

# AGENDA

**1. Operating Systems and Linux**

**2. Operating Systems and Linux**

**3. Version Control Systems: Git & GitHub**

I

# Operating Systems and Linux

# Course outline:

1. Operating Systems
2. Linux
3. Linux vs Windows
4. Installation of Linux
5. File Systems
6. File operations
7. Creating bash scripts
8. Usage of packages
9. Managing files. File permissions
10. Scheduled automation using cron

# Operating Systems

- An **operating system (OS)** is the program that, after being initially loaded into the computer by a boot program, manages all of the other application programs in a computer.
- The application programs make use of the operating system by making requests for services through a defined application program interface (**API**). In addition, users can interact directly with the operating system through a user interface such as a command line or a graphical user interface (**GUI**).

# Linux

- As an operating system, Linux is software that sits underneath all of the other software on a computer, receiving requests from those programs and relaying these requests to the computer's hardware.



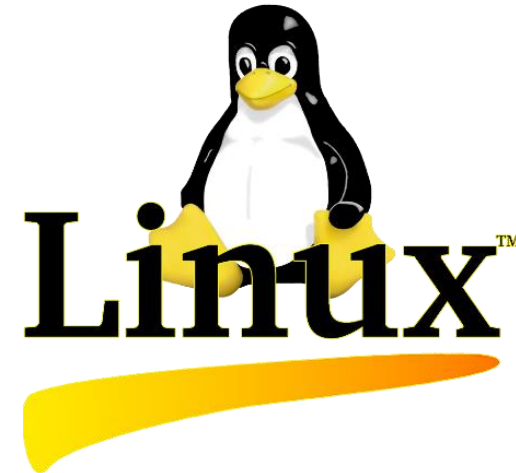
## Linux

Recommended for large servers, Big data analysis, Artificial intelligence processes.

