Scenario 1 - Committing Files

Step 5 of 5

Step 5 - Git Ignore

Sometimes there are particular files or directories you never want to commit, such as local development configuration. To ignore these files you create a *.gitignore* file in the root of the repository.

The *.gitignore* file allows you to define wildcards for the files you wish to ignore, for example *.tmp will ignore all files with the extension .tmp.

Any files matching a defined wildcard will not be displayed in a `git status` output and be ignored when attempting the `git add` command.

Task

Add and commit a *.gitignore* file to the repository to ignore all *.tmp files.

Protip

The `.gitignore` should be committed to the repository to ensure the rules apply across different machines.

SHOW SOLUTION

CONTINUE

Terminal

```
This is your command line, a safe place to practice & complete the scenario

$ git init
Initialized empty Git repository in /home/scrapbook/tutorial/.git/
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        hello-world.js

nothing added to commit but untracked files present (use "git add" to track)
$ git add hello-world.js
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   hello-world.js

$ git commit -m "add Hello World"
[master (root-commit) d49c256] add Hello World
 1 file changed, 1 insertion(+)
 create mode 100644 hello-world.js
$ vim .gitignore
$ git add .gitignore; git commit -m "Add gitignore"
[master 82920a7] Add gitignore
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
$
```

Scenario 2 - Committing Changes

Step 6 - Git Show

While git log tells you the commit author and message, to view the changes made in the commit you need to use the the command `git show`

Like with other commands, by default it will show the changes in the HEAD commit. Use `git show <commit-hash>` to view older changes.

CONTINUE

Terminal

```
ESC[1m+++ b/committed.jsESC[m
ESC[36m@@ -1 +1 @@ESC[m
ESC[31m-console.log("Committed File")ESC[m
ESC[32m+ESC[mESC[32mconsole.log("Demostrating changing a committed file")ESC[m
$ git diff
$ git diff committed.js
$ git log
ESC[33mcommit ae8696df20848bd02b03cecabca0d87c2a357200ESC[mESC[33m (ESC[mESC[1;36mHEAD -> ESC[mESC[1;32mmasterESC[mESC[33m)ESC[m
Author: Katacoda Scenario <scenario@katacoda.com>
Date:   Mon Dec 28 15:03:30 2020 +0000

    Changed the output message in committed.js

ESC[33mcommit 2ddfcee8637fb4751f34c2b22806928feb262bd0ESC[m
Author: Katacoda Scenario <scenario@katacoda.com>
Date:   Mon Dec 28 15:00:40 2020 +0000

    Initial Commit
$ git log --pretty=format:"%h %an %ar - %s"
ae8696d Katacoda Scenario 15 seconds ago - Changed the output message in committed.js
2ddfcee Katacoda Scenario 3 minutes ago - Initial Commit
$ git show
ESC[33mcommit ae8696df20848bd02b03cecabca0d87c2a357200ESC[mESC[33m (ESC[mESC[1;36mHEAD -> ESC[mESC[1;32mmasterESC[mESC[33m)ESC[m
Author: Katacoda Scenario <scenario@katacoda.com>
Date:   Mon Dec 28 15:03:30 2020 +0000

    Changed the output message in committed.js

ESC[1mdiff --git a/committed.js b/committed.jsESC[m
ESC[1mindex 12e7e7c..fc77969 100644ESC[m
ESC[1m--- a/committed.jsESC[m
ESC[1m+++ b/committed.jsESC[m
ESC[36m@@ -1 +1 @@ESC[m
ESC[31m-console.log("Committed File")ESC[m
ESC[32m+ESC[mESC[32mconsole.log("Demostrating changing a committed file")ESC[m
$
```

Scenario 3 - Working Remotely

Step 5 - Git Fetch

The command `git pull` is a combination of two different commands, `git fetch` and `git merge`. Fetch downloads the changes from the remote repository into a separate branch named *remotes/<remote-name>/<remote-branch-name>*. The branch can be accessed using `git checkout`.

Using `git fetch` is a great way to review the changes without affecting your current branch. The naming format of branches is flexible enough that you can have multiple remotes and branches with the same name and easily switch between them.

The following command will merge the fetched changes into master.

```
git merge remotes/<remote-name>/<remote-branch-name> master
```

We'll cover merging in more detail in a future scenario.

Task

Additional changes have been made in the *origin* repository. Use `git fetch` to download the changes and then checkout the branch to view them.

