



# Version Management



# Learning Objectives

The learning objectives are to

- Understand the three different aspects of git
  - Tracking changes in the files
  - Branching
  - Collaborating
- To understand git commands and apply them in various context.





Hw.py

```
print ("Hello World")
```

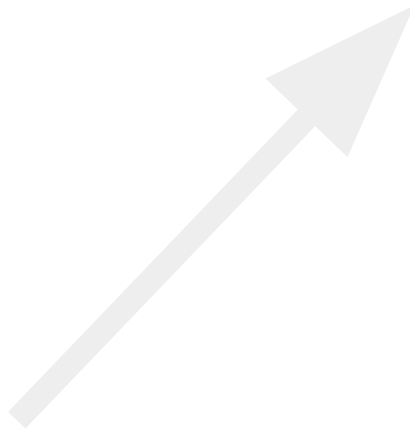


```
print ("Hello World, Arvind")
```

Hw1.py

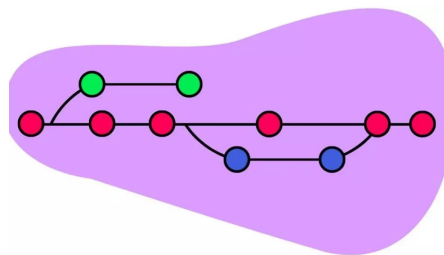
```
yourName= input()  
print ("Hello World," , yourName)
```

Hw2.py





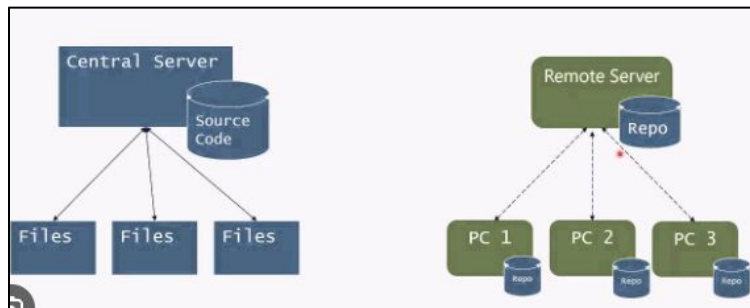
## Version Control System (VCS)



- Version Control System (VCS) is a system that manages the development of an evolving object.
- It is a system that records any changes made by the software developers.
- It is also known as
  - Source Code Management
  - Revision Control System
  - Source Code Control



## Types of Version Control System (VCS)



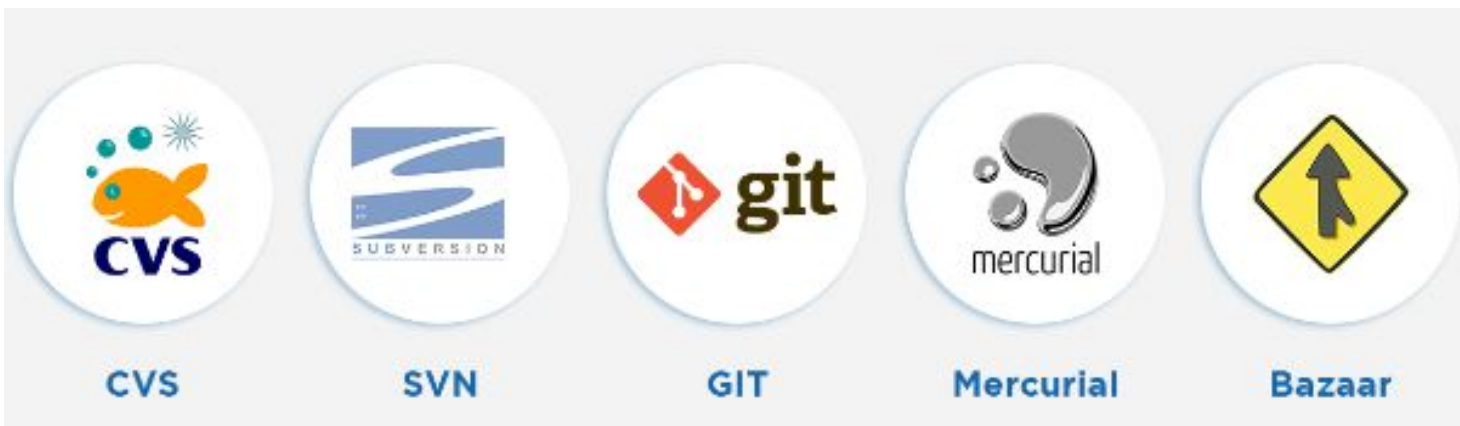
- There is only one central repository which is the server.
- Every user who needs to access the repository must be connected via network.

