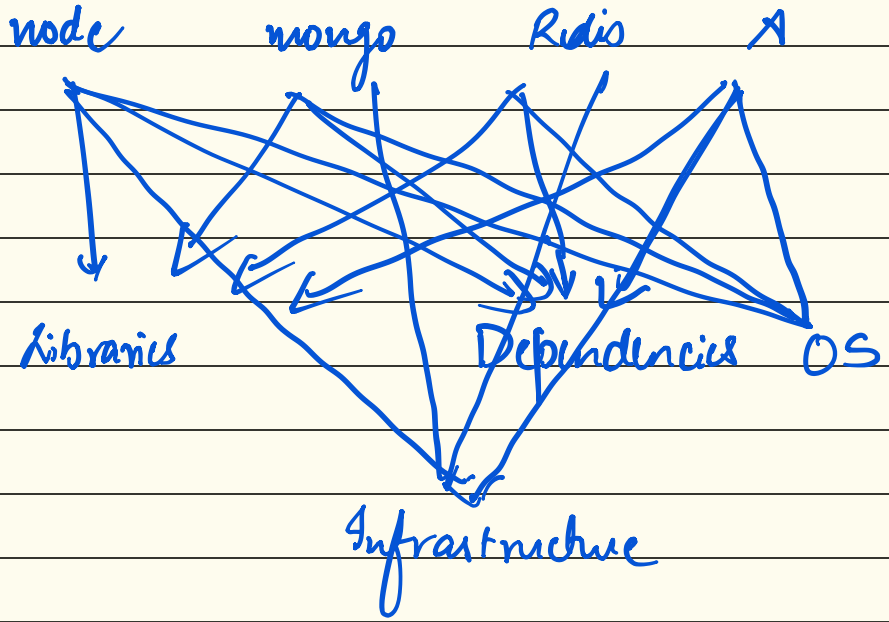
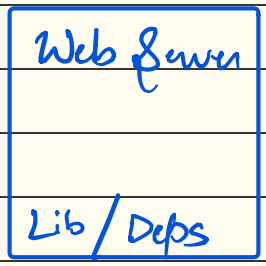
Dockers (Why do we need)

Concept → Matrix from Hell



1) Compatibility / Dependency
2) Long Setup Time
3) Different dev/prod environment

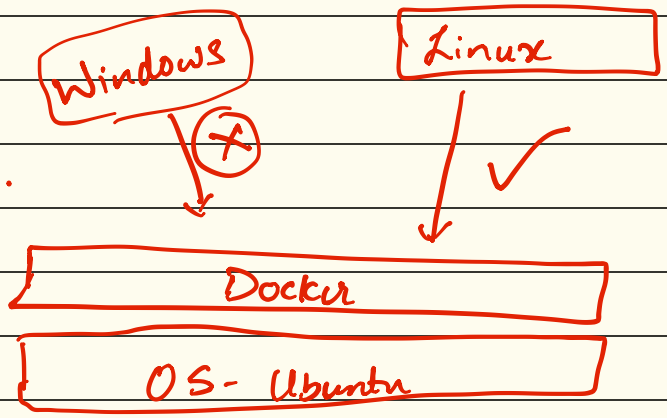
Using docker we can run each component in seperate containers



Operating System

↳ OS Kernel → responsible to interact with underlying hardware.
& software → consist of different UI, drivers, compilers, file managers, dev tools etc.

But they all share same OS kernel
"linux"



