Software Engineering

A Faculty of Engineering Course: CSEN 303

Development & Operations, & Distributed Version Control

5

Dr. Iman Awaad

iman.awaad@giu-uni.de

GERMAN INTERNATIONAL UNIVERSITY
الجامعة الألمانية الدولية

Acknowledgments

The slides are **heavily** based on the **slides** by **Prof. Dr. John Zaki**.

They are also **heavily** based on the slides and textbook by **Ian Somerville**.

The git basics slides are by an anonymous Hochschule Bonn-Rhein-Sieg student who gave the git lecture to our freshmen Master's Program in Autonomous Systems in March 2024 during the Foundation Course.

Their contribution is gratefully acknowledged.

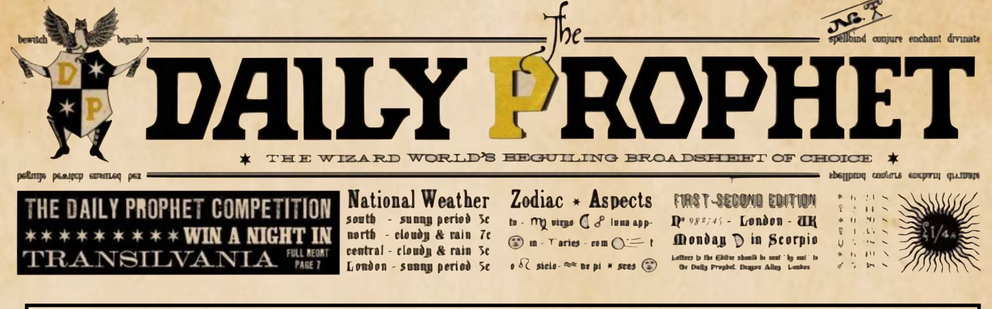
Any additional sources are referenced.

Distributed Version Control

- DevOps automation:
  - continuous integration
  - continuous delivery
  - continuous deployment
  - infrastructure as code
  - DevOps measurement
- Distributed version control
- git

How do we manage code and other files in software projects?

How do we enable safe and effective collaboration?



The Daily Prophet

THE WIZARD WORLD'S BEGUILING BROADSHEET OF CHOICE

THE DAILY PROPHET COMPETITION ★★★★★★★★★ WIN A NIGHT IN TRANSILVANIA FULL REPORT PAGE 7

National Weather
south - sunny period 5c
north - cloudy & rain 7c
central - cloudy & rain 5c
London - sunny period 5c

Zodiac ★ Aspects

FIRST-SECOND EDITION
London - UK
Monday ☽ in Scorpio

b-it-bots №1 AGAIN!

This is the second world championship title for the team after winning for the first time in 2019 in Sydney! The b-it-bots @Work team has done it again - the team has won the RoboCup@Work competition at RoboCup 2023 in Bordeaux, France! The official results of the competition can be found at https://atwork.robocup.org/rc2023/. The following wizards participated in the competition: Kevin (team leader), Gokul, Ravisankar, Shubham, Vamsi, Vivek, and Santosh. They were supported by others who did not attend the competition, including Sathwik, Kishan, Hamsa, Chaitanya, Deebul, and Wasil. In September, the b-it-bots @Home team also repeated their success from the 2019 ERL Smart City competition by winning two episodes and joint overall best team this year in Milton Keynes! Lucy, their favorite robot, was going about her business opening doors, and helping Muggles at home, supported by Minh, Manish, Hamsa, Melvin, Zain, Nada and Vivek. The b-it-bots also took part in a couple of other competitions this year. We caught up with them and Lucy in May at the METRICS HEARTMET competition at ICRA 2023 in London supported by Manish, Melvin, Bharath, Hamsa and Zain, building on their success from the competition at the Cobot Maker Space, University of Nottingham, in October 2022, where they were also supported by Ananya and Ekansh. You could also catch Bharath, Vamsi and Saad winning the METRICS HEARTMET competition in Florence with the Kinova arm, and Kevin, Vamsi, Ravisankar and Deebul placing 6th at the Robothon® - The Grand Challenge 2023 challenge, with support from Ludovico, Kishan and others. Readers may recall that in addition to the @Work wins, the b-it-bots were also world champions in the @Home League back in 2009.

The Daily Prophet, on behalf of all of our readers, offers our sincere congratulations to the world champions!

(Due to concerns raised about their winning streak, the b-it-bots seeks to clarify that they work closely with the Misuse of Muggle Artefacts Office at the Ministry to ensure their robots comply with all regulations.)

Recognition 2 — Ministry Affairs 3. — jobs 4 — misc 6

https://www.h-brs.de/en/a2s/news/new-victory-b-it-bots-third-robocupwork-world-championship-title-won-robocup-2025

https://www.h-brs.de/en/a2s/b-it-botswork

Competition rankings

- **1st** place: RoboCup World Championship 2025, Salvador, Brazil
- 3rd place: RoboCup World Championship 2024, Eindhoven, Netherlands
- **1st** place: RoboCup German Open 2024, Kassel, Germany
- **1st** place: RoboCup World Championship 2023, Bordeaux, France
- **1st** place: Coworker Assembly Test, RoboCup World Championship 2023, Bordeaux, France
- 2nd place: RoboCup World Championship 2021, Online
- 1st place: Episode 7 Shopping Pick and Pack, SciRoc 2019, Milton Keynes, UK
- **1st** place: RoboCup World Championship 2019, Sydney, Australia
- **1st** place: RoboCup German Open 2019, Magdeburg, Germany
- **1st** place: Cluttered Pick Challenge, RoboCup German Open 2019, Magdeburg, Germany
- 2nd place: RoboCup World Championship 2018, Montreal, Canada
- 2nd place: Arbitrary Surface Test, RoboCup World Championship 2018, Montreal, Canada
- **1st** place: Line Following Test, RoboCup World Championship 2018, Montreal, Canada
- 2nd place: RoboCup GermanOpen 2018, Magdeburg, Germany
- **1st** place: Arbitrary Surface Test and Line Following Test, RoboCup GermanOpen 2018, Magdeburg, Germany
- **1st** place: ERL Industrial Robots Season 2017/18 – Functionality Benchmark 2- Manipulation
- 4th place: RoboCup World Championship 2017, Nagoya, Japan
- 2nd place: RoboCup GermanOpen 2017, Magdeburg, Germany
- **1st** place: ERL Industrial Robots Season 2016/17 – Navigation Functionality
- 2nd place: RoboCup World Championship 2016, Leipzig, Germany
- **1st** place: RoCKIn 2015 – Control Functionality Benchmark, Lisbon, Portugal
- 2nd place: RoCKIn 2015 – Object Perception Functionality Benchmark, Lisbon, Portugal
- 2nd place: RoCKIn 2015 – Object Manipulation Functionality Benchmark, Lisbon, Portugal
- 3rd place: RoboCup World Championship 2015, Hefei, China
- 2nd place: RoboCup GermanOpen 2015, Magdeburg, Germany
- **1st** place: RoCKIn 2014, Toulouse, France
- **1st** place: RoCKIn 2014 – Object Perception Functionality Benchmark, Toulouse, France
- 2nd place: RoboCup World Championship 2014, João Pessoa, Brazil
- 3rd place: RoboCup GermanOpen 2014, Magdeburg, Germany
- 2nd place: IROS 2012 conference in Vilamoura, Portugal
- 3rd place: RoboCup World Championship 2012 in Mexico City, Mexico

https://www.h-brs.de/en/a2s/b-it-botshome

G. Chenchani, K. Patel, R. Selvaraju, S. Shinde, V. Kalagaturu, V. Mannava, D. Nair, I. Awaad, M. Wasil, S. Thoduka, S. Schneider, N. Hochgeschwender, and P. G. Plöger, "b-it-bots: Winners of RoboCup@Work 2023," in RoboCup 2023: Robot World Cup XXVI (C. Buche, A. Rossi, M. Simões, and U. Visser, eds.), (Cham), pp. 350–361, Springer Nature Switzerland, 2024.

Competition rankings

Lucy (2018-)

- ERL Smart City 2023, Milton Keynes, UK
  - *Joint best team*
  - ***1st** place*: Through the Door challenge
  - ***1st** place*: Socially Acceptable Item Delivery challenge
- *Joint 2nd place*: ICRA 2023 METRICS HEART-MET Physically Assistive Robot Challenge, London, UK
- METRICS HEART-MET Field Campaign 2022, Nottingham, UK
  - ***1st** place*: Person Detection challenge
  - ***1st** place*: Action Recognition challenge
  - ***1st** place*: Gesture Recognition challenge
- *5th place*: RoboCup World Championship (Domestic Standard Platform League) 2021, Online
- ***1st** place*: Episode 10 Open the Door, SciRoc 2019, Milton Keynes, UK
- *Best team in the Domestic Standard Platform League and 4th place overall*: RoboCup German Open 2019, Magdeburg, Germany
- *7th place*: RoboCup German Open 2018, Magdeburg, Germany

Jenny (2011-2016)

- *9th place:* RoboCup World Championship 2016, Leipzig, Germany
- *2nd place:* RoCKIn 2014 – Speech Understanding Functionality Benchmark, Toulouse, France
- *2nd place:* GermanOpen 2012 in Magdeburg, Germany
- *3rd place:* World Championship 2011 in Istanbul, Turkey
- *3rd place:* GermanOpen 2011 in Magdeburg, Germany

Johnny (2008 – 2010)

- *3rd place:* World Championship 2010 in Singapore
- ***1st** place:* GermanOpen 2010 in Magdeburg, Germany
- ***1st** place:* World Championship 2009 in Graz, Austria
- ***1st** place:* GermanOpen 2009 in Hannover, Germany
- *2nd place:* World Championship 2008 in Suzhou, China
- *2nd place:* GermanOpen 2008 in Hannover, Germany

What is a **development and operations** (DevOps)?

… is a modern software development approach that combines **development** (Dev) and **operations** (Ops) to **streamline collaboration**, **accelerate delivery**, and **improve software quality**.

What is a **distributed version control system** (DVCS)?

…is a system used for **tracking changes** in source code **during software development**. It enables **multiple developers** to **collaborate** on the same project efficiently and **manage different versions** of the codebase.

What is a **distributed version control system** (DVCS)?

…is a system used for **tracking changes** in source code **during software development**. It enables **multiple developers** to **collaborate** on the same project efficiently and **manage different versions** of the codebase.

… let's back up a little

Traditionally, three teams

software development	**software release**	**software support**
gave a '**final**' **version** of the software to a release team	built a **release version**, tested this and prepared release documentation before releasing the software to customers	provided **customer support**!

The original development team
were sometimes also responsible for implementing software changes.

Alternatively, the software may have been maintained by a separate '**maintenance team**'.

Software support

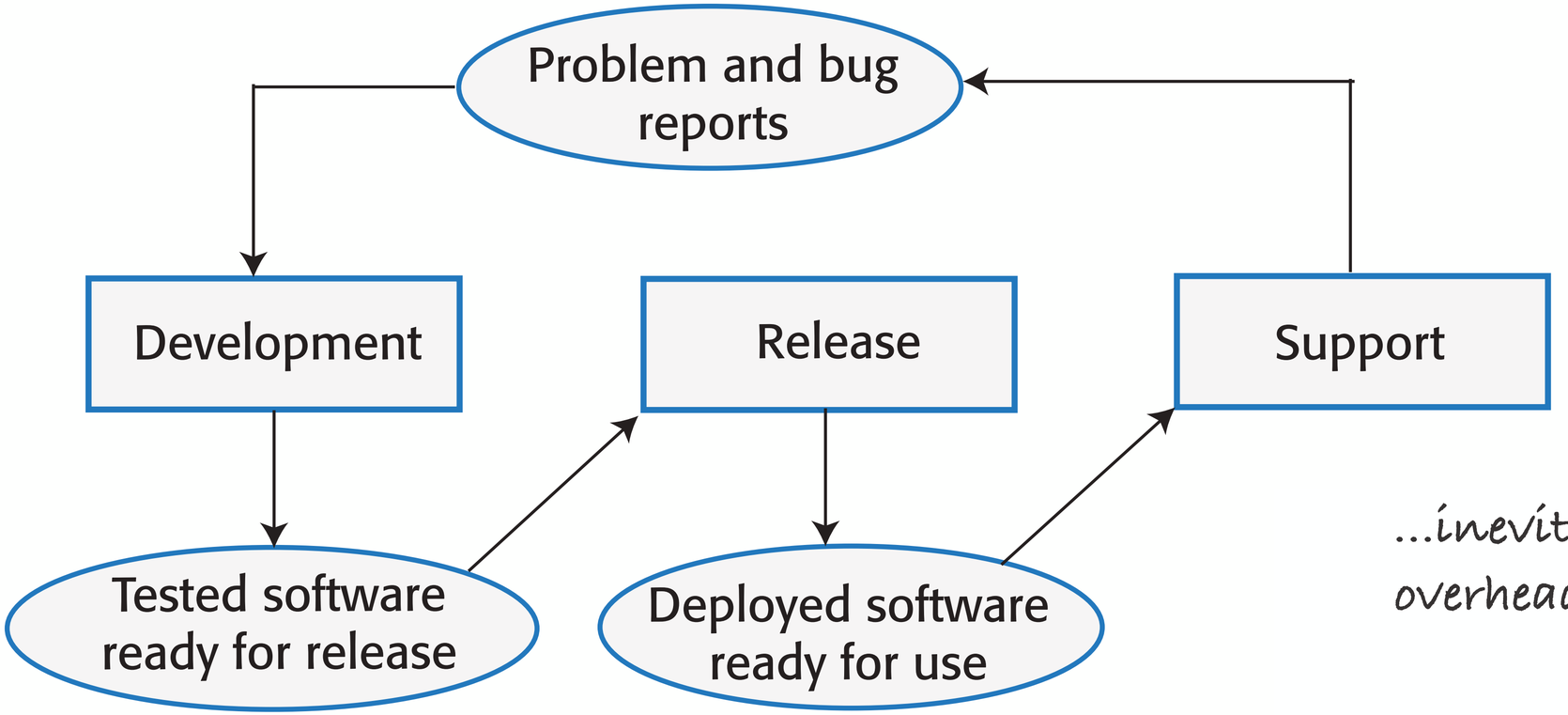


Figure 10.1 Development, release and support

ware support

provided
omer support!

...inevitable delays and overheads in this model

Can we do better?

Alternatively, the software may have been maintained by a

DevOps (Development+Operations)

Factors leading to the spread of DevOps

Agile software engineering: reduced development time, but traditional release process introduced a **bottleneck between development** and **deployment**

Amazon re-engineered their software around **services** and introduced an approach in which a service was developed and supported by the same team!

It became possible to release **software as a service**, running on a public or private **cloud**.

Software products were no longer released to users on physical media or downloads!

DevOps…

Everyone is responsible for everything
All team members develop, deliver and support the software

Everything that can be automated should be automated
All activities involved in testing, deployment and support should be automated where possible. Minimal manual involvement in deployment!

Measure first, change later
DevOps should be driven by a measurement program: collect data about system and its operation. Use the data to inform decisions about changing DevOps processes and tools

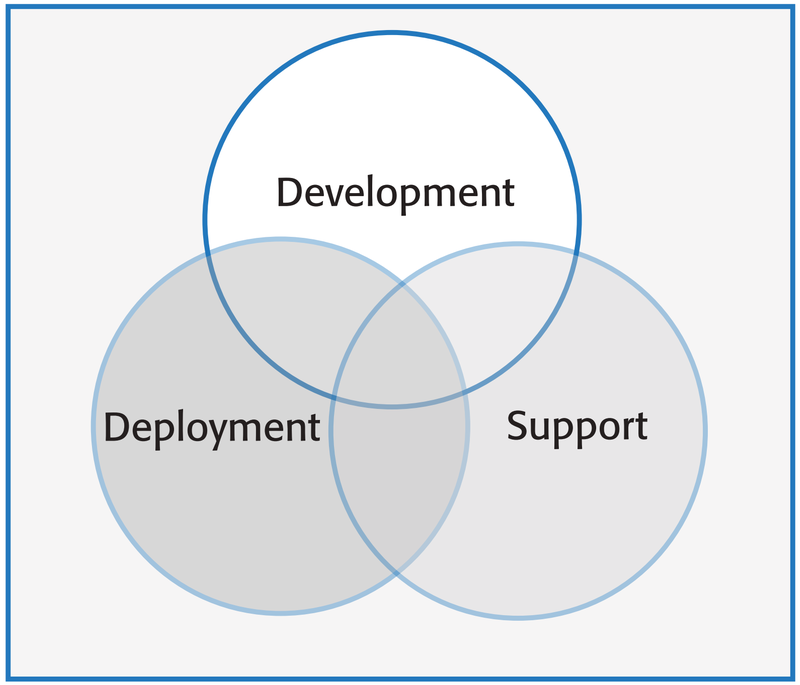
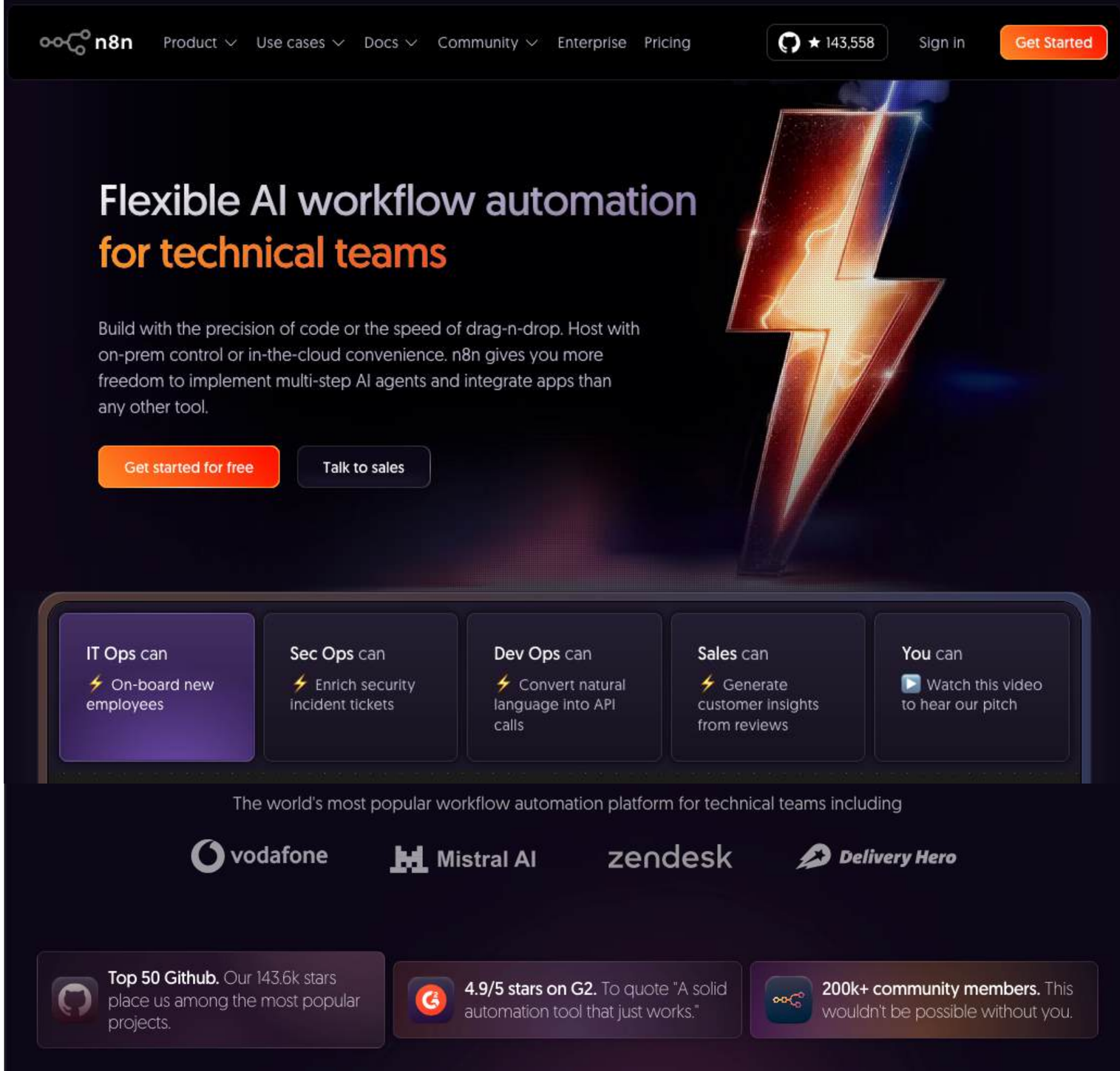


Figure 10.2 DevOps

In the age of AI

https://n8n.io



Benefits

Faster deployment
…because communication delays between people involved are dramatically reduced

Faster repair
…DevOps teams work together to get the software fixed ASAP! No need to find out which team was responsible for the problem and to wait for them to fix it…

Reduced risk
…increment of functionality in each release is small: less chance of feature interactions and other changes causing system failures or outages

More productive teams
…DevOps teams are happier and more productive than teams involved in the separate activities…less likely to leave to find jobs elsewhere

Code Management

…is a set of **software-supported practices** that is used to **manage** an evolving **codebase**.

ensures that changes made by different developers do not interfere

…makes it easy to create a product from its source code files and to run automated tests on that product

enables the creation of different product versions

Code Management and DevOps

…is a set of **software-supported practices** that is used to **manage** an evolving **codebase**.

