

Intro to Git and Github

Hyper Island - FED27

Agenda

- Who am I?
- Terminal / Command prompt / Powershell
 - *Exercise: Installing Git*
 - *Exercise: Terminal*
- Git & github
- Git 1 - Basics
 - *Exercise: Git 1 - Init and committing*
- Git 2 - Branches and merging
 - *Exercise: Git 2 - Branching and merging*
- Git 3 - Rebasing
 - *Exercise: Git 3 - Rebasing*
- Git 4 - Conflict management
 - *Exercise: Git 4 - Conflict management*
- Git 5 - Remotes etc.
- Github 1 - Basics
 - *Exercise: Github 1 - Creating your profile*
- Github 2 - Github Pages and more
 - *Exercise: Github 2 - Adding your personal website to github*
- Git GUI clients
- Browser debugging

Who am I?

Frontend developer

- Cancerfonden
- Volvo Cars
- Swish
- Important Looking Pirates VFX

Terminal

- Used to navigate and control your computer via text
- Mac: Terminal
- Windows: Command Prompt, Powershell (or third party programs)
- Always has a context of the folder it's running in

Terminal

Paths

How to know where you are, and where to go

Paths are defined in a similar way as a web page, an address separated by slashes

/Users/perenstrom/desktop  , or C:\Users\perenstrom 

Terminal

Paths

ABSOLUTE PATH

- Starts at your hard drive
- Begins with /  or C:\ 
- /Users/perenstrom/desktop will always mean the desktop folder inside your user folder, no matter where you are

RELATIVE PATH

- Starts at your current position
- If you're in your user folder, desktop/images means the folder named images inside the folder named desktop inside your user folder
- If you're in a folder called myfolder, desktop/images means the folder named images inside the folder named desktop inside the folder myfolder

Terminal

Paths

SPECIAL FOLDER NAMES

- `.` (period) means the current folder, so `./desktop/images` is the same as `desktop/images`
 - sometimes needed to explicitly specify
- `..` means the parent folder, so if you're in your user folder `..` means the `Users` folder
 - If you're in your desktop folder `../downloads` means the `downloads` folder inside your user folder (one step up, and then into Downloads)

Terminal

Navigating, creating folders, and listing content

- To move around in your terminal, you use the `cd` command
 - If you're in your user folder, `cd desktop` means to enter the desktop folder
 - Likewise, using absolute paths, `cd /Users/perenstrom/desktop`

If you wonder where you are, just type `pwd` and the terminal will print the current folder

- To list the contents of the folder you're in, use `ls`  or `dir` 
 - Some commands have options, or flags, a dash and a letter, to change the behavior
 - For example, the `ls` command has a flag `-l` to give a more detailed list of the folder contents
 - To use this, type the command `ls -l` instead

Terminal

Navigating, creating folders, and listing content

- To create a folder inside the current folder, use the command `mkdir`
 - This command (and many others) take an input, written after the command. In this case the name of the folder to create
 - To create a folder called `my-folder`, type `mkdir my-folder`

Terminal

Good to know

- Use `tab` to autocomplete commands and folders
 - If you start to type `cd /Users/per` and press `tab` the terminal will autocomplete to `cd /Users/perenstrom/`
- Use `up arrow` to step back through your history of commands, which can be handy if you don't want to type a command again and recently used it

Exercise - Installing Git

Exercise - Terminal

Git

What is it

- Version management
 - Instead of creating assignment.docx, assignment-2.docx, assignment-final.docx, assignment-final-2.docx, assignment-final-final-for-real.docx
 - Like the version history in Google Docs, or Dropbox
 - Like the history in Photoshop
- Manual control over every step in the history, we decide ourselves when and what to add as a history step

Git

Why do we use it?

- Makes it easy to see when something was introduced or removed in your code base
- Allows you to work on different branches of your code
- Makes collaborating on code seamless
- Allows us to easily release new versions when we're done with a feature, instead of always doing changes in the published web page

Git

Important!

Don't add sensitive information (API Keys, passwords, etc) to git!

While we *can* rewrite the git history, it's a pain, and once a password has been added to the history it's visible forever.

Git

Repository

A project in Git is called a *repository*. The repository contains a hidden folder that contains all information related to git, the full history, different branches, etc.

We create a new repository for every project or web site.

To initialize a repository in a folder on our computer we use the command `git init`. This creates an empty repository in that folder, ready to accept our changes.

