apsl

[De 0 a 100 en 60 segundos]

Pequeña introducción a Terraform y AWS [EC2 y S3]

Índice

- **Amazon Web Services**
  - Pequeña Introducción
  - **EC2**
  - **EBS**
  - **S3**
    - Introducción
    - Características
    - Tipos
    - Ventajas y desventajas
    - Detalles
- **Terraform**
  - Introducción
  - Características
  - Ventajas y desventajas
  - Operativa Básica
- **DEMO TIME!!!**
  - Despliegue Instancia EC2 y bucket S3

Amazon Web Services

- Filial de la empresa Amazon.com que provee un gran abanico de servicios cloud
- “Pay-as-you-go”
- Usado por particulares, empresas e incluso gobiernos (GovCloud)
    - Buscadores (DuckDuckGo, Ecosia)
    - Grandes empresas (Endesa, Orange, BBVA)
- Gran cantidad de data centers repartidos por el mundo
    - Apertura de tres en España para el 2022


Regiones
Próximamente


Las localizaciones en Aragón
aws
aws
1 Plataforma logística (Plhus) en Huesca
2
Junto al aeródromo de Villanueva de Gállego
3
aws
Polígono El Espartal en El Burgo de Ebro

```
→ dig cdn.ecosia.org
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.11.5-P4-5.1+b1-Debian <<>> cdn.ecosia.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65282
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1452
;; QUESTION SECTION:
;cdn.ecosia.org.                        IN      A

;; ANSWER SECTION:
cdn.ecosia.org.         211     IN      CNAME   d3fa767y2tlbh.cloudfront.net.
d3fa767y2tlbh.cloudfront.net. 52 IN     A       52.85.47.173
d3fa767y2tlbh.cloudfront.net. 52 IN     A       52.85.47.191
d3fa767y2tlbh.cloudfront.net. 52 IN     A       52.85.47.44
d3fa767y2tlbh.cloudfront.net. 52 IN     A       52.85.47.66

;; Query time: 51 msec
;; SERVER: 10.253.22.222#53(10.253.22.222)
;; WHEN: dom dic 15 13:30:50 CET 2019
;; MSG SIZE  rcvd: 149


~
→ ipi 52.85.47.173
{
  "ip": "52.85.47.173",
  "hostname": "server-52-85-47-173.mad50.r.cloudfront.net",
  "city": "Ashburn",
  "region": "Virginia",
  "country": "US",
  "loc": "39.0437,-77.4875",
  "org": "AS16509 Amazon.com, Inc.",
  "postal": "20149",
  "timezone": "America/New_York",
  "readme": "https://ipinfo.io/missingauth"
}%
```

```
                                My traceroute  [v0.93]
northernlights (192.168.15.95)                                  2019-12-16T09:26:47+0100
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                  Packets               Pings
 Host                                          Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. router.lan                                  0.0%     5    0.9   1.1   0.5   1.5   0.4
 2. 192.168.144.1                               0.0%     5    3.5   3.2   2.0   5.5   1.4
 3. 125.red-81-41-222.staticip.rima-tde.net     0.0%     5    4.1   3.4   2.6   4.1   0.6
 4. 242.red-81-41-222.staticip.rima-tde.net     0.0%     5   11.9  11.7  11.2  12.0   0.3
 5. 97.red-80-58-106.staticip.rima-tde.net      0.0%     5   10.4  10.8  10.1  11.9   0.8
 6. ae1-400-gramadte3.net.telefonicaglobalsolutions.com  0.0%  5  12.2  12.9  12.2  13.7  0.7
 7. 176.52.248.194                              0.0%     5   18.8  15.6  12.4  20.4   3.7
 8. so2-1-0-0-grtmiabr2.net.telefonicaglobalsolutions.com  0.0%  5  12.3  11.6  10.5  12.3  0.7
 9. 52.93.93.116                                0.0%     5   13.9  12.8  11.7  13.9   1.0
10. 52.93.93.131                                0.0%     5   14.8  18.2  14.8  25.7   4.4
11. 54.239.41.64                                0.0%     5   15.3  15.2  11.9  19.3   2.7
12. 52.93.17.126                                0.0%     5   17.7  17.9  15.1  20.1   1.9
13. 52.93.17.231                                0.0%     5   11.4  11.7  11.4  12.2   0.4
14. (waiting for reply)
15. (waiting for reply)
16. (waiting for reply)
17. (waiting for reply)
18. (waiting for reply)
19. server-13-224-106-73.mad50.r.cloudfront.net 0.0%     4   12.2  12.3  12.2  12.6   0.2
```