We need:

1. Source code management system (for ‘automating everything’; stores more than just code; the automation and measurement tools all interact with the code management system)

2. Automated system building process

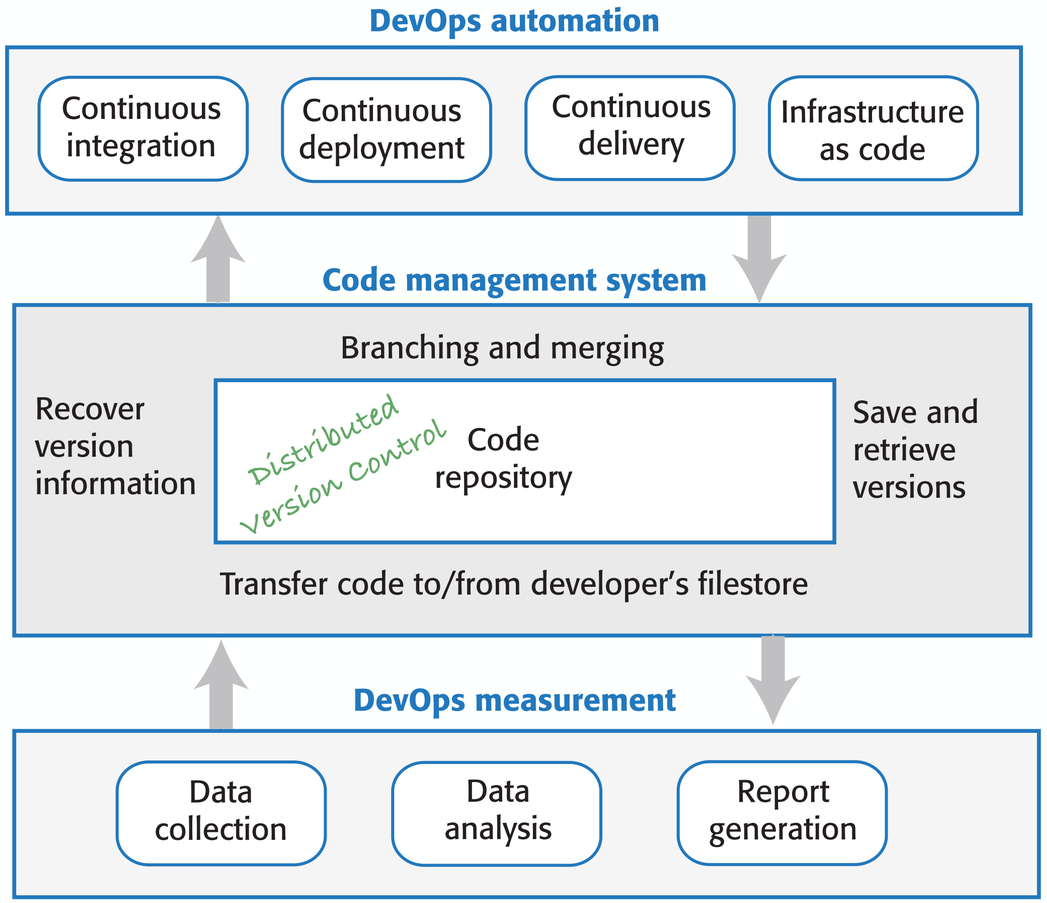


Figure 10.3 Code management and DevOps

Code Management Fundamentals

Code transfer Developers take code into their **personal file store** to work on it then return it to the shared code management system

Version storage and retrieval Files may be stored in **several different versions** and specific versions of these files can be retrieved

Merging and branching Parallel development branches may be created: changes made by developers in different branches may be merged

Version information Information about the different versions maintained in the system may be stored and retrieved

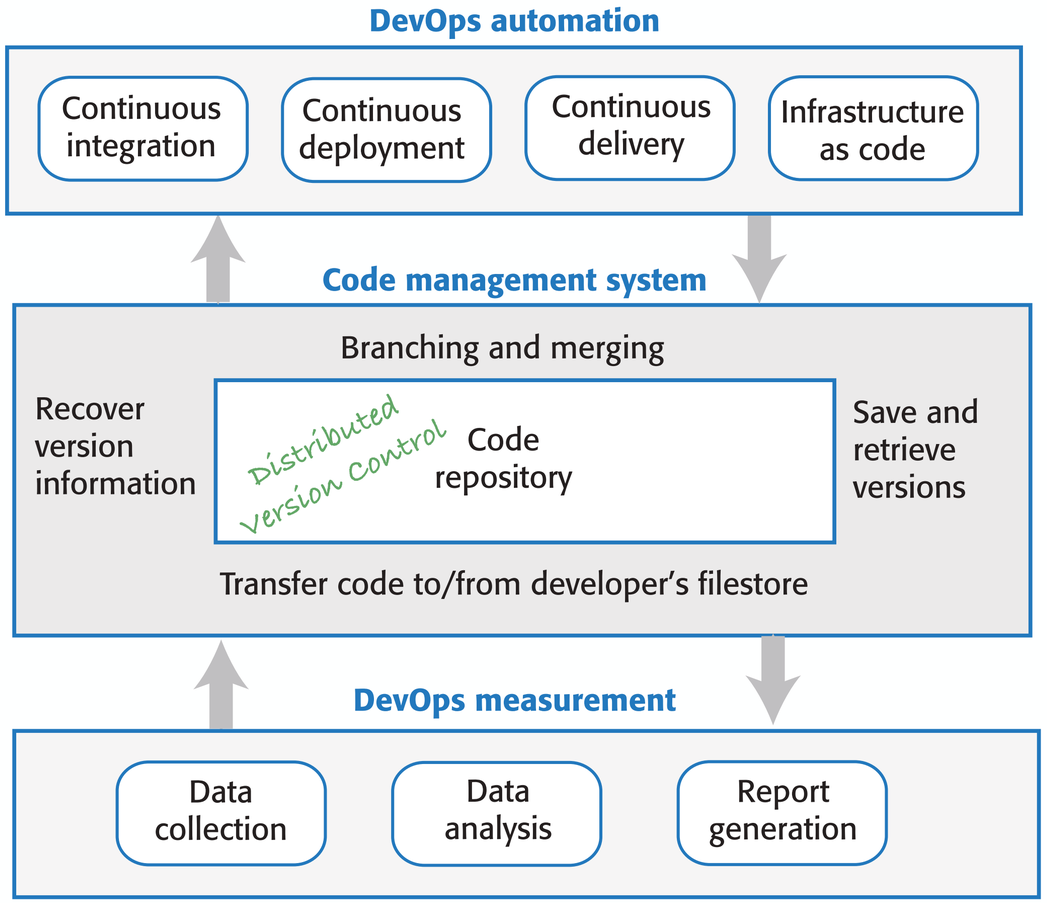


Figure 10.3 Code management and DevOps

Code Repository

- All **source code files** and file **versions**
- Other artefacts such as **configuration files**, **build scripts**, **shared libraries** and **versions of tools** used
- …includes a database of information on the stored files, *e.g.* version info, who has changed the files, what changes were made and when,…
- when files are transferred in/out, info about the different versions of files and their relationships may be updated
- Specific versions of files and info about these versions can always be retrieved from the repository

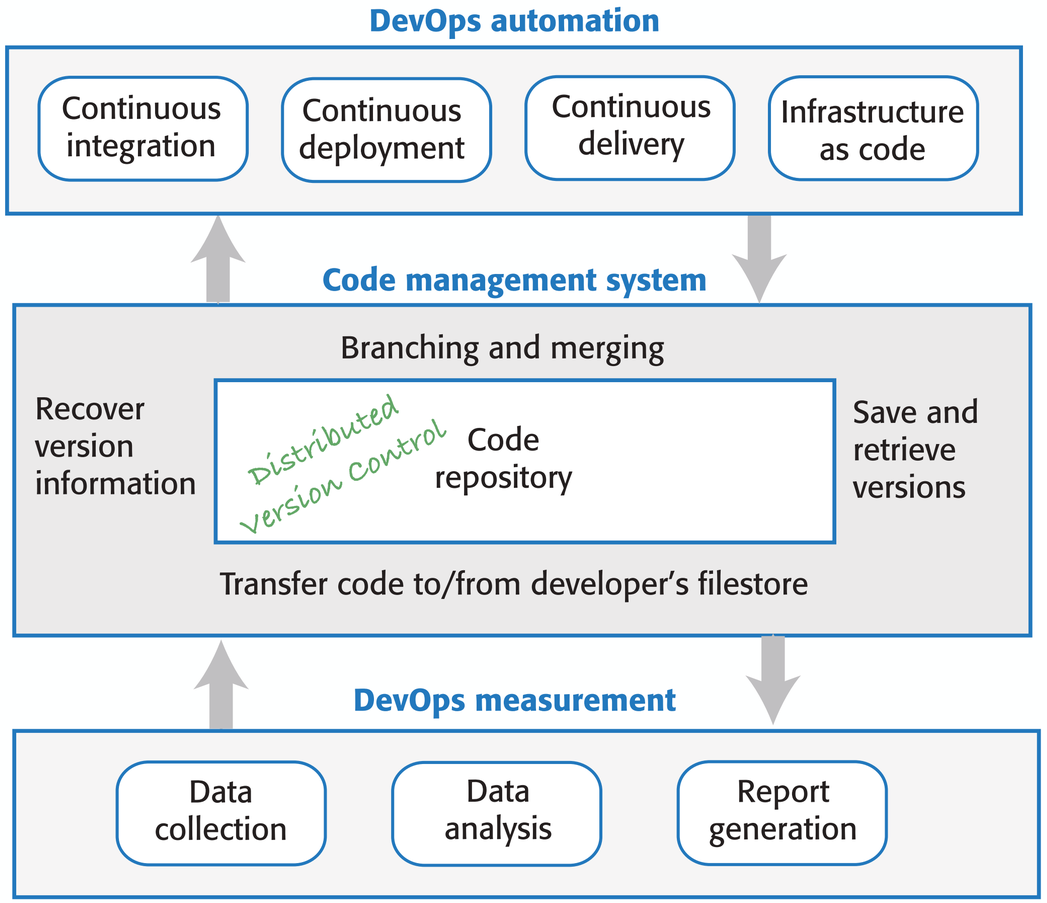


Figure 10.3 Code management and DevOps

Features of Code Management Systems

Version and release identification … Managed versions of a code file are uniquely identified when they are submitted to the system and can be retrieved using their identifier and other file attributes.

Change history recording … The reasons why changes to a code file have been made are recorded and maintained.

Independent development … Several developers can work on the same code file at the same time. When this is submitted to the code management system, a new version is created so that files are never overwritten by later changes.

Project support …All of the files associated with a project may be checked out at the same time.There is no need to check out files one at a time.

Storage management … The code management system includes efficient storage mechanisms so that it doesn't keep multiple copies of files that have only small differences.

What is a **git**?

…is a system used for **tracking changes** in source code **during software development**. It enables **multiple developers** to **collaborate** on the same project efficiently and **manage different versions** of the codebase.

What is a **git**?

…is a system used for **tracking changes** in source code **during software development**. It enables **multiple developers** to **collaborate** on the same project efficiently and **manage different versions** of the codebase.

Instead of only keeping the copies of the files that users are working on, Git maintains a clone of the repository on every user's computer

Benefits of Distributed Code Management

Resilience

If the shared repository is damaged or subjected to a cyberattack, work can continue, and the clones can be used to restore the shared repository

People can work offline if they don’t have a network connection

Speed

Committing changes to the repository is a fast, local operation and does not need data to be transferred over the network

Flexibility

Local experimentation is much simpler. Developers can safely experiment and try different approaches without exposing these to other project members.

git

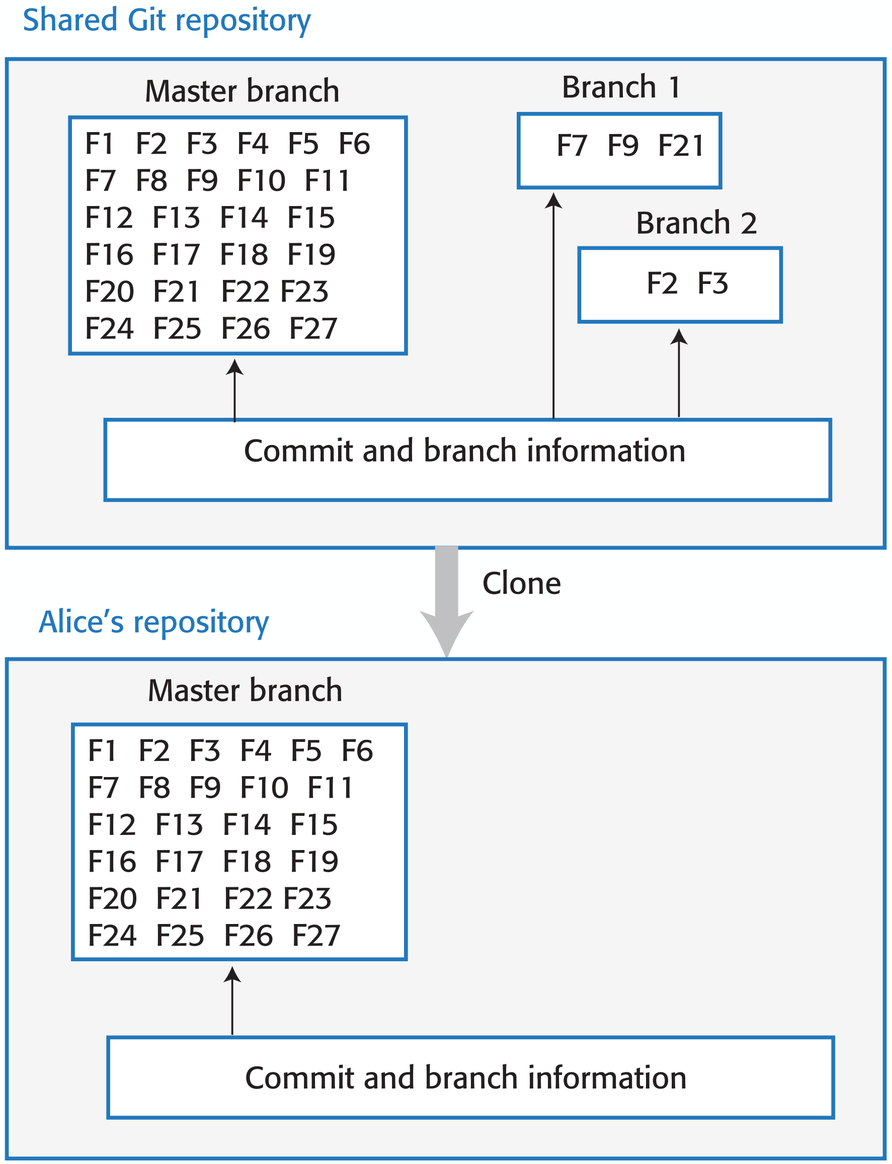


Figure 10.5 Repository cloning in Git

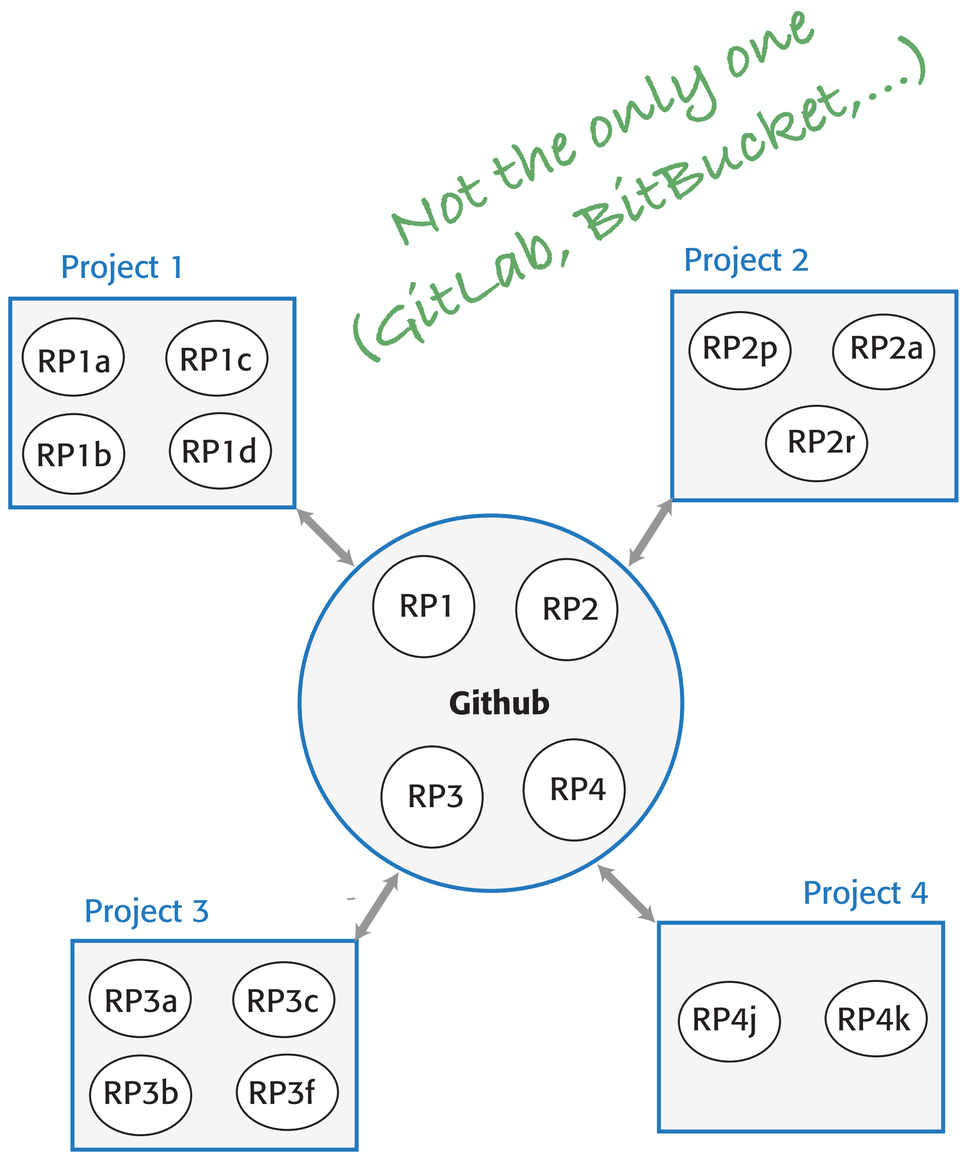


Figure 10.6 Git repositories

git

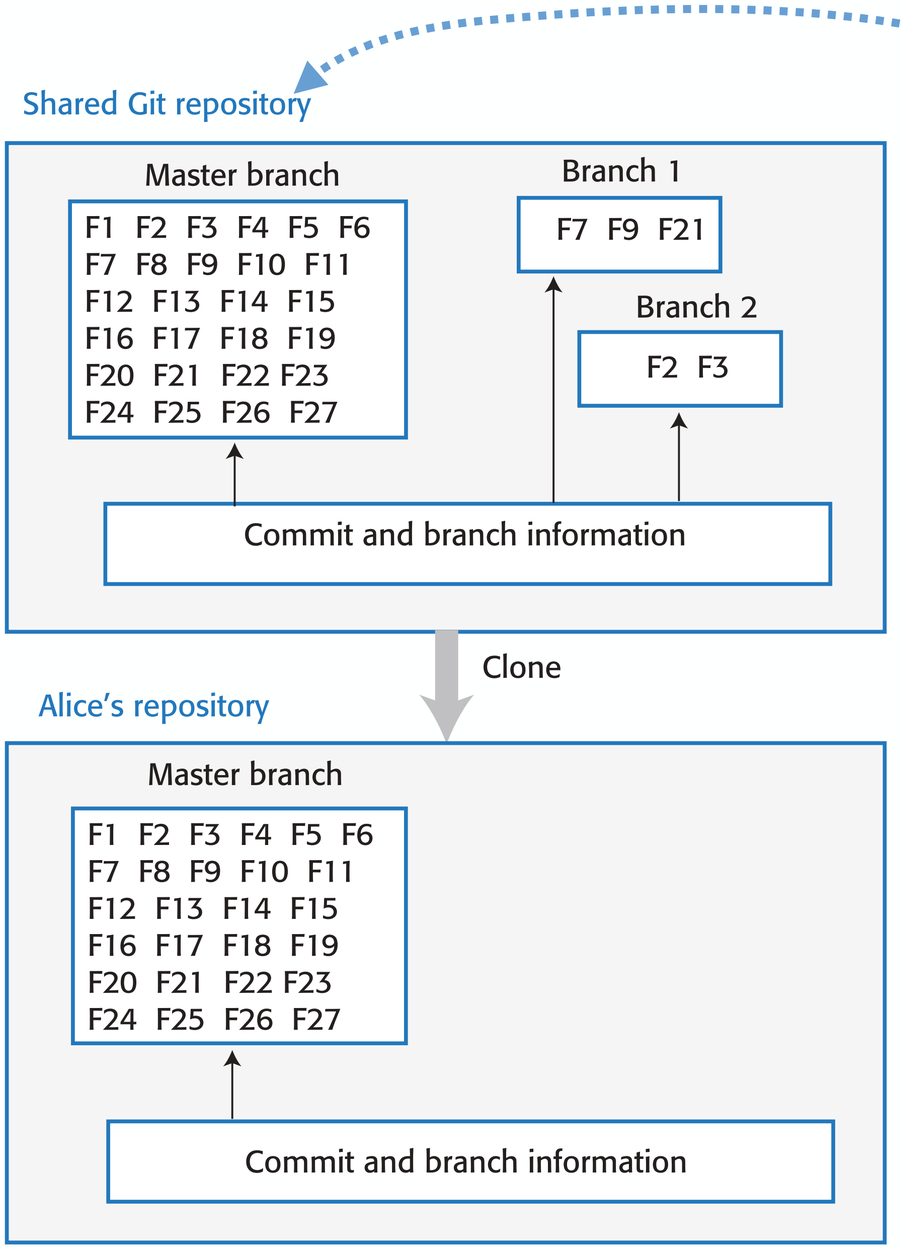


Figure 10.5 Repository cloning in Git

…a special "folder" for your project that Git uses to track and record changes. It contains all of your project files and the history of changes.

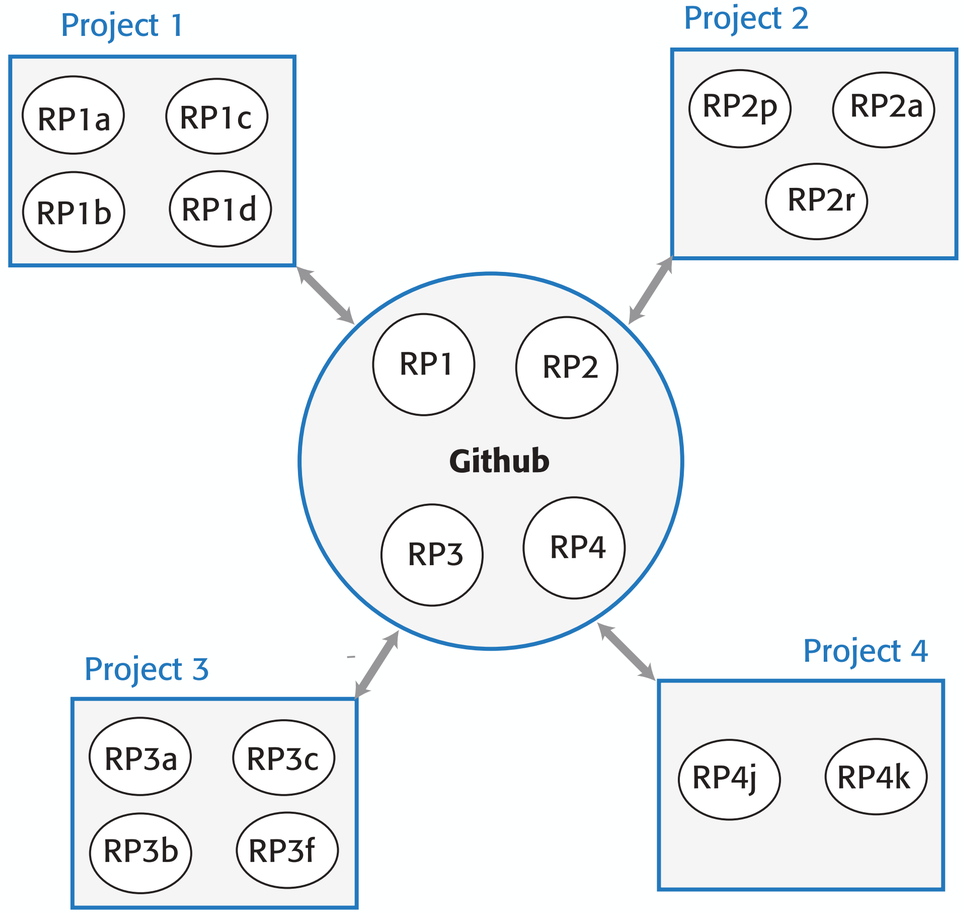


Figure 10.6 Git repositories

Where to start?
Account creation…

Create an account on GitHub (use your personal email address)

Setup two-factor authentication

Add an SSH key

To collaborate on GitHub (cloning/pulling/pushing code), you can use two methods:

1. HTTPS (using your username and password)
2. SSH (using a pair of cryptographic keys)

It is recommended to use SSH due to security reasons and ease of use.

Where to start?
Setting things up

Clone a repository (create a complete copy of someone else's Git repository on your computer using the **git clone** command)

$ git clone git@github.com:HBRS-AMR/Robile.git

Initialize a repository (create a new Git repository on your system)

$ git init . 

Basic flow: First command adds all files/directories to staging. Second command commits staged content as a new commit snapshot:

$ git add [files|directories]

$ git commit -m "..."

Where to start?
Staging versus commit snapshot

Staging area is a “prep” room where you get everything ready for the final presentation. That’s where you decide which files you want to include in your next commit (snapshot).

Ready, set, go! Committing is like taking a photo of your repository; all the changes in the staging area are recorded into your project’s history.

Where to start?
Branching and merging

```
$ git checkout -b [branch-name]

…

$ git checkout -b main

$ git merge [branch-name] main
```

Imagine writing a story in a notebook. Branching is like exploring different storylines without messing up with the original plot.

Merging is deciding that the storyline belongs to the main plot.

Branching and merging

A **branch** is an independent, stand-alone version that is created when a developer wishes to change a file.

The changes made by developers in their own branches may be **merged** to create a new shared branch.

The repository ensures that branch files that have been changed cannot overwrite repository files without a merge operation.