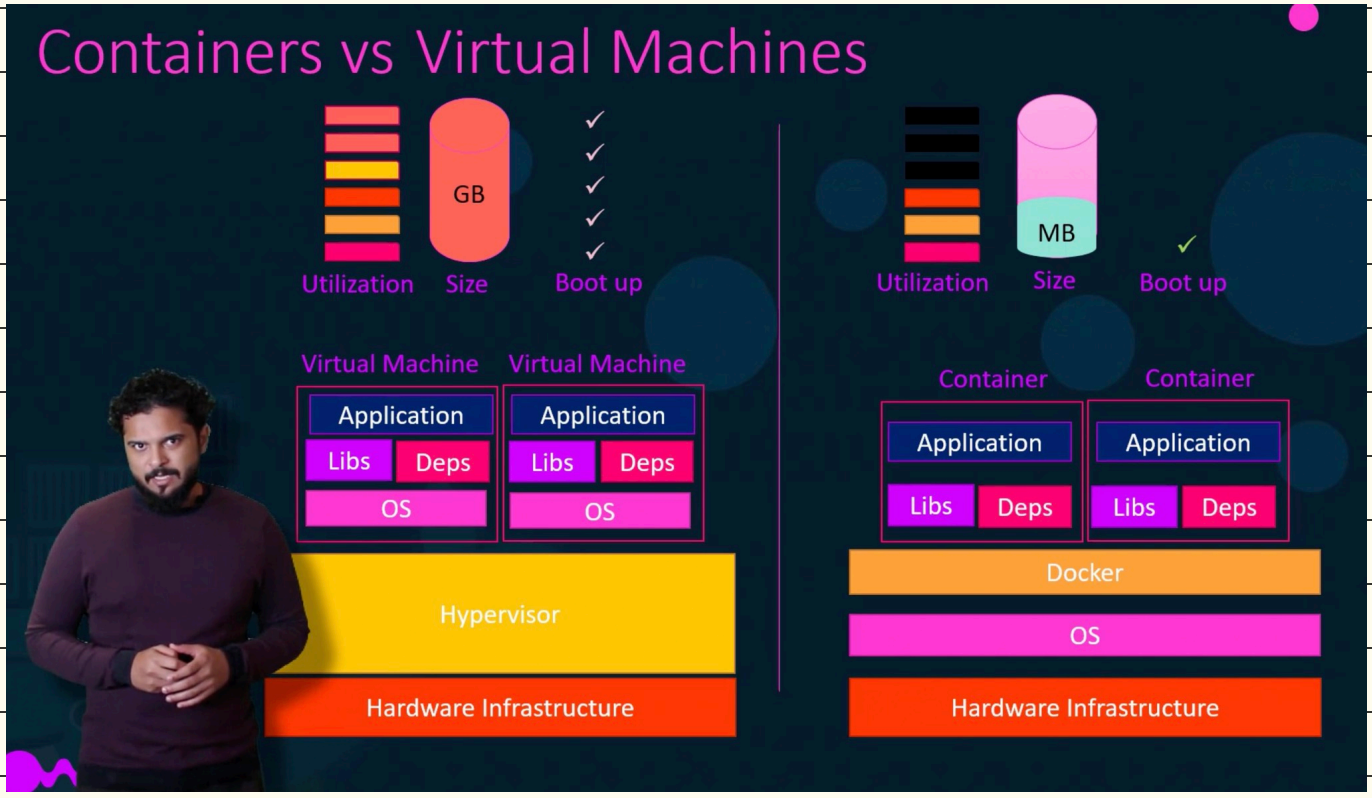
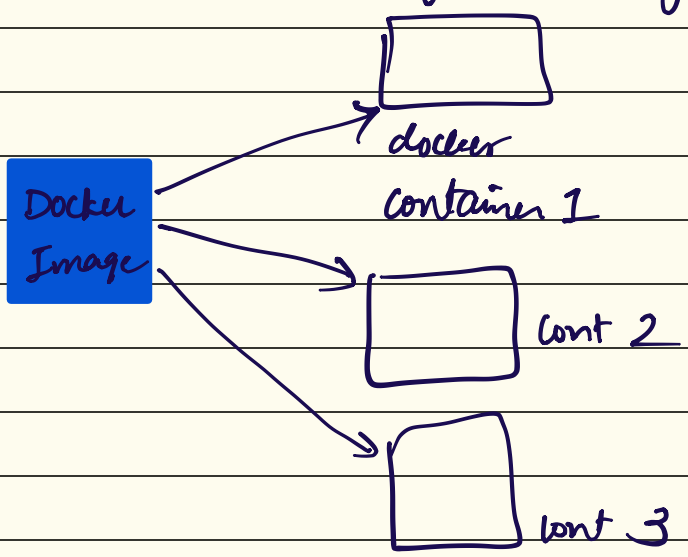


Image Vs Container

↳ package or template, used to create containers

↳ Containers are running instance of images having their own env.



