Introducción

En el presente proyecto se pretende instalar una red de servicios segura. Para ello se hará un uso exhaustivo de la tecnología de virtualización, con el fin de instalar todo el ambiente en una sola máquina física. Se creará una LAN virtual que será la que ofrece diversos servicios.

Algunos de los servicios ofrecidos deberán estar cifrados, para lo cual se requerirá autofirmar certificados, es decir, instalar una entidad certificadora propia capaz de firmar sus propios certificados.

Ambiente de desarrollo

- Oracle Virtual Box:

Sistema emulador de Ubuntu para utilizar como sistema operativo.

- Visual Studio Code y vim:

Editor de texto para los códigos fuente y archivos de configuración.

- Amazon Web Services:

Servicio de cloud computing para el WebServer y Database Server.

- Nginx:
- GoDaddy:

Proveedor de dominios.

- Bibliotecas:

minikube kubectl certbot

- Vagrant
- Chef-DK

Diseño

> *DataBase, WebServer y CMS sin contenedores: para los servicios de Database y WebServer sin contenedores, se decició utilizar AWS(Amazon Web Services) debido a la facilidad y eficacia de este, para el Database se utilizó el servicio RDS con una instalación de MySQL Community 8.0 y para el WebServer se utilizó el servicio EC2 con una instalación de Ubuntu 20.04, también se decidió utilizar Wordpress como CMS. CMS con Kubernetes: inicialmente se pensó en utilizar el servicio de EKS de AWS, pero se terminó utilizando una instalación local utilizando la herramienta minikube en Ubuntu 20.04. Acceso público al CMS con nombre de dominio (DNS): para el dominio se utilizó el proveedor de dominios GoDaddy. HTTPS: adicionalmente se utilizó certbot para añadir un certificado HTTPS al WebServer.*

Instalación de Servicios

RDS: Una vez creada la instancia de RDS con MySQL se accede a la instancia y se realizan las siguientes configuraciones:

```
Crear base de datos para Wordpress:
  mysql -h redes-proyecto.c6n74ykw4921.us-east-2.rds.amazonaws.com -uroot -p"password"
  create database wordpress;
```

EC2: Una vez creada la instancia de EC2 con Ubuntu 20.04 se accede a la instancia y se realizan las siguientes configuraciones (descargar el .pem para tener acceso por ssh al servidor):

Instalación Nginx:

```
      sudo apt update
        sudo apt install nginx
        sudo systemctl stop nginx.service
        sudo systemctl start nginx.service
        sudo systemctl enable nginx.service
```

Instalación MySQL Client:

```
      sudo apt install mysql-client
```

Instalar PHP:

```
      sudo apt-get install software-properties-common
        sudo add-apt-repository ppa:ondrej/php
        sudo apt update
        sudo apt install php7.4-fpm php7.4-common php7.4-mysql php7.4-gmp php7.4-curl
php7.4-intl php7.4-mbstring php7.4-xmlrpc php7.4-gd php7.4-xml php7.4-cli php7.4-zip
```

Descargar e instalar Wordpress:

```
      cd /tmp
      wget https://wordpress.org/latest.tar.gz
      tar -xvzf latest.tar.gz
      sudo mv wordpress /var/www/wordpress
```

Dar permisos a usuario www-data para wordpress:

```
      sudo chown -R www-data:www-data /var/www/wordpress/
          sudo chmod -R 755 /var/www/wordpress/
```

Configurar Wordpress:

```
      sudo vim /etc/php/7.4/fpm/php.ini

      file_uploads = On
      allow_url_fopen = On
      short_open_tag = On
      memory_limit = 256M
      cgi.fix_pathinfo = 0
      upload_max_filesize = 100M
      max_execution_time = 360
      date.timezone = America/Chicago
```

Configurar Nginx:

```
    sudo vim /etc/nginx/sites-available/wordpress

    server {
      listen 80;
      listen [::]:80;
      root /var/www/wordpress;
      index  index.php index.html index.htm;
      server_name proredes.xyz;

      client_max_body_size 100M;
      autoindex off;
      location / {
          try_files $uri $uri/ /index.php?$args;
      }

      location ~ \.php$ {
          include snippets/fastcgi-php.conf;
          fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
          fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
          include fastcgi_params;
      }
  }

  sudo ln -s /etc/nginx/sites-available/wordpress /etc/nginx/sites-enabled/
  sudo systemctl restart nginx.service
```

DNS:

> *En AWS, se accede al EC2 y se selecciona la instancia del WebServer, se obtiene el Public IPv4 address y luego se accede a la cuenta de GoDaddy y se configura el dominio para que utilice la Public IPv4 address de la instancia EC2. El dominio utilizado es: proredes.xyz*

HTTPS:

