Git Training Part-2

CONTENTS

Copyright @2020, BJIT Group. All Rights Reserved

GIT COMMIT MESSAGE GUIDELINE

This is a way to tell Git about the staged changes you made to the repository using the commit command. Follow these guidelines when writing good commit messages:

- Capitalized, short (50 chars or less) summary of changes.
- Put a blank line between the summery and description.
- Wrap the description 72 characters or less. Use a hyphen (-), followed by a single space with blank lines in between for bullet points if needed.
- Put a blank line.
- Footer line. The `footer` is usually an external reference, like a Jira/Redmine ID or issue number.
- Write the summary line and description of what you have done in the imperative mood.
- Imperative mood: a verb form that makes a command including orders, request, advice, instructions and warnings.

Copyright @2020, BJIT Group. All Rights Reserved

GIT COMMIT MESSAGE GUIDELINE

- A standard commit should contain the following sections:

  <Sub-system>: <Summary>

    Sub-system: Name of the system, Optional if there is no sub-system in particular.

    Summary: Concise/Core description of the commit.

  <Body>

    Detail of the commit message. If the summary can completely express everything, there is no need for a message body.

  <Footer>

    Fixes #redmine-id , represents the bug

    Refs #redmine-id , represents a continuation of a task. A task may have several commit to complete.

    Closes #redmine-id , represents the last commit of the task. Task should be closed.

- Full Conventional Commit Example.

  Calendar: add calendar.h file to fix the build error

  Without this file, it gives error - calendar.h file not found.

  Fixes: Refs. #1234

- An example of a git command or you can use GUI tool.

  ```
  $ git commit -m <Summary> -m <Description> -m <Footer>
  ```

Copyright @2020, BJIT Group. All Rights Reserved

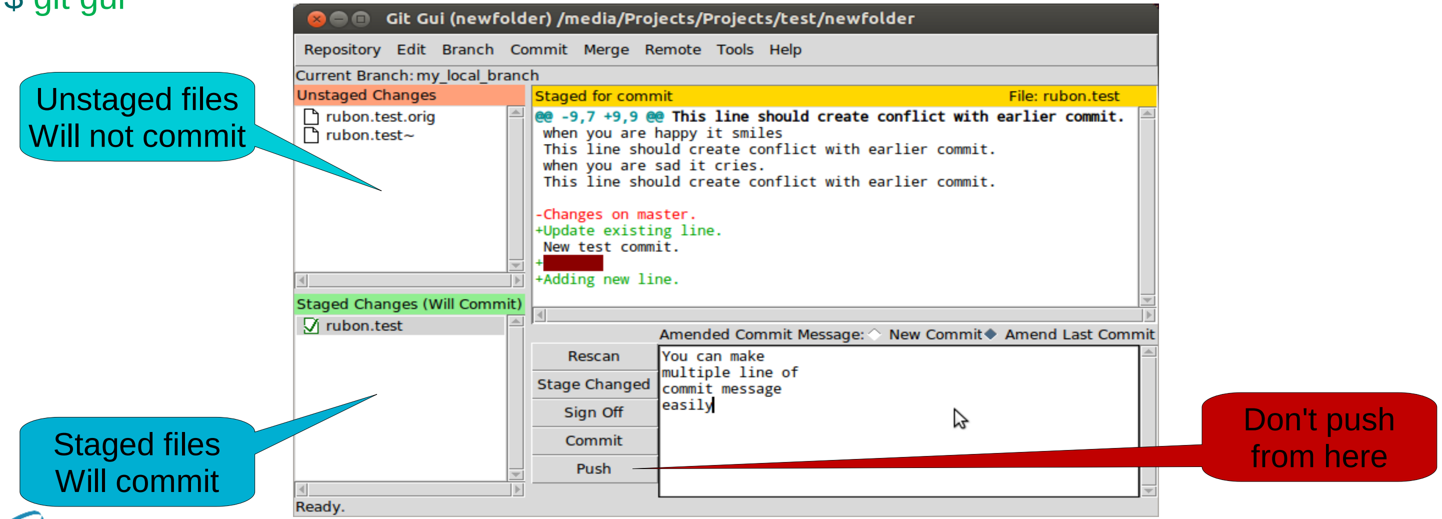
GIT COMMIT USING GUI TOOL

- You may need to update your last commit. You may don't want to make new commit, instead you need to update your last commit. Following command would work.

  ```
  $ git commit -a --amend -m "<Your commit message here>“
  $ git commit -a --amend –no-edit
  ```

- I personally prefer to use gui tool for this purpose, **Git GUI**.
- Using git gui you can easily add/remove files from stage, can see your changes, red marks, write multiple line of commit message and can commit. But you should not push from git gui.

  ```
  $ git gui
  ```


