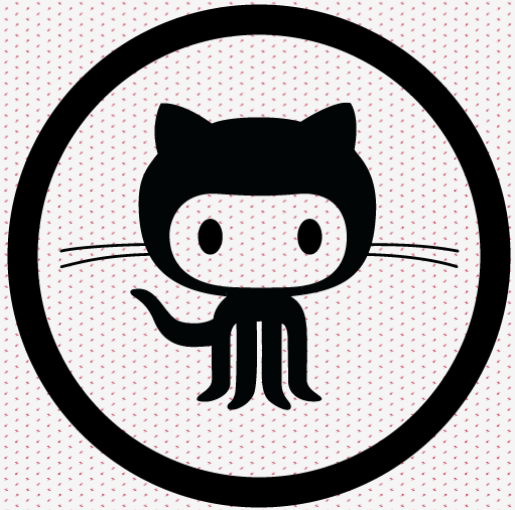
Introduction to DevOps

C431-8.2 : Students will be able to create pull and push requests using GIT and GIT Hub and also able to review the changes on GitHub

By Mr. Sulabh Tyagi

Overview

1. Install git and create a Github account
2. What is git?
3. How does git work?
4. What is GitHub?
5. Quick example using git and GitHub



Github icon

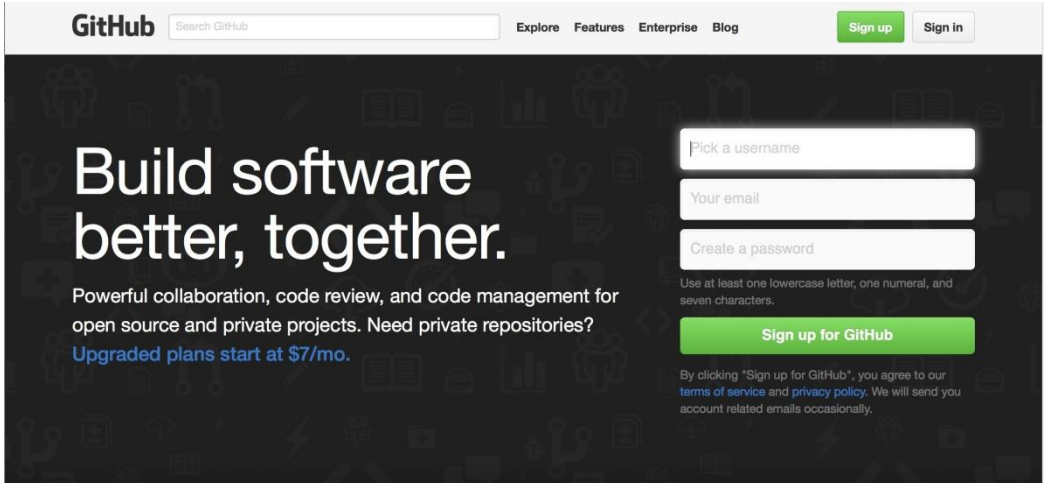
1 Install git and a create GitHub account

Install git

- **Linux (Debian)**
  - Command: sudo apt-get install git
- **Linux (Fedora)**
  - Command: sudo yum install git
- **Mac**
  - http://git-scm.com/download/mac
- **Windows**
  - http://git-scm.com/download/win

Create Github account

- www.github.com
- Free for public repositories



What is version control?

- A system that keeps logs of your changes
- Allows for collaborative development
- Allows you to know who made what changes and when
- **Allows you to revert any changes and go back to a previous state**

2 What is git?

What is version control?

- Distributed version control
- Users keep entire code and history on their location machines
  - Users can make any changes without internet access
  - (Except pushing and pulling changes from a remote server)

What is git?

- Started in 2005
- Created by Linus Torvald to aid in Linux kernel development

Git icon

What is git?

- Git isn’t the only version control system

- But (we think) it’s the best

3 How does git work?

How does git work?

- Can be complicated at first, but there are a few key concepts
- Important git terminology in following slides are blue

Key Concepts: Snapshots

- The way git keeps track of your code history
- Essentially records what all your files look like at a given point in time
- You decide when to take a snapshot, and of what files
- Have the ability to go back to visit any snapshot
  - Your snapshots from later on will stay around, too

Key Concepts: Commit

- The act of creating a snapshot
- Can be a noun or verb
  - “I commited code”
  - “I just made a new commit”
- Essentially, a project is made up of a bunch of commits

Key Concepts: Commit

- Commits contain three pieces of information:

1. Information about how the files changed from previously
2. A reference to the commit that came before it
   - Called the "parent commit"
3. A hash code name
   - Will look something like: fb2d2ec5069fc6776c80b3ad6b7cbde3cade4e

