==========

Phase-1

==========

# What is VCS. Why we need VCS.

VCS features.

#

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 Diagram

------------------------------------------------------

. (Souce area) | ..(Stage) | .git (Local Repository) |

| | |

------------------------------------------------------

- Centralized VCS - Ex: CVS/SVN/Clearcase/Perforce

- Distributed VCS - Ex: GIT/Bitkeeper

=========

Phase-2

=========

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 repository" --> "create reopository"

3. copy the repo URL from Github:

https://github.com/swapnilmahajan-git/DevOps\_Classes.git

4. Clone the source code from remote repository using 'git clone' command

$ git clone https://github.com/anuragmah-git/Devops-March-batch-1.git

5. $ cd Devops-March-batch-1

6. Create some sample code and submit the code to remote repo.

$ vim Login.java

$ git statistics

$ git add Login.java

$ git commit Login.java -m "login module code"

$ git push

#

Git Commit structure

commit id

User/Email

Timestamp

Commit message

#

SHA / Version / Revison /commit id

#

Setting up mandatory configurations:

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

$ git config --global user.name “yash"

$ git config --global user.email "abc@gmail.com"

$ git config --global credential.helper cache

Check the configurations using below command

$ git config --list

Git stores all configurations in below file

"cd /home/swapnil/.gitconfig"

=========

Phase-3

=========

# Understand Git Jargon.

- Remote Repository --- github server

- Working Directory ---- (.)

- Local Repository ----- (.git)

- Stage/"Staging Index" ---- (..)

#It is remote repo url for clone the files.

$ git remote -v

#If you want downlod remote repository then apply this commnad (It is first time activity).

$ git clone <repo-url>

#If you want to download modified content from remote repo then apply this command, after git pull your modified changes comes in source area.

$ git pull

#If you want to download modified content from remote repo then apply this command, But you dont want to merge modified content in your source area then apply this command

$ git fetch

# Skip staging

git commit -am "submit all pending changes"

# Creating up Local repository

1. create/initialize a local repository with 'git init' command

$ mkdir <create-folder>

$ cd <folder-name>

$ git init

(Note - Create same folder name repositry in the github server)

$ git remote add <folder-name> <repo-url>

$ git push <folder-name>

$ git push --set-upstream <folder-name> master

2. Observe the local repository

$ ls -a

3.create some source code.. ex: vim Login.java

4.Adding the file to STAGE.

$ git add <filename>

5.Commiting the file to local repository

$ git commit <filename> -m "login module code"

6.Check the log of Login.java using log command.

$ git log <filename>

========

Phase-4

========

# History - Deep Dive

$ git log <filename>

$ git log

Options:

- git log -n2

- git log --author="swapnil"

- git log --since="2016/12/13"

- git log --until="2016/12/13"

- git log --grep="DB code"

- git log --oneline (if we want to see logs on high level that time we can use this command ie. only commit ids and msg)

#Qn.

Show me all logs committed by user "Ram" AND has a commit message "math function" in it.

$ git log --author="Ram" --grep="math function" -n2

=======

Phase-5

=======

# GIT Commands

1.

$ git status

$ git diff <filename>

$ git diff --staged <filename>

$ git diff 089871935..fc894a6bf6d2e (if we want to see difference between two files)(use commit ids).

2. Deleting a file

A.) $ git rm <filename>

$ git commit <filename> -m "comment"

$ git push

4. Renaming a file/folder

A.) $ git mv Login.java Login1.java

$ git commit -m "rename Login"

