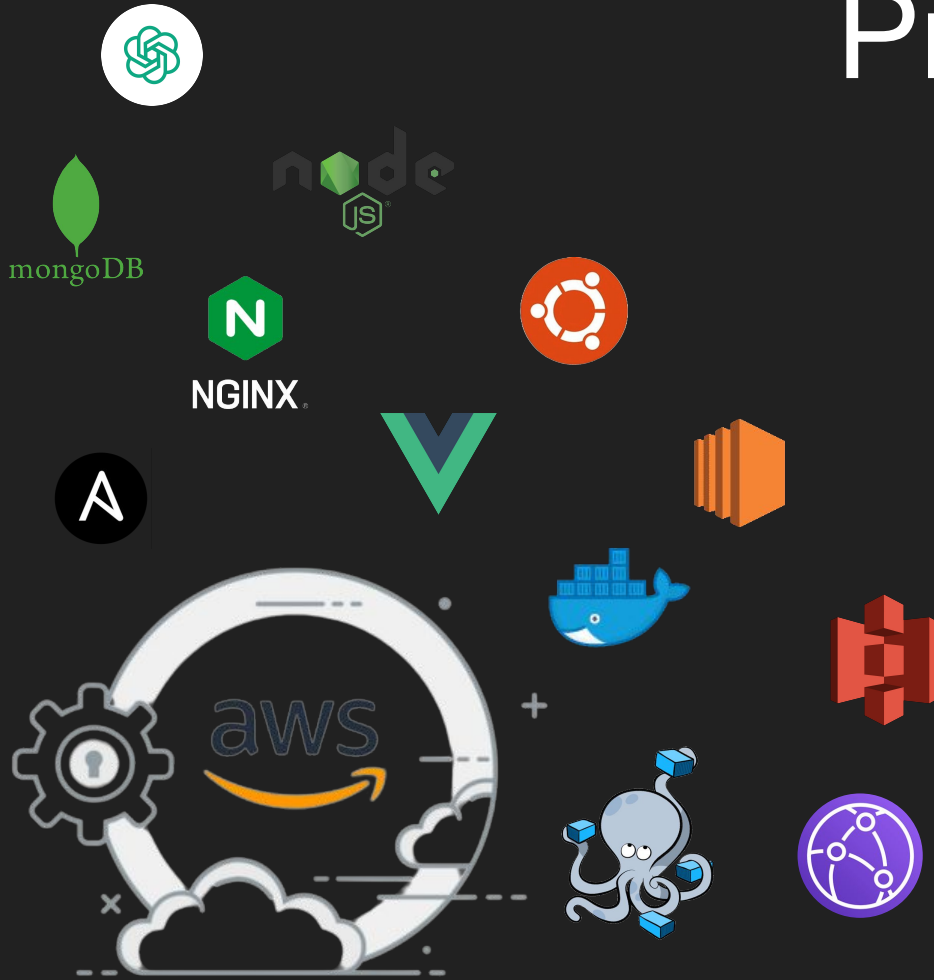


Proyecto Primer Parcial



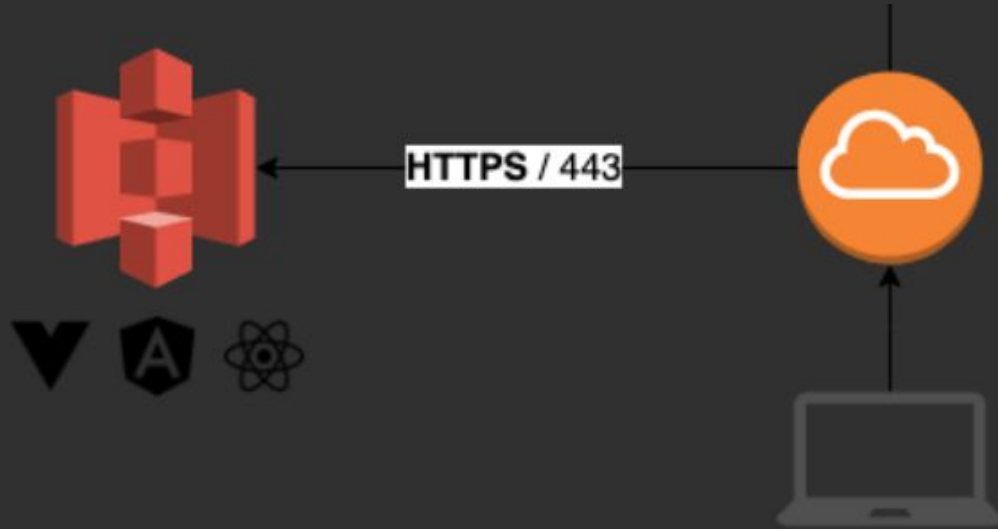
338832 Juan Ángel Cepeda

330836 Manuel Balderrama

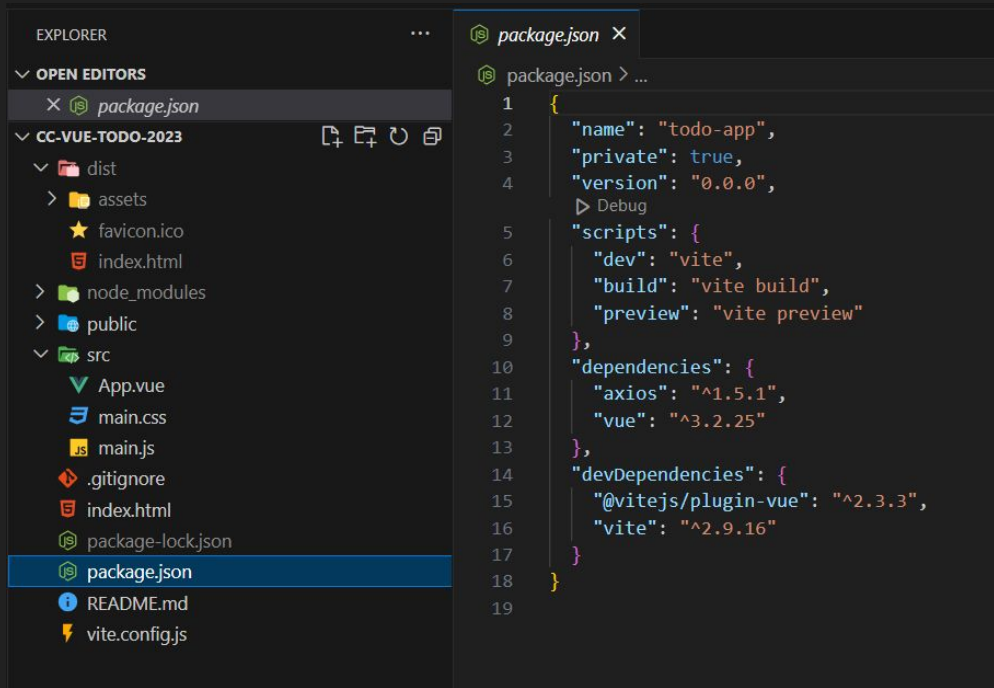
334144 Eduardo Rodriguez

334083 Miguel Cortinas

1. Web estático desplegado en S3



Hacemos Nuestro front con Vue y vite



Creamos la carpeta dist con el comando “build”, en esta carpeta se encuentran los archivos que se suben al bucket de S3 de amazon, aquí se encuentra nuestro sitio estático del to-do programado en src/App.vue, main.css y main.js



Buckets (1) [Información](#)

Los buckets son contenedores de datos almacenados en S3. [Más información](#)

< 1 > ⚙️

	Nombre ▲	Región de AWS ▼	Acceso ▼	Fecha de creación ▼
<input type="radio"/>	my-bucket-23-2	EE. UU. Este (Norte de Virginia) us-east-1	Público	27 Sep 2023 8:40:18 AM MST

Objetos (3)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

< 1 > ⚙️

<input type="checkbox"/>	Nombre ▲	Tipo ▼	Última modificación ▼	Tamaño ▼	Clase de almacenamiento ▼
<input type="checkbox"/>	assets/	Carpeta	-	-	-
<input type="checkbox"/>	favicon.ico	ico	27 Sep 2023 12:09:24 PM MST	4.2 KB	Estándar
<input type="checkbox"/>	index.html	html	27 Sep 2023 12:09:24 PM MST	455.0 B	Estándar

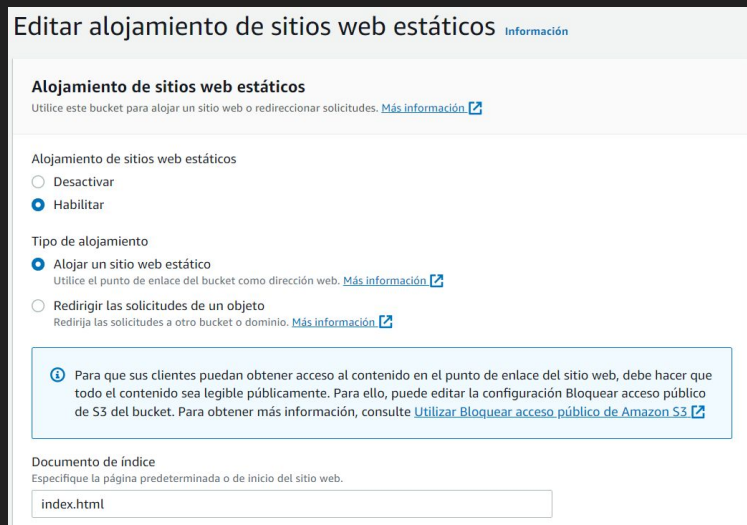
Creamos un bucket de amazon S3, dentro de este bucket cargamos nuestros archivos de la carpeta dist de nuestro proyecto



Habilitar el bucket para sitios estáticos



Nos dirigimos a propiedades, bajamos hasta el apartado de Alojamiento de sitios estáticos, habilitamos la opción, seleccionamos “Alojar un sitio web estático” y escribimos el documento índice, en este caso es index.html



