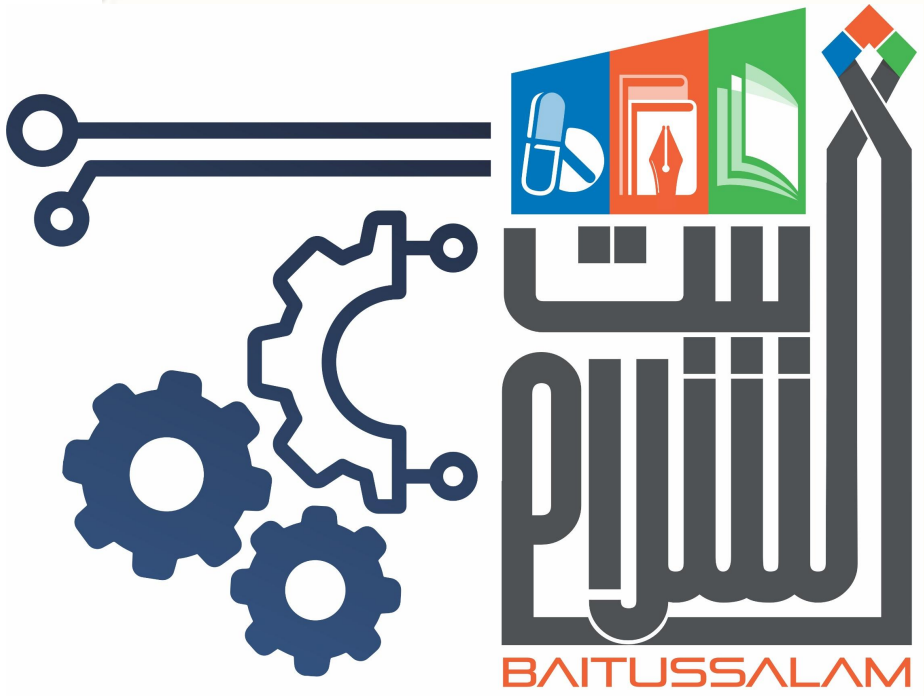


BAITUSSALAM
—TECH PARK—

GIT

BAITUSSALAM
—TECH PARK—

What is Version Control…?

- Version control, also known as source control, is the practice of tracking and managing changes to software code.
- Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter.

Benefits of Version Control Systems:

- Tracking Changes
- Collaboration
- History and Rollback
- Branching and merging
- Concurrency

What is GIT…?

- Git is a free and open-source version control system, originally created by **Linus Torvalds** in **2005**.

- Unlike older centralized version control systems such as SVN and CVS, **Git is distributed**: every developer has the full history of their code repository locally. This makes the initial clone of the repository slower, but subsequent operations such as commit, blame, diff, merge, and log dramatically faster.

- Git also has excellent support for branching, merging, and rewriting repository history

