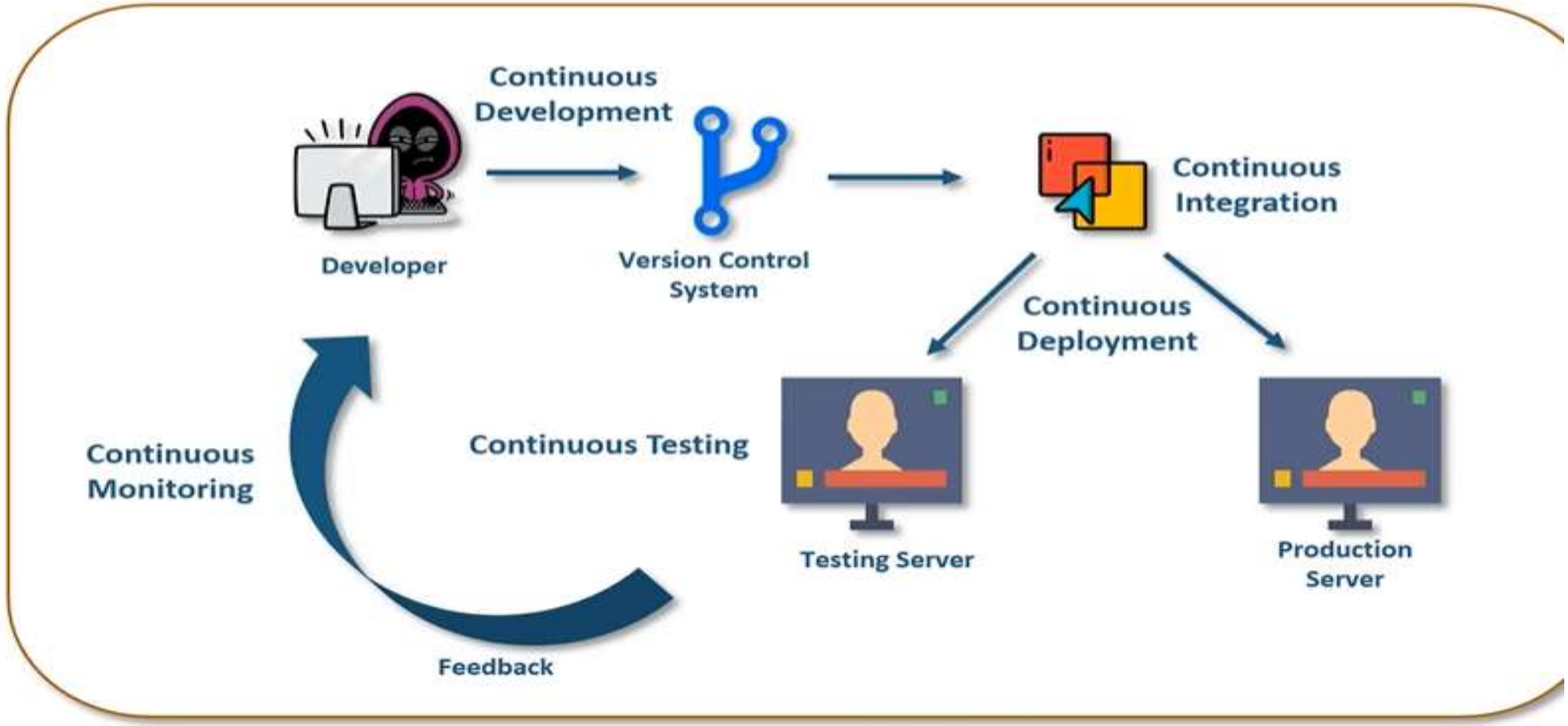
Devops is a software development methodology which improves the collaboration between developers and operations team using various automation tools. These automation tools are implemented using various stages which are a part of the Devops Lifecycle

How DevOps Works?

The Devops Lifecycle divides the SDLC lifecycle into the following stages:



How DevOps Works?



Automated CI/CD Pipeline

How DevOps Works?

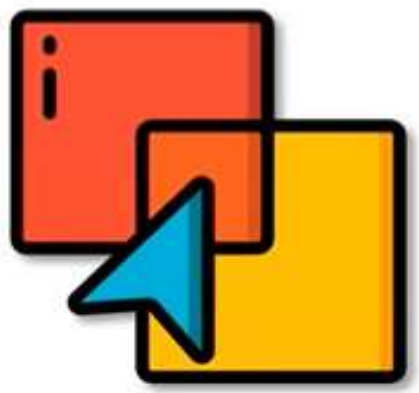
- **Continuous Development**
- Continuous Integration
- Continuous Deployment
- Continuous Testing
- Continuous Monitoring

This stage involves committing code to version control tools such as **Git** or **SVN** for maintaining the different versions of the code, and tools like **Ant**, **Maven**, **Gradle** for building/ packaging the code into an executable file that can be forwarded to the QAs for testing.

How DevOps Works?

- Continuous Development
- Continuous Integration
- Continuous Deployment
- Continuous Testing
- Continuous Monitoring

The stage is a critical point in the whole Devops Lifecycle. It deals with integrating the different stages of the devops lifecycle, and is therefore the key in automating the whole Devops Process



Continuous Development

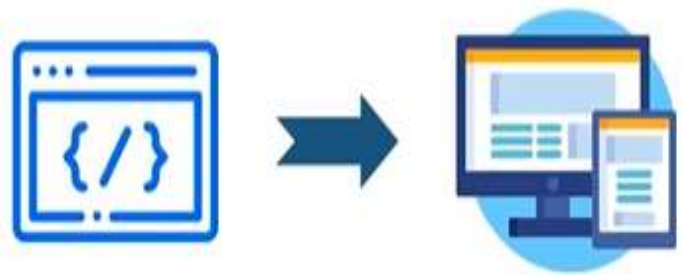
Continuous Integration

Continuous Deployment

Continuous Testing

Continuous Monitoring

In this stage the code is built, the environment or the application is containerized and is pushed on to the desired server. The key processes in this stage are Configuration Management, Virtualization and Containerization



Continuous Development

Continuous Integration

Continuous Deployment

Continuous Testing

Continuous Monitoring

The stage deals with automated testing of the application pushed by the developer. If there is an error, the message is sent back to the integration tool, this tool in turn notifies the developer of the error. If the test was a success, the message is sent to Integration tool which pushes the build on the production server

Continuous Development

Continuous Integration

Continuous Deployment

Continuous Testing

Continuous Monitoring

The stage continuously monitors the deployed application for bugs or crashes. It can also be setup to collect user feedback. The collected data is then sent to the developers to improve the application

DevOps Tools

We have discussed the Devops Methodology, but this methodology cannot be put into action without it's corresponding tools. Let us discuss the devops tools with their respective lifecycle stages

DevOps Tools

- Continuous Development
- Continuous Integration
- Continuous Deployment
- Continuous Testing
- Continuous Monitoring

Git is a distributed version-control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source-code management in software development, but it can be used to keep track of changes in any set of files