- Every user has a complete repository which is called **local repository** on their local computer.
- DVCS allows every user to work completely offline. But user need a network to share their repositories with other users.

Repository is a folder whose contents are tracked by VCS

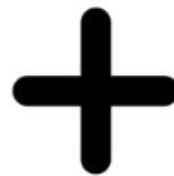


## Example of Version Control System (VCS)





**Git**





**GitHub**

Git and GitHub



## Git and GitHub Differences

 <b>git</b>	 <b>GitHub</b>
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the <b>Git</b> repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features





# What is Git?

- A Command line version control program
- Distributed Version controls system
- Cross Platform (Windows, Linux, iOS)
- Opensource, free
- No need for internet connection
- Every Git repository can act as a client and server
- Git tracks changes and not versions

```
edwork MINGW64 ~
└─$ git clone https://github.com/git-for-windows/git
Cloning into 'git'...
remote: Enumerating objects: 500937, done.
remote: Counting objects: 100% (3486/3486), done.
remote: Compressing objects: 100% (1415/1415), done.
remote: Total 500937 (delta 2494), reused 2917 (delta 2071), pack-reused 497451
receiving objects: 100% (500937/500937), 221.14 MiB | 1.86 MiB/s, done.
Resolving deltas: 100% (362274/362274), done.
Updating files: 100% (4031/4031), done.

edwork MINGW64 ~
└─$ cd git

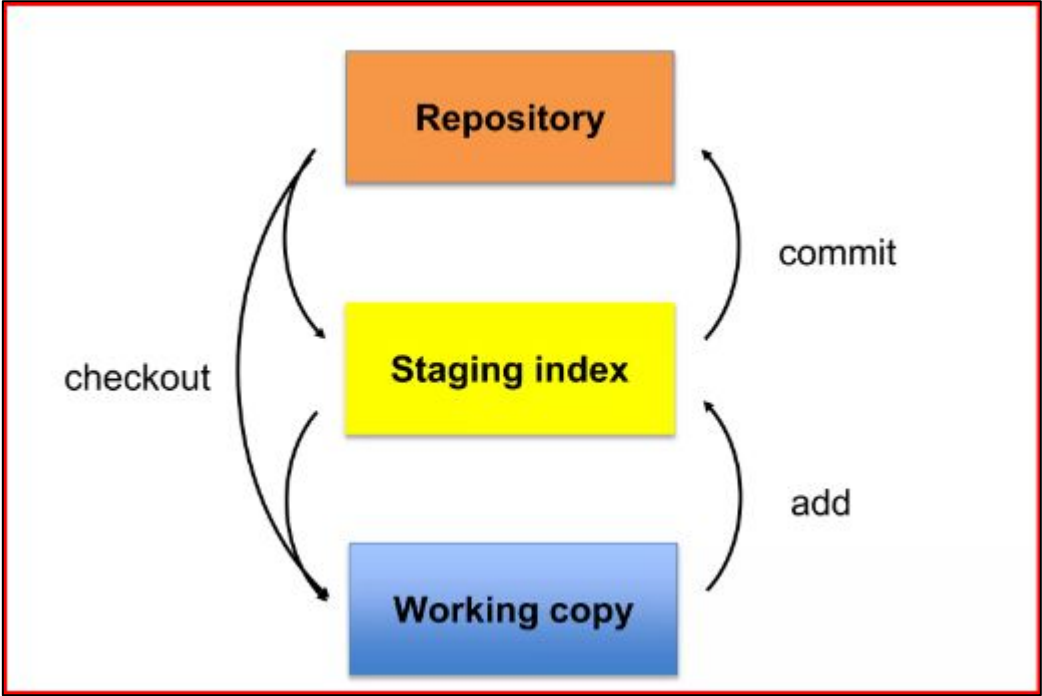
edwork MINGW64 ~/git (main)
└─$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

edwork MINGW64 ~/git (main)
└─$ |
```



# Git uses three tier architecture





# Initializing and Configuring Git

- Mkdir LearnGit
- Git init
- git config –global user.name = “Arvind Kiwelekar”
- git config –global user.email = ”[awk@dbatu.ac.in](mailto:awk@dbatu.ac.in)”
- Git config –local user.name = “

```
C:\Users\akiwe\LearnGit>git init
Reinitialized existing Git repository in C:/Users/akiwe/LearnGit/.git/

C:\Users\akiwe\LearnGit>git config -global user.name = "Arvind Kiwelekar"
error: did you mean '--global' (with two dashes)?

C:\Users\akiwe\LearnGit>git config --global user.name = "Arvind Kiwelekar"

C:\Users\akiwe\LearnGit>git config --global user.email = "Arvind Kiwelekar"

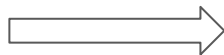
C:\Users\akiwe\LearnGit>git config --global user.email = "awk@dbatu.ac.in"

C:\Users\akiwe\LearnGit>
```



