GIT + GITHUB

AGENDA

- What is Git & Github
- How do I use Git & Github
- Lots of work
- Slides and code: https://github.com/ECE-Henrik/hack-your-education-e21

GIT AND GITHUB DESKTOP INSTALL

- Windows (https://git-scm.com/download/win)
- Mac (https://git-scm.com/download/mac)
- Linux (apt): sudo apt-get install git
- Linux (yum): sudo yum install git
- Github for Desktop

WHAT IS VERSION CONTROL

- A system that keeps records of your change history
- Allows a group to develop together
- Creates a history of changes
- Gives the posibility to revert to an ealier state

WHY?

- We make mistakes
- We want to track the reason why something changed
- We want to work together easily

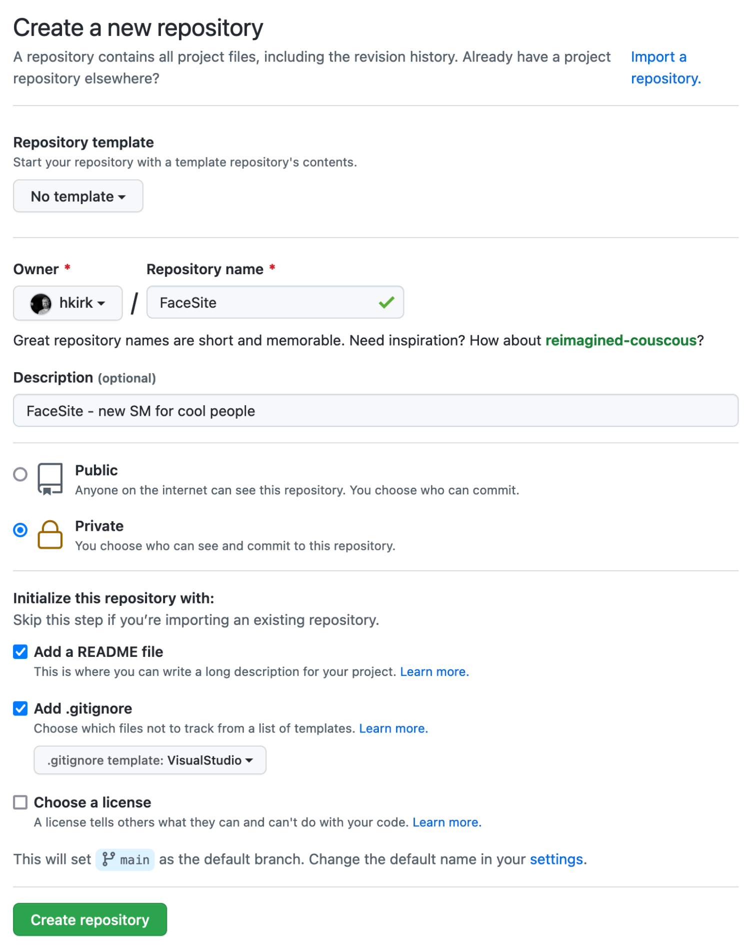
WHAT IS GIT

- Distributed version control
- Each user keep entire history and code on local machine
  - Changes can be made in offline mode
  - Require internet to share - ofcourse :)
- There are other VCS system out there
  - Subversion
  - CVS
  - etc.

WHAT IS GITHUB

- Place to host and share repositories
- Create an account on www.github.com
  - Free with limits
- On top of Git
  - UI, documentation, bug tracking, feature request, pull request
- Github is one platform:
  - Bitbucket, GitLab etc. are alternatives

CREATE PROJECT ON GITHUB



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner * **Repository name** *

hkirk ▾ / FaceSite ✓

Great repository names are short and memorable. Need inspiration? How about **reimagined-couscous**?

Description (optional)

FaceSite - new SM for cool people

○ **Public**
Anyone on the internet can see this repository. You choose who can commit.

● **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☑ **Add a README file**
This is where you can write a long description for your project. Learn more.