Configuración de permisos públicos

my-bucket-23-2

Información

Accesible públicamente

Objetos

Propiedades

Permisos

Métricas

Administración

Puntos de acceso

Información general sobre los permisos

Acceso

 Público

Bloquear acceso público (configuración del bucket)

Se concede acceso público a buckets y objetos a través de listas de control de acceso (ACL), políticas de bucket, políticas de pu
bloquee el acceso público a todos sus buckets y objetos de S3, active Bloquear todo acceso público. Esta configuración se apli
recomienda activar Bloquear todo acceso público pero, antes de aplicar cualquiera de estos ajustes, asegúrese de que sus apli
necesita cierto nivel de acceso público a sus buckets u objetos, puede personalizar los valores de configuración individuales a
específicas de almacenamiento. [Más información](#)

Editar

Bloquear *todo* el acceso público

 Desactivado

Entramos a la sección de permisos. Nos dirigimos a Bloquear acceso público, debemos de permitir el acceso para que se pueda acceder al sitio web, luego configuraremos CloudFront para añadir https, por el momento comprobaremos que funciona



Configuración de permisos públicos

Alojamiento de sitios web estáticos
Utilice este bucket para alojar un sitio web o redirigir solicitudes. [Más información](#)

Editar

Alojamiento de sitios web estáticos

Habilitada

Tipo de alojamiento

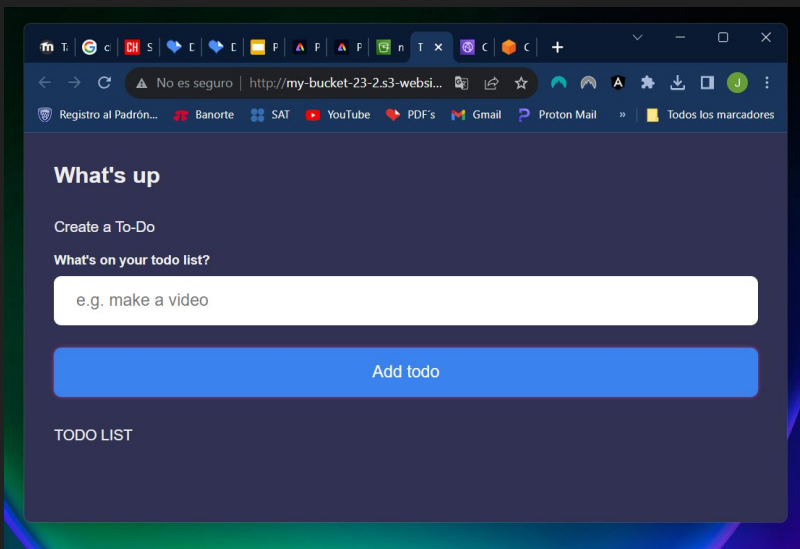
Alojamiento de buckets

Punto de enlace de sitio web del bucket

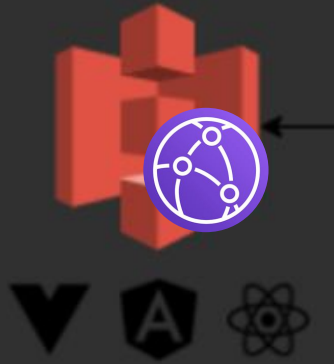
Al configurar su bucket como sitio web estático, el sitio web estará disponible en el punto de enlace del sitio web específico de la región de AWS del bucket. [Más información](#)