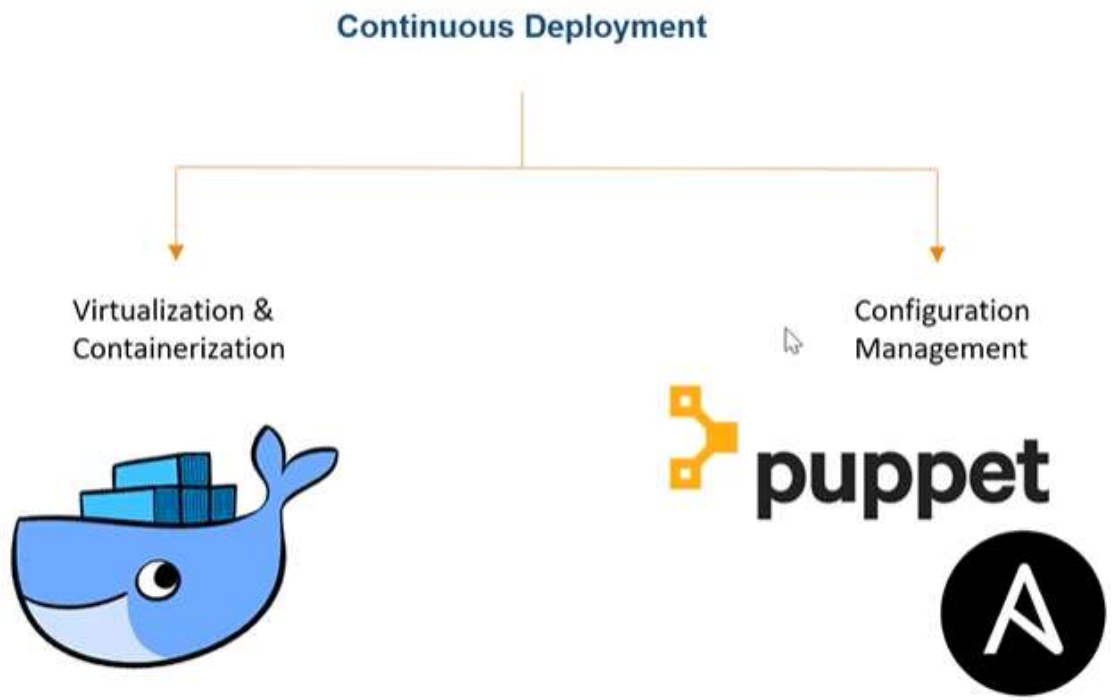
DevOps Tools

- Continuous Development
- Continuous Integration
- Continuous Deployment
- Continuous Testing
- Continuous Monitoring

Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery

DevOps Tools

- Continuous Development
- Continuous Integration
- Continuous Deployment
- Continuous Testing
- Continuous Monitoring



DevOps Tools

- Continuous Development
- Continuous Integration
- Continuous Deployment
- **Continuous Testing**
- Continuous Monitoring

Selenium is a portable software-testing framework used for web applications. It is an open source tool which is used for automating the tests carried out on web browsers (Web applications are tested using any web browser).

Continuous Development

Continuous Integration

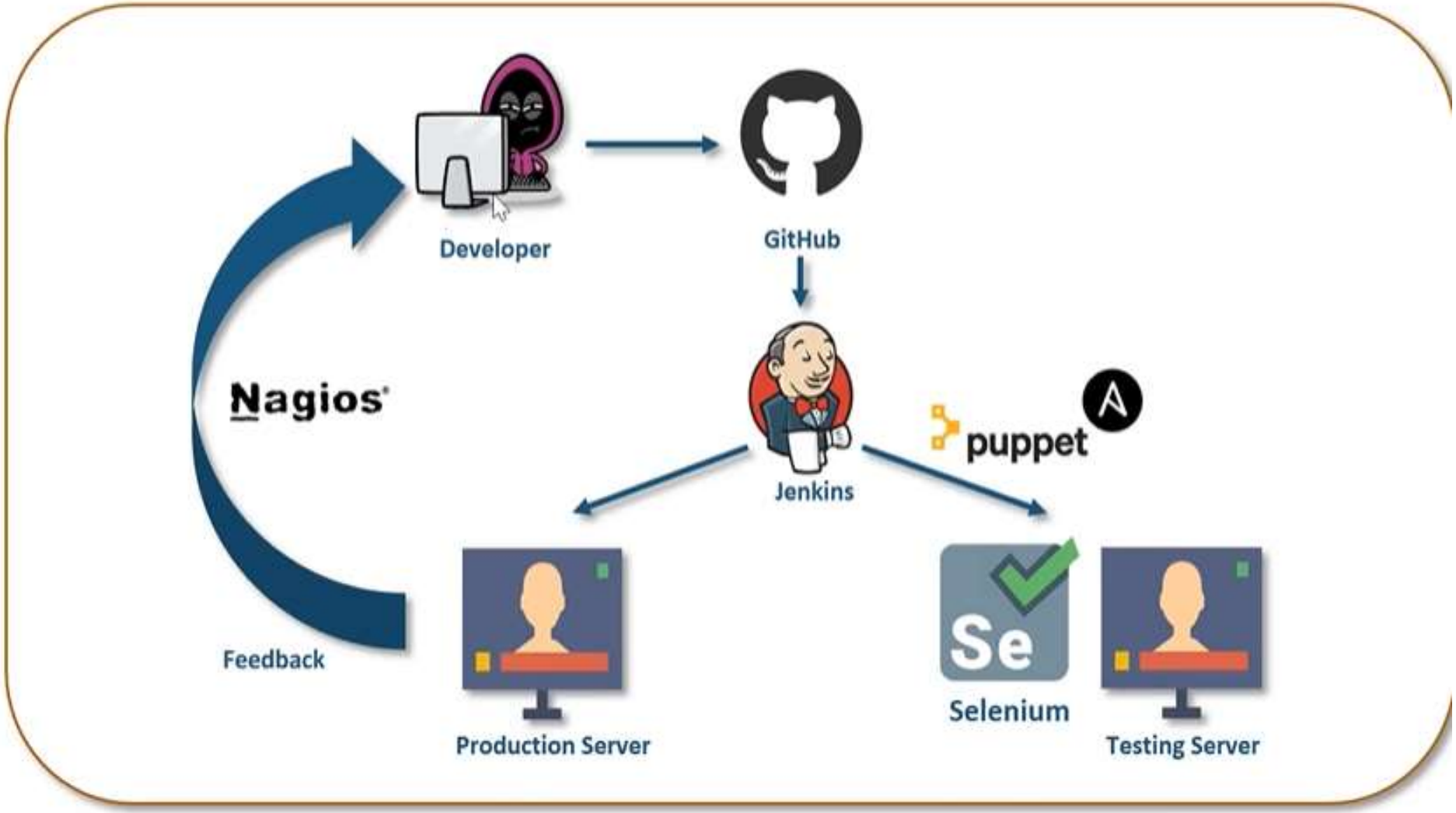
Continuous Deployment

Continuous Testing

Continuous Monitoring

Nagios is an open-source devops tool which is used for monitoring systems, networks and infrastructure. It also offers monitoring and alerting services for any configurable event.

