Git

Capgemini

Table of Content

What you will learn

In this course , you will learn how to use Git, the popular open-source version control software, to manage the source code for almost any project.

Learn to work with the most common Git commands, and use GitHub to clone, explore, and create templates from existing projects.

Create and manage repositories on GitHub

Compare the different states in Git and compare between branches and commits. Merge different branches of a file, handle conflicts.

Create and fork repositories on GitHub and push changes back after working after working on them locally.

Use the issue tracker in GitHub to collaborate among developers.
Rebase the files in the repository. Stash file versions and revert back to them.

Reviews the history of version control and demonstrates its fundamental concepts: check-in/checkout, forking, merging, commits, and distribution.

Software requirement

- Installation of Git Client
- https://git-scm.com/downloads

M001:GIT Fundamentals

Prerequisites

We assume that you are going to use Git to handle all levels of Java and Non-Java projects. So it will be good if you have some amount of exposure to software development life cycle and working knowledge of developing web-based and non web-based applications.

Day Wise Schedule

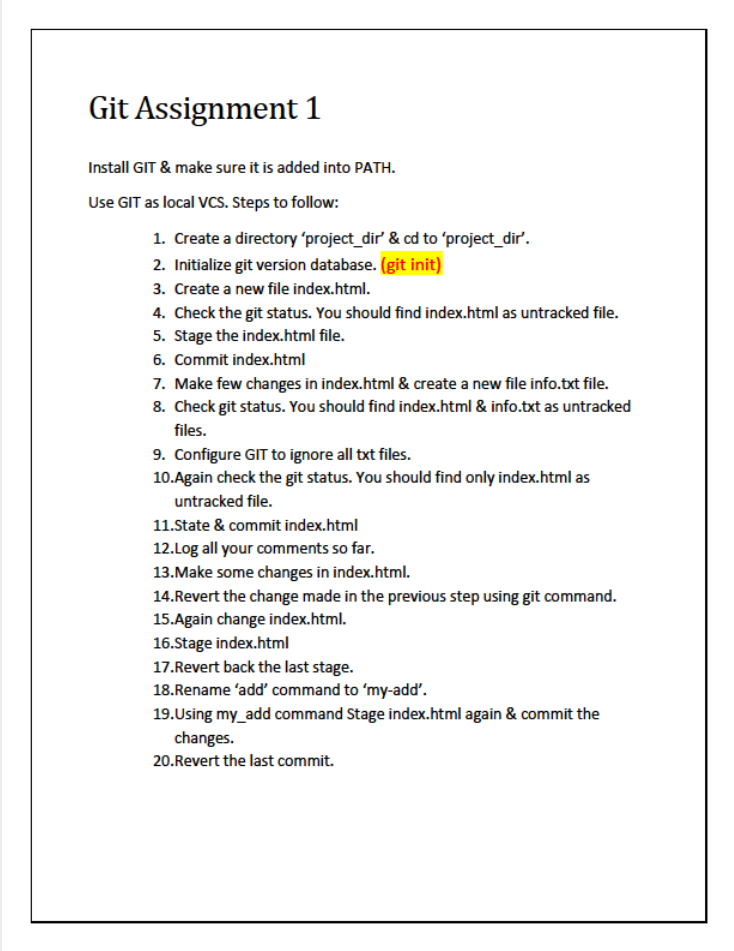
- Introduction into Version Control System
- GIT History
- Advantages of Git
- Git Installation on Windows
- Basic Git commands
- Git Repository
- Git Workflow
- Git Terminology
- Staging and commit
- Ignore files
- Changing commit
- Branching
- Merging
- Rollback
- Rebasing
- Conflict management
- Rebase vs Merge
- Working with remote GitHub repository