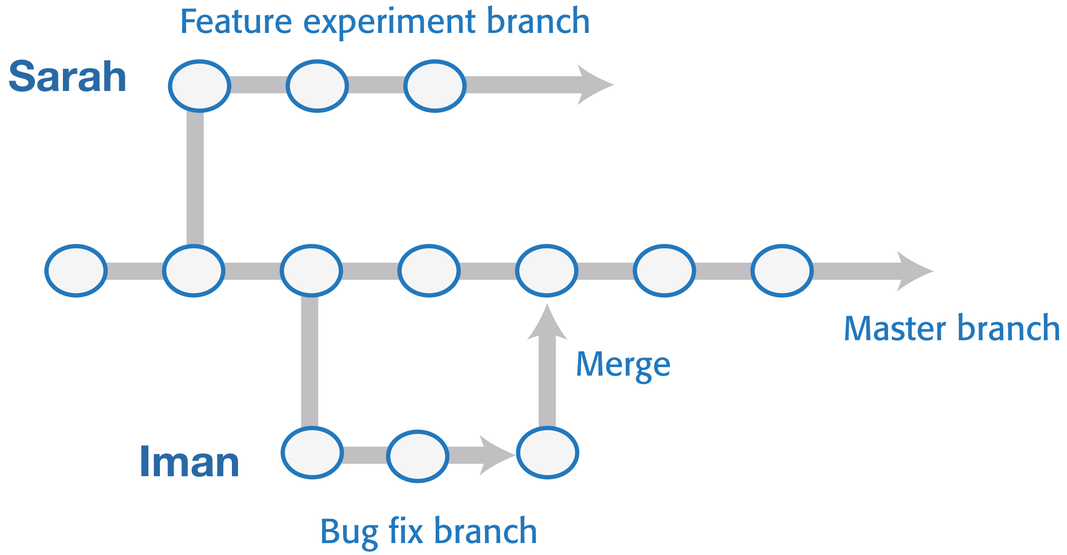


Figure 10.7 Branching and merging

Branching and merging

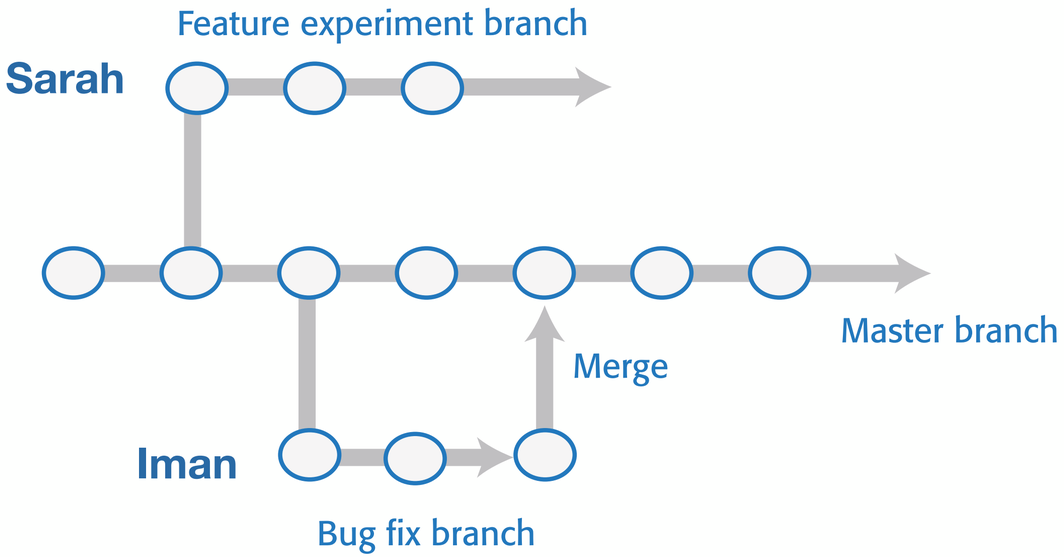


Figure 10.7 Branching and merging

e.g. If Iman or Sarah make mistakes on the branch they are working on, they can easily revert to the master file:

If they **commit** changes, while working, they can **revert** to earlier versions of the work they have done.

When they have finished and tested their code, they can then **replace the master file by merging** the work they have done with the master branch

Where to start?
Stashing

$ git stash

Stashing is like putting your work aside for a moment, without committing anything, or wishing to lose anything!

Very useful when jumping between branches on a project and always being interrupted in the middle of your work by a noisy project manager.

Where to start?
Pushing and pulling

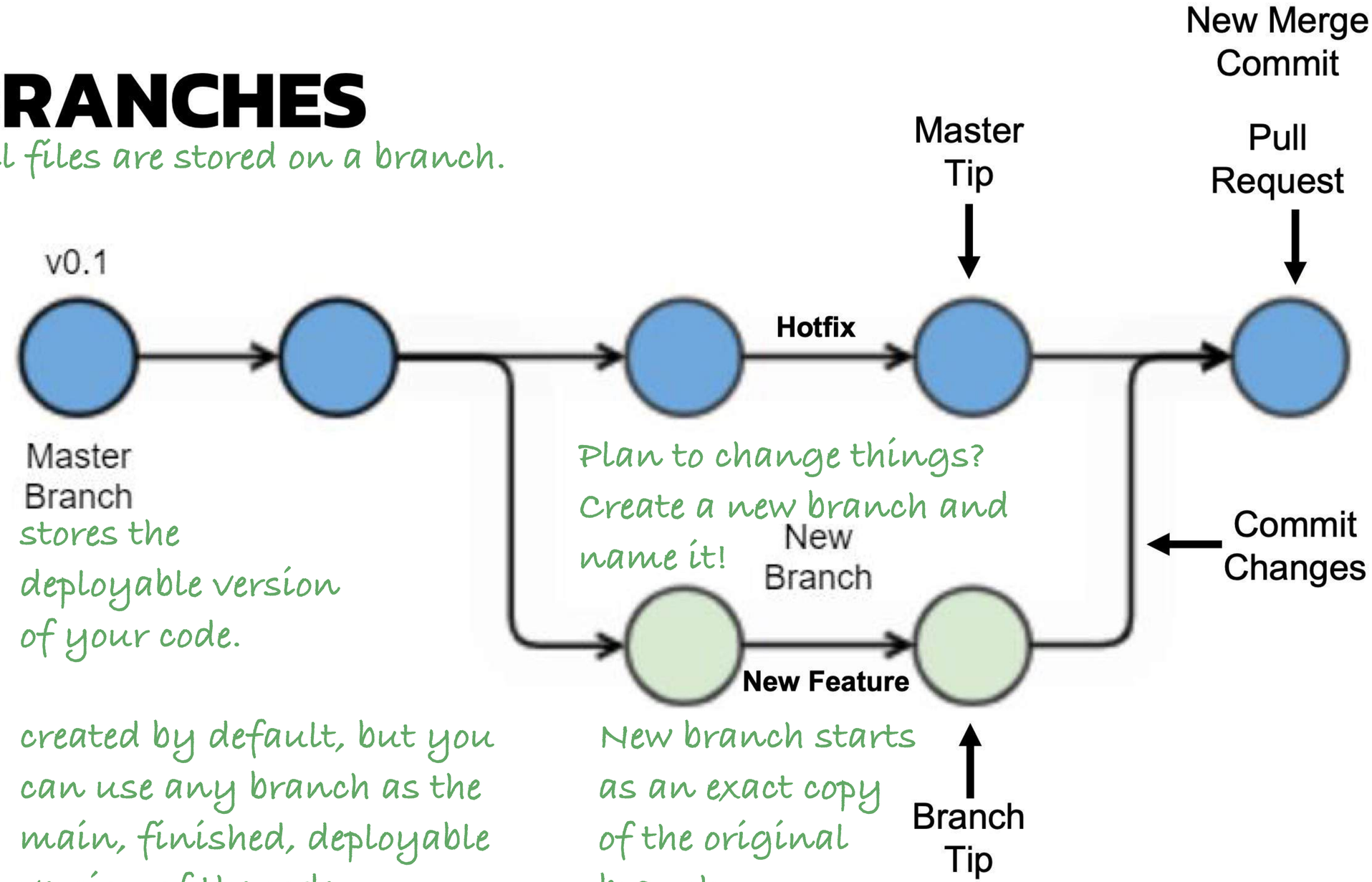
```
$ git push origin [branch-name]
```

```
$ git pull origin [branch-name]
```

Pushing code is sending updates to GitHub. Pulling is retrieving it.

BRANCHES

All files are stored on a branch.



stores the deployable version of your code.

created by default, but you can use any branch as the main, finished, deployable version of the code

New branch starts as an exact copy of the original branch

After merging, it should be deleted

git commands summary

HEAD -> location of where you are right now. (git show HEAD)
Remote -> A shared repository that allows many collaborators

PREREQUISITES:

sudo apt-get install git -> Installs git
A GitHub account

COMMANDS:

git --version
git config --global user.name "user_name"
git config --global user.email "email_id"

git init -> Initialise a folder as a Git repository
git status -> See current status of the repo
git add f_name -> Adds the f_name to the stage
git reset HEAD filename -> removes the file from the staging area to the commit at HEAD. or can use only f_name
git log-> list of prevoious commits
git commit -m "message here" -> From stage to repo.
git diff f_name -> Checks the difference between last commit and present changes(if any) | git diff --staged f_name -> to see difference of staged file

git checkout commit-id/HEAD filename -> Reverts all changes of the filename back to where the commit was. You can use HEAD.
git checkout b_name -> goes to branch b_name
git branch / b_name -> Which branch I am on? / creates a new branch b_name
git branch -d b_name -> Deletes the branch b_name
git checkout -- target_file_name -> reverts file to last commit

git clone remote_location clone_name -> remote_location- place on the web(filename, web address)
clone_name- The name of the repo to clone into (local copy)
git remote add remote_name remote_location -> Adds a remote as remote_name from remote_location
git remote -v -> Shows the remotes

git fetch -> Checks if any changes in the remote and brings those changes on a remote branch(origin/master)
git push origin b_name -> Pushes branch to the remote | if you use -u, command line will remember the parameters so next time you can use "git push"
git pull -> pulls changes from the remote repository

git tag tag_name -> creates a tag tag_name

Project: My recommendation…

- Create your GitHub accounts
- One person in each group should create a new GitHub repository
- Invite your team members as collaborators
- Let everyone clone the repository
- Everyone should checkout a new branch with their name
- Create a file, write something nice about yourself ;)
- Push it
- Create a pull request
- Merge all pull requests

Good to know…

Using a graphical user interface (GUI):

- …can be beneficial (easier to visualise all the branches, commits, their history,…)
- You can install it as a separate piece of software (for example GitKraken), or have it as part of your Integrated Development Environment (IDE).

Meaning of .gitignore

- Certain files are not supposed to be tracked. *e.g.*
  - Files containing confidential data such as secrets (.env, …)
  - Folders with dependencies or virtual environment configurations (node_modules, venv, …)
  - Temporary files
- These files/folders can be listed in .gitignore

INTRODUCTION

Developers **rarely** work **alone**
Development include many people, backend developers, frontend developers, DB administrators, …etc

Every change by every contributor **must be tracked** and controlled to enable collaboration, accountability and version management.

DISTRIBUTED VERSION CONTROL SYSTEMS

- **Critical tools in software development,** and key enablers for **social and collaborative coding.**
- Used by Software Engineers, DevOps**,** Data Scientists, Data Engineers **and** many other technology practitioners
- They are used not only by developers but other professionals for developing course contents, legal case management, use cases, books and even recipes.
- A version control system allows you to keep track of changes to your documents.
- Easy to recover older versions of your document if you make a mistake
- Makes collaboration with others much easier.

Git is a distributed version control system, which means that users anywhere in the world can have a copy of your project on their own computer. When they've made changes, they can sync their version to a remote server to share it with you.

- Many famous systems! Git is amongst the most popular and **GitHub is a highly popular Git-based hosted version control platform.**
- Use it from the **web interface** or from the **CLI**
- **Mostly required when you try to get a SE job, employers expect to see your GitHub!**

SOME TERMS

SSH PROTOCOL: is a method for secure remote login from one computer to another.

A REPOSITORY: contains your project folders that are set up for version control (repo).

A WORKING DIRECTORY contains the files and subdirectories on your computer that are associated with a Git repository (often called repo).

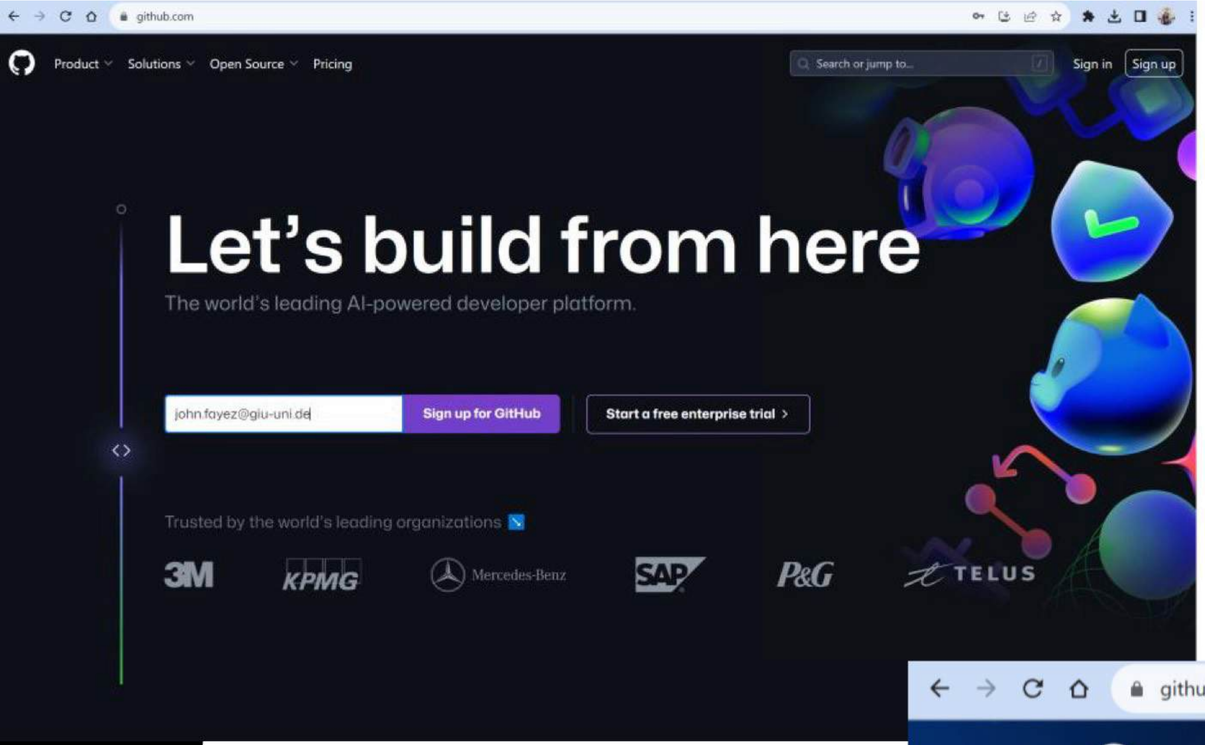
A CLONE is a copy of the repository that is in sync with GitHub repository

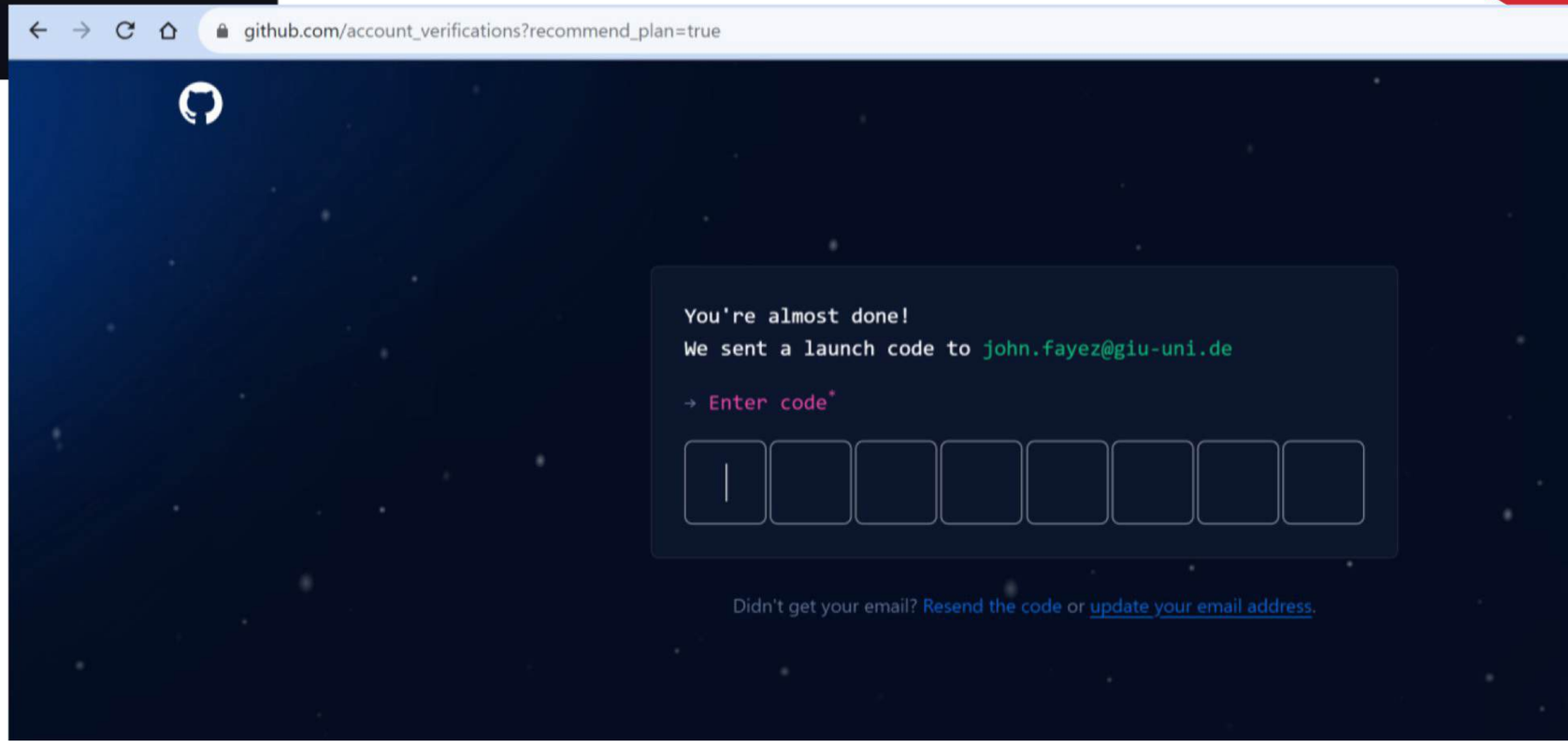
A FORK is a copy of a repository that forms the base for a new project

A PULL REQUEST is the way you request that someone reviews and approves your changes before they become final.

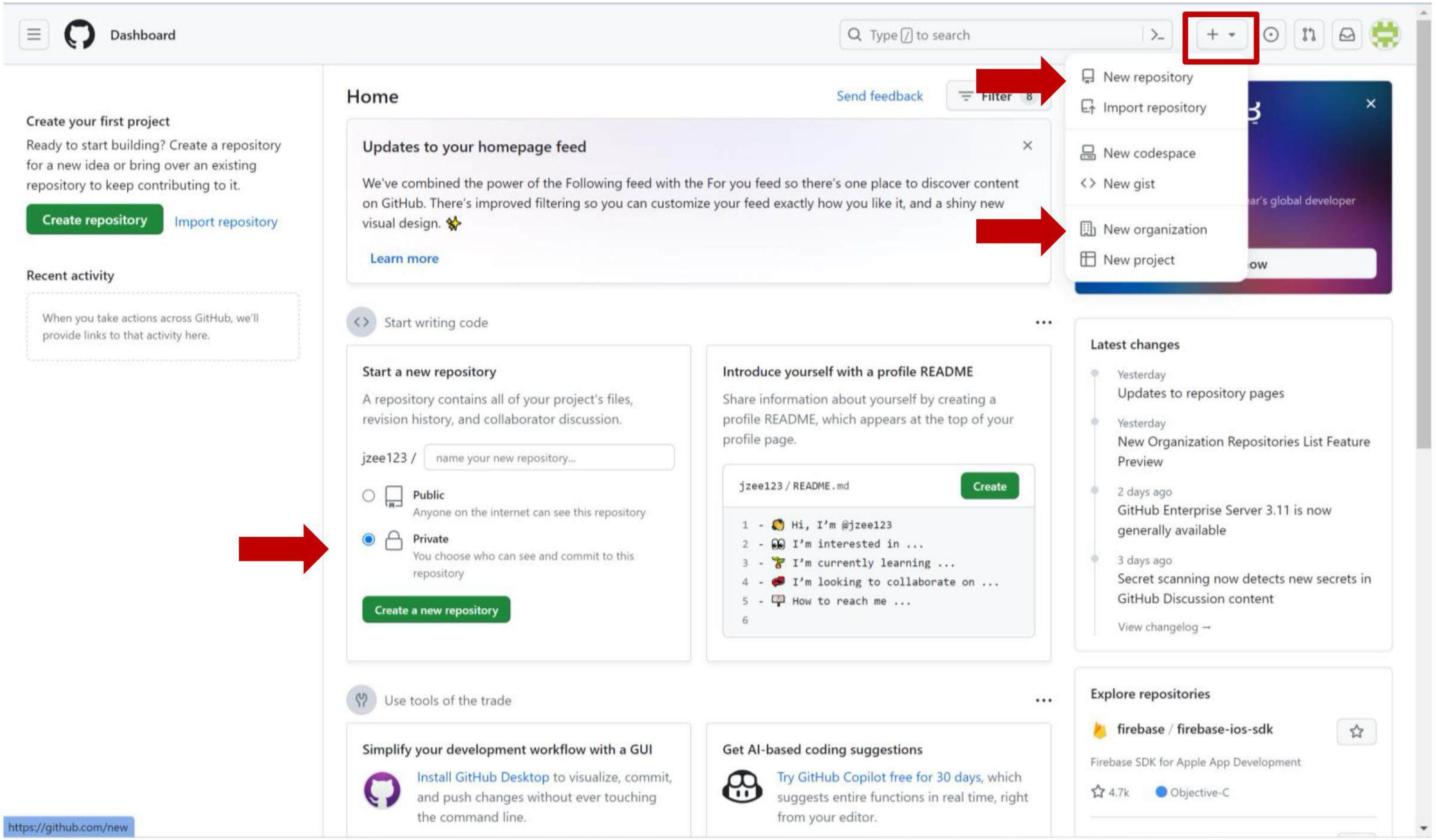
When starting out with a new repository, you only need create it once: either locally, and then push to GitHub, or by cloning an existing repository from GitHub.

