# Learn git with Gitlab

## About me

**Rajesh Kumar**

**DevOps Architect**

**@RajeshKumarIN | www.RajeshKumar.xyz**

## Agenda

- What is Git?
- Installing Git
- Getting Started
- Git Concepts and Architecture
- Making Changes to Files
- Using Git with a Real Project
- Undoing Changes
- Ignoring Files
- Navigating the Commit Tree
- Branching
- Merging Branches
- Working with remote repository

## What is git

Manage your source code versions

## Git is Popular

- **Distributed Version Control**
- Open source and free software
- Compatible with Unix-like Systems (Linux, Mac OSX, and Solaris) and Windows
- **Faster than other SCMs (100x in some cases)**

# No network Needed

- Performing a diff
- Viewing file history
- Commiting Changes
- Merging branches
- Obtaining other revision of file
- Switching branches

## Install

http://git-scm.com/downloads

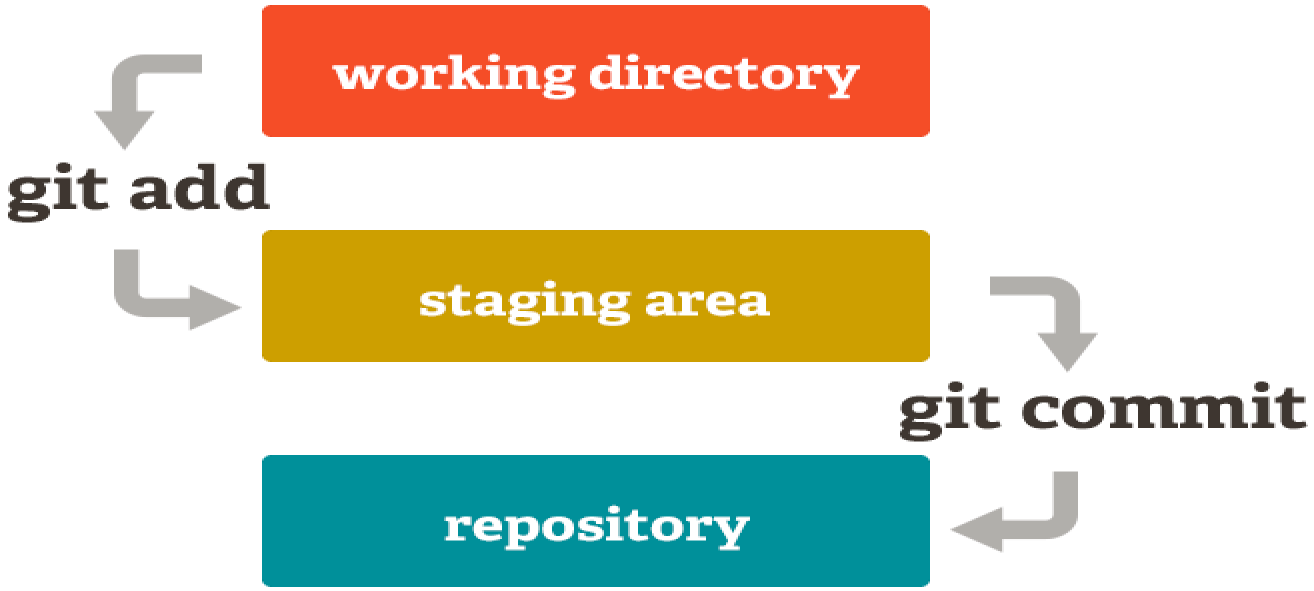
## Verify

```
> git --version
> which git (linux)
```

## Git Basic Workflow (working with local git repo)

- git init
  - It create a git empty repo. Also creates a .git in the current dir
- git add <directory tree>
  - Adds all files (except .git)
- git commit
  - Commits the changes (in this case initial commit)
  - Creates a branch named **master**
  - HEAD points at master

# Git workflow



## Configuring Git…

- git config --global user.name "rajesh kumar"
- git config --global user.email rajesh@scmGalaxy.com
- git config --list