Key Concepts: Repositories

- Often shortened to ‘repo’
- A collection of all the files and the history of those files
  - Consists of all your commits
  - Place where all your hard work is stored

Key Concepts: Repositories

- Can live on a local machine or on a remote server (GitHub!)
- The act of copying a repository from a remote server is called cloning
- Cloning from a remote server allows teams to work together

Key Concepts: Repositories

- The process of downloading commits that don't exist on your machine from a remote repository is called pulling changes
- The process of adding your local changes to the remote repository is called pushing changes

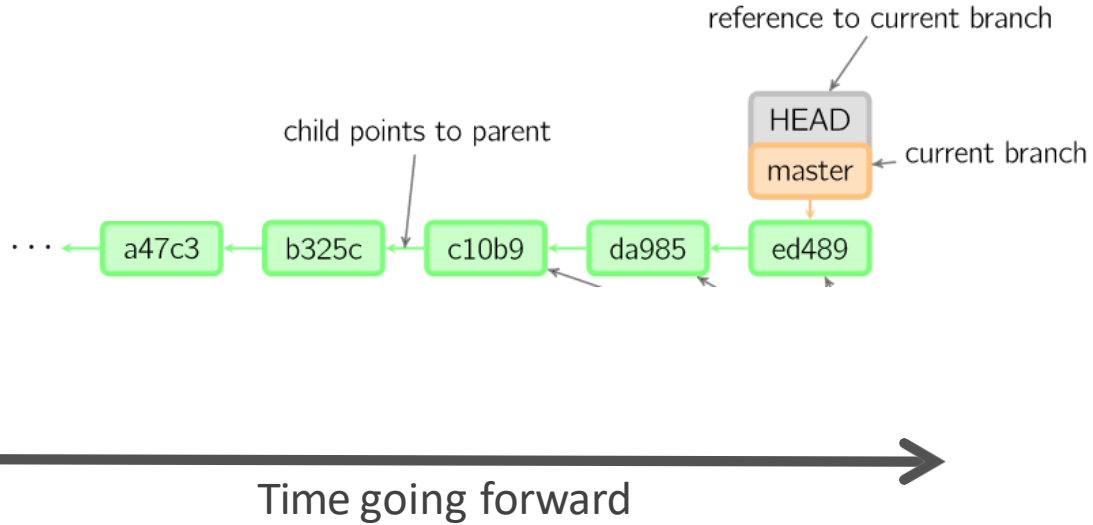
Key Concepts: Branches

- All commits in git live on some branch
- But there can be many branches
- The main branch in a project is called the master branch

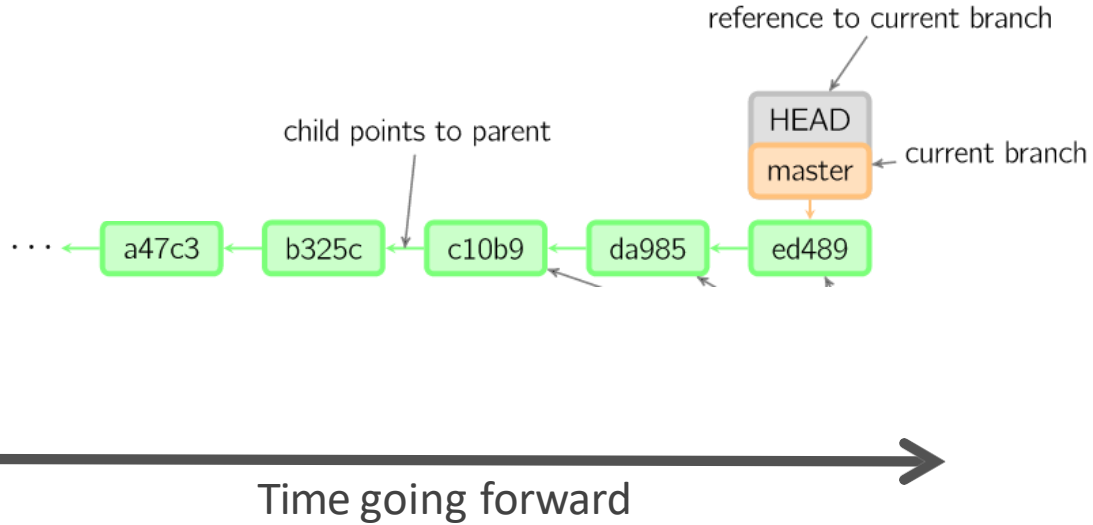
So, what does a typical project look like?