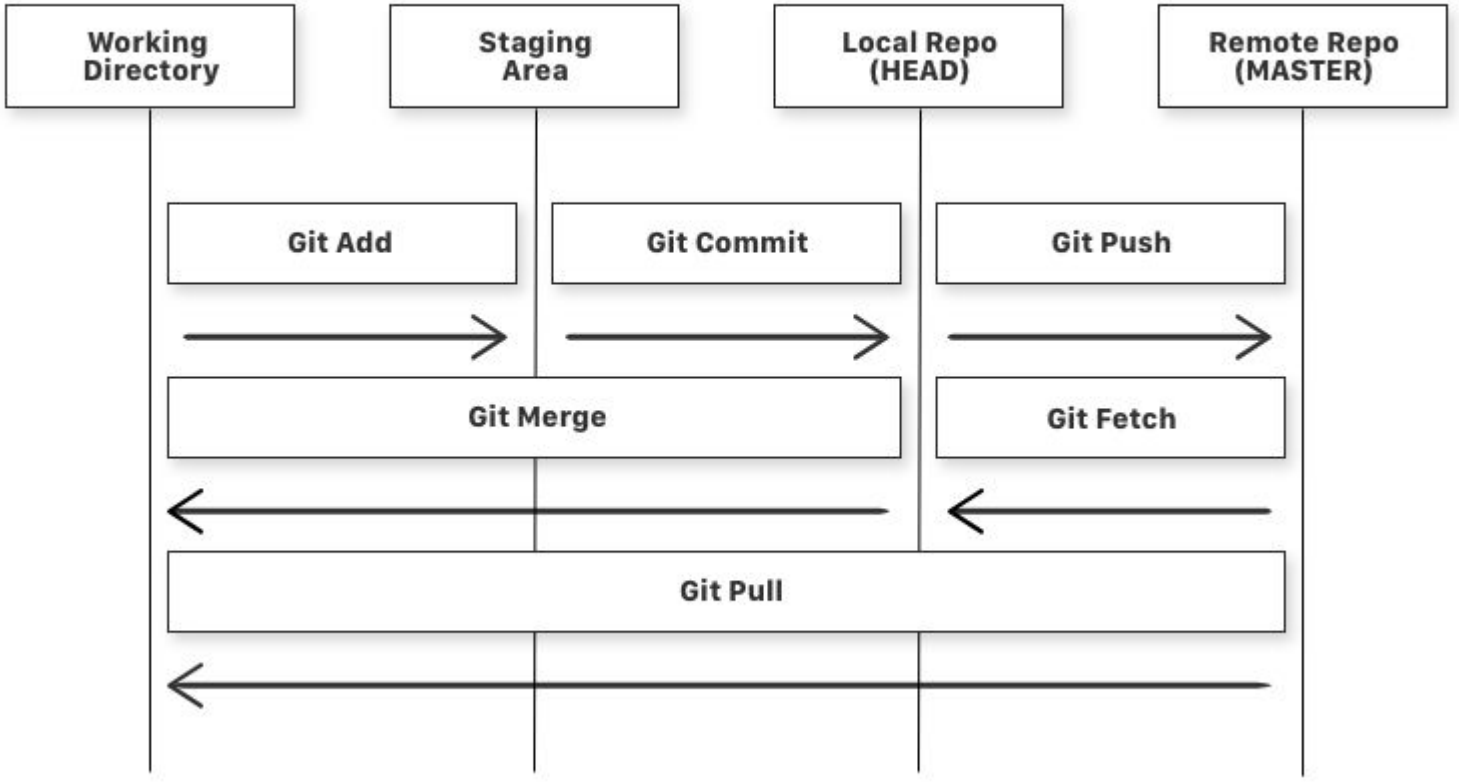
- Git is the most widely used version control system in the world today and is considered the modern standard for software development.

What is Repository…?

- A repository a.k.a. repo is nothing but a collection of source code.

Git Workflow…

- Working Directory, Staging Area, Local Repository and Remote Repository.



Install Git

For Debian / Ubuntu:

From your shell, install Git using apt-get:

```
sudo apt-get update
sudo apt-get install git
```

Verify the installation

```
git --version
```

Configure your Git username and email using the following commands
These details will be associated with any commits that you create:

```
git config --global user.name “Yasir Butt”
git config --global user.email “yasir@baitussalam.org”
```

Git Commands…

Setup:

Configuring the name

```
git config --global user.name “Firstname Lastname”
```

Configuring the email

```
git config --global user.name “Valid-email”
```

Git Basics:

Initialize an existing directory as a Git repository

```
git init
```

Retrieve an entire repository from a hosted location via URL

```
git clone [url]
```

Git Commands…

Stage & Snapshot:

Show the modified files in the working directory, staged for your next commit

```
git status
```

Add a file as it looks now to your next commit (stage)

```
git add [file]
```

Unstage a file while retaining the changes in working directory

```
git reset [file]
```

Diff of what is changed but not staged

```
git diff
```

Git Commands…

Stage & Snapshot:

Diff of what is staged but not yet committed

```
git diff --staged
```

Commit your staged content as a new commit snapshot

```
git commit -m “[descriptive message]”
```

Git Commands…

Branch & Merge:

List your branches. a * will appear next to the currently active branch

```
git branch
```

Create a new branch at the current commit

```
git branch [branch-name]
```

Switch to another branch and check it out into your working directory

```
git checkout
```