Useful References

- Pro-Git Book
- https://git-scm.com/book/en/v2
- Atlassian's Git tutorial
- https://www.atlassian.com/git/tutorials/
- Github's Interactive Tutorial
- https://try.github.io/
- A guide for the perplexed
- http://think-like-a-git.net/
- Git Immersion
- http://gitimmersion.com/
- Git from the bottom up (a look at exactly how Git works)
- http://ftp.newartisans.com/pub/git.from.bottom.up.pdf
- Linus Torvald's TechTalk at Google on Git (purely for entertainment)
- https://www.youtube.com/watch?v=4XpnKHJAok8
- https://git-scm.com/videos

Git – Basic Commands Assignment

double click on image



Git Assignment 1

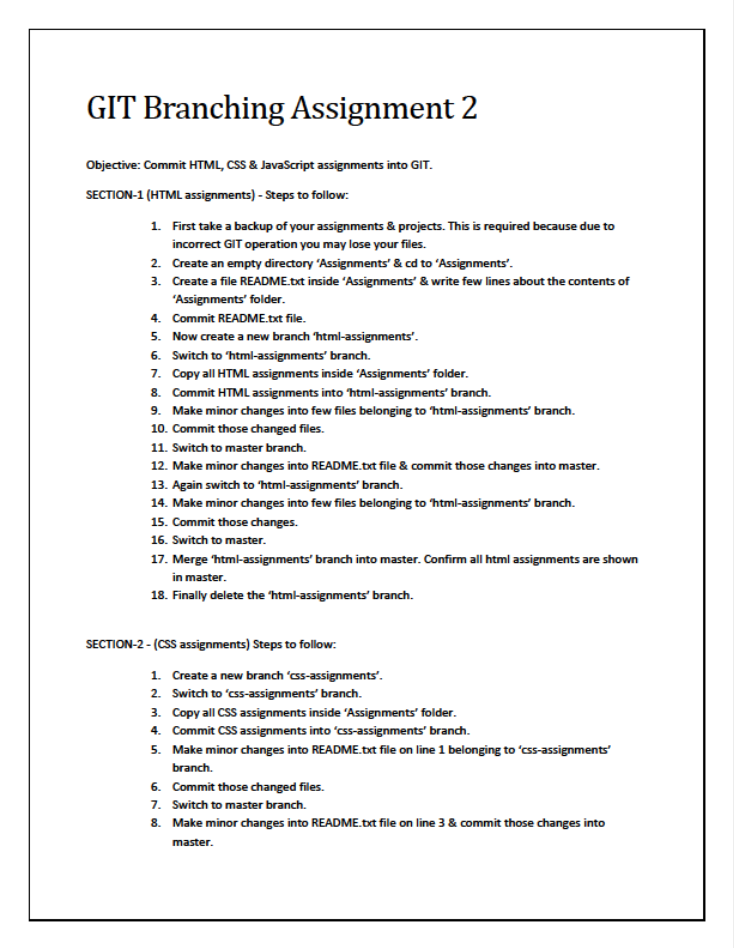
Install GIT & make sure it is added into PATH.

Use GIT as local VCS. Steps to follow:

1. Create a directory 'project_dir' & cd to 'project_dir'.
2. Initialize git version database. (git init)
3. Create a new file index.html.
4. Check the git status. You should find index.html as untracked file.
5. Stage the index.html file.
6. Commit index.html
7. Make few changes in index.html & create a new file info.txt file.
8. Check git status. You should find index.html & info.txt as untracked files.
9. Configure GIT to ignore all txt files.
10. Again check the git status. You should find only index.html as untracked file.
11. State & commit index.html
12. Log all your comments so far.
13. Make some changes in index.html.
14. Revert the change made in the previous step using git command.
15. Again change index.html.
16. Stage index.html
17. Revert back the last stage.
18. Rename 'add' command to 'my-add'.
19. Using my_add command Stage index.html again & commit the changes.
20. Revert the last commit.

Git - Branching Assignment

double click on image



GIT Branching Assignment 2

Objective: Commit HTML, CSS & JavaScript assignments into GIT.