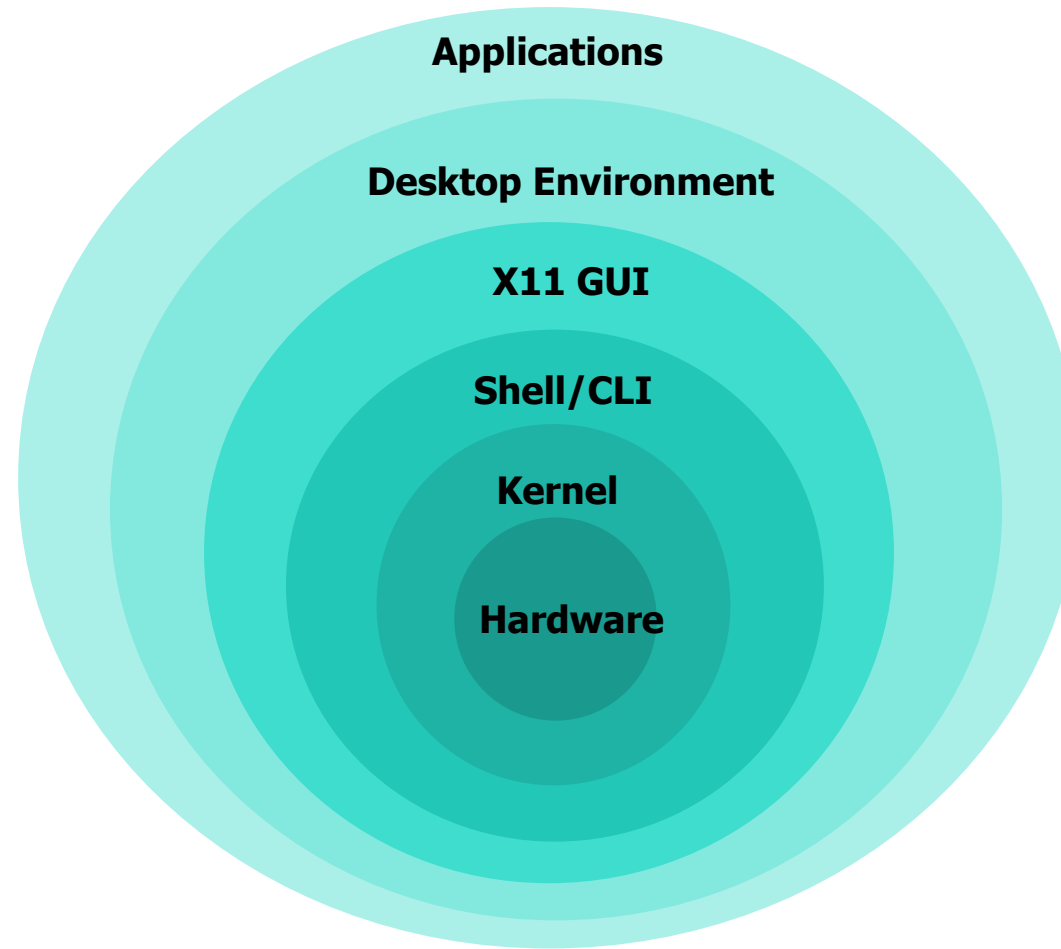


## Difference

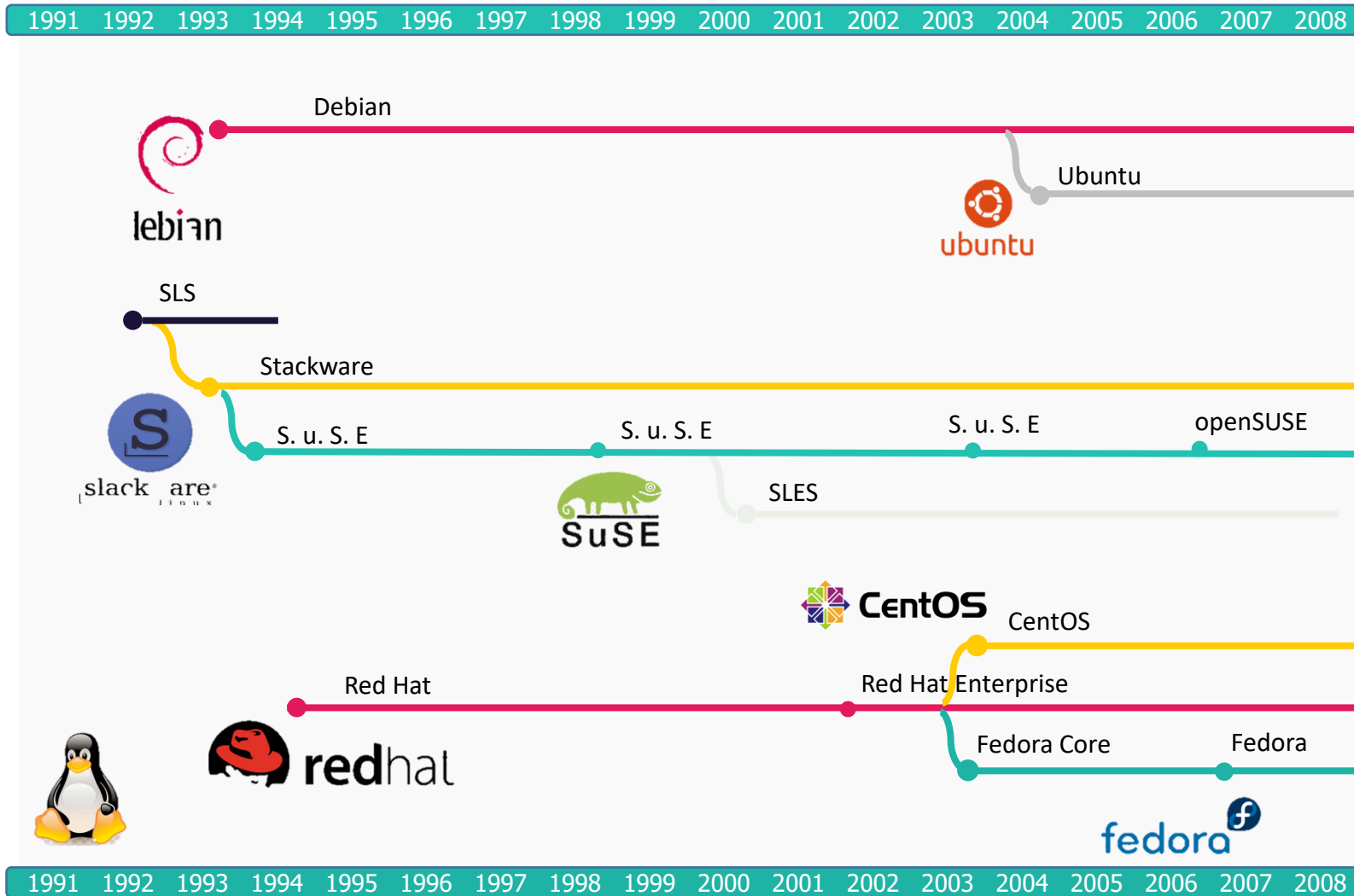
Linux can be run on the same device as other operating systems. It is an open source, free, secure operating system.



# Full Linux system diagram



# Main Linux distributions



# Linux vs Windows

## Linux

- The Linux kernel, and the GNU utilities and libraries which accompany it in most distributions, are entirely free and open source.
- More recent distributions of Linux are easier to use than previous variants.
- It has a strong focus on process management, system security, and uptime.
- There are thousands of programs available for Linux, and many are available as easy-to-install software packages — all for free.
- Linux is used by corporate, scientific, and academic organizations of every size.

## Windows

- Microsoft Windows usually costs between \$99.00 and \$199.00 USD for each licensed copy.
- One of its primary design characteristics is user friendliness and simplicity of basic system tasks.
- Many of the sacrifices it makes in the name of user-friendliness can lead to security vulnerabilities and system instability.
- Windows commands the highest number of desktop users, and therefore the largest selection of software.
- Microsoft Windows is choice for gamers, novice users, and business users who rely on Microsoft software.



# Installation of Linux



Download Ubuntu:

<https://ubuntu.com/download/desktop>



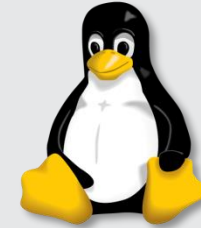
Download Virtual Box:

<https://www.virtualbox.org/wiki/Downloads>



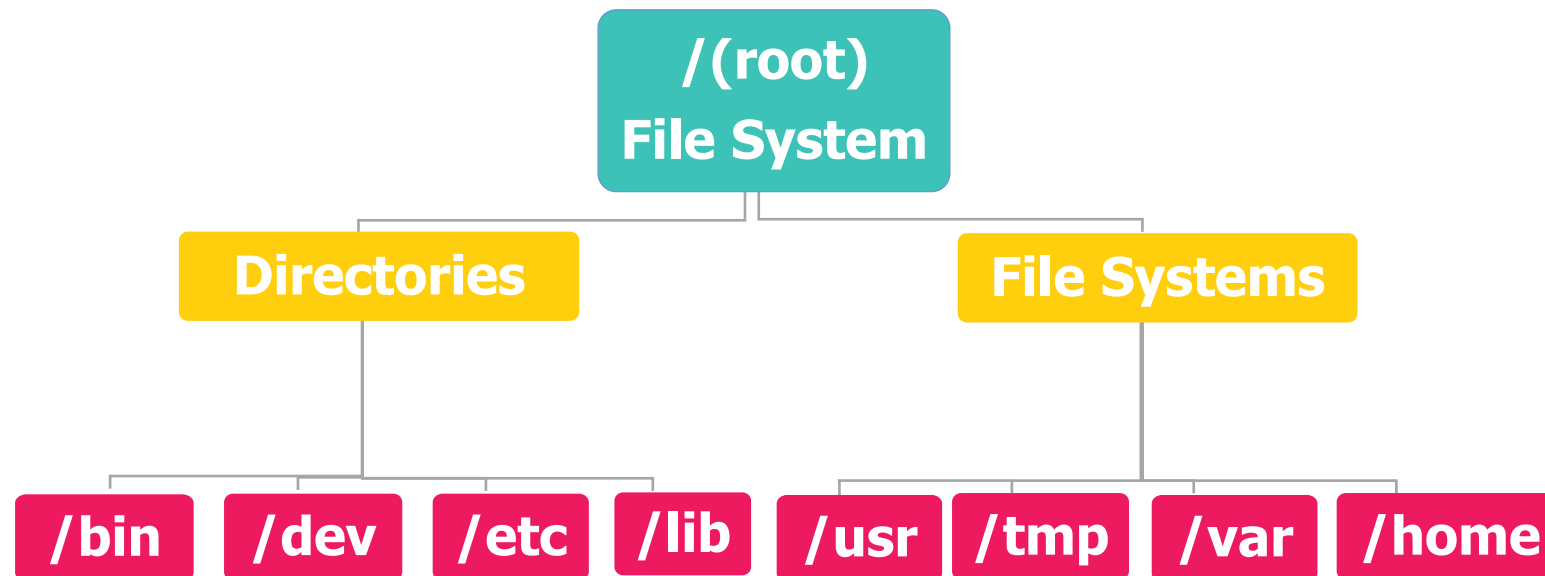
Download Docker:

<https://www.docker.com/products/docker-desktop>



# File systems

- A file system is a process that manages how and where data on a storage disk, typically a hard disk drive (HDD), is stored, accessed and managed. It is a logical disk component that manages a disk's internal operations as it relates to a computer and is abstract to a human user.