DevOps Tools



c

1. Which of these Software Development Methodologies are not suitable for large and complex projects?

A. Waterfall Model

B. Devops

C. Agile Methodology

D. None of these

a

2. Devops Methodology was focused on solving the problems between the customers and the software company.

A. True

B. False

c

3. Which of these principles are NOT included in Agile Methodologies?

A. Frequent Release Cycles

B. Focus on Customer Feedback

C. Eliminating Waste

D. None of these

d

4. Which Lifecycle stage in Devops helps in Transition from one stage to another?

A. Continuous Development

B. Continuous Testing

C. Continuous Monitoring

D. Continuous Integration

5. Which tool among the following helps in containerization?

A. Jenkins

B. Git

C. Kubernetes

D. Docker

Version Control with GIT

Agenda

01 WHAT IS VERSION CONTROL?

02 TYPES OF VERSION CONTROL SYSTEM

03 INTRODUCTION TO GIT

04 GIT LIFECYCLE

05 COMMON GIT COMMANDS

06 MERGING IN GIT

07 RESOLVING MERGE CONFLICTS

08 GIT WORKFLOW

What is Version Control?

Version control is a system that records/manages changes to documents, computer programs etc over time. It helps us tracking changes when multiple people work on the same project

Problems before Version Control

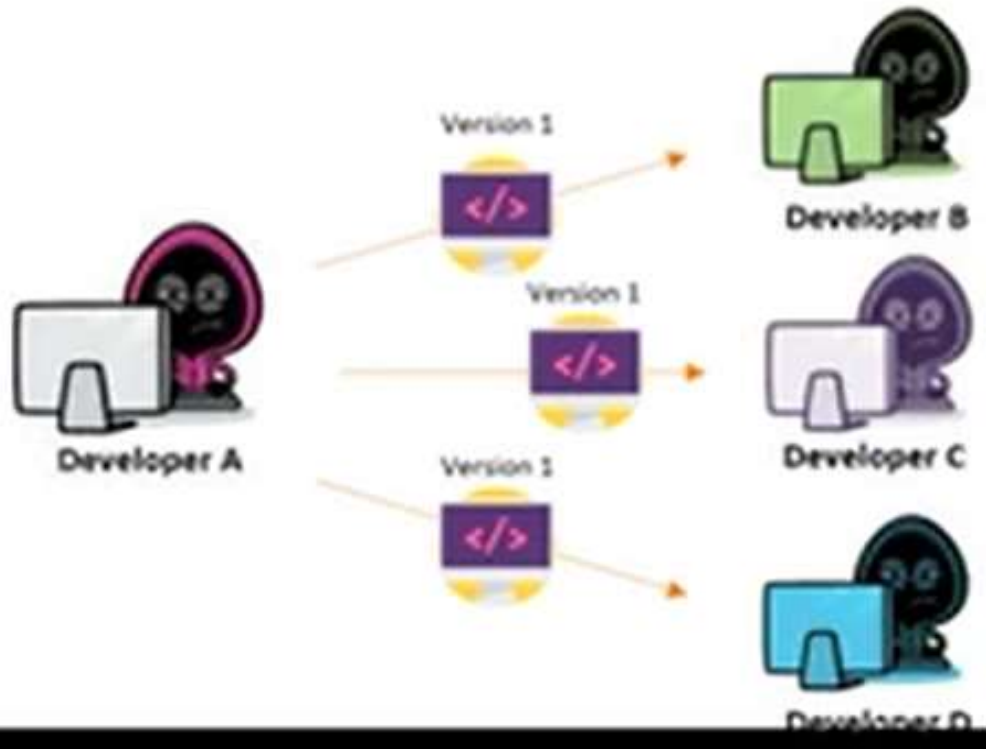
Imagine, Developer A creates a software, and starts a company with this software.

Developer A

Software

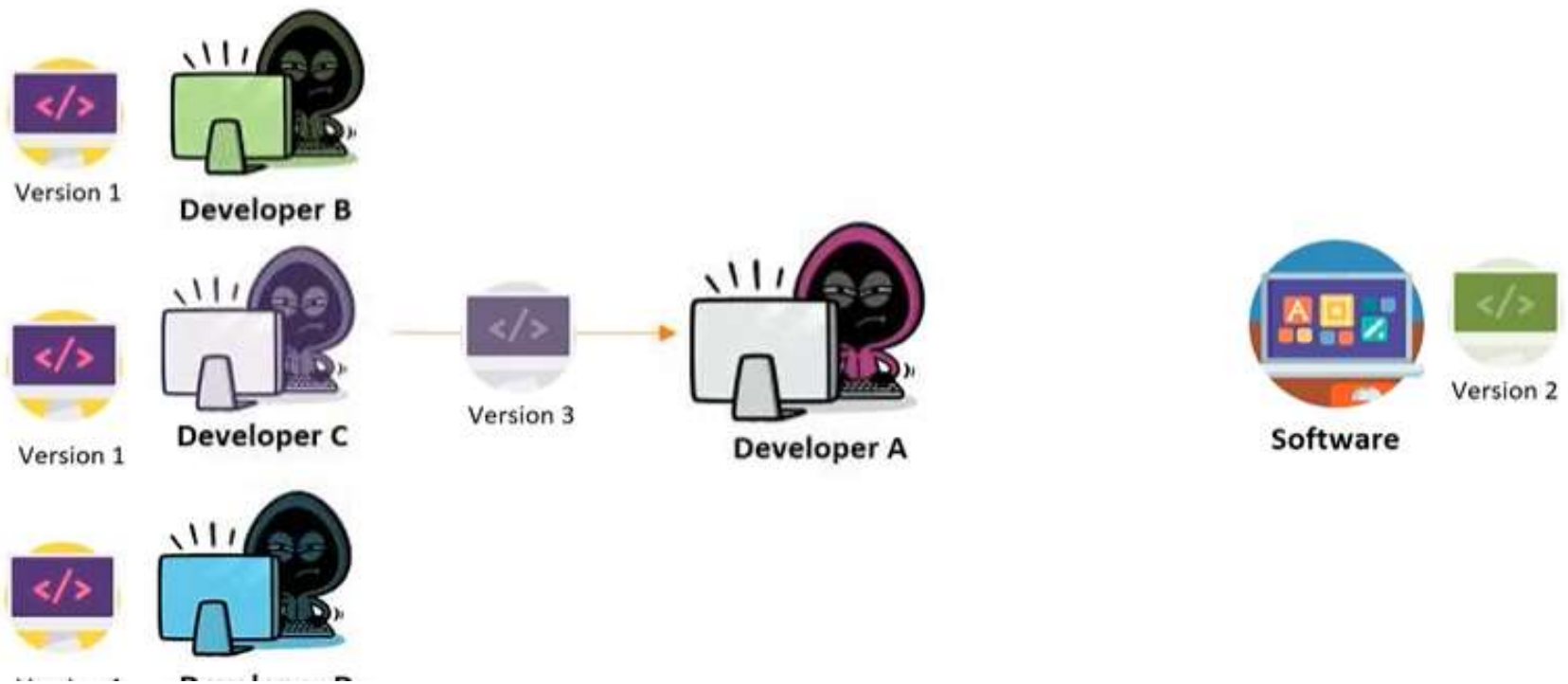
Problems before Version Control

As the company grows, Developer A hires more people to enhance the features of this software. Developer A shares the source code copy with each one of them to work on