SECTION-1 (HTML assignments) - Steps to follow:

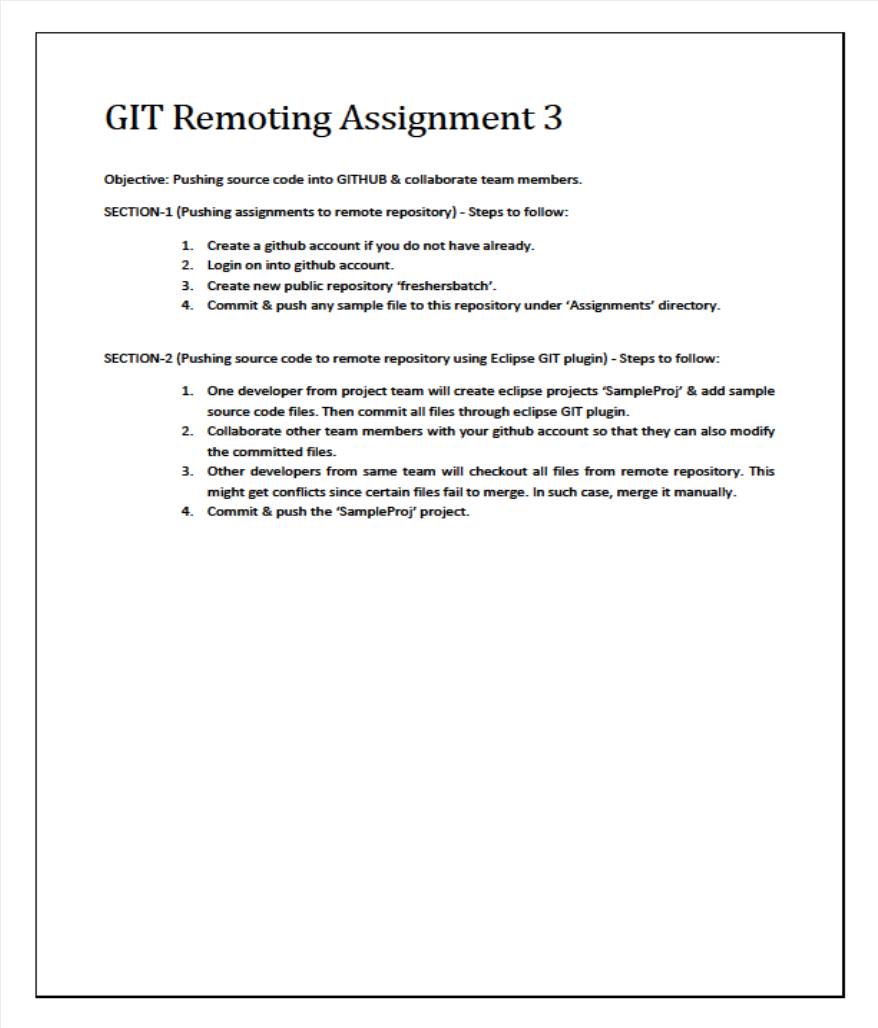
1. First take a backup of your assignments & projects. This is required because due to incorrect GIT operation you may lose your files.
2. Create an empty directory 'Assignments' & cd to 'Assignments'.
3. Create a file README.txt inside 'Assignments' & write few lines about the contents of 'Assignments' folder.
4. Commit README.txt file.
5. Now create a new branch 'html-assignments'.
6. Switch to 'html-assignments' branch.
7. Copy all HTML assignments inside 'Assignments' folder.
8. Commit HTML assignments into 'html-assignments' branch.
9. Make minor changes into few files belonging to 'html-assignments' branch.
10. Commit those changed files.
11. Switch to master branch.
12. Make minor changes into README.txt file & commit those changes into master.
13. Again switch to 'html-assignments' branch.
14. Make minor changes into few files belonging to 'html-assignments' branch.
15. Commit those changes.
16. Switch to master.
17. Merge 'html-assignments' branch into master. Confirm all html assignments are shown in master.
18. Finally delete the 'html-assignments' branch.