Protip

You can view a list of all the remote branches using the command `git branch -r`

SHOW SOLUTION

CONTINUE

Terminal

```
From /s/remote-project/1
 * branch            master     -> FETCH_HEAD
   d55a3b0..b0a7296  master     -> origin/master
Updating d55a3b0..b0a7296
Fast-forward
 new-file.txt | 1 +
 staging.txt  | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 new-file.txt
$ git log --grep="#1234"
ESC[33mcommit b0a7296d57b95f506a4ca3b5522724569c4dfcdaESC[mESC[33m (ESC[mESC[1;36mHEAD -> ESC[mESC[1;32mmasterESC[mESC[33m, ESC[mESC[1;31morigin/masterESC[
Author: Different User <DifferentUser@JoinScrapbook.com>
Date:   Mon Dec 28 15:11:02 2020 +0000

    Fix for Bug #1234
$ git fetch
remote: Counting objects: 2, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
From /s/remote-project/1
   b0a7296..9e394e3  master     -> origin/master
$ git checkout remotes/origin/master
Note: checking out 'remotes/origin/master'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 9e394e3 Fix for Bug #42
$
```

Scenario 4 - Undoing Changes

Step 5 - Git Revert

To revert multiple commits at once we use the character ~ to mean minus. For example, HEAD~2 is two commits from the head. This can be combined with the characters ... to say between two commits.

Task

Use the command `git revert HEAD...HEAD~2` ✓ to revert the commits between HEAD and HEAD~2.

Protip

You can use the command `git log --oneline` ↵ for a quick overview of the commit history.

SHOW SOLUTION

CONTINUE

Terminal

```
$ get reset --hard HEAD^
bash: get: command not found
$ git reset --hard HEAD^
HEAD is now at a19d671 Fixing Error
$ git status
On branch master
nothing to commit, working tree clean
$ git log
ESC[33mcommit fdbb96c3afdbee3166dca44cc9c9284a6fcba2f1ESC[mESC[33m (ESC[mESC[1;36mHEAD -> ESC[mESC[1;32mmasterESC[mESC[33m)ESC[m
Author: Katacoda Scenario <scenario@katacoda.com>
Date:   Mon Dec 28 15:15:44 2020 +0000

    Commit To Revert

ESC[33mcommit a19d6719622d597c8736f178b0eb3c14a5252e62ESC[m
Author: Katacoda Scenario <scenario@katacoda.com>
Date:   Mon Dec 28 15:12:33 2020 +0000

    Fixing Error

ESC[33mcommit cfbbd8f89c581a04584d3f0278f7bd6a8df9c343ESC[m
Author: Katacoda Scenario <scenario@katacoda.com>
Date:   Mon Dec 28 15:12:33 2020 +0000

    First Commit
$ git revert fdbb9c
fatal: bad revision 'fdbb9c'
$ git revert fdbb96c3afdbee3166dca44cc9c9284a6fcba2f1
[master 7af7556] Revert "Commit To Revert"
 1 file changed, 1 insertion(+), 1 deletion(-)
$ git revert HEAD...HEAD~2
[master 57ecc01] Revert "Revert "Commit To Revert""
 1 file changed, 1 insertion(+), 1 deletion(-)
[master d346701] Revert "Commit To Revert"
 1 file changed, 1 insertion(+), 1 deletion(-)
$
```

Scenario 5 - Fixing Merge Conflicts

Step 6 - Rebasing Pull Requests

This approach also applies when working with remote branches and can be applied when issuing a pull request using:

```
git pull --rebase
```

This will act as if you had done a pull request before each of your commits.

CONTINUE

Terminal

```
$ git add staging.txt
$ git commit --no-edit
On branch master
nothing to commit, working tree clean
$ git pull --no-edit origin master
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
From /s/remote-project/1
 * branch            master     -> FETCH_HEAD
   a361e1c..c71fbb3  master     -> origin/master
fatal: refusing to merge unrelated histories
$ git log --all --decorate --oneline
ESC[33mb9c1145ESC[mESC[33m (ESC[mESC[1;36mHEAD -> ESC[mESC[1;32mmasterESC[mESC[33m)ESC[m Fix for Bug #55
ESC[33mc71fbb3ESC[mESC[33m (ESC[mESC[1;31morigin/masterESC[mESC[33m)ESC[m Fix for Bug #58a
ESC[33m44d9750ESC[m Fix for Bug #58
ESC[33m0098d63ESC[m Fixing Error
ESC[33ma361e1cESC[m Fix for Bug #1234
ESC[33m85ae4e2ESC[m First Commit
$ git rebase feature/10
Current branch feature/10 is up to date.
$
$ git pull --rebase
There is no tracking information for the current branch.
Please specify which branch you want to rebase against.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> master

$
$
```