Git

Commit

- Every change in a git history is called a *commit*. It's a change we *commit* to.
- Every commit is stored as a difference between the previous step and the next, i.e. we don't store all code in all steps, only what has changed.
- These differences are called *diffs*.
- These changes can then be used to go back in time to any point in the code base's history.
- Every commit has an accompanying *commit message* that we write ourselves
 - The commit message should be brief and descriptive of the change
 - *The Will Rule*, any commit message should make sense if we say "This commit will" before the message
 - Good: "Add an image", "Fix bug in login", "Rework the home page"
 - Bad: "Added image", "Fixed bug", "Reworked the home page", "fix", "asdfasdfa"

Git

The stage, and tracking

- To commit changes to a file, we need to tell Git to *track* that file
- All files start out untracked
- Every uncommitted change we are working on needs its file to be tracked, and then added to *the stage*, before we can commit it
- We can – and often do – stage only part of what we've changed and commit that
- Tracking and staging are conveniently done in one command, the `git add` command
 - `git add` takes a third parameter which is the file we want to track and stage
 - If we want to track and stage a file called `index.html`, we use the command `git add index.html`
 - We can also track and stage all changed files at the same time with `git add .` (notice the period)

Git

How often do we commit?

- General rule is to commit small and often
 - This could be several times an hour, or a couple of times a day
- A goal is that the code base is always working at every commit
 - This means that when we're building web apps in Javascript, we should not commit code that is in a state that doesn't work
 - Doing many small commits is very helpful when collaborating, it makes it easy to track how files have evolved over time, and makes combining different changes from different people much easier

Exercise

Git 1: Init and committing

Git

Branches

- All commits in a git repository belongs to a branch
- By default a branch called `main` (or previously `master`) is created when initializing a repository
- We can look at all commits of a repository a bit like a tree, with branches of code diverging off, and unlike a tree merge together with the trunk again
- Branches are used heavily in development
 - Usually we have a `main` branch, which is always released to our url on the web
 - From that we branch off different branches where we can work on new features
 - On these branches we follow the commit small and often tactic
 - When we're done with our feature, we merge it back into `main` and release it to our users

Git

Branches

- Switching between branches are called Checking out
- To create a new branch and check it out we use the command `git checkout -b my-branch-name`
- Creating new branches are always made from the branch you currently have checked out, nothing is preventing us from branching from a branch
- In the terminal you will see what branch you're currently on, in my terminal it says `git: (main)` when I'm on the `main` branch

Git

Merging

- To merge our branch back into another branch we use the command `git merge`
- We merge a target branch, into the branch we have checked out
 - To merge a branch called `my-branch-name` into `main`, we first `git checkout main`, and then `git merge my-branch-name`
- There are a few merge *strategies* that git automatically chooses between
 - The two common ones are *fast-forward* and *merge commit*
 - *fast-forward* takes all commits on our `my-branch-name` and adds them one by one to the `main` branch
 - *merge commit* takes all our commits and mashes them together, and creates a new single commit on `main` with all of our changes
- Merging does not delete a branch, we can continue to commit to either branch afterwards

Exercise

Git 2: Branching and merging

Git

Rebasing

- Often when working on a project with others, stuff will change on `main` while you are working on your feature branch.
- We often want to incorporate those newest changes on `main` in our feature branch and continue our work.
- To do this we can either *merge* or *rebase*
 - Merging is the easiest, where we merge `main` into our branch (by checking out our feature branch and doing `git merge main`)
 - Rebasing means taking all of the commits we've done on our feature branch, and applying them one by one on the new state of `main`. This makes it look like we really started our branch from the latest commit on `main`
 - Rebasing makes for a cleaner git history, but can be a bit tricky to do

Exercise

Git 3: Rebasing

Git

Merge conflicts

- When the same line has changed in two different branches and we try to merge them together, we will encounter a *merge conflict*
- This means that Git can't automatically figure out what version to keep, and we have to help it out
- Merge conflicts pause a merge in a conflict stage, and waits for you to resolve the files and commit the correct version
- In the conflicting file it will look something like:

```
1  << HEAD
2  This is how the line looks on our current branch
3  =====
4  This is how the line looks on the branch we want to merge into our
5  >> main
```

- We select our change by replacing this whole block of five lines with the single line we want to keep
- We then commit this change with `git add .` and `git commit -m "merge branch main into feature"`

Git

Merge conflicts

- Merge conflicts often occur when we're done with a feature and want to merge it into `main`, when something has happened on `main`
- Best practice is to:
 - merge the other way around first, merge `main` into our feature branch
 - resolve the merge conflict on our feature branch and commit that
 - merge the feature branch into `main`, which should now be conflict free