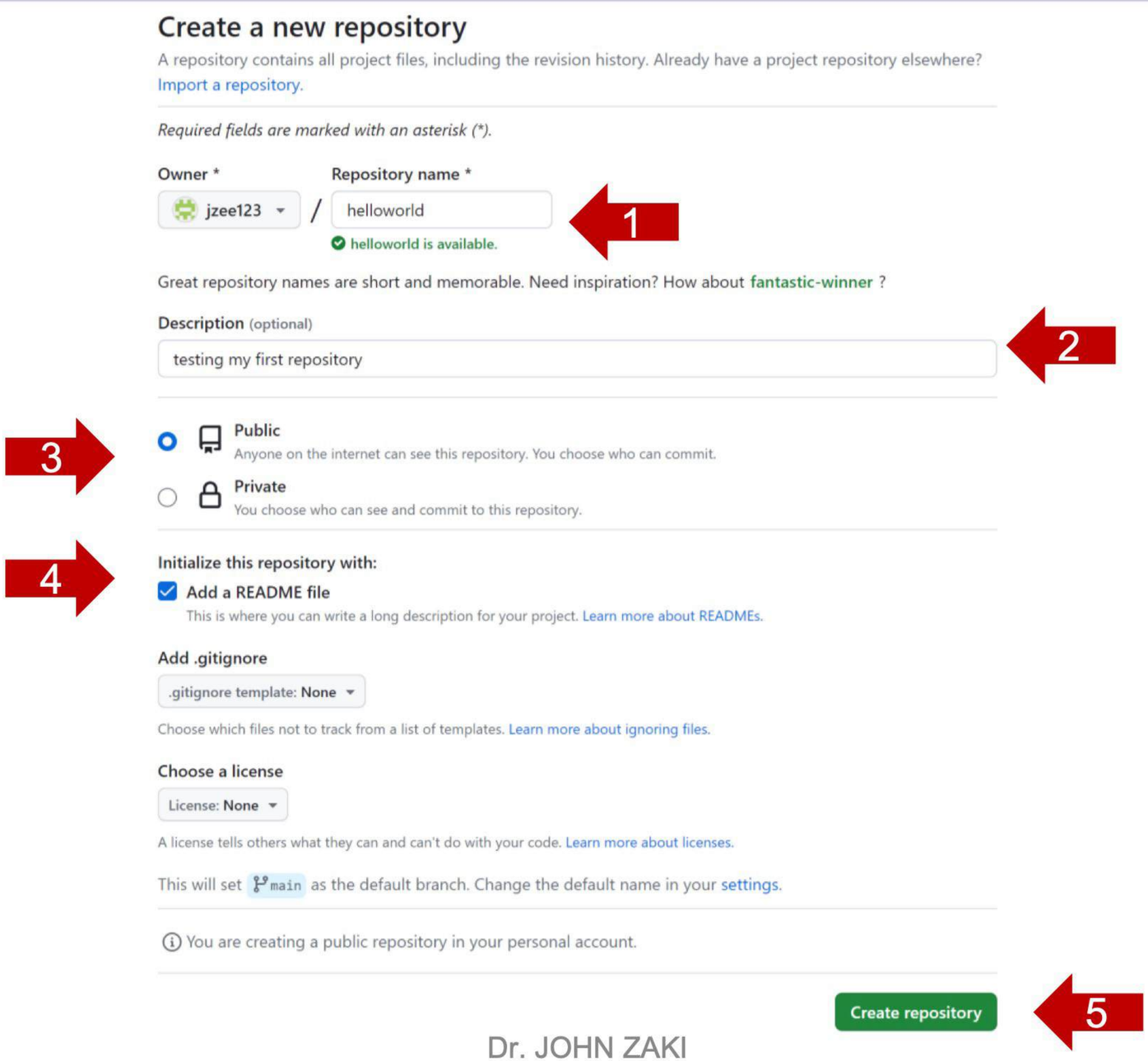
CREATE ACCOUNT, VERIFY YOUR EMAIL

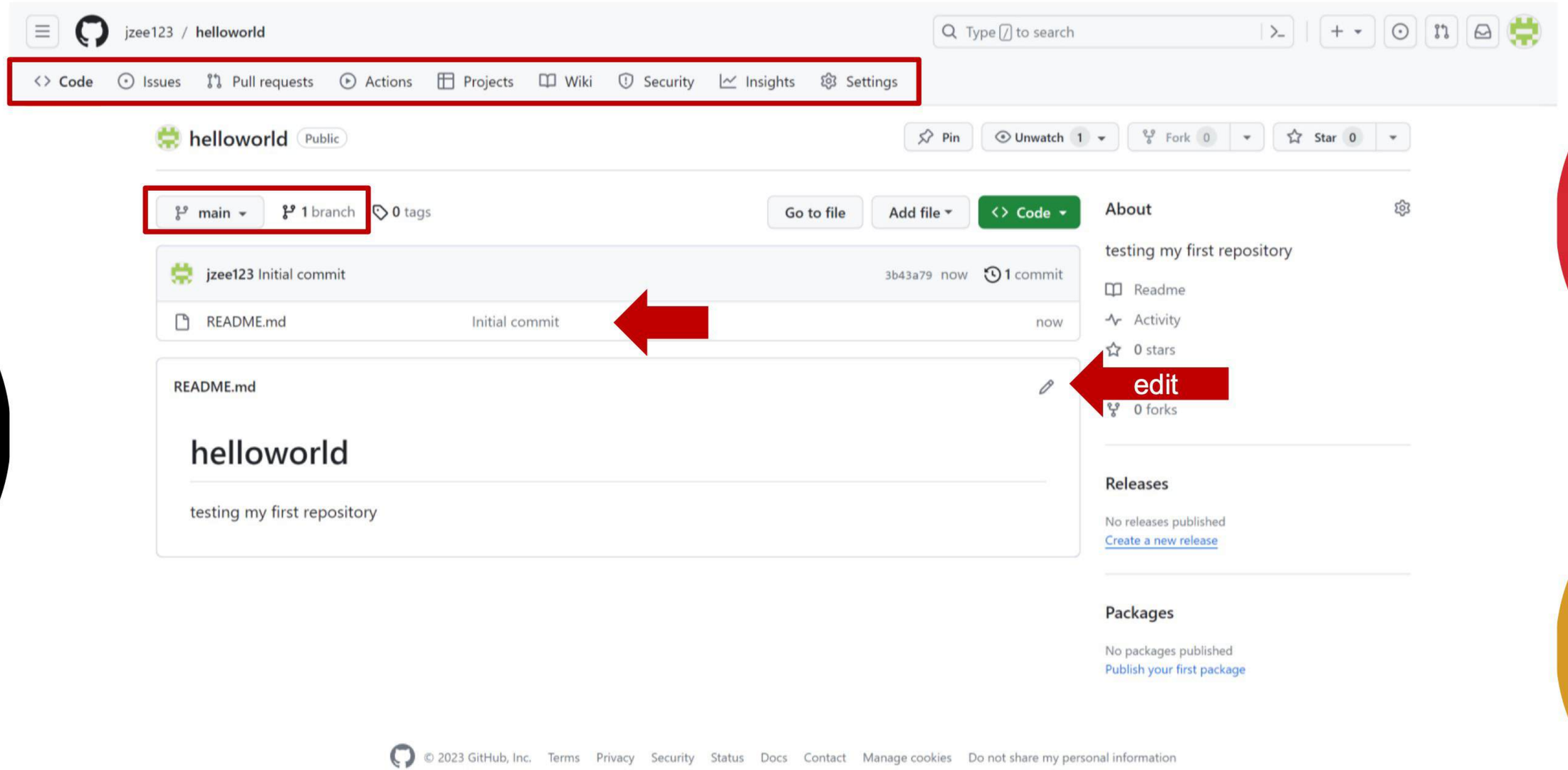


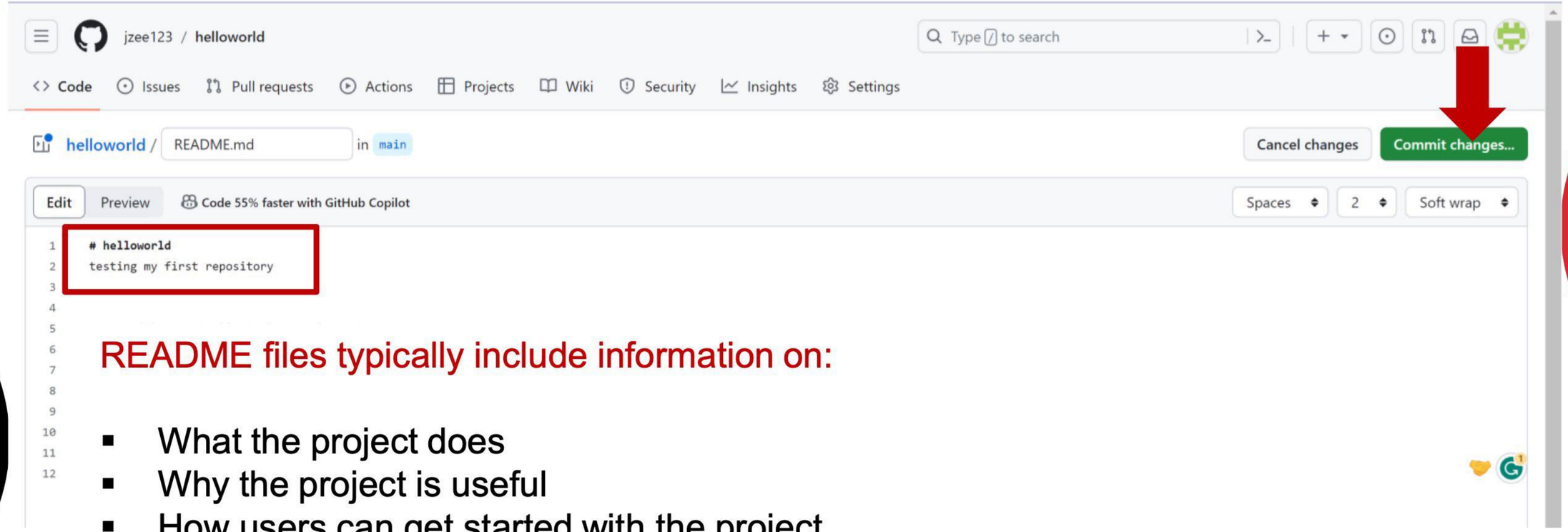


GIU GERMAN INTERNATIONAL UNIVERSITY


Dashboard
Type / to search
Create your first project
Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.
Create repository
Import repository
Recent activity
When you take actions across GitHub, we'll provide links to that activity here.
Home
Send feedback
Filter
Updates to your homepage feed
We've combined the power of the Following feed with the For you feed so there's one place to discover content on GitHub. There's improved filtering so you can customize your feed exactly how you like it, and a shiny new visual design.
Learn more
New repository
Import repository
New codespace
New gist
New organization
New project
Start writing code
Start a new repository
A repository contains all of your project's files, revision history, and collaborator discussion.
jzee123 /
name your new repository...
Public
Anyone on the internet can see this repository
Private
You choose who can see and commit to this repository
Create a new repository
Introduce yourself with a profile README
Share information about yourself by creating a profile README, which appears at the top of your profile page.
jzee123 / README.md
Create
1 - Hi, I'm @jzee123
2 - I'm interested in ...
3 - I'm currently learning ...
4 - I'm looking to collaborate on ...
5 - How to reach me ...
6
Latest changes
Yesterday
Updates to repository pages
Yesterday
New Organization Repositories List Feature Preview
2 days ago
GitHub Enterprise Server 3.11 is now generally available
3 days ago
Secret scanning now detects new secrets in GitHub Discussion content
View changelog →
Use tools of the trade
Simplify your development workflow with a GUI
Install GitHub Desktop to visualize, commit, and push changes without ever touching the command line.
Get AI-based coding suggestions
Try GitHub Copilot free for 30 days, which suggests entire functions in real time, right from your editor.
Explore repositories
firebase / firebase-ios-sdk
Firebase SDK for Apple App Development
4.7k
Objective-C
https://github.com/new


Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Required fields are marked with an asterisk (*).
Owner *
Repository name *
jzee123
/
helloworld
helloworld is available.
1
Great repository names are short and memorable. Need inspiration? How about fantastic-winner ?
Description (optional)
testing my first repository
2
3
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.
4
Initialize this repository with:
Add a README file
This is where you can write a long description for your project. Learn more about READMEs.
Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.
Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.
This will set main as the default branch. Change the default name in your settings.
You are creating a public repository in your personal account.
Create repository
5

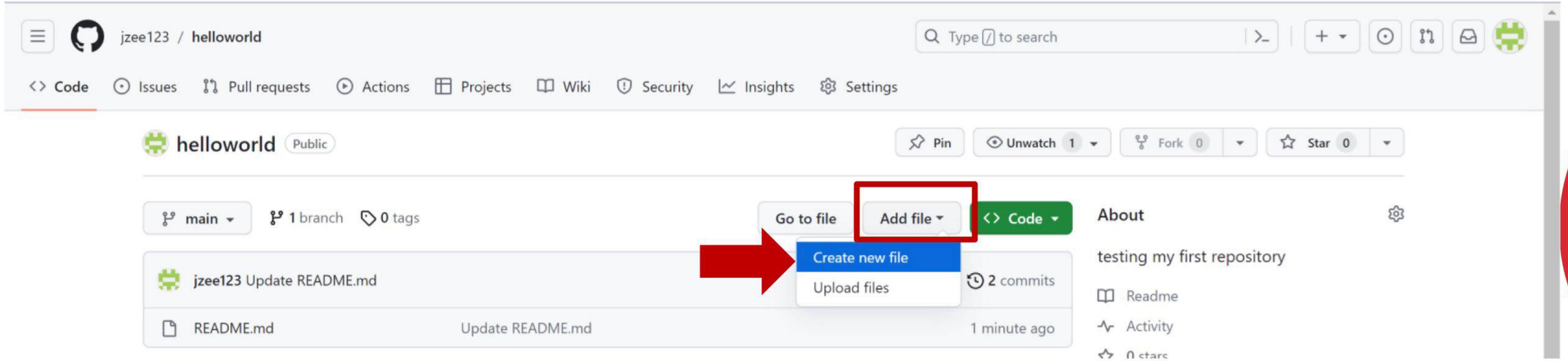

jzee123 / helloworld
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
helloworld
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 branch
0 tags
Go to file
Add file
Code
About
testing my first repository
jzee123 Initial commit
3b43a79 now
1 commit
README.md
Initial commit
now
Readme
Activity
0 stars
edit
README.md
0 forks
helloworld
testing my first repository
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2023 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact
Manage cookies
Do not share my personal information

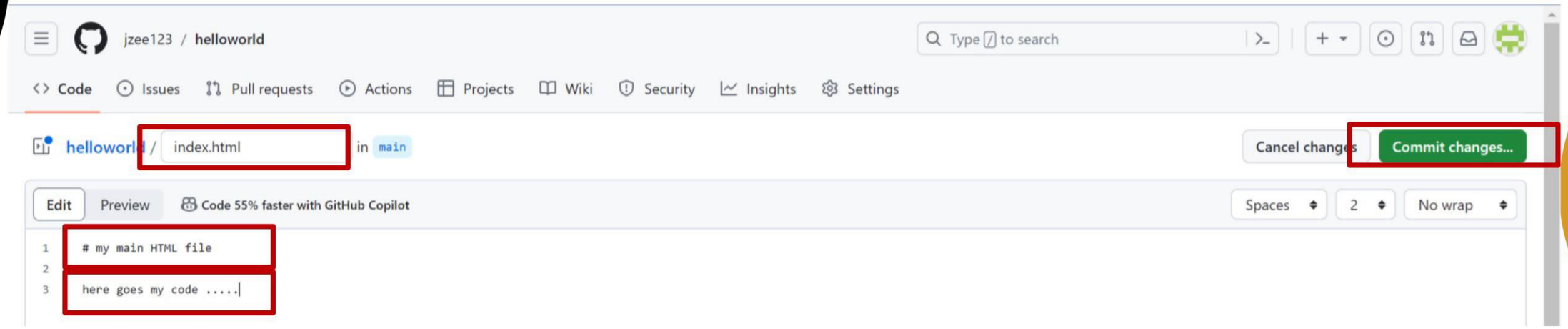


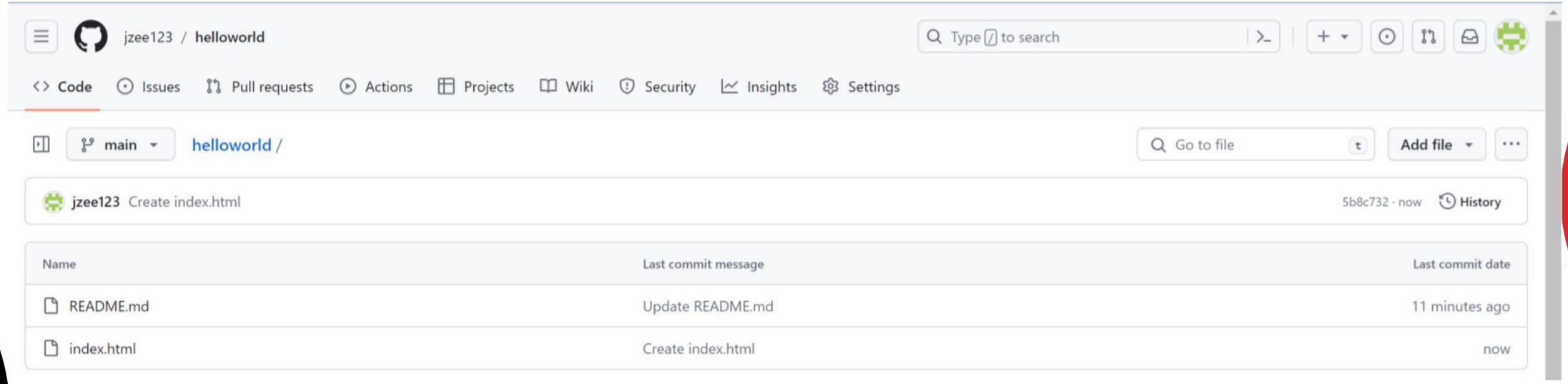
README files typically include information on:

- What the project does
- Why the project is useful
- How users can get started with the project
- Where users can get help with your project
- Who maintains and contributes to the project

If you put your README file in your repository's .git, root, or docs directory, GitHub will recognize it and automatically surface your README to repository visitors.

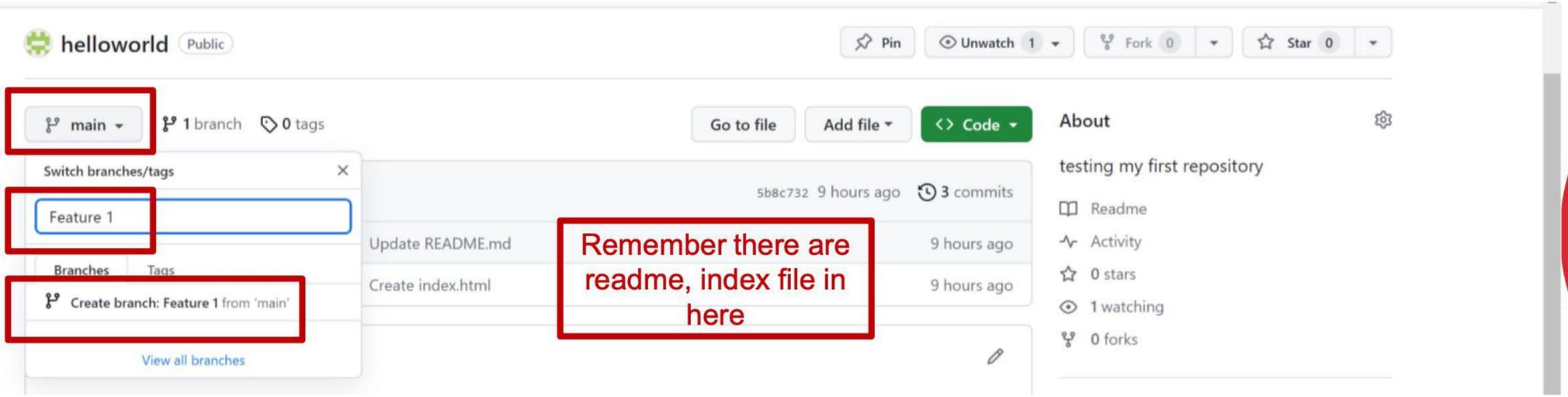

jzee123 / helloworld
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
helloworld
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 branch
0 tags
Go to file
Add file
Code
Create new file
Upload files
About
testing my first repository
jzee123 Update README.md
2 commits
README.md
Update README.md
1 minute ago
Readme
Activity

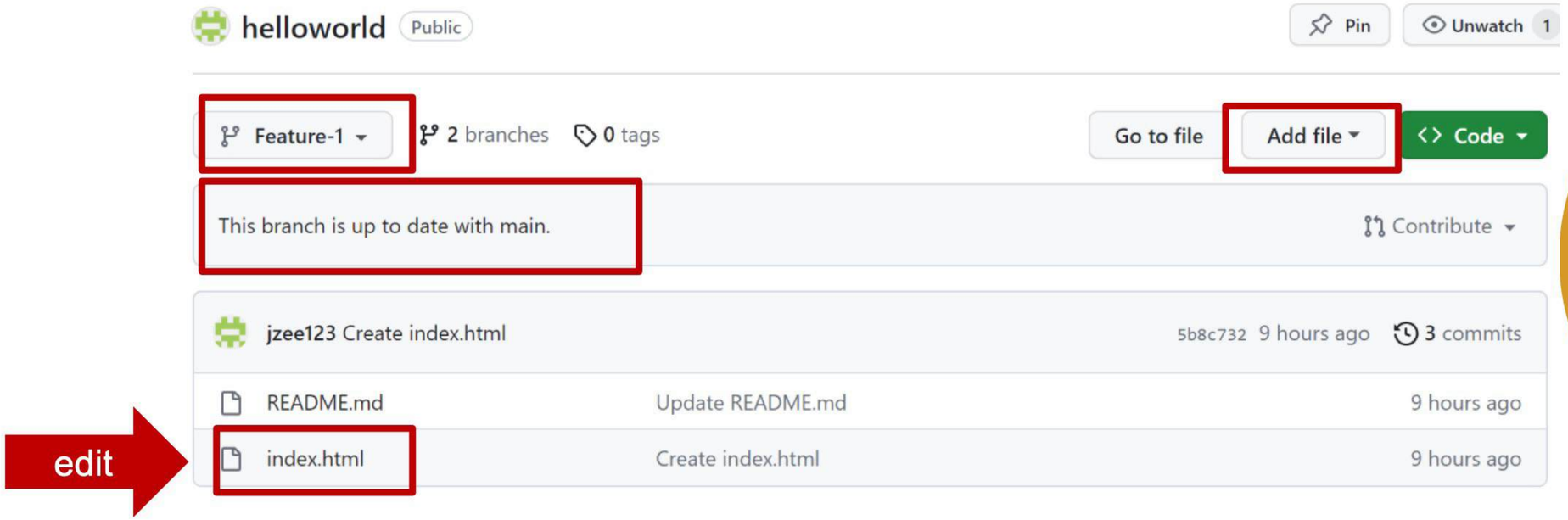

jzee123 / helloworld
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
helloworld / index.html
in main
Cancel changes
Commit changes...
Edit
Preview
Code 55% faster with GitHub Copilot
Spaces
2
No wrap
1 # my main HTML file
2
3 here goes my code .....

