Pràctica 4: Integració Continua (IC) amb Jenkins

1- Introducció i objectius de la pràctica m08uf4pr4

a) Lectura de les especificacions de l'activitat.
b) Conceptes bàsics de CI/CD
c) Instal·lació de Jenkins

2.- Conceptes bàsics de CI/CD

Una aplicació passa normalment per les següents etapes:

- Desenvolupament: L'equip de desenvolupadors treballen conjuntamet creant, actualitzant i pujant a un dipòsit continuament el codi. Un cop en el dipòsit el codi es compila (Build) i fan comprovacions de funcionament (Test).
- Distribució: Si el codi passat totes les proves es pot passar a un dipòsit des del qual es pugui enviar a producció en qualsevol moment. Aquesta etapa també es pot aprofitar per fer proves de funcionament en producció.
- Desplegament: Enviament del codi a producció per fer disponible l'aplicació als usuaris.

La Integració Continua o Continuous Integration (CI) d'una aplicació:

- Permet que un equip de desenvolupadors puguin treballar de manera conjunta però dedicant-se cadascú a desenvolupar i actualitzar parts diferents del codi.
- Quan un desenvolupador ha finalitzat una actualització del codi, pot pujar-la a un dipòsit compartit amb els altres membres de l'equip a on es troba la darrera versió del codi i fer un merge (és a dir una fusió) amb el seu nou codi.
- Un cop s'incorporen les modificacions d'un desenvolupador, es validen amb la compilació automàtica de l'aplicació i l'execució de diferents proves automatitzades per garantir que els canvis no hagin introduït una falla.
- Si una prova automàtica detecta un conflicte entre el codi nou i l'actual, la CI facilita la resolució d'aquests errors amb rapidesa.

La Distribució Continua o Continuous Delivery (CD) d'una aplicació:

- Envia de manera automàtica el codi validat a un dipòsit que pugui ser enviat en qualsevol moment a l'entorn en producció.
- Permet fer proves automatitzades en un entorn de producció abans de fer disponible el codi als usuaris. Si una prova automàtica detecta un conflicte en l'entorn de producció, aquesta fase facilita la resolució d'aquests errors amb rapidesa.
- Permet automatizar si es creu convenient el pas a producció fent disponible el codi als usuaris.

El Desplegament Continu o Continuous Deployment (CD) d'una aplicació:

- Envia de manera automàtica el nou codi als equips en producció i per tant fa disponible l'aplicació als usuaris. És una extensió de l'etapa anterior.

La metodologia de treball CI/CD (Continuous Integration + Continuous Delivery/Deployment) té com a objectiu automatitzar totes les etapes indicades anteriorment de manera:

- El temps que passa des de que un desenvolupador fa una actualització fins que els responsables d'operacions (administradors de sistemes) fan disponible en producció l'apliació als usuaris sigui el mínim possible.
- És minimitzin els conflictes entre etapes i responsables de cada etapa.
- Es pugui supervisar continuament l'aplicació per evitar errors de funcionament.
- Els usuaris puguin tenir disponibles com més aviat millor i a ple rendiment totes les millores que es produeixen en l'aplicació.

Un pipeline CI/CD o canalització CI/CD consisteix en seguir els passos descrits anteriorment d'una manera ordenada amb l'objectiu de fer disponible com més aviat millor el nou codi als usuaris. Una pipeline es ot fer:

- Manualment => Es pot desenvolupar codi de qualitat però passar-lo a producció serà un procés lent i poc eficient.
- Automàticament => Es pot desenvolupar codi de qualitat i passar-lo a producció de manera ràpida, eficient i segura.

Jenkins és una eina molt popular, multiplataforma, integrable amb Github, amb bona documentació, de codi lliure i que permet crear fàcilment pipelines CI/CD parcials o complets amb l'ajut d'una interfície web. És una eina que està disponible en serveis en el núvol com AWS, Azure,etc...

Tot i que implementar un procés complet de CI/CD no és possible per temps i complexitat si que podem fer una implementació mínima que ens permeti copsar el funcionament del procés.