# The Linux Command Line - Linux terminal

The Linux command line is a text interface to your computer. Using Linux terminal, user can manipulate files, folders, users, programs, etc.

- **lala@hp\_ubuntu**

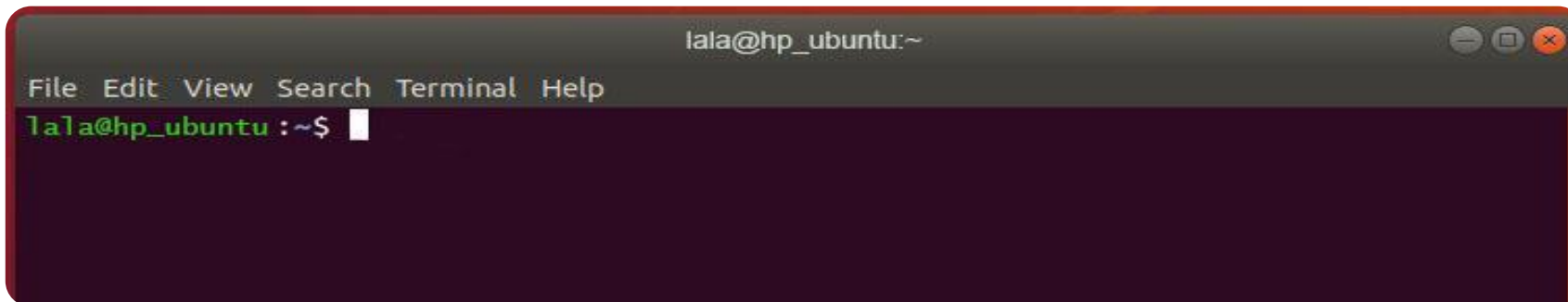


computer's name

user's name

- **\$** is a sign of shell\*
- **~** (tilde) is an abbreviation of `/home/user` folder

**Note:** \*shell provides you with an interface to the system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.



# Navigating File Systems

**A path** is a unique location to a file or a folder in a file system

- Relative path → defined as path related to the present working directory (pwd).
- Absolute path → defined as the specifying the location of a file or directory from the root directory (/).

## Relative path

```
qss@qss:~$ ls
Desktop      Music      rstudio-server-1.2.5019-amd64.deb  Videos
Documents   Pictures   rstudio-server-1.2.5019-amd64.deb.1
Downloads    Public     snap
examples.desktop  R          Templates
```

## Absolute path

```
qss@qss:~$ ls /home/qss/
Desktop      Music      rstudio-server-1.2.5019-amd64.deb  Videos
Documents   Pictures   rstudio-server-1.2.5019-amd64.deb.1
Downloads    Public     snap
examples.desktop  R          Templates
```

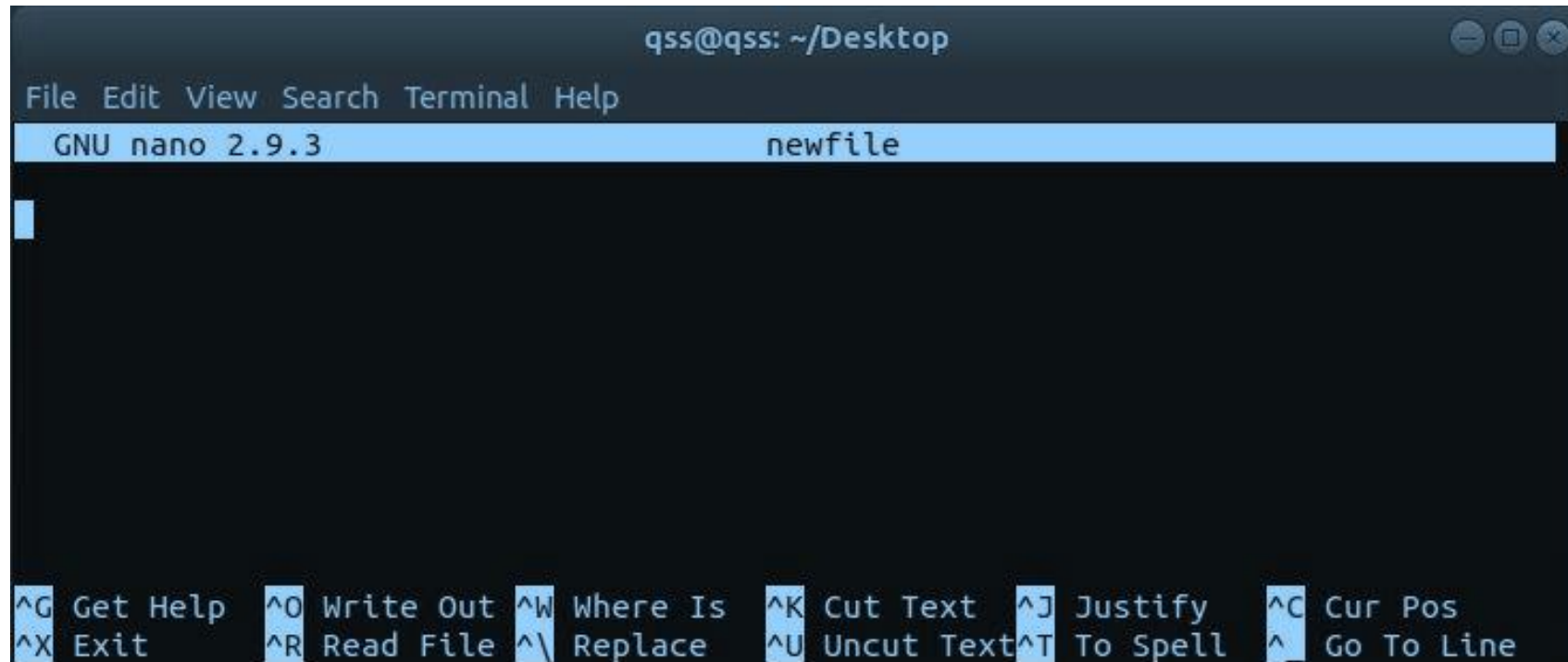
# Navigating File Systems

- Each directory has minimum of 2 directory which are hidden ". " and " .. " directories

Symbol	Description
.	This directory
..	The parent directory
/	Directory separator.