- A bunch of commits linked together that live on some branch, contained in a repository
- Following images taken and modified from:
  - http://marklodato.github.io/visual-git-guide/index-en.html

So, what does a typical project look like?

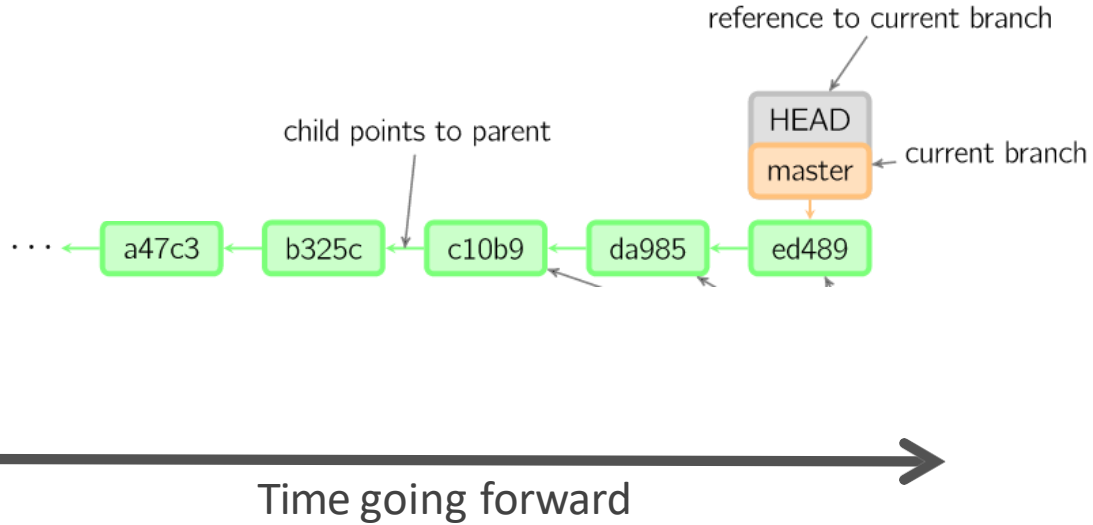


So, what is HEAD?



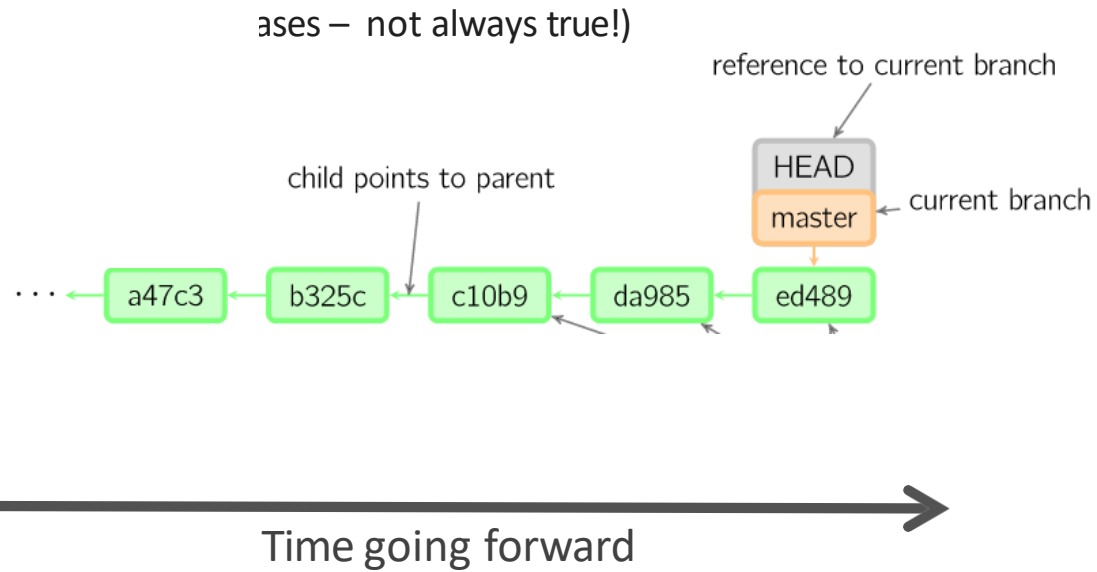
So, what is HEAD?