Docker Command

↳ docker run

↳ docker run nginx

↳ docker ps → shows all running containers

↳ container Id
image
command
created
status
port
name

↳ docker run -d --name <any_name> node → To name a container

↳ docker ps -a
↳ shows all container running / stopped container

↳ docker stop <container-id> → To stop

↳ docker rm < c_id > → To remove permentatly

↳ docker images → To see the no. of images & there sizes

```
docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f68d6e55e065        4 days ago          109MB
redis               latest              4760dc956b2d        15 months ago       107MB
ubuntu              latest              f975c5035748        16 months ago       112MB
alpine              latest              3fd9065eaf02        18 months ago       4.14MB
```

↳ docker rmi nginx → removes the images
↳ Remove all the container first of that image (dependent ones)
↳ Then remove the image

↳ docker pull nginx → download but not run.
↳ pull image & run in local.

↳ docker exec <container_name> cat /etc/hosts

↳ docker run kodecloud/app
↳ This logs all the O/P to the screen but will do nothing other than it in foreground.

↳ docker run -d kodecloud/app
↳ This runs in background mode

↳ docker attach a23 43d → (container id)

Docker Run

↳ docker run redis ⇕ → takes latest

docker run redis:4.0 → tag

The container is in non interactive mode by default. (cannot take input)

docker run -i redis:4.0

To attach a pseudo terminal → -t

↳ docker run -it Kodekloud/simple-prompt-docker

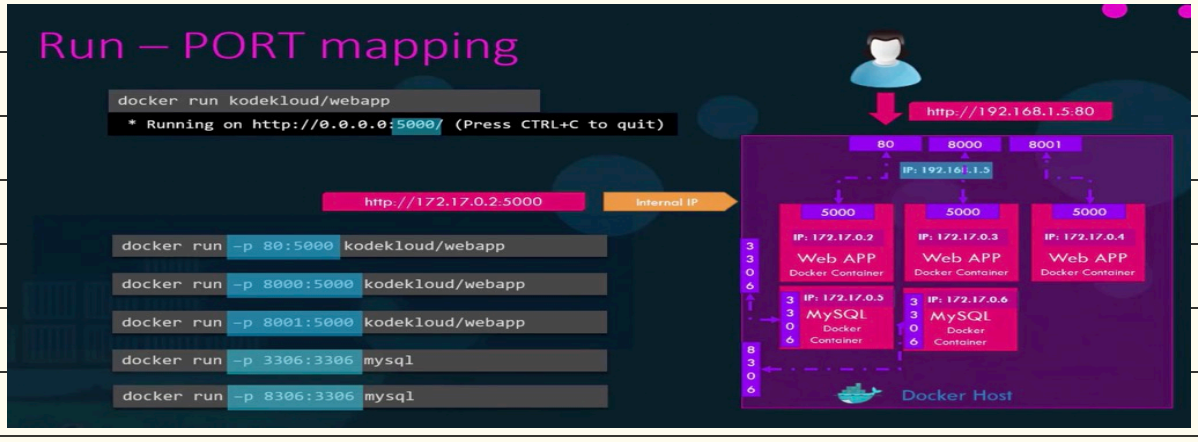
→ Run Port Mapping

→ To run a web application on web browser which is running in docker container

To port (-p)

↳ docker run -p 80:5000 Kodekloud/webapp

All traffic on docker host 80 will be redirected to 5000 inside the docker container

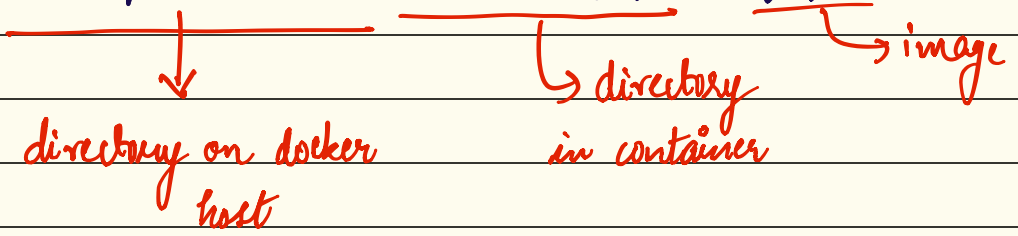


If we are running a container and then delete it, all the data from that container is gone.

To persist the data on the docker host rather than container.

