* nano <file_name>.ext ⟶ nano file editor

* git add . ⟶ dot indicates current working directory

* git push → Local to Remote push

* git stash → moving data from staged area to backend without committing the data

Note :- change user account if restricted

git branch

git branch -M main [change branch]

git show-ref [shows where is head]

git push origin HEAD:main [if Head in main]

git push -u origin main [set upstream]

git push -f origin main [force push main->main
master -> master]

A2) Demonstr
repository

merge conflict
repository

=> * git

* git
(switched

* git
* branch
* master

* git

Hello
- print
hello
+ print
hello

29-05-2024

A2) Demonstrate to create a project in remote repository and apply fork, merge, diff, merge conflict, branch, pull request concepts on repository using github

=> * git branch branch 1

(or)

* git checkout -b branch 1

[switched to new branch Branch 1]

* git branch

* branch 1 // current in green colour

* master

* git diff branch 2 // I am in branch 1 and if I modify something in branch 1

Hello welcome

- printf ("Enter your name"); // in red

\ no newline at end file

+ print ("Enter your group name please");

\ no new line at end file

05-06-2024

Merge Conflict

=> git init
git add .
git commit -m "message"

// create new branch
git checkout -b <new branch name>
git checkout <branch name> {i.e after creation}

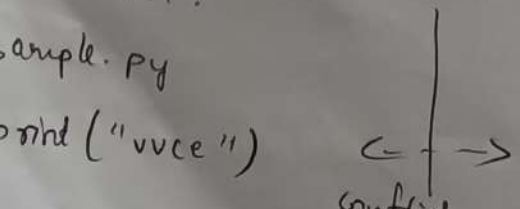
// check existing branch
git branch
* main
master * ← current branch

// push current branch and set the remote as upstream
git push --set-upstream origin master // if branch is not present in remote repo
git push // if branch already present

In github, "compare & pull request"

* Click on compare and pull request
* Add a title and description & create pull request
* If conflict exist => cannot be merged

Ex:- main branch:
sample.py
print("vvce")



master branch:
sample.py
print("atme")

Experiment - 3

Aim) Create a docker image for an application stored in local repository and run the application using docker image

Requirements:

* Docker desktop
* compilers based on your projects

System requirements:

* Enable hyper-V
* Enable WSL [Windows subsystem for Linux]

Containers:

Image:

// create a java file - App.java

```
public class App {
    public static void main (String args[])
    {
        System.out.println ("Hello ");
    }
}
```

* docker --version

16/24

* docker search mysql
* Create build

  docker build -t javaapp
* to run the docker app

  docker run javaapp

Docker File

Before build

```
FROM openjdk
WORKDIR /app
COPY . /app
RUN javac App.java
CMD ["java", "App"]
```

26-06-2024

A5) Demonstrate the process of integration github repository with Jenkins to automate the project execution in CI/CD pipeline

=> Requirements

* Jenkins
* Github repository
* Project with dependency

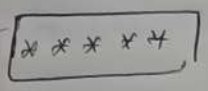
Steps :

* Create a folder on Desktop and create a sample python program in that folder.
* Add (or) clone the project using git command and
* Modify the code and again push to repository and get updated code in github repository

↳ Jenkins

* click on new item and enter the name for your project as python-project
* choose freestyle project click on ok
* In Project, in configuration, source code management choose git and provide the project url from github repository and
* paste it
* Specify branch name by */main from default master

if we modify build will trigger automatically, we have Poll scm to choose



Build steps

* choose execute window batch command and in that provide python "filename.py"
* click on Apply and save
* click on Build Now, and get console output

Download plugins for github

* goto dashboard >> manage jenkins
* click on plugins
* Click Available plugins and search for plugin "github plugin" and download

2/6/24

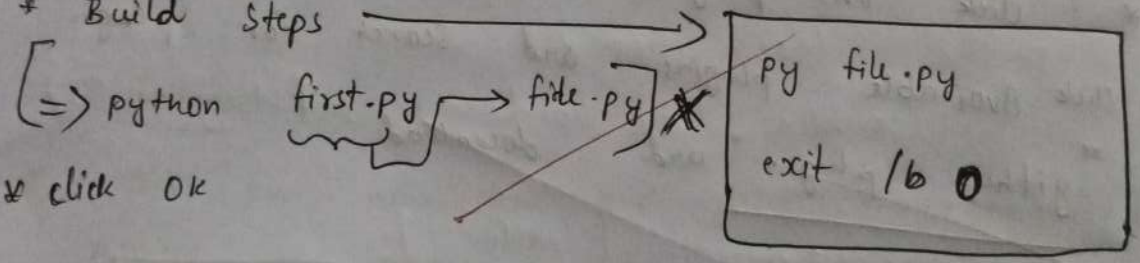
Jenkins

10-07-2024

* Go to GitHub
* Create a python file {.py}
* in local drive clone the Repo
* Modify the program
* add → commit → push