- A reference to the most recent commit



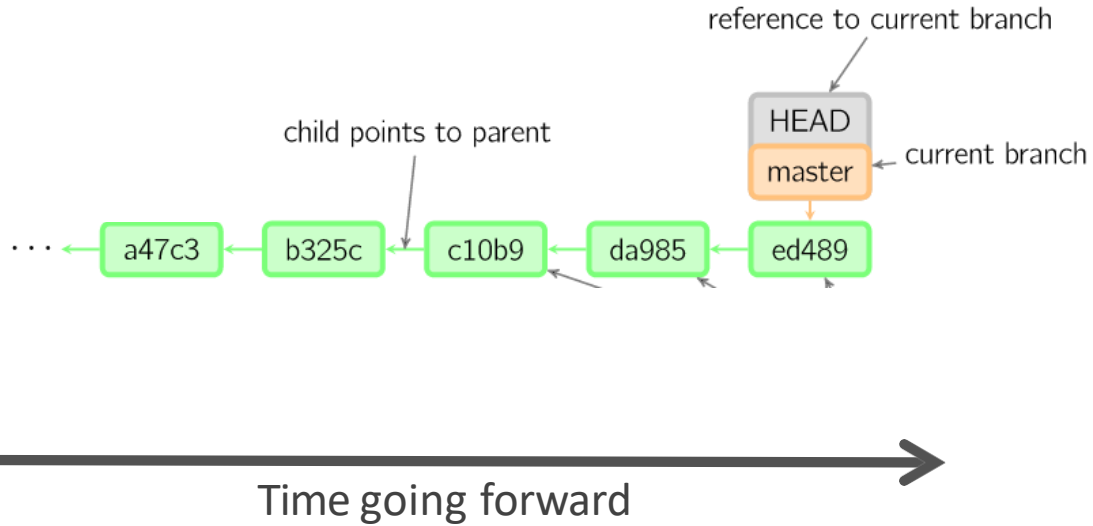
So, what is HEAD?

- A reference to the most recent commit



So, what is MASTER?

- The main branch in your project



Key Concepts: Branching off of the master branch

- The start of a branch points to a specific commit
- When you want to make any changes to your project you make a new branch based on a commit

Key Concepts: Branching off of the master branch

- In Git, branches are a part of your everyday development process.
- Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes.

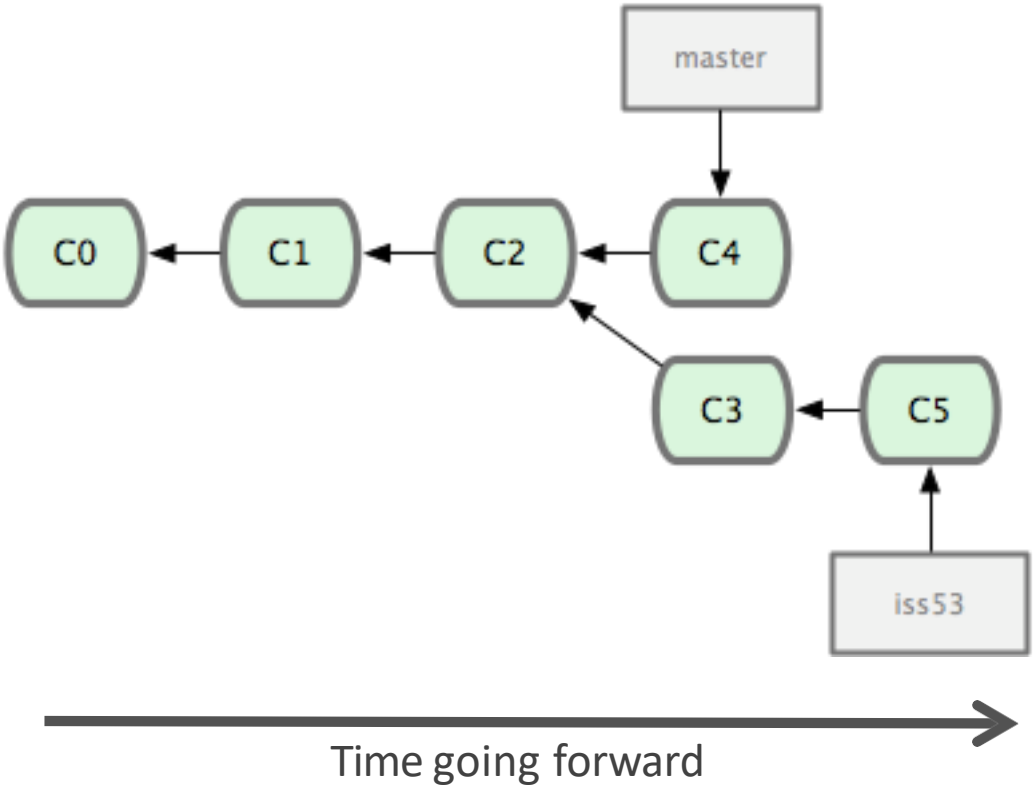
Key Concepts: Branching off of the master branch

- In Git, branches are a part of your everyday development process.
- Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes.
- This makes it harder for unstable code to get merged into the main code base, and it gives you the chance to clean up your future's history

Key Concepts: Branching off of the master branch

- In Git, branches are a part of your everyday development process.
- Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes.
- This makes it harder for unstable code to get merged into the main code base, and it gives you the chance to clean up your future's history before merging it into the main branch.