Ventajas

- Gran cantidad de servicios (no paran de sacar)
  - Bases de datos
  - Computación en la nube
  - Almacenamiento
- Low Cost
- Pagas por lo que usas
- Escalabilidad
- Uso sencillo (una vez conoces los servicios)
- Capacidad “ilimitada”
- Plataforma rápida, segura y fiable
- Nos ahorramos el mantenimiento del hierro y operaciones complejas

Desventajas

- Facturación puede llegar a ser confusa (requiere buena revisión)
- El soporte al cliente tiene diferentes tipos y además de pago
    - Developer
    - Business
    - Enterprise
- Algunos servicios EC2 están limitados
    - A muy gran escala

No es magia, hay un gran equipo humano detrás de todo

EC2 (*Elastic Compute Cloud*)

Servicio de AWS que nos permite lanzar instancias (máquinas virtuales) de forma rápida y escalable en la nube

Ventajas

- Diferentes tipos de instancias para multitud de usos
- Intel, AMD e incluso ARM
- Flexibilidad a la hora de añadir storage
- Uso de IP's elásticas
- Configuración y despliegues rápidos a partir de AMIs (*Amazon Machine Images*)
- Diversidad de AMIs con variedad de sistemas operativos (*Amazon Linux*, *Ubuntu*, *Windows Server*…)
- Mantenimiento mínimo (*no tocamos el hierro*)
- Gran capa de seguridad por parte de AWS
- Reserva de instancias (*descuento si las reservamos durante un tiempo determinado*)

Desventajas

- En instancias estándar (no dedicadas) compartimos el servidor con más gente, por tanto nuestro rendimiento se puede ver afectado
- El rendimiento de red viene limitado por el tipo de instancia
- No tenemos el mando de todo

Tipos de instancias

- **Uso general** (*a1*, *t3*, *t3a*, *m5…)* : equilibrio entre CPU, memoria y red. Ideales para aplicaciones con cargas balanceadas.
- **Optimizadas para CPU** (*c5*, *c4*...) : optimizadas para cargas de CPU intensivas, alto rendimiento a bajo coste
- **Optimizadas para memoria** (*r5*, *r5a*, *x1*) : diseñadas para ofrecer un rendimiento rápido en cargas de trabajo que procesan grandes conjuntos en memoria
- **Computación Acelerada** (*p3*, *g4*, *f1*) : utilizan aceleradores de hardware o coprocesadores para realizar funciones, como el cálculo o procesamiento de gráficos.
- **Optimizadas para almacenamiento** (*i3*, *d2*): pensadas para cargas de trabajo que necesitan acceso de escritura y lectura secuencial altos.

Para más información : https://ec2instances.info/

EBS (*Elastic Block Storage*)

- Servicio de almacenamiento que usamos en EC2
- Pensado para cargas de cualquier escala
- Diferentes tipos de storage
- Capacidad de guardar snapshots
- Cifrado
- Asignación y redimensión en caliente

Tipos de volúmenes EBS

- **Unidades de estado sólido** (*SSD*)
  - **SSD de IOPS provisionadas** (*io1*) : uso intensivo de operaciones de “E/S” -> [I/O]
  - **SSD de uso general** (*gp2*) : uso balanceado (volumen arranque sistema, apps…)
- **Discos Duros** (*HDD*)
  - **HDD optimizados para procesamiento** (*st1*) : cargas de trabajo de procesamiento intensivo a las que se accede con frecuencia
  - **HDD fríos** (*scl*) : cargas de trabajo a las que se accede con menos frecuencia

Detalles a tener en cuenta

- Se asocia una “Key Pair” a las instancias para el acceso por SSH, esta clave es muy valiosa y la debemos guardar bajo llave.
- Las instancias a nivel de red forman parte de una VPC (*Virtual Private Cloud*) y de una subnet específica.
- Si no asignamos una IP elástica a nuestra instancia, se nos asignará una IP pública dinámicamente.
- Las IP’s elásticas no tienen coste siempre que las estés usando.
- Usaremos los Security Groups para limitar el acceso a nuestra instancia (viene a ser un firewall)