Exercise

Git 4: Conflict management

Git

Stashing

- Sometimes you want to switch branch to look at something on another branch, but you have changed some files that you are not ready to commit yet
- You can use `git stash` to temporarily store your ongoing changes without committing them
- You can then switch between branches, do your thing, and then come back to your branch
- When you're ready to continue your ongoing work, use `git stash pop` to bring your changes out from storage

Git

Amending

- You can change your last commit if you noticed that you did something wrong and don't want to make a whole new commit to fix that, or if you want to change your commit message
- If you want to add a new change, add the file with `git add .`
- Then commit while using the `--amend` flag: `git commit --amend -m "My edited message"`

Git

Remotes: Push and Pull

So far we've only worked with a local git repository on your computer, but this is usually not the case. Usually our code repository is also stored somewhere online, in for example GitHub (which we'll come to later). These remote locations of your local repository is called *remotes*.

- Remotes are other locations of your repository
- A local branch is usually tied to a remote branch
- To fetch the latest changes of a branch from the remote (say if the code has changed on github by one of your colleagues), you *pull* those changes to your local branch
- When you have done something to a local branch and want to upload it to the remote, you *push* your changes to the remote branch

Git

Bonus: Git på svenska

When talking about Git in Swedish, things get weird quite quickly

"Kan du pulla mina ändringar till branchen?"

A GitHub user named Björn has made a list of Swedish terms for the Git terms we've come across, to make it easier to communicate with fellow Swedes. Some examples:

"Kan du rycka grenen jag just ympade och knuffa till github?"

"Skicka en ryckbegäran när du är färdig med sammanfogningen!"

"Hoppsan, jag råkade visst kraftknuffa mot mäster-grenen"

Full list at: <https://github.com/bjorne/git-pa-svenska>

Full disclaimer, no-one uses this

GitHub

Git vs Github

- Git is the underlying mechanic of committing changes etc.
- GitHub is a web service which can act as a remote for your repositories
- GitHub also adds additional functionalities and work flows for a complete development management solution
- GitHub is owned by Microsoft (and since recently moved to their AI department...)

GitHub

Repositories

- In GitHub you can create your own repositories, or have access to other's repositories
- Repositories can be *Private* (hidden to anyone without access), or *Public* (visible to anyone)
- Repositories (and users) can be part of organizations, which in turn can have their own repositories



Anatomy of a repository - Code tab

Code tab – A repository's home page, allows you to browse the source code and read the README

The screenshot shows the GitHub repository interface for 'perenstrom/awards-night'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository name 'perenstrom / awards-night' is displayed, along with a search bar and various repository stats: 3 Branches, 18 Tags, 374 Commits, and 0 forks. The 'Code' tab is currently selected. On the right, there's an 'About' section with the repository description 'WIP. Awards night betting visualiser', a link to the website 'www.awardsnight.app', and metrics for stars (0), activity (1 watching), and forks (0). The main content area shows a list of recent commits:

Author	Commit Message	Date	Commits
perenstrom	Merge branch 'develop'	36ae5d9 · 5 months ago	374 Commits
app	Add admin action to close betting	6 months ago	
components	Fix enabling/disabling of nomination list depend...	6 months ago	
lib	Rework caching for betting actions	6 months ago	
prisma	Add awardsFinished to year schema	6 months ago	



Anatomy of a repository - Code tab

Watching repositories – Watched repositories gives you a notification if something happens in them

The screenshot shows the GitHub repository interface for 'perenstrom / awards-night'. The 'Code' tab is selected. In the top navigation bar, there is a 'Unwatch' button with a count of '1'. This button is highlighted with a red box. Below the navigation bar, the repository name 'awards-night' and its status as 'Public' are displayed. On the left, there are dropdown menus for 'master', '3 Branches', and '18 Tags'. A search bar labeled 'Go to file' is followed by a '+' button and a 'Code' button. To the right of the search bar, there is an 'About' section with the following details:

- WIP. Awards night betting visualiser
- www.awardsnight.app
- Readme
- Activity
- 0 stars
- 1 watching
- 0 forks

A list of recent commits is shown on the left side of the page:

Author	Commit Message	Date
perenstrom	Merge branch 'develop'	36ae5d9 · 5 months ago
app	Add admin action to close betting	6 months ago
components	Fix enabling/disabling of nomination list depend...	6 months ago
lib	Rework caching for betting actions	6 months ago
prisma	Add awardsFinished to year schema	6 months ago

GitHub

Anatomy of a repository - Code tab

Starring repositories – A bit like *liking*. Can be a good indicator of how good a tool is