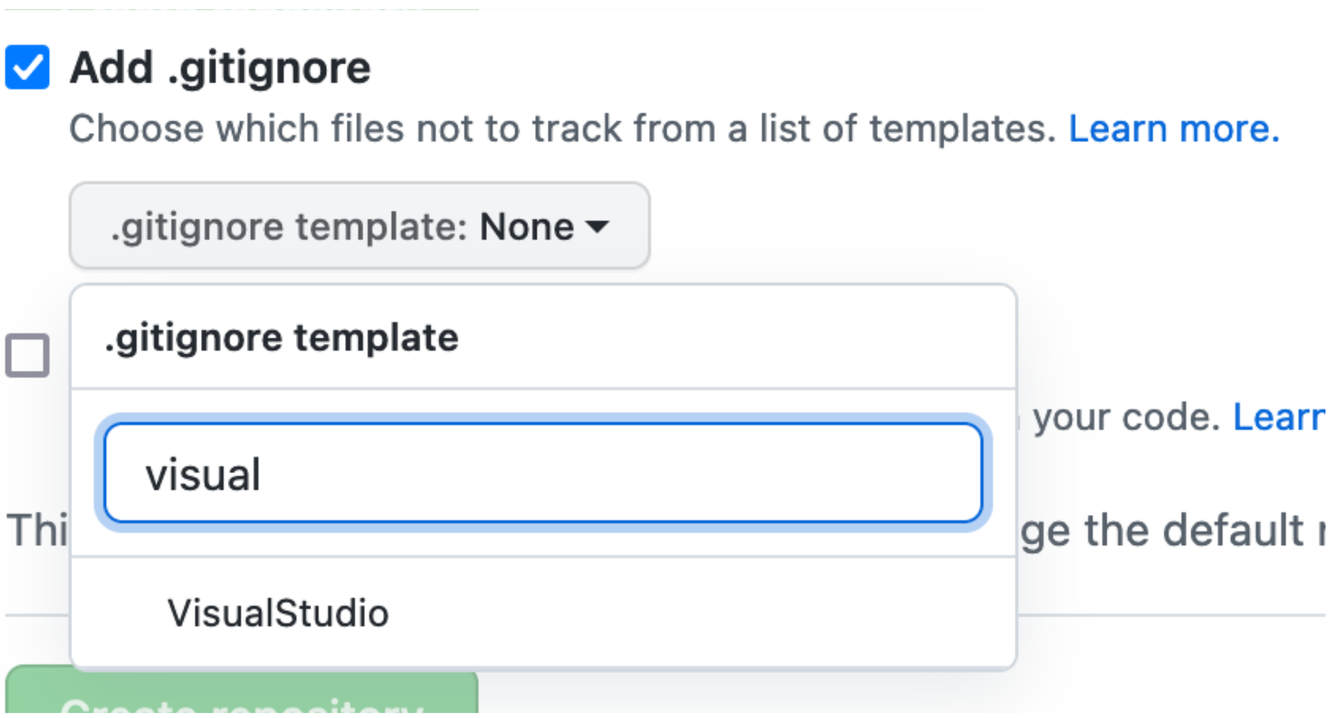
☑ **Add .gitignore**
Choose which files not to track from a list of templates. Learn more.

.gitignore template: VisualStudio ▾

☐ **Choose a license**
A license tells others what they can and can't do with your code. Learn more.

This will set `main` as the default branch. Change the default name in your settings.