S3 (*Simple Storage Service*)

Servicio de almacenamiento de objetos

- Creado para almacenar y recuperar cualquier volumen de datos desde cualquier ubicación
- A diferencia de los sistemas de almacenamiento tradicionales (bloque, fichero), aquí se tratan como objetos
- Añade metainformación e identificador único a cada objeto

Características

- Escalabilidad (no tenemos un límite de datos fijo)
- Disponibilidad de datos (99,99% anual)
- Seguridad (encriptación y cifrado de transferencia)
- Buen rendimiento
- Gran durabilidad (99,999999999%)

Tipos de almacenamiento

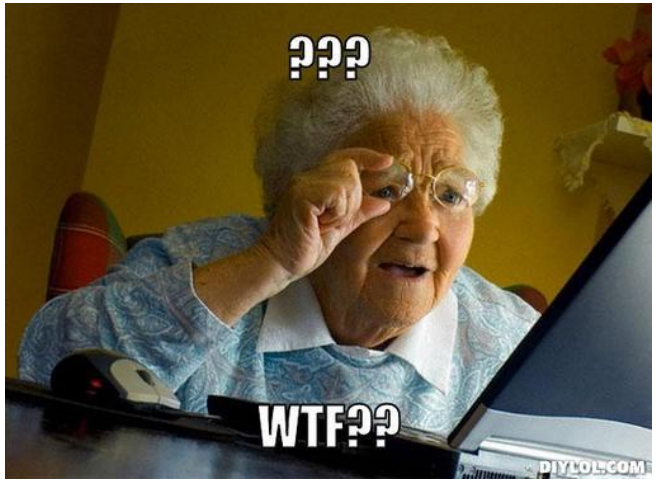
- **Estándar** : pensado para almacenar datos a los que se accede con frecuencia
- **Inteligente** : diseñado para optimizar costos mediante la migración automática de los datos a la capa de acceso más rentable, sin afectar a rendimiento
- **Estándar** (*acceso poco frecuente*) : datos a los que se accede con poca frecuencia
- **Única zona** (*acceso poco frecuente*) : datos a los que se accede con poca frecuencia, solo en una única zona de disponibilidad, más barato.
- **Glacier** : almacenamiento seguro, duradero y de bajo costo para el archivado de datos, ideal si no necesitas el fichero inmediatamente
- **Glacier Deep Archive** : admite retención a largo plazo y conservación de datos a los que se accede una o dos veces al año

Ventajas

- Fácil distribución del contenido
- Gran integración con aplicaciones de terceros
- Nos permite servir estáticos
- Permite crear redirecciones
- Alta disponibilidad en todas las zonas (dependiendo de la clase)
- Podemos monitorizar el uso del bucket (pagando)
- Precio económico

Desventajas

- El precio de recuperación de ficheros en clases tipo Glacier o S3 IA puede ser alto
- La interfaz web podría ser mejorable
- No podemos descargar directorios subidos como .zip



Ahora que conocemos el provider...

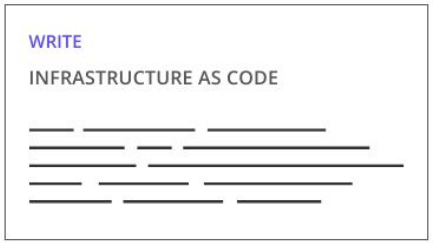
andrés calla ya queremos ver terraform

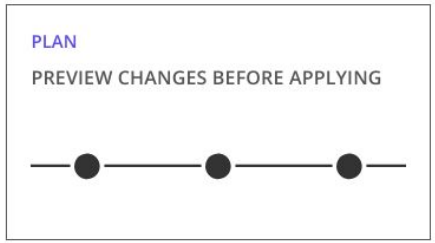
Terraform

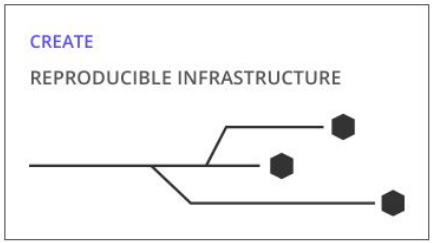
- Herramienta que nos permite crear, modificar y versionar nuestra infraestructura de forma segura y eficiente.
- Código abierto y desarrollado por Hashicorp (*gran combo*)
- Nos permite tratar nuestra infraestructura como código
- Gestión del grafo de recursos; gestiona la dependencia entre recursos y paraleliza la creación y modificación de recursos no dependientes (“*sabe como casarlo todo*”)
- Construye nuestra infraestructura de forma eficiente y en un tiempo reducido