The screenshot shows a GitHub repository page for 'perenstrom / awards-night'. The 'Code' tab is selected. At the top right, there is a 'Star' button with '0' stars, which is highlighted with a red box. Below the header, there's a search bar and several navigation links: Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository card shows it's public and has 3 branches and 18 tags. The commit list includes:

- perenstrom Merge branch 'develop' · 36ae5d9 · 5 months ago · 374 Commits
- app Add admin action to close betting · 6 months ago
- components Fix enabling/disabling of nomination list depend... · 6 months ago
- lib Rework caching for betting actions · 6 months ago
- prisma Add awardsFinished to year schema · 6 months ago

On the right side, there's an 'About' section with the repository description 'WIP. Awards night betting visualiser', a link to 'www.awardsnight.app', and stats: 0 stars, 1 watching, and 0 forks.

GitHub

Anatomy of a repository - Code tab

Forking repositories – If you want to create your own version from another repository, you fork that repository

The screenshot shows a GitHub repository page for 'awards-night'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation is a header with the repository name 'awards-night' (Public), a 'Unpin' button, an 'Unwatch' button (with 1 watch), a 'Fork' button (with 0 forks, highlighted with a red box), a 'Star' button (with 0 stars), and user profile icons. The main content area displays the 'master' branch, 3 branches, and 18 tags. A search bar allows navigating to specific files. On the right, there's an 'About' section with details about the repository, including its purpose ('WIP. Awards night betting visualiser'), a link to 'www.awardsnight.app', and metrics like 0 stars, 1 watching, and 0 forks.

perenstrom / awards-night

Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

awards-night Public

Unpin Unwatch 1 Fork 0 Star 0

master 3 Branches 18 Tags Go to file + Code About

perenstrom Merge branch 'develop' 36ae5d9 · 5 months ago 374 Commits

app Add admin action to close betting 6 months ago

components Fix enabling/disabling of nomination list depend... 6 months ago

lib Rework caching for betting actions 6 months ago

prisma Add awardsFinished to year schema 6 months ago

WIP. Awards night betting visualiser
www.awardsnight.app

Readme Activity 0 stars 1 watching 0 forks

GitHub

Anatomy of a repository - Code tab

Cloning – If you press the Code button, you will get info on how to clone (create a local copy) the repository

The screenshot shows the GitHub repository page for the user 'perenstrom' with the repository name 'awards-night'. The 'Code' tab is selected, and its dropdown menu is open, highlighted with a red box. The menu contains options for cloning the repository via Local (Clone, HTTPS, SSH, GitHub CLI), opening it with GitHub Desktop, or downloading it as a ZIP file. The repository details on the right include the description 'WIP. Awards night betting visualiser', the URL 'www.awardsnight.app', and metrics like 0 stars, 1 watching, and 0 forks. Deployment history shows two entries: 'Preview' and 'Production', both from 5 months ago.

master ▾ 3 Branches 18 Tags

Go to file t + <> Code ▾

perenstrom Merge branch 'develop' ✓

app Add admin

components Fix enable

lib Rework

prisma Add awa

public Fix landi

services Add adm

types Add awardsFinished to year-map 6 months ago

utils Remove old files and dependencies last year

any example Update to react 16 and auth 0.4 0 months ago

Local Codespaces

Clone

HTTPS SSH GitHub CLI

<https://github.com/perenstrom/awards-night>

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

About

WIP. Awards night betting visualiser

www.awardsnight.app

Readme

Activity

0 stars

1 watching

0 forks

Deployments 252

Preview 5 months ago

Production 5 months ago

+ 250 deployments



Anatomy of a repository – Issues tab

Issues – Can be used for project management, or bug reporting for your users

The screenshot shows the GitHub Issues tab for the repository `perenstrom / awards-night`. The top navigation bar includes links for Code, Issues (1), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A search bar with placeholder text "Type / to search" is positioned above the issues list. Below the search bar are buttons for Labels, Milestones, and a prominent green "New issue" button. The main area displays a list of open issues, with filters for Open (1) and Closed (16) issues. One issue is highlighted: #18, titled "Example Issue", which was opened by `perenstrom` and is described as having been opened "now".



Anatomy of a repository – Projects tab

Projects – Used to organize issues, track goals, etc.