## Create a git repo

```
> git init
```

# Add file to git repo

```
> git add <filename>
```

## Submit Changes to Repos

```
> git commit –m"This is my First commit"
```

# Check and Verify the work

git log - Check the list of commit to repos

&

git status – To see the workspace status

# Working with remote repo (gitlab)

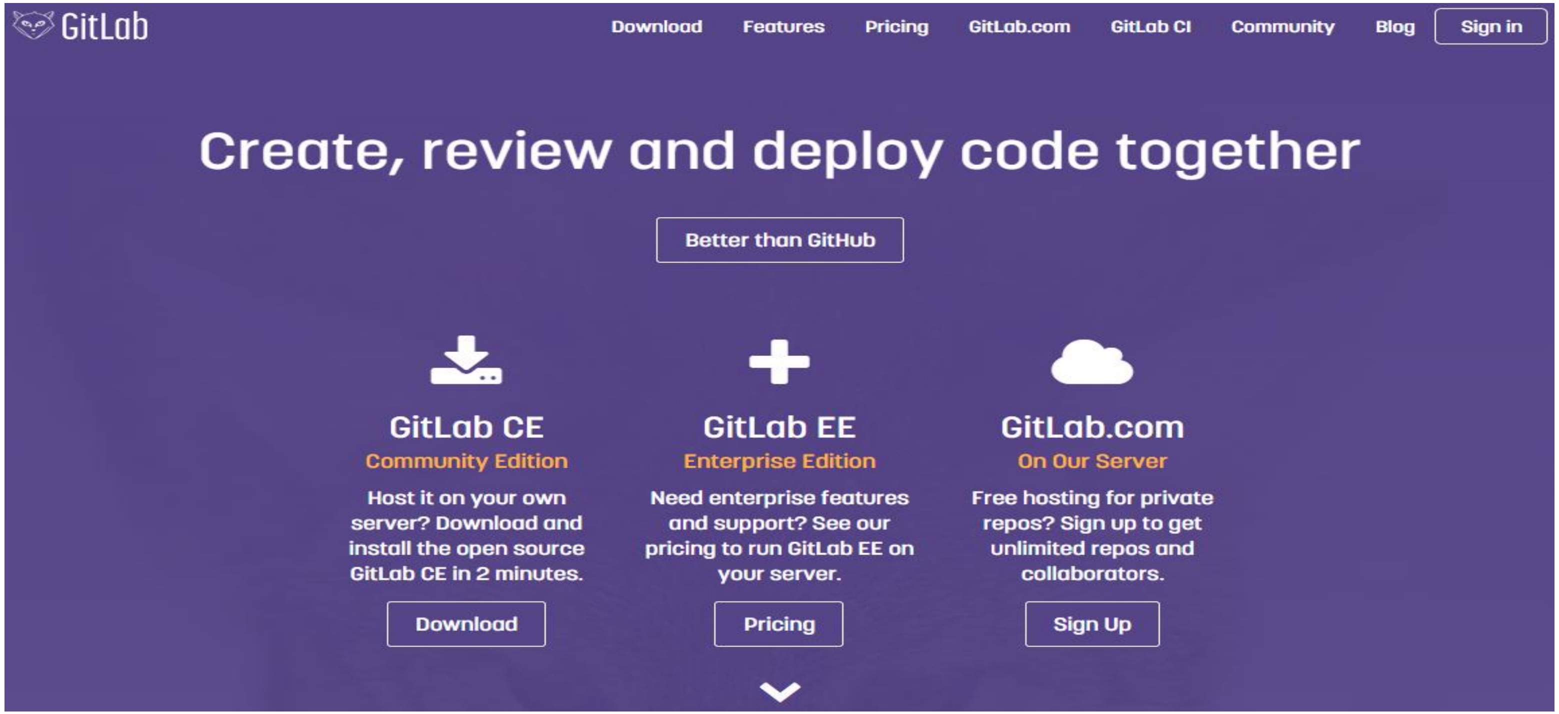
## You Decide?

https://gitlab.com/

Or

Local Hosted Gitlab

# Create a new repos in gitlab



Create, review and deploy code together

GitLab CE
Community Edition
Host it on your own server? Download and install the open source GitLab CE in 2 minutes.

GitLab EE
Enterprise Edition
Need enterprise features and support? See our pricing to run GitLab EE on your server.

GitLab.com
On Our Server
Free hosting for private repos? Sign up to get unlimited repos and collaborators.