3.- Instal·lació de Jenkins

a) Crea un carpeta de nom **m08uf4pr4** dins de la teva màquina física. Entra a la nova carpeta i crea una carpeta de nom **vm**. Accedeix a **vm** i crea el següent fitxer **Vagrantfile**:

```
IMATGE_BOX = "debian/bookworm64"
NOM_SISTEMA = "jenkins.fjeclot.net"
NOM_VM="m08uf4pr4"
MEMORIA = 2048
CPUS = 2

Vagrant.configure("2") do |config|
  config.vm.box = IMATGE_BOX
  config.vm.hostname = NOM_SISTEMA
  config.vm.network "public_network"
  config.vm.provider "virtualbox" do |v|
    v.name = NOM_VM
    v.memory = MEMORIA
    v.cpus = CPUS
    v.customize ['modifyvm', :id, '--clipboard', 'bidirectional']
  end

  config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update -y
    sudo apt-get install -y net-tools
    sudo apt-get install -y whois
    sudo apt-get install -y aptitude
    sudo apt-get install zip unzip
    sudo apt-get install git
    sudo apt-get -y install apt-transport-https ca-certificates curl gnupg2 software-properties-common
    curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
    sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"
    sudo apt-get update -y
    sudo apt-get -y install docker-ce docker-ce-cli containerd.io docker-compose
    sudo gpasswd -a vagrant docker
    exit
  SHELL
end
```

b) Accedeix a la màquina virtual. Comprova l'adreça IP de la interfície **eth1** de la màquina virtual.

c) Surt de la màquina virtual i modifica el fitxer **hosts** de la teva màquina física per resoldre el nom de sistema de la màquina virtual a l'adreça IP de la nova màquina virtual. Comprova que pots fer ping de la màquina física a la virtual amb el nom de sistema de la màquina virtual.

d) Accedeix a la màquina virtual. Instal·la Jenkins:

```
sudo apt-get update
sudo apt-get install -y openjdk-17-jdk
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo apt-key add -
sudo echo "deb https://pkg.jenkins.io/debian-stable binary/" | sudo tee -a /etc/apt/sources.list
sudo apt-get update -y
sudo apt-get install -y jenkins
```

e) Comprova que l'eina **Jenkins** està funcionant correctament. Executa:

- **systemctl status jenkins | grep Active** => Comprova que està **active (running)** i **enabled**.

f) Comprova amb:

- **sudo systemctl status jenkins** => Contrasenya inicial de l'usuari **admin** d'administració de **Jenkins**. Comprova també el fitxer a on pots trobar de manera alternativa aquesta informació.
- **sudo netstat -atupn | grep java** => Que el port utilitzat per l'aplicació web d'administració de l'eina **Jenkins** és el **8080/tcp**.

g) Surt de la màquina virtual i des de la **màquina física** accedeix a l'aplicació web d'administració de **Jenkins** amb un navegador utilitzant el nom de la màquina virtual, el port pel qual escolta l'aplicació d'administració de Jenkins, el nom d'usuari administrador de Jenkins i la contrasenya inicial.

h) Instal·la el plugins recomanats a la secció *"Install suggested plugins"*. Això pot trigar uns minuts.

i) Crea un compte d'administrador que tingui com a **Username** el nom d'usuari del teu compte de **Github** i **e-mail address** el correu de l'usuari de **Github**. La contrasenya serà **FjeClot2324#**. Com a **Full name** utilitza també el nom d'usuari del teu compte de **Github**.

j) Un cop hagis accedit per primera vegada, surt de l'eina d'administració de Jenkins i torna a accedir amb el nou nom i contrasenyes.

NOTA: En el cas de tenir problemes amb la contrasenya en el futur, haureu de seguir la explicació de com fer un reset de la contrasenya d'administrador aquí.

4.- Prova de funcionament de Jenkins: Sincronització de Jenkins amb Github

4.1 - Preparant un dipòsit de Github per fer proves de funcionament de Jenkins

a) Crea dins de la **màquina física** una carpeta de nom **projectes** i dins de **projectes** una altra de nom **ppv**. Accedeix a **ppv** i crea una nova carpeta de nom **app** i descarrega dins de la carpeta **app** els 2 fitxers d'una aplicació de nom **ppv.html**, **ppv.php** que es troben a un dipòsit de **Githhub**. Executa:

```
wget https://raw.githubusercontent.com/dcolla2/ppv/main/app/index.html
wget https://raw.githubusercontent.com/dcolla2/ppv/main/app/ppv.php
```

i després torna a entrar a **ppv** i descarrega el fitxer **VERSIO** que es troba a:

```
wget https://raw.githubusercontent.com/dcolla2/ppv/main/VERSIO
```

b) Crea dins de **ppv** un fitxer de nom **Dockerfile** amb el següent contingut:

```
FROM php:7.4-apache
WORKDIR /var/www/html
COPY app .
EXPOSE 80
```

c) Inicia un dipòsit de **git** dins de la carpeta **ppv**, i a continuació fes un **add** i un **commit** de la carpeta **app** amb els fitxers de l'aplicació, del fitxer **VERSIO** i de **Dockerfile**. El comentari del commit serà *"Commit 1 de l'aplicatiu ppv"*.

d) Crea un dipòsit **públic** de **Github** de nom **ppv**. A continuació segueix les instruccions que dóna la web de **Github** per sincronitzar el dipòsit local amb el remote i pujar les versions del dipòsit local al remot.

e) Comprova que s'ha pujat les versions del dipòsit local al dipòsit remot.