<http://my-bucket-23-2.s3-website-us-east-1.amazonaws.com>

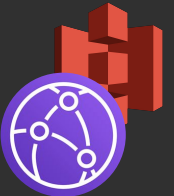
Dentro de la pestaña de propiedades de nuestro bucket, bajamos hasta la sección de Alojamiento de sitios web, nos proporciona un link, vamos hacia él y nos debería de mostrar nuestro sitio desplegado si todo funciona correctamente, Podemos notar que está funcionando con http



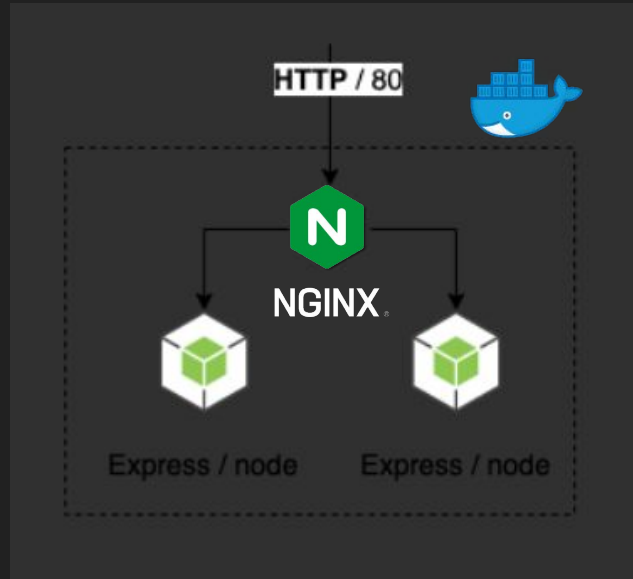
Comprobar que nuestro sitio web funcione con https



<http://my-bucket-23-2.s3-website-us-east-1.amazonaws.com/>



2. Dockerización de Back-End con balanceo de cargas



Configuración de Backend

```
App.vue JS task.js Extension: Vue Language Features (Volar)
nyapp > models > JS task.js > ...
1  const mongoose = require('mongoose');
2
3  const schema = mongoose.Schema({
4    _name: String,
5    _done: Boolean
6  });
7
8  class Task {
9    constructor(name, done){
10      this._name = name;
11      this._done = done;
12    }
13
14    get name(){
15      return this._name;
16    }
17    set name(v){
18      this._name = v;
19    }
20
21    get done(){
22      return this._done;
23    }
24    set done(v){
25      this._done = v;
26    }
27  }
28
29  schema.loadClass(Task);
30  module.exports = mongoose.model('Task', schema);
```

Crear el modelo task para generar los to-do's



mongoDB

Configuración de Backend

```
▼ App.vue • JS tasks.js • Extension: Vue Language Features (Volar)
myapp > controllers > JS tasks.js > create
1  const express = require('express');
2  const Task = require('../models/task');
3
4  function list(req, res, next) {
5    Task.find().then(objs => res.status(200).json({
6      message: "Ok",
7      objs: objs
8    })).catch(ex => res.status(500).json({
9      message: "Error List",
10     obj: ex
11   }));
12 }
13
14 > function index(req, res, next) { ...
15 }
16
17 > function create(req, res, next) { ...
18 }
19
20 > function replace(req, res, next) { ...
21 }
22
23 > function update(req, res, next) { ...
24 }
25
26 > function destroy(req, res, next) { ...
27 }
28
29 module.exports = {
30   list,
31   index,
32   create,
33   replace,
34   update,
35   destroy
36 };
```

Creamos el controller tasks



mongoDB

Configuración de Backend

App.vue JS tasks.js ...\controllers JS tasks.js ...\routes X

myapp > routes > JS tasks.js > ...

```
1 var express = require('express');
2 var router = express.Router();
3 const controller = require('../controllers/tasks');
4
5 router.get('/', controller.list);
6
7 router.get('/:id', controller.index);
8
9 router.post('/', controller.create);
10
11 router.put('/:id', controller.replace);
12
13 router.patch('/:id', controller.update);
14
15 router.delete('/:id', controller.destroy);
16
17 module.exports = router;
18
```

Creamos las rutas



mongoDB

Configuración de Backend

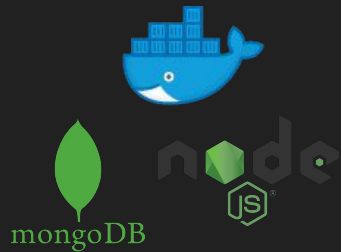
```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
const mongoose = require('mongoose');
const cors = require('cors');

const indexRouter = require('./routes/index');
const tasksRouter = require('./routes/tasks');

mongoose.connect('mongodb://18.217.86.175:27017');
const db = mongoose.connection;
var app = express();

db.on('open', ()=>{
  console.log('ok');
})
db.on('error', ()=>{
  console.log('Not ok');
})
```

Conectamos el back con
mongo



Configuración de Backend

myapp > Dockerfile > ...

```
1 FROM node
2 WORKDIR /app
3 COPY . .
4 RUN npm install
5 ENV INSTANCE=1
6 ENV PORT=3000
7 EXPOSE ${PORT}
8 CMD PORT=${PORT} INSTANCE=${INSTANCE} node bin/www
9
```