IGNORE



☑ Add .gitignore

Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

.gitignore template

visual

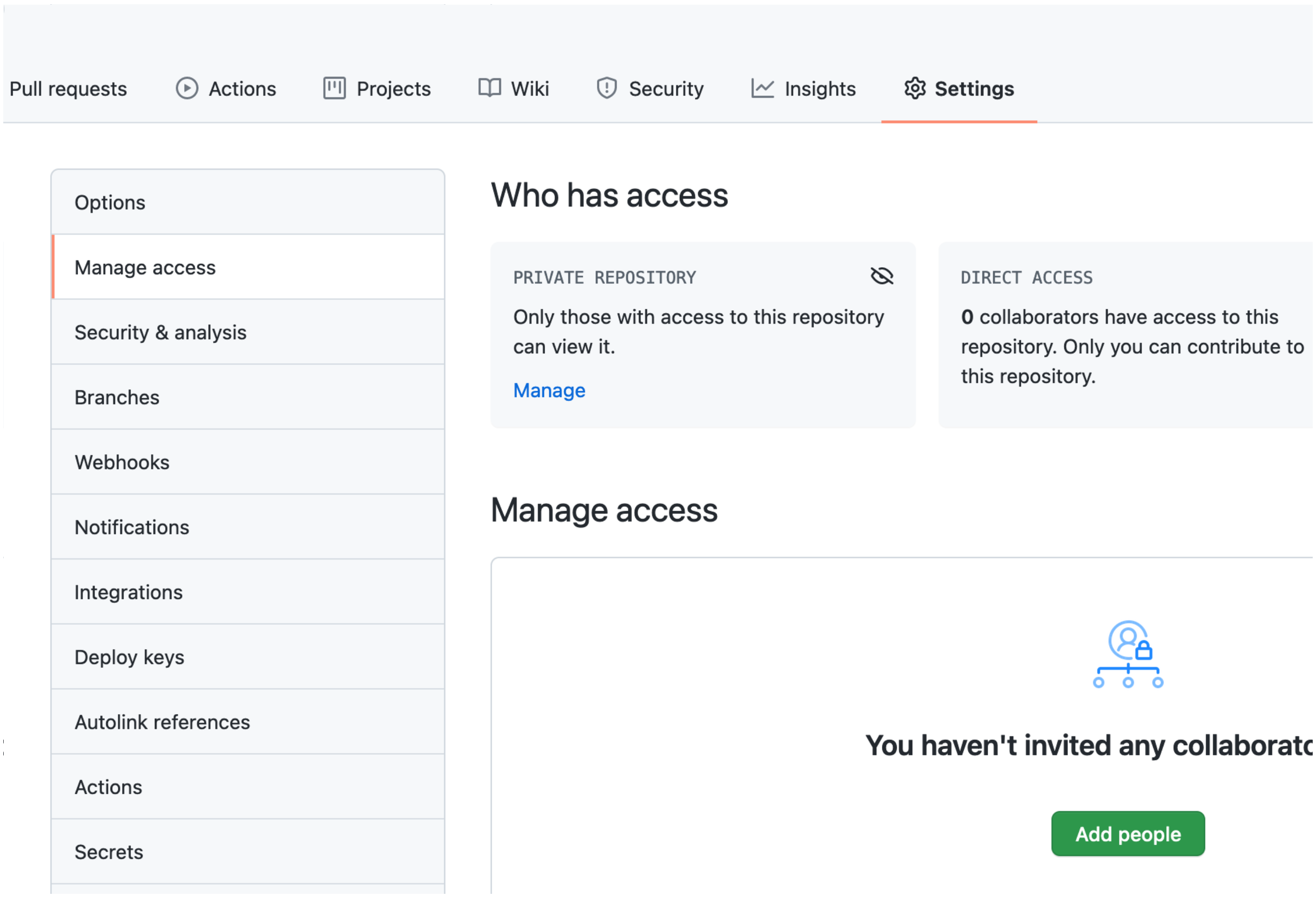
VisualStudio

IGNORE

- Just a file called '.gitignore'
- Which contains regexp of files to be ignored

```
.svn
log/*.log
tmp/**
node_modules/
.sass-cache
```

ADD COLLABORATORS



Pull requests　Actions　Projects　Wiki　Security　Insights　Settings

Options
Manage access
Security & analysis
Branches
Webhooks
Notifications
Integrations
Deploy keys
Autolink references
Actions
Secrets

Who has access

PRIVATE REPOSITORY
Only those with access to this repository can view it.
Manage

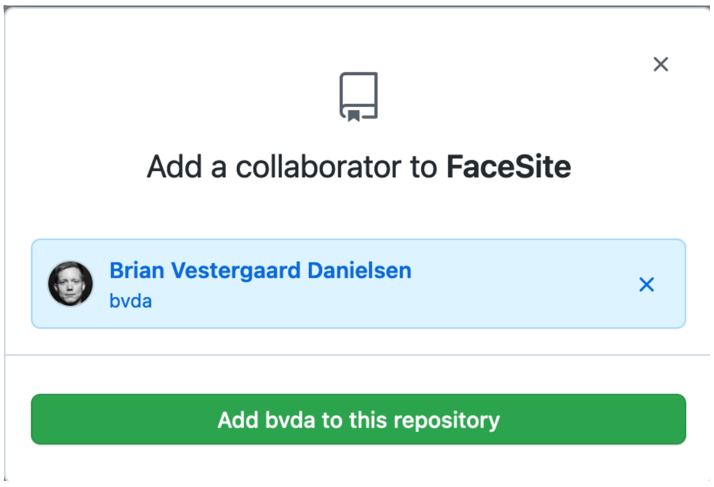
DIRECT ACCESS
0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborato

