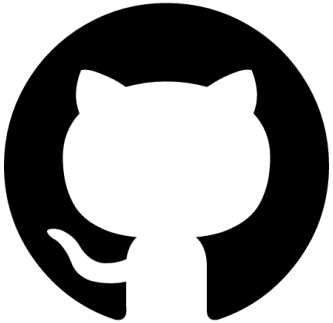
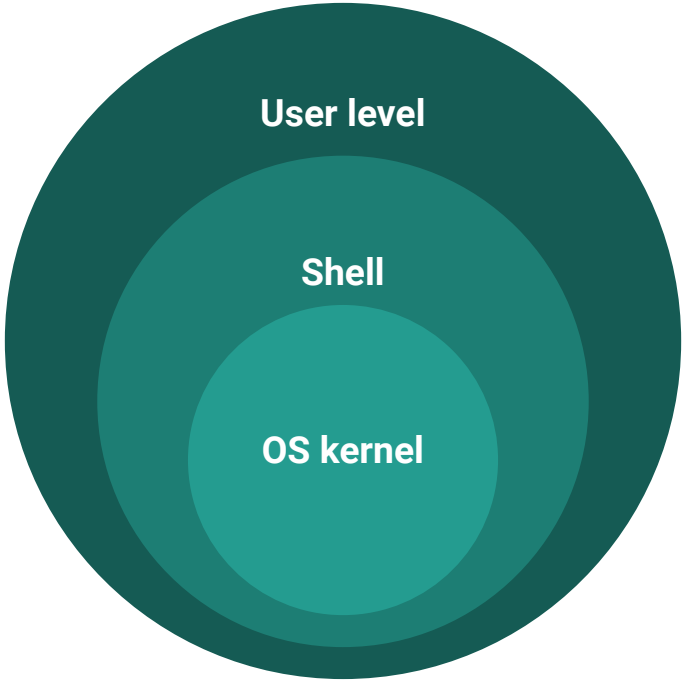
Basics of **Linux** & **Git**



What is a Shell?



Operating system:
Manages the execution of all processes on the hardware (microprocessor)

Shell:
Command line interface between the user and the OS

Terminal:
Console for using commands in shell script in Linux.

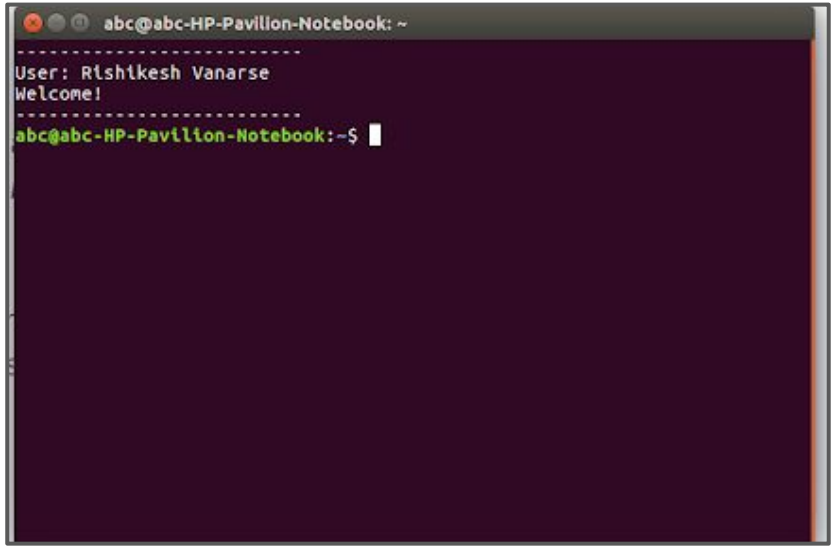
BASH (**B**ourne **A**gain **SH**ell)

- Modified shell script for UNIX written by Steve Bourne
- Used to interact with the operating system through scripts
- Can be used through:
  - Terminal
  - .bash files

Linux Terminal

- Interprets Shell commands *(similar to a python interpreter interpreting python commands)*

- All commands typed into the terminal are Shell script commands



Basic Shell commands

- **pwd** *(present working directory)*

- **ls** *(list)*
  - ls -l

- **cd <*name_of_folder*>** *(Change Directory)*
  - cd ..
  - cd ~
  - cd /
  - cd <*path_to_folder*>

- **sudo <*command*>** *(Super-User Do)*

More shell commands

- **mkdir <*name_of_folder*>** *(Make Directory)*
- **gedit <*filename*>** *(Create/Open text file)*
- **./<*filename*>** *(Run a file from the current directory)*
- **echo <*text*>** *(Same as print)*
- **man <*command*>** *(Manual of the command)*

Other utilities

- ***<command> && <command>***
  (Execute both commands one after another)
- **clear**
- **exit**

Shortcuts:

Ctrl + Alt + T	*Open new terminal*
Ctrl + C	*Stop running process*
Tab	*Autocomplete*
UP Arrow Key	*Previous command*
Ctrl + Shift + C / ***Ctrl + Shift + V***	*Copy/Paste*

Installing Linux Packages:

1. sudo apt-get update
   (updates existing packages)

2. sudo apt-get install <*package_name*>

Installing python libraries:

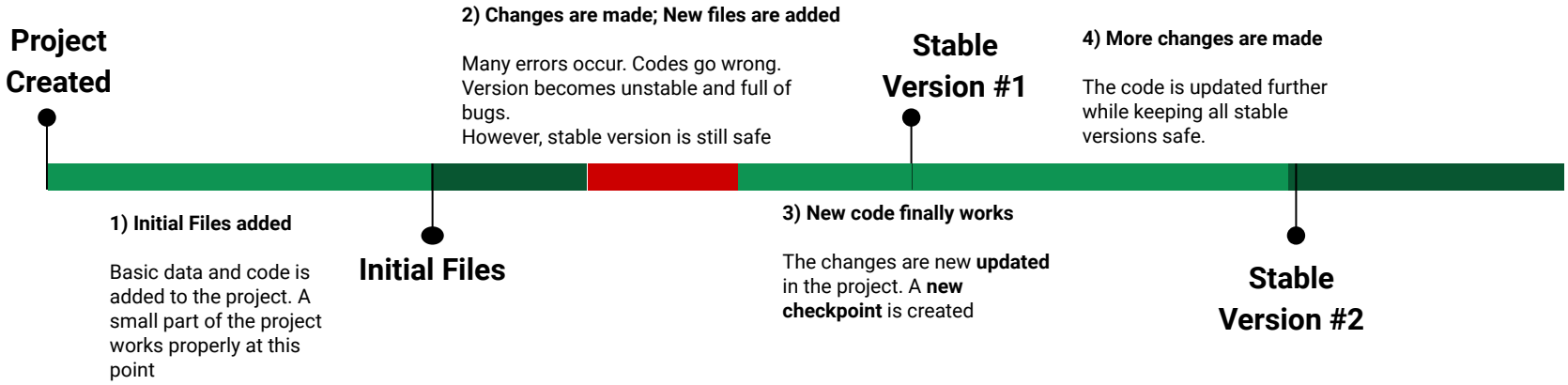
1. pip install <*library_name*>

Creating & Running Bash Files

1. Write down all commands to be executed, line by line, in a text file
2. Save the file as <*filename*>**.bash**
3. In a terminal, go to the folder where the file is saved
4. Give ‘execute permissions’ to the file by typing: **chmod +x <*filename*>.bash**
5. Run the file using ***./<filename>*.bash**

Version Control Systems

Version control example



Why do we need a version control system?

- Multiple people working on the same project
- Multiple stages of development of a project
- Multiple parallel branches in a project
- Need for an all-inclusive documentation of changes made in codes

Git - installation

For Ubuntu:

```
sudo apt-get update
sudo apt-get install git
```

Setting up your user:

```
git config --global user.name “Rishikesh Vanarse”
git config --global user.email “rishikesh.vanarse@gmail.com”
```

Creating a Local Repository

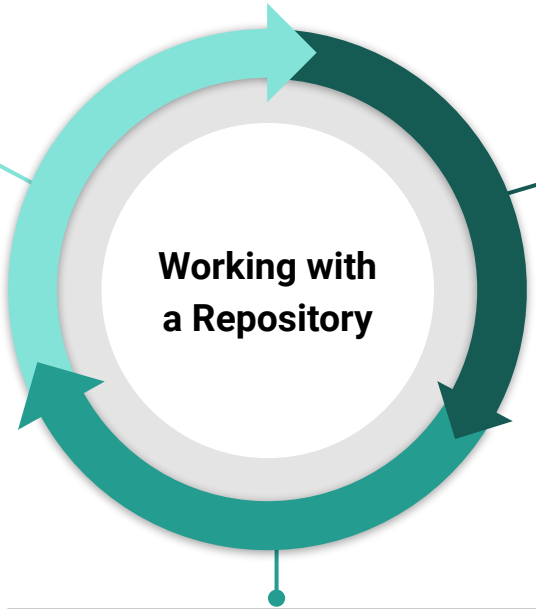
Repository - A collection of all the source code, content, history, versions and branches of a project

Creating a repository -

1. Create the folder using **mkdir** and enter the folder using **cd**.
2. Run the following command:
   ```
   git init
   ```

STEP 3 - Committing

A **'commit'** is like a **checkpoint** in your project.
Staged files are updated in the repository.
If things go wrong later, you can 'revert' back to a previous commit.



STEP 1 - Making Changes

The updated repository already exists on your computer.
You can now make any required **changes, additions, removals,** etc. and test them.
These changes are yet to be 'added' to the repository.

STEP 2 - Staging

You have made changes to certain files in step 1. Choose which of these changes you want to finalize. **Keep them ready** for the next step.
Note: Only staged files will be committed.

Staging and Committing

Staging:

```
git add *
git add <filename>
```

Committing:

```
git commit -m "<commit message>"
```

Checking status of the repository *(staged/unstaged changes, pending/previous commits, etc.)*

```
git status
git diff
git log
```

Distributed VCS - GITHUB

Multiple people working on the same project:

- Online repository (stable version)
- Local copy on the computer of every developer

- Every developer works on their own part of the project on their own computer.
- Once their respective task is complete, the online version of the repository is updated by ‘pushing’ the code online.

Open source development:
Code is made freely available online. People from all over the world can contribute to its development

Working with Online Repositories

1. If your computer does not have a copy of the repository, ***clone*** it using:
   `git clone <link_to_repository>`
   If you already have a copy, update it (***pull*** *the changes made by others*) using:
   `git pull <link_to_repository>` or `git pull`

2. Work with your copy of the repository like a local repository. ***Stage*** and ***commit*** your changes.

3. Once you are ready to update the online version of the repository, ***push*** your commits using:
   `git push -u origin master` or `git push -u origin <branch_name>`

Forks and Pull requests

Fork - An **online** copy of an already existing repository.

Why fork a repository?
You may not have access to make changes to someone's repository.

Solution:

1. Fork the repository on your account. Now you can make changes to this online copy.
2. Once you're confident about the changes, send a request to the owner of the original repository to update it with your changes. This request is called a **'pull request'.**

Branches

- Development on multiple parallel tasks on the same project
- Multiple versions of the same project

BRANCHES EXAMPLE -