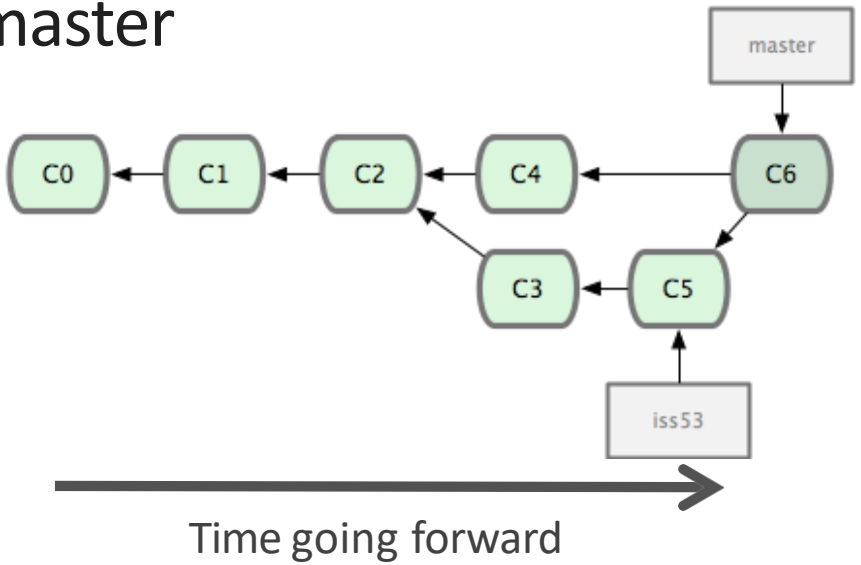
Key Concepts: Branching off of the master branch



Images from:
http://codingdomain.com/git/merging/

Key Concepts: Merging

- Once you're done with your feature, you merge it back into master



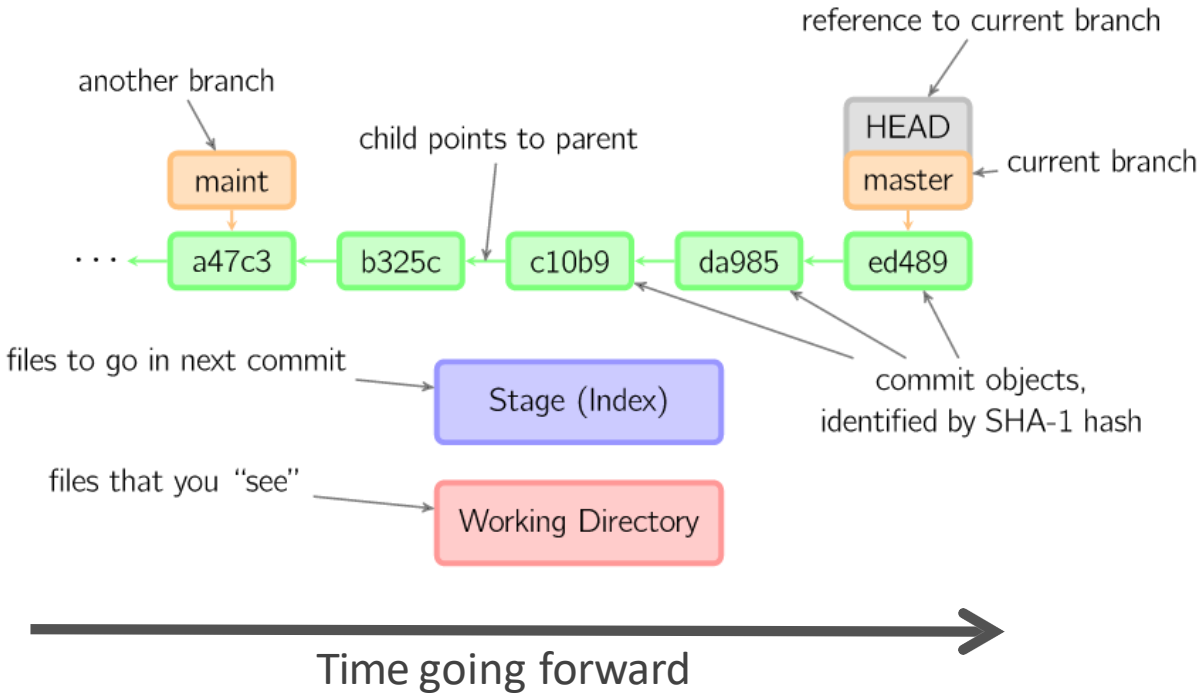
Key Concepts: How do you make a commit anyway?