# Editing and viewing files

- **GNU nano** is a text editor for **Unix-like** computing systems or operating environments using a command line interface.
- Start nano: **nano *newfilename***



The screenshot shows the GNU nano 2.9.3 text editor running in a terminal window. The window title is "qss@qss: ~/Desktop". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The status bar at the top shows "GNU nano 2.9.3" and "newfile". The main editing area is empty with a cursor at the first line. The bottom status bar displays various keyboard shortcuts: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, ^X Exit, ^R Read File, ^\ Replace, ^U Uncut Text, ^T To Spell, and ^\_ Go To Line.

II

# Operating Systems and Linux

# Creating bash scripts

- Using linux terminal and bash commands (commands / codes that run in terminal), you can create bash scripts, ending with .sh

Step 1: create .sh file using nano text editor

>>> nano sample\_bash\_script.sh

Step 2: add #!/bin/bash to the first line

Step 3: add your bash scripts, then save your file

Step 4: run your file using bash command

>>> bash sample\_bash\_script.sh

```
GNU nano 4.8 sample_bash_script.sh
#!/bin/bash/
ls
echo "Hello, there"
date
```

```
root@a099579ec593:~/Desktop# bash sample_bash_script.sh
GPL-3 abc sample_bash_script.sh
Hello, there
Mon Mar  1 00:29:09 +04 2021
```



# Usage of packages

- To do some tasks, to use some programmes, data scientists may need other functionalities
- In Linux, data scientists can install new packages using package managers
- Depending on distribution and linux kernel, package managers may vary
- Package managers are responsible for installing, removing, updating packages.

# File ownership and permissions

- Sometimes there can be some problems with file ownership, especially when installing and configuring some softwares.

Part	Description
1	Shows whether it is directory or file. "d" means directory, "-" means file.
2	Permissions of owner: <b>rw</b> x read- <b>w</b> rite- <b>e</b> xecute
3	Permissions of members of the owner group : r-x read and <b>e</b> xecute
4	Permissions of other users : r-x read and execute

d	rwX	r-X	r-X
1	2	3	
4			

**Example:** contents of a directory

```
root@a099579ec593:~/Desktop# ls -al
total 48
drwxr-xr-x 3 root root 4096 Feb 28 23:37 .
drwx----- 1 root root 4096 Feb 28 22:55 ..
-rw-r--r-- 1 root root 35147 Feb 28 07:48 GPL-3
drwxr-xr-x 2 root root 4096 Feb 28 11:32 abc
```

# Changing file permissions

Code	Description
<b>chmod +rwx filename</b>	Give all permissions to everyone
<b>chmod -rwx filename</b>	Remove all permissions from everyone
<b>chmod ugo+rwx foldername</b>	To give read, write, and execute to everyone. (u – users / g – group / o –others)
<b>chown username filename</b>	Make user owner of file

# Changing file permissions

Code	Description
chmod 0 filename	Remove all permissions from everyone
chmod 1 filename	Give executable permission for everyone
chmod 2 foldername	To give write permission to everyone.
chown 4 filename	To give read permission to everyone.
chmod 777 foldername	Give read, write, and execute permissions for everyone
chmod 700 foldername	Give read, write, and execute permissions for the user only
chmod 327 foldername	Give write and execute (3) permission for the user, w (2) for the group, and read, write, and execute for the users.

Permission numbers are:

- 0 = ---
- 1 = --x
- 2 = -w-
- 3 = -wx
- 4 = r-
- 5 = r-x
- 6 = rw-
- 7 = rwx

# Scheduled automation

- Crontab's scheduled automation enables data scientist/data engineers to automate scheduled tasks such as model updating, backup data, etc.
- For instance, given crontab implies that everyday, at 10:00 AM given command will run.

```
# For more information see the manual pages of crontab(5) and cron(8)
#
#      m      h      dom      mon      dow      command
#      0      10     *        *        *        python3 /home/user/modelling/train_model.py
```

Abbreviation	Description	Allowed values
m	Minute	0-59
h	Hour	0-23
dom	Day of month	1-31
mon	Month	1-12 or JAN-DEC
dow	Day of week	0-6 or MON-SUN

# Scheduled automation

Abbreviation	Description
*	Any value
,	Value separator
-	Range of values
/	Step values



**git + GitHub**



Data Science  
Academy

III

# Version Control Systems: Git & GitHub

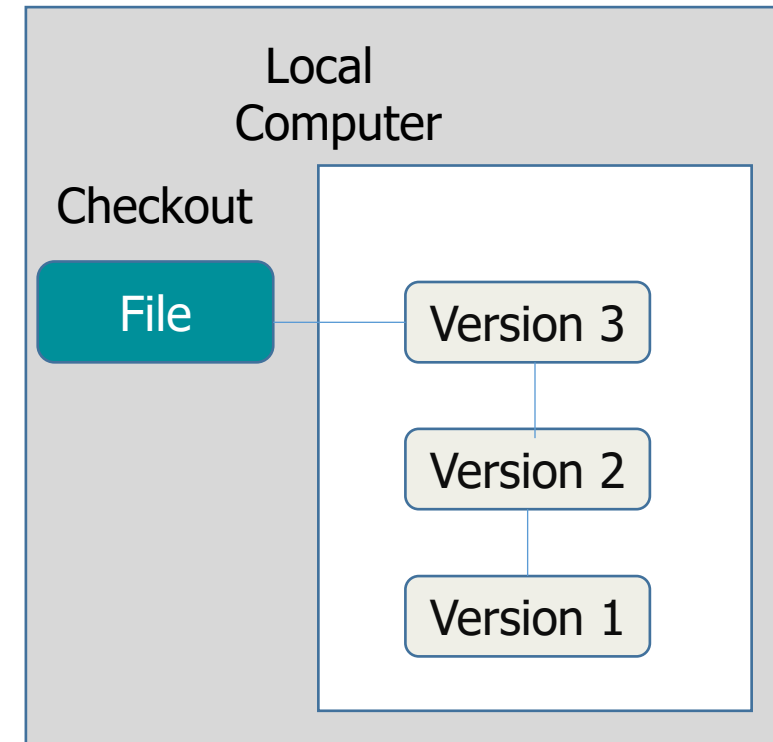


# About Version Control

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. You will use software source code as the files being version controlled, though in reality you can do this with nearly any type of file on a computer.