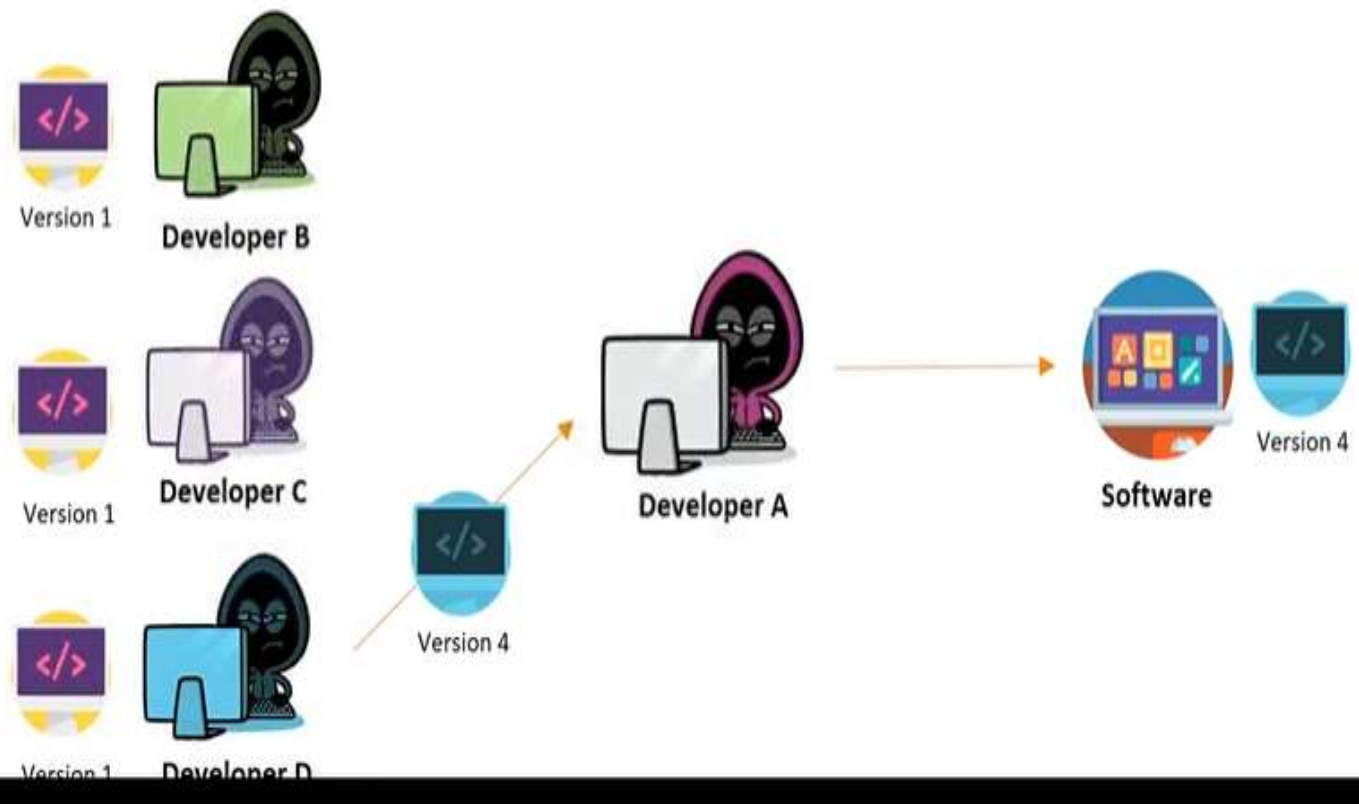


Problems before Version Control

Now, the problem starts here, Developer C also finished his work, and submits the changes to Developer A. But, Developer C worked on the code of Version 1.



Similarly when Developer C is done with his work, submits the work to Developer A.
Developer A verifies it, manually integrates the changes with Version 3