# Typical Git workflow

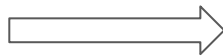
Add File



```
c:> git add hw.py  
c:> git status
```



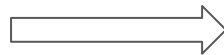
Make changes



```
c:> notepad hw.py  
c:>
```



Commit



```
c:> git commit -m "Hello World program  
created"  
c:>
```



# Git Branching

Git branch is a working copy of the main repository where you can add a feature or fix a bug.



A Python program to generate OTP and send it over email

```
import smtplib
import random
import string
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# Generate a random 6-digit OTP
def generate_otp():
    return ''.join(random.choice(string.digits) for _ in range(6))

# Email configuration
sender_email = "your_email@gmail.com"
sender_password = "your_password"
recipient_email = "recipient_email@example.com"

# Generate OTP
otp = generate_otp()

# Create the email content
subject = "Your OTP Code"
message = f"Your OTP code is: {otp}"

# Setup the email server
smtp_server = "smtp.gmail.com"
smtp_port = 587

try:
    server = smtplib.SMTP(smtp_server, smtp_port)
    server.starttls()
    server.login(sender_email, sender_password)

    # Create and send the email
    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = recipient_email
    msg['Subject'] = subject
    msg.attach(MIMEText(message, 'plain'))

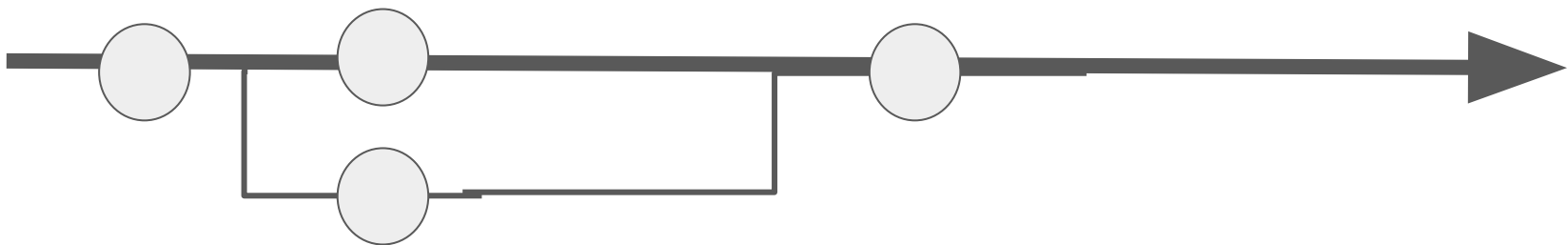
    server.sendmail(sender_email, recipient_email, msg.as_string())
    server.quit()

    print(f"OTP sent to {recipient_email}")

except smtplib.SMTPException as e:
    print(f"An error occurred: {e}")
```



# Branch Commands

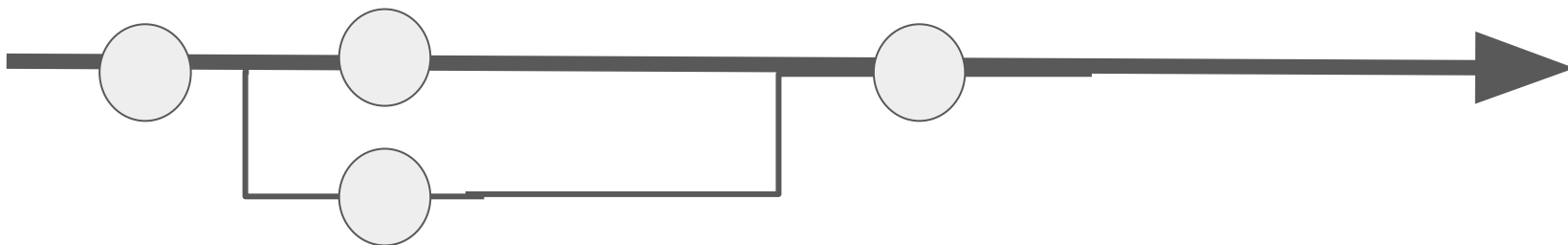


```
C:\> git checkout -b FunctionalOTP  
C:\> git checkout master  
C:\> git merge FunctionalOTP
```

