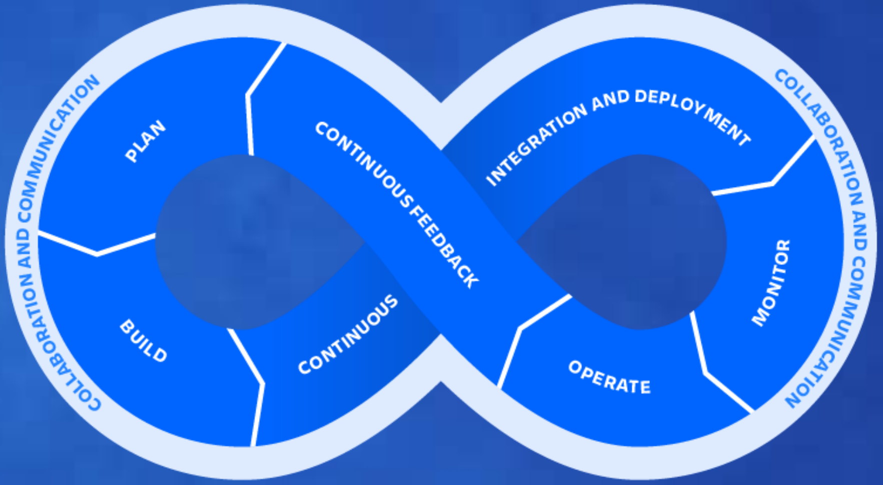

PLAN
CONTINUOUS FEEDBACK
INTEGRATION AND DEPLOYMENT
MONITOR
OPERATE
CONTINUOUS
BUILD
COLLABORATION AND COMMUNICATION
COLLABORATION AND COMMUNICATION

Session 3.
Version Control with Git

Ram N Sangwan

Agenda

- Version control and Git
- Install Git
- Common commands in Git
- Working with Remote Repositories
- Branching and Merging in Git
- DevCs support for Git

Version Control Systems

- Version control is all about managing multiple versions of documents, programs, web sites, etc.
  - Almost all “real” projects use some kind of version control
  - Essential for team projects, but also very useful for individual projects
- Some well-known version control systems are CVS, Subversion, Mercurial, and Git
  - CVS and Subversion use a “central” repository; users “check out” files, work on them, and “check them in”
  - Mercurial and Git treat all repositories as equal
- Distributed systems like Mercurial and Git are newer and are gradually replacing centralized systems like CVS and Subversion

