# Git & Version Control

Foundation Course @H-BRS, March 2024

# Introduction to Version Control

# What is Version Control?

Imagine you're working on a group project.

Version Control System (VCS) is a tool that does the following:

- Keep track of changes made to files in the project
- Allows you to see who changed what and when
- Lets you revert state of a particular file (or files) to an earlier version if needed
- Mix-and-match work by multiple people collaborating at the same time

# Introduction to Git

# What is Git?

Git is just one of the different types of VCSs.

Why do we teach it?

- It’s the most widely used VCS today
- You’ll need it during your studies and future work

# Getting started

Git has phenomenal official documentation. In order to get started, make sure you have Git installed on your computer.

Then, we'll do the following:

1. Configure your global identity
2. Look into preferences and the .gitconfig file

# Configure your global identity

When collaborating on a project, it’s important that people know who you are.

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

# Setting up preferences

When naming the main branch of your repository (more about this later), it is a convention to set the name to “main”.

- Historically, the default branch name was “master”
- Many teams have switched to a more inclusive naming convention, which is what we’ll be using as well (“main”)

```
$ git config --global init.defaultBranch main
```

# Checking preferences

```
$ git config --list
user.name=John Doe
user.email=johndoe@example.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

# Introduction to GitHub

# What is GitHub?

Git is version control system. GitHub is the online platform for storing Git repositories in the cloud.

It’s not the only such platform! There’s GitLab, BitBucket… GitHub is just the most commonly used one.

Many of the resources you’ll need during your studies exist on GitHub.

Private

github.com

Product Solutions Open Source Pricing

Search or jump to... /

Sign in Sign up

# HBRS-AMR

Overview | Repositories 10 | Projects | Packages | People

## Popular repositories

**Robile** — Public

The repository for Robile robots

Python 1 7

**kelo_tulip** — Public

Forked from a2s-institute/kelo_tulip

KELO platform driver

C

**robile_gazebo** — Public

Forked from a2s-institute/robile_gazebo

Gazebo simulation files for KELO ROBILE

C++ 1

**robile_description** — Public

Forked from a2s-institute/robile_description

URDF and mesh files for KELO ROBILE

CMake

**robile_navigation** — Public

Forked from a2s-institute/robile_navigation

Navigation launch files for robile in gazebo and real robot

CMake 3

**SS23_Assignment_BehaviorTree** — Public template

Python

## People

This organization has no public members. You must be a member to see who's a part of this organization.

## Top languages

Python CMake C++ C

# Create an account

Create an account on GitHub (if you don’t have it already).

Setup two-factor authentication (if you don’t have it already).

# 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. We'll use the official documentation by GitHub to go through the setup together.

# Generating a new SSH key and adding it to the ssh-agent

After you've checked for existing SSH keys, you can generate a new SSH key to use for authentication, then add it to the ssh-agent.

Mac Windows Linux

In this article



## About SSH key passphrases

You can access and write data in repositories on GitHub.com using SSH (Secure Shell Protocol). When you connect via SSH, you authenticate using a private key file on your local machine. For more information, see "About SSH."

When you generate an SSH key, you can add a passphrase to further secure the key. Whenever you use the key, you must enter the passphrase. If your key has a passphrase and you don't want to enter the passphrase every time you use the key, you can add your key to the SSH agent. The SSH agent manages your SSH keys and remembers your passphrase.

If you don't already have an SSH key, you must generate a new SSH key to use for authentication. If you're unsure whether you already have an SSH key, you can check for existing keys. For more information, see "Checking for existing SSH keys."

If you want to use a hardware security key to authenticate to GitHub, you must generate a new SSH

# What did we just do?

1. First we generate the SSH key
    a. This creates two keys: private key (**never share it**) and a public key (that one you'll share with GitHub)
    b. You'll be asked to enter a passphrase, make sure it's not easily guessable, but also something that you'll remember
2. Adding the SSH key to the ssh-agent
    a. Ssh-agent is a security guard program on your computer for managing keys
    b. Adding the private key tells the ssh-agent tells it to use it for locking/unlocking things automatically

Private https://docs.github.com/en/authentication/connecting-to-github-with-ssh

GitHub Docs Version: Free, Pro, & Team Search GitHub Docs Sign up

Authentication / Connect with SSH

Authentication /

# Connecting to GitHub with SSH

You can connect to GitHub using the Secure Shell Protocol (SSH), which provides a secure channel over an unsecured network.



## Help and support

**Did this doc help you?**

**Help us make these docs great!**

All GitHub docs are open source. See something

**Still need help?**

Ask the GitHub community

https://docs.github.com/en/authentication/connecting-to-github-with-ssh

# Basics of working with Git

# What is a Git repository?

Git repository is 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.

# Cloning a repository

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

You can create a complete copy of someone else's Git repository on your computer using the **git clone** command.

# Initializing a repository