The screenshot shows the GitHub interface for the repository `perenstrom / awards-night`. The top navigation bar includes links for Code, Issues (1), Pull requests, Actions, Projects (1, highlighted with a red underline), Wiki, Security, Insights, and Settings. A search bar at the top right contains the placeholder "Type / to search". Below the navigation, there's a search bar with the query "is:closed" and a "Link a project" button. A green "New project" button is also visible. The main content area displays a summary with "1 Open" and "1 Closed" issues. A "Kanban board" section is shown, labeled as "Private" and updated on Feb 19. A "Sort" dropdown menu is located in the top right corner of this section. The bottom of the page features a footer with the GitHub logo and links to Terms, Privacy, Security, Status, Docs, Contact, Manage cookies, and a "Do not share my personal information" option.

perenstrom / awards-night

Type / to search

Code Issues 1 Pull requests Actions Projects 1 Wiki Security Insights Settings

is:closed

Link a project

New project

1 Open 1 Closed Sort

Kanban board Private #1 updated on Feb 19



© 2025 GitHub, Inc.

Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

GitHub

Anatomy of a repository – Projects tab

Projects – Used to organize issues, track goals, etc.

The screenshot shows a GitHub Kanban board titled "Kanban board" under the "perenstrom / Projects" section. The board has four columns: "To do" (1 item), "In progress" (0 items), "Waiting for release" (0 items), and "Done" (multiple items). The "To do" column contains an item titled "awards-night #18" with the description "Example Issue". The "Done" column has several items, with the top one being "awards-night #18" and others partially visible. A search bar at the top right allows for keyword filtering. Navigation icons for "Add status update", "View 1", "New view", and "More" are also present.



Anatomy of a repository – Pull requests tab

Pull Requests – When you have made changes and want to merge them into a project on GitHub, you open a Pull Request (PR). Named because you want the repository to *Pull* your changes into their code base, another name could be "Merge request".

Vital for collaboration with others. Every time a change you've made are to be added to the product, you open a PR, your colleagues review and approve it, and then you merge it.

The screenshot shows the GitHub interface for the repository `perenstrom / awards-night`. The top navigation bar includes links for Code, Issues (1), Pull requests (selected), Actions, Projects (1), Wiki, Security, Insights, and Settings. Below the navigation is a search bar with the placeholder "Type / to search" and various filter buttons: Filters, Labels (12), Milestones (1), and a green "New pull request" button. A link to "Clear current search query, filters, and sorts" is also present. At the bottom, there are filters for Open (0) and Closed (1) pull requests, along with dropdown menus for Author, Label, Projects, Milestones, Reviews, Assignee, and Sort. A single pull request is listed: #17 by `App router`, which was merged on Feb 22, 2024. A comment icon indicates one comment on this pull request.

GitHub

Anatomy of a repository – Pull requests tab

To open a new PR, press the New pull request button.

The screenshot shows the GitHub interface for a repository named 'perenstrom-fed27'. The 'Pull requests' tab is selected, indicated by a red underline. A modal window titled 'Label issues and pull requests for new contributors' is displayed, explaining that GitHub will help potential first-time contributors discover issues labeled with 'good first issue'. A 'Dismiss' button is in the top right corner of the modal. Below the modal, there are filters for 'Filters' (set to 'is:pr is:open'), a search bar, and buttons for 'Labels' (9) and 'Milestones' (0). A prominent green button labeled 'New pull request' is highlighted with a red box. At the bottom of the page, there is a summary of pull requests: '1 Open' and '1 Closed', along with a link to an open pull request: '#2 Update README.md' opened 1 minute ago by perenstrom-fed27. The GitHub logo is in the bottom right corner.

GitHub

Anatomy of a repository – Pull requests tab

In the top you will select target branch, and source branch.

- Fill in a good descriptive title of your change (1)
- Write a description of your change (2)
- Assign yourself (usually) (3)
- Add reviewers if needed (4)
- Press Create pull request (5)

Open a pull request

The change you just made was written to a new branch named `perenstrom-fed27-patch-1`. Create a pull request below to propose these changes. [Learn more about diff comparisons here.](#)

base: main compare: perenstrom-fed27-patch-1

Add a title
① Update README.md

Add a description
② I've changed a line in the Readme, have a look!