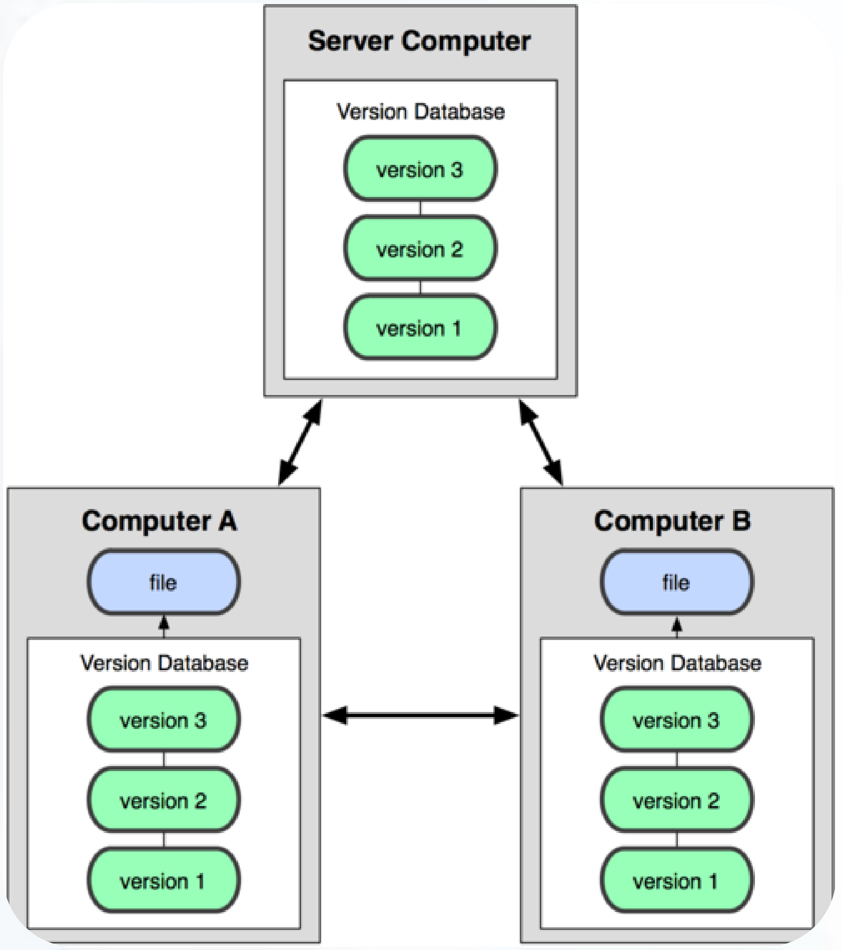
Git

- Created by Linus Torvalds, creator of Linux, in 2005
  - Came out of Linux development community
  - Designed to do version control on Linux kernel
- Goals of Git:
  - Speed
  - Support for non-linear development (thousands of parallel branches)
  - Fully distributed
  - Able to handle large projects efficiently

    (A "git" is a cranky old man. Linus meant himself.)

Distributed VCS - Git

- In git, mercurial, etc., you don't "checkout“ from a central repo
  - you "clone" it and "pull" changes from it
- Your local repo is a complete copy of everything on the remote server
  - yours is "just as good" as theirs
- Many operations are local:
  - check in/out from local repo
  - commit changes to local repo
  - local repo keeps version history
- When you're ready, you can "push" changes back to server



Download and install Git

http://git-scm.com/downloads

Or on your Oracle Linux System

```
# yum install git -y
```

Introduce Yourself to Git

- Enter these lines (with appropriate changes):

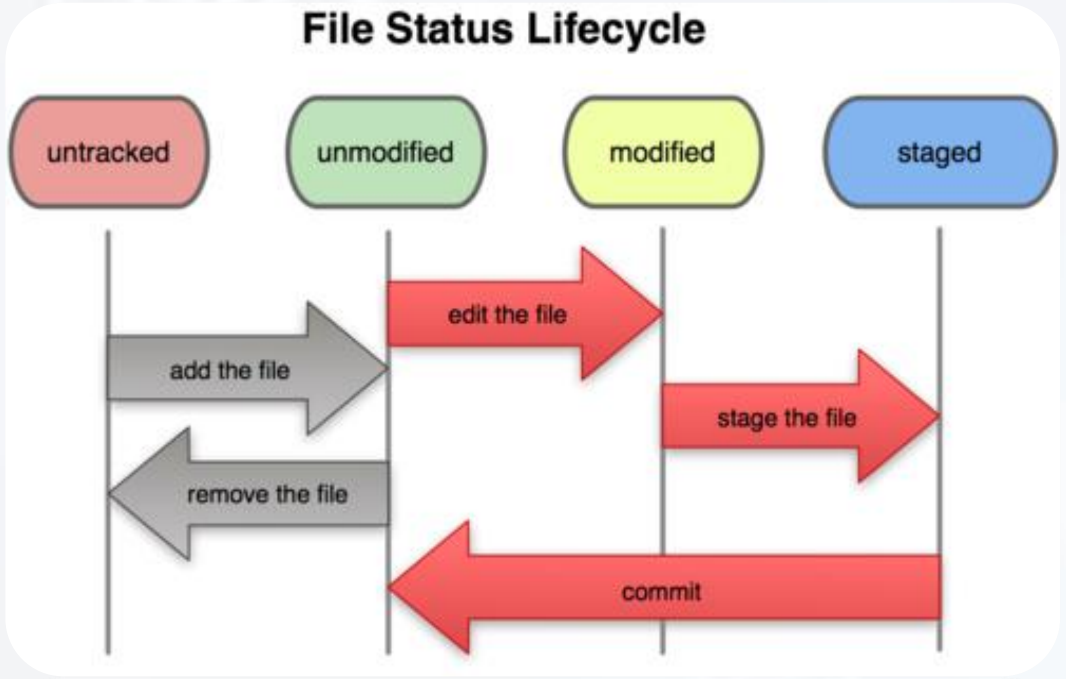
  git config --global user.name “Sangwan Ram"

  git config --global user.email ramnsangwan@gmail.com

- You only need to do this once
  If you want to use a different name/email address for a particular project, you can change it for just that project
  - Use the above commands, but leave out the ***--global***