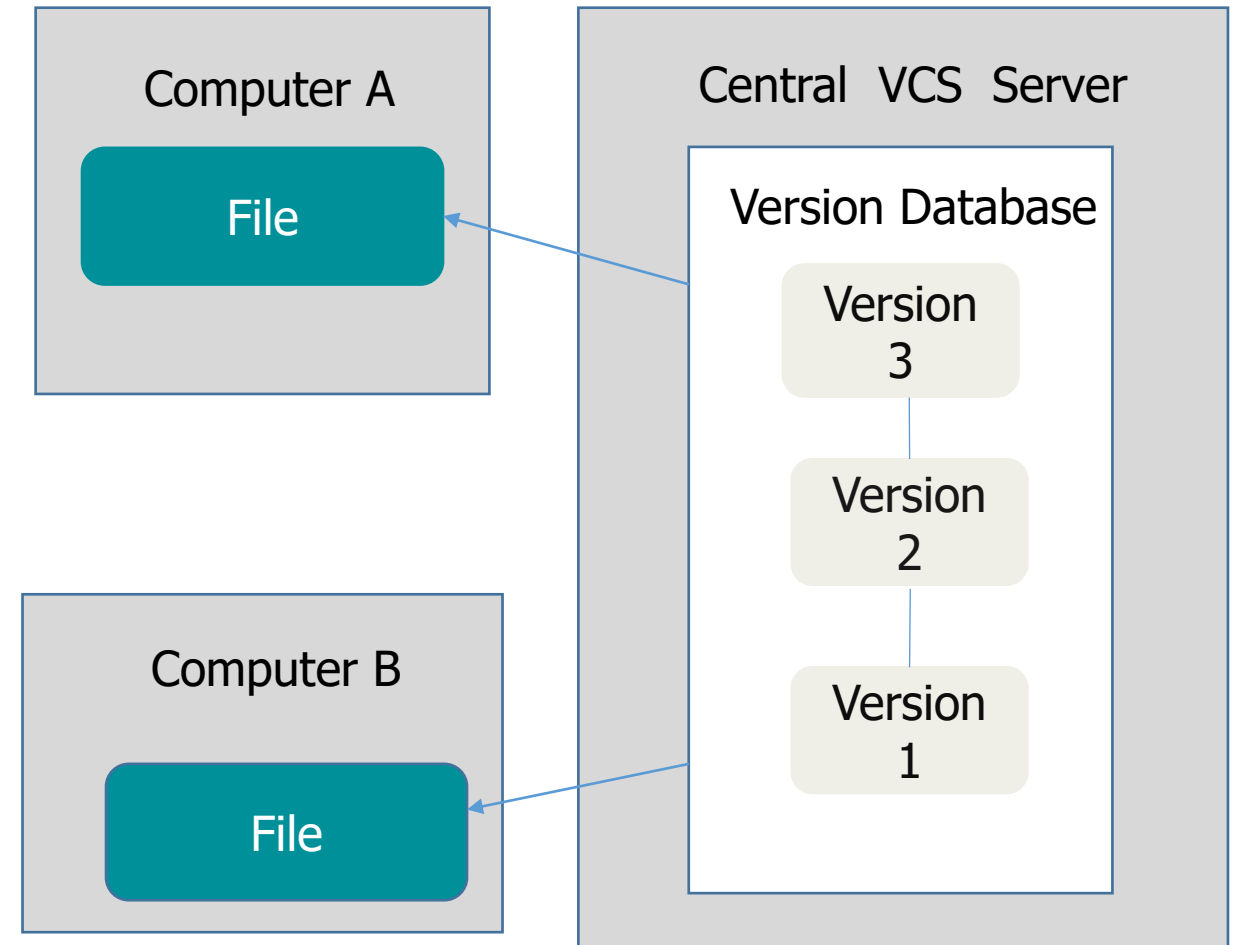
# Local Version Control Systems

- Many people's version-control method of choice is to copy files into another directory (perhaps a time-stamped directory, if they're clever). This approach is very common because it is so simple, but it is also incredibly error prone. It is easy to forget which directory you're in and accidentally write to the wrong file or copy over files you don't mean to.



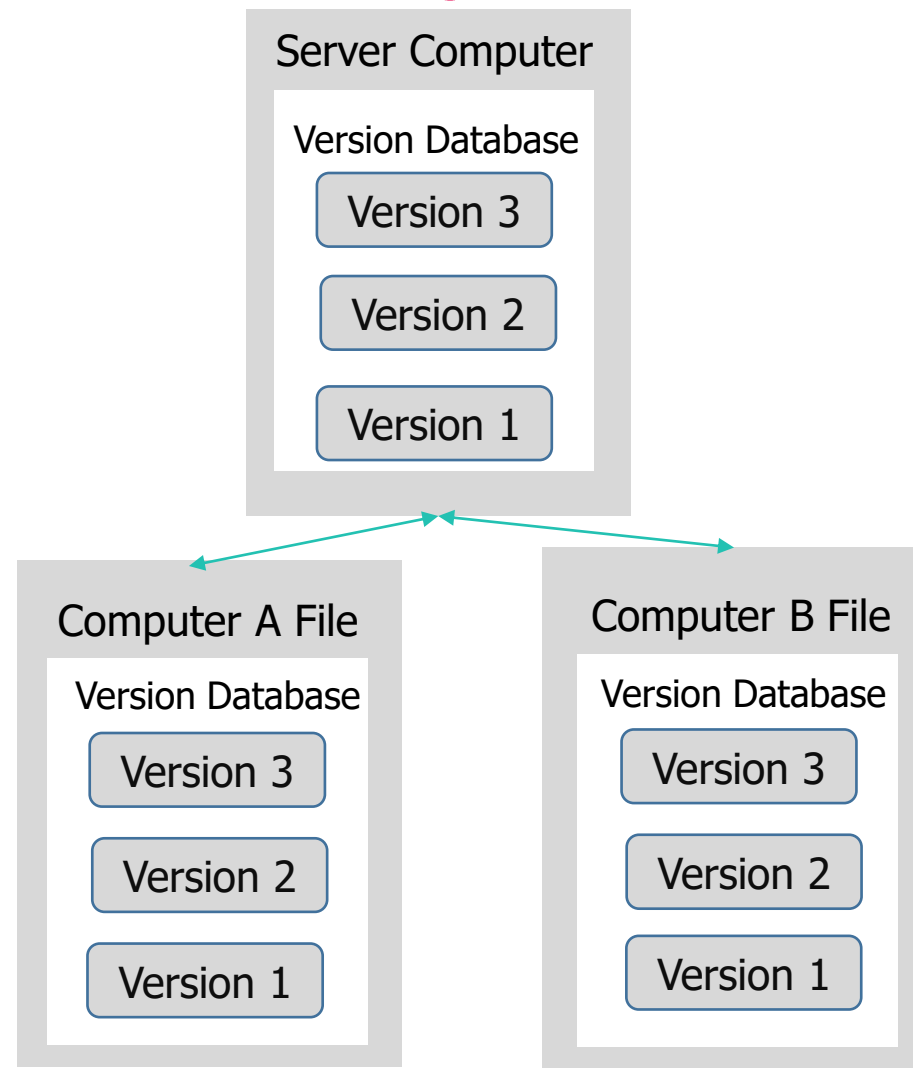
# Centralized Version Control Systems

- When we need to collaborate with developers on other systems. To deal with this problem, Centralized Version Control Systems (CVCSs) were developed. These systems (such as CVS, Subversion, and Perforce) have a single server that contains all the versioned files, and a number of clients that check out files from that central place.