④ Reviewers
No reviews

③ Assignees
perenstrom-fed27

Labels
None yet

Projects
None yet

Milestone
No milestone

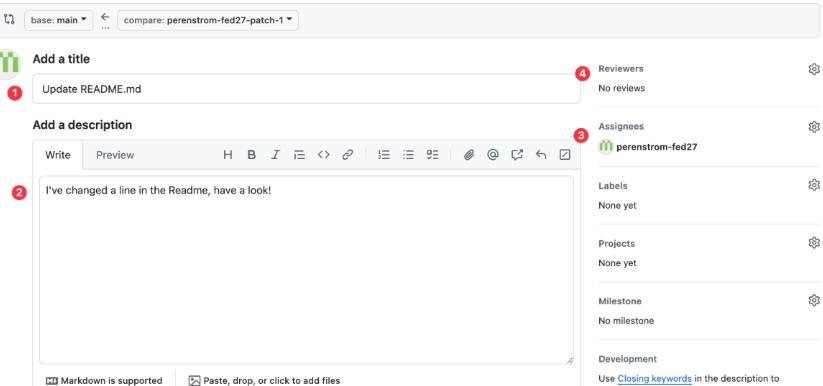
Development
Use [Closing keywords](#) in the description to automatically close issues

Markdown is supported Paste, drop, or click to add files

Create pull request

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Helpful resources
[GitHub Community Guidelines](#)



GitHub

Anatomy of a repository – Pull requests tab

The PR page now display a timeline of changes, with your description at the start.

The screenshot shows a GitHub repository page for 'perenstrom-fed27 / perenstrom-fed27'. The 'Pull requests' tab is selected, showing a single open pull request titled 'Update README.md #2'. The pull request has 1 commit from 'perenstrom-fed27-patch-1' merged into the 'main' branch. A comment from 'perenstrom-fed27' states: 'I've changed a line in the Readme, have a look!'. The pull request is verified and has a SHA of '4bcb3d9'. The right sidebar shows the pull request's status: 'Reviewers' (No reviews), 'Assignees' (perenstrom-fed27), and 'Labels'.

perenstrom-fed27 / perenstrom-fed27

Type to search

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

Update README.md #2

Open perenstrom-fed27 wants to merge 1 commit into `main` from `perenstrom-fed27-patch-1`

Conversation 0 Commits 1 Checks 0 Files changed 1

perenstrom-fed27 commented now

I've changed a line in the Readme, have a look!

Verified 4bcb3d9

Reviewers
No reviews
Still in progress? [Convert to draft](#)

Assignees
perenstrom-fed27

Labels



Anatomy of a repository – Pull requests tab

The Commits tab shows all commits that has been made on the branch you want to merge.

The screenshot shows the GitHub interface for a repository named `perenstrom-fed27 / perenstrom-fed27`. The `Pull requests` tab is active, indicated by a red underline. The main title of the pull request is `Update README.md #2`. A green button labeled `Open` is visible. The description states: `perenstrom-fed27 wants to merge 1 commit into main from perenstrom-fed27-patch-1`. Below the description, there are sections for `Conversation 0`, `Commits 1`, `Checks 0`, and `Files changed 1`. A summary at the bottom right shows `+1 -1 3 0 0 0 0`. A detailed view of the single commit is shown below, titled `Update README.md`, authored by `perenstrom-fed27` 8 minutes ago, with a `Verified` status, a commit hash of `4bcb3d9`, and standard copy/paste/share icons.

GitHub

Anatomy of a repository – Pull requests tab

The Files changed tab shows all the files that has changed, and all changes in those files.

Here we see that I've changed line 3, the red part is the old one, and the green part is the new.

The screenshot shows the GitHub interface for a pull request. The top navigation bar includes tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). A summary at the top right indicates +1 -1 changes. Below the tabs are filters for Changes from all commits, File filter, Conversations, Jump to, and settings. A progress bar shows 0 / 1 files viewed. Action buttons include Review in codespace and Review changes.

The main area displays the file README.md with a diff view. The file contains the following content:

```
... ... @@ -1,3 +1,3 @@
1 1 ## Hi there 🎉
2 2
3 - Here's a description, I've added some text here.
3 + Here's a description, I've changed some text here.
```

The first two lines are standard text. The third line is highlighted with a pink background, indicating it was deleted. The fourth line is highlighted with a green background, indicating it was added.

GitHub

Anatomy of a repository – Pull requests tab

If you hover a line, a plus appears which you can click to add a comment. This comment can either be posted directly, or added to a Review.

The screenshot shows the GitHub Pull Requests tab for a repository. At the top, there are tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). The 'Files changed' tab is selected. On the right, there are status indicators: +1 (green), -1 (red), and several neutral icons. Below the tabs, there are filters: 'Changes from all commits' (dropdown), 'File filter' (dropdown), 'Conversations' (dropdown), 'Jump to' (dropdown), and a gear icon. To the right of these are buttons for 'Review in codespace' and 'Review changes' (which is highlighted in green).