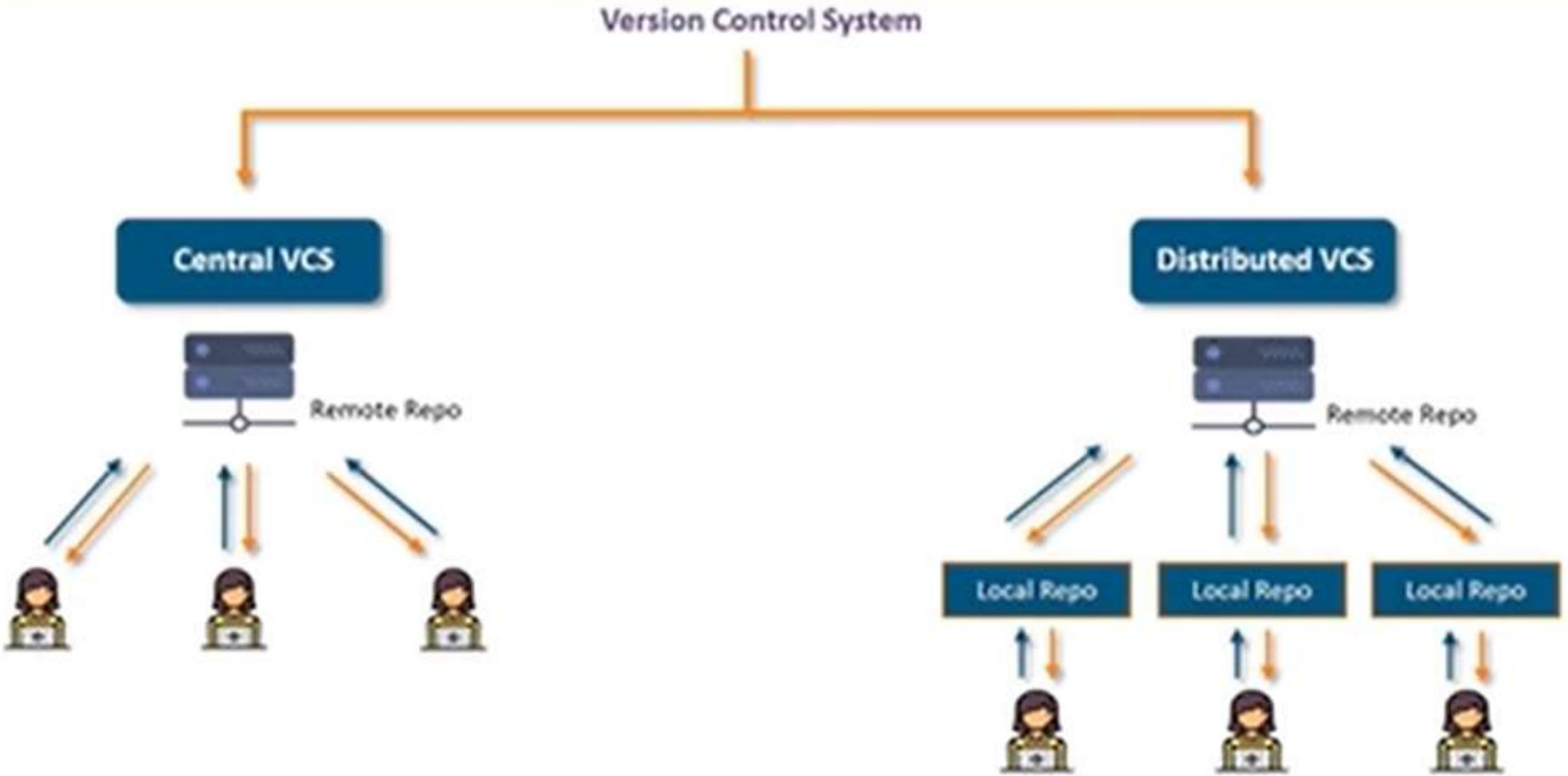
Problems before Version Control

- Versioning was Manual
- Team Collaboration was a time consuming and hectic task
- No easy access to previous versions
- Multiple Version took a lot of space

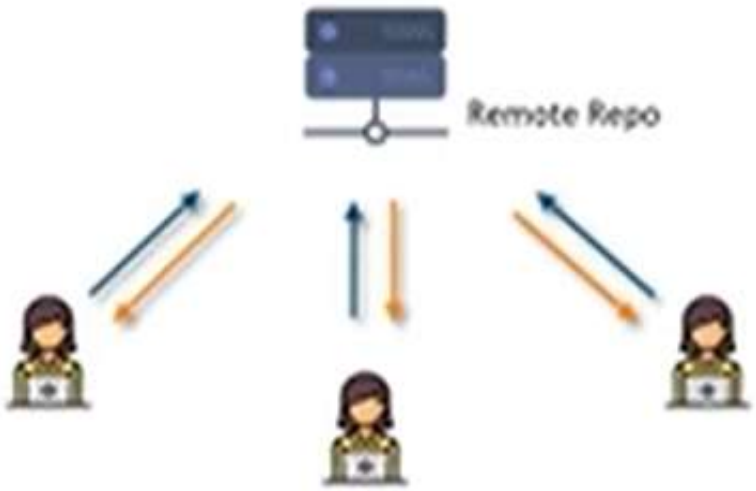
Advantages of Version Control

- Versioning is Automatic
- Team Collaboration is simple
- Easy Access to previous Versions
- Only modified code is stored across different versions, hence saves storage

Types of Version Control System



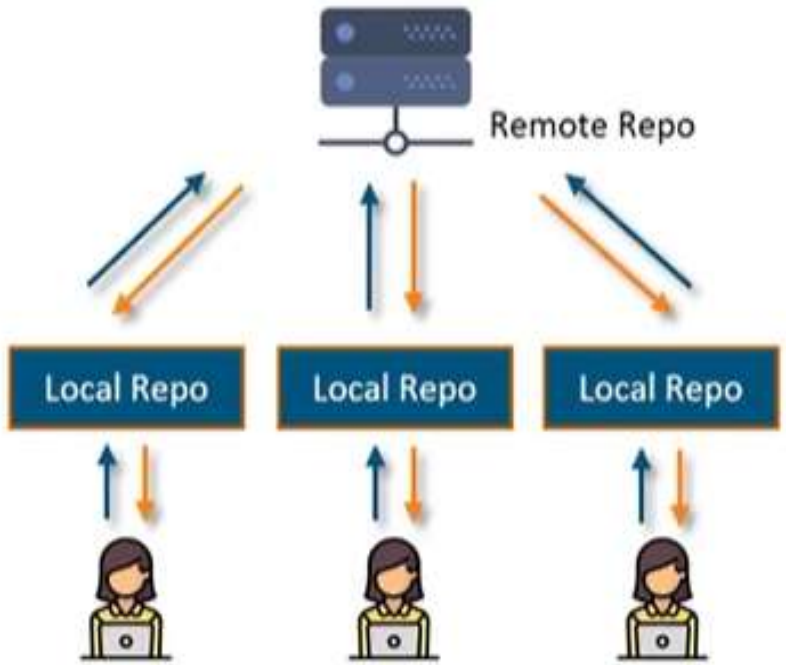
Centralized Version Control System



Centralized VCS

- Centralized Version Control System has one single copy of code in the central server
- Developers will have to "commit" their changes in the code to this central server
- "Committing" a change simply means recording the change in the central system

Distributed Version Control System



Distributed VCS

- In Distributed VCS, one does not necessarily rely on a central server to store all the versions of a project's file
- Every developer "clones" a copy of the main repository on their local system
- This also copies, all the past versions of the code on the local system too
- Therefore, the developer need not be connected to the internet to work on the code

Difference between DVCS and CVCS

Distributed VCS

- Everything except pushing and pulling can be done without Internet Connection
- Every Developer has full version history on local hard drive
- Committing and retrieving action is faster since data is on local drive
- Not Good for storing large files which are binary in nature, this would increase the repo size at every commit
- If a project has a lot of commits, downloading them may take a lot of time

Centralized VCS

- Needs a dedicated internet connection for every operation
- Developers just have the working copy and no version history on their local drive
- Committing and retrieving action is slower since it happens on the internet
- Good for storing large files, since version history is not downloaded
- Not dependent on the number of commits

Examples of CVCS

What is SVN?

- Apache Subversion is a software versioning and revision control system distributed as open source under the Apache License
- It is based on Centralized Version Control Architecture
- The development started in 2000, and this version finally became available in 2004
- It is still constantly being developed by a small but active open source community