# Distributed Version Control Systems

- In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients don't just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history. Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.



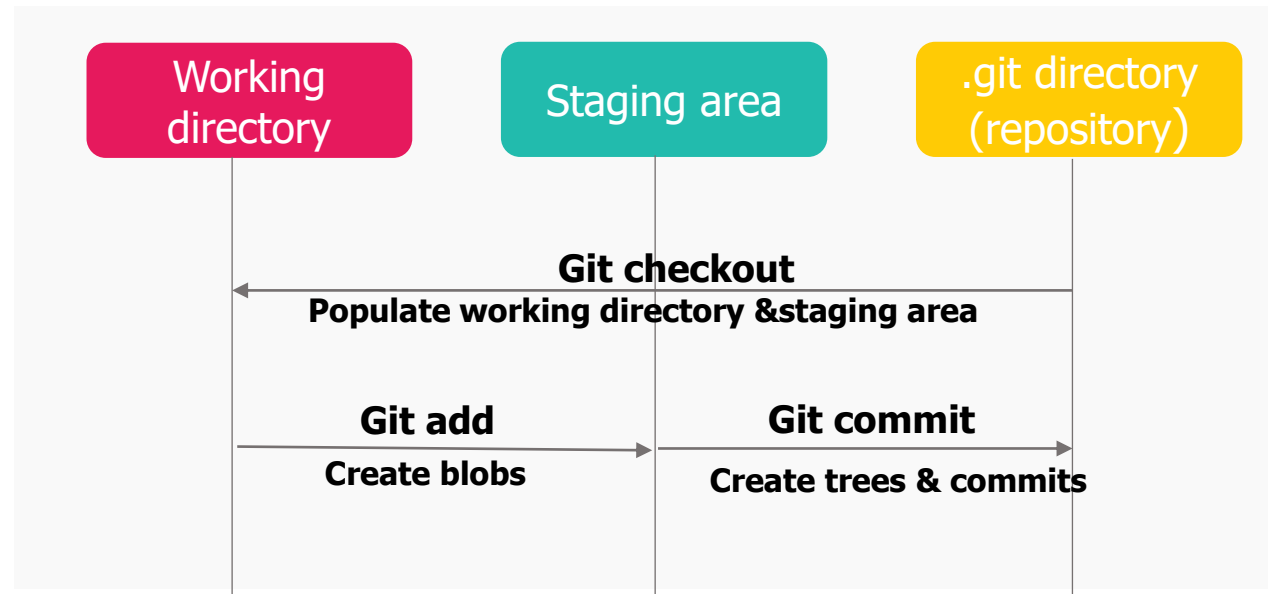
# Git

- Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
  - Git is a free and open-source version control system, originally created by Linus Torvalds in 2005.
  - Git is the most widely used version control system in the world today and is considered the modern standard for software development.
  - Git also has excellent support for branching, merging, and rewriting repository history, which has led to many innovative and powerful workflows and tools.

# The Three local States

Git has three main states that your files can reside in: modified, staged, and committed:

- Modified means that you have changed the file but have not committed it to your database yet.
- Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
- Committed means that the data is safely stored in your local database.



# Installation of Git



**Install Git:**

<https://git-scm.com/downloads>

**Git for Mac:**

<https://sourceforge.net/projects/git-osx-installer/files/>



**Open a terminal and verify installation by typing:**

**git --version**



GitLab



# Configure Git

- Configure the author name and email address to be used with your commits:

**git config --global user.name "Sam Smith"**

**git config --global user.email sam@example.com**

- Creating new folder: **mkdir project\_name**
- Directing git to that folder: **cd project\_name**
- Listing files in folder: **ls**
- Directory path: **pwd**
- Initializes .git repo in current folder: **git init .**



# Initialization and basics

- View the status of your files in the working directory and staging area: **git status**
- Adds file with the given name contents to the staging area: **git add file\_name**
- Adds all modified files to the staging area: **git add .**
- Commits modifications: **git commit -m "commit message "**
- Allows to write multiple line messages: **git commit**

**Note:** In Git, "add" means "add to staging area" so it will be part of the next commit.

# Useful tips

- Updating previous commit using --amend:

**git commit -m "initial commit"**

**git add forgotten\_file**

**git commit --amend**

- Deleting files from repository:

**git rm file name**

**git commit -m " file name "**

# GitHub

- GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.

<https://github.com>



# Useful Definitions

**Repository** - contains all of your project's files and stores each file's revision history.

**Remote repository** - A remote in Git is a common repository that all team members use to exchange their changes.

**Local repository** - is on your computer and has all the files and their commit history, enabling full diffs, history review, and committing when offline.

**Branch** - based workflow that supports teams and projects where deployments are made regularly.

**Master** - is a naming convention for a branch. Master can be seen as a "default" branch of repository.

**Push** - sends the recent commit history from your local repository up to GitHub.

**Pull** - grabs any changes from the GitHub repository and merges them into your local repository.

**Clone** - downloading a copy of the source code from source control.

**Fork** - is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

**Fetch** - will download the remote content but not update your local repo's working state, leaving your current work intact.





# Pushing repository to GitHub

- Pushing changes from master branch of local repository to master branch of remote repository:

**git remote add origin [https://github.com/UserName/directory\\_name.git](https://github.com/UserName/directory_name.git)**

- Adding project to github:

**git push -u origin master**

 Aytantabriz Update Advanced data preprocessing.R		Latest commit b13d68a 10 hours ago
 <a href="#">Advanced data preprocessing.R</a>	Update Advanced data preprocessing.R	10 hours ago
 <a href="#">EDA.R</a>	Ammendi yoxlayiriq	11 hours ago
 <a href="#">Join data.R</a>	Ammendi yoxlayiriq	11 hours ago

# Cloning

- Cloning is a process of creating an identical copy of a Git Remote Repository to the local machine. When we clone a repository, all the files are downloaded to the local machine but the remote git repository remains unchanged. Making changes and committing them on your local repository (cloned repository) will not affect the remote repository that you cloned in any way. These changes made on the local machine can be synced with the remote repository anytime the user wants.

