DevTech Training

Short Course - Day 1

Training Staff

- **WORGIE V. FLORES**
  ETRACS Developer

In This Training

In This Training

- ETRACS Environment Setups
- ETRACS Deployment Setups
- Working Environment Setup
- Git

In This Training

- Virtualization
- Virtual Machine vs Docker
- Docker
- iReport Designer

ETRACS Environment Setups

Standalone

- Windows, Mac & Linux
- MySQL / MSSQL
- Java
- Git

Docker Deployment

- Windows, Mac & Linux
- MySQL / MSSQL
- Docker Engine
- Git

- Optional Add-ons
  - Hypervisor

ETRACS Deployment Setups

	Standalone	Docker
Main		✓
Province	✓	✓
Municipality	✓	✓
City	✓	✓
Remote	✓	

- Barangay, Hospital, Market, Terminals, etc…

Working Environment Setup

Working Environment Setup

- **WSL 2**
- **Docker Desktop**
- **Ubuntu** (18 or 20) from the Microsoft Store
- **Database Engine** (MySQL / MSSQL)
- **Java** 1.8

About your setup

- **WSL** 2
  - Windows Subsystem For Linux (**WSL**) is a tool provided by Microsoft to run Linux natively on Windows
  - Essentially providing a full Linux shell that can interact with your Windows file system
  - WSL 2, is a new version that powers the architecture to run ELF64 Linux binaries on Windows, and increase the file system performance, as well as adding full system call compatibility

About your setup

- **Docker Desktop**
  - An easy-to-install application for your Mac or Windows environment that enables you to build and share containerized applications and micro-services
  - Includes **Docker Engine**, **Docker CLI** client, **Docker Compose**, **Docker Content Trust**, and **Credential Helper**

Check Setup Status

- Press **Windows Logo** + **S**, then type **PowerShell**, and then open the "**Windows PowerShell**" app

- In the **Windows PowerShell** console window, execute the command:

```
wsl -l -v
```

- Result should be:

```
  NAME                   STATE           VERSION
* Ubuntu-20.04           Running         2
```

Check Setup Status

- Press **Windows Logo** + **S**, then type **Ubuntu**, and then open the "**Ubuntu-20**" app

- In the **Ubuntu** console window, execute the command:

```
docker   -v
```

- Result should be:

```
ubuntu@ubuntu-server:~$ docker -v
Docker version 20.10.8, build 3967b7d
```

Git

(Play Video 01)

