Git & Git Hub
==========

what is Git? (local)
- versioning of your code

you are writing code, and you have written code for feature1 ->

you are writing code, for feature2

- multiple people work on the projects / collaborate

base code

- developer-1 (feature1)

- developer-2 (feature2)

- developer-3 (feature3)

what is github? (website / remote)

lending-club-proj-01

- conf
  - spark.conf
    spark.app.name = lendingclub
    spark.executor.cores = 5
    spark.executor.memory = 10G

  - project.conf
    input.file.path = "/user/....."
    hive.database = retaildb

- lib
  - utils.py (creating spark session)
  - transformations.py (write all the transformations)

main.py (entry point and we call various functions from here)

logger.py (to handle logging)

visual code studio

pycharm

lending-club-proj-01 (local repository) - local (git)

this has to be hosted online (on a website) - remote (github)

github.com / bitbucket, gitlab (online platforms)

session - 2
==========

- how to install git on your laptop (local)

- how to create github account online (remote)

I am using a macbook

git-scm.com

on windows instead of terminal you can get

git bash (terminal)

when windows users are installing git

ide - visual studio, pycharm, eclipse

vs code download

how to create a github account

github.com

session 3
==========

1st scenario
==========

=> you start by creating a repository in github

=> git clone https://github.com/bigdatabysumit-trendytech/financeproject.git

origin by convention means the remote from where we cloned.

= created a new file transformations.py
= modified readme.md

git status

=> if I add a new file then it shows as untracked

=> if I modify an existing file then it shows modified

whenever we make changes (unstaged) -> stage -> commit

1. make changes to existing files / add new files (unstaged)

2. we have to stage the changes (so that they are to be committed)

3. we can commit all the staged changes

git add .

git status

git commit -m "making changes to readme and adding a new transformations file"

1. I modified readme file and created a new transformation file

2. git add .

3. git commit -m "meaningful message"

our local (laptop) is ahead of remote (github) by 1 commit.

git push origin main

1. we created a repository on github

2. we cloned that repository on our local

3. we made a few changes and added new files

4. you will commit the changes to git (2 step process)

git add . (to stage the changes)

git commit (to commit it)

5. you will push these changes to github

git push origin main

session 4
==========

scenario 2:

you want to start the development on local

git push origin main

when we clone a remote project then origin is referring to that.

but when we start developing on local, then we need to set the origin.

git remote add origin
https://github.com/bigdatabysumit-trendytech/lendingclubproj.git

git remote -v

git branch (tell us on which branch we are)

we are on master branch

we should rename this master branch to main branch

git branch -M main

git push origin main

git push -u origin main

git push

scenario 1:
workflow when starting from github (remote)

github repo -> clone -> make changes -> add -> commit -> push

scenario 2:
workflow when starting with git locally

we create a project structure locally -> git init -> git remote add origin <remote-url> -> push

session 5
==========

branches in git

git branch (to check which branch)

git branch -M main (to rename a branch name)

git branch feature1 (to create a new branch)

git checkout feature1 (to switch to feature1 branch)

git checkout -b feature2 (to create and navigate to a new branch)

git checkout -b feature3 feature2 (it will take the base as feature2)

git branch -d feature3

learngitbranching.js.org

session 6
==========

we learnt branching in git

feature1

feature2

git push origin feature1

if you want to merge your changes to main branch

then create a pull request

once this pull request is approved and code is merged then we will see both main and feature1 having the same code.

when you are making changes and you feel you did something wrong in the code base and you want to go back to the previous code base then how to achieve.

1st scenario - you made changes which are not even staged
git restore <filename>

2nd scenario - you made changes and have staged those changes
git restore --staged <filename>
git restore <filename>

3rd scenario - you made changes and have committed those changes

git log (shows the commit history, the most recent commit will be at the top)

git reset head~1

git log

git reset bf914703674450b1e2ed5f0208afcea8bdb501b7

git reset --hard bf914703674450b1e2ed5f0208afcea8bdb501b7

session 7
==========

1st - we starting the project by creating a repository in github

2nd - we starting a project by creating project in local

3rd - you want to contribute to an existing project
within your org / open source project

on youtube I have a SQL playlist

Notes for this playlist is updated on github

bigdatabysumitm

github.com

fork - get a copy of the code from some other github account to your account

1. fork the project so that a copy of that project is created in your github.

2.
https://github.com/bigdatabysumit-trendytech/NotesOfYouTubeSQLSeries.git

git clone
https://github.com/bigdatabysumit-trendytech/NotesOfYouTubeSQLSeries.git

and create a new branch to make changes

3. we made the desired changes

4. committed the changes by first adding and then commit

5. push the changes to remote

git push origin feature1

these changes are pushed to my github account and not on the github of the original project

by convention

- upstream (the actual project from where I have forked)
bigdatabysumitm

- origin (our github)
bigdatabysumit-trendytech

6. create a pull request so that your changes from your github account in feature1 branch can be merged to upstream main branch.