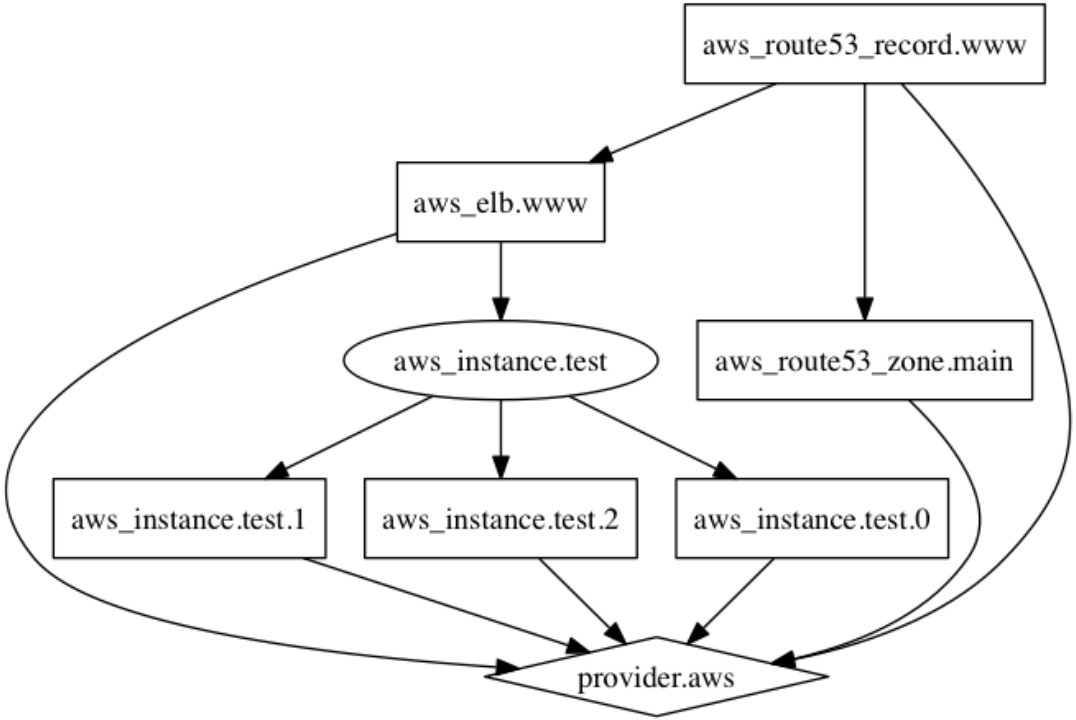
SIMPLE AND POWERFUL

HashiCorp Terraform enables you to safely and predictably create, change, and improve infrastructure. It is an open source tool that codifies APIs into declarative configuration files that can be shared amongst team members, treated as code, edited, reviewed, and versioned.








aws_route53_record.www
aws_elb.www
aws_instance.test
aws_route53_zone.main
aws_instance.test.1
aws_instance.test.2
aws_instance.test.0
provider.aws

Ventajas

- Es un binario ejecutable sin dependencias (descargar y usar)
- Definimos cómo queremos que sea nuestra infraestructura y él la crea por nosotros
- Mantenemos el estado de la infraestructura como código guardado en nuestro repo
- Podemos importar infraestructura que no teníamos definida como código
- Lenguaje declarativo y simple
- Gran cantidad de providers (AWS, Google Cloud, Azure)
- Planes de ejecución para saber que vamos a crear/destruir
- Mínima interacción para grandes cambios