Merge the specified branch’s history into the current one

```
git merge [branch]
```

Git Commands…

Inspect & Compare:

Show the commit history for the currently active branch

```
git log
```

Show the commits on branchA that are not on branchB

```
git log branchB..branchA
```

Show the commits that changed file, even across renames

```
git log --follow [file]
```

Show any object in Git in human-readable format

```
git show [SHA]
```

Show the diff of what is in branchA that is not in branchB

```
git diff branchB...branchA
```

Git Commands…

Share & Update:

Add a git URL as an alias

```
git remote add [alias] [url]
```

Fetch down all the branches from that Git remote

```
git fetch [alias]
```

Merge a remote branch into your current branch to bring it up to date

```
git merge [alias]/[branch]
```

Transmit local branch commits to the remote repository branch

```
git push [alias] [branch]
```

Fetch and merge any commits from the tracking remote branch

```
git pull
```

Git Commands…

Tracking Path Changes:

Delete the file from project and stage the removal for commit

```
git rm [file]
```

Change an existing file path and stage the move

```
git mv [existing-path] [new-path]
```

Show all commit logs with indication of any paths that moved

```
git log --stat -M
```

Git Commands…

Rewrite History:

Apply any commits of current branch ahead of specified one

```
git rebase [branch]
```

Clear staging area, rewrite working tree from specific commit

```
git reset --hard [commit]
```

Git Commands…

Temporary Commits:

Save modified and staged changes

```
git stash
```

List stack-order of stashed file changes

```
git stash list
```

Write working from top of stash stack

```
git stash pop
```

Discard the changes from top of stash stack

```
git stash drop
```

What is a Git SSH Key…?

- An SSH key is an access credential for the SSH (secure shell) network protocol.
- SSH is used for remote file transfer, network management, and remote operating system access.
- SSH uses a pair of keys to initiate a secure handshake between remote parties.
- The key pair contains a public and private key.

SSH Key…

- SSH keys are generated through a public key cryptographic algorithm, the most common being RSA or DSA or ECC.

- SSH keys are generated through a mathematical formula that takes 2 prime numbers and a random seed variable to output the public and private key. This is a one-way formula that ensures the public key can be derived from the private key but the private key cannot be derived from the public key.

- SSH keys are created using a key generation tool. The SSH command line tool suite includes a keygen tool.

- Most git hosting providers offer guides on how to create an SSH Key.

Create an SSH Key… (1 of 3)

- Execute the following to begin the key creation

  ssh-keygen -t rsa -b 4096 -C “yasir@baitussalam.org”

  OR

  ssh-keygen -t ed25519 -C “email”

  This command will create a new SSH key using the email as a label

- Enter a file in which to save the key.

- You can specify a file location or press “Enter” to accept the default file location.

  Enter a file in which to save the key (/Users/you/.ssh/id_rsa): [Press enter]

Create an SSH Key… (2 of 3)

- The next prompt will ask for a secure passphrase. A passphrase will add an additional layer of security to the SSH and will be required anytime the SSH key is used. If someone gains access to the computer that private keys are stored on, they could also gain access to any system that uses that key. Adding a passphrase to keys will prevent this scenario.

  Enter passphrase (empty for no passphrase): [Type a passphrase

  Enter same passphrase again: [Type passphrase again]

- At this point, a new SSH key will have been generated at the previously specified file path.

Create an SSH Key… (3 of 3)

- Add the new SSH key to the ssh-agent
- The ssh-agent is responsible for holding private keys. Think of it like a keychain. In addition to holding private keys it also brokers requests to sign SSH requests with the private keys so that private keys are never passed around unsecurly.
- Before adding the new SSH key to the ssh-agent first ensure the ssh-agent is running by executing:

  eval "$(ssh-agent -s)"

- Once the ssh-agent is running the following command will add the new SSH key to the local SSH agent.

  ssh-add ~/.ssh/id_ed25519 (The new SSH key is now registered and ready to use!)
  cat ~/.ssh/id_ed25519.pub (Copy output of this command and paste it in GitHub SSH key section)

The End

BAITUSSALAM
—TECH PARK—