Creamos nuestro Dockerfile para construir la imagen de nuestro backend



mongoDB



Configuración de Backend

```
myapp > docker-compose.yml
1  version: '2.1.1'
2  services:
3    app-backend-to-do-1:
4      build: .
5      image: to-do-backend-cc-proyect:2.1.1
6      restart: always
7      ports:
8        - 3000:3000
9      environment:
10       INSTANCE: 1
11       PORT: 3000
12    app-backend-to-do-2:
13      build: .
14      image: to-do-backend-cc-proyect:2.1.1
15      restart: always
16      ports:
17        - 3001:3001
18      environment:
19       INSTANCE: 2
20       PORT: 3001
21    balancer:
22      image: nginx
23      volumes:
24        - ./nginx.conf:/etc/nginx/nginx.conf
25      ports:
26        - 80:80
27      depends_on:
28        - app-backend-to-do-1
29        - app-backend-to-do-2
30
```

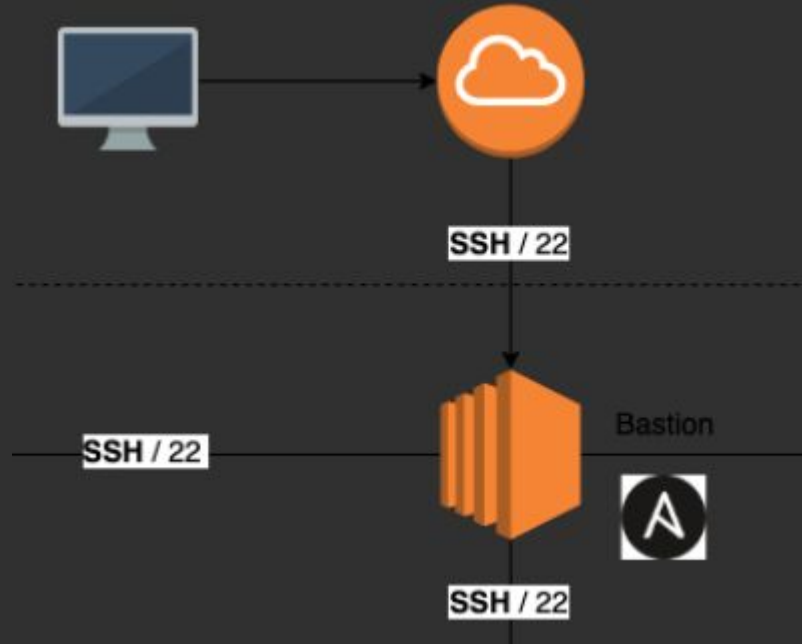
Creamos nuestro docker-compose.yml, para levantar nuestro balanceador y nuestras dos instancias del web server, definimos los puertos de entrada del web server y el balanceador de cargas nginx




mongoDB



3. Configuración de Ec2 para instancia de bastion



Grupos de seguridad

 sg-07ed6ff4a4fa37439 (Bastion)

▼ Reglas de entrada

< 1 >

Nombre	ID de la regla del grupo d...	Intervalo de pu...	Protocolo	Origen
-	sgr-0225d2030604e3ea2	22	TCP	0.0.0.0/0

▼ Reglas de salida

< 1 >

Nombre	ID de la regla del grupo d...	Intervalo de pu...	Protocolo	Destino
-	sgr-0504a85d05e673a29	Todo	Todo	0.0.0.0/0

Creamos una instancia de un servidor ubuntu con las reglas de seguridad de bastión, ese nos servirá para conectarnos a los web servers, nginx y mongo, permite la entrada con llaves pem y desde cualquier lugar con ssh

Nos conectamos a bastión con ssh, actualizamos los paquetes e instalamos Ansible

```
C:\cc-23-2\keys>ssh -i Web-server-23-2.pem ubuntu@3.14.149.136
The authenticity of host '3.14.149.136 (3.14.149.136)' can't be established.
ED25519 key fingerprint is SHA256:TVa0jkibOL7hEMckKfMvx513AGp7F6G8g8YdDJQSeKc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.14.149.136' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)
```

```
ubuntu@ip-172-31-25-251:~$ sudo apt update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
```



Nos conectamos a bastión con ssh, actualizamos los paquetes, instalamos Ansible, python

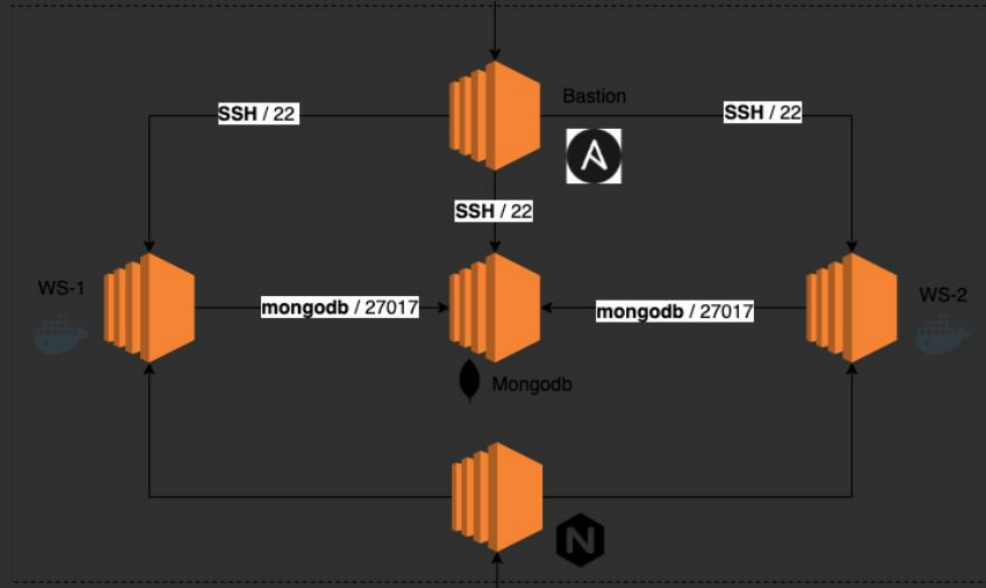
```
ubuntu@ip-172-31-25-251:~$ sudo apt install ansible
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ieee-data python3-argcomplete python3-dnspython python3-jmespath python3-kerberos python3-libcloud python3-lockfile
  python3-netaddr python3-ntlm-auth python3-packaging python3-pycryptodome python3-requests-kerberos
  python3-requests-ntlm python3-requests-toolbelt python3-selinux python3-simplejson python3-winrm python3-xlrd python3-xlsxwriter
Suggested packages:
  cowsay sshpass python3-sniffio python3-trio python-lockfile-doc ipython3 python-netaddr-docs
```



Vamos a crear nuestras instancias de web-apps, base de datos de mongo y balanceador de cargas principal, para luego agregar estas direcciones a host y configurar nuestro playbook



4. Configuración de instancias en AWS



Creamos una instancia llamada nginx, que podrá recibir entradas de ssh desde el bastion y peticiones http desde nuestro front-end

Nombre y etiquetas [Información](#)

Nombre

nginx

[Agregar etiquetas adicionales](#)

▼ Imágenes de aplicaciones y sistemas operativos (Amazon Machine Image) [Información](#)

Una AMI es una plantilla que contiene la configuración de software (sistema operativo, servidor de aplicaciones y aplicaciones) necesaria para lanzar la instancia. Busque o examine las AMI si no ve lo que busca a continuación.