> *Accesar por SSH el servidor, utilizando privilegios sudo Verificar versión de snapd: sudo snap install core; sudo snap refresh core Remover certbot-auto o cualquier otro paquete de Certbot OS Instalar certbot: sudo snap install --classic certbot sudo ln -s /snap/bin/certbot /usr/bin/certbot Obtener el certificado: sudo certbot certonly --nginx*

Kubernetes:

- Instalar minikube en Ubuntu 20.04
- Crear archivos .yaml con su respectivo contenido (archivos adjuntos) -mysql-deployment.yaml -wordpress-deployment.yaml
- Crear archivo kustomization.yaml y añadir el siguiente contenido (reemplazar password):

```
secretGenerator:
- name: mysql-pass
  literals:
  - password=YOUR_PASSWORD
```

```
resources:
  - mysql-deployment.yaml
  - wordpress-deployment.yaml
```

- Aplicar el archivo de kustomization

```
kubectl apply -k ./
```

- Verificar secrets:

```
kubectl get secrets
```

- Verificar los volumenes persistentes:

```
kubectl get pvc
```

- Verificar el estado de los pods:

```
kubectl get pods
```

- Verificar que el servicio wordpress está en ejecución:

```
kubectl get services wordpress
```

- Obtener el url del servicio:

```
minikube service wordpress --url
```

##Servicios con Docker-Compose Docker: La instalación de Docker y sus derivados como Docker-Compose es muy sencillo, ya que el sitio web tiene una guía detallada sobre todo lo necesario y los comandos que debemos seguir para una instalación exitosa. https://docs.docker.com/compose/install/

MySQL: Nos hemos apoyado en un tutorial muy práctico para montar el servidor MySQL en un contenedor de Docker-Compose, donde podemos ver la sintáxis necesaria para nuestro docker-compose.yml y crear un usuario root https://linuxhint.com/mysql_docker_compose/

WordPress: En la misma documentación de Docker, podremos encontrar la guía para realizar la instalación del servicio WordPress en un contenedor Docker-Compose. https://docs.docker.com/samples/wordpress/

OwnCloud: Se utilizó una guía, la cual con un poco de configuración en cuanto a los directorios, se logró implementar el servicio OwnCloud en un contenedor local. https://www.osradar.com/deploy-owncloud-using-docker-compose/

VPN con PaaS

Se deberá de instalar los siguientes programas:

https://downloads.chef.io/tools/infra-server
https://downloads.chef.io/tools/workstation
https://www.vagrantup.com/docs/installation

Configuración de chef:

Creamos una carpeta

```
mkdir vpn
```

Inicializamos las máquinas en vagrant:

```
vagrant init bento/ubuntu-18.04
vagrant plugin install vagrant-omnibus
```

Chef generate repo chef:

```
rm -rf chef/cookbooks/example/
rm -rf chef/data_bags/example/
rm -rf chef/environments/example.json
rm -rf chef/roles/example.json

chef generate cookbook chef/cookbooks/vpn-cookbook -b
chef generate recipe chef/cookbooks/vpn-cookbook vpn-recipe
```

Dentro de la receta colocamos el siguiente código:

```
apt_update 'update'
package 'apache2' do
  action:install
end

package 'openvpn' do
  action:install
end
```

En el vagrant file escribimos lo siguiente:

```
  Vagrant.configure("2") do |config|
    config.omnibus.chef_version = :latest
    config.vm.box = "bento/ubuntu-18.04"
    config.vm.network "forwarded_port", guest: 80, host: 8082
    config.vm.network "forwarded_port", guest: 1195, host: 1195

    chef_repo_path = "./chef"

    config.vm.provision :chef_solo do |chef|
        chef.install = false
        chef.arguments = "--chef-license accept"
        chef.data_bags_path = 'chef/data_bags'
        chef.environments_path = 'chef/environments'
        chef.roles_path = 'chef/roles'
        chef.cookbooks_path = 'chef/cookbooks'
        chef.run_list = [
          'recipe[vpn-cookbook::vpn-recipe]'
        ]
    end

  end
```

En el berkfile debemos de meter las dependencias:

```
source 'https://supermarket.chef.io'
  cookbook 'vpn', '~> 0.4.0'
metadata
```

Para correr la máquina debemos de hacer lo siguiente:

```
  vagrant up
  vagrant provision
  vagrant ssh
```

Terraform

You have to add your ssh key to the digital ocean account and create an API key. After getting the API key, you have to create an environment variable call DIGITALOCEAN_TOKEN with the value of the API.

```
terraform {
  required_providers {
    digitalocean = {
        source = "digitalocean/digitalocean"
    }
  }
}

# Get ssh key
data "digitalocean_ssh_key" "terraform" {
  name = "terraform"
}

# Create a droplet
resource "digitalocean_droplet" "ubuntu" {
    image = "ubuntu-18-04-x64"
    name = "ubuntu"
    region = "nyc1"
    size = "s-1vcpu-1gb"
    private_networking = true
    ssh_keys = [
      data.digitalocean_ssh_key.terraform.id
    ]
}

provider "chef" {
  server_url = "https://api.chef.io/organizations/example/"

  # You can set up a "Client" within the Chef Server management console.
  client_name  = "terraform"
  key_material = "${file("chef-terraform.pem")}"
}
```

Actividades realizadas por estudiante

Nota: Se asume que todas las actividades descritas en esta sección fueron desarrolladas en partes iguales por cada uno de los integrantes del grupo.