Desventajas

- Al igual que despliegas tu infraestructura de forma muy rápida también la puedes acabar liando de forma igual de rápida
- El fichero de estado *(.tfstate)* por normal general se guarda en local, y puede ser fácil perderlo

Definición de la infraestructura

- **Config** : configuración de providers
- **Resource**: manejar el ciclo de vida de un recurso
- **Data** : información remota de un proveedor
- **Module** : instanciación de un módulo
- **Variable** : definición de variables
- **Locals** : variables locales
- **Outputs** : salidas de información hacia cli o otros módulos

Ejemplo resource

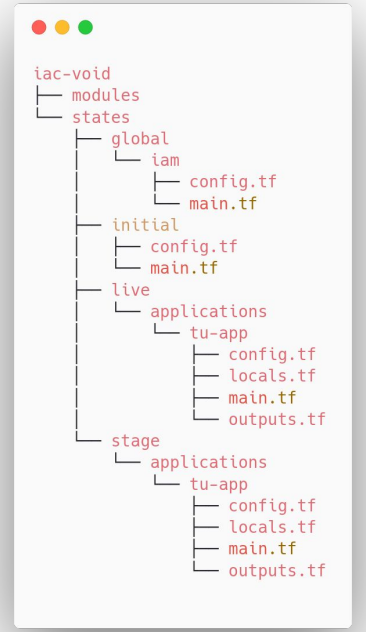
```
resource "aws_s3_bucket" "emili-darder-website" {
  bucket = "emili-darder-cracks"
  acl    = "public-read"

  website {
    index_document = "index.html"
    error_document = "error.html"
  }

  tags = {
    Name  = "Emili Darder - Website"
    Class = "Máquinas"
  }
}
```

Jerarquía recomendada

- **Initial** -> recursos iniciales para almacenamiento de estados remotos (s3 estados, DynamoDB concurrencia)
- **Global** -> definición de estados comunes entre entornos
- **Live** -> recursos para producción
- **Stage** -> recursos para preproducción



Estados Terraform

- Fichero json (*.tfstate*)
- Información completa de todos los recursos remotos creados. Metainformación de lo que se ha ejecutado.
  - Versión de terraform
  - Provider
  - Estado de los recursos de la última vez que se ejecutó
- Por defecto se guarda en local, aunque también podemos guardarlo remotamente (bucket s3)
- Permite control de concurrencia, evitar que se ejecuten dos comandos terraform con el mismo estado pero con definiciones diferentes, para ello se usa una base de datos compartida (*DynamoDB*)

```
{
  "version": 4,
  "terraform_version": "0.12.18",
  "serial": 24,
  "lineage": "a0d1a3d7-2949-164e-0c0b-6c7d7a474377",
  "outputs": {},
  "resources": [
    {
      "mode": "data",
      "type": "aws_ami",
      "name": "ubuntu",
      "provider": "provider.aws",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "architecture": "x86_64",
            "block_device_mappings": [
              {
                "device_name": "/dev/sda1",
                "ebs": {
                  "delete_on_termination": "true",
                  "encrypted": "false",
                  "iops": "0",
                  "snapshot_id": "snap-00283bf187e442699",
                  "volume_size": "8",
                  "volume_type": "gp2"
                },
                "no_device": "",
                "virtual_name": ""
              },
.......
```