🔍

Busque en nuestro catálogo completo que incluye miles de imágenes de sistemas operativos y aplicaciones

Recientes

Inicio rápido

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu®

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

🔍

Buscar más AMI

Inclusión de AMI de AWS, Marketplace y la comunidad

A collection of logos in the bottom right corner. It includes the MongoDB logo (a green leaf), the AWS logo (a black circle with a white 'A'), and a cartoon octopus holding blue cubes. There are also some orange and red geometric shapes.

Configuramos la seguridad para que reciba solo ssh desde bastion, http y https desde cualquier lugar, la seguridad de la petición se realiza sobre el cors de configuración de nginx.conf

vpc-0bb99aa35a1d6a90f (predeterminado) ↻

Subred **Información**

subnet-099de07c029a1d3a8
VPC: vpc-0bb99aa35a1d6a90f Propietario: 045065688865
Zona de disponibilidad: us-east-2b Direcciones IP disponibles: 4090
CIDR: 172.31.16.0/20 ↻ Crear nueva subred ↗

Asignar automáticamente la IP pública **Información**

Habilitar

Firewall (grupos de seguridad) Información
Un grupo de seguridad es un conjunto de reglas de firewall que controlan el tráfico de la instancia. Agregue reglas para permitir que un tráfico específico llegue a la instancia.

☒ Crear grupo de seguridad ☐ Seleccionar un grupo de seguridad existente

Nombre del grupo de seguridad - obligatorio

nginx

Este grupo de seguridad se agregará a todas las interfaces de red. El nombre no se puede editar después de crear el grupo de seguridad. La longitud máxima es de 255 caracteres. Caracteres válidos: a-z, A-Z, 0-9, espacios y _ - / () #, @ [] += & ; () ! \$ *

Descripción - obligatorio **Información**

nginx-permite-bastion-ssh-http-https-cualquier-lugar

Reglas de grupos de seguridad de entrada

▼ Regla del grupo de seguridad 1 (TCP, 22, 0.0.0.0/0) Eliminar

▼ Regla del grupo de seguridad 2 (TCP, 80, 0.0.0.0/0) Eliminar

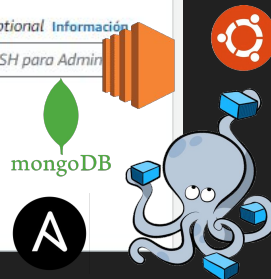
Tipo Información	Protocolo Información	Intervalo de puertos Información
HTTP	TCP	80
Tipo de origen Información	Origen Información	Descripción - optional Información
Cualquier lugar	🔍 Agregue CIDR, lista de prefijos 0.0.0.0/0 ✕	por ejemplo, SSH para Admin Desk

▼ Regla del grupo de seguridad 3 (TCP, 443, 0.0.0.0/0) Eliminar

Tipo Información	Protocolo Información	Intervalo de puertos Información
HTTPS	TCP	443
Tipo de origen Información	Origen Información	Descripción - optional Información
Cualquier lugar	🔍 Agregue CIDR, lista de prefijos 0.0.0.0/0 ✕	por ejemplo, SSH para Admin Desk

Agregar regla del grupo de seguridad

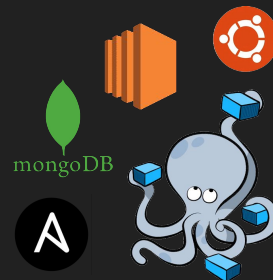
► Configuración de red avanzada





Escribimos nuestro archivo de nginx.config para hacer el balanceo de cargas y configuramos nuestro cors, para recibir solicitudes del front, habilitamos el puerto 80 de la entrada del balanceador de cargas

```
nginx > nginx.conf
1  worker_process auto;
2  pid /run/nginx.pid;
3
4  events {
5      worker_connections 768;
6  }
7
8  http {
9      upstream app{
10         server 172.31.44.127:80;
11         server 172.31.47.132:80;
12     }
13     server{
14         location /{
15             proxy_pass http://app;
16         }
17     }
18
19     add_header Access-Control-Allow-Origin https://d16uqncjh27k17.cloudfront.net;
20     add_header Access-Control-Allow-Methods GET, POST, PUT, DELETE, OPTIONS;
21     add_header Access-Control-Allow-Headers Content-Type, Authorization;
22
23     access_log /var/log/nginx/access.log;
24     error_log /var/log/nginx/error.log;
25
26     gzip on;
27 }
```

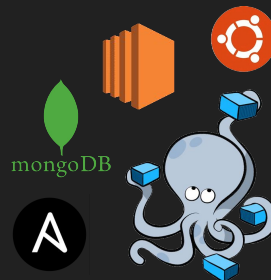


Ahora haremos la configuración de las instancias de los web-sever, solo mostramos la primera, ya que son iguales, solo a la segunda cambiará su nombre con proy-2 igualmente con ubuntu, aqui solo mostraremos el grupo de seguridad, configurando las entradas por bastion y desde el nginx

Nombre y etiquetas [Información](#)

Nombre

[Agregar etiquetas adicionales](#)



Ahora haremos la configuración de las instancias de los web-sever, igualmente con ubuntu, aqui solo mostraremos el grupo de seguridad, configurando las entradas por bastion y desde el nginx configurado anteriormente