The main area shows a file named README.md with two changes: one deletion and one addition. The deleted line is '## Hi there'. The added line is '+ Here's a description, I've changed some text here.' A blue arrow points to the '+' sign in the second line, with the text 'Hover here' written above it. The line numbers 1, 2, and 3 are visible on the left. The code editor has a toolbar below it with buttons for Write, Preview, and various text formats (bold, italic, etc.). A large input field at the bottom is labeled 'Leave a comment'. At the very bottom, there are links for 'Markdown is supported' and 'Paste, drop, or click to add files'.

GitHub

Anatomy of a repository – Pull requests tab

To review someone else's changes (or your own I guess), press the Review changes (or Finish your review), add a comment, choose if you approve or want some changes to be made, and press Submit review.

The screenshot shows the GitHub Pull Requests tab for a repository. At the top, there are tabs for Conversation (0), Commits (1), Checks (0), Files changed (1), and a summary of +1 -1 changes. Below the tabs is a toolbar with filters: Changes from all commits, File filter, Conversations, Jump to, and a gear icon. To the right of the toolbar are buttons for 0 / 1 files viewed, Review in codespace, and a green button labeled "Finish your review 1". A red box highlights the "Finish your review" button.

The main area displays the README.md file with two changes:

- Line 1: @@ -1,3 +1,3 @@
- Line 2: ## Hi there 🌟
- Line 3: - Here's a description, I've added some text here.
- Line 3: + Here's a description, I've changed some text here.

A comment from user perenstrom is shown, with a Pending status and the message "This looks good! 👍". Below the comment is a reply input field.

A modal window titled "Finish your review" is open. It contains a rich text editor with a "Write" tab (highlighted by a red box) and a "Preview" tab. The preview area shows the text "Looks good!". Below the editor, it says "Markdown is supported" and "Paste, drop, or click to add files".

At the bottom of the modal, there are three options:

- Comment**: Submit general feedback without explicit approval.
- Approve**: Submit feedback approving these changes. (highlighted by a red box)
- Request changes**

GitHub

Anatomy of a repository – Pull requests tab

Our colleagues review can now be seen in the Conversation tab.

The screenshot shows the GitHub Conversation tab for a pull request. At the top, a message from user **perenstrom** indicates they approved the changes [1 minute ago](#). Below this, **perenstrom** left a comment saying "Looks good!" with a smiling emoji. The code diff for **README.md** shows two additions: "## Hi there 🤝" and "Here's a description, I've changed some text here." A reply from **perenstrom** states "This looks good! 🤘" with a smiling emoji. A "Reply..." button and a "Resolve conversation" button are at the bottom.

perenstrom approved these changes [1 minute ago](#)

View reviewed changes

perenstrom left a comment

Contributor ...

Looks good!

README.md

```
... ... @@ -1,3 +1,3 @@
1 1 ## Hi there 🤝
2 2
3 - Here's a description, I've added some text here.
   + Here's a description, I've changed some text here.
```

perenstrom [1 minute ago](#)

This looks good! 🤘

Contributor ...

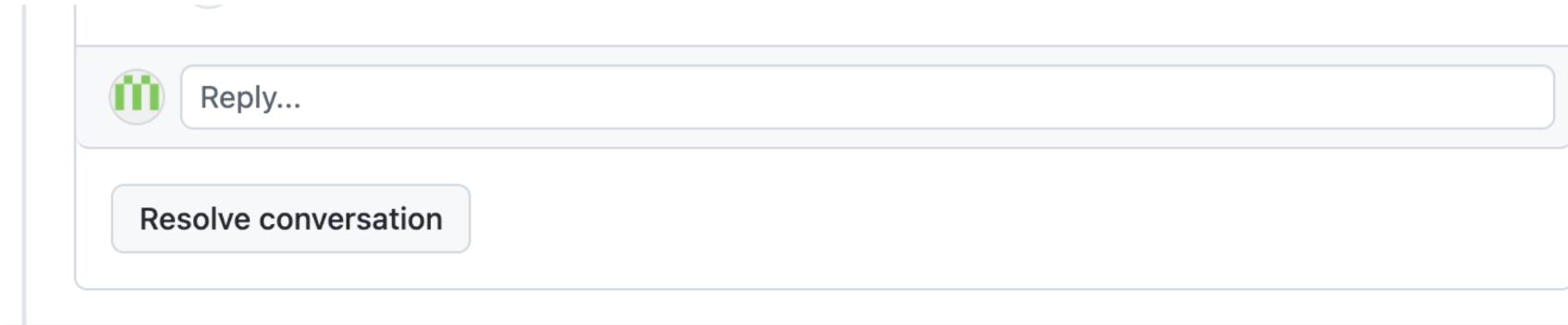
Reply...