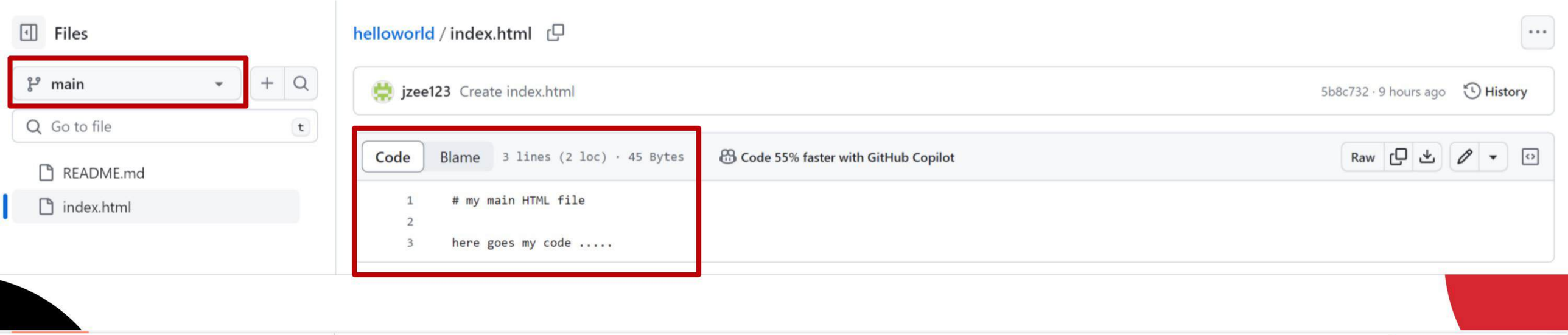


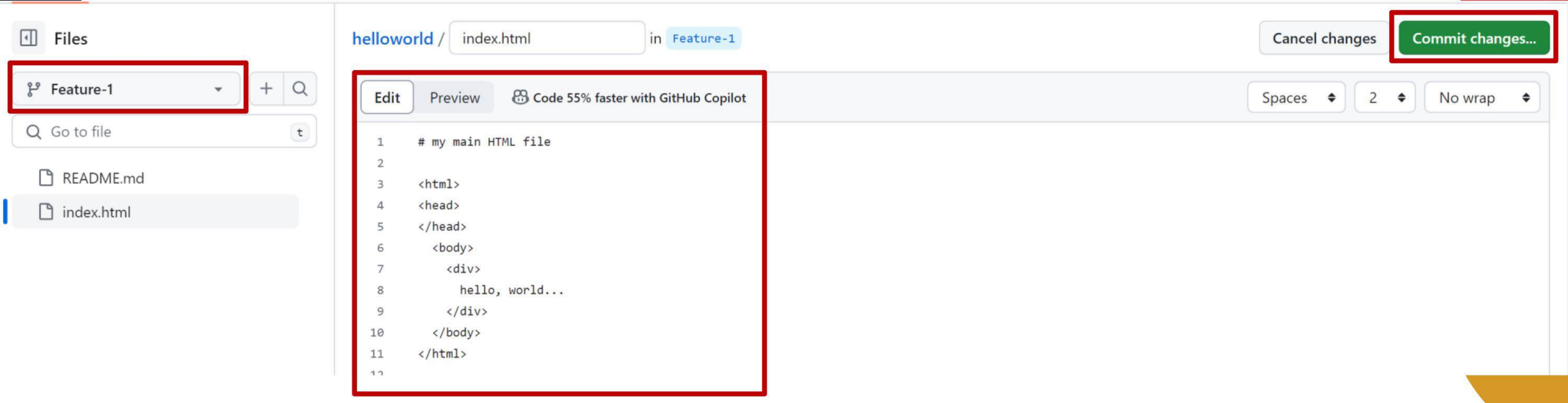
Index.html file is listed, showing when it was added

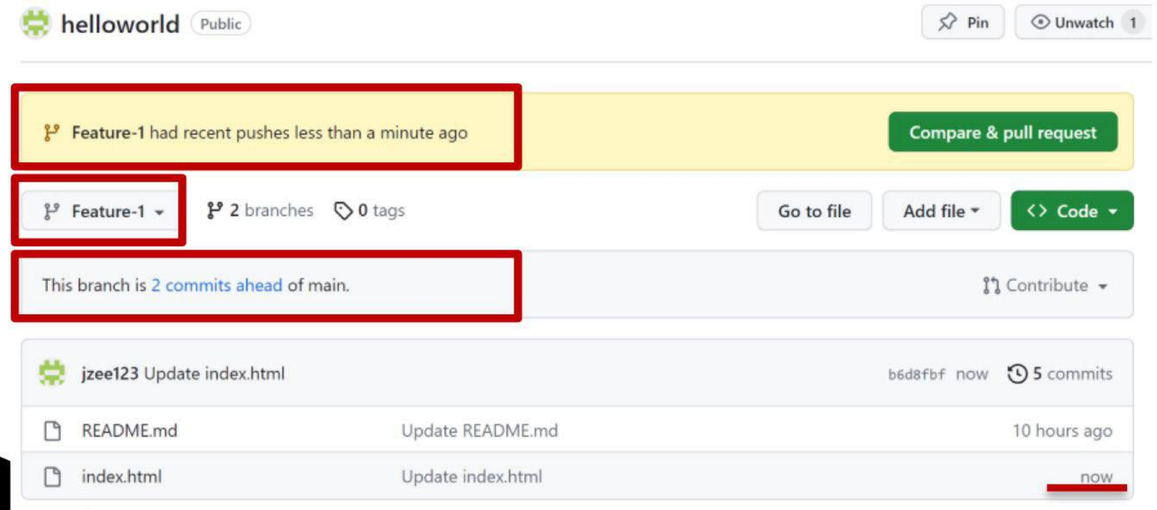
GitHub Branches

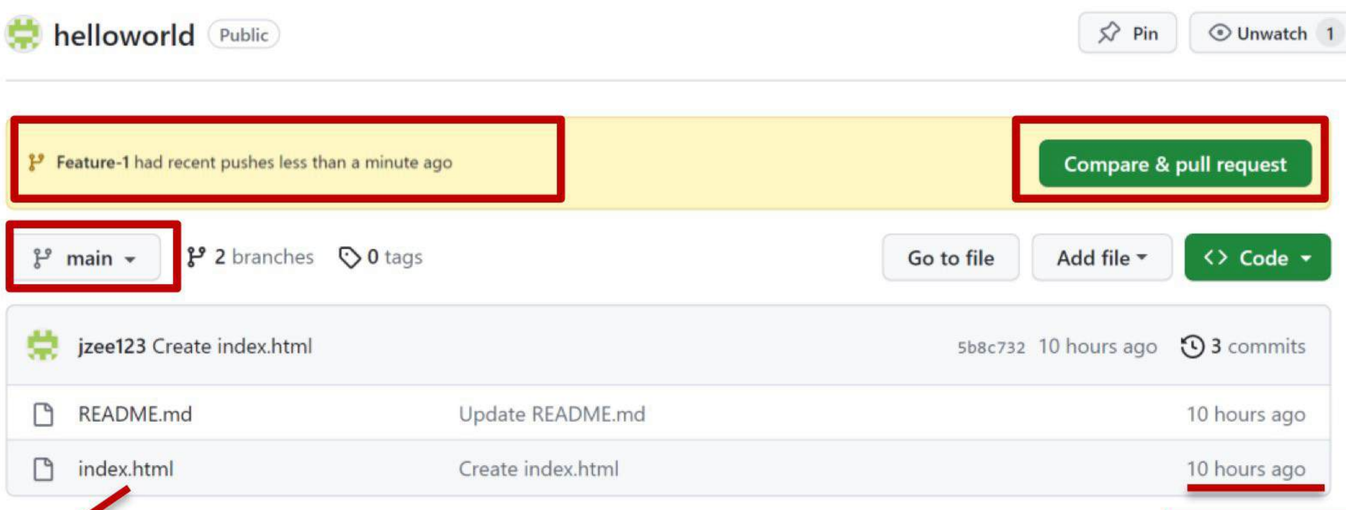

helloworld
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 branch
0 tags
Go to file
Add file
Code
About
testing my first repository
Switch branches/tags
Feature 1
Branches
Tags
Create branch: Feature 1 from 'main'
View all branches
5b8c732 9 hours ago
3 commits
Update README.md
9 hours ago
Create index.html
9 hours ago
Remember there are readme, index file in here
Readme
Activity
0 stars
1 watching
0 forks

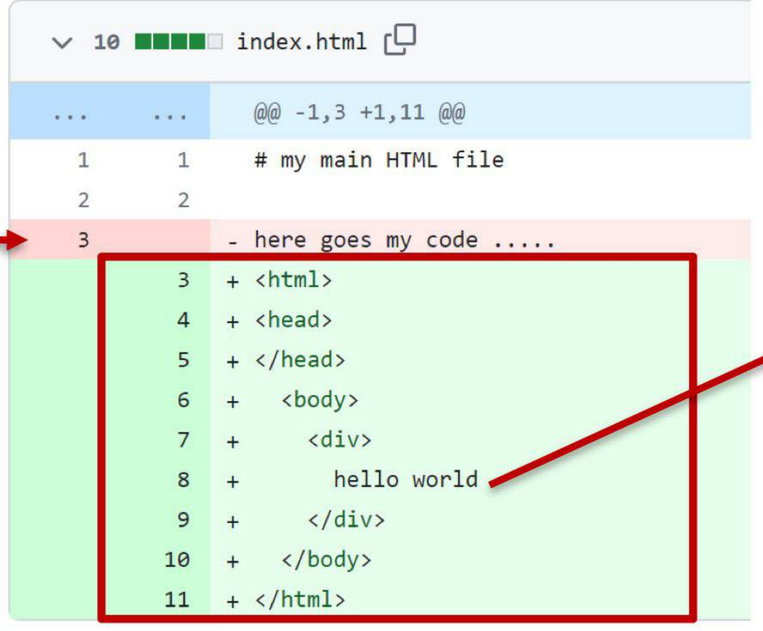

helloworld
Public
Pin
Unwatch 1
Feature-1
2 branches
0 tags
Go to file
Add file
Code
This branch is up to date with main.
Contribute
jzee123 Create index.html
5b8c732 9 hours ago
3 commits
README.md
Update README.md
9 hours ago
index.html
Create index.html
9 hours ago
edit


Files
main
Go to file
t
README.md
index.html
helloworld / index.html
jzee123 Create index.html
5b8c732 · 9 hours ago
History
Code
Blame
3 lines (2 loc) · 45 Bytes
Code 55% faster with GitHub Copilot
Raw
1 # my main HTML file
2
3 here goes my code .....


Files
Feature-1
Go to file
t
README.md
index.html
helloworld / index.html in Feature-1
Cancel changes
Commit changes...
Edit
Preview
Code 55% faster with GitHub Copilot
Spaces
2
No wrap
1 # my main HTML file
2
3 <html>
4 <head>
5 </head>
6 <body>
7
8 hello, world...
9
10 </body>
11 </html>







GIU GERMAN INTERNATIONAL UNIVERSITY

base: main ← compare: Feature-1 ✓ Able to merge. These branches can be automatically merged.

Add a title

Feature 1 Pull request

Add a description

Write Preview

Add your description here...

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

Create pull request

Code Issues Pull requests 1 Actions Projects

Filters is:pr is:open

1 Open ✓ 0 Closed

Feature 1 Pull request

#1 opened 9 minutes ago by jzee123

Feature 1 Pull request #1

Open jzee123 wants to merge 2 commits into `main` from `Feature-1`

Conversation 0 Commits 2 Checks 0 Files changed 1

jzee123 commented 7 minutes ago Owner

No description provided.

jzee123 added 2 commits 3 hours ago

Update index.html ... Verified 50fc07f

Update index.html Verified b6d8fbf

jzee123 self-assigned this 4 minutes ago

Add more commits by pushing to the `Feature-1` branch on jzee123/helloworld.

Continuous integration has not been set up

GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Add more commits by pushing to the `Feature-1` branch on jzee123/helloworld.

Merge pull request #1 from jzee123/Feature-1

Feature 1 Pull request

This commit will be authored by 153213369+jzee123@users.noreply.github.com

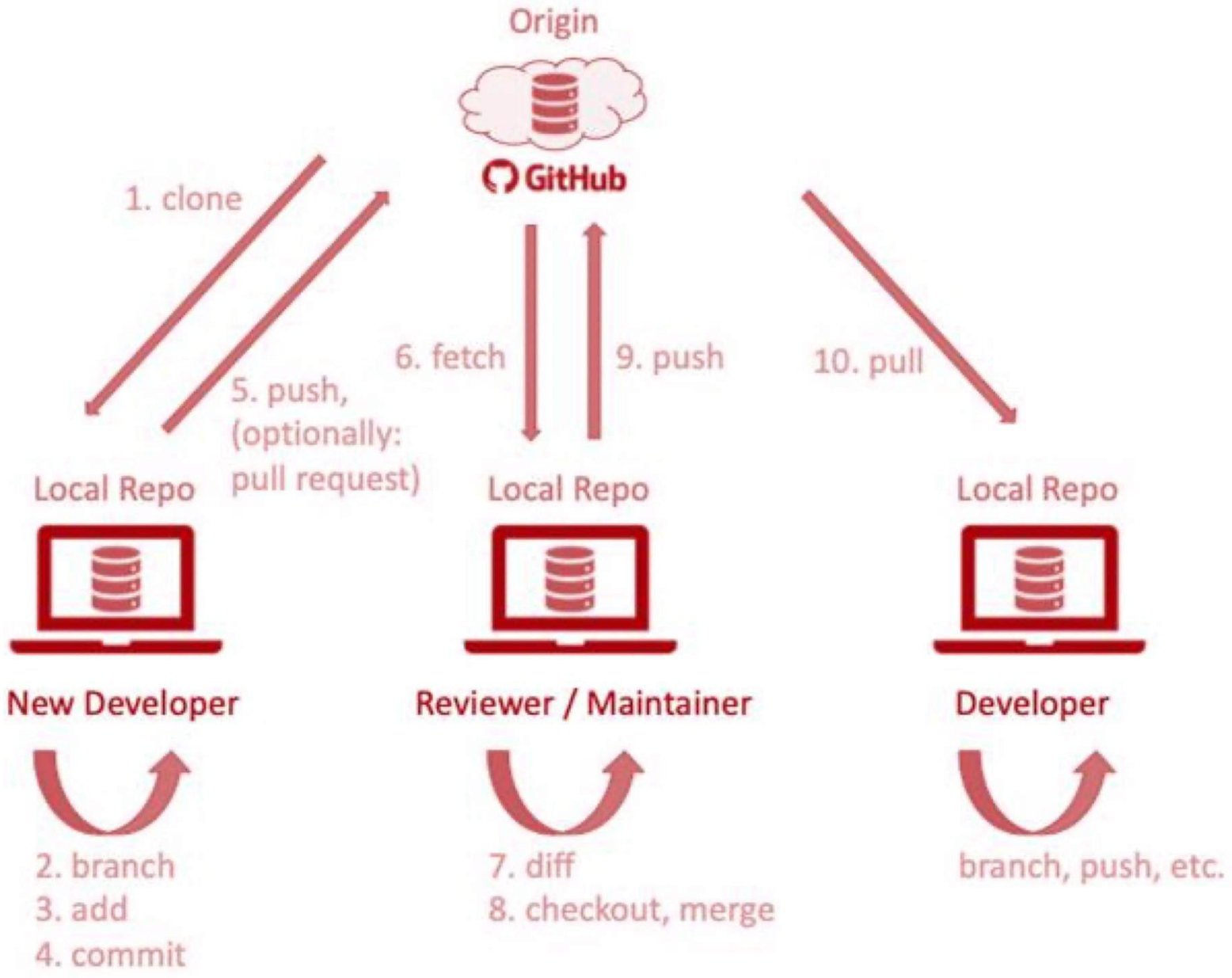
Confirm merge Cancel

FORK & CLONE

CLONE

EX:
A NEW DEVELOPER JOINS THE TEAM TO COLLABORATE ON THE PROJECT

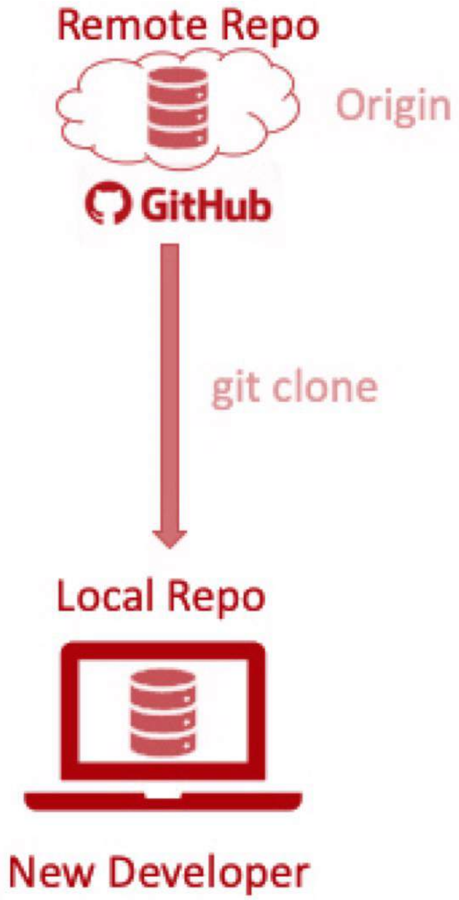
USE git <command>



CLONE

CLONE refers to creating a copy of a repository on your local machine. Cloned copies can be kept in sync between the two locations

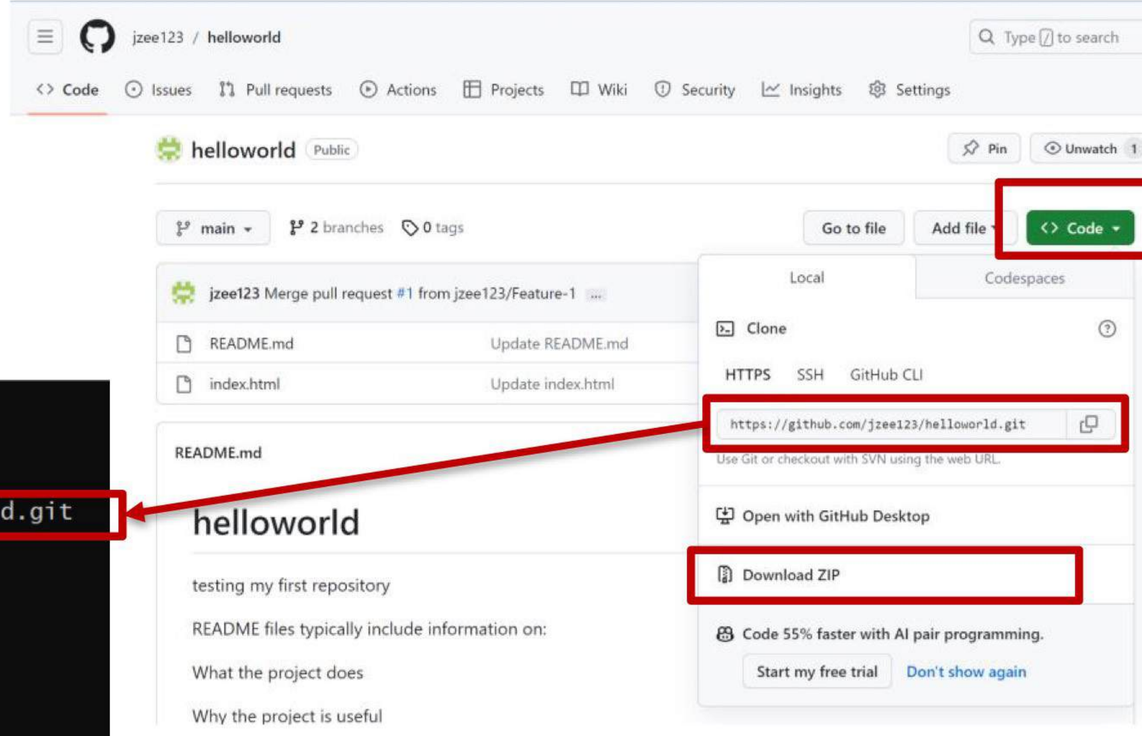
EX: A new developer joins the team to collaborate on the project. This developer can create an identical copy of the remote repo (origin) using the git clone operation.

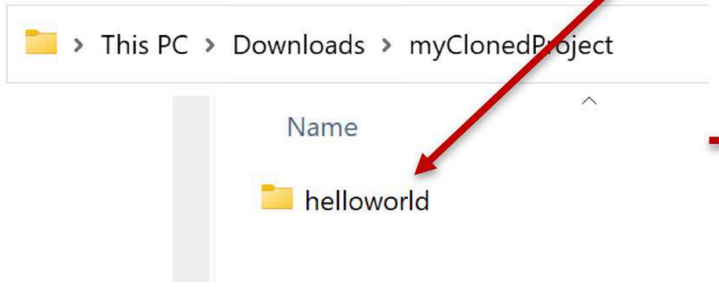


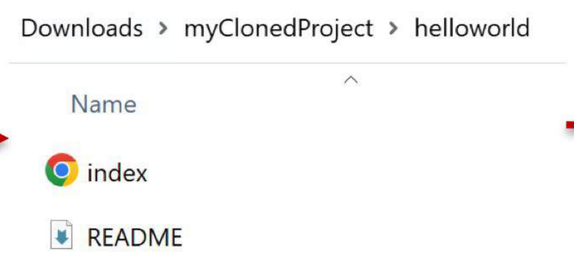
CLONE

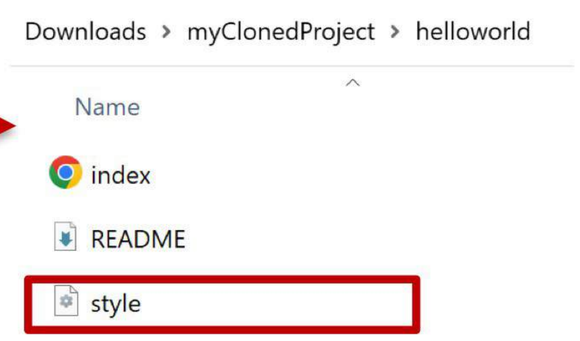
Git CMD

```
C:\Users\Surface\Downloads>md myClonedProject

C:\Users\Surface\Downloads>cd myClonedProject

C:\Users\Surface\Downloads\myClonedProject>git clone https://github.com/jzee123/helloworld.git
Cloning into 'helloworld'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 16 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (16/16), 4.16 KiB | 4.16 MiB/s, done.

C:\Users\Surface\Downloads\myClonedProject>
```









ADD, COMMIT, AND PUSH

Run the “git add FILENAME or use * if you have more than one file to update.

```
Surface@DESKTOP-5T1S4PO MINGW64 ~/downloads/myClonedProject
$ cd ./helloworld

Surface@DESKTOP-5T1S4PO MINGW64 ~/downloads/myClonedProject/helloworld (main)
$ git add style.css
```

This moves the changed files into **a staging area on the GitHub repository.**

When you are ready to commit the changes, use git commit –m “message”

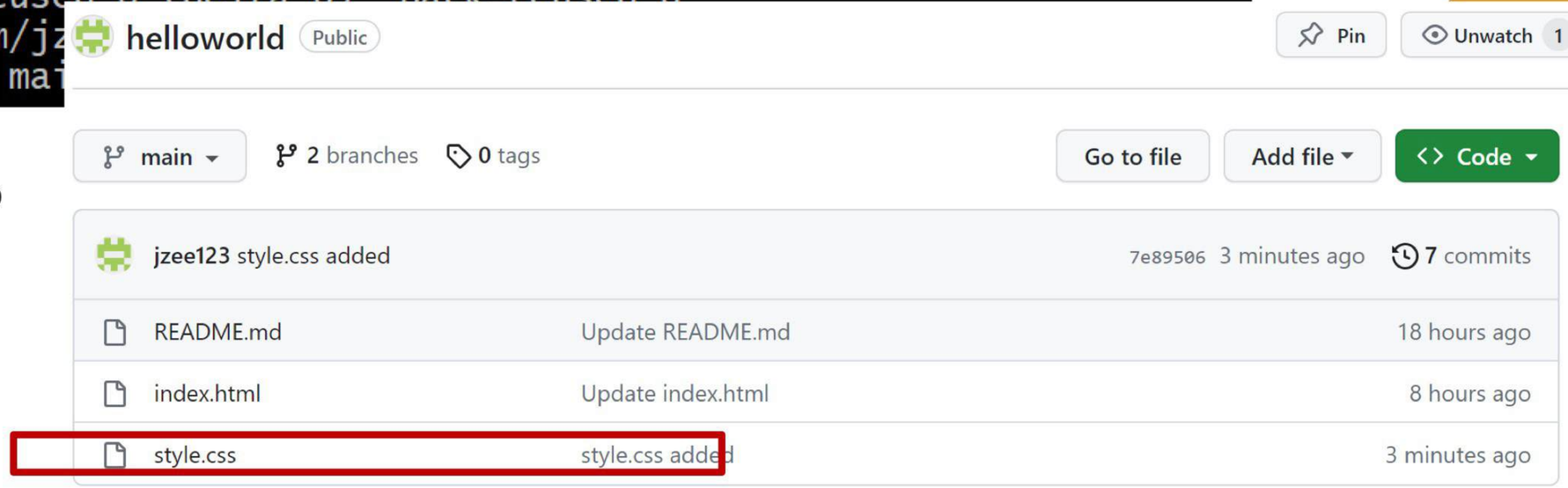
```
Surface@DESKTOP-5T1S4PO MINGW64 ~/downloads/myClonedProject/helloworld (main)
$ git commit -m "style.css added"
[main 7e89506] style.css added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 style.css
```