Regla del grupo de seguridad 1 (TCP, 22, sg-07ed6ff4a4fa37439) Eliminar

Tipo Información
ssh

Tipo de origen Información
Personalizada

Regla del grupo de seguridad 2 (TCP, 80, sg-07ed6ff4a4fa37439) Eliminar

Tipo Información
HTTP

Tipo de origen Información
Personalizada

Agregar regla del grupo de seguridad

Grupos de seguridad

- backendServer
sg-07c6ec8a076c312d2
- launch-wizard-1
sg-00a9912ff2320898c
- nginx**
sg-01f85feb99c0b62ca
- Servers
sg-0b54aa7170a400c73
- Bastion-sec
sg-0cb7290d15fa7916c
- Bastion
sg-07ed6ff4a4fa37439

por ejemplo, SSH para Admin Desk

Configuración de red avanzada



Por último creamos nuestra instancia de base de datos de mongo, el cual tendrá por seguridad el ssh desde bastión y las entradas de TCP por el puerto default de mongo, el 27017, desde los web-app-servers

Nombre y etiquetas [Información](#)

Nombre

[Agregar etiquetas adicionales](#)

Grupos de seguridad

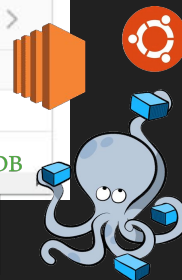
 sg-075c5724423a057fb (mongodb-proy)

▼ Reglas de entrada

Nombre	ID de la regla del grupo d...	Intervalo de pu...	Protocolo	Origen	Grupos de seguridad
-	sgr-047bcb7395e3efa16	22	TCP	sg-07ed6ff4a4fa37439	mongodb-proy
-	sgr-056b97bd185d4b374	27017	TCP	sg-07acb2070fa713bca	mongodb-proy

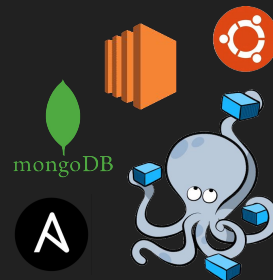
 mongoDB

A

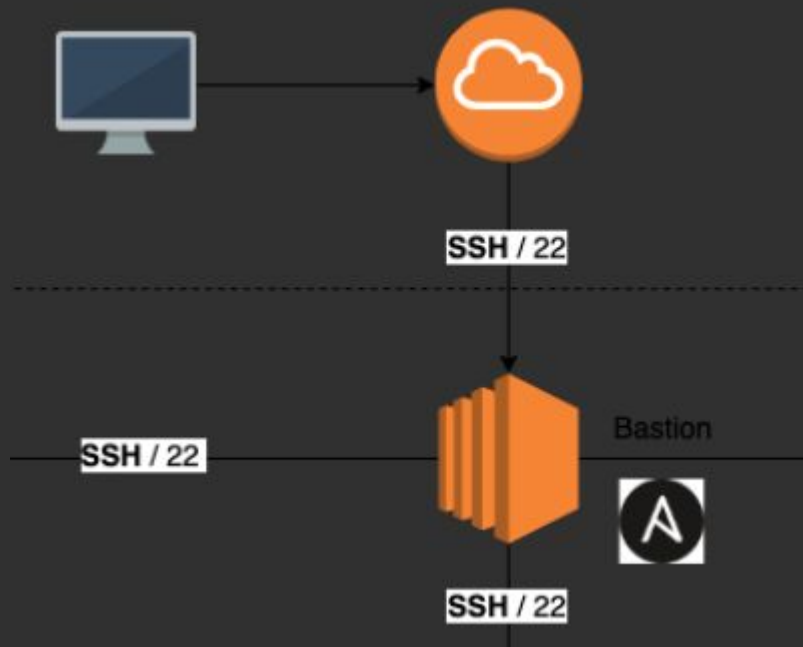


Listo, verificamos nuestras, instancias, estas deben de estar en ejecución, revisaremos sus ip´s privadas para agregar estas configuraciones al host de ansible y poder controlarlas desde ahí, nuestro playbook hará las configuraciones para cada instancia según sea el caso

<input type="checkbox"/>	Name ▾	ID de la instancia	Estado de la i... ▲	Tipo de inst... ▾	Comprobación ...	Estado de la ...	Zona de dispon... ▾	DNS de IPv4 pública ▾
<input type="checkbox"/>	BastionProject	i-08eaecf368d455d1d	✔ En ejecución 🔍 🔍	t2.micro	✔ 2/2 comprobaci...	Sin alarmas +	us-east-2b	ec2-3-14-149-136.us-e...
<input type="checkbox"/>	nginx	i-08ad3a070b6e96bd5	✔ En ejecución 🔍 🔍	t2.micro	✔ 2/2 comprobaci...	Sin alarmas +	us-east-2b	ec2-18-222-216-227.us...
<input type="checkbox"/>	web-server-proy-1	i-017773a5ac0320fd7	✔ En ejecución 🔍 🔍	t2.micro	✔ 2/2 comprobaci...	Sin alarmas +	us-east-2b	ec2-18-219-232-146.us...
<input type="checkbox"/>	web-server-proy-2	i-01bc0927055dd49ca	✔ En ejecución 🔍 🔍	t2.micro	✔ 2/2 comprobaci...	Sin alarmas +	us-east-2c	ec2-18-225-32-250.us-...
<input type="checkbox"/>	mongodb-proy	i-01c8d924674321c86	✔ En ejecución 🔍 🔍	t2.micro	–	Sin alarmas +	us-east-2b	ec2-18-217-86-175.us-...