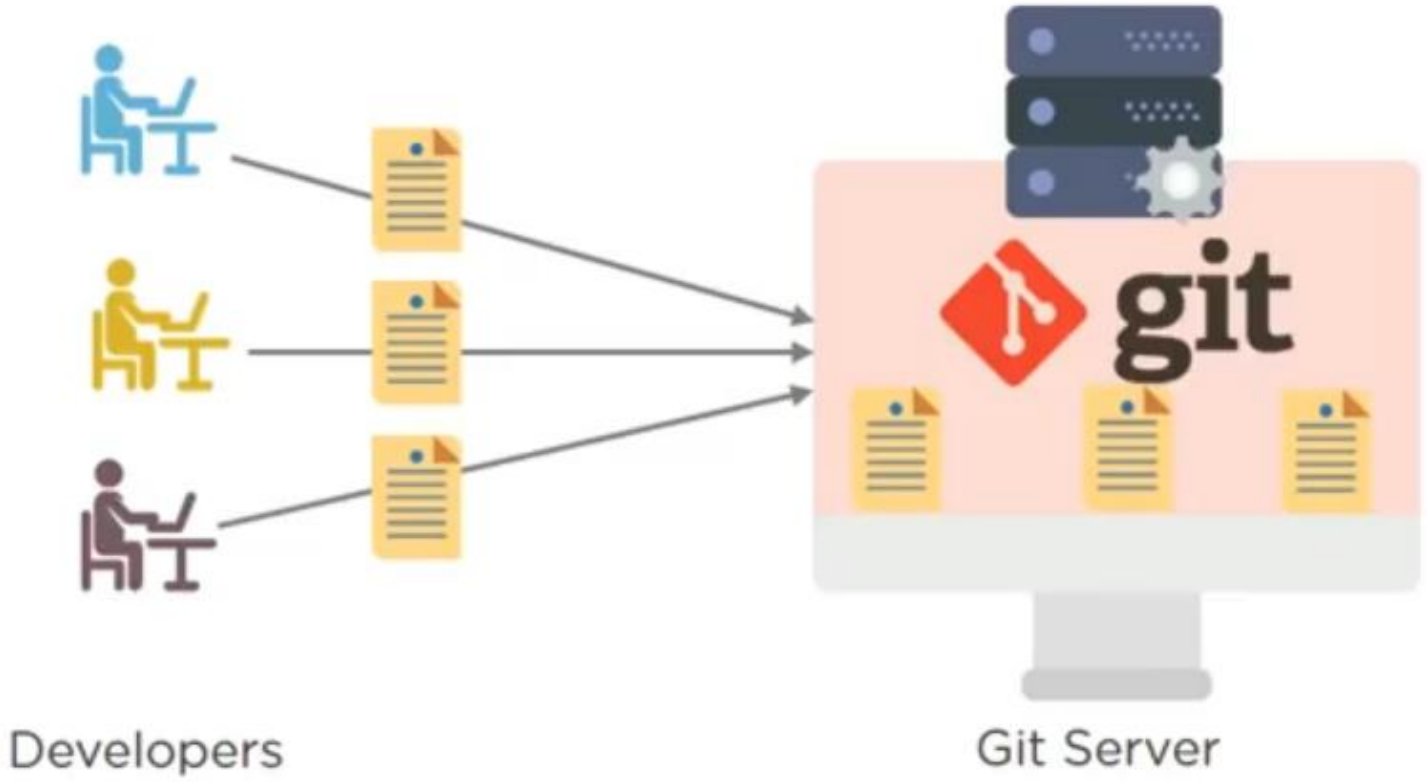
About Git

- Introduction
- Features
- Workflow
- Branching
- Commands
- Demo

What is Git ?

- Git is a distributed version control tool
- It is a popular version control system
- It is used for:
  - Tracking code changes
  - Tracking who made changes
  - Coding collaboration
  - Maintaining historical and current versions of source code
- It allows multiple developers to work together
- Supports non-linear development because of its thousands of parallel branches

What is Git ?