4.2- Creació d'una tasca e Jenkins dins de la màquina virtual per descarregar els codis de ppv

a) Accedeix a la la pàgina d'administració de **Jenkins** amb l'usuari que vas crear a l'apartat **3**.

b) Selecciona ***+ Item Nova*** per crear un nou projecte CI/CD de **Jenkins**.

c) Escriu a ***Enter an item name*** el nom del projecte: **ppv**.

d) Selecciona la opció ***Freestyle project***

e) Fes clic a **OK** (a la part inferior de la pàgina) i es crearà el projecte.

4.3- Configuració inicial del projecte per establir connexió amb el dipòsit de Github

a) A la secció ***General*** selecciona:
- ***Descripció →*** CI/CD Projecte PPV
- ***Github project → Project url →*** Escriu la URL del teu dipòsit de Github.

b) A la secció ***Gestor del Codi Font*** selecciona:
- ***Git***
- ***Repository URL →*** Escriu la URL del teu dipòsit de Github.
- **Credentials → none**
- ***Branches to build → Branch Specifier (blank for 'any') →*** Esborra master i deixa-ho en blanc per treballar amb qualsevol branca

c) Al final de la pàgina de configuració prem el botó ***Desa*** per desar la configuració.

4.4- Comprovació de la sincronització

a) Accedeix a **Dashboard** → Accedeix la nova tasca **ppv** que ha aparegut.

b) Selecciona l'opció **Construir ara**.

c) Comprova si l'operació ha estat un exit. Si ho ha a la secció **Build History** comprova que el primer Build identificat amb **#1** té la icona i en cas contrari la icona .

d) Si tot ha anat bé, accedeix dins del Build identificat amb **#1** i selecciona **Console Output**. Comprova:
- Que la sincronització s'ha fet amb el Github del teu dipòsit buscant la línia de log que comença per **Cloning repository**.
- Que l'operació ha finalitzat amb un **Finished: SUCCES**
- A la línea **Build in workspace** troba a quin directori de la màquina virtual descarrega **Jenkins** els fitxers del projecte.

e) Finalment, comprova que els fitxers de l'aplicació realment es troben al directori **/var/lib/jenkins/workspaces/ppv** tal i com indica **Jenkins** en els **logs** del build realitzat.

5.- Desplegament manual amb dockers sobre un equip remot d'una primera versió d'una aplicació web a partir del codi pujat a Github utilitzant Jenkins

a) Afegeix l'usuari **jenkins** de la màquina virtual al grup **dockers.** Executa:

sudo gpasswd -a jenkins docker

i a continuació reinicia el servei **jenkins** → **sudo systemctl restart jenkins**

b) Dins de l'eina **Jenkins** selecciona la tasca **ppv** des del **Dashboard**. Accedeix a la secció **Configura**. A la secció **Build Steps** selecciona **Add build step** → **Executa shell**. Crea el següent script:

```
cd /var/lib/jenkins/workspace/ppv
vact=$(cat VERSIO)
if [ -f ./DARRERA_VERSIO_DESPLEGADA ]; then
      vant=$(cat DARRERA_VERSIO_DESPLEGADA)
else
      vant="0.0"
fi
if [ "$vant" != "$vact" ];then
      if [ "$(docker ps -a -q -f name=ppv_docker)" ]; then
            docker stop ppv_docker
            docker rm ppv_docker
            docker rmi ppv_imatge:$vant
      fi
      if [ ! -z "$(docker images -q ppv_imatge:$vant 2> /dev/null)" ]; then
            docker rmi ppv_imatge:$vant
      fi
      docker build -t ppv_imatge:$vact .
      docker run --name ppv_docker -i -t -d -p 80:80 ppv_imatge:$vact
      echo $vact > ./DARRERA_VERSIO_DESPLEGADA
fi
```

Després fes a clic a **Desa** al final de la pàgina.

c) Accedeix a **Dashboard** → Accedeix a la tasca **ppv**. Selecciona l'opció **Construir ara**. Comprova que l'operació ha estat un exit.

d) Dins de la màquina virtual executa:

docker images → Comprova que s'ha creat una imatge de nom **ppv_image:1.0**. A continuació executa: **docker ps -a** → comprova que s'ha creat un contenidor de nom **ppv_docker**.

e) Comprova que pots accedir des de la màquina física a l'aplicació amb l'adreça **http://jenkins:80**.