Add people

INVITE USERS



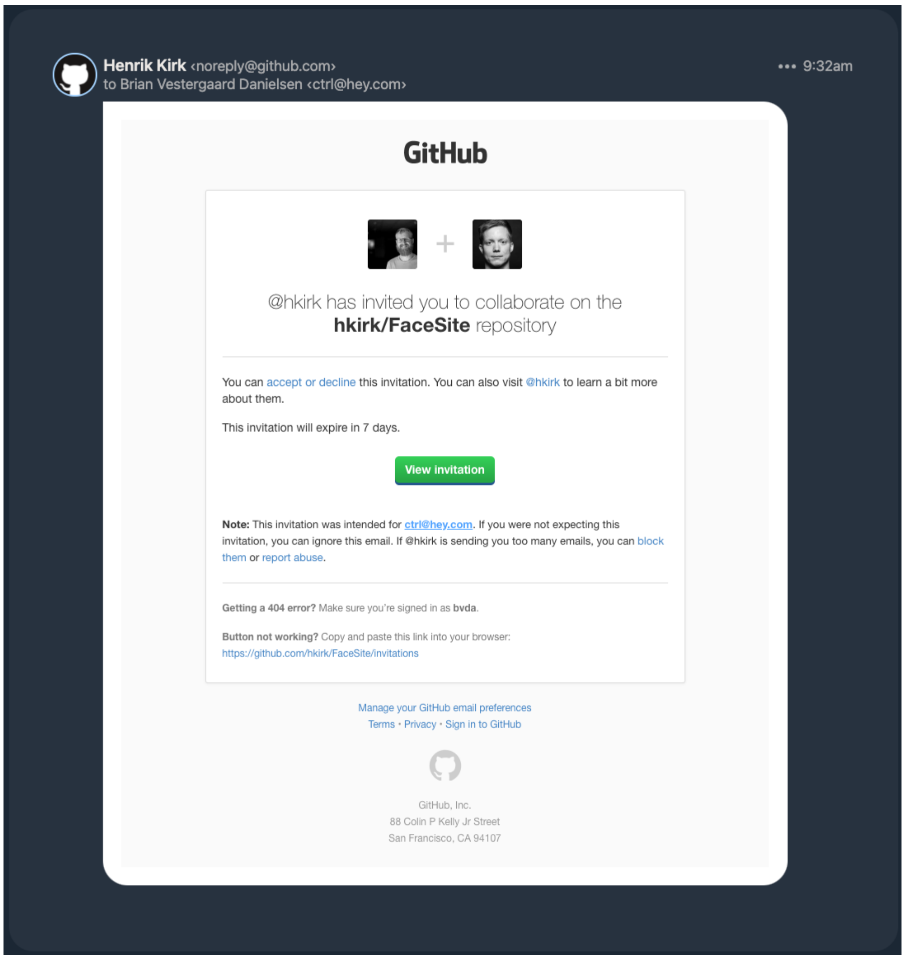
Add a collaborator to **FaceSite**

Brian Vestergaard Danielsen
bvda

Add bvda to this repository

CONFIRMATION

Remember to answer confirmation email



Henrik Kirk ‹noreply@github.com›
to Brian Vestergaard Danielsen ‹ctrl@hey.com›

••• 9:32am

GitHub

@hkirk has invited you to collaborate on the **hkirk/FaceSite** repository

You can accept or decline this invitation. You can also visit @hkirk to learn a bit more about them.

This invitation will expire in 7 days.

Note: This invitation was intended for ctrl@hey.com. If you were not expecting this invitation, you can ignore this email. If @hkirk is sending you too many emails, you can block them or report abuse.

Getting a 404 error? Make sure you're signed in as **bvda**.

Button not working? Copy and paste this link into your browser:
https://github.com/hkirk/FaceSite/invitations

Manage your GitHub email preferences
Terms • Privacy • Sign in to GitHub

GitHub, Inc.
88 Colin P Kelly Jr Street
San Francisco, CA 94107

CREATE PROJECT LOCALY

Or locally on console

```
$ git init
## later attach a remote repository
$ git add remote origin https://github.com/hkirk/FaceSite
```

COMMIT

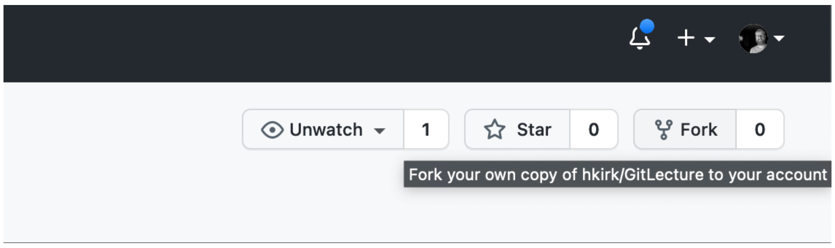
- The act of creating a snapshot
  - and the actually snapshot
- A repository consists of a series of commits
- Each commit consists of
  - Information about how file has changed
  - Reference to previous commit (parent commit)
  - A hash code

REPOSITORY

- A collection of the files
- and their history
- Will live locally and possible also on a remote server
  - Cloning is the act of copying the content
- Pulling from a repository
  - Copying remote changes to local
- Pushing to a repository
  - Moving local changes to remote

CLONING REPOSITORY

1. Goto www.github.com/hkirk/GitLecture
2. Fork this to optain you own copy



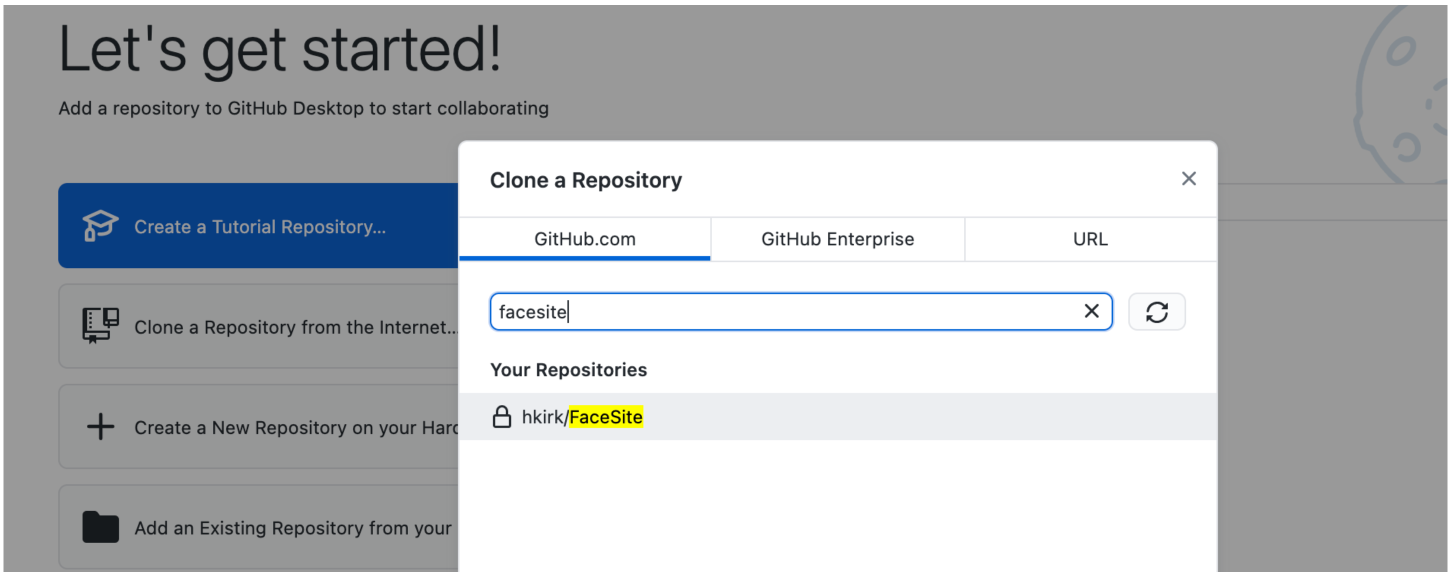
Fork your own copy of hkirk/GitLecture to your account

