# GIT

=================

# What is VCS. Why we need VCS.
VCS features.

--> Refer 'ProGit' for official documentation.

#
GIT Installation (Ubuntu):

```
$ sudo apt-get update
$ sudo apt-get install git
```

Verify Installation:

```
    which git
    git --version
```

GIT Uninstallation:

```
$ sudo apt-get remove git
```

#
Git Architecture
- end-to-end git work-flow

#
Staging Index/Stage

- Skip staging

```
    $ git commit -am "submit all pending changes"
```

Note: If you want to skip the staging, you need to commit all pending changes.
For new file, you have to go throgh the 'stage' process.

# Show all the files that are modified as part of a commit (with content)

```
git show <SHA>
git show b85a6e123
```

#
- Git Commit structure

SHA value / commit ID
User & email
Date & time stamp
Commit message

# How SHA is generated?
SHA / Version / Revison / Commit Id

--7:00 AM weekend batch--
1. VM creations
2. Vim editor shortcuts

#
Creating remote repository in github
===============
1. create an account in github.com
   URL: https://github.com

2. login github.com with your credentials.
   click on "new" --> give a name "flipkart-ecomerce" --> "create reopository"

3. copy the repo URL from Github:
   https://github.com/nageshvkn/flipkart-ecomerce.git

4. Clone the source code from remote repository using 'git clone' command
   git clone https://github.com/nageshvkn/flipkart-ecomerce.git

5. cd "flipkart-ecomerce" and observe ".git" folder. ".git" is called as "Local Reposiotory".

6. Create some sample code and submit the code to remote repo.
```
--> touch Login.java
--> git add Login.java
--> git commit Login.java -m "login module code"
--> git push

--> git log Login.java
```

#
Setting up mandatory configurations:

==================================================
```
$ git config --global user.name "Nageswara Rao P"
$ git config --global user.email "nageshvkn@gmail.com"
```

Check the configurations using below command
```
$ git config --list
```

Git stores all configurations in below file
"$USER_HOME/.gitconfig"

Save Git user's password
```
$ git config --global credential.helper store
```

# Understand Git Jargon.

- Remote Repository
- Working Directory
- Local Repository
- Stage/"Staging Index"
- SHA

# History
```
        $ git log Login.java
        $ git log
```

#Qn.
Show me all commits made by user "Ram" AND has a commit message "math function" in it.

```
$ git log --author="Ram" --grep="math function"
```

# GIT Commands

1.
# See the content change of a file which is in 'source' area
```
$ git diff Login.java
```

# See the content change of a file which is in 'stage' area
```
$ git diff --staged Login.java
```

# See the content change of a file after the commit
$ git show 123abc456

2. Deleting a file

```
    A.) git rm OMS.java
        git commit OMS.java -m "comment"
        git push
```

3. Renaming a file/folder

```
A.) git mv Login.java Login1.java
        git commit -m "rename Login"
```

Note:
Git will carry the history of old file to new file. To check complete history..

```
$ git log --follow Login1.java
```

# Undoing the changes:
undo/revert local changes from source area

```
$ git checkout -- LoginWeb.java
```

Unstage the changes from STAGE area

```
$ git reset HEAD LoginWeb.java
```

# BRANCHING
A. What is a branch?
B. Why and When we create a branch?
C. Branching Stratogies

# List all the branches

```
$ git branch
```

# Creating a new branch

```
$ git branch dev_1.2.3
```

# Push new branch to remote repository

```
$ git push origin dev_1.2.3
```

# Switching from one branch to another
```
        $ git checkout dev_1.2.3
```

# Creating and switching to a newly created branch
```
        $ git checkout -b dev_1.2.4
```

# How do you clone a remote repository with a particular branch as default
```
        $ git clone -b dev_1.2.4 https://github.com/nageshvkn/flipkart899.git
```

# List all remote branches
```
        $ git branch -r
```

# Deleting a branch
```
          $ git branch -d dev_1.2.3
          $ git push -d origin dev_1.2.3
```

Note: How do you restore a branch? how do you identify the sha value. more about 'reflog'
???

Merging:
=========
MERGING:
=================

# List all the branches
```
        $ git branch
```

# Create a new branch
```
        $ git branch dev_1.2.3
```

# Take a file which is common in master and dev_1.2.3 branch.
   ex: Login.java

# Make sure Login.java has below content in master and dev_1.2.3 branch
```
        Login.java
                line#5 --> int a = 10;
```

# To create a conflict situation, we need to modify same line in two branches with different content.

```
ex: Login.java (master)
            line#5 --> int a =100;

    Login.java (dev_1.2.3)
            line#5 --> int a = 200;
```

# Now merge the code from dev_1.2.3 to master branch
Before merge, make sure that you are on target branch i.e master (source branch: dev_1.2.3)

```
$ git checkout master
```

# You have master code now, since currently checkout branch is master. Now bring/merge the changes from dev_1.2.3 to master by running below command.
Git merge command merges the changes from dev_1.2.3 to master.

```
$ git merge dev_1.2.3
```

# Run git status command to list conflict file

#
Resolve the conflict and commit the merge

# Run git push command to move the merge to Remote.

# What is Conflict:
If two users modify the same file in source and target branches and if the same line has different content, git can't decide which user's code it has to take. we call this situation as conflict.

# How do you resolve the conflict:
- Open the conflict file--> remove conflict markers--> select the right content based on the discussion with developers
- git add
- git commit (after compilation and some z sanity testing)
- git push

#
git remote
  PULL
  FETCH

  PUSH
  CLONE

NOTES:
SVN & Git differences:
=========================
- Centralized VCS -   Ex: CVS/SVN/Clearcase/Perforce
- Distributed VCS -   Ex: GIT/Bitkeeper

1.
SVN is a centralised version control system. In svn, when a user checkoutâ€™s the repository, he will get only the source code. Repository is not distributed to the user. In SVN multiple users connect to a single/centralized server. If that server goes down, no user will be able to work.

Where as Git is distributed VCS. In the sense, when user clones the repository, he will get source code as well as complete repository on his local disk. since repository is distributed across multiple users and each user has their own local repository, user will be able to submit the changes locally without connecting to the master server. (i.e he can also work offline)

2.
Git has staging area. Users can leverage this staging area to stage/save all the changes related to some issue/bug and commit everything at once into local repository. hence, all the changes will be recorded as part of single sha/commit. user's get an opportunity to logically group the changes and commit as single changeset. this enables better tracking.

3.
Since git is distributed in nature, almost all transactions happen locally in the client machine and hence less overhead on the server so git is faster.

3A.
Users are free to create any no. branches in Git. where as in svn, all branches reside

in server side, so may cause svn to be overloaded with multiple requests.

4.
Git seems to be good with merging.

5.
Git stores each change as SHA values but SVN uses simple numbers as revisions.