Fase 4 - Activitat 11.1: CI/CD - Part I: Conceptes bàsics. Instal·lació de Jenkins.

0- Identificació del grup i activitat:

Curs: ASIX2
Projecte: GP2 DevOps i Cloud Computing
Fase: 4
Activitat: 11.1
Grup:
Membres:

1- Introducció i objectius de l'activitat 11.1

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...

3.- Instal·lació de Jenkins

a) Crea un carpeta de nom **gp1f4act11** a la carpeta **projectes** 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_NODE = "pipelinecicd"
MEMORIA = 2048
CPUS = 2
TARGETA_XARXA = "xxxxxxxxx"

Vagrant.configure("2") do |config|
  config.vm.box = IMATGE_BOX
  config.vm.hostname = NOM_NODE
  config.vm.network "public_network", bridge: TARGETA_XARXA
  config.vm.provider "virtualbox" do |v|
    v.name = NOM_NODE
    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 -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
```

A on hauràs de canviar xxxxxxxxx pel nom de la teva targeta de xarxa dins del teu equip físic com ho van fer a l'activitat **gp1f4a9.2**.

b) Inicia i accedeix a la màquina virtual executant. 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 install -y openjdk-17-jdk
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo apt-key add -
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** => El nom d'usuari administrador i la seva contrasenya inicial. Comprova també el fitxer a on pots trobar de manera alternativa aquesta informació.
- **sudo netstat -atupn | grep java** que pots veure el port que utilitza l'aplicació web d'administració de l'eina **Jenkins**.

g) Surt de la màquina virtual i accedeix 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 virtual una carpeta de nom **pizzes**. Accedeix-hi i descarrega dins de la carpeta els 2 fitxers d'una aplicació de nom **pizza.html** i **pizza.php** que es troben a un dipòsit de **Githhub**. Executa:

```
wget https://raw.githubusercontent.com/globproj2/pizza/main/pizza.html
wget https://raw.githubusercontent.com/globproj2/pizza/main/pizza.php
```

b) Fes la **configuració inicial** de **git** dins de la màquina virtual: Executa (sense sudo):

```
git config --global user.email "xxxxxxxx"    # "xxxxxxxxx" és el correu del teu compte de Github)
git config --global user.name "yyyyyyy"    # "yyyyyyy" és el teu nom d'usuari de Github)
```

c) Inicia un dipòsit de **git** dins de la carpeta **pizzes**, i a continuació fes un **add** i un **commit** dels dos fitxers de l'aplicació. El comentari del commit serà *"Commit 1 de l'aplicatiu pizzes"*.

d) Crea un dipòsit **públic** de **Github** de nom **pizzes**. 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'un projecte CI/CD de Jenkins

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: **pizzes**.

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

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

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

a) A la secció ***General*** selecciona:
- ***Descripció →*** CI/CD pizzes
- ***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.
- ***Branches to build → Branch Specifier (blank for 'any') →*** En blanc per treballar amb qualsevol branca

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

2.3- Comprovació de la sincronització

a) Accedeix a **Dashboard** → **pizzes**.

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, vol dir que Jenkins pot sincronitzar-se amb Github. Dins del Build identificat amb #1 selecciona Console Output i comprova:
- Que la sincronització amb Github és amb el teu dipòsit i que tot ha anat bé
- A on ha descarregat Jenkins els fitxer del projecte.

e) Finalment, comprova que els fitxers de l'aplicació realment es troben al directori indicat per Jenkins.

d) Entra dins d'un **Build** que hagi funcionant correctament i comprova que s'ha establert una connexió via **Git** amb el dipòsit demanat i la branca **main**.

Lliurament de l'activitat

a) Mostra l'adreça IP i nom de la màquina virtual.
b) Mostra el fitxer hosts de la màquina física.
c) Comprova que pots accedir a la pàgina d'administració de Jenkins des de la teva màquina física.
d) Mostra el dipòsit pizzes de Github.
e) Mostra el projecte pizzes de Jenkins.
f) Mostra que el build #1 (o posterior) ha funcionat correctament
g) Mostra el el build #1 ha establert una connexió amb el teu dipòsit de Github i ha descarregat els fitxers.
h) Data límit per obtenir el 100% de la nota: **dimecres 11-12-23** a les **17.45**.