ADD, COMMIT, AND PUSH

Use the "git push" command. This will push all the committed changes into the repository.

```
surface@DESKTOP-5T1S4PO MINGW64 ~/downloads/myClonedProject/helloworld (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/jz
   99d1dc6..7e89506  mai
```

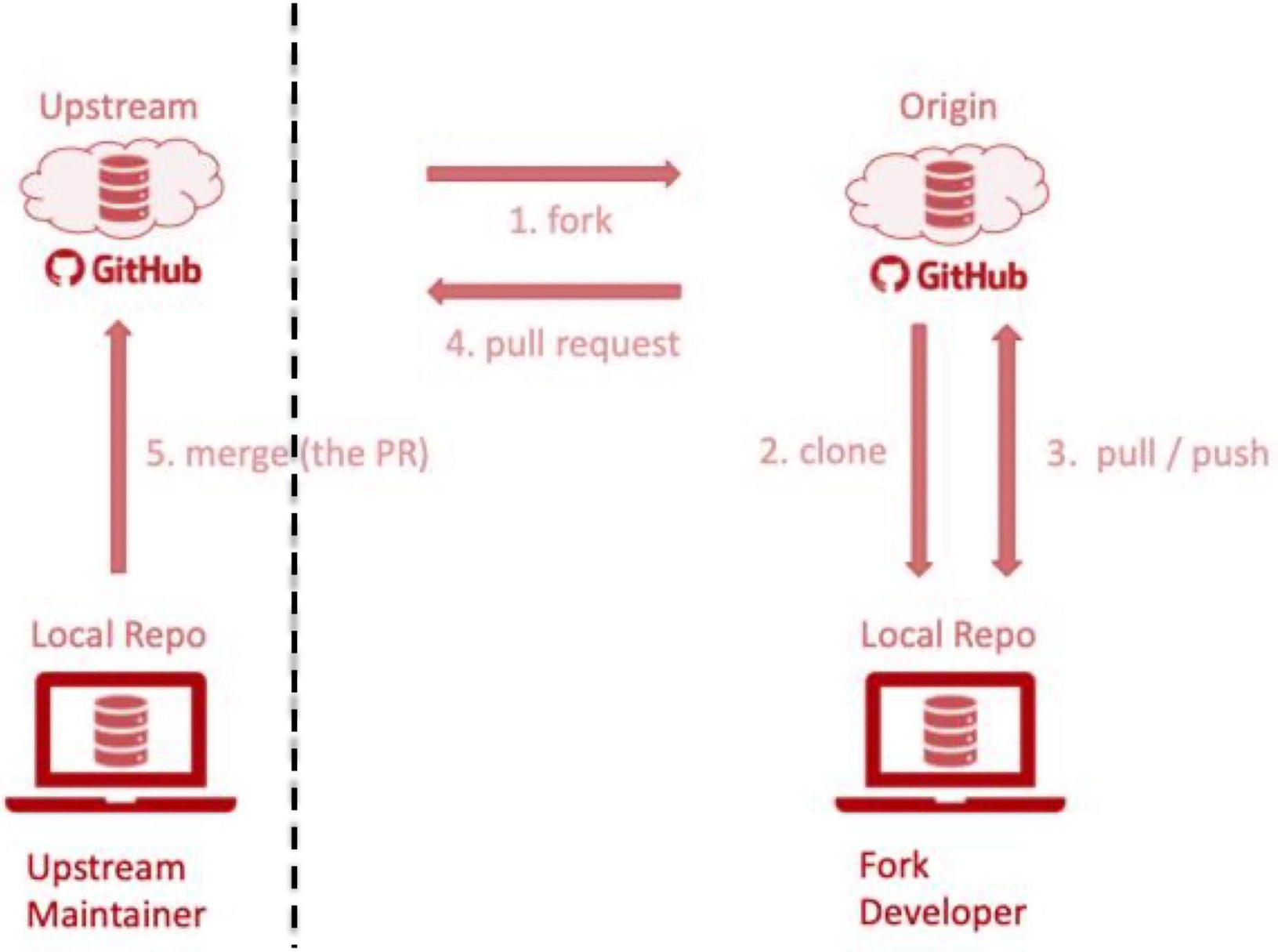
Check the online repo



FORK

GOOD PRACTICE TO FORK THE REPO IN YOUR ONLINE GitHub ACCOUNT FIRST.

FORK SHOULD ONLY BE DONE THROUGH THE GUI



FORK

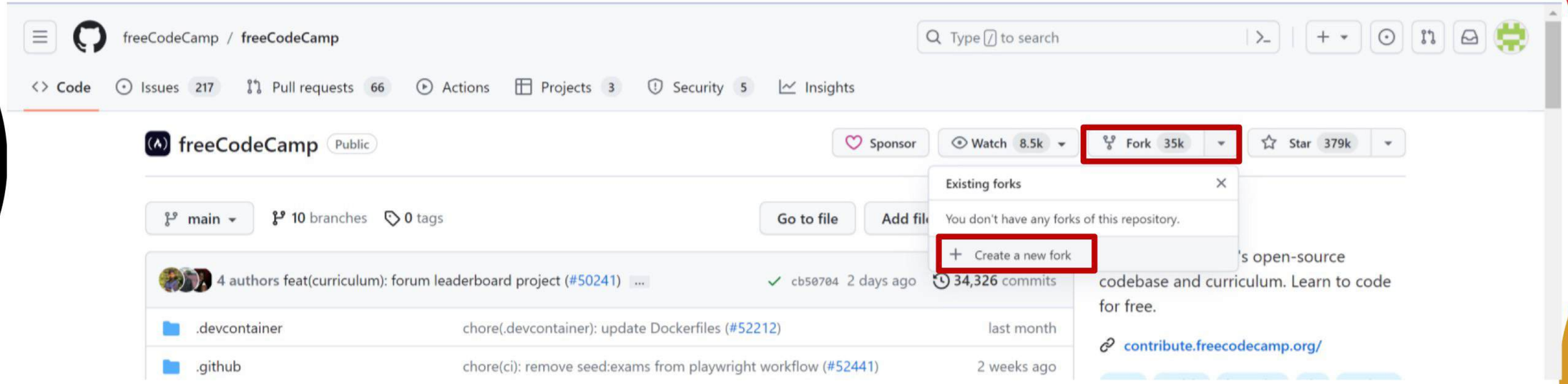
Forking allows you to modify or extend a project without affecting the original project. Usually, this is used to take an existing project as a starting point for a new project.

You can fork any public project by going to its GitHub project page and clicking on the Fork button towards the top of the page.

The project from which you create the fork is called the upstream project.

Once a project has been forked, the developers with access to the fork can work on updating and making changes to the fork using the same workflow as described previously in the clone. That is, the forked copy of the project now becomes the origin and developers with access to origin can create clones of it on their local machines, where they can create and merge branches, and synchronize changes with the origin using pull and push.

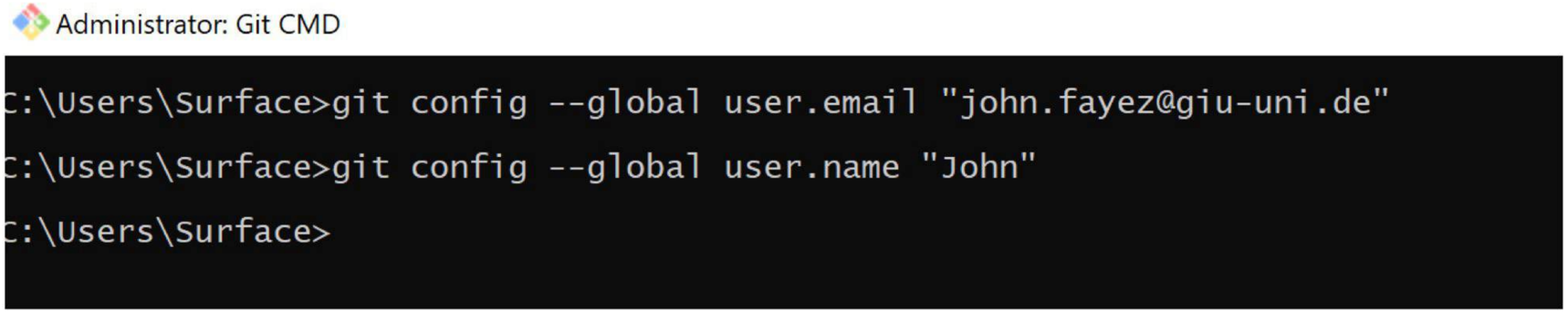
FORK


freeCodeCamp / freeCodeCamp
Type / to search
Code
Issues 217
Pull requests 66
Actions
Projects 3
Security 5
Insights
freeCodeCamp
Public
Sponsor
Watch 8.5k
Fork 35k
Star 379k
Existing forks
You don't have any forks of this repository.
Create a new fork
main
10 branches
0 tags
Go to file
4 authors feat(curriculum): forum leaderboard project (#50241)
cb50704 2 days ago
34,326 commits
.devcontainer
chore(.devcontainer): update Dockerfiles (#52212)
last month
.github
chore(ci): remove seed:exams from playwright workflow (#52441)
2 weeks ago
codebase and curriculum. Learn to code
for free.
contribute.freecodecamp.org/

USE IN TERMINAL

INSTALL AND CONFIGURE git

- Install git for Windows, for MacOS, comes with Linux.
- Or Install github desktop
- Open the git CMD as administrator
- Configure your email
- Configure your name



SSH KEYS

An SSH key is an access credential in the SSH protocol. Its function is similar to that of user names and passwords, but the keys are primarily used for automated processes

To generate an SSH key

1. Launch a terminal. If you are using Windows, launch Git Bash.
2. Type the following command in your terminal, replacing <your email address> with the email address that is linked to your Github account. A new SSH key will be generated.

```
ssh-keygen -t rsa -b 4096 -C "<your email address>“
```

You will be prompted to enter a directory to save the key. You can simply press Enter to accept the default location, which is an .ssh folder in the home directory. This means you will be able to locate the key in ~/.ssh/id_rsa.

SSH KEYS

1.You will be prompted to choose a passphrase. You also have the option not to create a passphrase. To skip the passphrase, press Enter twice to confirm that the passphrase is empty.

2.navigate to the .ssh directory, and check the contents of the directory, run the following commands in the terminal:

cd ~/.ssh THEN use command ls to list the content of the directory

When you list the contents of the .ssh directory, you should see id_rsa and id_rsa.pub in the list of contents, where id_rsa is the private version of your key and id_rsa.pub is the public version of your key.
Add the SSH key to the ssh-agent, for the authentication process.
To start the ssh-agent, run the following command in the terminal:

```
eval "$(ssh-agent -s)"
```

To add the key to the agent, run the following command in the terminal:

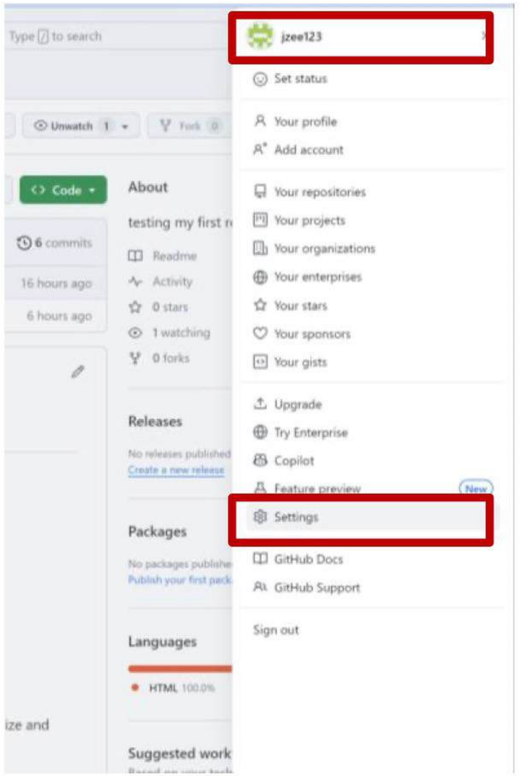
```
ssh-add ~/.ssh/id_rsa
```

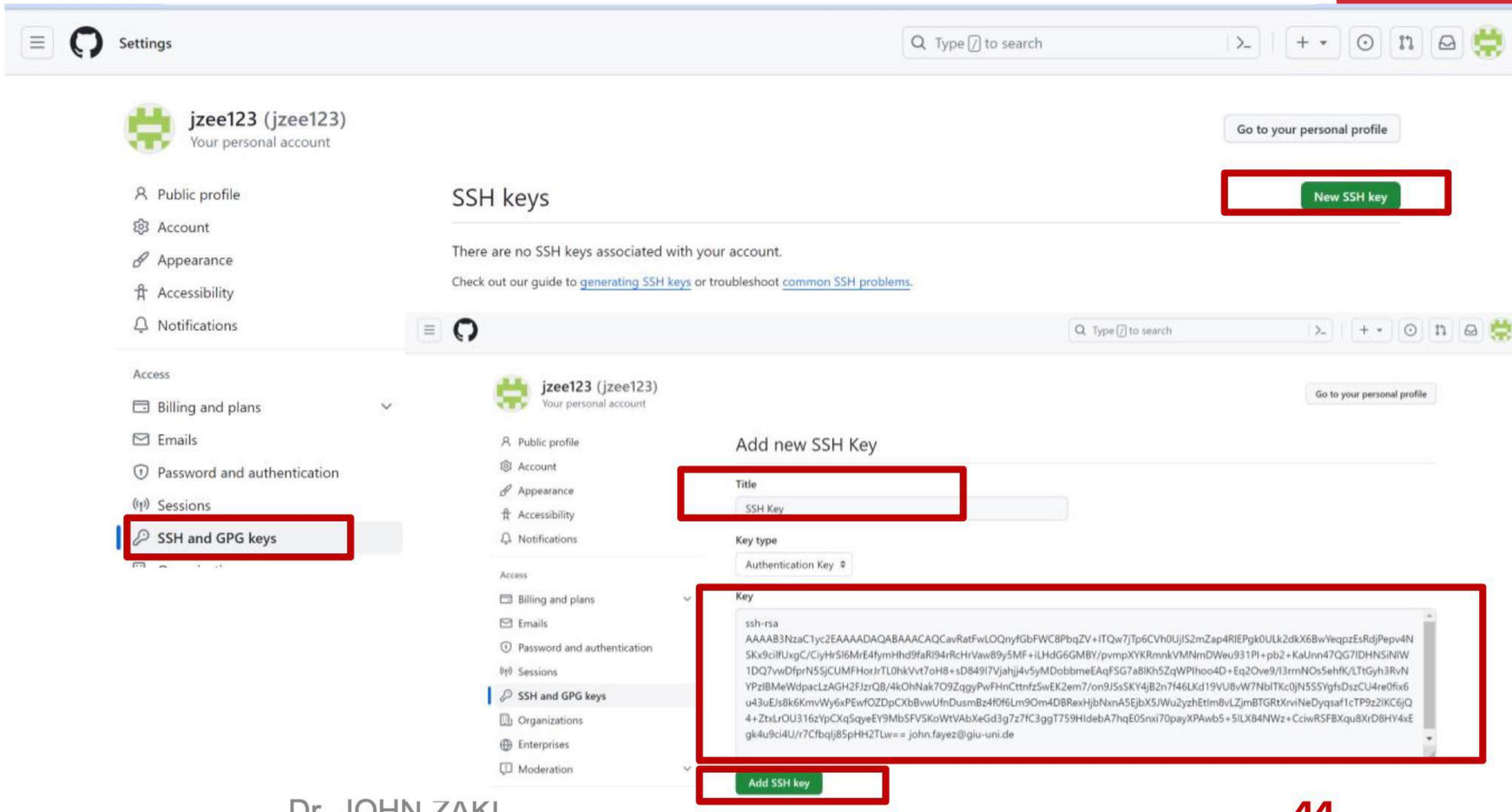
```
C:\Users\Surface>ssh-keygen -t rsa -b 4096 -C "john.fayez@giu-uni.de"
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Surface/.ssh/id_rsa):
Created directory 'C:\Users\Surface/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Surface/.ssh/id_rsa.
Your public key has been saved in C:\Users\Surface/.ssh/id_rsa.pub.
The key fingerprint is:
                                                    john.fayez@giu-uni.de
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|                 |
|              .  |
|         .     + =|
|          o     o X+|
|         . S   E =+*|
|          o     o==+|
|            .  +o==*o|
|           .. .+O..*O|
|         oo .o..**o=|
+----[SHA256]-----+
```

BASH TERMINAL

ADD KEYS TO REPO

To add an SSH key to GitHub, copy the SSH key that you previously generated.
In the terminal, run the following command: cat ~/.ssh/id_rsa.pub | clip





aka Command Line Interface

CLI EXERCISE

Configure your email and name
git config --global user.email you@example.com
git config --global user.name "Your Name"

create a new local repository using **git init**

Create a new file using **touch** (or just create the file in the working directory)
Touch <NAME OF THE NEW FILE>

add a file to the repo using **git add**
Git add <NAME OF THE NEW FILE>
If you had more than one file to add use **Git add ***

commit changes using **git commit** with a message (exactly like you used to do in the repo GUI)
git commit -m "added newfile"

That created a default branch called master

CLI EXERCISE

create a branch using **git branch**
git branch my1stbranch

switch to a branch using **git checkout**
git checkout my1stbranch

Use git branch to check that you switched form the master to the branch you created
git branch

ALTERNATIVE: Create the branch and making it active at the same time.
git checkout -b my1stbranch

check the status of files changed using **git status**

review recent commits using **git log**

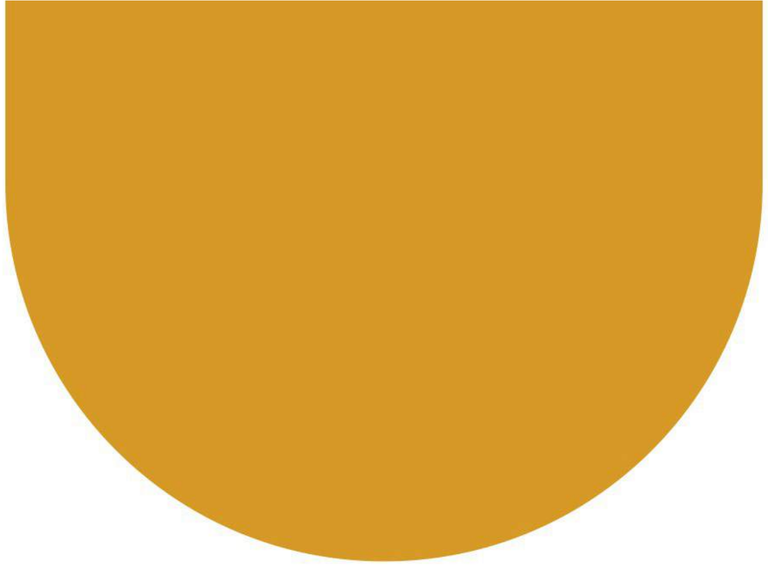
CLI EXERCISE

revert changes using **git revert**
git revert HEAD --no-edit
If you don't use –no-edit, you will be faced by an editor screen.

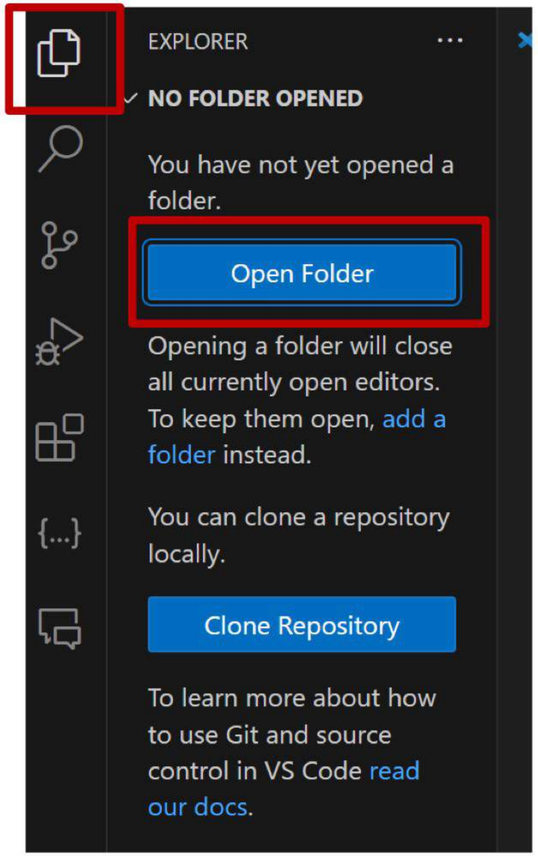
Switch to master branch (git checkout master)

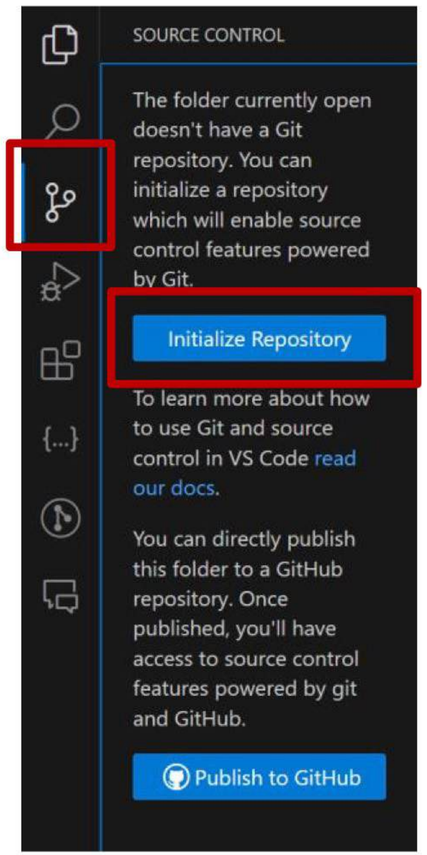
merge changes in your active branch into another branch using **git merge**
git merge my1stbranch
git log

Delete the branch that was created
git branch -d my1stbranch

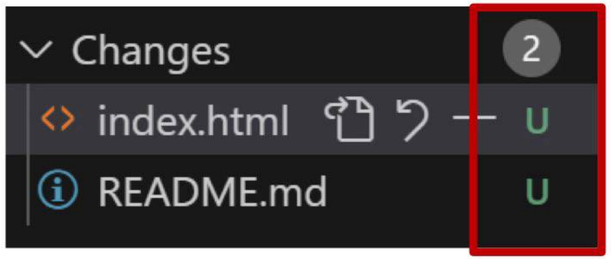


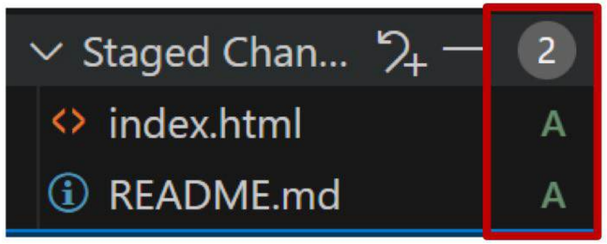
git &VS CODE

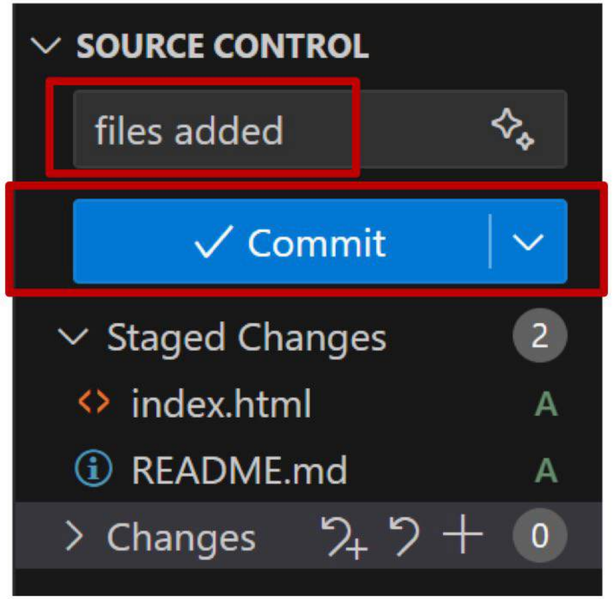

EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
You can clone a repository locally.
Clone Repository
To learn more about how to use Git and source control in VS Code read our docs.

