Fase 4 - Activitat 11.2: CI/CD - Part II: Preparant l'entorn de desenvolupament

0- Identificació del grup i activitat:

Curs: ASIX2
Projecte: GP2 DevOps i Cloud Computing
Fase: 4
Activitat: 11.2
Grup:
Membres:

1- Introducció i objectius de l'activitat 11.2

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

2.- Màquina pel desenvolupament d'aplicacions i fer proves d'integració continua

a) Dins de la carpeta nom **gp1f4act11** de la teva màquina física (que va ser creada a l'activitat **gp1f4a11.1** i per tant, ja existeix) crea una nova carpeta de nom **desenvolupaments**.

b) Dins de la carpeta de nom **gp1f4act11** de la teva màquina física una carpeta de nom **vm_desenvolupador**. Accedeix a **vm_desenvolupador** i crea el següent fitxer **Vagrantfile**:

```
IMATGE_BOX = "debian/bookworm64"
NOM_NODE = "desenvolupador"
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.synced_folder "../desenvolupaments", "/home/vagrant/desenvolupaments"

  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.

3.- Codi inicial i dipòsits de l'aplicació per fer proves d'integració continua

a) 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)

b) Inicia i accedeix a la màquina virtual desenvolupador creada a l'apartat anterior. Dins de la màquina virtual, accedeix a la carpeta desenvolupaments i clona un projecte de nom **ipcalc** executant:

git clone https://github.com/globproj2/ipcalc.git

c) Entra dins de la carpeta i**pcalc** que s'ha creat dins de la teva màquina virtual i esborra els remotes de manera que deixin d'apuntar al meu dipòsit. Executa:

git remote remove origin

d) Dins del teu compte de **Github** crea un dipòsit **públic** de nom **ipcalc**.

e) Troba l'adreça URL del teu dipòsit Github a la secció **Quick setup**.

f) Dins de la carpeta **ipcalc** de la teva màquina virtual afegeix l'adreça URL del teu dipòsit. Executa:

git remote add origin <url del teu dipòsit remot>

a on has de canviar per **<url del teu dipòsit remot>** per l'adreça URL del teu dipòsit trobada a l'aparat **d)**.

g) Puja el teu dipòsit local al teu dipòsit remot. Executa: **git push -u origin main**

h) Comprova que s'ha pujat el teu dipòsit local al teu dipòsit remot.

4.- Build i alguns tests de funcionament

4.1- Build

a) Inicia l'aplicació executant dins de la carpeta **ipcalc** de la màquina virtual l'ordre: **docker-compose up -d**

b) Comprova que s'ha creat la imatge de nom **ipcalc_ipcalc:latest** i el contenidor **ipcalc_ipcalc_1**.

4.2- Test End to End

a) Escala l'aplicació. Fes que s'executi sobre 3 contenidors amb l'ordre: **docker-compose scale ipcalc=3** i comprova que s'han creat els contenidor **ipcalc_ipcalc_1** a **ipcalc_ipcalc_3**.

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

c) Accedeix des de la màquina física a l'aplicació amb l'adreça IP trobada al punt anterior.

d) Comprova manulament que l'aplicació dóna els resultats correctes per l'adreça **192.168.1.167/27**.

e) Comprova manualment que surt un missatge d'error correcte si escrius malament l'adreça IP.

f) Si atures els contenidors de l'aplicació amb **docker-compose down** comprova que l'aplicació deixa d'estar disponible.

4.3- Test unitari de funcionament amb PHPUnit

a) Dins de la carpeta **ipcalc** executa la prova de test unitari que ja està preparada dins del fitxer testUnitari.php de l'aplicació. Executa l'orde:

```
docker exec -it ipcalc_ipcalc_1 phpunit /var/www/html/testUnitari.php
```

i comprova que el resultat és:

```
vagrant@desenvolupador:~/desenvolupaments/ipcalc$ docker exec -it ipcalc_ipcalc_1 phpunit /var/www/html/testUnitari.php

PHPUnit 9.6.15 by Sebastian Bergmann and contributors.

...                                                                 3 / 3 (100%)

Time: 00:00.006, Memory: 4.00 MB

OK (3 tests, 6 assertions)
```

Lliurament de l'activitat

a) Mostra el teu dipòsit de Github amb tots els fitxers de l'aplicació.
b) Mostra la imatge **ipcalc_ipcalc:latest** a la teva màquina virtual.
c) Mostra els contenidors **ipcalc_ipcalc_1** a **ipcalc_ipcalc_3** a la teva màquina virtual.
d) Accedeix a l'aplicació des de la màquina física i fes una prova amb l'adreça **192.168.1.167/27**.
e) Executa el test unitari i comprova que funciona correctament.
f) Data límit per obtenir el 100% de la nota: **dimecres 13-12-23** a les **21.00**.