7. some approvers will approve and merge it.

our feature1 branch in my github has the latest code
but main branch in my github is still not having the latest code

I want to get the latest code in main branch of local

and also my github

main branch in upsteam is upto date

feature1 branch in my github/local is upto date

git remote add upstream
https://github.com/bigdatabysumitm/NotesOfYouTubeSQLSeries.git

git remote -v

git pull upstream main

session 8
==========

keep the changes in a place so that we can later retrieve it.

git stash (to park the changes somewhere)

git stash pop (to retrieve the changes)

git stash clear (to clear the stash area)

to stash even the untracked file.. the new files that you have created you need to write

git stash -u

git stash

git stash -u

git stash list

git stash pop

git stash pop stash@{2}

git stash clear

git stash save "meaningful message"

session 9
==========

what is a merge conflict and how to deal with it..

main

feature1 feature2

git diff <branchname>

CICD
=====

continuous integration and continuous deployment

source control - github, gitlab, bitbucket

multiple developers working on a project

RetailAnalysis

Jira ticket
RA-17843

github
feature-RA-17843

Branching Strategy
==========

feature-RA-17843

Dev - all the developers code will be merged and some basic tests can be run

Test - your QA team would test

UAT - users will be testing the changes

Prod - final product

feature -> dev -> test -> uat -> main

you developed a feature in your feature-RA-17843

build -> test -> packaged -> deployed

compiled & build -> unit tests -> jar file is created (artifact) -> deployed (scp)

build - you create a virtual environment with all the dependencies installed

test - run unit testing / checking code quality

package - if its a java project a jar is created
in case of python it can be a zip file

deploy - deploy the code in the server (scp)
g02.itversity.com

if we have to do all of the above manually...

20 developers

we can automate this pipeline

CICD pipeline

CI - build & test

CD - Package & deploy

git push to feature branch

feature -> dev (CICD)

dev -> test (CICD)

test - uat (CICD)

uat -> main (CICD)

Automation server - Jenkins

pipeline as code

build -> test -> package -> deploy

build -> package -> deploy

Session2
==========

Deploying & Configuring Jenkins Server

cloud.google.com (GCP)

$300 free credits (3 months)

console.cloud.google.com

ssh to your jenkins server
==========
sudo apt-get install pip
sudo apt-get install sshpass

install the plugins
==========

manage jenkins -> plugins

dashboard view
github branch source
pipeline declarative
pipeline stage view

Session3
==========

branching structure

feature
dev
test
uat
main

you start working on your local

RetailProject

git init

git add .

git commit -m "adding the files"

git branch -M main

go to github and create a repo with the name RetailProject

git remote add origin
https://github.com/bigdatabysumit-trendytech/RetailProject.git

git push origin main

main, dev, uat, test all have the same code

git checkout -b feature-rp-50001

after making changes

git add .

git commit -m "message"

git push origin feature-rp-50001

when you execute the above a new feature branch will be created in github

Session 4
==========

making configurations in jenkins so that it can read the github events

branch creation
git push
pull request

we have to establish the connection between github and jenkins

and it has to be done from both ways...

in jenkins
==========

manage jenkins -> configure system

I generated a token - settings -> developer settings -> personal access token

jenkins -> github

in github
=========

go to your repository -> settings -> webhooks

<jenkinsurl>/github-webhook/

http://104.199.146.146/github-webhook/

========

set my lab credentials in jenkins for doing ssh or scp

Session 5
==========

how exactly we create our jenkins pipeline

jenkinsfile is written as a groovy script

declarative syntax

Now go to your jenkins UI and create a multi branch pipeline

create a multi branch pipeline

======

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo "build completed successful"
            }
        }
        stage('Test') {
            steps {
```

```
            echo "test completed successful"
        }
    }

    stage('Package') {
        steps {
            echo "package completed successful"
        }
    }

    stage('Deploy') {
        steps {
            echo "deploy completed successful"
        }
    }

  }
}

==========

pipeline {
  agent any

  environment {
    LABS = credentials('labcreds')
  }

  stages {
    stage('Build') {
        steps {
            sh 'pip3 install --user pipenv'
            sh '/bitnami/jenkins/home/.local/bin/pipenv --rm || exit 0'
            sh '/bitnami/jenkins/home/.local/bin/pipenv install'
        }
    }

     stage('Test') {
        steps {
            sh '/bitnami/jenkins/home/.local/bin/pipenv run pytest'
        }
    }

     stage('Package') {
        steps {
            sh 'zip -r retailproject.zip .'
```

```
        }
      }

      stage('Deploy') {
          steps {
             sh 'sshpass -p $LABS_PSW scp -o StrictHostKeyChecking=no -r .
$LABS_USR@g02.itversity.com:/home/itv005857/retailproject'
          }
      }

   }
}
=========
```