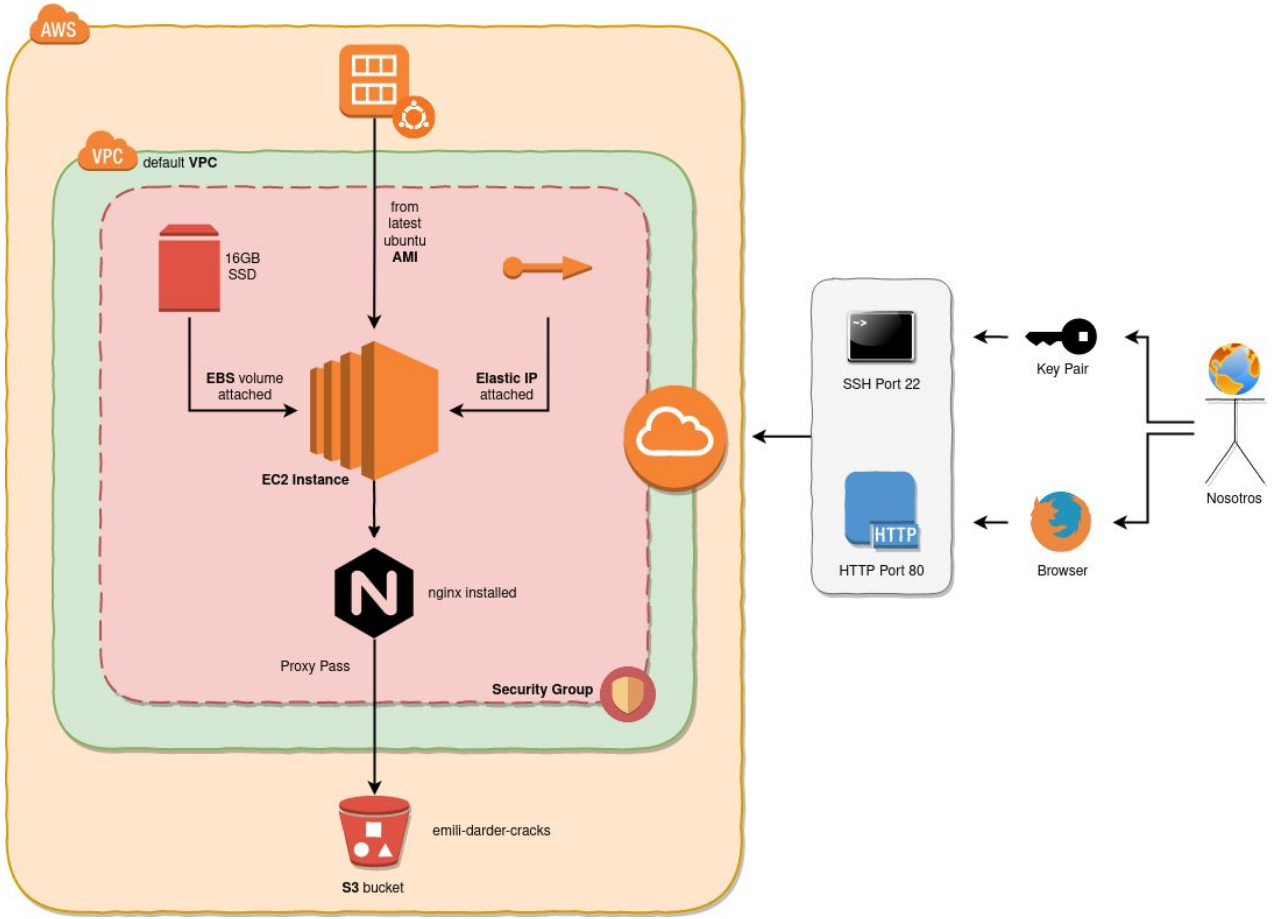
Operativa básica

- **terraform init** : inicializa la infraestructura, carga los módulos remotos y resources de providers
- **terraform plan** : comprueba las diferencias entre el estado real y el estado almacenado, te dice que va a crear y cómo lo va a crear
- **terraform apply** : hace un plan y espera nuestra confirmación para aplicar los cambios
- **terraform destroy** : hace un plan y espera nuestra confirmación para destruir los recursos. Destruye todos los recursos que tuviésemos definidos en el estado.
- **terraform import** : permite importar un recurso existente a nuestro estado actual (nuestra IaC)
- **terraform validate** : comprueba la sintaxis de los ficheros

Si no hay ninguna pregunta...
Empezamos con la DEMO

Escenario DEMO

1. Despliegue bucket S3
2. Despliegue instancia EC2
3. Acceso a la instancia por SSH y visualización del bucket por HTTP falseando /etc/hosts
4. Probaremos la potencia de la máquina usando Hashcat


AWS
VPC
default VPC
from
latest
ubuntu
AMI
16GB
SSD
EBS volume
attached
Elastic IP
attached
EC2 Instance
nginx installed
Proxy Pass
Security Group
emili-darder-cracks
S3 bucket
SSH Port 22
HTTP Port 80
HTTP
Key Pair
Browser
Nosotros


¡¡¡AL TURRÓN!!!

¿Preguntas?

[Gracias]