3.

```
$ git clone git@github.com:hkirk/GitLecture.git
```

CLONING VIA GITHUB DESKTOP

or


Let's get started!
Add a repository to GitHub Desktop to start collaborating
Create a Tutorial Repository...
Clone a Repository from the Internet...
Create a New Repository on your Hard
Add an Existing Repository from your
Clone a Repository
GitHub.com
GitHub Enterprise
URL
facesite
Your Repositories
hkirk/FaceSite

Speaker notes

If you have cloned before forking

```
$ git remote -v
$ git remote remove origin
$ git remote add origin git@github.com:Henrik-Personal/GitLecture.git
$ git push --set-upstream origin main
```

BASIC GIT COMMANDS 1

- **status**
  - shows which branch you are one (more about branches later)
  - shows working tree information
  - shows how your branch are compared to remove branch
- **log**
  - shows you snapshot history
  - *-n*, *--oneline*, *--graph*

BASIC GIT COMMANDS 2

- **add**
  - adds file to staging
- **commit**
  - commits files added to staging to repository
  - *-m*

OTHER USEFULL (NON-GIT) COMMANDS

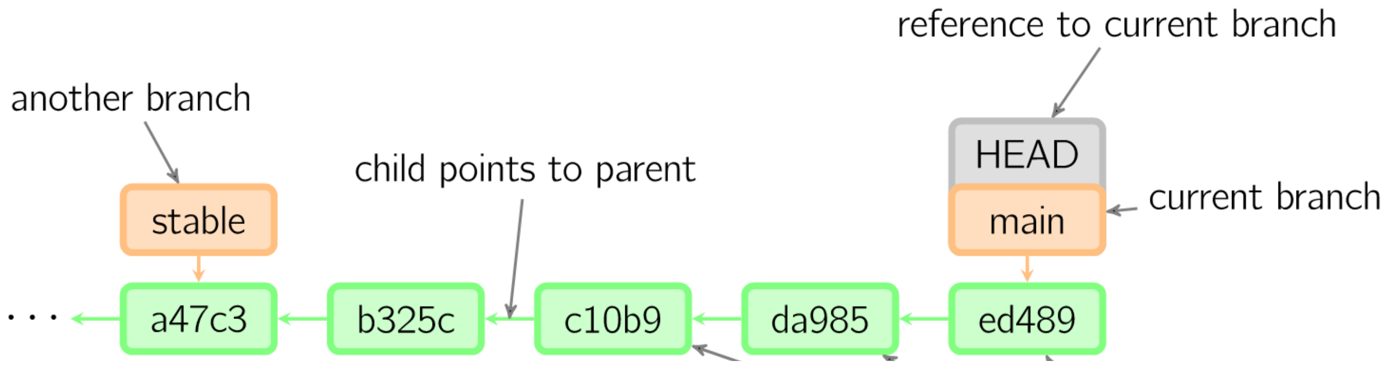
- **touch**
  - creates an empty file
- **echo**
  - prints
- **>>**
  - appends output from left side to file on right side
  - `echo "Hej" >> file.txt`
- **>**
  - overwrites file on right side