Scenario 6 - Experiments Using Branches

Step 5 - Clean Up Branches

Cleaning up branches is important to remove the amount of noise and confusion. To delete a branch you need to provide the argument `-d`, for example `git branch -d <branch_name>`

Task

Now the branch has been merged into master it is no longer required. Delete your new branch to keep your repository clean and understandable.

CONTINUE

Terminal

```
This is your command line, a safe place to practice & complete the scenario

$ git branch new_branch master
$ git cheackout new_branch
git: 'cheackout' is not a git command. See 'git --help'.

The most similar command is
       checkout
$ git checkout new_branch
Switched to branch 'new_branch'
$ git branch -va
  master    ESC[m 0f52df4 First Commit on master
* ESC[32mnew_branchESC[m 9af4838 Commit on branch
$ git checkout master
Switched to branch 'master'
$ git merge new_branch
Updating 0f52df4..69db10f
Fast-forward
 new-file-6.txt | 1 +
 staging.txt    | 2 +
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 new-file-6.txt
$ git push origin new_branch
Counting objects: 9, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (9/9), 749 bytes | 374.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0)
To /s/remote-project/1
 * [new branch]      new_branch -> new_branch
$
$ git branch -d new_branch
Deleted branch new_branch (was 69db10f).
$
```

Scenario 7 - Finding Bugs

Git Blame

While having a "blame" culture isn't desirable, it can be useful to know who worked on certain sections of the file to help with improvements in future. This is where `git blame` can help.

`git blame <file>` shows the revision and author who last modified each line of a file.

Example

Running blame on a file will output who last touched each line.

```
git blame list.html ✓
```

If we know the lines which we're concerned with then we can use the *-L* parameter to provide a range of lines to output.

```
git blame -L 6,8 list.html ✓
```

CONTINUE

Terminal

```
$ cat list.html
<ul>
<li>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</li>
<li>Aliquam tincidunt mauris eu risus.</li>
<li>Vestibulum auctor dapibus neque.</li>
<li>Morbi in sem quis dui placerat ornare. Pellentesque odio nisi, euismod in, pharetra a.</li>
<li>Praesent dapibus, neque id cursus faucibus, tortor neque egestas augue, eu vulputate magna eros eu erat.</li>
</ul>
$ git bisect good
Bisecting: 0 revisions left to test after this (roughly 1 step)
[5bcf4d77dc5aee4408c01c5ad6a18b89b0e5ab36] Adding Quisque
$ cat list.html
<ul>
<li>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</li>
<li>Aliquam tincidunt mauris eu risus.</li>
<li>Vestibulum auctor dapibus neque.</li>
<li>Morbi in sem quis dui placerat ornare. Pellentesque odio nisi, euismod in, pharetra a.</li>
<li>Praesent dapibus, neque id cursus faucibus, tortor neque egestas augue, eu vulputate magna eros eu erat.</li>
Vestibulum auctor dapibus neque
<li>Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed.</li>
</ul>
$ git bisect bad
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[7efbaa63b72d39791d12e7e2b050aaadeefe7e81] Adding Vestibulum
$ git blame list.html
^7ba2231 (Katacoda Scenario 2020-12-28 15:31:53 +0000  1) <ul>
^7ba2231 (Katacoda Scenario 2020-12-28 15:31:53 +0000  2) <li>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</li>
^7ba2231 (Katacoda Scenario 2020-12-28 15:31:53 +0000  3) <li>Aliquam tincidunt mauris eu risus.</li>
^7ba2231 (Katacoda Scenario 2020-12-28 15:31:53 +0000  4) <li>Vestibulum auctor dapibus neque.</li>
eab3e28d (Katacoda Scenario 2020-12-28 15:31:53 +0000  5) <li>Morbi in sem quis dui placerat ornare. Pellentesque odio nisi, euismod in, pharetra a.</li>
0d065c4c (Katacoda Scenario 2020-12-28 15:31:53 +0000  6) <li>Praesent dapibus, neque id cursus faucibus, tortor neque egestas augue, eu vulputate magna er
7efbaa63 (Katacoda Scenario 2020-12-28 15:33:33 +0000  7) Vestibulum auctor dapibus neque
5bcf4d77 (Katacoda Scenario 2020-12-28 15:33:33 +0000  8) <li>Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed.</
888066a6 (Katacoda Scenario 2020-12-28 15:33:33 +0000  9) <li>Pellentesque fermentum dolor</li>
^7ba2231 (Katacoda Scenario 2020-12-28 15:31:53 +0000 10) </ul>
...skipping...
```

