**to validate GIT version**

git --version

**Generate SSH keys using below**

ssh-keygen -t rsa -b 4096 (generate SSH keys)

and copy the public key on GitHub account to establish SSH connectivity

**to set up user name and email:**

git config –global user.name “<user\_name>”

git config –global user.email “<user\_email\_id>”

git config –list > List all key-value configurations

git config --get user.name > Get the value of a single key

Clone a repository from remote hosts (GitHub, GitLab, DagsHub, etc.)

$ git clone <remote\_repo\_url>

Initialize git tracking inside the current directory

$ git init

Create a git-tracked repository inside a new directory

$ git init [dir\_name]

Clone only a specific branch

$ git clone -branch <branch\_name> <repo\_url>

Cloning into a specified directory

$ git clone <repo\_url> <dir\_name>

**git init :**

Executing git init creates a .git subdirectory in the current working directory, which contains all of the necessary Git metadata for the new repository. This metadata includes subdirectories for objects, refs, and template files. A HEAD file is also created which points to the currently checked out commit.

Managing remote repositories

List remote repos

$ git remote

Create a new connection called <remote> to a remote repository on servers like GitHub, GitLab, DagsHub, etc.

$ git remote add <remote> <url\_to\_remote>

Remove a connection to a remote repo called <remote>

$ git remote rm <remote>

Rename a remote connection

$ git remote rename <old\_name> <new\_name>

<<<<<<A note on cloning >>>>

There are two primary methods of cloning a repository - HTTPS syntax and

SSH syntax. While SSH cloning is generally considered a bit more secure

because you have to use an SSH key for authentication, HTTPS cloning is

much simpler and the recommended cloning option by GitHub.

HTTPS

$ git clone https://github.com/your\_username/repo\_name.git

SSH

$ git clone [git@github.com:user\_name/repo\_name.git](mailto:git@github.com:user_name/repo_name.git)

**git alias --**

It is important to note that there is no direct git alias command. Aliases are created through the use of the [git config](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config) command and the Git configuration files. As with other configuration values, aliases can be created in a local or global scope.

To better understand Git aliases let us create some examples.

$ git config --global alias.co checkout

$ git config --global alias.br branch

$ git config --global alias.ci commit

$ git config --global alias.st status

Add a file or directory to git for tracking

$ git add <filename\_or\_dir>

Add all untracked and tracked files inside the current directory to git

$ git add .

Remove a file from a working directory or staging area

$ git rm <<filename\_or\_dir>

See changes in the local repository

$ git status

Saving a snapshot of the staged changes with a custom message

$ git commit -m "[Commit message]"

Staging changes in all tracked files and committing with a message

$ git commit -am "[Commit message]"

Editing the message of the latest commit

$ git commit --amend -m “[New commit message]”