Version control & git

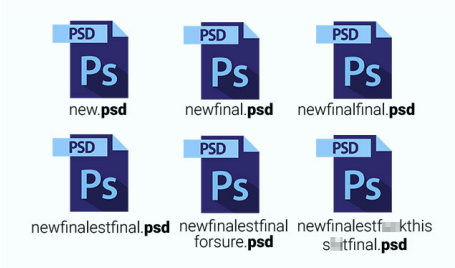
An indroduction

LISCO Lab Meeting. 11.05.2020

Why do we need version control?

Does this sound familiar?

- Shared code updates via email
- Made updates on productive code
- Accidentally overwrote some file, which can never be retrieved again



Let's get away from this . . .

Why do we need version control?

. . . and look forward to:

- Making changes with confidence, and reverting them if needed
- Easily deploying code to production servers
- Understanding who made a change, when and why it happened

The Basics - Tracking Changes

- Track changes that happen within directories or files
- **Repository**: The set of files and directories tracked by version control
- Tell version control which files to track ...
- ... or clone repository from a server

As you make changes, it will track each change behind the scenes, until you are ready to **commit** those changes.

The Basics - Committing and Changesets

- Submit your changes as a collection of actions: a **commit**
- The changeset is given a unique revision ID (hash)

A commit includes:

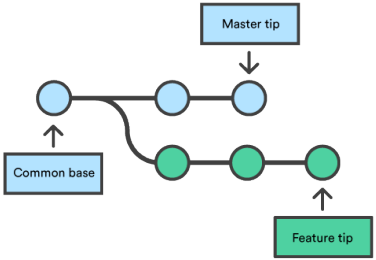
- reference to the person committing
- a timestamp
- affected files and the changes
- a comment from the author

The Basics - Branching

- **Branch**: a copy/snapshot of a repository
- switch between branches and commit without altering the "original repository" (**master**)

Why branching?

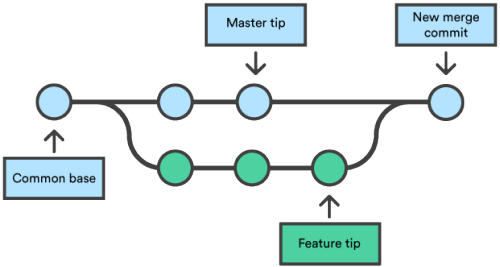
- do changes without worrying of breaking things
- experimental changes
- work on a new feature



The Basics - Merging

Happy with the experimental changes in your branch? Merge it!

- Changes in your branch will be applied to master
- master contains most recent version combined with changes from the branch
- **conflict** may arise when merging → requires manual intervention



The Basics - Pulling and Pushing

Pulling:

- Get latest version when team members **commit** changes
- **Pulling** only downloads changes since last request from server
- These changes are applied to your local copy of the repository
- Might result in a **conflict**

Pushing

- Provide your changes to the team members
- Repository on the server now contains your changes

Get Started

Get Started

Please go to

tinyurl.com/liscogit

and download the template.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow

$ git config --global user.name <name>

Define author name to be used for all commits. Replace name by email to define email address.

$ git init <directory>

Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

$ git status

List which files are staged, unstaged, and untracked.

$ git add <files>

Stage all changes in <files>for the next commit. If <files>are untracked, they are added to the repo.

$ git commit -m "message"

Commit the staged snapshot. The flag -a can be used to add all unstaged files to the commit.

$ git log

Display the entire commit history

$ git branch

List all of the branches in your repo. Add a <branch>argument to create a new branch with the name <branch>.

$ git checkout -b <branch>

Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

$ git merge <branch>

Merge <branch>into the current branch.

Git-Workflow - Remote Repository

$ git pull

Fetch the remote's copy of current branch and immediately merge it into the local copy.

$ git fetch origin <branch>

Fetches a specific <branch>, from the repo. Leave off <branch>to fetch all remote refs.

$ git push

Push the current branch to remote repo, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

Git-Workflow - Remote Repository

$ git pull

Fetch the remote's copy of current branch and immediately merge it into the local copy.

$ git fetch origin <branch>

Fetches a specific <branch>, from the repo. Leave off <branch>to fetch all remote refs.

$ git push

Push the current branch to remote repo, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

Git-Workflow - Remote Repository

$ git pull

Fetch the remote's copy of current branch and immediately merge it into the local copy.

$ git fetch origin <branch>

Fetches a specific <branch>, from the repo. Leave off <branch>to fetch all remote refs.

$ git push

Push the current branch to remote repo, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

Git-Workflow - Remote Repository

$ git pull

Fetch the remote's copy of current branch and immediately merge it into the local copy.

$ git fetch origin <branch>

Fetches a specific <branch>, from the repo. Leave off <branch>to fetch all remote refs.

$ git push

Push the current branch to remote repo, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

Thanks

Author	Commit	Message	Date
Kally Brunner	`0a9bbad`	ups ausversehen familienfotos hinzugefügt	2019-01-28
Kally Brunner	`2a855fe`	BOI ich hab gerade den dümmsten Bug meines Lebens gefangen	2019-01-28