

# Learn You some GIT

Florian Willich

Quality and Usability Lab  
Berlin Institute of Technology

May 26, 2015

# Table of Contents

GIT

Create Repositories

Make Changes

Group Changes

Suppress Tracking

Save Fragments

Synchronize Changes

The simple five

Miscellaneous

This talk is on Github:  
<https://github.com/c-bebop/git>

# Create Repositories

```
$ git init project-name
```

Creates a new local repository with the specified name

# Create Repositories

```
$ git init project-name
```

Creates a new local repository with the specified name

```
$ git clone url
```

Downloads a project and its entire version history but only the master branch!

# Create Repositories

```
$ git init project-name
```

Creates a new local repository with the specified name

```
$ git clone url
```

Downloads a project and its entire version history but only the master branch!

```
$ git fetch remote-branch/local-branch
```

lets you fetch the remote branch and create a local branch

# Make Changes

\$ `git status`

Most important command! Lists all new or modified files to be committed

# Make Changes

\$ `git status`

Most important command! Lists all new or modified files to be committed

\$ `git add file`

Snapshots the file in preparation for versioning



# Make Changes

\$ `git status`

Most important command! Lists all new or modified files to be committed

\$ `git add file`

Snapshots the file in preparation for versioning

\$ `git commit -m "descriptive message"`

Records file snapshots permanently in version history

# Make Changes

\$ `git status`

Most important command! Lists all new or modified files to be committed

\$ `git add file`

Snapshots the file in preparation for versioning

\$ `git commit -m "descriptive message"`

Records file snapshots permanently in version history

\$ `git commit -am "descriptive message"`

Snapshots all tracked files in preparation for versioning & records file snapshots permanently in version history

# Group Changes

`$ git branch`

Lists all local branches in the current repository

# Group Changes

\$ `git branch`

Lists all local branches in the current repository

\$ `git branch branch-name`

Creates a new branch with the specified branch name

# Group Changes

\$ `git branch`

Lists all local branches in the current repository

\$ `git branch branch-name`

Creates a new branch with the specified branch name

\$ `git checkout branch-name`

Switches to the specified branch and updates the working directory

# Group Changes

\$ `git branch`

Lists all local branches in the current repository

\$ `git branch branch-name`

Creates a new branch with the specified branch name

\$ `git checkout branch-name`

Switches to the specified branch and updates the working directory

\$ `git merge branch-name`

Combines the specified branch's history into the current branch

# Group Changes

\$ `git branch`

Lists all local branches in the current repository

\$ `git branch branch-name`

Creates a new branch with the specified branch name

\$ `git checkout branch-name`

Switches to the specified branch and updates the working directory

\$ `git merge branch-name`

Combines the specified branch's history into the current branch

\$ `git branch -d branch-name`

Deletes the specified branch

# Suppress Tracking

By creating a file called `.gitignore` (yes its a hidden file) in the root directory, you can specify all the files you want git to ignore.



# Suppress Tracking

By creating a file called `.gitignore` (yes its a hidden file) in the root directory, you can specify all the files you want git to ignore.

Examples for files you don't want to track:

- ▶ `*.log`
- ▶ `*.config`
- ▶ `my-secret-passwords.secret`
- ▶ Any IDE related files

# Save Fragments

```
$ git stash
```

Temporarily stores all modified tracked files

# Save Fragments

`$ git stash`

Temporarily stores all modified tracked files

`$ git stash pop`

Restores the most recently stashed files

# Synchronize Changes

\$ `git pull`

Downloads bookmark history and incorporates changes

Shortcut for: `git fetch` and `git merge`

# Synchronize Changes

\$ `git pull`

Downloads bookmark history and incorporates changes

Shortcut for: `git fetch` and `git merge`

\$ `git push`

Uploads all local branch commits

# Synchronize Changes

\$ `git pull`

Downloads bookmark history and incorporates changes

Shortcut for: `git fetch` and `git merge`

\$ `git push`

Uploads all local branch commits

\$ `git merge branch`

Merges the specified branch changes into the the branch you're currently in

# The simple five

- ▶ `$ git status`
- ▶ `$ git pull`
- ▶ `$ git add file`
- ▶ `$ git commit -m "descriptive message"`
- ▶ `$ git push`

And please DON'T use `git commit -am "message"`!

```
$ git checkout hash
```

Use this command only to look up the state of the commit.

```
$ git revert hash
```

Use this command to revert to the hash. This implicitly creates a new commit with the state of hash you reverting to and does not change your history!



# Thank You

Now you've learned yourself some GIT!

Thank You!

Questions?