Scenario 8 - Being Picky With Git

Step 3 - Continuing Cherry Picking After Conflict

Once the conflicts have been resolved you can continue with the cherry pick using the command `git cherry-pick --continue`.

Similar to using *merge*, resolving a cherry-pick will result in a commit.

Task

Complete the cherry pick by first adding the previously conflicted item and then using the *--continue* option.

`git add list2.html` ✓

`git cherry-pick --continue` ✓

At this point the default editor, in this case vim, will pop up allowing you to edit the cherry-picked commit message to include details of the conflict and how it was resolved. To save and quit vim type *:wq*

CONTINUE

Terminal

```
fatal: cherry-pick failed
$ git status
On branch master
You are currently cherry-picking commit 8bc5954.
  (fix conflicts and run "git cherry-pick --continue")
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both added:      list2.html

no changes added to commit (use "git add" and/or "git commit -a")
$ git diff
ESC[1mdiff --cc list2.htmlESC[m
ESC[1mindex 4d4407a,ae60808..0000000ESC[m
ESC[1m--- a/list2.htmlESC[m
ESC[1m+++ b/list2.htmlESC[m
ESC[36m@@@ -1,3 -1,6 +1,10 @@@ESC[m
  <ul>ESC[m
ESC[32m++<<<<<<< HEADESC[m
ESC[32m +<li>Empty</li>ESC[m
ESC[32m++=======ESC[m
ESC[32m+ <li>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</li>ESC[m
ESC[32m+ <li>Aliquam tincidunt mauris eu risus.</li>ESC[m
ESC[32m+ <li>Vestibulum auctor dapibus neque.</li>ESC[m
ESC[32m+ <li>dapibus neque.</li>ESC[m
ESC[32m++>>>>>>> 8bc5954... Creating Second ListESC[m
  </ul>ESC[m
$ git checkout --theirs list2.html
$ git add list2.html
$ git cherry-pick --continue
[master f3ca659] Creating Second List
 Date: Mon Dec 28 16:03:22 2020 +0000
 1 file changed, 4 insertions(+), 1 deletion(-)
$
```

Scenario 9 - Re-writing History

changes and the end result will be applied to the repository.

Splitting Commits

After defining we want to *edit* the commit we are now in a state that allows us to change the history.

1. As we want to split an existing commit we first we need to remove it using `git reset HEAD^` ✓.
2. The commit has been removed but the files still exist. We can now perform the commits as we previously desire, as two separate actions.

Execute the commands:

```
git add file3.txt ✓
git commit -m "File 3" ✓
git add file4.txt ✓
git commit -m "File 4" ✓
```

Saving

Once happy with the state of the repository, we tell Git to continue the rebase and update the repository with *--continue*.

```
git rebase --continue ✓
```

You can see the output and the two new commits using `git log --oneline` ✓

CONTINUE

Terminal

```
Untracked files:
        file2.txt

nothing added to commit but untracked files present
$ git add file4.txt
fatal: pathspec 'file4.txt' did not match any files
$ git commit -m "File 4"
interactive rebase in progress; onto 248691f
Last command done (1 command done):
   edit 248691f Adding File 4
Next command to do (1 remaining command):
   pick 4ba63b3 Adding File 3
You are currently editing a commit while rebasing branch 'master' on '248691f'.

Untracked files:
        file2.txt

nothing added to commit but untracked files present
$ git rebase --continue
Successfully rebased and updated refs/heads/master.
$ git log --oneline
ESC[33m340345bESC[mESC[33m (ESC[mESC[1;36mHEAD -> ESC[mESC[1;32mmasterESC[mESC[33m)ESC[m Adding File 3 and File 4
ESC[33meb50a8aESC[m Adding File 1
ESC[33m90a26d0ESC[m TODO
ESC[33m7075bc1ESC[m Final Item
ESC[33m184ea76ESC[m New Item
ESC[33m968d317ESC[m Initial comit of the list
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file2.txt

nothing added to commit but untracked files present (use "git add" to track)
$
```

Git SCM - Lab 101

- Wiki
- Code Snippets

Sync repository remotely

Now that you've created an empty project on GitLab, you can now track the local repository to a remote one which can be used to collaborate work within a team.