## Git Basic Workflow
## (working with remote git repo)

## Situation

**1. Do you have local repo already**

**OR**

**2. You don’t have any local repo**

## You have local repo already

```
git remote add origin url <ssh or https>
git push -u origin master
```

# You don’t have any local repo

- **Working with Remote Repos**
  - git clone
    - Creates a git repo from an existing repo
    - All remote branches are tracked
    - Remote HEAD branch checked out as your initial master branch as well
  - git add <directory tree>
    - Adds all files (except .git)
  - git commit
    - Commits the changes (in this case initial commit)
    - Creates a branch named **master**
    - HEAD points at master

Questions?

http://bit.ly/scmgalaxy-forum

## Lets Start the git now ☺

## History

- Source Code Control Systems (SCCS)
  - 1972, Closed Source, free with Unix
- Revision Control System (RCS)
  - 1982, Open Source
- Concurrent Version System (CVS)
  - 1986-1990, open source
- Apache Subversion (SVN)

## BitKeeper SCM

- 2000, closed source, proprietry
- Distributed version control
- “community  version” was free
-  used for source code of the Linux Kernal from 2002-2005
- Controversial to use proprietary SCM for an open source project
- April – 2005 – The community version not free any

## Git is born

- April 2005
- Created  by Linus Torvalds
- Replacement for bitKeeper to manage Linux Kernal source code

## Git is Popular

- Distributed Version Control
- Open source and free software
- Compatible with Unix-like Systems (Linux, Mac OSX, and Solaris) and Windows
- Faster than other SCMs (100x in some cases)

# Git is a hit

- Explosion in Popularity
- No official statics
- GitHub is launched in 2008 to host Git repositories
  - 2009: over 50,000 repositories, over 100,000 users
  - 2011: over 2 million repository's, over 1 million users

  - 2015: over 21.1 million repos, over 9 million users

## Distributed version Control

- No Single Master Repositories; Just many working copies
  - Each with their own combination of change sets
  - Imagine changes to a document as sets A, B, C, D, E, F
    - Repo 1: A, B, C, D, E, F
    - Repo 2: A, B, C, D
    - Repo 3: A, B, C, E

# Distributed version Control

- No Need to communicate with a central server
  - Faster
  - No network access required
  - No single failure point
  - Encourages participation and “forking” of projects
  - Developers can work independently
  - Submit change sets for inclusion or rejection

## Who should use Git

- Anyone wanting to track edits
  - Review a history log of changes made
  - View differences between versions
  - Retire old versions
- Anyone needing to share changes with collaborators
- Anyone not afraid of command line tools