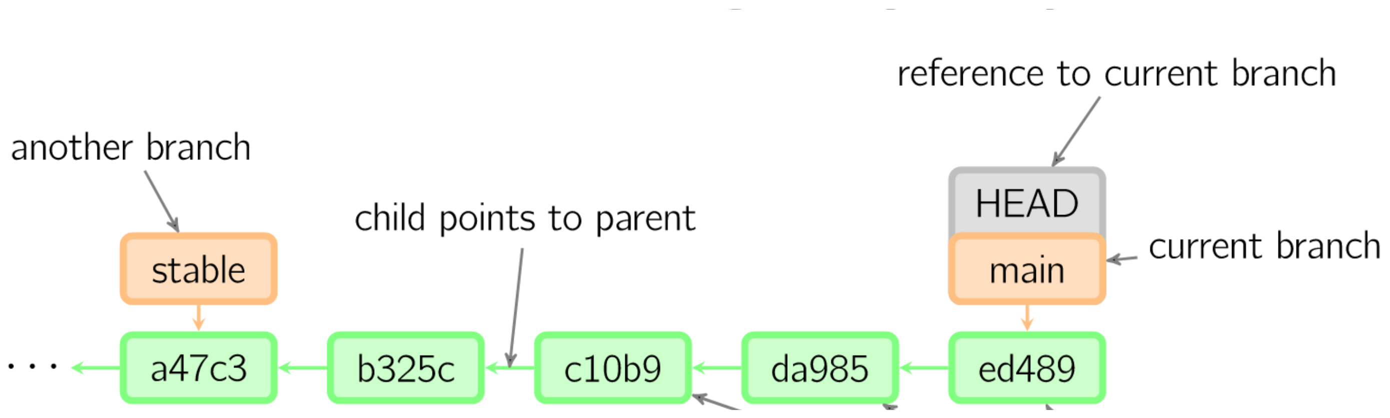
EX.: WHAT IS IN THE DIRECTORY

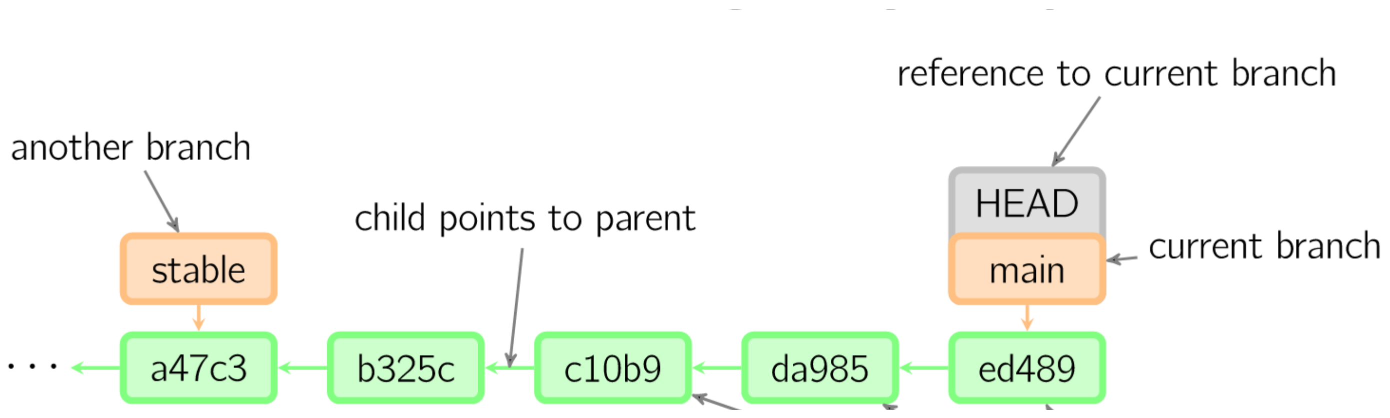
1. What does **status** and **log** say in GitLecture?
2. Create a file 'plan.txt' and **add** to staging
3. Check *status*
4. **add** and **commit** and check **status**
5. Change content of 'plan.txt'
6. Check **status**,
7. Then **add** and **commit**
8. How does the **log** look now?

BRANCHES

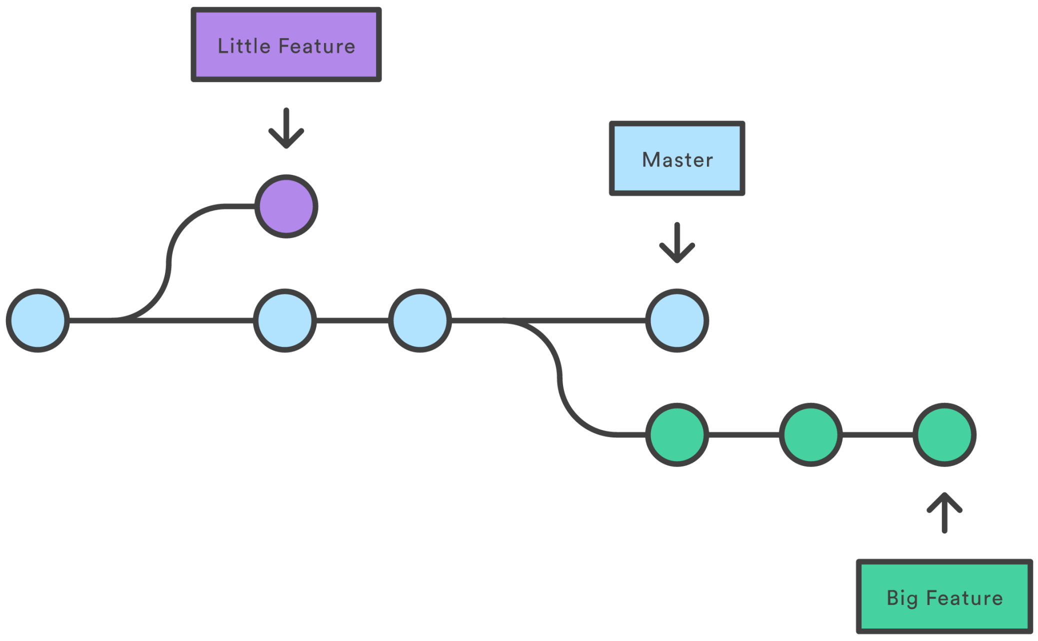
- A commit must 'live' on some branch
- There can be many parallel branches simultanius
- The main branch is typically called 'main', 'master' or 'development'


reference to current branch
another branch
child points to parent
HEAD
current branch
stable
main
a47c3
b325c
c10b9
da985
ed489


reference to current branch
another branch
HEAD
child points to parent
current branch
stable
main
. . .
a47c3
b325c
c10b9
da985
ed489


reference to current branch
another branch
child points to parent
HEAD
current branch
stable
main
. . .
a47c3
b325c
c10b9
da985
ed489

