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

0- Identificació del grup i activitat:

Curs: ASIX2
Projecte: PJ9 DevOps i Cloud Computing
Fase: 4
Activitat: 11.1
Grup/Individual: Individual
Membres/Alumne:

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 el codi en un dipòsit, es fan les comprovacions necessàries per assegurar-se del seu correcte funcionament. Tot i que un codi funcioni en fase de desenvolupament, això no assegura que en fase de producció l'aplicació funcioni correctament.
- Distribució/Desplegament: El codi passa les proves per comprovar si funciona correctament en fase de producció, i si és així, es pot enviar a producció per l'aplicació o actualitzacions disponibles 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.
- Un cop pujades les modificacions d'un desenvolupador, es validen per garantir que els canvis funcionen i no provoquen errors.
- Si les modificacions no funcionen o provoquen errors, s'ha d'informar al programador per solucionar els problemes i tornar a començar el procés.
- Les comprovacions i l'enviament d'informes al programador es poden automatitzar.

La Distribució/Desplegmanet Continua o Continuous Delivery/Deployment (CD) permet

- Comprova que les noves actualitzacions o aplicacions poden funcionar a l'entorn de producció abans de fer-lo disponible als usuaris.
- Si les modificacions no funcionen o provoquen errors en producció, s'haurà d'informar al programador per solucionar els problemes i tornar a començar el procés.
- Permet desplegar l'aplicació en producció per fer-lo disponible als usuaris si passa totes les proves.
- Les comprovacions i el desplegament es poden automatizar.

La metodologia de treball CI/CD (Continuous Integration + Continuous Delivery/Deployment) té com a objectiu automatitzar al màxim els passos indicats anteriorment de manera que:

- 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.- Creació de màquines virtuals

a) Crea una carpeta de nom **a11** dins de la carpeta **f4** que es troba a **pj9**. A continuació, dins de la carpeta **a11** crea una carpeta de nom **pj9f4a11.1**.

b) Crea un fitxer **Vagrantfile** amb aquest contingut:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

################################
#### Definició de variables ####
################################

IMATGE_BOX_NODES = "debian/bookworm64"
PROVIDER = "virtualbox"
NOM_BASE_OPS="produccio"
NOM_BASE_DEV="desenvolupament"
NOM_DOMINI_NODES=".clotfje.net"
MEMORIA_RAM_OPS = 2048
MEMORIA_RAM_DEV = 1024
NUM_CPUS_OPS = 2
NUM_CPUS_DEV = 1
TARGETA_XARXA = "enp8s0f1"

Vagrant.configure("2") do |node|

  ### Definició de la màquina de producció###

  node.vm.define "produccio" do |prod|
    prod.vm.box = IMATGE_BOX_NODES
    prod.vm.hostname = NOM_BASE_OPS+NOM_DOMINI_NODES
    prod.vm.network "public_network", bridge: TARGETA_XARXA
    prod.vm.provider "virtualbox" do |provprod|
      provprod.name = NOM_BASE_OPS
      provprod.memory = MEMORIA_RAM_OPS
      provprod.cpus = NUM_CPUS_OPS
      provprod.customize ['modifyvm', :id, '--clipboard', 'bidirectional', '--groups', '/PIPELINE']
    end
  end

  ### Definició de la màquina de desenvolupament###

  node.vm.define "desenvolupament" do |dev|
    dev.vm.box = IMATGE_BOX_NODES
    dev.vm.hostname = NOM_BASE_DEV+NOM_DOMINI_NODES
    dev.vm.network "public_network", bridge: TARGETA_XARXA
    dev.vm.provider "virtualbox" do |provdev|
      provdev.name = NOM_BASE_DEV
      provdev.memory = MEMORIA_RAM_DEV
      provdev.cpus = NUM_CPUS_DEV
      provdev.customize ['modifyvm', :id, '--clipboard', 'bidirectional', '--groups', '/PIPELINE']
    end
  end