- There are a lot of ‘states’ and ‘places’ a file can be
- Local on your computer: the ‘working directory’
- When a file is ready to be put in a commit you add it onto the ‘index’ or ‘staging’
  - Staging is the new preferred term – but you can see both ‘index’ and ‘staging’ being used

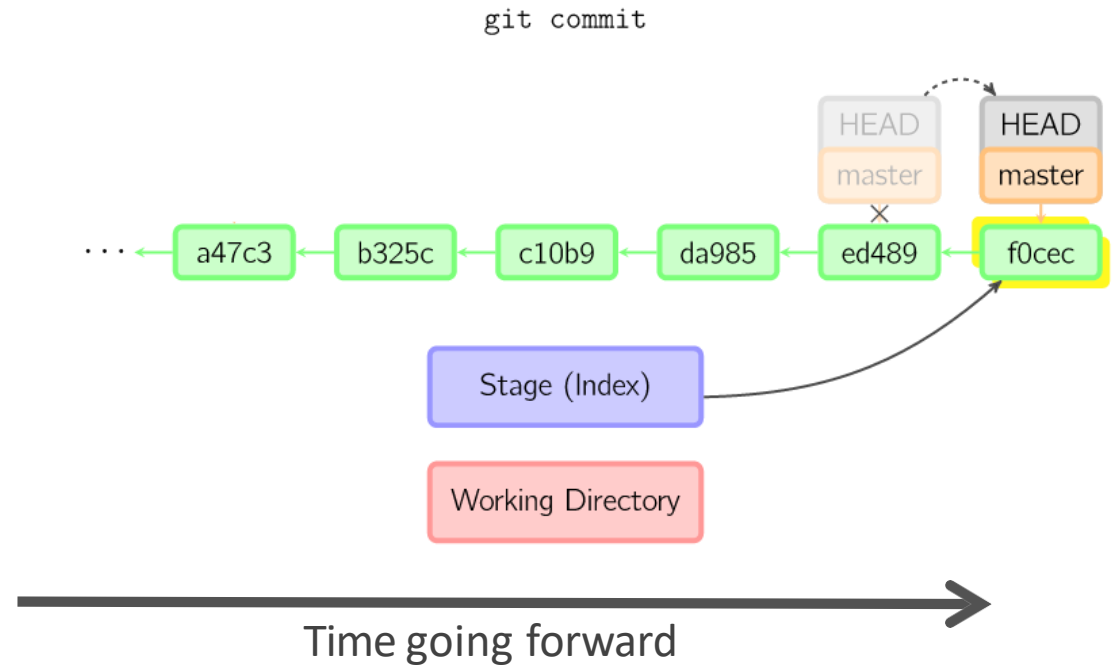
Key Concepts: How do you make a commit anyway?

- The process:
  - Make some changes to a file
  - Use the ‘git add’ command to put the file onto the staging environment
  - Use the ‘git commit’ command to create a new commit’

Key Concepts: How do you make a commit anyway?



Key Concepts: How do you make a commit anyway?



4 What is GitHub?

What is GitHub?

- www.github.com
- Largest web-based git repository hosting service
  - Aka, hosts 'remote repositories'
- Allows for code collaboration with anyone online
- Adds extra functionality on top of git
  - UI, documentation, bug tracking, feature requests, pull requests, *and more!*

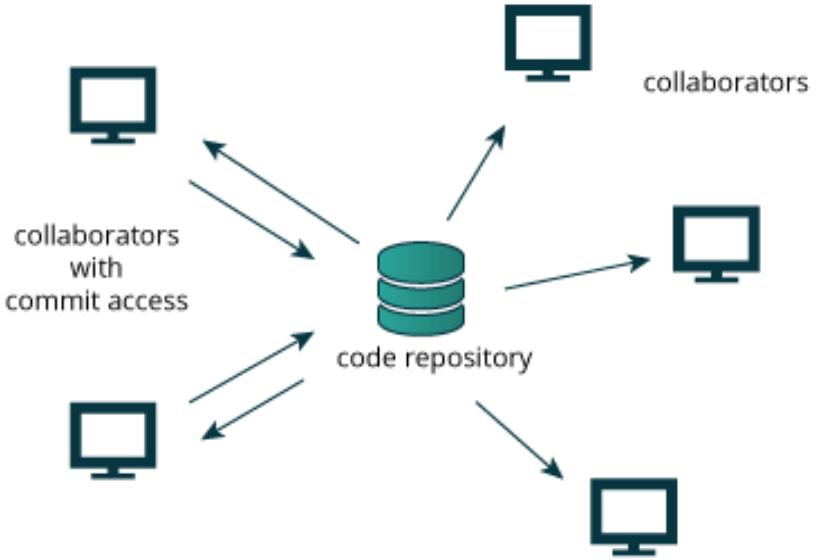
Octocat!