BRANCHES



COMMANDS REGARDING BRANCHES

- **checkout [name]**
  - switches to given branch
- **checkout -b [name]**
  - Creates the branch and switches
- **branch**
  - Lists all branches
- **branch [name]**
  - Creates a new branch
- **diff [name]**
  - Show differences on current and `[name]` branched

EX. BRANCHING

1. Which branch are you on?
2. Create a new **branch** '[initials]-branch', and show branches
3. Switch to the new branch and watch what the **status** displays now
4. How do you workspace now look?
5. Create a new file 'file.txt' containing you name
6. **Add** file and commit, check **log**
7. Switch back to the main branch
8. Create a file 'file2.txt' with some code
9. Add a commit file and check **log**
10. What are the difference on main and your new branch

MERGING

- Once a feature is done - you want to merge it back to you 'main' branch

```
      A---B---C topic
     /
D---E---F---G  main
```

MERGING

- Once a feature is done - you want to merge it back to you 'main' branch

```
      A---B---C topic
     /
D---E---F---G  main
```

```
$ git merge topic
```

MERGING

- Once a feature is done - you want to merge it back to you 'main' branch

```
      A---B---C topic
     /
D---E---F---G  main
```

```
$ git merge topic
```

```
      A---B---C topic
     /         \
D---E---F---G---H main
```

MERGING CONTINUED AND CLEANUP

- **merge** and **diff**
  - can handle multiple branches at once
- **branch -d [name]**
  - Deletes the branch with `[name]`

EX. MERGING

1. Create a branch 'uppercase' and check this out.
2. Edit the file greeting.txt and make an uppercase greeting
3. **Add** the file and commit. Check **log** with **`--oneline --graph --all`**
4. Checkout the 'main' branch
5. Check content of greeting.txt with `cat greeting.txt` or in an editor
6. What is the **diff**erence between 'uppercase' into 'main'
7. Then **merge** 'uppercase' into 'main'
8. What is the content of greeting.txt now?
9. Delete the uppercase branch

FIXING CONFLICTS

- Merging a branch can resolve in conflicts
  - This is merges that git cannot it self resolve.
- Example of conflict in a file

```
<<<<<<< HEAD
foreach (var i in range)
{
=======
for (int i = 0; i < 10; i++)
{
>>>>>>> conflicting-branch
```

RESOLVING CONFLICTS

- **status** will show unmerge paths
- Steps:
  1. Manually resolve each file
  2. **add** add each file to mark resolution
     - or **merge --abort** to abort merge

EX. WORKING WITH CONFLICTS

1. **merge** the branch 'origin/conflicting-greeting'
2. Use **status** to show changes
3. Use an editor to fix the conflicts
4. **status** also show instructions for how to resolve conflicts
5. What do **'log --oneline --graph --all'** show now?

AMEND

- Some times we commit and miss something important
    - amend can help us fix this
- **amend**
    - Replaces lastest commit on current branch

EX. USING AMEND

1. Create a file 'bar.txt', **add** and **commit**
2. What do **status** look like?
3. What do **log -p** show?
4. Guess which name appears the most on the enrollment list for today
5. Add that name to 'bar.txt' and **add** it
6. Amend these changes by **commit --amend**
7. Check **log p**
8. Try amending againg, what happens?

SHARING ON GITHUB

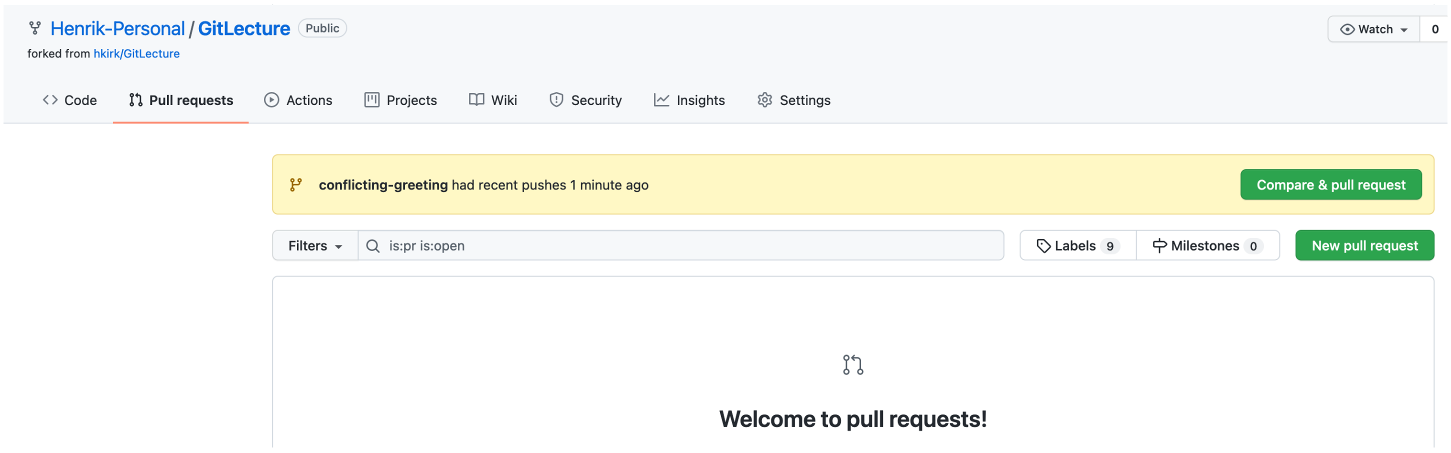
- **push**
  - Pushes current branch to remote
- **push --set-upstream**
  - Tells which remote a branch should be pushed to as default
  - Only nessesary for new branches
- **pull**
  - Incorporates changes from remote into current branch
  - shorthand for `git fetch; git merge FETCH_HEAD`

PULL REQUESTS

- Forking a repository on GitHub creates a new repository with the same code
  - and a link between these
- A pull requests is the tool to synchronize the to repositories

Note if you share access to a private repository pull request is no nessesary - because there are only one repository.

