Contents

[Contents 1](#_Toc513477988)

[1 git 2](#_Toc513477989)

[1.1 Commands 2](#_Toc513477990)

[1.1.1 git init 2](#_Toc513477991)

[1.1.2 git remote add origin 2](#_Toc513477992)

[1.1.3 git pull <branch\_name> <remote\_URL/remote\_name> 2](#_Toc513477993)

[1.1.4 git status 2](#_Toc513477994)

[1.1.5 git add 2](#_Toc513477995)

[1.1.6 git commit 2](#_Toc513477996)

[1.1.7 git log 3](#_Toc513477997)

[1.1.8 git branch 3](#_Toc513477998)

[1.1.9 git checkout 3](#_Toc513477999)

[1.1.10 git merge 4](#_Toc513478000)

[1.1.11 Difference Between git pull and git fetch 4](#_Toc513478001)

[1.1.12 git rebase 4](#_Toc513478002)

[1.1.13 Difference between merging and rebasing 4](#_Toc513478003)

[1.1.14 git push 5](#_Toc513478004)

[2 Docker 6](#_Toc513478005)

[2.1 Overview 6](#_Toc513478006)

[2.2 Docker In A Nutshell 7](#_Toc513478007)

[2.3 Docker Example 7](#_Toc513478008)

[7](#_Toc513478009)

# git

## Commands

### git init

Initialize a local Git repository. .git folder will be created in your local drive.

### git remote add origin

* Example

git remote add origin "https://github.com/MohantyOne/mohantys\_git.git"

* Description

Add a remote repository

### git pull <branch\_name> <remote\_URL/remote\_name>

* Example

git pull origin master

* Description

To get the latest version of a repository run git pull. This pulls the changes from the remote repository to the local computer.

### git status

* Example

git status

* Description

git status will return the current working branch. If a file is in the staging area, but not committed, it shows with git status. Or, if there are no changes it’ll return nothing to commit, working directory clean.

### git add

* Example

# To add all files not staged:

$ git add **.**

**OR**

$ git add **-**A

# To stage a specific file:

$ git add index.html

# To stage an entire directory:

$ git add css

* Description

Adds files in to the staging area for Git. Before a file is available to commit to a repository, the file needs to be added to the Git index (staging area). There are a few different ways to use git add, by adding entire directories, specific files, or all unstaged files.

### git commit

* Example

# Adding a commit with message

$ git commit -m "Commit message in quotes"

# Adding all commit with message

$ git commit **.** -m "Commit message in quotes"

OR

$ git commit -a -m "Commit message in quotes"

* Description

Record the changes made to the files to a local repository. For easy reference, each commit has a unique ID.

Its best practice to include a message with each commit explaining the changes made in a commit. Adding a commit message helps to find a particular change or understanding the changes.

### git log

* Example

# Show entire git log

$ git log

# Show git log with date parameters

$ git log --<after/before/since/until>=<date>

# Show git log based on commit author

$ git log --<author>="Author Name"

* Description

To show the chronological commit history for a repository. This helps give context and history for a repository. git log is available immediately on a recently cloned repository to see history.

### git branch

* Example

# Create a new branch

$ git branch <branch\_name>

# List all remote or local branches

$ git branch -a

# Delete a branch

$ git branch -d <branch\_name>

* Description

To determine what branch the local repository is on, add a new branch, or delete a branch.

### git checkout

* Example

# Checkout an existing branch

$ git checkout <branch\_name>

# Checkout and create a new branch with that name

$ git checkout -b <new\_branch>

* Description

To start working in a different branch, use git checkout to switch branches.

### git merge

* Example

# Merge changes into current branch

$ git merge <branch\_name>

* Description

Integrate branches together. *git merge* combines the changes from one branch to another branch. For example, merge the changes made in a staging branch into the stable branch.

### Difference Between git pull and git fetch

git pull will download latest changes from the remote repository and **automatically merge** those changes in the local repository. It doesn’t give you a chance to review the changes and as a result conflicts might occur (and they often do). One important thing to keep in mind is that git pull will merge only into the current working branch. Other branches will stay unaffected.

If you only want to download the latest changes and review them before merging or want to merge at a later time, git fetch is your friend. Like git pull, it will download latest changes. Unlike git pull, it will not merge those changes. You might wonder where the changes are stored since they are not merged. The answer is that they are stored in your local repository in what are called remote tracking branches. A remote tracking branch is a local copy (or mirror) of a remote branch. For example, when you run git branch -a you might notice origin/master in the output which is the remote tracking branch for master.

In short: Both are similar with one key difference: git fetch only downloads latest changes where as git pull also merges them.

### git rebase

* Example

$ git rebase master

* Description

Rebase is one of two Git utilities that specializes in integrating changes from one branch onto another. The other change integration utility is git merge. Merge is always a forward moving change record. Alternatively, rebase has powerful history rewriting features.

### Difference between merging and rebasing

### git push

* Example

# Merge changes into current branch

$ git merge <branch\_name>

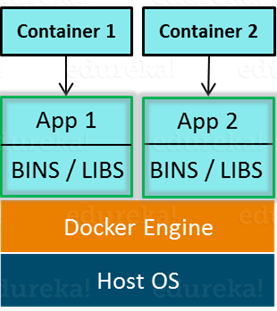
* Description

Integrate branches together. *git merge* combines the changes from one branch to another branch. For example, merge the changes made in a staging branch into the stable branch.

# Docker

## Overview

* Docker is an open platform for developing, shipping, and running applications.
* Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
* With Docker, you can manage your infrastructure in the same ways you manage your applications.
* By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.
* Docker is a containerization platform that packages your application and all its dependencies together in the form of Containers to ensure that your application works seamlessly in any environment.



* As you can see in the diagram on the above, each application will run on a separate container and will have its own set of libraries and dependencies. This also ensures that there is process level isolation, meaning each application is independent of other applications, giving developers surety that they can build applications that will not interfere with one another.
* As a developer, I can build a container which has different applications installed on it and give it to my QA team who will only need to run the container to replicate the developer environment.

## Docker In A Nutshell

* Docker file builds a Docker image and that image contains the entire project’s code.
* You can run that image to create as many Docker containers as you want.
* Then this image can be uploaded on Docker hub, from Docker hub any one can pull the image and build a container.

Container

Virtual Machine

Docker Container

Image

Image

Docker Image

Container

Container

Container

Container

Container

Docker Image

Docker file

Docker Image

Docker Hub

Git Repository

Docker Container is a runtime instance of a Docker image.

## Docker Example

Testing

Jenkins Server

Docker file

Push the code to Git

Repo

Complex requirements for a micro-service are written in easy to write Docker File.

Production

Staging

Git Repo

* CI server pull it down and build the exact environment that will be used in production to run the test suite without needing to configure the CI server at all.
* Deploy it out to a staging environment for tester.
* Roll exactly what you had in development, testing and staging into production.
* Create Complex requirements for a micro-service within an easy to write in a Docker File.
* Push the code up to the Git Repo