  ### Aprovisionament de les màquines ###
  node.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update -y
    sudo apt-get install -y net-tools whois aptitude git zip unzip curl
  SHELL
end
```

A on hauràs de canviar **xxxxxxxxxx** pel nom de la teva targeta de xarxa dins del teu equip físic (igual que ho hem fet per exemple a l'activitat **9.1** apartat **3.1** punt **b**).

b) Inicia i accedeix a la màquina virtual executant. Comprova l'adreça IP de les interfícies **eth1** de les màquines virtual. Comprova també els noms de les màquines virtuals.

c) Surt de les màquines virtuals i modifica el fitxer **hosts** de la teva màquina física per resoldre els noms de sistema de les màquines virtuals a les seves adreça IP. Comprova que pots fer ping de la màquina física a les virtuals utilitzant els seus noms de sistema.

4.- Instal·lació de Jenkins dins de la màquina de producció

a) Accedeix a la màquina virtual de **producció**. Instal·la **Jenkins** dins de la màquina executant:

```
sudo apt-get install -y openjdk-17-jdk gnupg
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
```

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

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

c) Comprova amb:

- **sudo systemctl status jenkins** => La contrasenya inicial per accedir l'eina **Jenkins**. Comprova també indica el lloc i nom del fitxer a on pots trobar de manera alternativa aquesta informació.
- **sudo netstat -atupn | grep java** el port que utilitzat per accedir a l'eina web d'administració de l'eina **Jenkins**.

d) 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.

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

f) Crea un compte d'administrador que tingui com a **Username** el nom d'usuari del teu compte de **Github**, com a contrasenya **FjeClot2425#**, Com a **Full name** utilitza també el nom d'usuari del teu compte de **Github** i com **e-mail address** el correu de l'usuari de **Github**.

g) 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.

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

5.1 - Preparant un projecte dins de la màquina de desenvolupament, crea un dipòsit Git local i sincronitza-ho amb un dipòsit de Github.

a) Crea dins de la màquina virtual desenvolupament 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 de **desenvolupament**. Executa (sense sudo):

git config --global user.email "xxxxxxxx" a on **"xxxxxxxxx"** és el correu del teu compte de **Github**
git config --global user.name "yyyyyyy" a on **"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 a la secció **…or push an existing repository from the command line**.

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

5.2- Creació d'un projecte CI/CD de Jenkins dins de la màquina de producció

a) Accedeix a la la pàgina d'administració de **Jenkins** amb l'usuari que vas crear a l'apartat **4**.

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.

5.3- Configuració el projecte CI/CD de Jenkins per establir connexió amb el dipòsit de Github

a) A la secció ***General*** selecciona:
- ***Descripció →*** Pipeline 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.
- **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.

5.4- 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**, el primer Build identificat amb **#1** tindrà una icona verda al costat indicat que ha funcionat:

En cas contrari sortirà una icona vermella d'error:

d) Si tot ha anat bé, vol dir que **Jenkins** pot sincronitzar-se amb **Github**. Comprova:

- Dins de la secció **pizzes → Espai de treball** que s'han descarregat els fitxers dins del sistema.
- Comprova dins del Build **#1 → Console Output → Building in workspace** que els fitxers es descarreguen dins del directori **/var/lib/jenkins/workspace/pizzes**.
- Comprova dins del Build **#1 → Console Output → git config remote.origin.url** que els dipòsit remot de Github és el correcte.
- La darrera línia indica que el procés ha finalitzat amb èxit (SUCCESS)

Lliurament de l'activitat

a) Mostra en la màquina de **desenvolupament** el projecte sincronitzat amb **Github**. Dins de la carpeta **pizzes** executa:

- **ls -ls**
- **git log**
- **git remote -v**

b) Mostra el dipòsit **pizzes** de **Github**.

c) Mostra l'adreça IP de la màquina de **producció**. Executa: **ip addr show eht1**

d) Accedeix a **Jenkins** des de la màquina física utilitzant el **nom** de la màquina virtual de **producció**.

e) Mostra que el build **#1** (o posterior) ha funcionat correctament.

f) Mostra que el el build **#1** ha establert una connexió amb el dipòsit de Github i ha descarregat els fitxers. Comprova accedeix al projecte pizzes i:

- Dins d'**Espai de treball** mostra els fitxers descarregats.
- Dins de **Build #1** → **Console output** → mostra la connexió amb el dipòsit **Github**
- Dins de **Build #1** → **Console output** → mostra que el procés ha finalitzat amb exit.

g) Data límit per obtenir el 100% de la nota: **dimecres 27-1-25** a les **17.45**.