In order to sync your repository to the remote one, you'll need to add the remote repository;

```
git remote add origin http://git.itworx.cloud/<username>
/simple-html-project.git
```

And use the `git push` command to send your changes remotely;

```
git push -u origin master ✓
```

Outcome

- You have created a new project which can be used to sync your local git repository to it.
- You have syncronized your local repository to the remote GitLab repository.

CONTINUE

README.md | style.css

```
./
  README.md
  file.tmp
  index.html
  style.css
```

```
1 html, body {
2     margin:0; padding:0; background-color:#999;
3 }
4
```

Terminal

```
$ git push -u origin master
Username for 'https://github.com': rrhawk
Password for 'https://rrhawk@github.com':
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/rrhawk/simple-html-project.git/'
$ git push -u origin master
Username for 'https://github.com': rrhawk
Password for 'https://rrhawk@github.com':
Counting objects: 13, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (13/13), 1.38 KiB | 0 bytes/s, done.
Total 13 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/rrhawk/simple-html-project.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
$
```

Read the guide

rrhawk / simple-html-project

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Custom notification settings

Looking for a little more control? Now you can choose which types of activity you'd like to be notified about per repository.

Got it

master 1 branch 0 tags Go to file Add file Code

My Name Add stylesheet style.css `60619d6` 9 minutes ago 4 commits

.gitignore	Adding .gitignore	17 minutes ago
README.md	Modify README.md	10 minutes ago
index.html	Initial commit	18 minutes ago
style.css	Add stylesheet style.css	9 minutes ago

README.md

Simple HTML Project

This is a simple HTML project tracked by git.

Adding some new content to inspect how git diff works. 123123123123

About

My simple HTML Project

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

HTML 76.3% CSS 23.7%

Git SCM - Lab 102

Pull the updates

To get latest refs and content from your remote projects, use;

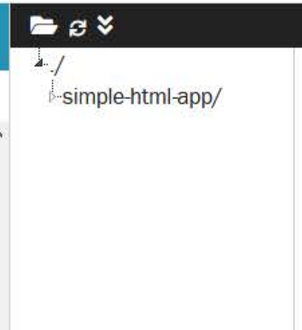
`git pull origin ✓`

This will fetch the remote changes and merge them with the local branch and update the content of the working tree. The output will look like;

```
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From http://git.itworx.cloud/aossama/simple-html-app
   1459192..90f500b  master     -> origin/master
Updating 1459192..90f500b
Fast-forward
 CHANGELOG | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 CHANGELOG
```

Now inspect the local working directory using `ls ✓`, and you'll notice the newly created **CHANELOG** file has been applied to the tree.

CONTINUE



Terminal

```
$ ls
Dockerfile  Jenkinsfile  License  README.md  about.html  index.html
$ git pull origin
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/rrhawk/simple-html-app
   0bd9ba2..cc487d5  master     -> origin/master
Updating 0bd9ba2..cc487d5
Fast-forward
 ChangeLOG | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 ChangeLOG
$ ls
ChangeLOG  Dockerfile  Jenkinsfile  License  README.md  about.html  index.html
$
```

Git SCM - Lab 201

After you've merged the branch, it's now safe to do some house keeping and delete the merged branch, use;

```
git branch -d my-feature ✓
```

Resolving conflict

If the two branches you're trying to merge both changed the same part of the same file, Git won't be able to figure out which version to use. When such a situation occurs, it stops right before the merge commit so that you can resolve the conflicts manually.