6.- Desplegament manual amb dockers sobre un equip remot d'una nova versió d'una aplicació web a partir del codi pujat a Github utilitzant Jenkins

a) A la teva **màquina física** canvia el contingut del fitxer i**ndex.html** de l'aplicació:

- A la línia **6**, el nou codi html ha de ser:
  <title> FULLGUEROLA COUNTY FUTBOL LEAGUE - PAY PER VIEW APP - MATCHDAY 2</title>

- A la línia **9**, el nou codi html ha de ser:
  ** SELECT A FUTBOL MATCH - MATCHDAY 2 **

- A la línia **33**, el nou codi html ha de ser:
  **Author username: xxxxxxxx/b> **
  a on xxxxxxxx representa el teu nom d'usuari de **Github**.

b) A la teva **màquina física** canvia el valor dins del fitxer **VERSIO** de **1.0** a **2.0**.

c) A continuació fes un **add**, un **commit** amb el comentari que vulguis i un **push** al teu dipòsit de **Github**.

d) Des de **Jenkins** fes un nou **build**: Accedeix a **Dashboard** → Accedeix a la tasca **ppv**. Selecciona l'opció **Construir ara**. Comprova que l'operació ha estat un exit.

e) Accedeix a l'aplicació i comprova que els canvis han tingut efecte.

7.- Desplegament automatizats amb dockers sobre un equip remot d'una nova versió d'una aplicació web a partir del codi pujat a Github utilitzant Jenkins

a) Dins de la pàgina de configuració del projecte **ppv** de Jenkins, selecciona **Build Triggers** --> **Build Periodically** i programa **Jenkins** perquè amb una periodicitat de **5 minuts, a totes les hores del dia, cada dia de la setmana, cada dia del mes i tots els mesos de l'any** es connecti amb **Github** per descarregar la versió que hi ha pujada i si s'ha fet una actualització de l'aplicació llavors desplegar-la la nova versió.

A la secció **Schedule** escriu:

**H/5 * * * *

Després fes a clic a **Desa** al final de la pàgina.

b) A la teva **màquina física** canvia el contingut del fitxer i**ndex.html** de l'aplicació:

- A la línia **6**, el nou codi html ha de ser:
  <title> FULLGUEROLA COUNTY FUTBOL LEAGUE - PAY PER VIEW APP - MATCHDAY 3</title>

- A la línia **9**, el nou codi html ha de ser:
  ** SELECT A FUTBOL MATCH - MATCHDAY 3 **

c) A la teva **màquina física** canvia el valor dins del fitxer **VERSIO** de **2.0** a **3.0**.

d) A continuació fes un **add**, un **commit** i un **push** al teu dipòsit de **Github**. Escriu el comentari del **commit** que vulguis.

e) Espera a que **Jenkins** es connecti a **Github** i si cal faci automàticament l'actualització de l'aplicació.

f) Comprova que s'han modificat la imatge i els contenidor amb **docker ps -a** i **docker images**.

g) Accedeix a l'aplicació i comprova que els canvis han tingut efecte.

Lliurament de la pràctica

1- Comprovació de la **configuració** del projecte **ppv** de **Jenkins**. Això inclou:

a) Nom del projecte
b) Descripció
c) URL del projecte a la secció General → Github project → Project URL
d) URL del projecte a la Git → Repository URL
e) Script de la secció Build Steps → Add build step → Executa shell
f) Configuració de tasques programades a Build Triggers --> Build Periodically → Schedule

2- Realització d'una modificació d'**index.html** i número de versió dins **VERSIO** a la teva màquina física que s'haurà de pujar al teu dipòsit de **Github**. Comprovació que des de **Jenkins** i **manualment** s'actualitza l'aplicació amb la nova versió.

3- Realització d'una altra modificació d'**index.html** i número de versió dins **VERSIO** a la teva màquina física que s'haurà de pujar al teu dipòsit de **Github**. Comprovació de que **Jenkins** posa en marxa de manera automàtica una actualització de l'aplicació amb la nova versió.

4- Comprovació des de la màquina física que la nova versió està en marxa accedints des del navegador.

Data de lliurament: a partir del dia **21-2-24**.