$ git push

#

Carry the history for new file from old file

- $ git log --follow <new-name>

=======

Phase-6

=======

# Undoing the changes:

# undo/revert local changes from source area in exits files. ie. present in remote repo.

$ git checkout <filename>

# Unstage the changes from STAGE area

$ git reset HEAD <filename>

# Move changes from .git to source area

$ git reset --mixed HEAD~1

# Move changes from .git to stage area

$ git reset --soft HEAD~1

# Remove changes from everywhere

$ git reset --hard HEAD~1

# Show all the files with content difference that are modified as part of a commit.

git show <SHA-VALUE>

# List only files that are committed as part of a revision.

$ git diff-tree --no-commit-id --name-only -r <SHA-VALUE>

=======

Phase-7

=======

# BRANCHING

A. What is a branch?

B. Why and When we create a branch?

C. Branching Stratogy overview

# List all the branches

$ git branch

# Creating a new branch

$ git branch new\_branch\_name

# List all remote branches

$ git branch -r

# Switching to a newly created branch

$ git checkout new\_branch\_name

# Creating and switching to a newly created branch

$ git checkout -b new\_branch\_name

# First push newly created branch into remote repo

$ git push --set-upstream <repository-name> <branch-name>

or

$ git push -u origin --all

# List all remote branches

$ git branch -r

# Merge files from source branch to target branch

$ git merge <source-branch-name>

# HEAD

HEAD is a pointer in git which

- Always points to the latest commit in the repository

- Always points to tip of the current reposiotory

- Always points to parent of the next commit

========

Phase-8

========

Merging:

=========

# FAST-FORWARD MERGE

BEFORE MERGE:

-------------

master

|

----C1--C2---C3

|

----------C4----C5

dev\_1.2.3

AFTER MERGE:

-------------

master

|

----C1----C2---C3------C4----C5

|

dev\_1.2.3

Excercise:

===========

# List all the branches

$ git branch

# Create a new branch

$ git branch <new-branch>

# Add some changes to newely created branch i.e <new-branch>

$ Modify/Add some files

$ git add

$ git commit

# Now merge the code from <new-branch> 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 in source since currently checkout branch is master. Now bring/merge the changes from <new-branch> to master by running below command.

Git merge command merges the changes from <new-branch> to master.

Verify the chages.

$ git merge <new-branch>

# Observce the changes from dev\_1.2.3 into master branch.

# git push

MERGING W/ CONFLICT: (It is called as Real-Merge)

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

# 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;

# If your data in working or stage area then, Before entring to new branch apply below command

$ git stash

$ git checkout new-branch-name

$ git stash pop

$ git merge new-branch-name

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 new-branch 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. (<<<<<<<< >>>>>>>>>)

- git add

- git commit

- git push

#

git remote

PULL

FETCH

PUSH

CLONE

Store credentials:

$ git config credential.helper store

# If we want to delete branch locally use below command

$ git branch -d <branch-name> (after delete branch please copy that id for recover ex.(06e9029))

# If we want to delete branch locally focefully use below command

$ git branch -D <branch-name>

# If we want to delete remote branches use below command

$ git push -u origin --delete <branch-name>

or

$ git push origin --delete <branch-name>

# If you want to recover local branches then use below command

$ git branch <branch-name> <deleted-branch-id>

# How do you clone the particular branch?

$ git clone -b <branch\_name> --single-branch <git\_repo\_url>

# Cherry-pick

If we want to merge specific command from another branch in base branch that time we can use the following command

$ git cherry-pick <SHA-VALUE>

# rebase

- We have two branches master and feature

- First time in master branch two files m1,m2, on the base of this branch we created a feature branch,

now in this branch same files m1,m2 and one new file f1 this file is created in feature branch,

- but after creation of feature branch someone push one more file i.e m3 in master branch, but our feature branch base is m2.

- But, I want continue my work in feature branch so we created one more file ie. f2 in feature branch.

- so see following steps :

$ git branch

- master (m1,m2,m3)

- feature\* (m1,m2, f1,f2) feature branch is created based on m2, so feature branch base is m2.

so we need to change base of the feature branch, then use below command.

$ git rebase master (apply this command in feature branch)

- now in feature branch has five file ie. (m1,m2,m3,f1,f2) now base of feature branch is m3.

now we want feature branch data in master branch, then use below command

$ git checkout master

$ git rebase feature (feature branch changes come into master branch)

# How do you restore a deleted file? Or previous changes of a file?

1.) - $ git log --diff-filter=D --summary (get SHA-value of deleted file using this command.)

2.) - $ git checkout $commit <SHA-VALUE>~1 <filename>

3.) - $ git status (file present in stage area)