Git Gui (newfolder) /media/Projects/Projects/test/newfolder
Repository Edit Branch Commit Merge Remote Tools Help
Current Branch: my_local_branch
Unstaged Changes
rubon.test.orig
rubon.test~
Staged Changes (Will Commit)
rubon.test
Staged for commit
File: rubon.test
Amended Commit Message: New Commit Amend Last Commit
Rescan
Stage Changed
Sign Off
Commit
Push
Ready.
Unstaged files Will not commit
Staged files Will commit
Don't push from here

```
@@ -9,7 +9,9 @@ This line should create conflict with earlier commit.
 when you are happy it smiles
 This line should create conflict with earlier commit.
 when you are sad it cries.
 This line should create conflict with earlier commit.

-Changes on master.
+Update existing line.
 New test commit.
+
+Adding new line.
```

```
You can make
multiple line of
commit message
easily
```

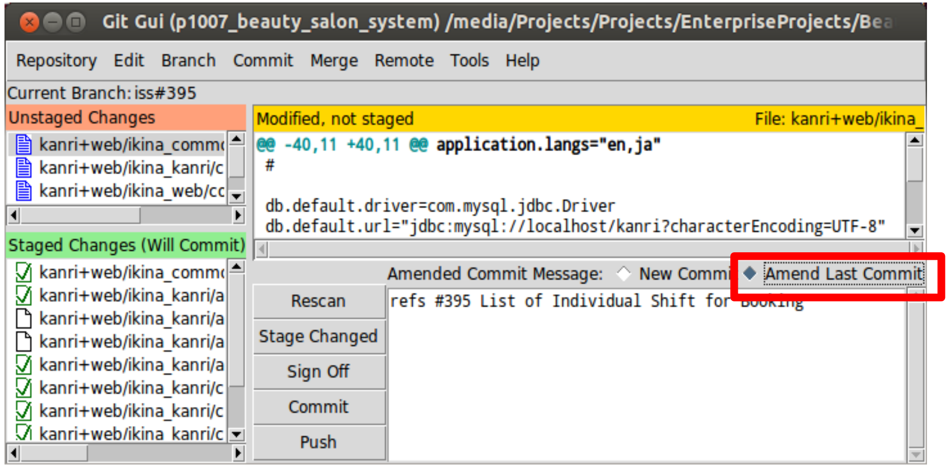
Copyright @2020, BJIT Group. All Rights Reserved

UPDATE YOUR COMMIT – commit amend (1/2)

- You may need to update your commit for various reasons:
  - You may find a problem/bug by yourself. So you need to update your commit.
  - Reviewer might give review feedback that you need to fix. So you need to update your commit.
  - For any reason you checkout to different branch without finishing your current task. So you committed partial task. Now you come back to that work, finish your task. So you need to update your commit.
- You can update your last commit very easily. Here is the command.

```
$ git commit -a --amend -m "Write your comment here“
$ git commit –a --amend –no-edit        // keep the last commit message
```

- It's much more simple using git gui:



```
kanri+web/ikina_comm
kanri+web/ikina_kanri/c
kanri+web/ikina_web/cc
```

```
@@ -40,11 +40,11 @@ application.langs="en,ja"
 #

 db.default.driver=com.mysql.jdbc.Driver
 db.default.url="jdbc:mysql://localhost/kanri?characterEncoding=UTF-8"
```

```
kanri+web/ikina_comm
kanri+web/ikina_kanri/a
kanri+web/ikina_kanri/a
kanri+web/ikina_kanri/a
kanri+web/ikina_kanri/a
kanri+web/ikina_kanri/c
kanri+web/ikina_kanri/c
kanri+web/ikina_kanri/c
```

```
refs #395 List of Individual Shift for Booking
```

Copyright @2020, BJIT Group. All Rights Reserved

GIT IMPORTANT COMMAND

- To Undo a Conflict
```
$ git merge --abort
$ git rebase --abort
```
- To copy specific commit to different branch
```
$ gitk // Copy the sha1 value
$ git checkout <branchName>
$ git cherry-pick sha1
```
- To remove last commit/specific commit
```
$ git checkout <branchName>
$ git reset --hard HEAD~1
```
- To recover deleted commit
```
$ git reflog
$ git checkout <sha>/HEAD@{N}
```
- To find Hash ID of local/remote branch
```
$ git rev-parse <branchname>
$ git rev-parse origin/<branchname>
```
- To recover deleted branch
```
$ git reflog
$ git branch <branchname> ID
$ git checkout -b <newBranchName> ID
```

Copyright @2020, BJIT Group. All Rights Reserved

GIT IMPORTANT COMMAND

- To check merged branches
  ```
  $ git branch –merged
  ```

- To check no-merged branches
  ```
  $ git branch --no-merged
  ```

- To search by date --before/--after
  ```
  $ git log --after="2022-11-1"
  $ git log --after="2022-11-1" --before="2022-11-30“
  ```