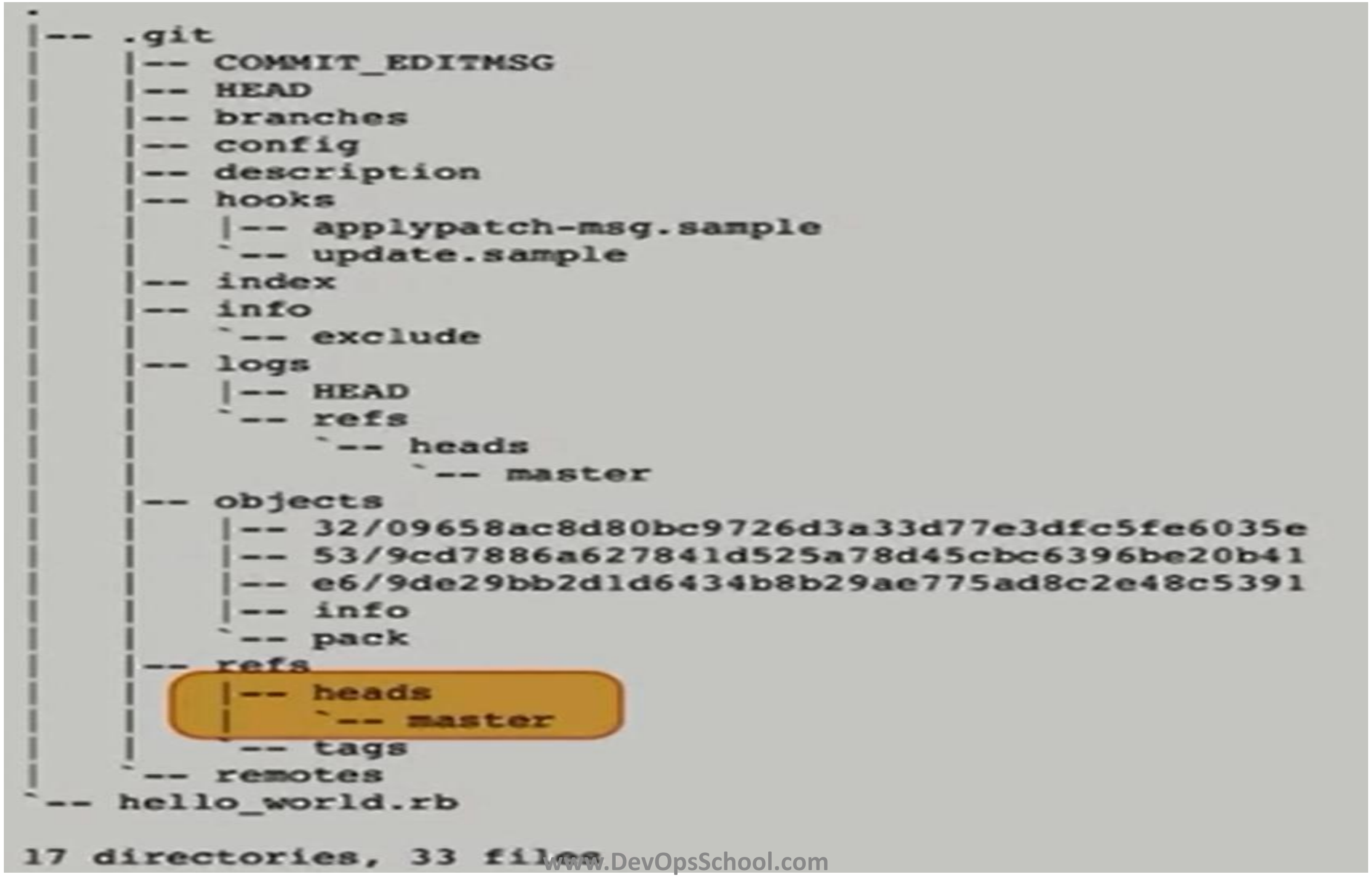
# Programmers and Developers

- HTML, CSS, JavaScript
  - PHP, Ruby, Ruby on railes, Perl Python, ASP
  - Java, C, C++, C#, Objective C
  - Action Script, Coffee Script, Haskell, Scala, Shell Scripts
- Not as useful for tracking non-text files
  - Image, movies, music, fonts
  - Word processing files, spreadsheets, PDFs

## No network Needed

- Performing a diff
- Viewing file history
- Commiting Changes
- Merging branches
- Obtaining other revision of file
- Switching branches

## git init



```
.
|-- .git
|   |-- COMMIT_EDITMSG
|   |-- HEAD
|   |-- branches
|   |-- config
|   |-- description
|   |-- hooks
|   |   |-- applypatch-msg.sample
|   |   `-- update.sample
|   |-- index
|   |-- info
|   |   `-- exclude
|   |-- logs
|   |   |-- HEAD
|   |   `-- refs
|   |       `-- heads
|   |           `-- master
|   |-- objects
|   |   |-- 32/09658ac8d80bc9726d3a33d77e3dfc5fe6035e
|   |   |-- 53/9cd7886a627841d525a78d45cbc6396be20b41
|   |   |-- e6/9de29bb2d1d6434b8b29ae775ad8c2e48c5391
|   |   |-- info
|   |   `-- pack
|   |-- refs
|   |   |-- heads
|   |   |   `-- master
|   |   `-- tags
|   `-- remotes
`-- hello_world.rb

17 directories, 33 files
```

## The Git Repository

- .git directory
  - **Config – Repo private configuration file (.ini style)**
  - Description – Repo description
  - Hooks/* - hooking scripts
  - **Objects/* - The object repository**
  - **Refs/heads/* - branches (like “master”)**
  - **Refs/tags/* - tags**
  - **Refs/remotes/* - tracking others**
  - Logs/* - logs
  - Index – changes to commit
  - HEAD – points to one of the branches (the “current

## Objects

- Every object has a SHA1 to uniquely identify it
- Objects consist of 4 types:
  - Blobs (the contents of a file)
  - Trees (directories of blobs or other trees)
  - Commits
    - A Tree
    - Plus zero or more parent commits
  - Tags
    - An object (usually a commit)

A SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long SHA-1 produces a

# Configuring Git

- System
  - /etc/gitconfig
  - Program file\git\etc\gotconfig
- User
  - ~/.gitconfig
  - $HOME\.gitconfig
- Project

## Configuring Git…

- System
  - git config --system
- User
  - git config --global
- Project
  - git config

# Configuring Git…

- git config --global user.name "rajesh kumar"
- git config --global user.email someon@nowehre.com
- git config --list

- more .gitconfig
- git config --global core.editor "vim“
- git config --global color.ui true

# Configuring Git…

- git config --list
- git config --global user.name "rajesh kumar"
- git config --global user.email
- git config --global core.editor "vim“
- git config --global color.ui true