Disadvantages of SVN

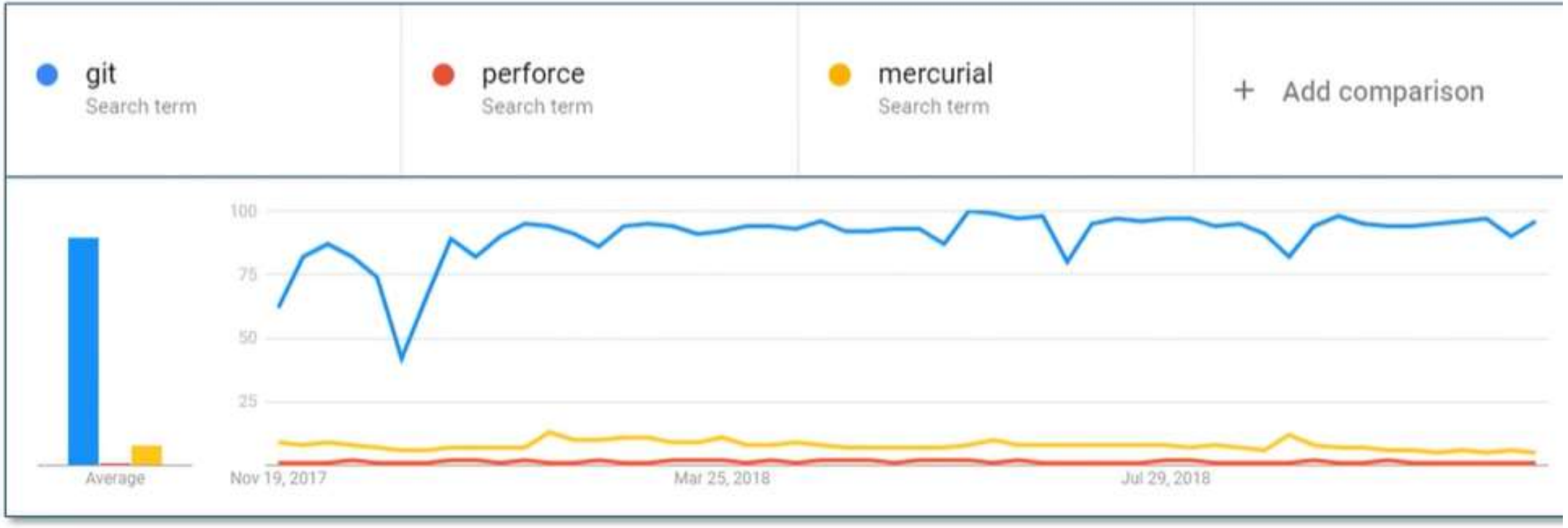
- Constantly needs an Internet Connection for any operation
- Version History is not downloaded or maintained on the local system
- Slower than DVCS, since requires internet for every operation
- Conflicts have to be resolved manually

Examples of DVCS

PERFORCE

Why Git?

Git is the most popular tool among all the DVCS tools.



What is Git?

Git is a version-control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source-code management in software development, but it can be used to keep track of changes in any set of files.

Following are the lifecycle stages of files in Git

Working Directory

Staging Area

Commit

Git Lifecycle

- The place where your project resides in your local disk
- This project may or may not be tracked by git
- In either case, the directory is called the working directory
- The project can be tracked by git, by using the command *git init*
- By doing *git init*, it automatically creates a hidden .git folder

Git Lifecycle

Working Directory

Staging Area

Commit

- Once we are in the working directory, we have to specify which files are to be tracked by git
- We do not specify all files to be tracked in git, because some files could be temporary data which is being generated while execution
- To add files in the staging area, we use the command *git add*