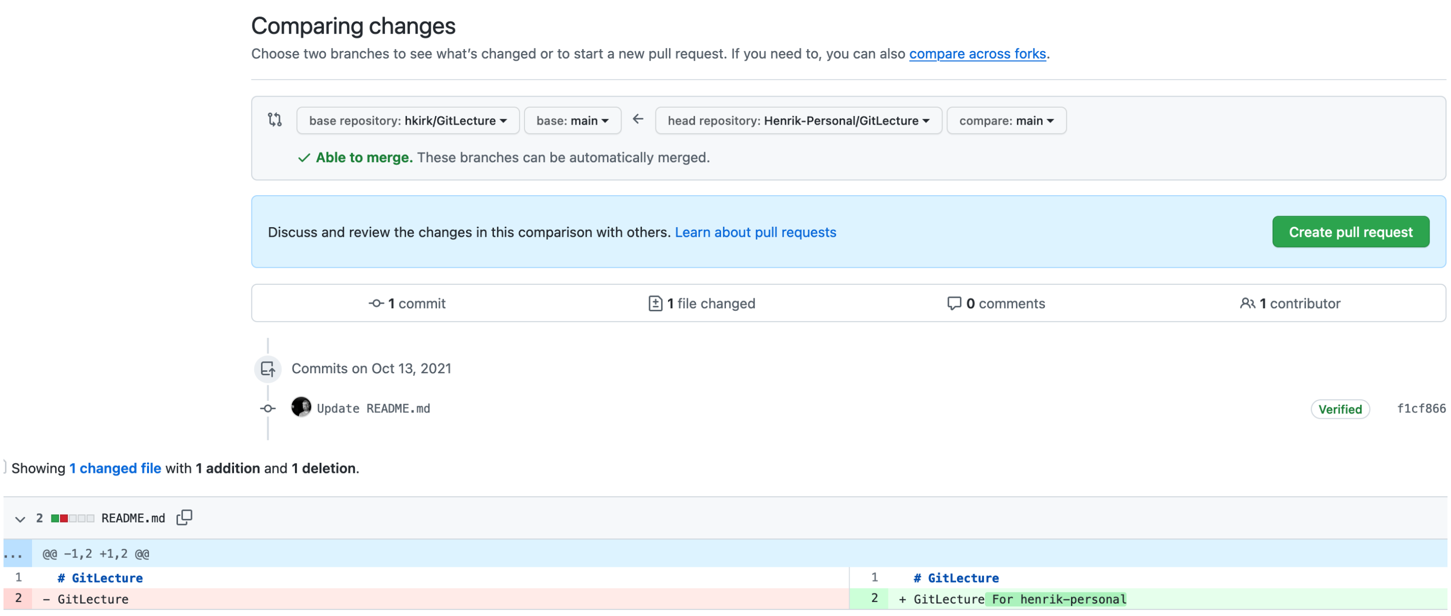
CREATING A PULL REQUEST 1



conflicting-greeting had recent pushes 1 minute ago

Welcome to pull requests!

CREATING A PULL REQUEST 1



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: hkirk/GitLecture ▾ | base: main ▾ ← head repository: Henrik-Personal/GitLecture ▾ | compare: main ▾

✓ **Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others. Learn about pull requests

1 commit | **1** file changed | **0** comments | **1** contributor

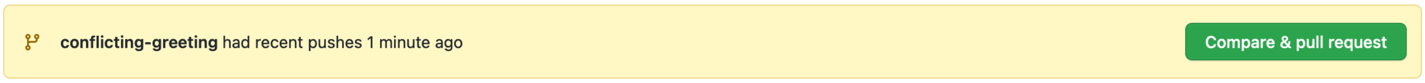
Commits on Oct 13, 2021

Update README.md — Verified — f1cf866

Showing 1 changed file with **1 addition** and **1 deletion**.

```
2 README.md
@@ -1,2 +1,2 @@
1    # GitLecture                 1    # GitLecture
2  - GitLecture                   2  + GitLecture For henrik-personal
```

UPDATES FROM MAIN REPOSITORY


conflicting-greeting had recent pushes 1 minute ago

EX. WORKING WITH GITHUB

1. **push** your changes on the branches 'main' and '[initials]-branch' to your github account
2. Create a pull request to 'hkirk/GitLecture'

Optional: (https://github.com/ECE-Henrik/hack-your-education-e21)[https://github.com/ECE-Henrik/hack-your-education-e21]

REFERENCES:

- A Visual Git Reference