What is GitHub?

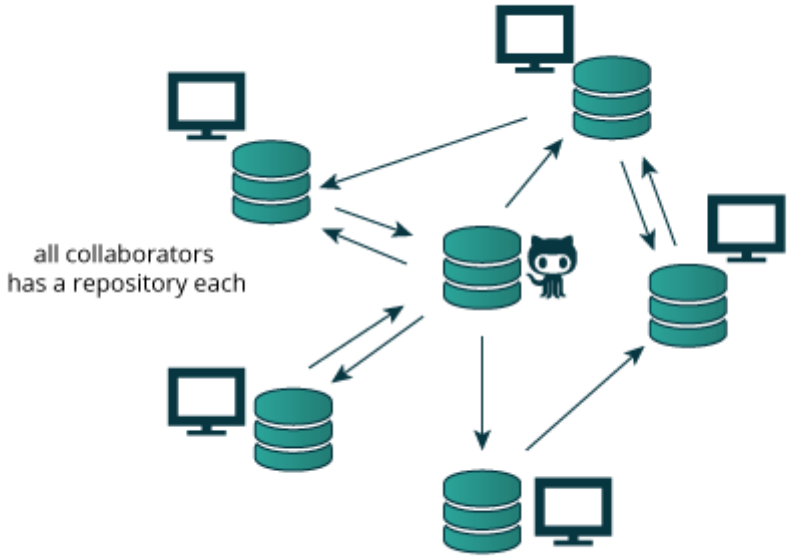
- Founded in 2008
- Also has an Enterprise edition for businesses

Octocat!

Centralized VCS



Distributed VCS



What is CD?

- Continuous Delivery is a practise to continuously (any-time) release software
- Code changes are continuously built, tested & pushed to a non-production environment by using automation tools
- Software delivery cycles are more rapid and effective

Before Jenkins Pipeline

Over the years, there have been multiple Jenkins pipeline releases including, **Jenkins Build flow**, **Jenkins Build Pipeline plugin**, **Jenkins Workflow**, etc.

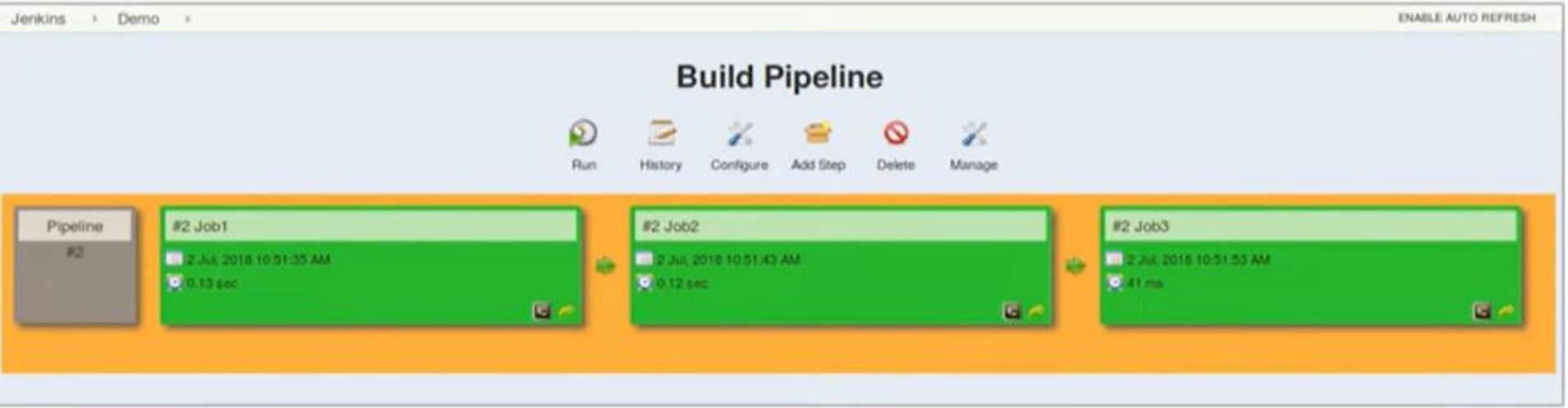
What are the key features of these plugins?