Git Lifecycle

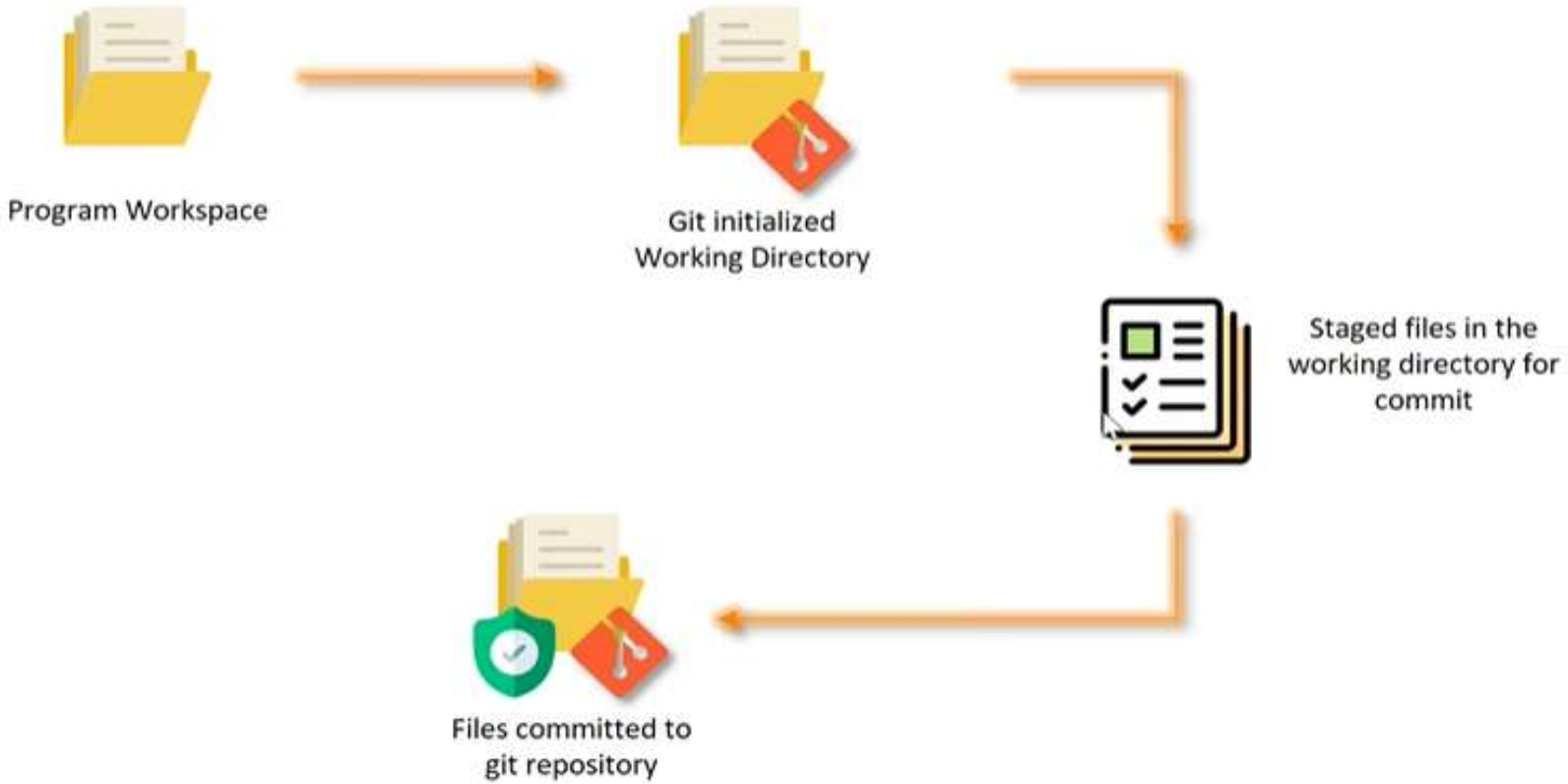
Working Directory

Staging Area

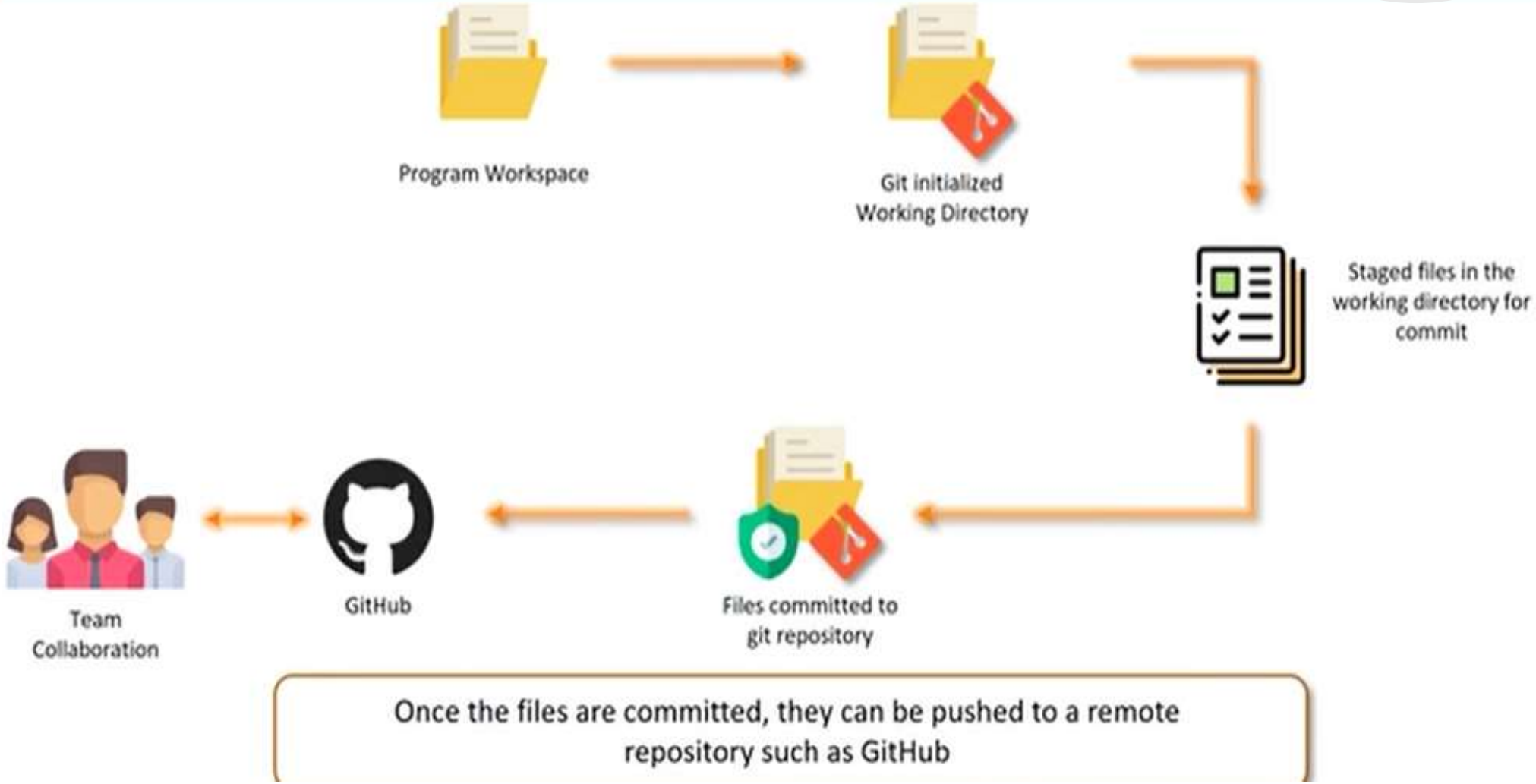
Commit

- Once the files are selected and are ready in the staging area, they can now be saved in repository
- Saving a file in the repository of git is known as doing a commit
- When we commit a repository in git, the commit is identified by a commit id
- The command for initializing this process is *git commit –m "message"*

Git Lifecycle



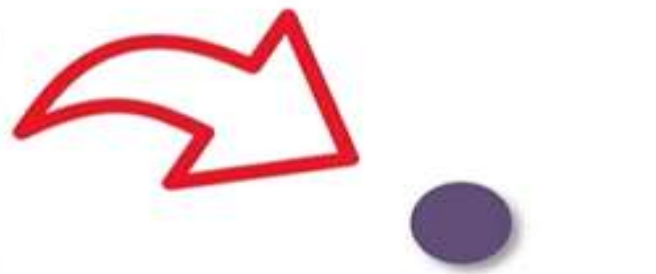
Git Lifecycle



How does Git work?

Any project which is saved on git, is saved using a commit. The commit is identified using a commit ID.

Project Folder



Commit ID: 00001

How does Git work?

When we edit the project or add any new functionality, the new code is again committed to git, a new commit ID is assigned to this modified project. The older code is stored by git, and will be accessible by it's assigned Commit ID

Project Folder

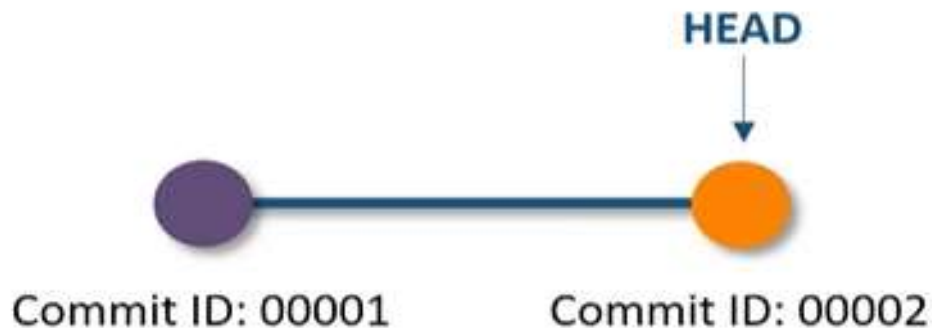
Commit ID: 00002

Commit ID: 00001

How does Git work?

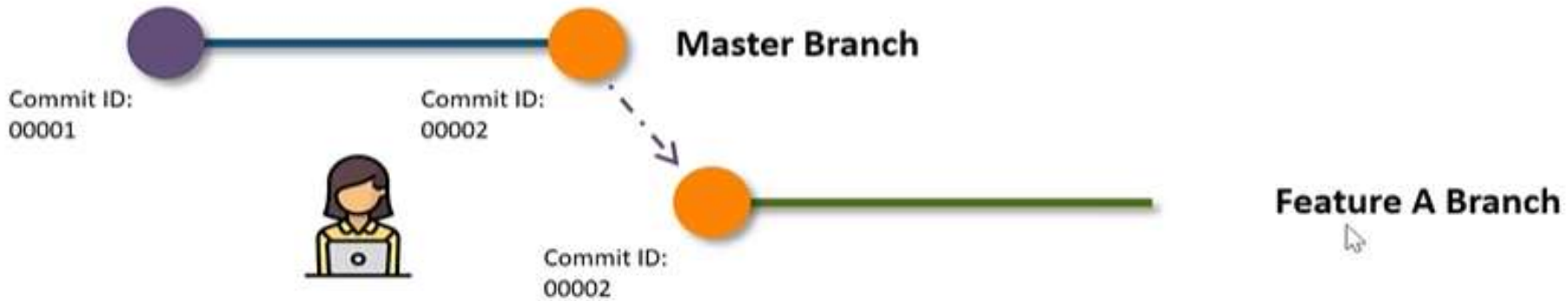
All these commits are bound to a **branch.** Any new commits made will be added to this branch. A branch always points to the latest commit. The pointer to the latest commit is known as **HEAD**