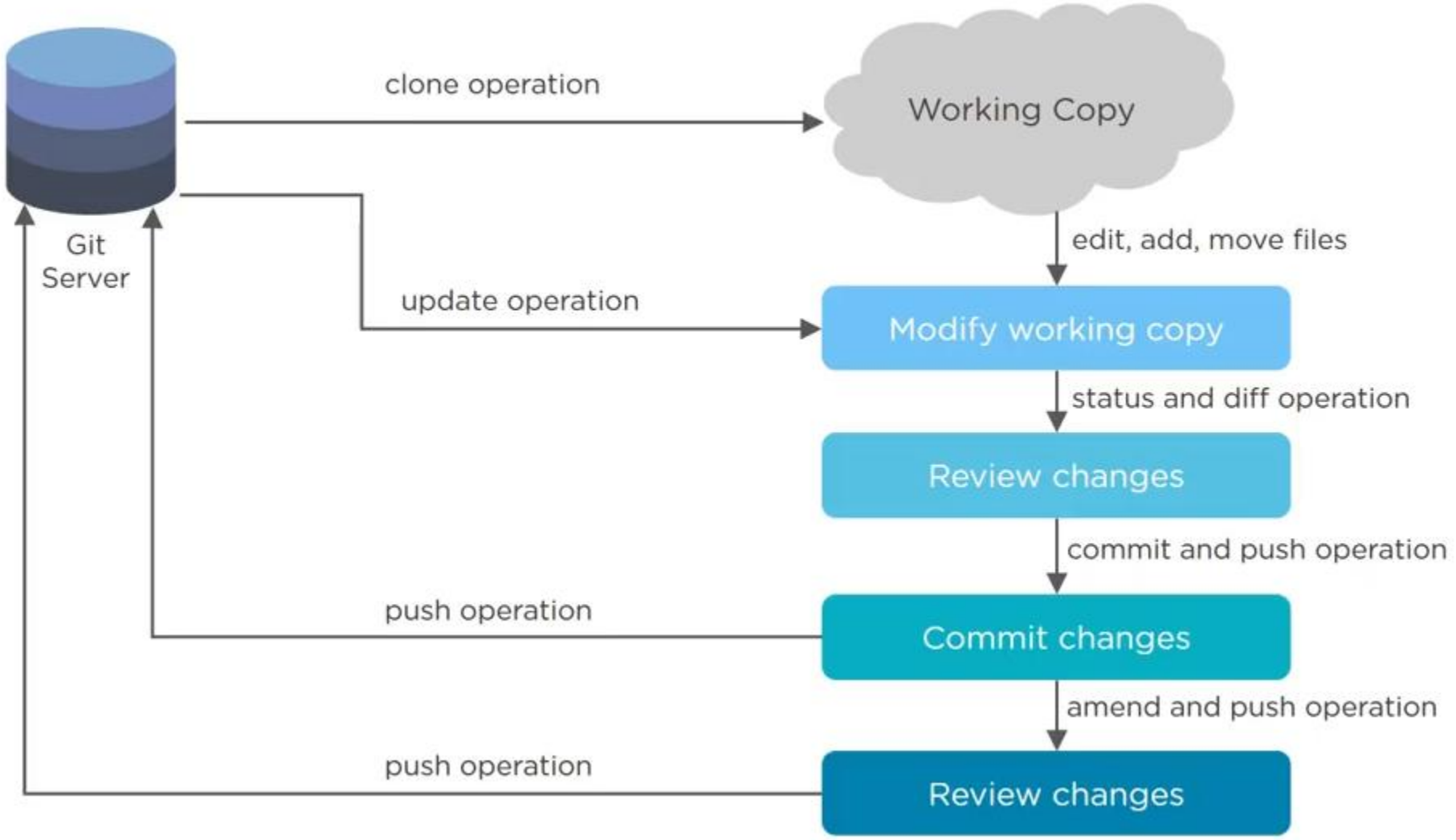


Allows multiple developers to work together

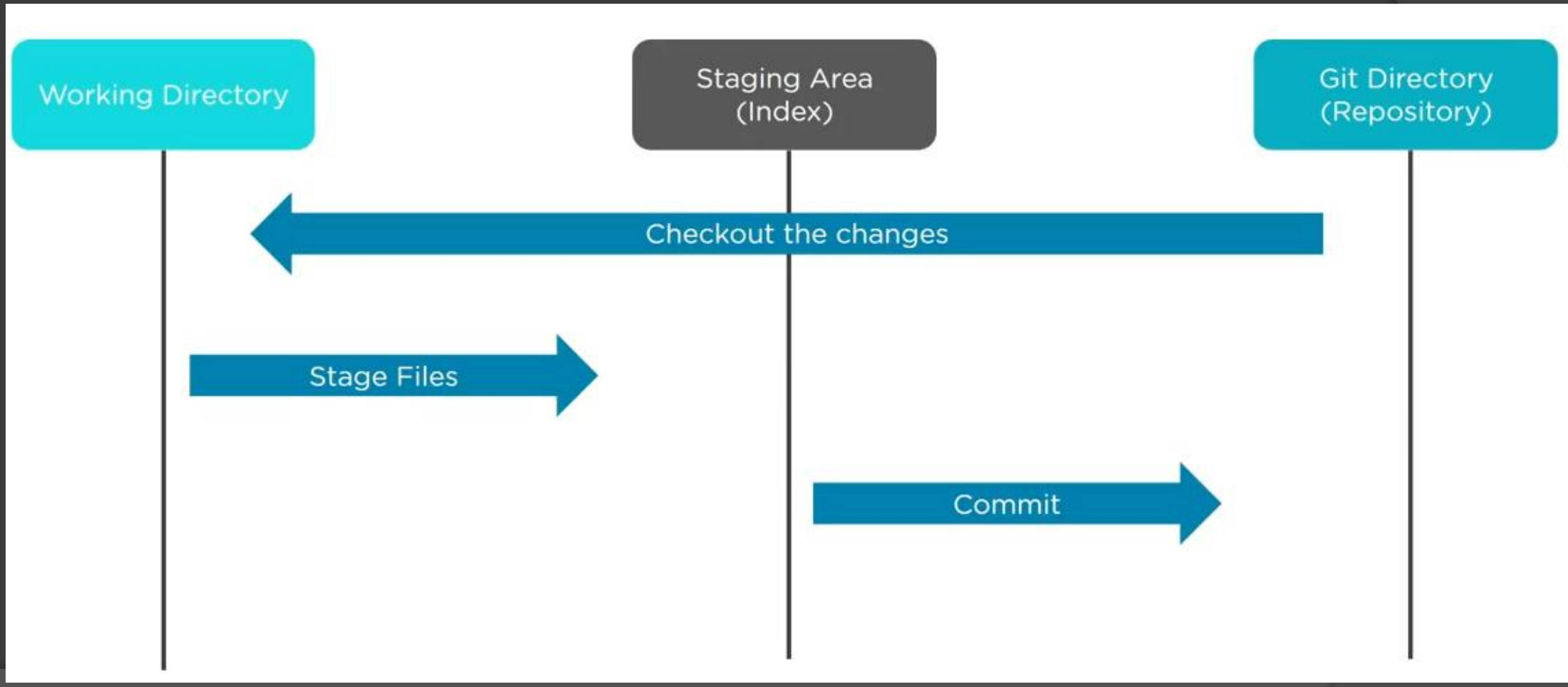
Features of Git

- Free and Open Source
- Tracks History
- Supports Non-Linear Development
- Creates Backup
- Scalable
- Supports Collaboration
- Branching
- Distributed Development