* Jenkins

* Dashboard → new project → free style project
* Configure
* Source code manager → Git → provide URL
* choose the branch {default (or) main}
* Poll SCM [5 -star] [with space] not necessary
* Build Steps



* click Ok
* Build now

Console output

* for output → status of Build → success

10-07-2024

10-07-2024

c1) Integrate communication channel with Jenkins for status of project and also enable email notification for build (slack)

Group:- vvcegroup.slack.com

* Slack -> Dashboaard
* Manage Jenkins
* goto Plugins
* type /search -> slack Notification plugin
  Global slack Notifier Plugin

General config

* ~~Enable~~ Manage Jenkins
* click system
* At end their is slack, workspace - vvcegroup
  credential - devops lab
* Click Add -> click Jenkins
* Domain no changes
* Kind -> Secret text

URL : https://my.slack.com/services/new/jenkins-ci

* New configuration
* channel => #devops-demo
  click Add CI jenkins integration
* copy ~~secret~~ key /token
* goto Jenkins paste slack text
* ID, new name
* Click on Add

(A2)

git branch -M main → change branch

git checkout -b new-branch ⇒ create new branch ⇒ git branch ⇒ * main & new

git checkout main → change to main branch

stay in new-branch

git diff main → [current → new-branch] compare current branch with main

---

① → git merge main → changes in new-branch are merged with main

---

② → Pull request → tells others about changes you have pushed

new branch

compare & Pull request → in GitHub

base : main compare : new-branch

Add new feature

create pull request

merge pull request

confirm merge

git reset → to undoing any changes (or) commit

---

git pull origin main [Remote to local] [stay in main]

Merge conflict

changes in both branch at same place in vs code manually same any one or both and commit it and see git diff empty and change branch and and perform

⇒ git merge other branch

A3

- Project
- Freestyle
- SCM => Git select
  - Board to build => */main
- Git hub
- New repo
- Create new file
- python file & code
- Right side commit changes
- Copy URL & pase in Jenkins
- Build triger -> Pole SCM
- Build step -> python first.py
  exit 16 0

Slack -> vvce email

Docker -> vvce email -> pass mail -> [redacted]
user -> hemanth01234567

Github -> vvce email => NumKillah

Jenkins -> sivatej

Enable Hyper-V] In Turn windows feature on/off

Hyper-V - Enabler .bat

Hemanth01234567

↑ user name, Password

(B1) & (B2)

* Dokes Hub -> cfg main ,
* Docker desktop
* VS code -> Java file .java -> demo.java
* Dockerfile =>

```
FROM openjdk
WORKDIR /app
COPY . /app
RUN javac demo.java
CMD ["java", "demo"]
```

=> commands

* docker build -t program1 . // upto here (B1)
* docker images
* docker login -u vvce@vvce.ac.in
  password : _____
* docker tag <image_id> hemanth01234567/program1:latest
* docker push hemanth01234567/program1:latest

(82) Slack

* In Jenkins ⇒ Dashboard > Manage Jenkins > Plugins [Available plugins]
* search slack => ☑ ☑ Slack notifications & Global Slack
* and install wait till success and Restart
* Dashboard >> Manage Jenkins >> System
* search slack → in workspace => private group ⇒ add
* kind secrete text
* in new tab search my-slack.com/services/new/Jenkins-ci
* choose [devops-demo] channel
* Add jenkins CI integration
* In step-3 copy token ID credentials ☐ as value
* Paste it in secrete ☐ of above step * id → 13
* Credentials → id 13 [Add]

* New item => any name
* in Build steps => Execute windows batch command
  => java -version
* Post build action => Notify success
  select Slack Notification Notify every failure
* Build now

Maven

org.apache.maven - quick start 1.4

Group id: ex123

Artifact id : ex123

---

* Eclipse
* Create New Maven project in other
* org.apache.maven.archetype - quickstart 1.4 [filter]
* use local (00) any system directory
* Group Id : exd2342
  Art — u ———] finish then enter "y"
* search simple json copy [1.1.1] => Maven code
* paste in pom.xml in <dependencies>
* In primary demo, create folder exd2342file
* in ex12342file create emp.json
* { "firstname": "name" }
* Next in src/java create New class java
* write the code there
* copy emp.json in format like ("..\\ex12342file\\emp.json");

08-05-2024

Exp - 01

GitBash and GitHub

1) Demonstrate and create project in local and remote repository using GitBash and GitHub and apply init, status, log, add, commit, push, config, clone and reset commands on repository

=> Required tools :- GitBash [Local], GitHub [remote]

Git commands :-

* status -> display status of the repository
* git add <file_path> -> to include to tracked files
* git commit -m "message"
* Author identity :
  git config <--global / --local> user.email "email@gmail.com"
  git config <--global / --local> user.name "user_name"
* git init -> initialize the git in Folder
* git log -> shows commit logs + Author
* git reset --hard <commit_id> -> commit back
* git clone <github_repo_https_link>

15-05-2024

* cd .git -> to change directory to git
* cd .. -> return from git directory