```
$ git init .
```

Note the dot at the end: in Linux (and other UNIX-like operating systems), that represents the current directory in the filesystem.

You can use this command to create a new Git repository on your system.

# Basic flow

```
$ git add [files|directories]
$ git commit -m “...”
```

First command adds all files/directories to staging.

Second command commits staged content as a new commit snapshot.

# Staging area vs 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.

# 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.

# 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.

# Pushing and pulling

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

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

# Let’s get to work!

# What will you do?

1. Divide into groups
2. One person in each group should create a new GitHub repository
3. Invite your team members as collaborators
4. Let everyone clone the repository
5. Everyone should checkout a new branch with their name
6. Create a file, write something nice about yourself!
7. Push it
8. Create a pull request
9. Merge all pull requests

# Good-to-know stuff

# Using a GUI client

Using a graphical user interface can be beneficial (easier to visualize all the branches, commits, their history, etc.)

You can install it as a separate piece of software (for example GitKraken), or have it as part of your Integrated Development Environment (IDE).

Workspace / Malo Mart | electron | support.gitkraken.com | vscode-qa | vscode-qa | Personal

workspace | repository: electron | branch: 17-x-y

Undo | Redo | Pull | Push | Branch | Stash | Pop | Terminal

Viewing 1800/1800 | Show All

Filter (⌘ + Option + f)

LOCAL 2/2
- 17-x-y
- main

REMOTE 188/188

PULL REQUESTS 4

82
- My Pull Requests 0
- Assigned To Me 0
- Awaiting My Review 0
- All Pull Requests 4
- On Hold 2
- origin
  - #32082 fix: gtk_native_dialog_ru...
  - #28288 build: add boringssl hea...

JIRA ISSUES 0

TEAMS

Team: Design Team
- Jonathan Silva @jonathans
- @jsilvafour
  - electron 17-x-y +2 -2
    - . /package.json +1 -1
    - spec/static /main.js +1 -1
- @kraken8
- Keif Kraken @keif1

TAGS 1610/1610

SUBMODULES 0

GITHUB ACTIONS 0

| BRANCH / TAG | COMMIT MESSAGE | |
|---|---|---|
| | // WIP (3 + 1) | 3 hours ago |
| main +1 | Bump v18.0.0-nightly.20211202 | |
| roller/chromiu... | chore: bump chromium in DEPS to 98.0.4742.0 | |
| roller/chromiu... | chore: bump chromium in DEPS to 98.0.4742.0 | |
| roller/node/main | chore: bump node in DEPS to v16.13.1 | |