Git Workflow



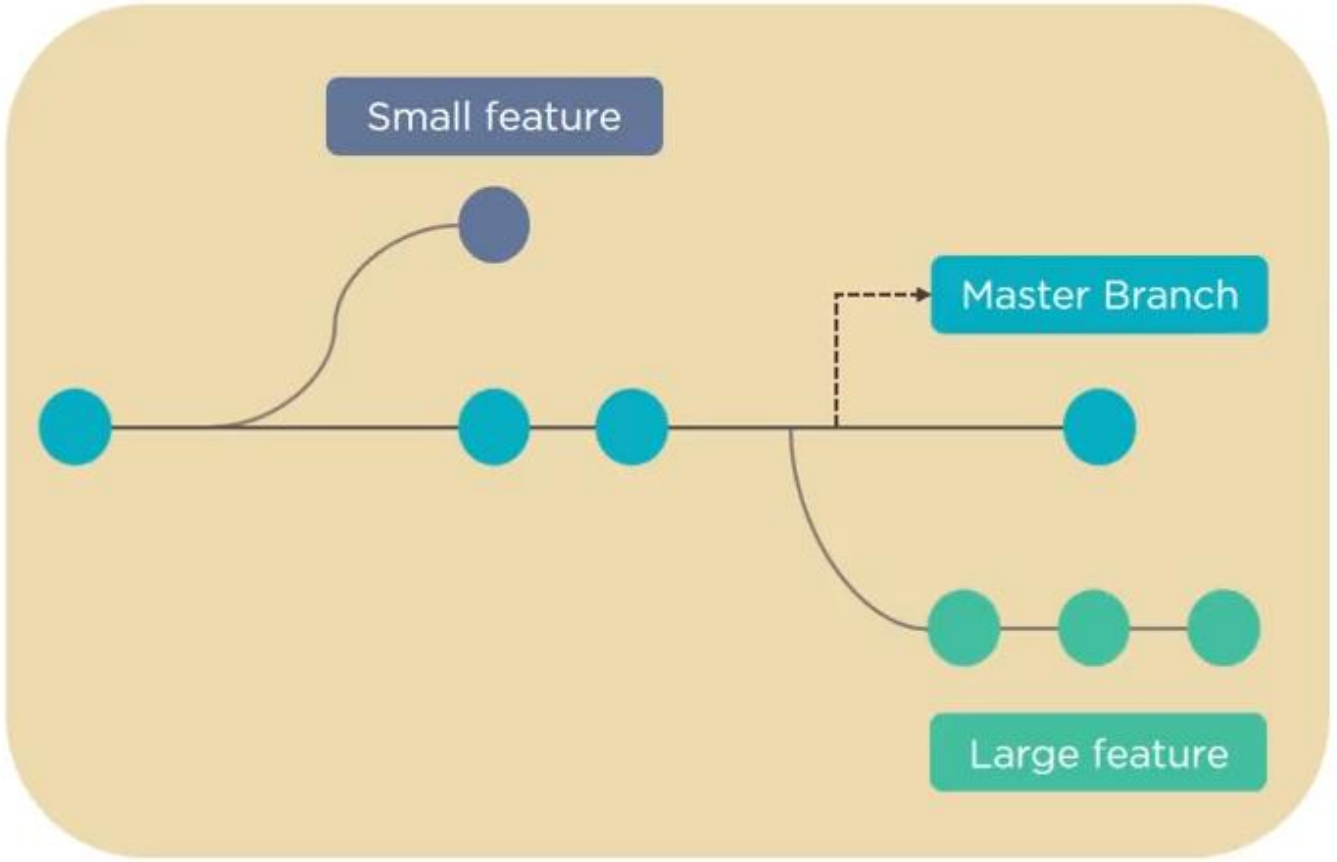
Git Workflow - 3 States



Branch in Git

- It is used to keep your changes until they are ready
- You can do your work on a branch while the main branch (master) remains stable. After you are done with your work, you can merge it to the main branch

Branch in Git



- The diagram shows there is a master branch
- There are 2 more branches, Small feature and Large feature working separately

Basic Commands in Git

Git Commands

git config

- A convenience function that is used to set **Git** configuration values on a global or local project level
- These configuration levels correspond to the **.gitconfig** text files

Git Commands

```
git  init
```

- Create a new **Git** repository or initialize a new empty repository
- Creates a **.git** subdirectory in the current working directory, which contains all of the necessary Git metadata for the new repository

Git Commands

git clone

- Used to target an existing repository and creates a clone, or copy of the target repository

Git Commands

git status

- Gives all the necessary information about the current branch.
- Displays the state of the working directory and the staging area
- It lets you see which changes have been staged, which haven't, and which files aren't being tracked by **Git**

Git Commands

git add

- Adds a change in the working directory to the staging area

Git Commands

git commit

- The most-used command of Git. Once we reach a certain point in development, we want to save our changes (maybe after a specific task or issue).
- Git commit is like setting a checkpoint in the development process which you can go back to later if needed.
- We also need to write a short message to explain what we have developed or changed in the source code.

Git Commands

git remote

- Manage set of tracked repositories

Git Commands

git push

- Uploads your commits to the remote repository.