5. Configuración de ansible en bastion



Pasamos nuestra llave privada al bastión, para que pueda conectarse con los demás servidores

```
ubuntu@ip-172-31-25-251:~$ ls
Web-server-23-2.pem
ubuntu@ip-172-31-25-251:~$ |
```

Configuramos nuestros host, especificando la ruta de la llave, y la dirección ip privada, ya que todos se encuentran en la misma subred

```
ubuntu@ip-172-31-25-251: /€ × + ▾
GNU nano 6.2                                hosts *
[ms-ws1]
172.31.27.225 ansible_user=ubuntu ansible_ssh_private_key_file=/home/ubuntu/Web-server-23-2.pem
[ms-ws2]
172.31.36.4 ansible_user=ubuntu ansible_ssh_private_key_file=/home/ubuntu/Web-server-23-2.pem
[mongodb]
172.31.28.221 ansible_user=ubuntu ansible_ssh_private_key_file=/home/ubuntu/Web-server-23-2.pem
[nginx]
172.31.26.10 ansible_user=ubuntu ansible_ssh_private_key_file=/home/ubuntu/Web-server-23-2.pem
```



Hacemos un ping para verificar la conexión con nuestros servidores

```
ubuntu@ip-172-31-25-251:~$ ls
```

```
Web-server-23-2.pem
```

```
ubuntu@ip-172-31-25-251:~$ |
```

```
ubuntu@ip-172-31-25-251:~$ ansible all -m ping
```

```
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
```

```
172.31.28.221 | SUCCESS => {
```

```
  "ansible_facts": {
```

```
    "discovered_interpreter_python": "/usr/bin/python3"
```

```
  },
```

```
  "changed": false,
```

```
  "ping": "pong"
```

```
}
```

```
172.31.36.4 | SUCCESS => {
```

```
  "ansible_facts": {
```

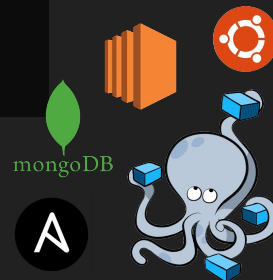
```
    "discovered_interpreter_python": "/usr/bin/python3"
```

```
  },
```

```
  "changed": false,
```

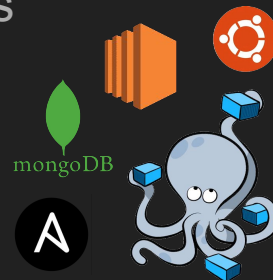
```
  "ping": "pong"
```

```
}
```

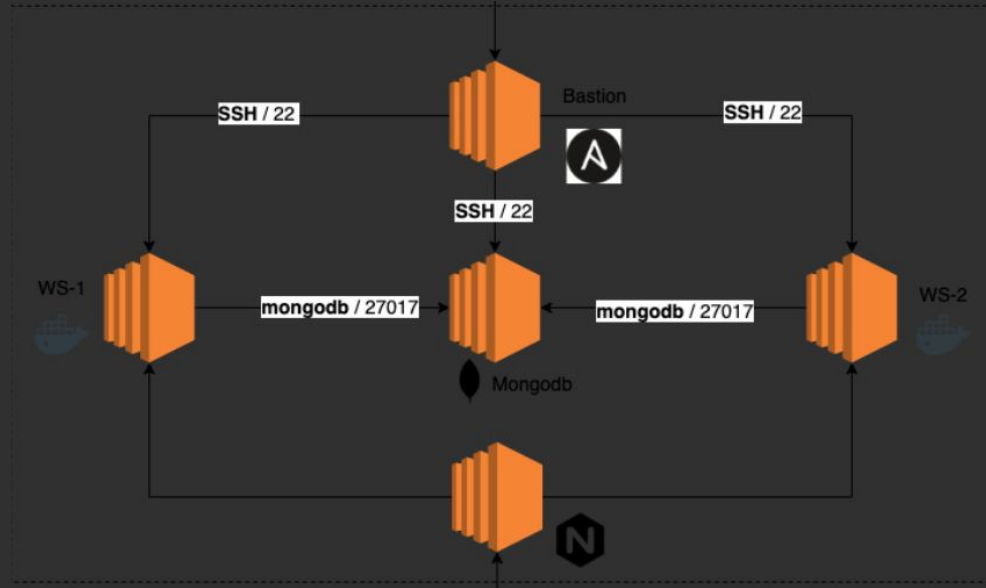



```
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
172.31.26.10 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ubuntu@ip-172-31-25-251:~$ |
```

Como podemos ver nuestros servidores ya se encuentran conectados con nuestro bastión, ya podemos hacer nuestro playbook para levantar los servicios de la infraestructura



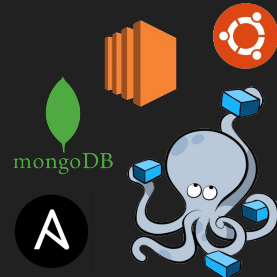
6. Configuración de playbook para servicios



Ahora, hacemos nuestro archivo playbook con las configuraciones necesarias para cada servidor.

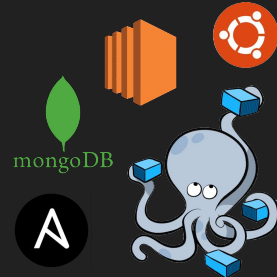
A los cuatro les actualizaremos docker, git, y haremos pull del repo del proyecto

```
playbook.yml
1  - hosts: ['ms-ws1', 'ms-ws2','mongodb','nginx']
2    become: yes
3    tasks:
4      - name: Install curl
5        apt:
6          name: curl
7          state: present
8      - name: Install Docker
9        shell: 'curl -fsSL https://get.docker.com -o get-docker.sh && sudo sh get-docker.sh'
10     - name: Start Docker
11       shell: 'sudo service docker start'
12       ignore_errors: yes
13     - name: Instalar Git
14       shell: 'sudo apt-get install git'
15     - name: Crear carpeta para el proyecto
16     - name: Clonar el repositorio
17       command: chdir=/home/ubuntu/proyecto
18       shell: 'sudo git clone https://github.com/Juan-Angel-Cepeda/cloud-proyect'
```



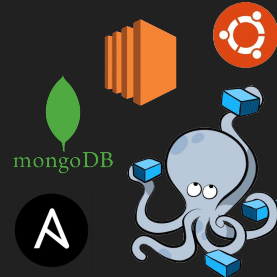
Para los web servers, nos dirigimos a la carpeta donde se encuentra nuestro back-end y hacemos docker-compose-up

```
- hosts: ['ms-ws1', 'ms-ws2']  
  become: yes  
  tasks:  
    - name: Acceder a la carpeta y hacer docker compose up  
      shell: 'cd /home/ubuntu/proyecto/cloud-proyect/myapp && docker compose up -d'
```



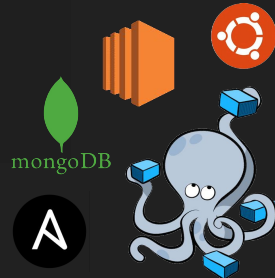
Para la base de datos de mongo, obtenemos la imagen de mongo, iniciamos un contenedor y los ponemos en escucha por el puerto 27017 de docker con el 27017 de nuestra instancia

```
- hosts: ['mongodb']  
  become: yes  
  tasks:  
    - name: Instalar la imagen de mongo  
      shell: 'docker pull mongo'  
    - name: levantar mongo  
      shell: 'docker run -d -p 27017:27017 --name mongoddb mongo'
```