- To create a new branch
- To shift to existing branch
- To merge branch



# Deleting and Renaming Branches



```
C:\> git branch -d b1  
C:\> git branch -D b1  
C:\> git branch -m Main
```

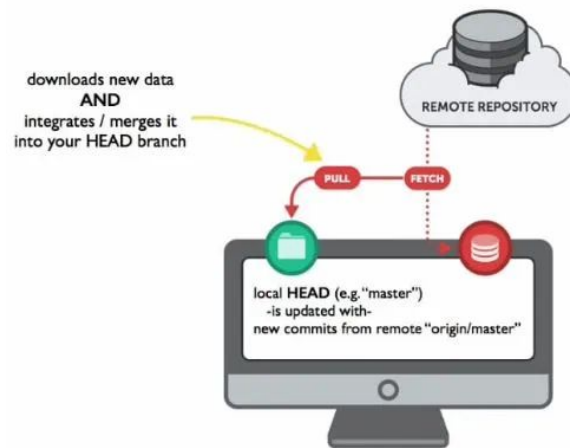
- **-d** means deleting the branch only if the branch is **pushed and merged** with the remote branch.
- **-D** means **deleting forcefully** without checking whether the branch is pushed or not.
- Renaming current branch `git branch -m <new_branch_name>` (current branch)
- Renaming any branch `git branch -m <old_branch_name> <new_branch_name>`





# Working with Remote Repository

- The **git pull** command is used to fetch and download content from a remote repository and immediately update the local repository to match the content.
- The **git pull** command is actually a combination of two other commands, git fetch followed by git merge.

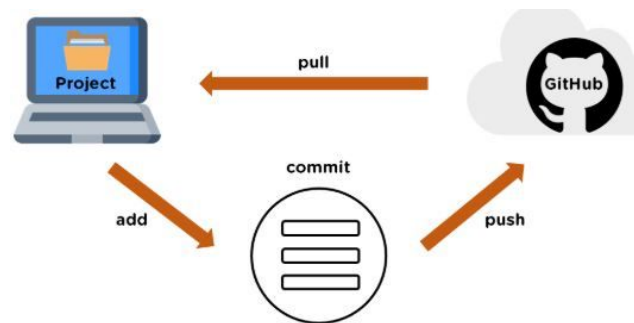


```
C:/> git remote add origin https://github.com/akiwelekar/otp\_assignment.git  
c:/> git pull origin master
```



# Working with Remote Repository

- The **git push** command is used to upload local repository content to a remote repository.
- It's the counterpart to **git fetch** but whereas fetching imports commits to local branches, pushing exports commits to remote branches.
- Remote branches are configured using the **git remote** command.



sets upstream tracking branch      remote repo branch

```
git push -u origin master
```

Primary cmd to push commits to remote repo      alias of remote repo



Quiz time

1. What is Git primarily used for?
  - a. Social networking
  - b. Version control
  - c. Graphic design
  - d. Video editing





Quiz time

1. Which command is used to initialize a new Git repository?
  - a. `git new`
  - b. `git init`
  - c. `git start`
  - d. `git create`





## Quiz time

1. What is the purpose of the "git clone" command in Git?
  - a. To create a new branch
  - b. To delete a repository
  - c. To make a copy of a remote repository locally
  - d. To merge two branches





## Quiz time

1. What is a "commit" in Git?
  - a. A saved snapshot of changes in the working directory
  - b. A comment added to the repository
  - c. A remote repository on GitHub
  - d. A branch in Git





## Quiz time

1. In Git, what is the purpose of a "branch"?
  - a. To create a new repository
  - b. To store all previous commits
  - c. To work on a new feature or bugfix without affecting the main codebase
  - d. To merge two repositories





## Quiz time

1. What is a "pull request" in GitHub?
  - a. A request to delete a repository
  - b. A request to merge changes from a branch into the main branch
  - c. A request for technical support
  - d. A request to create a new repository







## Quiz time

1. What is the purpose of the `.gitignore` file in a Git repository?
  - a. To track all files in the repository
  - b. To specify which files or directories should be excluded from version control
  - c. To list all contributors to the repository
  - d. To create a new branch





## Quiz time

1. What is the purpose of the `.gitignore` file in a Git repository?
  - a. To track all files in the repository
  - b. To specify which files or directories should be excluded from version control
  - c. To list all contributors to the repository
  - d. To create a new branch





## Quiz time

1. What is the purpose of the `.gitignore` file in a Git repository?
  - a. To track all files in the repository
  - b. To specify which files or directories should be excluded from version control
  - c. To list all contributors to the repository
  - d. To create a new branch