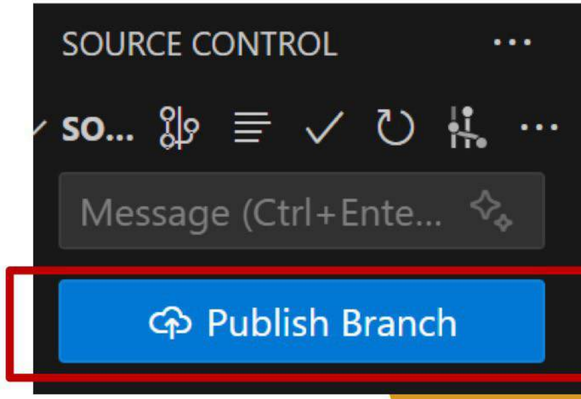

SOURCE CONTROL
The folder currently open doesn't have a Git repository. You can initialize a repository which will enable source control features powered by Git.
Initialize Repository
To learn more about how to use Git and source control in VS Code read our docs.
You can directly publish this folder to a GitHub repository. Once published, you'll have access to source control features powered by git and GitHub.
Publish to GitHub

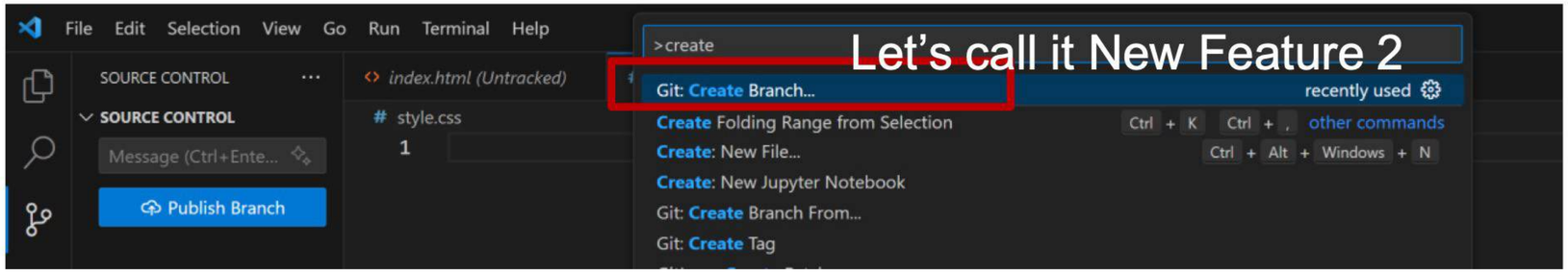

main*
0
0
0


Changes
2
index.html
U
README.md
U

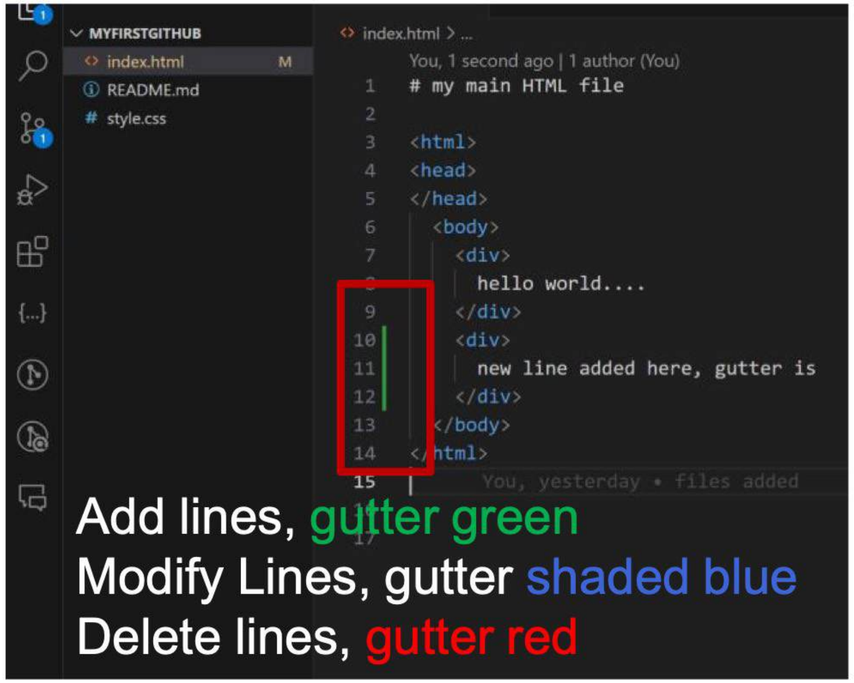

Staged Chan...
2
index.html
A
README.md
A


SOURCE CONTROL
files added
Commit
Staged Changes
2
index.html
A
README.md
A
Changes
0

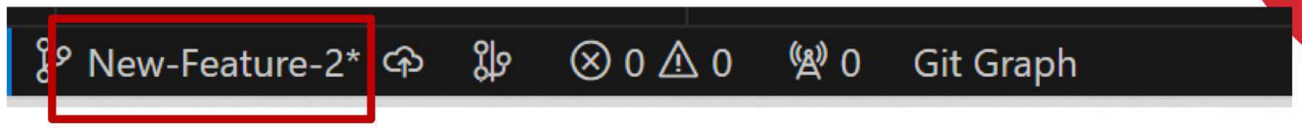

SOURCE CONTROL
SO...
Message (Ctrl+Ente...
Publish Branch

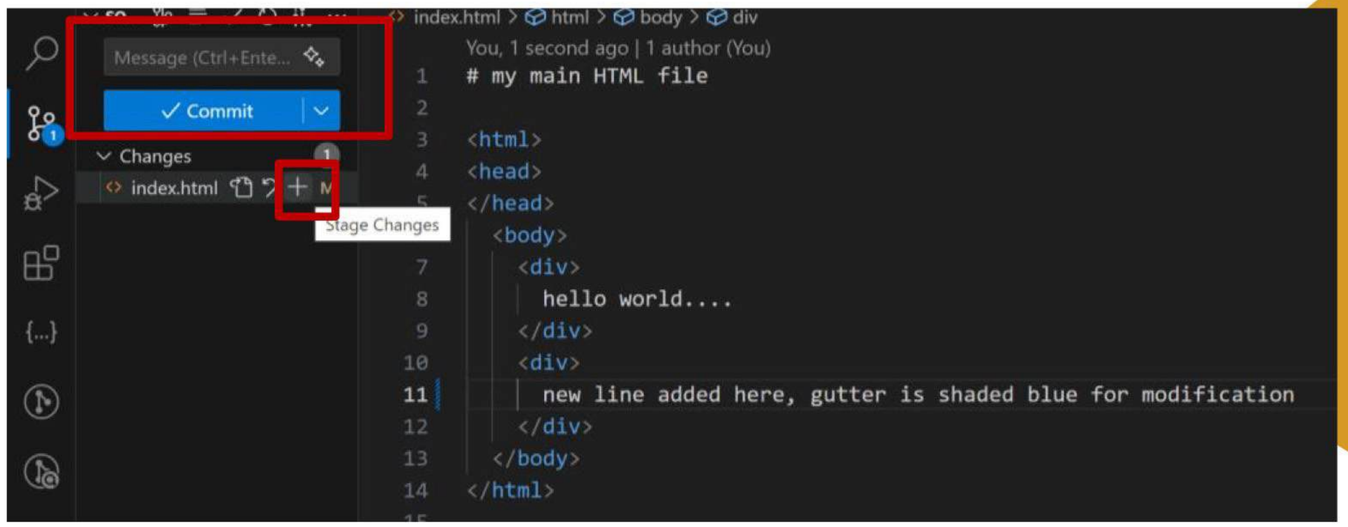


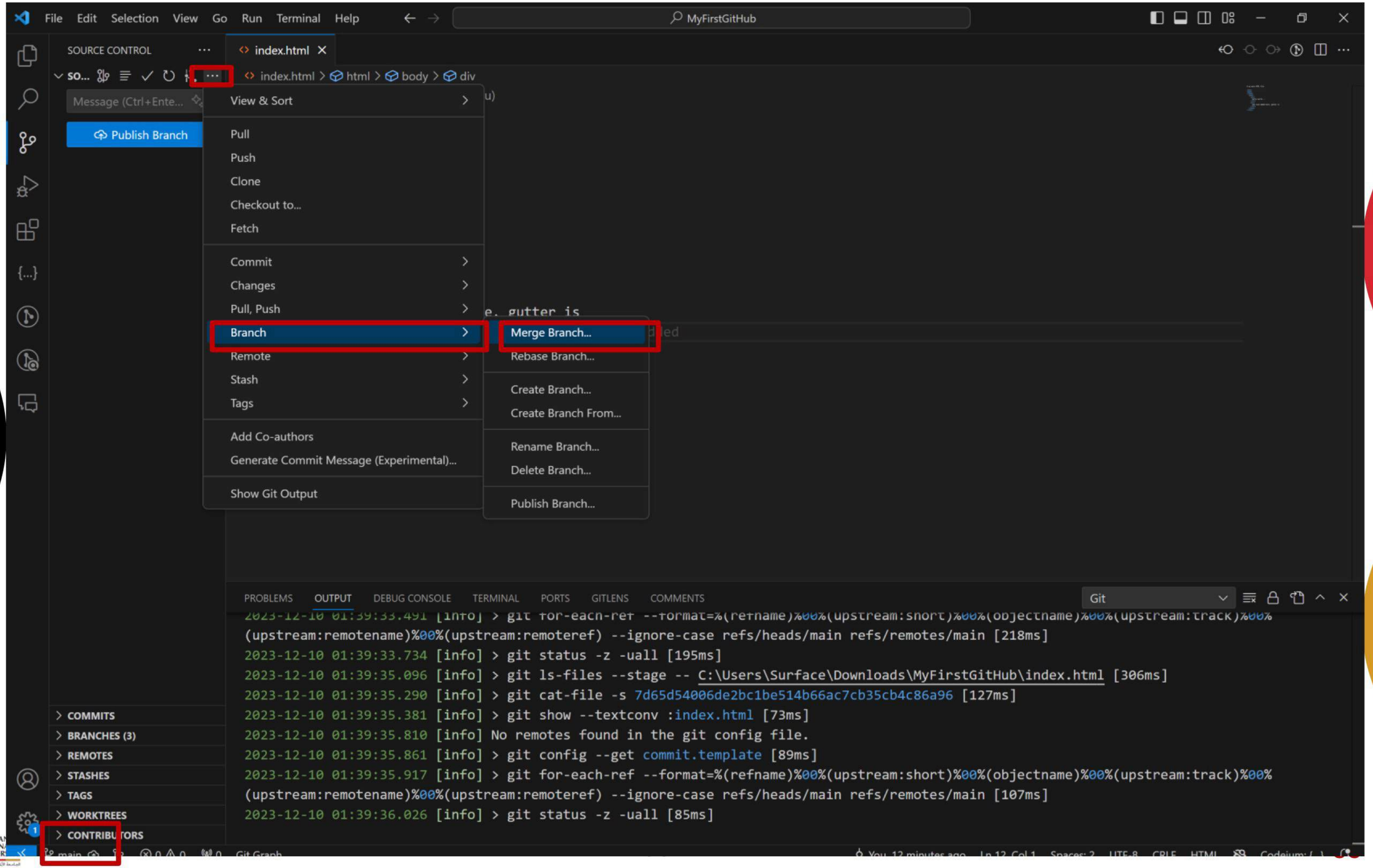
Let's call it New Feature 2

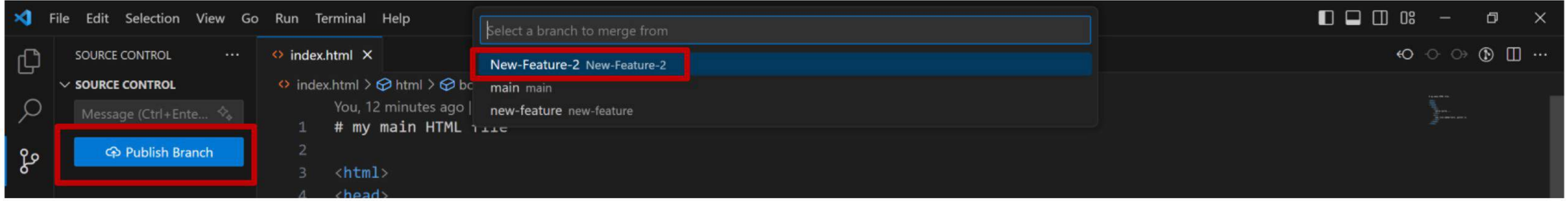


Add lines, gutter green
Modify Lines, gutter shaded blue
Delete lines, gutter red






File Edit Selection View Go Run Terminal Help
MyFirstGitHub
SOURCE CONTROL
index.html
index.html > html > body > div
Message (Ctrl+Ente...
Publish Branch
View & Sort
Pull
Push
Clone
Checkout to...
Fetch
Commit
Changes
Pull, Push
Branch
Remote
Stash
Tags
Add Co-authors
Generate Commit Message (Experimental)...
Show Git Output
Merge Branch...
Rebase Branch...
Create Branch...
Create Branch From...
Rename Branch...
Delete Branch...
Publish Branch...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS
Git
(upstream:remotename)%00%(upstream:remoteref) --ignore-case refs/heads/main refs/remotes/main [218ms]
2023-12-10 01:39:33.734 [info] > git status -z -uall [195ms]
2023-12-10 01:39:35.096 [info] > git ls-files --stage -- C:\Users\Surface\Downloads\MyFirstGitHub\index.html [306ms]
2023-12-10 01:39:35.290 [info] > git cat-file -s 7d65d54006de2bc1be514b66ac7cb35cb4c86a96 [127ms]
2023-12-10 01:39:35.381 [info] > git show --textconv :index.html [73ms]
2023-12-10 01:39:35.810 [info] No remotes found in the git config file.
2023-12-10 01:39:35.861 [info] > git config --get commit.template [89ms]
2023-12-10 01:39:35.917 [info] > git for-each-ref --format=%(refname)%00%(upstream:short)%00%(objectname)%00%(upstream:track)%00%
(upstream:remotename)%00%(upstream:remoteref) --ignore-case refs/heads/main refs/remotes/main [107ms]
2023-12-10 01:39:36.026 [info] > git status -z -uall [85ms]
COMMITS
BRANCHES (3)
REMOTES
STASHES
TAGS
WORKTREES
CONTRIBUTORS



MyFirstGitHub Public

main 1 branch 0 tags

jzee123 added

README.md	files added
index.html	added
style.css	added

MyFirstGitHub / index.html

jzee123 added

Code Blame 16 lines (13 loc) · 202 Bytes Code 55% faster with GitHub Copilot

```
1   # my main HTML file
2
3   <html>
4   <head>
5   </head>
6     <body>
7       <div>
8         hello world....
9       </div>
10      <div>
11        new line added here, gutter is shaded blue for modification
12      </div>
13    </body>
14  </html>
```

SUMMARY

01 GitHub repo online

02 Clone, Fork, and Terminal

03 Git & VS Code

DevOps automation

“Automate whatever can be automated”

…dramatically reduces **time** and **costs**
for integration, deployment and delivery

Process automation increases **reliability** and **reproducibility**.

Automation info is encoded in scripts and system models
that can be checked, reviewed, versioned and stored in the project repository

DevOps automation

Continuous integration
Each time a developer commits a change to the project's master branch, an **executable version** of the system is built and tested

Continuous delivery
A **simulation of the product's operating environment** is created and the executable software version is tested

Continuous deployment
A **new release** of the system is made available to users every time a change is made to the master branch of the software

Infrastructure as code
Machine-readable models of the infrastructure (network, servers, routers, etc.) on which the product executes are used by configuration management tools to build the software's **execution platform**: Software to be installed, such as compilers and libraries and a DBMS, are included in the infrastructure model

DevOps automation

Continuous integration
Each time a developer commits a change to the project's master branch, an **executable version** of the system is built and tested

Continuous delivery
A **simulation of the product's operating environment** is created and the executable software version is tested

Continuous deployment
A **new release** of the system is made available to users every time a change is made to the master branch of the software

Infrastructure as code
Machine-readable models of the infrastructure (network, servers, routers, etc.) on which the product executes are used by configuration management tools to build the software's **execution platform**: Software to be installed, such as compilers and libraries and a DBMS, are included in the infrastructure model

System integration

System integration (*aka* system building) is the process of:

1. gathering all of the elements required in a working system
2. moving them into the right directories
3. putting them together to create an operational system

Activities that are part of the system integration process include:

- Installing database software & setting up the database with the appropriate schema
- Loading test data into the database
- Compiling the files that make up the product
- Linking the compiled code with the libraries and other components used
- Checking that external services used are operational
- Deleting old configuration files and moving configuration files to the correct locations
- Running a set of system tests to check that the integration has been successful

Continuous integration

…integrated version of the system is **created** and **tested** every time a change is **pushed** to the system's shared repository: the repository sends a message to an integration server to build a new version of the product

Advantage: faster to find and fix bugs

If you make a small change and some system tests fail, the problem almost certainly lies in the new code that you have pushed to the **project repo**.

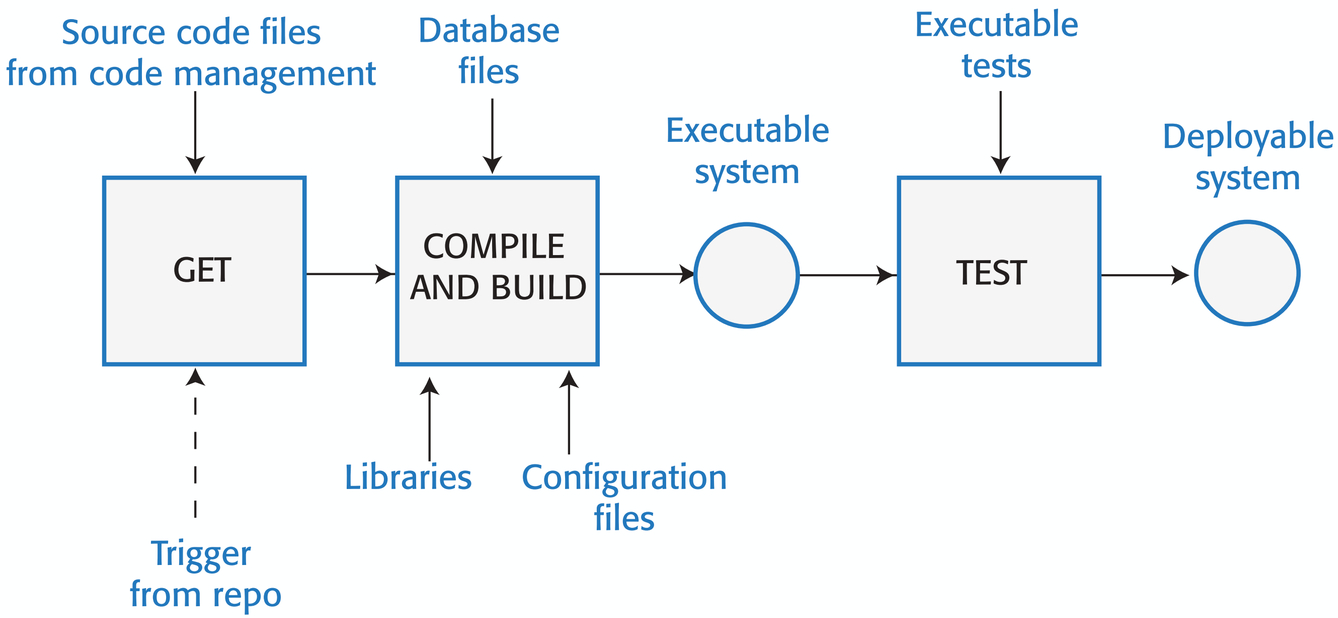


Figure 10.9 Continuous integration

"Don't break the build!"

Breaking the build… integrate locally first!

…**pushing code** to the project repository which, when integrated, **causes some of the system tests to fail**!

…it happens!

Find and fix the problem…

To **avoid** breaking the build follow the '**integrate twice**' approach:

Integrate & test on your own computer before pushing code to the project repo to trigger the integration server!