Git Commands

```
git  pull
```

- Used to get updates from the remote repository

Git Commands

git fetch

- Download objects and refs from another repository

Git Commands

git branch

- Used to create, list, rename, and delete branches

Git Commands

git checkout

- Used mostly for switching from one branch to another
- Restore working tree files

Git Commands

```
git  status
```

- Show the working tree status

Git Commands

git diff

- Show changes between commits, commit and working tree, etc…

Git Commands

```
git  log
```

- Show commit logs

Git Commands

```
git  --help
```

- Shows the help information

Demo on Git

Configure Git for the first time

```
git config --global user.name "Juan Dela Cruz"

git config --global user.email "jdelacruz@gmail.com"
```

Display the configuration settings

```
git config --list
```

To check the version

```
git --version
```

To check the help information

```
git --help
```

Create a new local repository

1. Open your Windows File Manager (press WIN + E)
2. Create a folder "**training**" under drive C
3. Switch back to your **Ubuntu** console window and execute the following commands:

```
## Go to your training folder
cd /mnt/c/training

## Initialize a Git repository
git init test-repo

## Go to the repository folder
cd test-repo
```

Adding files to the repository

```
## create some files
touch file1.txt
touch file2.txt

## check the status
## untrack files are in red color
git status

## stage the files
git add file1.txt file2.txt

## check the status of staged files
## staged files are in green color
git status

## commit your changes
git commit -m 'my first commit'
```

Check repository logs

```
## display logs with 30 max lines
git log -n 30

## display logs with 30 max lines
## and with graphical representation
git log --graph -n 30
```

Cloning repository from GitHub

```
## Go to the training folder
cd /mnt/c/training

## Clone a remote repository
git clone https://github.com/ramesesinc/training-202206.git

## Go to the repository folder
cd training-202206

## check the remote endpoints
git remote -v
```

Create a local repository registry

```
## Go to the training folder
cd /mnt/c/training

## Create a gitrepo directory
mkdir gitrepo

## Create a repository registry
git init --bare gitrepo/training-202206.git
```

Mount a local repository registry from file

```
## Go to the training folder
cd /mnt/c/training

## Go to the working repository
cd training-202206

## Check the current remote endpoints
git remote -v

## Register a remote endpoint
git remote add localfile file:///mnt/c/training/gitrepo/training-202206.git

## Check the current remote endpoints
## localfile must already be added
git remote -v
```

Mount a local repository registry from your local server

```
## Go to the training folder
cd /mnt/c/training

## Go to the working repository
cd training-202206

## Check the current remote endpoints
git remote -v

## Register a remote endpoint
git remote add localserver ubuntu@192.168.0.10:gitrepo/training-202206.git

## Check the current remote endpoints
## localfile must already be added
git remote -v
```

Pull updates from remote repository

```
## Go to your repository folder
cd /mnt/c/training/training-202206

## Pull updates
git pull

## Check logs for commit messages
git log --graph -n 30
```

Pull updates from other remote repository

```
## Go to the working repository folder
cd /mnt/c/training/training-202206

## Check the available remote endpoints
git remote -v

## Pull updates from the remote name
git pull localfile master

## Check the logs
## with maximum 30 lines
## with graphical representation
git log --graph -n 30
```

Push updates to remote repository

```
## Go to your repository folder
cd /mnt/c/training/training-202206

## Push updates
git push
```

Push updates to other remote repository

```
## Go to the working repository folder
cd /mnt/c/training/training-202206

## Check the available remote endpoints
git remote -v

## Push updates using the localfile
git push localfile master

## Push updates using the localserver
git push localserver master
```

Create a branch to fix isolated bug

```
## Go to the working repository folder
cd /mnt/c/training/training-202206

## Force to checkout the main branch (master)
git checkout master

## Pull updates from remote orgin
## before doing anything
git pull

## Create a branch
git branch fix-feature

## Checkout the created branch (fix-feature)
git checkout fix-feature
```

Create a branch to fix isolated bug

```
## perform the needed fix
## for this branch

## Stage the changes
git add .

## Commit your changes
git commit -m 'i fixed something here'

## Checkout the master branch
## and merge the fix-feature branch
git checkout master
git merge fix-feature

## Push all local commits
git push
```

Documentations

- **Git**

  https://git-scm.com/doc

- **GitHub**

  https://github.com

Next Topic

- Virtualization
- Virtual Machine vs Docker
- Docker
- iReport Designer
- Report Editing and Management