How conflicts are presented When Git encounters a conflict during a merge, It will edit the content of the affected files with visual indicators that mark both sides of the conflicted content. These visual markers are: <<<<<<<, =======, and >>>>>>>. Its helpful to search a project for these indicators during a merge to find where conflicts need to be resolved.

```
here is some content not affected by the conflict
<<<<<<< master
this is conflicted text from master
=======
this is conflicted text from feature branch
>>>>>>> feature-717
```

CONTINUE

about.html | delete-me.html

./
- about.html
- simple-html-app/

```
1 Useless content that will be trashed later on!
2
```

Terminal

```
  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.

Since Git 2.0, Git defaults to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master

$ git status
On branch master
nothing to commit, working directory clean
$ git merge my-feature
Already up-to-date.
$ git branch --no-merged
$ git branch -d my-feature
Deleted branch my-feature (was 0988eec).
$
```

Файл Правка Вид Журнал Закладки Инструменты Справка

https://github.com/rrhawk/simple-html-app/tree/my-feature

rrhawk / simple-html-app

Unwatch 1 | Star 0 | Fork 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

Custom notification settings

Looking for a little more control? Now you can choose which types of activity you'd like to be notified about per repository.

Got it

my-feature had recent pushes 1 minute ago — Compare & pull request

my-feature | 2 branches | 0 tags | Go to file | Add file | Code

This branch is 2 commits ahead, 6 commits behind master. — Pull request | Compare

Sergey_Kosolapov and **Sergey_Kosolapov** Add about us page — 0988eec 3 minutes ago — 2 commits

simple-html-app	first commit lab201	3 minutes ago
about.html	Add about us page	3 minutes ago

Help people interested in this repository understand your project by adding a README. — Add a README

About

Yet another simple HTML

Apache-2.0 License

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Contributors 2

aossama Ahmed Ossama

rrhawk Sergey Kosolapov

Languages

HTML 83.2% | Dockerfile 16.8%

ENG 8:23 PM 12/28/2020

https://www.katacoda.com/aossama/scenarios/git-scm-lab-202

O'REILLY Katacoda

KATACODA OVERVIEW & SOLUTIONS

YOUR PROFILE LOG OUT

Git SCM - Lab 202

Step 5 of 5

- Add, commit and push the modified content.

Review, Approve and Merge

1. Review the *Commits* and *Changes* on the Merge request page.
2. Optionally you can specify that the source branch be removed upon it's merge.
3. When satisfied with the merge request, go ahead and click on **Merge** button.

This will cause a new commit on the master branch with the content of the *contact-us* branch.

Pull Latest Content

Now as the master has changed on the remote upstream repository, you'll need to pull the latest changes on your local repository. Use;

`cd ../` ✓

`git checkout master` ✓

`git pull` ✓

To download and update your working directory to the latest changes on the repository.

CONTINUE

app.js

```
../
  another-project-atomic/
  app.js
  package.json
  project-atomic/
```

```
1 const express = require('express')
2 const app = express()
3
4 // Landing page
5 app.get('/', (req, res) => res.send('Hello World!'))
6
7 // About us page
8 app.get('/about', (req, res) => res.send('Yet another about us page!'))
```

Terminal

```
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 364 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/rrhawk/project-atomic.git
   b09bde8..637d83f  main -> main
$ cd ../
$ git checkout master
Already on 'master'
$ git pull
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 21 (delta 4), reused 6 (delta 2), pack-reused 0
Unpacking objects: 100% (21/21), done.
From https://github.com/rrhawk/project-atomic
 * [new branch]      contact-us -> origin/contact-us
 * [new branch]      feature-01 -> origin/feature-01
 * [new branch]      main       -> origin/main
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> master

$
```

ENG 10:01 PM 12/28/2020

https://github.com/rrhawk/project-atomic/pull/1

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

Add feature 1 #1

Edit Open with

Merged **rrhawk** merged 1 commit into `main` from `feature-01` now

Conversation 0 Commits 1 Checks 0 Files changed 2 +11 −0

rrhawk commented 17 seconds ago Owner

No description provided.

Add feature 1 b611d4d

rrhawk merged commit **b09bde8** into `main` now Revert

Pull request successfully merged and closed

You're all set—the `feature-01` branch can be safely deleted. Delete branch

Write Preview

Leave a comment

New **Video support! Upload MP4 and MOV file types.** Attach files by dragging & dropping, selecting or pasting them.

Comment

Remember, contributions to this repository should follow its contributing guidelines.

ProTip! Add .patch or .diff to the end of URLs for Git's plaintext views.

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Successfully merging this pull request may close these issues.
None yet

Notifications Customize
Unsubscribe
You're receiving notifications because you're watching this repository.

1 participant