Manipulating Containers with the docker :-

docker version   
docker run hello-world   
docker run busybox echo hi there ( replace the text)  
docker run busybox ls (list all folders )   
docker ps ( list all running commands)  
docker run busybox ping [google.com](http://google.com)   
docker ps   
docker ps --all  
docker run= docker create + docker start   
docker create hello-world ( id generated)  
docker start -a (-a is to watch for output)   
docker start (gives id only instead of output)  
docker system prune ( to remove stopped containers entirely )  
docker create busybox echo hi there   
docker logs ( to get logs of container )  
docker create busybox ping [google.com](http://google.com)   
docker stop ( SIGTERM it gives time to save something if want)  
docker kill ( SIGKILL shut down immediately)   
docker exec -it ( execute additional command in a container)  
Eg, docker exec -it redis-cli   
-it = -i -t  
docker exec -it sh ( getting cmd in a container)   
  
Ctrl+d ( stop cmd in container) / exit   
docker run -it busybox sh   
docker run   
touch hi there ( to make new file )  
  
Building Custom Images through docker server : -

Docker file( configuration to define how our container should behave ) -> docker client -> docker server -> usable image!  
  
For creating a Dockerfile ( below is given flow)   
Eg,   
  
mkdir redis-image   
cd redis-image  
Inside redis-image dir create file called Dockerfile and write below lines of things   
  
Specify a base image eg FROM alpine   
^   
Run some commands to install additional images eg RUN apk add --update redis  
^  
Specify a command to run on container startup eg CMD ["redis-server"]  
  
After writing to Dockerfile , run below commands   
  
docker build .   
  
After running above cmd , generate one id once successful then run   
  
docker run ( it start the redis server )   
  
Order of codes are impt in Dockerfile   
  
Make real projects with docker   
  
Inside dockerfile   
FROM node:14-alpine   
WORKDIR /usr/app  
COPY ./package.json ./   
RUN npm install  
COPY ./ ./   
CMD["npm","start"]  
  
docker build .   
  
docker build -t shalu7/simpleweb .   
docker run shalu7/simpleweb   
docker run -p 5000:8080 shalu7/simpleweb   
  
docker compose with multiple Local containers:-   
  
  
docker-compose.yml   
docker-compose up   
docker run -d redis   
docker-compose up -d ( start containers in background)  
docker-compose down ( to stop container)  
docker-compose up --build   
docker-compose ps   
  
  
Creating a Production - Grade Workflow :-  
  
Given overview of react app implementation  
npm install -g create-react-app  
create-react-app frontend

Instead of above 2 lines run

npx create-react-app frontend  
cd frontend  
npm run start   
npm run test   
npm run build

Now create [Dockerfile.dev](http://Dockerfile.dev) in ur project directory for development

In order to build we use docker build . but now throws an error

Use below cmnds

docker build -f [Dockerfile.dev](http://Dockerfile.dev) . where f -> files

pwd - present working directory

docker run -p 3000:3000 -v $(pwd):/app <id> (docker image to our m/c)( not work in windows powershell) -> throw errors

docker run -p 3001:3000 -v /app/node\_modules -v $(pwd):/app <id> (setting up volume for dynamic change)

Now create docker-compose.yml file in ur project directory for shorthand command for above one   
docker-compose up

For test :   
docker run <container-id> npm run test (override the CMD context )

docker run -it <id> npm run test (to get more friendly console)

docker exec -it <id > npm run test ( to run test cmd in 2nd console using container id )

After adding services for test in docker-compose.yml file in project directory   
docker-compose up --build

With above approach we can’t write anything in the test console as it’s running in development one

docker attach <id> ( not able to do so )  
docker exec -it <id> sh ( open shell)   
ps ( after this we see it's not actually running npm run test rather running npm and start.js that's why not able to attach anything in console   
Create new Dockerfile for production and add nginx for production container   
docker build .   
docker run -p 8080:80 <id>

version: '3'

services:

  web:

    build:

      context: .

      dockerfile: Dockerfile.dev

    ports:

      - "3001:3000"

    volumes:

      - /app/node\_modules

      - .:/app

  tests:

    build:

      context: .

      dockerfile: Dockerfile.dev

    volumes:

      - /app/node\_modules

      - .:/app

    command: ["npm","run","test"]

Continuous integration and deployment:-  
  
.travis.yml file   
  
  
Building a multi-Container application:-  
  
Create Fib Calculator application  
  
Multi-Container Deployments to AWS:-  
  
Dockerrun.aws.json file -> to specify imgae to run from docker hub