- Miércoles 9 de Junio - 6pm
    - Reunión General y prevista del proyecto.
- Viernes 11 de Junio - 6pm
    - Investigación de contenido y asignación de responsabilidades.
- Miércoles 16 de Junio - 6pm
    - Nos reunimos para confirmar si las asignaciones son posibles de realizar para cada miembro del proyecto antes de continuar a la etapa de desarrollo.
- Miércoles 23 de Junio - 6pm
    - Reunión para mostrar avances, posibles contratiempos y sugerencias
- Martes 29 - Miércoles 30 de Junio
    - Reuniones finales para coordinar entregables, compartir resultados y analizar metas así como objetivos faltantes, se crean los entregables para tenerlos listos en el drive de la clase.

Autoevaluación

- Emanuelle: [2] Aprendizaje de Servicios. [3] Aprendizaje de Central Telefónica. [3] Aprendizaje de IaC. [4] Aprendizaje de PaaS.
- Fabrizio: [4] Aprendizaje de Servicios. [5] Aprendizaje de Central Telefónica. [2] Aprendizaje de IaC. [3] Aprendizaje de PaaS.
- Kevin: [4] Aprendizaje de Servicios. [3] Aprendizaje de Central Telefónica. [2] Aprendizaje de IaC. [3] Aprendizaje de PaaS.
- Marco: [5] Aprendizaje de Servicios. [2] Aprendizaje de Central Telefónica. [3] Aprendizaje de IaC. [2] Aprendizaje de PaaS.

Lecciones Aprendidas

- Emanuelle: Considero que este proyecto es de bastante utilidad porque nos ayuda a poder crear nuestro propio medio coorporativo utilizando la virtualidad, este nos permitió a nosotros conocer sobre varias tecnologías que desconocemos, como fue en el caso de un centro telefónico, donde nunca nos imaginamos que este podría llegar a ser un proyecto en nuestra carrera en el TEC. Es interesante observar cómo todos estos servicios funcionan en sí y cómo se pueden integrar utilizando varios paradigmas de infraestructura, como los es PaaS e IaC.
- Fabrizio: Creo que el proyecto fue de bastante provecho al trabajar un poco con las tecnologías que realmente se utilizan en la calle, tales como telefonía móvil, vpn, contenedores, etc. En lo personal ya había trabajado con algunas de ellas mas no había tenido la oportunidad de configurarlas para su uso o similar, por lo que me resultó sumamente interesante. Finalmente, creo que un buen consejo a futuros estudiantes que deban realizar la asignacion es basicamente investigar bastante y conprer bastante bien cada una de las secciones de los servicios, para evitar más que todo errores inesperados o similares.

- Kevin: Siento que el alcance del proyecto estuvo bien calculado, compartimos distintas opiniones respecto a la asignación del trabajo y siempre es bueno estar haciendo reuniones con los compañero para ir verificando el progreso de todos así como para entender lo que cada uno está investigando y poder complementar el conocimiento general, y lo unico que puedo recomendarle a las futuras generaciones es, que investiguen bien sobre los diferentes servicios y no se encierren con uno solo, ya que tal vez unos serán más fáciles de trabajar que otros y si están bajo presión eso les puede salvar el día.
- Marco: Me gustó como distribuimos las tareas a realizar, realizamos un buen trabajo en equipo, por mi parte, disfruté aprender a utilizar servicios de AWS y Kubernetes, también fue interesante ver la cantidad de servicios que se pueden desarrollar, por lo general uno tiene noción de unos pocos, inclusive me llamó la atención que uno puede desarrollar su propio VPN e incluso una central telefónica. A futuras generaciones le recomiendo no estresarse al ver estos temas, es cuestión de investigar y tomar el tiempo para comprender los servicios.

Bibliografía

- https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/
- https://minikube.sigs.k8s.io/docs/tutorials/multi_node/
- https://aws.amazon.com/free
- https://certbot.eff.org/lets-encrypt/ubuntufocal-nginx
- https://docs.docker.com/compose/install/
- https://linuxhint.com/mysql_docker_compose/
- https://docs.docker.com/samples/wordpress/
- https://www.osradar.com/deploy-owncloud-using-docker-compose/
- https://www.asterisk.org/
- https://www.sip.us/
- https://docs.chef.io/workstation/install_workstation/
- https://docs.chef.io/chef_install_script/