SECTION-2 - (CSS assignments) Steps to follow:

1. Create a new branch 'css-assignments'.
2. Switch to 'css-assignments' branch.
3. Copy all CSS assignments inside 'Assignments' folder.
4. Commit CSS assignments into 'css-assignments' branch.
5. Make minor changes into README.txt file on line 1 belonging to 'css-assignments' branch.
6. Commit those changed files.
7. Switch to master branch.
8. Make minor changes into README.txt file on line 3 & commit those changes into master.

Git –Remote Repository Assignment

double click on image



GIT Remoting Assignment 3

Objective: Pushing source code into GITHUB & collaborate team members.

SECTION-1 (Pushing assignments to remote repository) - Steps to follow:

1. Create a github account if you do not have already.
2. Login on into github account.
3. Create new public repository 'freshersbatch'.
4. Commit & push any sample file to this repository under 'Assignments' directory.

SECTION-2 (Pushing source code to remote repository using Eclipse GIT plugin) - Steps to follow:

1. One developer from project team will create eclipse projects 'SampleProj' & add sample source code files. Then commit all files through eclipse GIT plugin.
2. Collaborate other team members with your github account so that they can also modify the committed files.
3. Other developers from same team will checkout all files from remote repository. This might get conflicts since certain files fail to merge. In such case, merge it manually.
4. Commit & push the 'SampleProj' project.

M002:GIT Intermediate

Day Wise Schedule

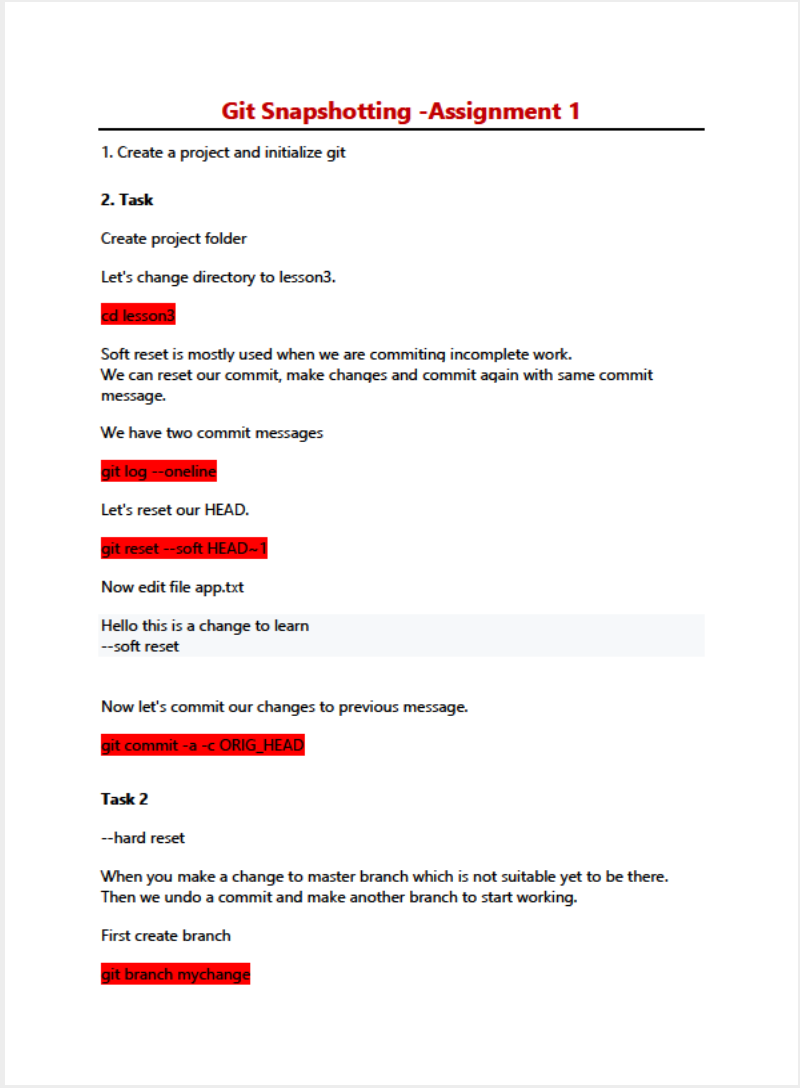
- Git with Eclipse
- Additional configuration options for Git
- Details on the Git views
- Performing Git operations in Eclipse
- Using Eclipse with GitHub
- Writing good commit messages
- Contributing to Egit
- Eclipse Git Resources
- Rebasing with Git
- Fork and Remote Repos
- References