| trop/16-x-y-bp-fi... | fix: gtk_native_dialog_run() calls show() internally | 12 hours ago |
| trop/17-x-y-bp-fi... | fix: gtk_native_dialog_run() calls show() internally | |
| trop/15-x-y-bp-fi... | fix: gtk_native_dialog_run() calls show() internally | |
| trop/14-x-y-bp-fi... | fix: gtk_native_dialog_run() calls show() internally | |
| | fix: gtk_native_dialog_run() calls show() internally (#32049) | |
| child-window-pr... | test: remove "nativeWindowOpen: false" from renderer tests | |
| 17-x-y | fix: avoid double call in OnRefreshComplete on aura platforms (#32070) | |
| v18.0.0-nightly.202... | Bump v18.0.0-nightly.20211201 | |
| | chore: bump chromium in DEPS to 98.0.4740.0 | |
| | chore: bump chromium in DEPS to 98.0.4740.0 | |
| | fix: window.open not overriding parent's webPreferences | yesterday |
| v13.6.3 +1 | Bump v13.6.3 | |
| | fix: avoid double call in OnRefreshComplete on aura platforms (#32052) | |
| | build: add CI path-filtering for docs-only changes (#32054) | |
| cherry-pick/13-x... | Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/1fcfb942bd | |
| cherry-pick/13-x... | Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/1a8af2da50e4 | |
| cherry-pick/13-x... | Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/f781748dcb3c | |
| cherry-pick/13-x... | Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/855df1837e | |
| cherry-pick/13-x... | Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/f0a63e1f361f | |
| cherry-pick/13-x... | Merge branch '13-x-y' into cherry-pick/13-x-y/chromium/27eb11a28555 | |
| | chore: cherry-pick a5f54612590d from chromium (#31901) | |
| | fix: generate valid config.gypi (#31989) | |

4 file changes on 17-x-y

Path | Tree

Unstaged Files (2) — Stage all changes

Collapse All
- spec
  - static
    - main.js
- ELECTRON_VERSION

Staged Files (2) — Unstage all changes
- mkdocs.yml
- package.json

Commit Message — Amend

Summary

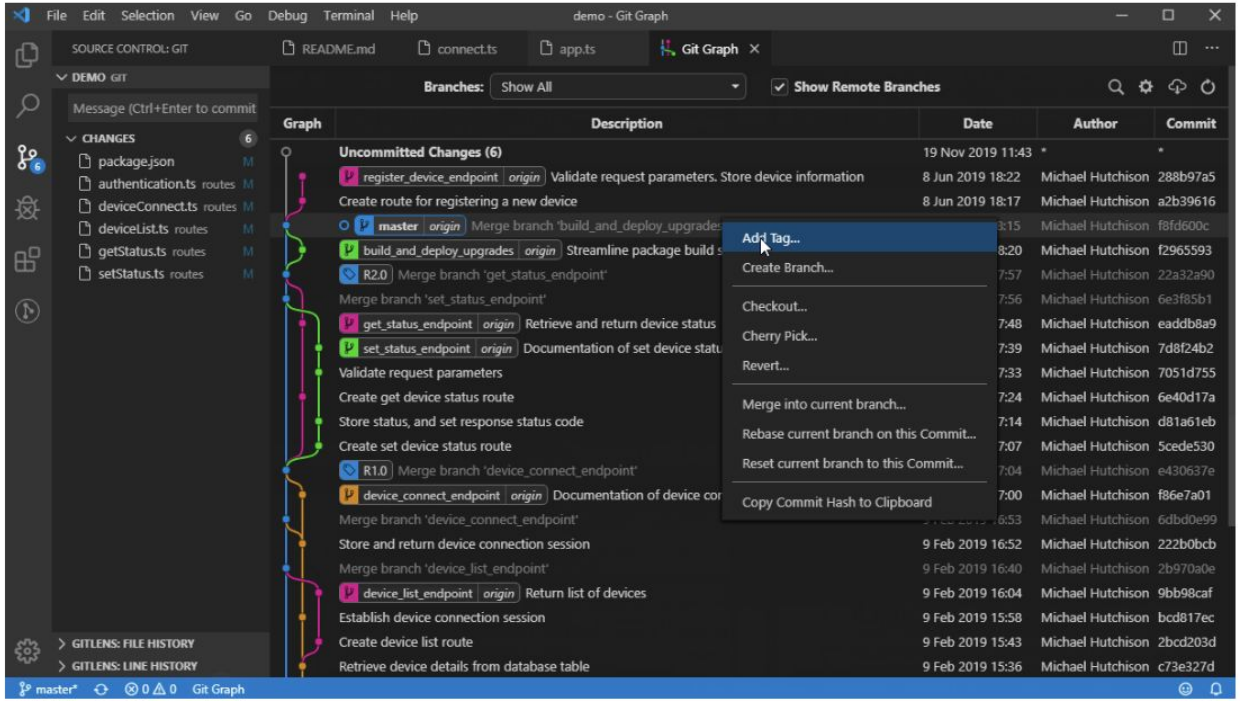
Description

Type a message to commit

120% | Feedback | PRO | 8.2.0-RC14

# Git Graph extension for Visual Studio Code

View a Git Graph of your repository, and easily perform Git actions from the graph. Configurable to look the way you want!



# Meaning of .gitignore

Certain files are not supposed to be tracked. For example:

- 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

Code Blame

```
# IDEs
.idea
.vscode

# Python & virtual environment
__pycache__
venv

# OS
.DS_Store
Thumbs.db

# Maps
maps
```