- Represent multiple Jenkins jobs as one pipeline
- What do these pipelines do?

  These pipelines are a collection of Jenkins jobs which trigger each other in a specified sequence

Build Plugin Pipeline Example

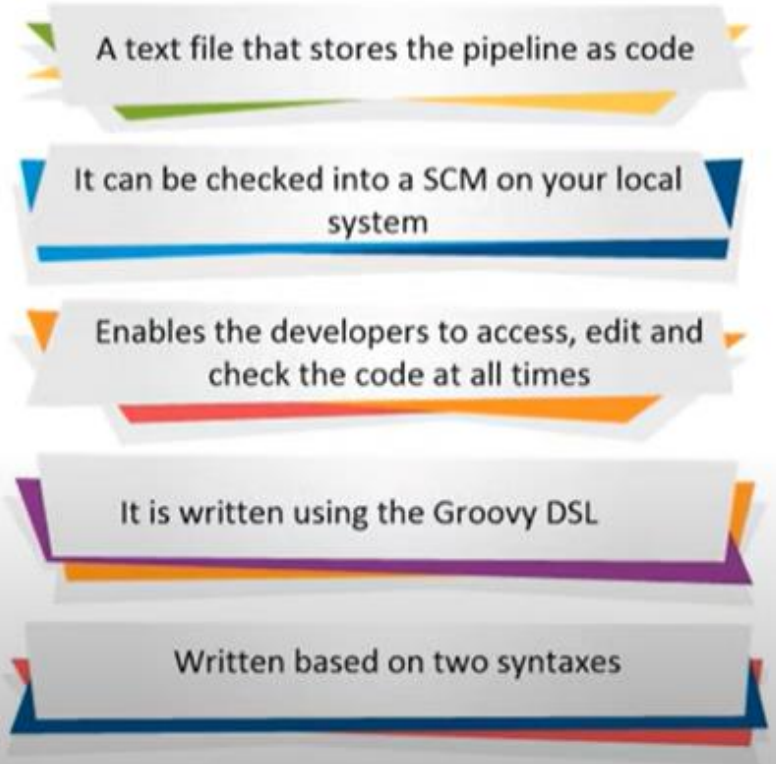
- **3 jobs**: Job1 → building, Job2 → testing the application and Job3 → deployment
- Chain these jobs together & run using Build Pipeline Plugin (better for small deployment)



What Is A Jenkins Pipeline?

- **Jenkins pipeline** is a single platform that runs the entire *pipeline as code*
- All the standard jobs defined by Jenkins are manually written in one script and they can be stored in a VCS
- Instead of building several jobs for each phase, you can now code the entire workflow and put it in a Jenkinsfile

What Is A Jenkinsfile?



Two Ways Of Writing Jenkinsfile

DECLARATIVE PIPELINE

- Recent feature
- Simpler groovy syntax
- Code is written locally in a file and is checked into a SCM
- The code is defined within a 'pipeline' block

SCRIPTED PIPELINE

- Traditional way of writing the code
- Stricter groovy syntax
- Code is written on the Jenkins UI instance
- The code is defined within a 'node' block

Pipeline Concepts

Pipeline: A user defined block which contains all the stages. It is a key part of declarative pipeline syntax.

Note: Stages are explained in the following slides

```
pipeline {

}
```

Node: A node is a machine that executes an entire workflow. It is a key part of the scripted pipeline syntax.

```
node {

}
```

Pipeline Concepts

Agent: instructs Jenkins to allocate an executor for the builds. It is defined for an entire pipeline or a specific stage.
It has the following parameters:

- *Any*: Runs pipeline/ stage on any available agent
- *None*: applied at the root of the pipeline, it indicates that there is no global agent for the entire pipeline & each stage must specify its own agent
- *Label*: Executes the pipeline/stage on the labelled agent.
- *Docker*: Uses docker container as an execution environment for the pipeline or a specific stage.

Pipeline Concepts

Stages: It contains all the work, each stage performs a specific task.

```
pipeline {
        agent any
        stages {
                stage ('Build') {
                        ...
                }
                stage ('Test') {
                        ...
                }
                stage ('QA') {
                        ...
                }
                stage ('Deploy') {
                        ...
                }
                stage ('Monitor') {
                        ...
                }
        }
}
```

Steps: steps are carried out in sequence to execute a stage

```
pipeline {
        agent any
        stages {
                stage ('Build') {
                        steps {
                                echo 'Running build phase...'
                        }
                }
        }
}
```