Basic Git workflow

- Modify files in your working directory.
- Stage files, adding snapshots of them to your staging area.
- Commit, which takes the files in the staging area and stores that snapshot permanently to your Git directory



Create and fill a repository

- Creates the repository (a directory named .git). You seldom (if ever) need to look inside this directory. Initialize it.

  cd <directory you want to use>

  git init

- Add files to the repository

  git add .

  or

  git add newFile1 newFolder1 newFolder2 newFile2

- Commit your changes

  git commit –m "Initial commit"

Clone a Repository from elsewhere

git clone URL
git clone URL mypath
git clone git://github.com/rest_of_path/file.git

- All repositories are equal
  - But you can treat some particular repository (such as one on Github) as the "master" directory
- Typically, each team member works in his/her own repository, and "merges" with other repositories as appropriate

Important Git commands

Command	Description
git clone url [dir]	copy a Git repository so you can add to it
git add file	adds file contents to the staging area
git commit	records a snapshot of the staging area
git status	view the status of your files in the working directory and staging area
git diff	shows diff of what is staged and what is modified but unstaged
git help [command]	get help info about a particular command
git pull	fetch from a remote repo and try to merge into the current branch
git push	push your new branches and data to a remote repository
others: init, reset, branch, checkout, merge, log, tag	

init and the .git repository

- When you said ***git init*** in your project directory, or when you cloned an existing project, you created a repository
  - The repository is a subdirectory named .git containing various files
  - The dot indicates a “hidden” directory
  - You do not work directly with the contents of that directory; various git commands do that for you.

Choose an editor

- When you “commit,” git will require you to type in a commit message
- For longer commit messages, you will use an editor
- The default editor is vim
- To change the default editor:
  git config --global core.editor /path/to/editor
- You may also want to turn on colors:
  git config --global color.ui auto

An Example Workflow

```
$ vi devops.txt
$ git status
    no changes added to commit
    (use "git add" and/or "git commit -a")
$ git status -s
    M devops.txt
$ git diff
    diff --git a/devops.txt b/devops.txt
$ git add devops.txt
$ git status
    # modified: devops.txt
$ git diff --cached
    diff --git a/devops.txt b/devops.txt
$ git commit -m "Created new text file"
```

Branching and Merging

- Git uses branching heavily to switch between multiple tasks.
- To create a new local branch:

  git branch name
- To list all local branches: (* = current branch)

  git branch
- To switch to a given local branch:

  git checkout branchname
- To merge changes from a branch into the local master:

  git checkout master

  git merge branchname

Merge Conflicts

- The conflicting file will contain <<< and >>> sections to indicate where Git was unable to resolve a conflict:

```
<<<<<<< HEAD:index.html
<div id="footer">todo: message here</div>
=======
<div id="footer">
thanks for visiting our site
</div>
>>>>>>> SpecialBranch:index.html
```

branch 1's version

branch 2's version

- Find all such sections, and edit them to the proper state (whichever of the two versions is newer / better / more correct).

DevCS Support for Git

- A project uses Git repositories hosted on Oracle Cloud to store and version control your application's source code files.
- After you create an Oracle Developer Cloud Service project, you'll want to create a Git repository to upload your application files.
- To upload your files to the project Git repository,
  - you must create a Git repository in the project,
  - clone the created project Git repository to your local computer, populate it with the application files,
  - add and commit the files to the local Git repository,
  - and then push the commits of the local Git repository to the project Git repository.

The End

When I say I hate CVS with a passion, I have to also say that if there are any SVN [Subversion] users in the audience, you might want to leave. Because my hatred of CVS has meant that I see Subversion as being the most pointless project ever started. The slogan of Subversion for a while was "CVS done right", or something like that, and if you start with that kind of slogan, there's nowhere you can go. There is no way to do CVS right.

--Linus Torvalds

Thank You