Useful References

- Git for Eclipse
- https://www.vogella.com/tutorials/Git/article.html
- Git Rebase
- https://www.atlassian.com/git/tutorials/rewriting-history/git-rebase
- Git snapshotting
- https://git-scm.com/book/en/v2/Appendix-C%3A-Git-Commands-Basic-Snapshotting
- Advanced Git Tutorial
- https://www.youtube.com/watch?v=0SJCYPsef54
- Advanced Git For Developers
- https://www.youtube.com/watch?v=duqBHik7nRo
- Forking
- https://help.github.com/en/articles/fork-a-repo

Git - Assignment

double click on image



Git Snapshotting -Assignment 1

1. Create a project and initialize git

2. Task

Create project folder

Let's change directory to lesson3.

```
cd lesson3
```

Soft reset is mostly used when we are commiting incomplete work.
We can reset our commit, make changes and commit again with same commit message.

We have two commit messages

```
git log --oneline
```

Let's reset our HEAD.

```
git reset --soft HEAD~1
```

Now edit file app.txt

```
Hello this is a change to learn
--soft reset
```

Now let's commit our changes to previous message.

```
git commit -a -c ORIG_HEAD
```

Task 2

--hard reset

When you make a change to master branch which is not suitable yet to be there.
Then we undo a commit and make another branch to start working.

First create branch

```
git branch mychange
```

M003:GIT Advanced

Day Wise Schedule

- Git Tools
- Signing Your Work
- Searching
- Rewriting History
- Reset Demystified
- Advanced Merging
- Debugging with Git
- Submodules
- Bundling
- Replace
- Credential Storage
- Git Internals
- Plumbing and Porcelain
- Git Objects
- Packfiles

Day Wise Schedule

- Transfer Protocols
- Maintenance and Data Recovery
- Environment Variables

Useful References

- Pro-Git Book
- https://git-scm.com/book/en/v2
- Atlassian’s Git tutorial
- https://www.atlassian.com/git/tutorials/

Git - Advanced Assignment

double click on image

Advanced Git Exercises

Exercise 1 - Under The Hood of a Simple Commit

Overview

In this exercise, we'll create a simple commit, and then peek under the hood at the objects stored in our `.git` folder to gain some insight into how things work.

Exercise

1. Create a new folder and initialize it as a git repo
2. Create a file, stage it, and commit it to your new repo
3. Look at your `.git` folder, using `tree` if you have it
4. Inspect the objects in your `.git/objects` folder using `git cat-file`. See if you can find the tree, blob, and commit objects for your recent commit.
5. Look at your `.git/HEAD` and `.git/refs/heads/master` files and see if you can figure out where these references are pointing to.

