Git Lab Guide

Lab 1

Duration:
20 min

Objectives:
Practice in creating Git repository and performing first commits. Understand basics of branching in Git.

Description:
Create Git repository, customize Git. Perform first commits. Work with branches: create, switch, edit files, commit changes. Perform first merge.

Notations:
(*) - commit symbol.

Task:

1. **Creating repository. Performing first commits**

   - Create Alpha Git repository.
   - Set user name and email in local Git configurations.
   - Create index.html file with the following content:

```
<html>
<head>
    <title>Hello gt</title>
</head>
<body>
    First git repo
</body>
</html>
```

   - Add file to index, commit changes (m1).
   - Create styles.css file with the following content:

```
body {
    background-color: #ccc;
    font-size: 15px;
}
```

   - Add file to index, commit changes (m2). Check log.

2. **Working with branches**

   - Create js-feature branch.
   - Create title-fix branch and switch to it. Edit index.html:

```
- <title>Hello gt</title>
+ <title>Hello git</title>
```

- Add and commit bugfix (tf1).
- Switch back to master. Add the following code to index.html:

```
<body>
    First git repo
    + Prepare to first merge
</body>
```

- Add and commit changes (m3). Merge title-fix into master branch (m4).

Lab 2

Duration:
20 min

Objectives:
Learn how to customize Git history, practice in creating aliases.

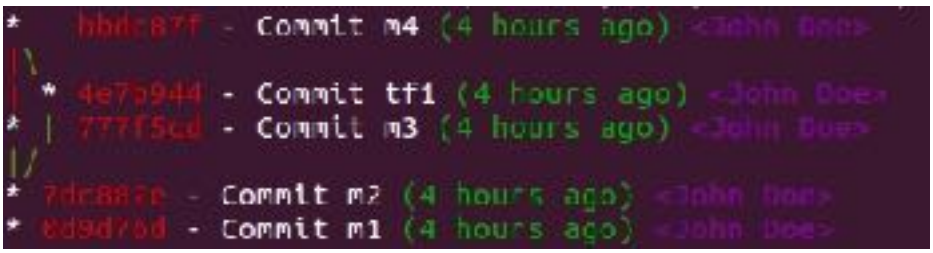
Description:
Configure git log, create aliases.

Notations:
(*) - commit symbol.

Task:

1. **Working with Git history**

   - Work with git log.
   - Configure git log, so its output would look like as follows:



2. **Creating aliases**

   - Configure aliases for the following commands:
     - Format log output (from the task above)
     - git checkout -b
     - git commit -am

Lab 3

Duration:

1 hour 20 min

Objectives:

Learn how to rewrite Git history. Practice in parallel work with remote repository. Get acquainted with Git specific commands: cherry-pick, rebase, stash, tag.

Description:

Rewrite Git history by squashing commits together. Use git cherry-pick to take a commit from somewhere else, and "play it back" wherever you are right now. Rebase commits from one branch into another one. Create remote repository, practice in pushing branches to it and cloning it into local repo. Work with tags. Remove remote branches. Stash changes to be able to switch branches, without commiting what you've been working on.

Notations:

(*) - commit symbol.

Task:

1. **Rewriting Git history**

   - Reset master branch to (m3) commit.
   - Edit index.html as follows:

```
<body>
    First git repo
    - Prepare to first merge
    + Prepare to first rebase
</body>
```

   - Add and commit changes (m5).
   - For the current master branch, squash two last commits (m3) and (m5) into one commit (m6).

2. **Cherry-picking commit**

   - Apply commit from title-fix branch into master branch.

3. **Rebasing**

   - Switch to js-feature branch.
   - Add the following code to index.html:

```
<head>
    <title>Hello gt</title>
    + <script>console.log("hello git!")</script>
</head>
```

- Add and commit changes (jf1).
- Rebase commits from master branch into js-feature branch.
- Create js-feature-old branch, set branch pointer to (jf1) commit.

4. **Working with remote repository**

- Create remote repo - remote at github/bitbucket or in file system.
- Push all local branches from Alpha to created repo.
- Clone remote repo into another one local repo (Sigma).
- In Alpha, rollback 3 last commits in js-feature branch and push it to remote repo.

5. **Tagging**

- In Alpha, add tag to last but one commit in master branch.
- In Sigma, switch to that tag.

6. **Removing remote branch**

- Remove js-feature-old branch from the remote repo using console.

7. **Stashing your work**

- In Alpha, add the following code to title-fix:

```
- <title>Hello git<title>
+ <title>Hello git, goodbye SVN<title>
```

- Making no commit, switch to master branch.
- Remove style.css file, commit changes.
- Switch back to title-fix branch and commit previously made changes.

Lab Guide Solution

Lab 1

1. Creating repository. Performing first commits

- Create Alpha Git repository.

```
$ mkdir Gitlab
$ cd Gitlab
$ mkdir Alpha
$ cd Alpha
$ git init
```

- Set user name and email in local Git configurations.

```
$ git config user.name "John Doe"
$ git config user.email "john@gmail.com"
```

- Create index.html file with the following content:

```
<html>
<head>
    <title>Hello gt</title>
</head>
<body>
    First git repo
</body>
</html>
```

```
$ nano index.html
// fill created file with given data
```

- Add file to index, commit changes (m1).

```
$ git add index.html
$ git commit -m "Commit m1"
```

- Create styles.css file with the following content:

```
body {
    background-color: #ccc;
    font-size: 15px;
}
```

```
$ nano styles.css
// fill created file with given data
```

- Add file to index, commit changes (m2). Check log.

```
$ git add styles.css
```

```
$ git commit -m "Commit m2"
$ git log
```