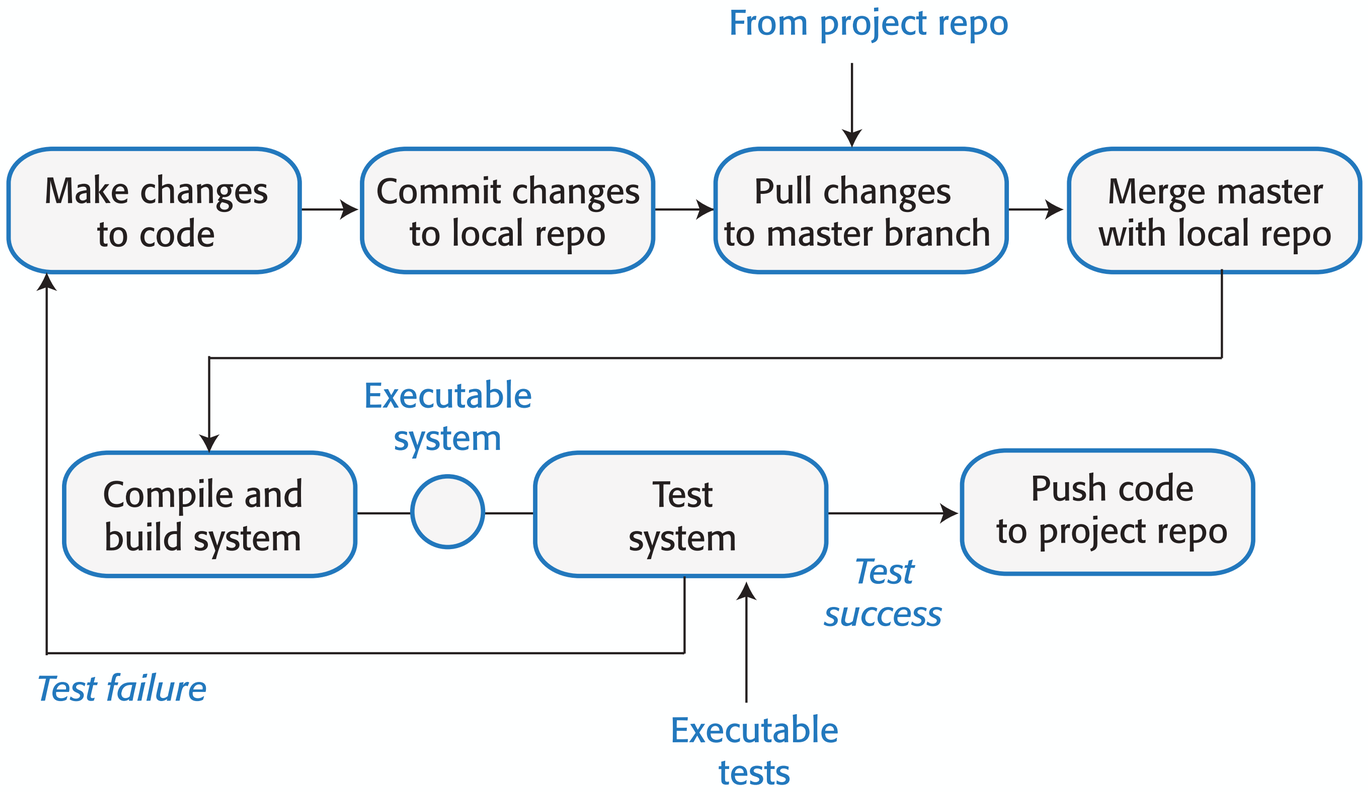


Figure 10.10 Local integration

System building and dependencies

Continuous integration only works if developers do not have to wait for the results of their tests of the integrated system

…Some activities in the build process (*e.g.* populating a database or compiling hundreds of system files) are slow

Automated build process minimises time spent on such activities

Incremental building: only those parts of the system that have been changed are rebuilt enables fast system building

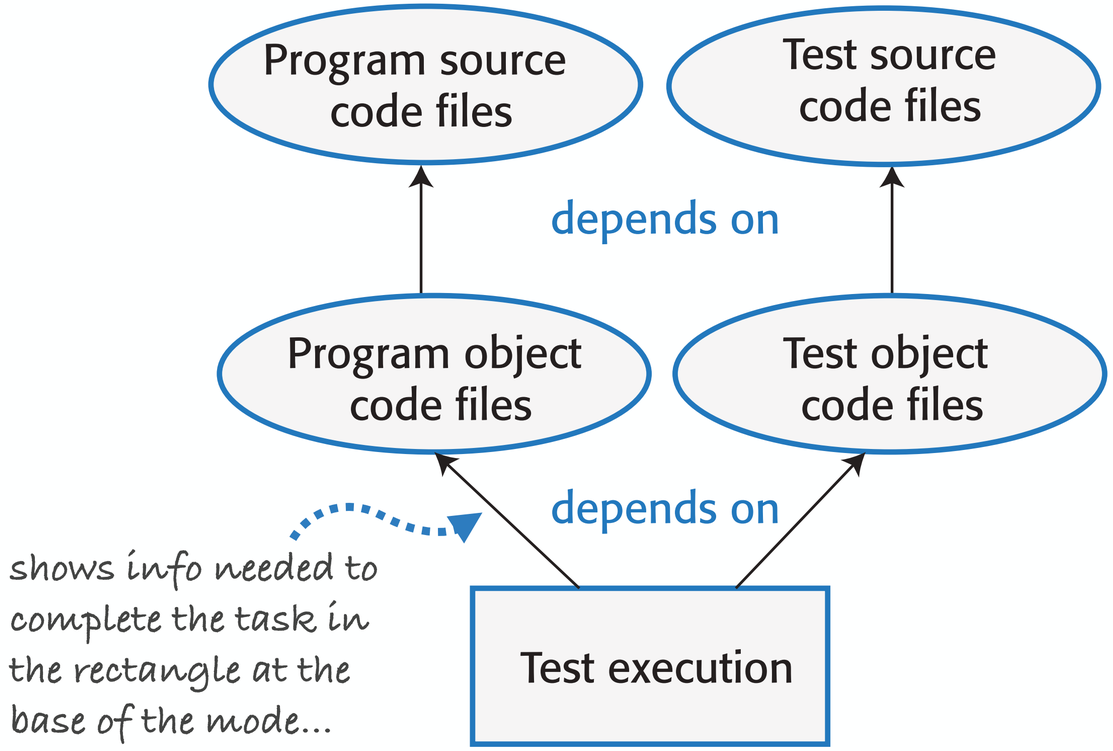


Figure 10.11 A dependency model showing dependencies for test execution

File Dependencies

Running system tests depends on **executable object code** for:

- the program being tested
- the system tests

These depend on source code for the system and the tests that are compiled to create the object code…

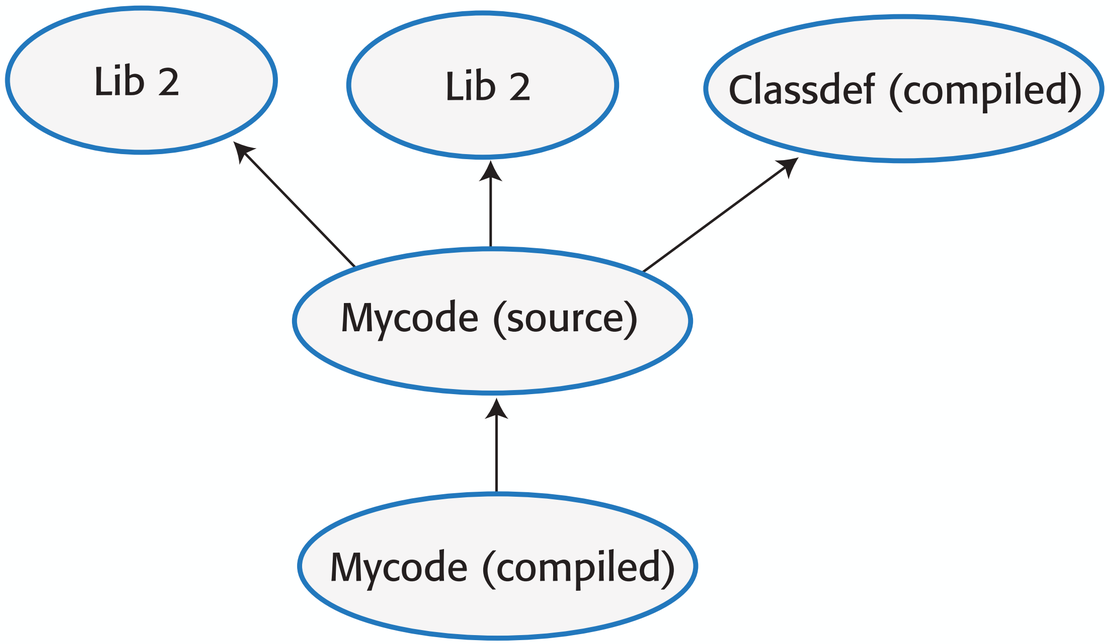


Figure 10.12 Lower-level dependency model that shows the file dependencies involved in creating the object code for a source code files called `Mycode`

File Dependencies

An automated build system uses the file modification **timestamp** to decide if a source code file has been changed

Modification date of **compiled code** > modification date of the **source code**:

No changes have been made to the source code…does nothing

Modification date of the **compiled code** < the modification date of the **source code**:

Recompiles the source
and replaces the existing file of compiled code with an updated version

File Dependencies

An automated build system uses the file modification timestamp to decide if a source code file has been changed

Modification date of **compiled code** **>** modification date of the **source code**:

…but, modification date of `Classdef` > modification date of the source code of `Mycode`…
`Mycode` has to be recompiled to incorporate these changes.

Modification date of the **compiled code** **<** the modification date of the **source code**:

Recompiles the source
and replaces the existing file of compiled code with an updated version.

DevOps automation

Continuous integration
Each time a developer commits a change to the project's master branch, an **executable version** of the system is built and tested

☛ **Continuous delivery**
A **simulation of the product's operating environment** is created and the executable software version is tested

☛ **Continuous deployment**
A **new release** of the system is made available to users every time a change is made to the master branch of the software

Infrastructure as code
Machine-readable models of the infrastructure (network, servers, routers, etc.) on which the product executes are used by configuration management tools to build the software's **execution platform**: Software to be installed, such as compilers and libraries and a DBMS, are included in the infrastructure model

Continuous delivery and deployment

Real environment in which software runs will be different from our development system… bugs may be revealed that weren't in test environment

Continuous delivery: after making changes to a system, we ensure that it is ready for delivery to customers!

Must test it in a production environment to make sure that environmental factors do not cause system failures or slow down its performance!

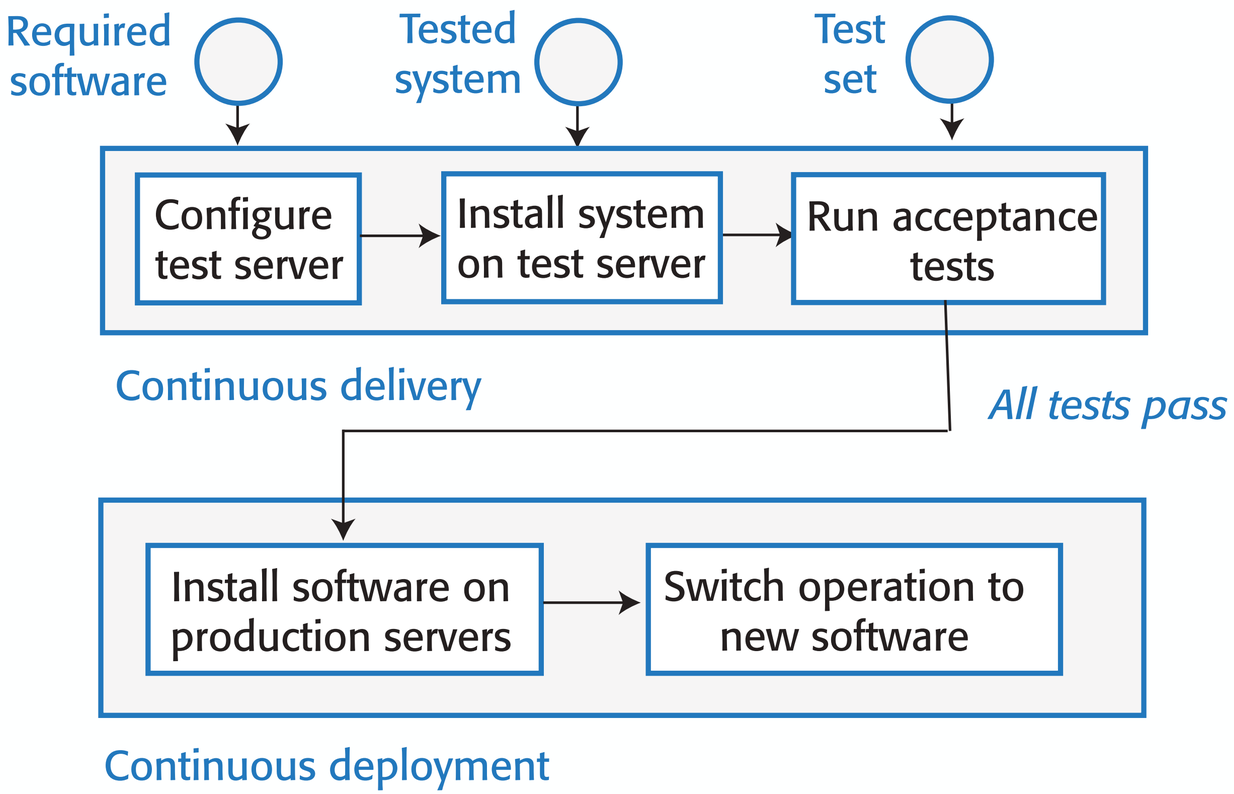


Figure 10.13 Continuous delivery and deployment

Deployment pipeline

After initial integration testing, a **staged test environment** is created: a replica of the actual production environment in which the system will run.

System acceptance tests (*e.g.* functionality, load and performance tests) are run to check the software works as expected.

If all of these tests pass, the changed software is installed on the production servers.

To **deploy** the system, you momentarily stop all new requests for service and leave the older version to process the outstanding transactions.

Once these have been completed, you **switch to the new version** of the system and **restart processing**.

Benefits of continuous deployment

Reduced costs
Must invest in a completely automated deployment pipeline. Manual deployment is a time-consuming and error-prone process. Setting up an automated system is expensive and time-consuming but you can recover these costs quickly if you make regular updates to your product.

Faster problem solving
If a problem occurs, it will probably only affect a small part of the system and it will be obvious what the source of that problem is. If you bundle many changes into a single release, finding and fixing problems is more difficult.

Faster customer feedback
You can deploy new features when they are ready for customer use. You can ask them for feedback on these features and use this feedback to identify improvements that you need to make.

A/B testing
This is an option if you have a large customer base and use several servers for deployment. You can deploy a new version of the software on some servers and leave the older version running on others. You then use the load balancer to divert some customers to the new version while others use the older version. You can then **measure** and **assess** how new features are used to see if they do what you expect.

DevOps automation

Continuous integration
Each time a developer commits a change to the project's master branch, an **executable version** of the system is built and tested

Continuous delivery
A **simulation of the product's operating environment** is created and the executable software version is tested

Continuous deployment
A **new release** of the system is made available to users every time a change is made to the master branch of the software

☛ **Infrastructure as code**
Machine-readable models of the infrastructure (network, servers, routers, etc.) on which the product executes are used by configuration management tools to build the software's **execution platform**: Software to be installed, such as compilers and libraries and a DBMS, are included in the infrastructure model

Infrastructure as code

Usually many different physical or virtual servers (web servers, database servers, file servers, etc.) that do different things. These have different configurations and run different software packages: difficult to keep track of the software installed on each machine.

Solution: infrastructure as code

Automate updates to the software using a model of the infrastructure written in a machine-processable language.

Configuration management (CM) tools such as **Puppet** and **Chef** can automatically install software and services on servers according to the infrastructure definition…

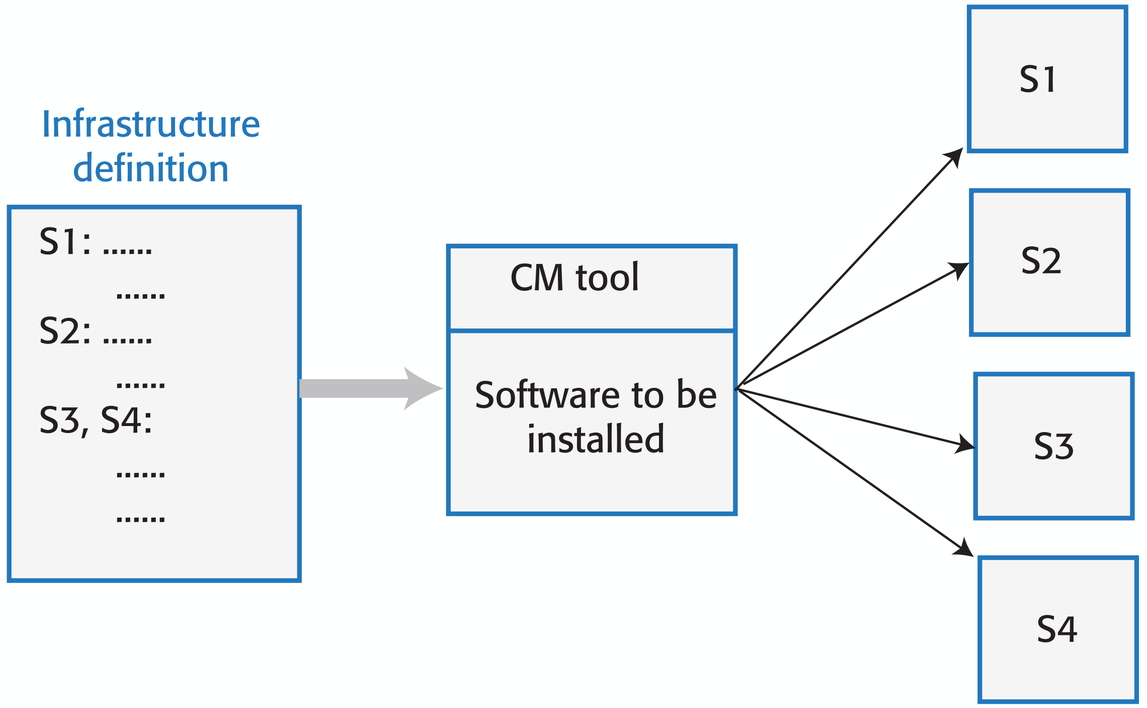


Figure 10.14 Infrastructure as code

Containers

Using containers makes it very simple to provide identical execution environments.

…provides a stand-alone execution environment running on top of an operating system.

The software installed in a **Docker container** is specified using a **Dockerfile**, which is, essentially, a definition of your software infrastructure as code.

You build an executable container image by processing the Dockerfile.

For each type of server that you use, you define the environment that you need and build an image for execution. You can run an application container as a test system or as an operational system; there is no distinction between them.

When you update your software, you rerun the image creation process to create a new image that includes the modified software. You can then start these images alongside the existing system and divert service requests to them.

DevOps measurement

Apply DevOps principle of “automating everything” to software measurement!

Collect data about the software and use a monitoring system to collect data about its performance and availability. Four types of software development measurement:

1. **Process measurement** You collect and analyse data about your development, testing and deployment processes.
2. **Service measurement** You collect and analyse data about the software’s performance, reliability and acceptability to customers.
3. **Usage measurement** You collect and analyse data about how customers use your product.
4. **Business success measurement** You collect and analyse data about how your product contributes to the overall success of the business.

Automating measurement

Payal Chakravarty (IBM) DevOps measurement based around a metrics scorecard with 9 metrics:

Process metrics we want to see:

- decreases in the number of failed deployments, the mean time to recovery after a service failure and the lead time from development to deployment
- increases in deployment frequency and number of lines of changed code that are shipped

Service metrics

- availability and performance should be stable or improving, number of customer complaints should be decreasing, and number of new customers should be increasing…

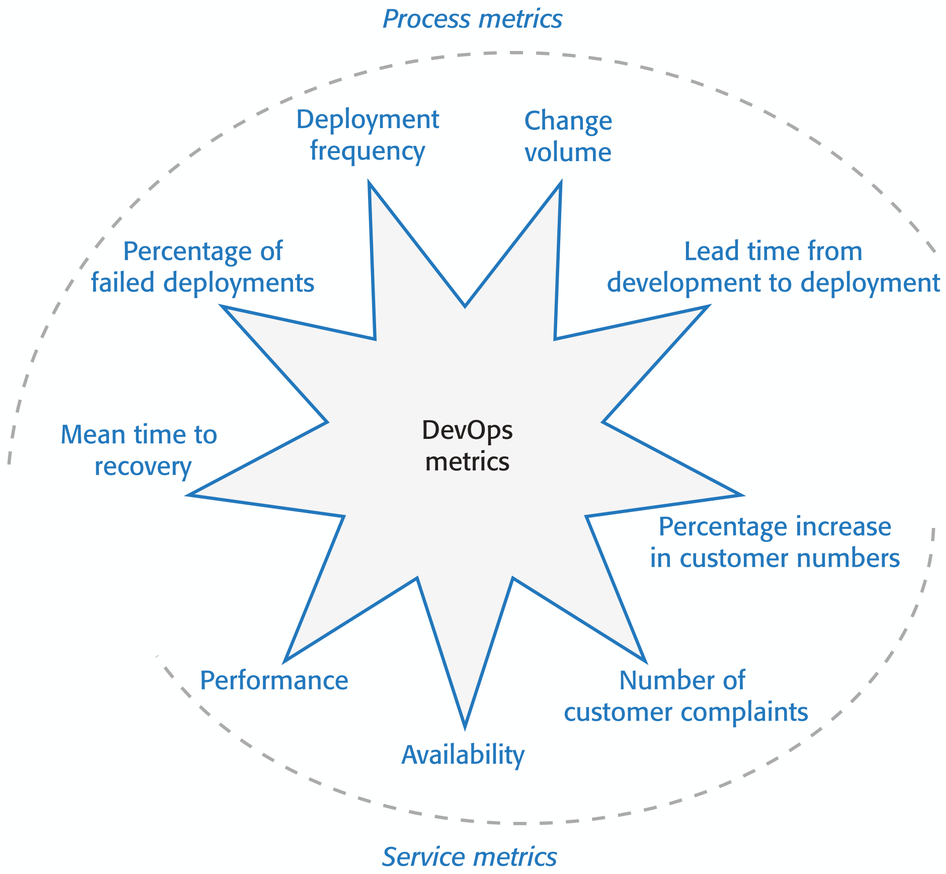


Figure 10.15 Metrics used in the DevOps scorecard

Metrics trends

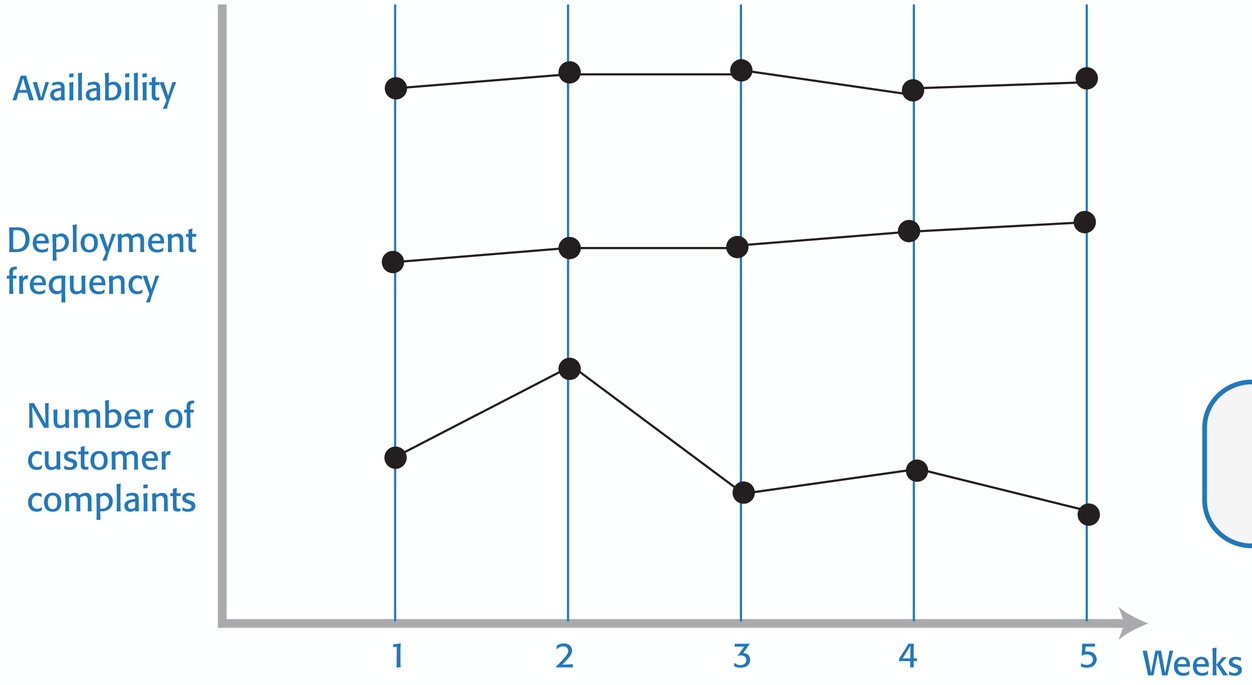


Figure 10.16 Metrics trends

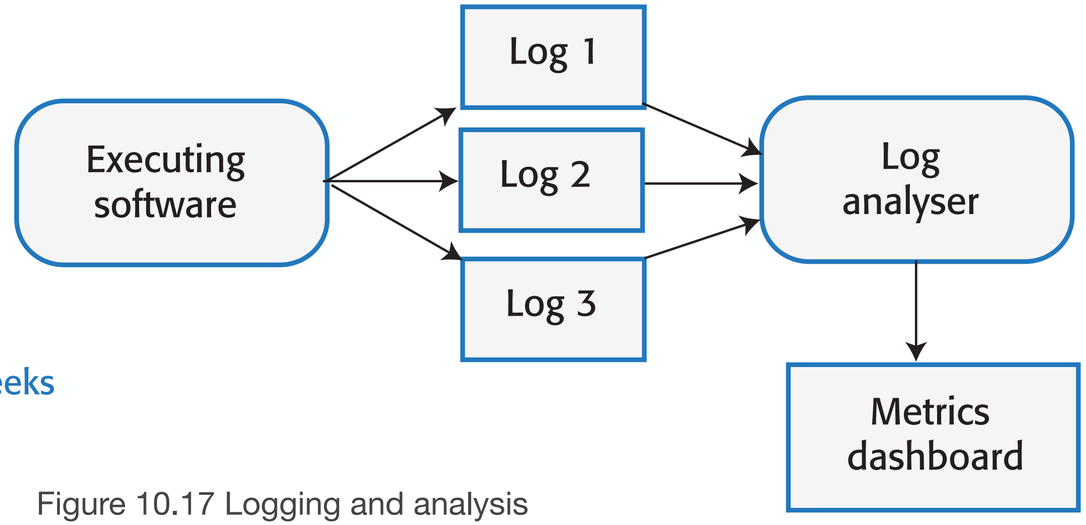


Figure 10.17 Logging and analysis

In the age of AI…

https://n8n.io