Project Folder



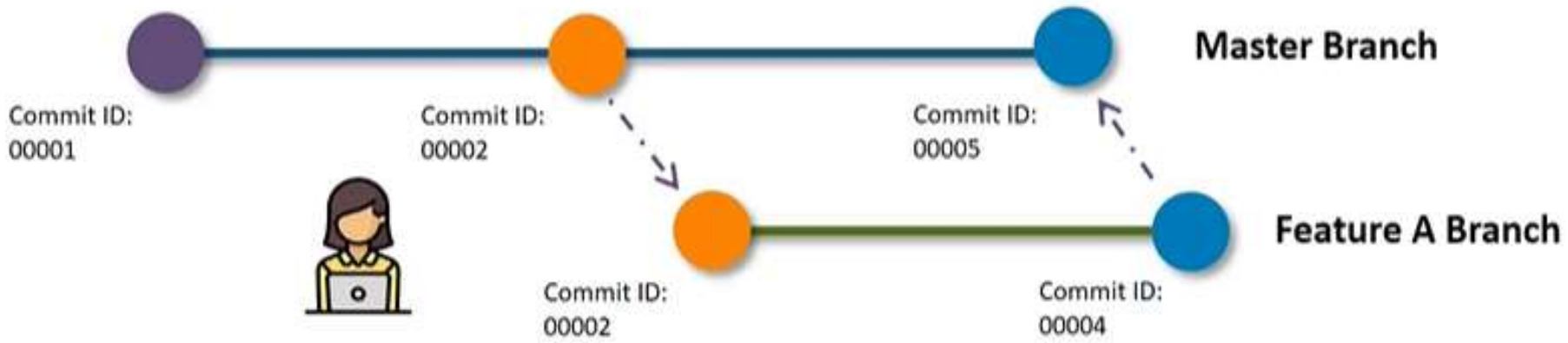
How does Git work?

Say, a developer has been assigned enhance this code by adding Feature A. The code is assigned to this developer in a separate branch “Feature A”. This is done, so that master contains only the code which is finished, finalized and is on production



How does Git work?

Once the code is finished, tested and ready we can merge the Feature A branch, with the master branch and now the code is available on the production servers as well



Common Git Commands

You can do the following tasks, when working with git. Let us explore the commands related to each of these tasks

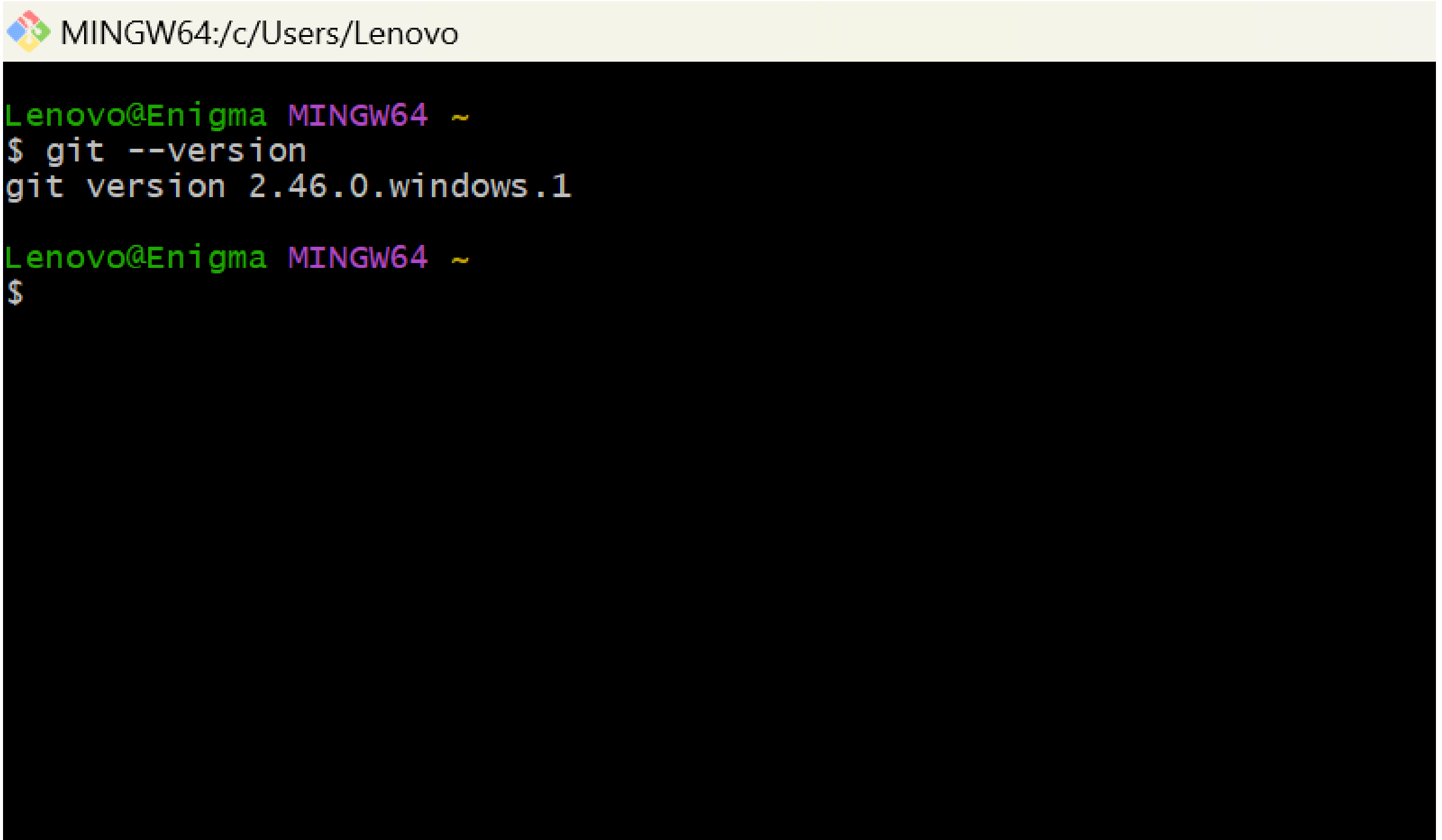
Creating Repository

Making Changes

Parallel Development

Syncing Repositories

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Lenovo\Documents\Devops\gitrepo>
PS C:\Users\Lenovo\Documents\Devops\gitrepo>
PS C:\Users\Lenovo\Documents\Devops\gitrepo> $ git --version
$ : The term '$' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the
spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ $ git --version
+ ~
    + CategoryInfo          : ObjectNotFound: ($:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\Lenovo\Documents\Devops\gitrepo> git --version
git version 2.46.0.windows.1
PS C:\Users\Lenovo\Documents\Devops\gitrepo> mkdir myrepo


    Directory: C:\Users\Lenovo\Documents\Devops\gitrepo


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         8/25/2024  12:02 AM                myrepo


PS C:\Users\Lenovo\Documents\Devops\gitrepo>
```


MINGW64:/c/Users/Lenovo
Lenovo@Enigma MINGW64 ~
$ git --version
git version 2.46.0.windows.1
Lenovo@Enigma MINGW64 ~
$