Resolve conversation

GitHub

Anatomy of a repository – Pull requests tab

If you want, you can reply to comments, and/or mark a conversation as "resolved"





Anatomy of a repository – Pull requests tab

When everyone is happy, press the Merge pull request to merge your changes into the target branch!

 **Changes reviewed**
No applicable reviews submitted by reviewers with write access.

 1 approval >

 **No conflicts with base branch**
Merging can be performed automatically.

Merge pull request ▾ You can also merge this with the command line. [View command line instructions.](#)

GitHub

Anatomy of a repository – Pull requests tab

The merge event can now be seen in the history, and there's a Delete branch button to remove your branch (if it was temporary). This deletion can always be reverted!

A screenshot of a GitHub pull request history page. At the top, a purple circular icon with a hand icon represents a merge event. Next to it is a green circular icon with a grid icon representing a user. The text reads: "perenstrom-fed27 merged commit 394d877 into main now". To the right is a "Revert" button. Below this is a horizontal grey line. Underneath the line, there is a purple rectangular box containing a white icon of a branch merging into another branch. To its right, the text says "Pull request successfully merged and closed". Below that, a message states: "You're all set — the perenstrom-fed27-patch-1 branch can be safely deleted." To the right of this message is a "Delete branch" button.

Markdown

The Microsoft Word™ of the web

- On the web, everything* is in plain text. That is, it doesn't have any formatting
- Markdown to the rescue!
- Markdown lets you add styling to text by using a simple syntax
- Markdown files end in the file extension `.md`
- All of this presentation, and all exercises are written in Markdown

Cheat sheet: <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>

Markdown

Syntax - Headings

Headings are made by starting the line with a `#`, the more `#` the smaller the heading.

```
# My heading will render a level 1 heading, corresponding to the <h1> html tag. ("Markdown") on this slide uses this
```

```
## My heading will render a level 2 heading, corresponding to the <h2> html tag. ("Syntax") on this slide uses this
```

...and so on

Markdown

Syntax - Formatting

- To make text **bold**, write `**my bold text**` or `_my italic text_`
- To make text *italic*, write `*my italic text*` or `_my italic text_`
- To make text ~~crossed out~~, write `~my crossed out text~`
- To make `inline code`, write `asdf` (surround with "back-ticks")
- To make multiline code, write a line with only three back-ticks (``) above, and below:

```
1 Here is
2 some multiline
3 code
```

Markdown

Syntax - Links and images

- To make a link, write `[Link text](https://google.com)`
- To inline an image, write `![Image description](https://google.com/logo.png)`

Exercise

GitHub 1: Creating your profile

GitHub 2

Publishing on GitHub Pages

- Publishing your code is called *deploying*
- A great way to get started is with GitHub Pages
- Every GitHub account comes with its own web page at `<username>.github.io`
- To publish your web page at that url, the Git Repository on Github must be named exactly the same
- Other repositories can also be published, but will be at `<username>.github.io/<repository-name>`
- Can be turned on in the repository settings

GitHub 2

Connecting existing code to a GitHub repository

- If you have a Git repository locally (if you've run `git init` on your computer)
- 2 ways
 - Cloning a repository and adding code locally
 - Creating an empty repository and point a local repository to that

Exercise

GitHub 2: Adding your personal website to GitHub

Git GUI Clients

You don't *have* to use the terminal

- There are many free and paid Git Clients
- These allow you to
 - see the graph of branches
 - select what to commit by clicking files
 - see diffs nicely formatted
 - commit and push
 - handle merge conflicts
- I don't use the terminal for Git, but it's very good to know what your client does behind the scenes, sometimes you will get in situations where the Git client isn't enough

Git GUI Clients

Suggestions

- Git GUI (Windows, Free) – Comes with the Git for Windows installer
- VS Code (Win/Mac, Free) – Your favourite editor actually comes with a Git client!
- GitHub desktop (Win/Mac, Free) – GitHub's own client
 - <https://github.com/apps/desktop>
- SourceTree (Win/Mac, Free) – Atlassian's client
 - <https://www.sourcetreeapp.com/>
- GitKraken (Win/Mac, Free & Paid) – My preference!
 - <https://www.gitkraken.com/>
- Fork (Win/Mac, Paid) – A popular premium choice
 - <https://git-fork.com/>

That's all folks!

Resources

- [Getting started with the terminal](#)
- [Safe terminal experimentation](#)
- [The Git manual](#)
- [GitHub's introduction to Git](#)
- [GitHub's Markdown documentation](#)
- [Personal Access Tokens in GitHub](#)
- [This talk](#)
- [Exercises](#)