# git Config Priorities

Lowest

- /etc/gitconfig
- ~/.gitconfig
- .git/config

Highest

# git workflow

- Working with local repos
  - git init
    - Creates a .git in the current dir
  - git add <directory tree>
    - Adds all files (except .git)
  - git commit
    - Commits the changes (in this case initial commit)
    - Creates a branch named **master**
    - HEAD points at master
- Working with Remote Repos
  - git clone
    - Creates a git repo from an existing repo
    - All remote branches are tracked
    - Remote HEAD branch checked out as your initial master branch as well
  - git pull
  - git push

## Head

- Pointer to “tip” of the current branch in repository
- Last state of repository, what was last checked out
- Points to parent next commit
  - Where writing commits takes place

## Excersise

- Create

# Daily git tasks

- Additions
  - $ git add file
  - $ git add .
- Removal
  - $ git rm file
- Renames
  - $ git mv old new
- Status
  - $ git status
    - Tellus you differences between HEAD, index, and working directory
- Ready to commit

## SCM Operations

- Bootstrap
  - Init
  - Checkout
  - Switch branch
- Modify
  - Add, delete, rename
  - Commit
- Information
  - Status
  - Diff
- Reference
  - Tag
  - Branch
- Collaborate
  - Clone
  - Pull, fetch
  - push

## Git Help Command

# Git commands

**$ git <options> <command> <options>**

Every day use..

add branch checkout clone commit

config diff fetch gc grep

init log merge mv pull

push rebase remote reset rm

# Commit Message best practices

- Short single-line summary (less than 50 characters)
- Keep each line to less than 72 characters
- Write a commit messages in present tense, not past tense
  - "fix bugs" or "fixes bug", not "fixed bug"

# Commit Message best practices

- Bullets points are usually asterisks or hypens
- Can add “ticket tracking numbers” from bugs or support requests
- Can develop shorthand for your organization
  - “[css,js] ”
  - “bugfix: ”
  - “#24223 - ”

# Commit Message best practices

- Be clear and descriptive
  - Bad: “Fix typo”
  - Good: “Add missing > in project section of HTML
  - Bad: “Update login code”
  - Good: “Change user authentication to use Blowfish”
  - Bad: “Updates member report, we should discuss if this is right next week”

## git log

```
commit 1837f0b7056c64cef103210b07539b6313612ba3
Author: rajesh kumar <someon@nowehre.com>
Date:   Thu Dec 6 01:16:03 2012 -0800


    first commit
```

```
git log --oneline
git log --oneline --graph
git log --format=short ==
```

- git log –n 1/2/3/0
- git log --since=2012-05-05
- git log --until=2012-04-23
- git log --grep="init"
- git log head

**See the log of remote repos**

git fetch origin

git log origin/master

Eq. to

git log HEAD..origin/master

# Referring to commits

When we submit any changes to the repository,

- Git generate a checksum for each change set
  - Checksum algorithm convert data into a simple number
  - Same data always equals same checksum
- Data Integrity is fundamental
  - Changing data would change checksum
- Git uses SHA-1 hash algorithm to create checksums
  - 40 character hexadecimal string (0-9,a-f)
  - Example: 1837f0b7056c64cef103210b07539b6313612ba3

# git diff

- git diff SHA1…SHA2
- git diff HEAD~1..HEAD
- git diff HEAD~1..

## git rm filename

Remove files from the working tree and from the index

# git clean

- Remove untracked files from the working tree

## Remove untracked files from the working tree

```
git clean -f
```

# git mv filename.txt filename1.txt

## git reset

Reset the staging area to match the most recent commit, but leave the working directory unchanged. This unstages *all* files without overwriting any changes, giving you the opportunity to re-build the staged snapshot from scratch.

## git reset <file>

Remove the specified file from the staging area, but leave the working directory unchanged

## git reset --hard

Reset the staging area and the working directory to match the most recent commit. In addition to unstaging changes, the --hard flag tells Git to overwrite all changes in the working directory, too. Put another way: this *obliterates* all uncommitted changes, so make sure you really want to throw away your local developments before using it.

## git revert #

# VS

- Rm - Remove files from the workspace?
- Clean - Remove untracked files from the working tree

# git index

- The git “index” is where you place files you want committed to the git repository.
- Before you “commit” (checkin) files to the git repository, you need to first place the files in the git “index”.

- The git index goes by many names. But they all refer to the same thing. Some of the names you may have heard:
  - Index
  - Cache
  - Directory cache
  - Current directory cache
  - Staging area

# Question