49

SDLC model (Interview)

6-11-22 Sunday

* Challenges →

* Agile - have to same team at the end of Project else Project failure occure.
  - Development needs to the fast
  - Requirements changes frequetly

* waterfal → (Best suited)
  - complete Requirements are clear & fixed
  - Product defination is stable

* DevOps (Best suited) (Development team + IT operation)
  - Requirement Chang frequetly
  - Development needs to be agile
  - operation ——"—— Agile

Strategy
Contineous development, Testing, integration deployment & monitoring in SDL

Tools → developing (git, jira, subversion) Building (maven, gradle (for mobile application))

TestNG, Selanium (Testing)

Integration: Bamboo, Jenkins

PAGE No.
DATE / /

* Version control system →
  ① central (subversion) (many work
  ② distributed (Git)

* central VCS →
  - many work but not at a time (mutual exclu
  - if central server (Repository) crashed all goes
  -

* distributed VCS →
  - many work at a time
  - many version with new change, everytime
  - can access any where.

desk processing CCE exam

* How to install Git →
  ↓ download git
  - download 64 bit Git for windows
  - open it next - next
  - choose

07 Sept screening

9-11-22

[Git has 85 command Total]

Any folder ↳
  Right click ↳
    Git bash here ↳
      Git init

* Branching → Seperate your code logicaly
  $
  > git init
  > git branch
  > git status
  > git add -a

> git commit -m "Last commit before branching"

> git branch demo
> git checkout demo ("; chang head pointer to demo)
> git status
> git checkout master
> git log
> git add -A
> git commit -m "
> git log
> git diff master
> git checkout
> git merge demo
> git Log
> git rebase master.
> git branch -d demo
> git branch
> git log
> git checkout -b demo ("create & checkout)
> git add -A
> git commit -m "Last uedate on branching"
> git checkout master
> git rebase master
> git checkout demo
> git rebase master
> git checkout master
> git merge demo

PAGE No.
DATE / /

* Forking →

* Integration →
  Before continuous int. after

* Jenkin installation :
  Jenkins.msi → Download Jenkins
  ↳ LTS 2.361

clone - clone a repository in to new directory
init - create an empty Git repository / reinitialize an existing one

add - Add file contents to the index
mv - move / rename or symlink
restore - Restore working tree files
rm - Remove files from the working tree & form the index

bisect - use binary search to find the commit that introduced a bug
diff - show changes bet^n commits, commit & working tree etc
grep - Print lines matching a pattern
log - show commit logs
show - show various types of objects
status - show the working tree status

branch - List, create, or delete braches
commit - Record changes to the repository
merge - Join two or more developement histories together
rebase - Reapply commits on top of another base tip
reset - Reset current HEAD to the specified state
switch - Switch branches
tag - create, list, delete / verify a tag object signed with GPG

4.9

* Git hub → uses for version control
* docker

fetch - Download objects & refs from another reposit
PULL - fetch from & integrated with another or local branch
Push - update remote refs along with associated objects.

* after installtion →
  Create job ↳
    manage jenkins
      ↳ con

> git remote -v
> git 'Push origin' master
>
(Dev-op)

* Docker → (containers)
  - To solve Ram (share) utilization of machine
  - CICT process
    - work environment
    - RAM consumed by container very less
    - easily run appl^n by Packaging
    - light weight & easily shared. to any OS

windows featus
  ↳ ✓ Hyper V
  ✓ Virtual machine
  ✓ window Hypervision
  ✓ subsystem
    save

Restart

download docker ↳
  install for windows ↳
  (docs.docker.com)

> Inside cat "Path"

hub.docker.com ↳
Register yourself

↳ continew with free
↳ verify email

installer → (4.14.0)
click on dowloaded installer file
↳ configure ✓✓ ok
↳ close & Log out

windows powershell
↳ docker --version
> start
↳ Accept
> wsl --install (To install)
> wsl --update

~~Power~~ Shell
Run as administrator
> wsl --update
> Link (command enter
> docker images
> docker pull busybox
> docker image
> docker run busybox
> docker run busybox echo "Hello world"
> docker Pull httpd
> docker run -it -d httpd
> docker images
> docker ps
> docker ps -a

4.9

> docker exec -it "ID httpd" (copy)
> docker pull ubuntu
> docker images
> docker -it -d ubuntu
> docker ps
> docker exec -it "container ID" bash
> ~~sudo~~ apt update
> add-apt-
> docker stop (ID)

[w3schools.io/file/yaml-introduction/]

notes of yaml

> docker exec -it (ID) bash
> Python3 --version
> ls
> cd users
> cat index.html
> index.html
<h1> hello world </h1>
> ls
> docker ~~commit~~ ps
> docker commit (ID) (account name)/myubuntu
> docker images
> docker login
hematitedoc
pass....
> docker push hematitedoc/myubuntu (*)
> docker pull hematitedoc/myubuntu:latest
> docker image
> docker rmi (ID)

ubuntu
busybox
httpd

> docker run -it -d (*)
> docker ps
> docker exec -it (ID) bash

microservices
docker compose, built.

PAGE No.
DATE / /

> cd usr
> ls > $ apt update > apt install python3
> python3 --version
> docker stop (ID)
> docker kill (ID)
> docker ps -a
> docker start (ID)
> docker restart (ID)
>