(-v)

docker run -v /opt/datadir:/var/lib/mysql mysql



⇒ To check specific details of container, then instead of docker ps use ⇒

docker inspect <docker_name> → return data in json format

```
docker inspect blissful_hopper
[
    {
        "Id": "35505f7810d17291261a43391d4b6c0846594d415ce4f4d0a6ffbf9cc5109048",
        "Name": "/blissful_hopper",
        "Path": "python",
        "Args": [
            "app.py"
        ],
        "State": {
            "Status": "running",
            "Running": true,
        },

        "Mounts": [],
        "Config": {
           "Entrypoint": [
                "python",
                "app.py"
            ],
        },
        "NetworkSettings": {..}
    }
]
```

To see the logs of container which is running in background (-d).

docker logs <container_id>

docker images

→ how to create our own image

1. OS Ubuntu
2. update apt repo
3. Install dependencies using apt
4. Install python dep using pip
5. Copy source code to /opt folder
6. Run the web server using "flask" command.

→ Writing the docker file for the above

i) name a file docker file

Dockerfile

```
FROM Ubuntu
RUN apt-get update
RUN apt-get python

RUN pip install flask
RUN pip install flask-mysql

COPY ./opt/source-code

ENTRY POINT FLASK_APP = /opt/source-code/app.py flask run
```

docker build Dockerfile -t yash/my-web-app. → build locally

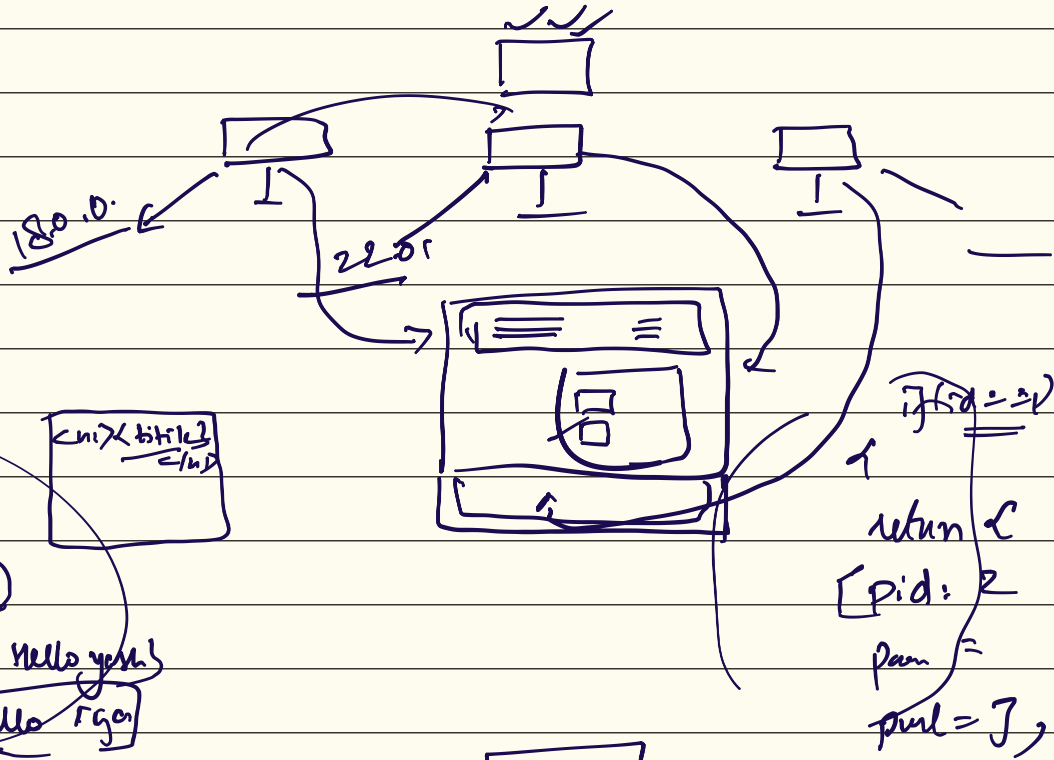
docker push yash/my-web-app → build on the dockerhub