Exercise 2- References

Overview

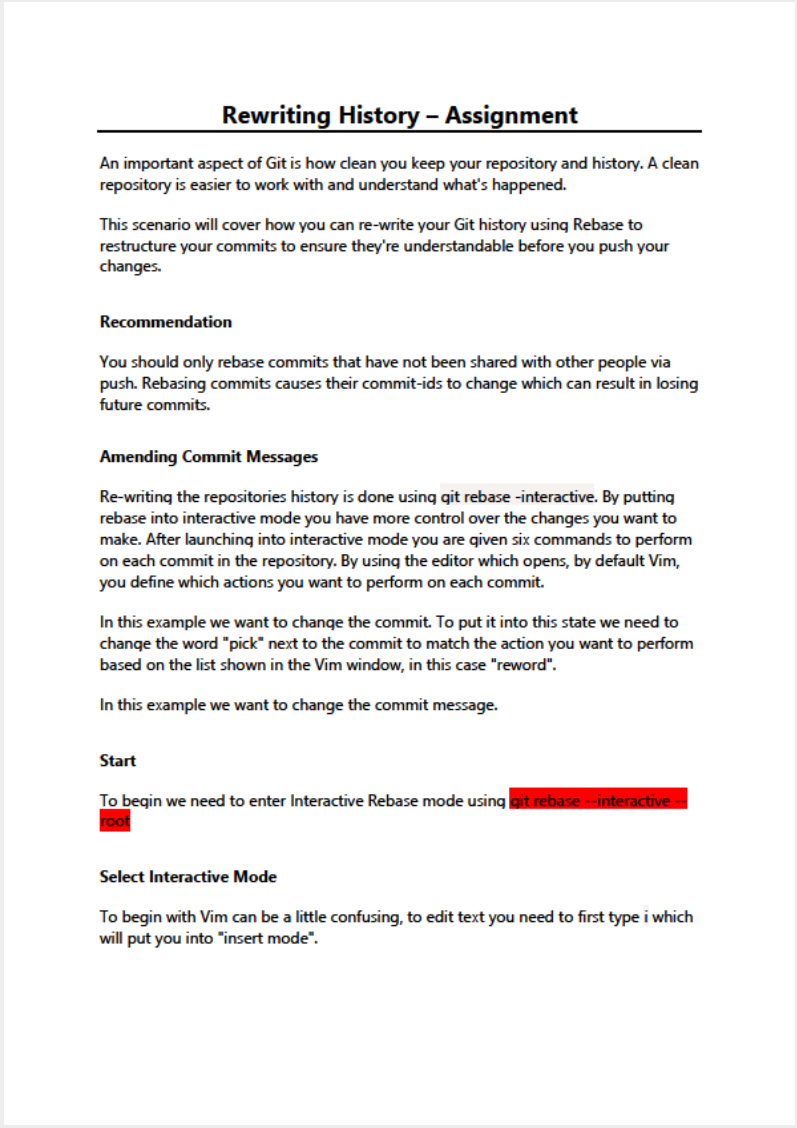
In this exercise, we'll take a look at our references (`refs`) and create some lightweight and annotated tags. Then we'll make a dangling commit from a "detached HEAD" state and learn why this isn't a great idea.

Prerequisite

advanced-git-exercises-master.zip

You should have the `advanced-git-exercises` repository cloned locally. Checkout the `exercise3` branch to begin:

Git - Assignment
double click on image



Rewriting History – Assignment

An important aspect of Git is how clean you keep your repository and history. A clean repository is easier to work with and understand what's happened.

This scenario will cover how you can re-write your Git history using Rebase to restructure your commits to ensure they're understandable before you push your changes.

Recommendation

You should only rebase commits that have not been shared with other people via push. Rebasing commits causes their commit-ids to change which can result in losing future commits.

Amending Commit Messages

Re-writing the repositories history is done using `git rebase -interactive`. By putting rebase into interactive mode you have more control over the changes you want to make. After launching into interactive mode you are given six commands to perform on each commit in the repository. By using the editor which opens, by default Vim, you define which actions you want to perform on each commit.

In this example we want to change the commit. To put it into this state we need to change the word "pick" next to the commit to match the action you want to perform based on the list shown in the Vim window, in this case "reword".

In this example we want to change the commit message.

Start

To begin we need to enter Interactive Rebase mode using `git rebase --interactive --root`

Select Interactive Mode

To begin with Vim can be a little confusing, to edit text you need to first type `i` which will put you into "insert mode".

Thank you for your Participation

kindly reach p-v.sasirekha@capgemini.com for queries.