**git clone [https://github.com/user\\_name/repository.git](https://github.com/user_name/repository.git)**

- Or we can clone project in Github itself:

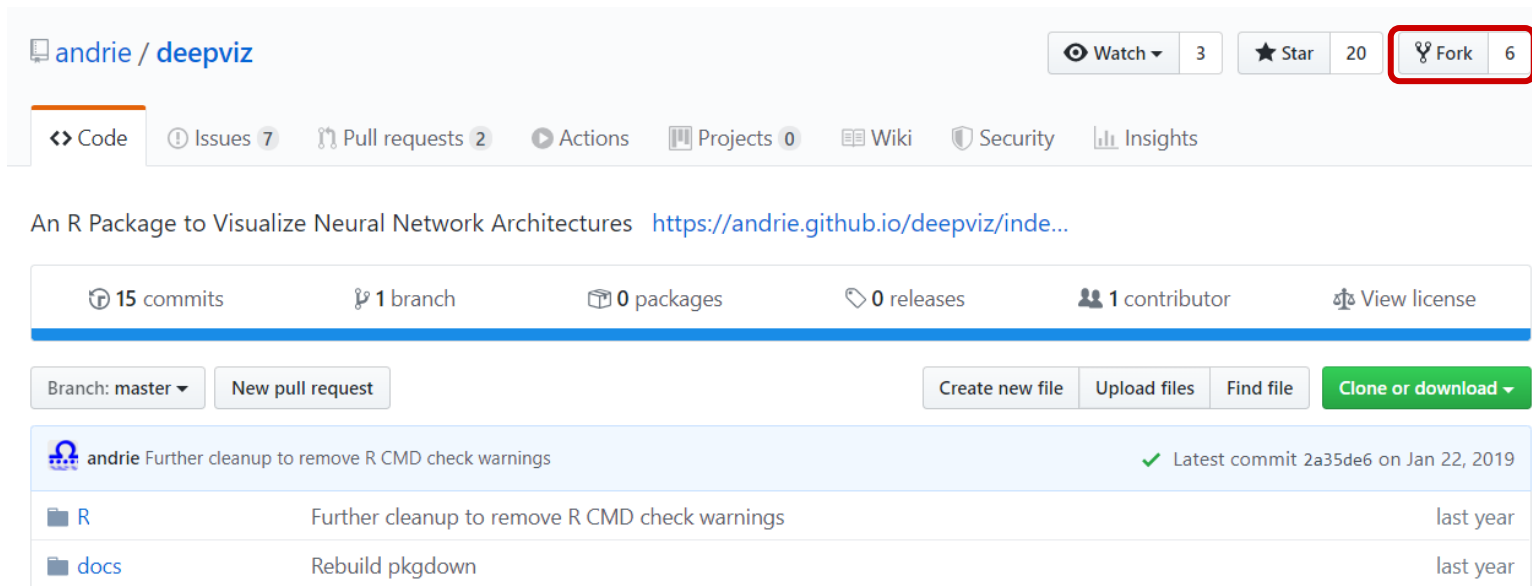


# Fetch and pull changes

- ***git push***: in order to push changes to remote repo
- Downloads commits, files, and refs from a remote repository into your local repo: **git fetch**
- Combine multiple sequences of commits into one unified history. In the most frequent use cases, git merge is used to combine two branches: **git merge**
- Used to fetch and download content from a remote repository and immediately update the local repository to match that content: **git pull = git fetch + git merge**

# Contributing To A Project : Forking Projects

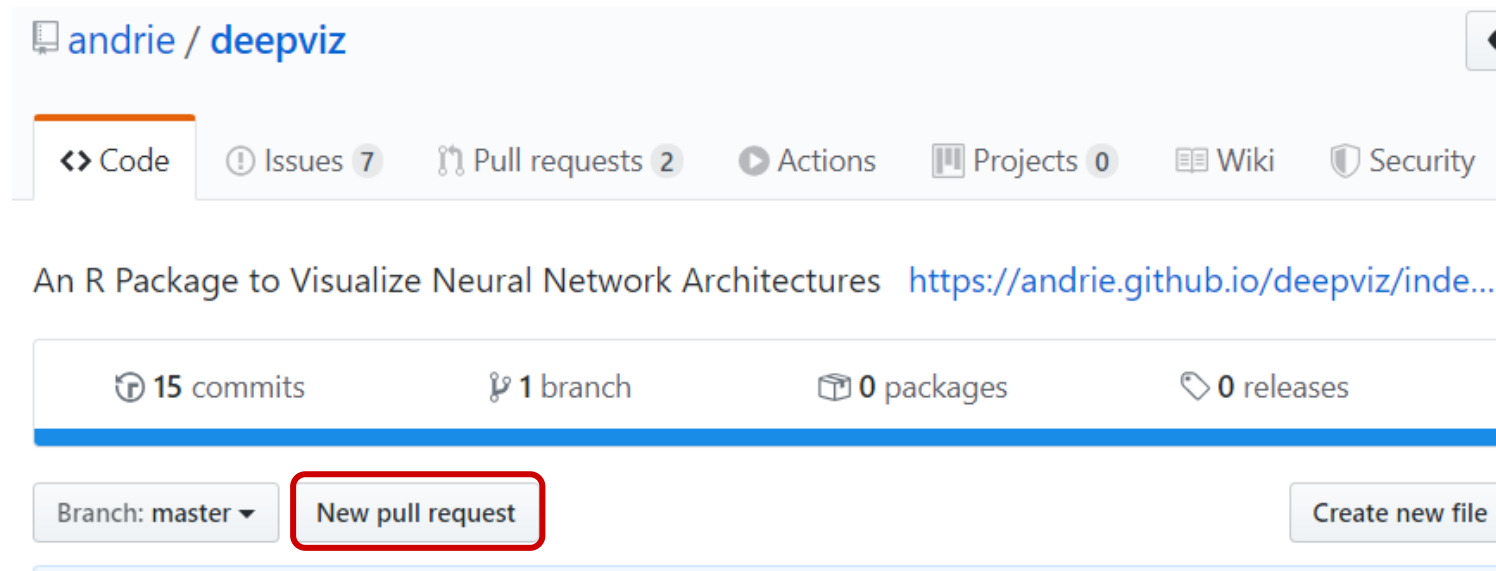
- If you want to contribute to an existing project to which you don't have push access, you can "fork" the project. When you "fork" a project, GitHub will make a copy of the project that is entirely yours; it lives in your namespace, and you can push to it.