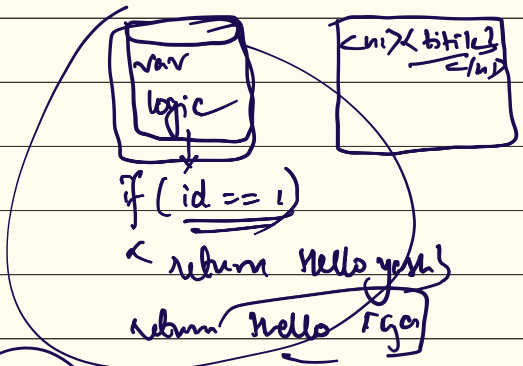
yash → acc. name

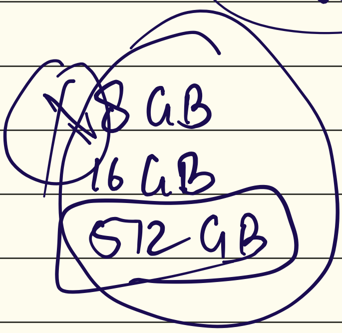
my-web-app → image name

github

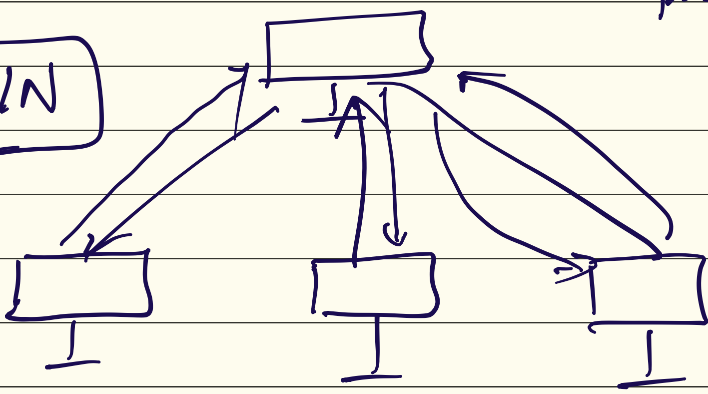
↳ git



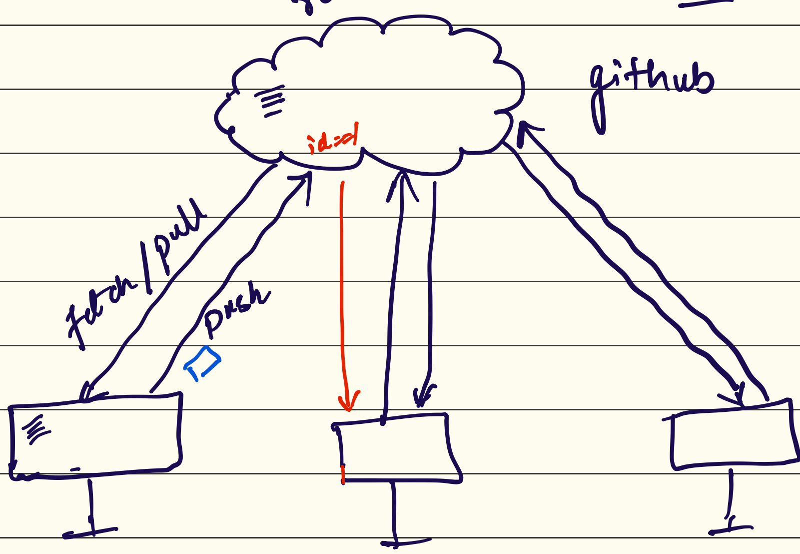




SVN



remote



github

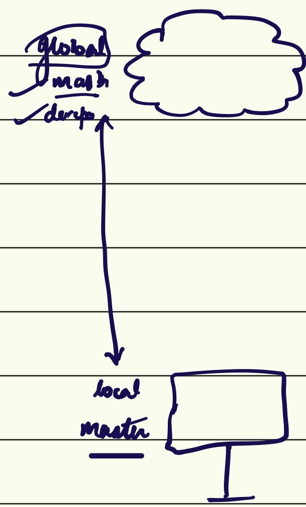
repository

↳ ssh link

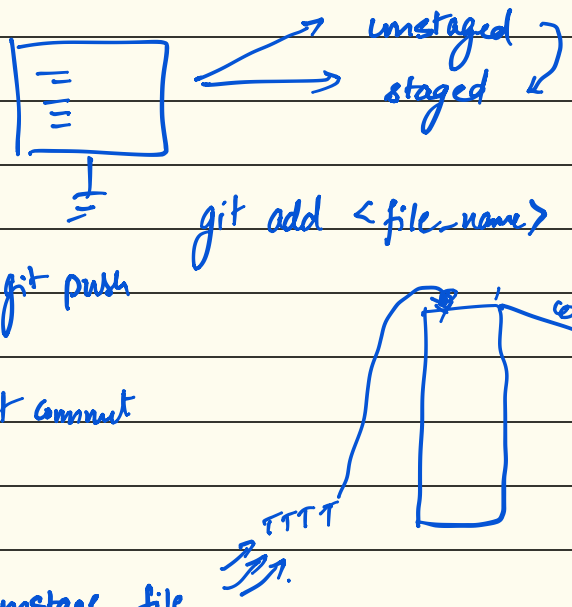
https link

clone

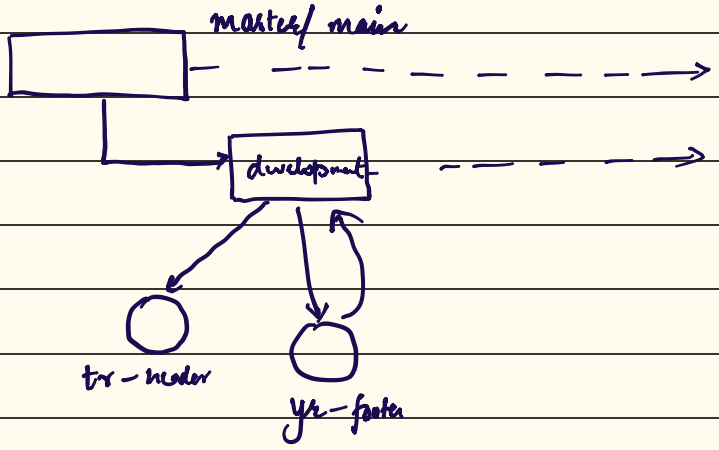
git init



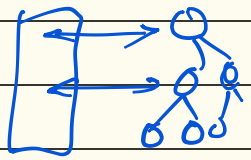
commits → piece of code, that is done by you and you commit that the code is ready to be sent.