- To search by message --grep
  ```
  $ git log --grep="resolve"
  ```

- To search by author --author
  ```
  $ git log --author="Ujjal“
  ```

- To search by file -- <filename>
  ```
  $ git log -- README.md
  ```

Copyright @2020, BJIT Group. All Rights Reserved

WORKING GUIDELINE – BEST PRACTICES

- Don’t work on master branch, develop branch, or your main branch. Work only in your local branch.
- Create your local branch for a specific task.
- After finishing your task, commit it and push it to master branch(if you want to push into master).
- For every separate tasks or tickets create separate local branches, work on that branch, commit and push to master.
- If there is any review feedback, switch to that branch, fix and commit additional patch-set.
- Let’s see the steps of a total workflow:

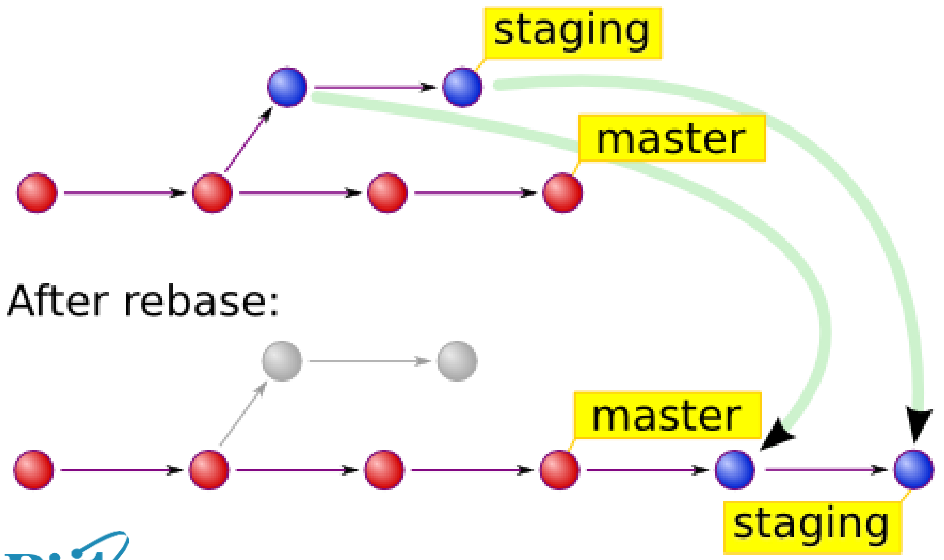
```
$ git clone or git pull              // to take latest update
$ git checkout -b iss#303            // create and checkout to local branch
Finish your task
$ git commit                     // only commit, don't push
```

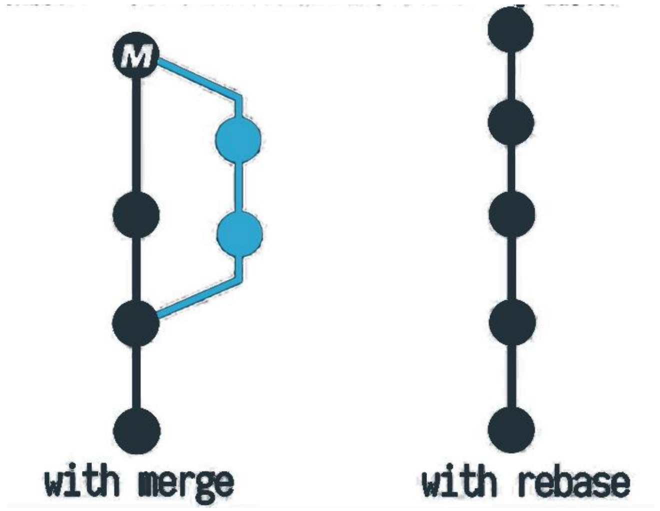
- Always keep your master or main development branch clean. This will help you to take latest updated source by Git pull.
- Let’s think about another scenario. Suppose you are working on a feature in master branch. During your development you receive another high priority bug fixing task. What will you do?
- The simple solution would be if you work on your local branch. Commit your current incomplete task in your local branch. Create/Checkout to the high priority bug task branch, finish it, push it. Now back to your earlier feature task local branch, finish your task, amend a commit and push.

Copyright @2020, BJIT Group. All Rights Reserved

GIT REBASE

Rebasing is the process of moving or combining a sequence of commits to a new base commit.
Git rebase command: $ git rebase <branchname>





Copyright @2020, BJIT Group. All Rights Reserved

GIT Hooks

Git Hooks are build-in feature in Git. Git Hooks are scripts that run before/after certain events or actions. If Git hooks script failed, then it prevents the actions from happening. For example, if commit message does not maintain conventional commit message, it prevents to commit the code.

There are two groups for Git hooks. One is server-side hooks, and another is client-side hooks. Server-side hooks runs on your remote repository. It is triggered before/after certain events/actions happened on your remote repository. Client-side hooks are triggered on your local repository.

Why we use Git hooks?

- Increase productivity
- Save time
- Run custom scripts
- Enforce standards that prevents error

Copyright @2020, BJIT Group. All Rights Reserved

Thank you for watching

Get in touch with us:

www.bjitgroup.com