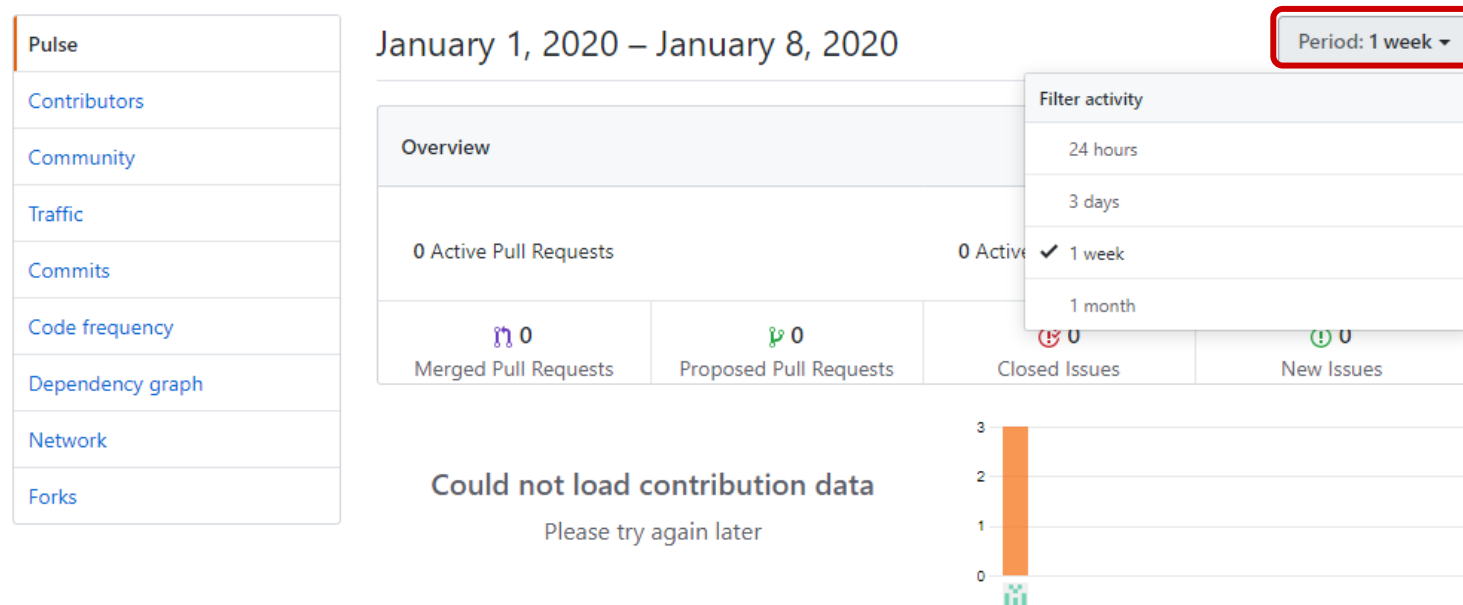
# PULL request

- Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.



# PULSE

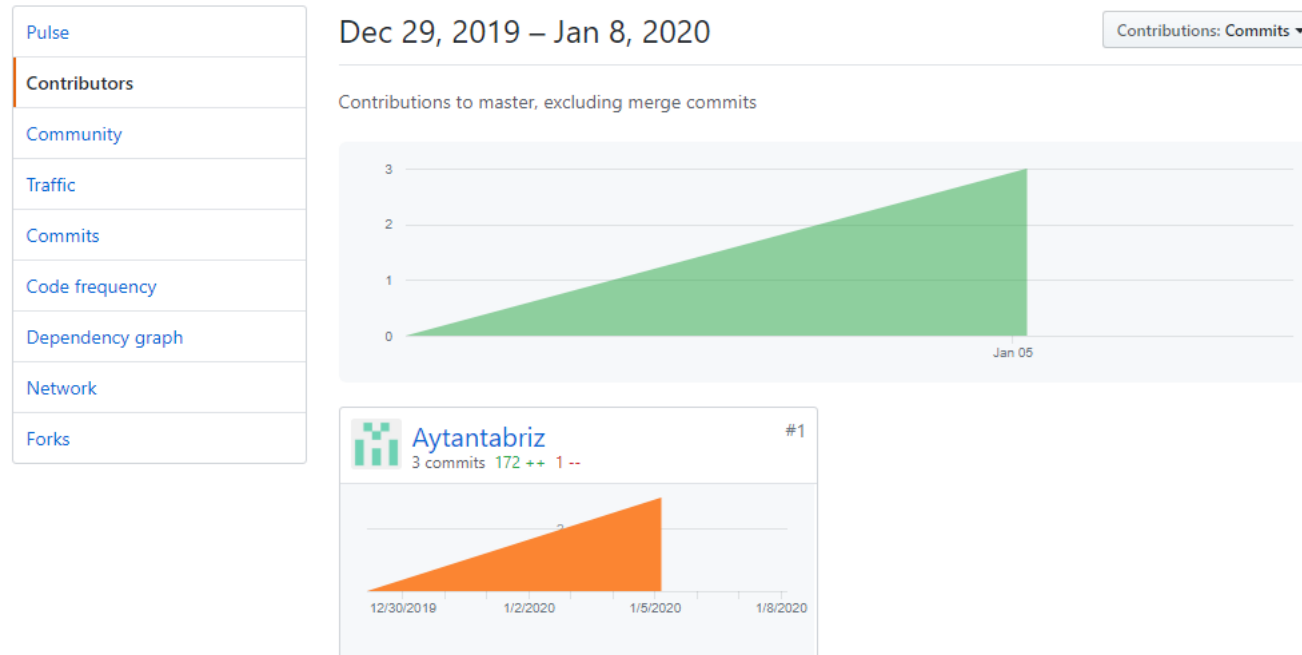
- Pulse is an overview of a repository's activity. It includes a list of open and merged pull requests, open and closed issues, and a graph showing the commit activity for the top 15 users who committed to the default branch of the project in the selected **time period**.



- By default, Pulse shows the last seven days of repository activity. To choose a different time period, click the **Period** dropdown in the upper-right corner of the Pulse overview.

# Graphs

- A contributor is someone from the outside not on the **core development team** of the project that wants to contribute some changes to a project. You can view the top 100 contributors to a repository in the contributors graph. Merge commits and empty commits aren't counted as contributions for this graph.



- Optionally, to view contributors during a specific time period, click, then drag until the time period is selected.

# Thank you!