Branches



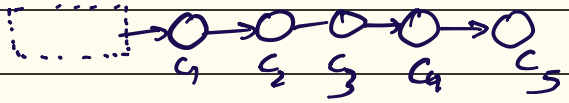
To stage a file → git add <file_name>
↳ all files → git add .



To make a commit → git commit -m "first commit"
↳ feat: worked on payment module
↳ fix: Handled null scenario in p.m.

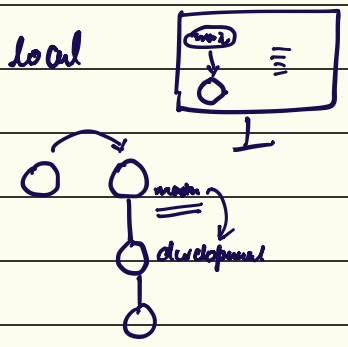
To check the branch → git status

fetch / pull / push

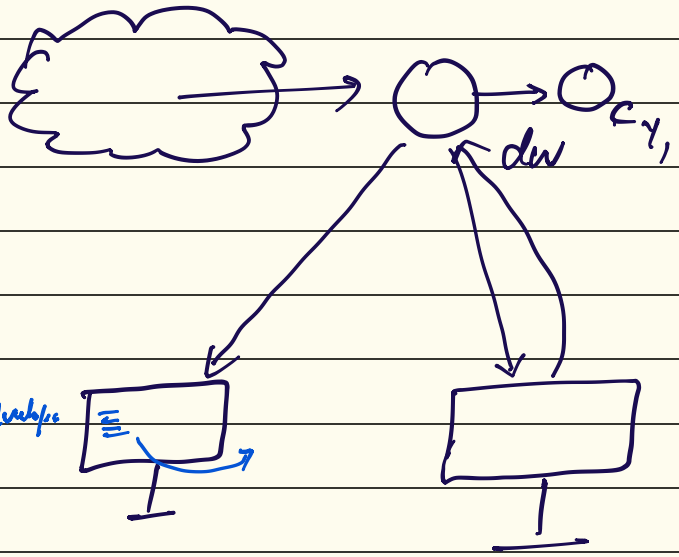


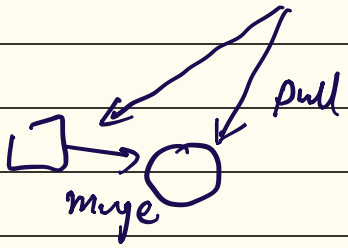
git fetch origin master

fetch the branch if it does not exist in local



- git branch -m "development"
- git checkout development
- git checkout -b "development"