Para nuestro nginx, obtenemos la imagen de nginx, levantamos un contenedor de nginx, ponemos en escucha el puerto 80 del contenedor con el 80 de la instancia, y hacemos un bind volume para la configuración de nginx del proyecto con el nginx.config del contenedor

```
- hosts: ['nginx']
  become: yes
  tasks:
    - name: Instalar la imagen de nginx
      shell: 'docker pull nginx'
    - name: levantar nginx
      docker_container:
        name: nginx
        image: nginx
        ports:
          - 80:80
        volumes:
          - /home/ubuntu/proyecto/cloud-proyect/nginx/nginx.conf:/etc/nginx/nginx.conf
        state: started
```



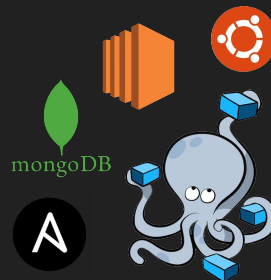
Corremos nuestro playbook

ansible-playbook playbook.yml

```
PLAY [ms-ws1,ms-ws2,mongodb,nginx] *****
TASK [Gathering Facts] *****ok: [172.31.28.221]
ok: [172.31.26.10]
ok: [172.31.27.225]
ok: [172.31.36.4]

TASK [Install curl] *****
ok: [172.31.27.225]
ok: [172.31.36.4]
ok: [172.31.28.221]
ok: [172.31.26.10]

TASK [Install Docker] *****
|
```

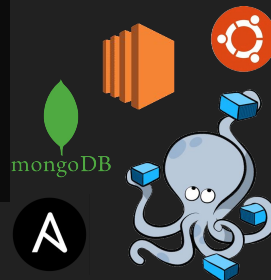


Corremos nuestro playbook

```
TASK [Install Docker] *****
[WARNING]: Consider using the get_url or uri module rather than running 'curl'. If
you need to use command because get_url or uri is insufficient you can add 'warn:
false' to this command task or set 'command_warnings=False' in ansible.cfg to get
rid of this message.
changed: [172.31.28.221]
changed: [172.31.26.10]
changed: [172.31.36.4]
changed: [172.31.27.225]

TASK [Start Docker] *****
[WARNING]: Consider using 'become', 'become_method', and 'become_user' rather than
running sudo
changed: [172.31.28.221]
changed: [172.31.27.225]
changed: [172.31.36.4]
changed: [172.31.26.10]

TASK [Instalar Git] *****
ok: [172.31.27.225]
ok: [172.31.28.221]
ok: [172.31.36.4]
ok: [172.31.26.10]
```



Corremos nuestro playbook

```
TASK [Clonar el repositorio] *****
changed: [172.31.28.221]
changed: [172.31.36.4]
changed: [172.31.26.10]
changed: [172.31.27.225]

PLAY [ms-ws1,ms-ws2] *****

TASK [Gathering Facts] *****
ok: [172.31.27.225]
ok: [172.31.36.4]

TASK [Acceder a la carpeta y hacer docker compose up] *****
changed: [172.31.36.4]
changed: [172.31.27.225]

PLAY [nginx] *****

TASK [Gathering Facts] *****
ok: [172.31.26.10]

TASK [Instalar python] *****
changed: [172.31.26.10]

TASK [Instalar python docker] *****
ok: [172.31.26.10]

TASK [Instalar la biblioteca de python de docker] *****
ok: [172.31.26.10]

TASK [Establecer ansible_python] *****
ok: [172.31.26.10]

TASK [Instalar la imagen de nginx] *****
changed: [172.31.26.10]
```



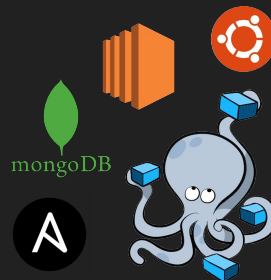
Corremos nuestro playbook

```
TASK [Instalar la imagen de nginx] *****
changed: [172.31.26.10]
```

```
TASK [levantar nginx] *****
[DEPRECATION WARNING]: The container_default_behavior option will change its default
value from "compatibility" to "no_defaults" in community.general 3.0.0. To remove
this warning, please specify an explicit value for it now. This feature will be
removed from community.general in version 3.0.0. Deprecation warnings can be
disabled by setting deprecation_warnings=False in ansible.cfg.
changed: [172.31.26.10]
```

```
PLAY RECAP *****
```

172.31.26.10	: ok=14	changed=7	unreachable=0	failed=0	skipped=0	rescued=0	ignored=1
172.31.27.225	: ok=9	changed=5	unreachable=0	failed=0	skipped=0	rescued=0	ignored=1
172.31.28.221	: ok=7	changed=4	unreachable=0	failed=0	skipped=0	rescued=0	ignored=1
172.31.36.4	: ok=9	changed=5	unreachable=0	failed=0	skipped=0	rescued=0	ignored=1



! Listo

← → ↻ No es seguro | http://my-bucket-23-2.s3-website-us-east-1.amazonaws.com

Registro al Padrón... Banorte SAT YouTube PDF's Gmail Proton Mail Drive Mercado Libre Udemey

To-do

Create a To-Do

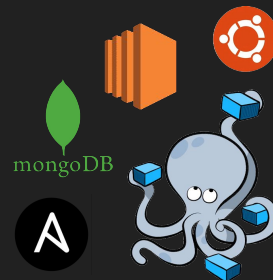
What's on your todo list?

pasa cloud

Add todo

To Do List

No hay To-do's!



! Listo

← → ↻ No es seguro | http://my-bucket-23-2.s3-website-us-east-1.amazonaws.com

Registro al Padrón... Banorte SAT YouTube PDF's Gmail Proton Mail Drive Mercado Libre Udemý AMEX PayPal » Todos los marcadores

Create a To-Do

What's on your todo list?

Add todo

To Do List

pasar cloud Delete

hacer un curso de front Delete

hacer el chat con AWS Delete

pagarle a Jeff Besos por dejarnos usar sus EC2 Delete