3. Working with branches

- Create js-feature branch.

```
$ git branch js-feature
```

- Create title-fix branch and switch to it. Edit index.html:

```
- <title>Hello gt<title>
+ <title>Hello git<title>
```

```
$ git checkout -b title-fix
$ nano index.html
// edit file
```

- Add and commit bugfix (tf1).

```
$ git add index.html
$ git commit -m "Commit tf1"
```

- Switch back to master. Add the following code to index.html:

```
<body>
    First git repo
    + Prepare to first merge
</body>
```

```
$ git checkout master
$ nano index.html
// edit file
```

- Add and commit changes (m3). Merge title-fix into master branch (m4).

```
$ git add index.html
$ git commit -m "Commit m3"
$ git merge title-fix
```

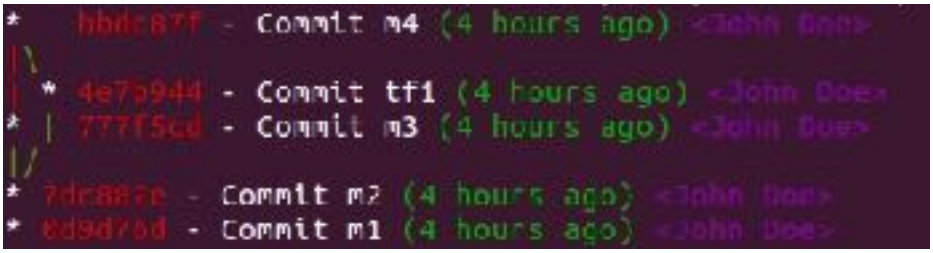
Lab 2

1. Working with Git history

- Work with git log.

```
$ git log
$ git log --graph
// try different flags
```

- Configure git log, so its output would look like as follows:




```
$ git log --pretty=format:'%Cred%h%Creset - %s %C(#00aa00)(%cr)
%Creset %C(#8800aa)<%an>%Creset' --graph
```

3. **Creating aliases**

- Configure aliases for the following commands:
  - Format log output (from the task above)

```
$ git config --local alias.mylog "log --
pretty=format:'%Cred%h%Creset - %s %C(#00aa00)(%cr)%Creset
%C(#8800aa)<%an>%Creset' --graph”
```

  - git checkout -b

```
$ git config --local alias.cob "checkout -b"
```

  - git commit -am

```
$ git config --local alias.cob "commit -am"
```

Lab 3

Task:

1. **Rewriting Git history**

- Reset master branch to (m3) commit.

```
$ git reset --hard 777f5cd4
```

- Edit index.html as follows:

```
<body>
    First git repo
    - Prepare to first merge
    + Prepare to first rebase
</body>
```

```
$ nano index.html
// edit file
```

- Add and commit changes (m5).

```
$ git add index.html
$ git commit -m "Commit m5"
```

- For the current master branch, squash two last commits (m3) and (m5) into one commit (m6).

```
$ git rebase -i HEAD~2
// use squash command to commit (m5)
// fix commit message
```

8. Cherry-picking commit

- Apply commit from title-fix branch into master branch.

```
$ git cherry-pick 4e7b944
```

9. Rebasing

- Switch to js-feature branch.

```
$ git checkout js-feature
```

- Add the following code to index.html:

```
<head>
    <title>Hello gt<title>
    + <script>console.log("hello git!")</script>
</head>
```

```
$ nano index.html
// edit file
```

- Add and commit changes (jf1).

```
$ git add index.html
$ git commit -m "Commit jf1"
```

- Rebase commits from master branch into js-feature branch.

```
$ git rebase master
// while rebasing, problem is found
$ nano index.html
// fix conflicts
$ git add index.html
$ git rebase --continue
```

- Create js-feature-old branch, set branch pointer to (jf1) commit.

```
$ git checkout -b js-feature-old
$ git reflog
// find commit (jf1)
$ git reset --hard 90a7795
```

10. Working with remote repository

- Create remote repo - remote at github/bitbucket or in file system.

```
// we are in Gitlab/Alpha right now
$ cd ..
$ mkdir Remote
$ cd Remote
$ git init --bare
```

- Push all local branches from Alpha to created repo.

```
$ cd ../Alpha
$ git remote add origin ../Remote
$ git checkout master
$ git push -u origin master
$ git checkout title-fix
$ git push -u origin title-fix
$ git checkout js-feature
$ git push -u origin js-feature
$ git checkout js-feature-old
$ git push -u origin js-feature-old
```

- Clone remote repo into another one local repo (Sigma).

```
$ cd ..
$ git clone ~/Documents/Gitlab/Remote Sigma
```

- In Alpha, rollback 3 last commits in js-feature branch and push it to remote repo.

```
$ cd Alpha
$ git checkout js-feature
$ git reset --hard HEAD~3
$ git push origin js-feature --force
```

11. Tagging

- In Alpha, add tag to last but one commit in master branch.

```
$ git checkout master
$ git checkout HEAD~1
$ git tag v.0.1
```

- In Sigma, switch to that tag.

```
$ git push --tags
$ cd ../Sigma/
$ git fetch --tags
$ git checkout tags/v.0.1
```

12. Removing remote branch

- Remove js-feature-old branch from the remote repo using console.

```
$ git push origin --delete js-feature-old
```

13. Stashing your work

- In Alpha, add the following code to title-fix:

```
- <title>Hello git<title>
+ <title>Hello git, goodbye SVN<title>
```

```
$ cd ../Alpha
$ git checkout title-fix
$ nano index.html
// edit file
```

- Making no commit, switch to master branch.

```
$ git stash
$ git checkout master
```

- Remove style.css file, commit changes.

```
$ git rm styles.css
$ git commit -m "Commit m7"
```

- Switch back to title-fix branch and commit previously made changes.

```
$ git checkout title-fix
$ git stash pop
$ git commit -am "Commit tf2"
```