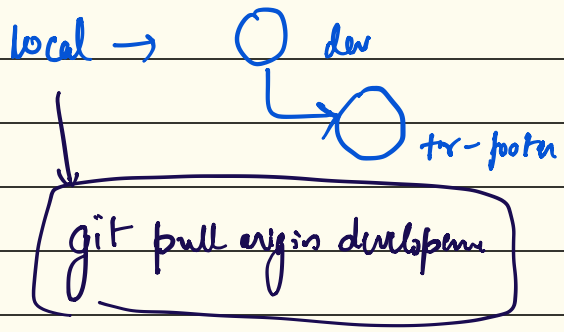


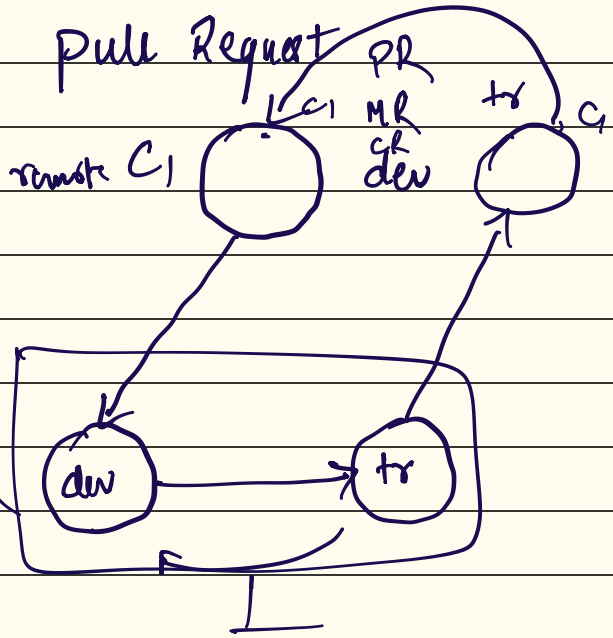


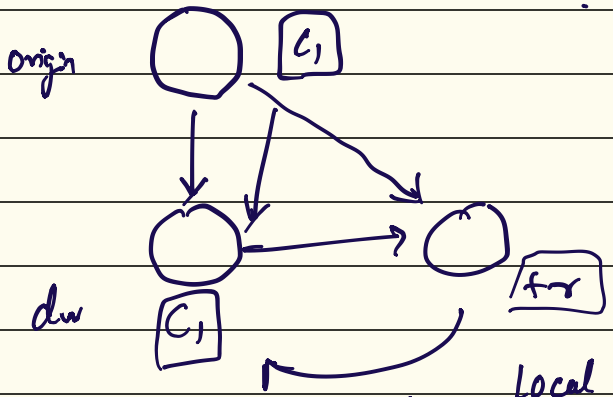
pull = fetch + merge

pull request

git fetch origin development

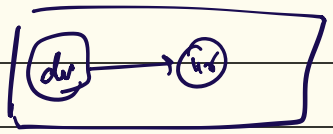




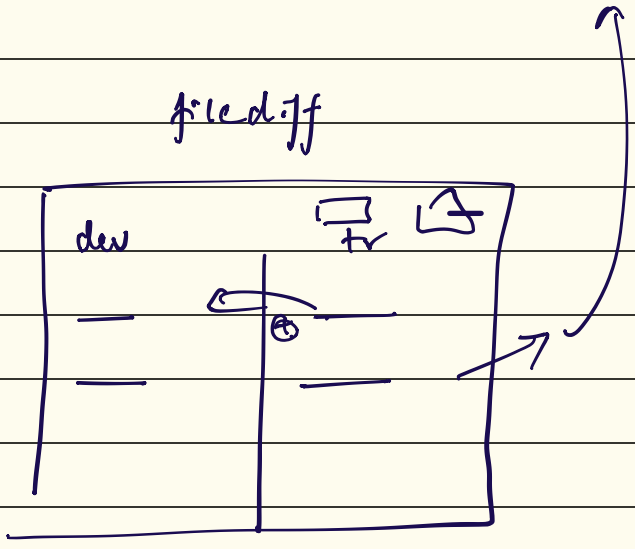


Local merge

git pull
dev
git pull



file diff



git config list

↳ git config --global user.name "Tejas"

user.email "_____"

--global, --local, --system