Lleva tu proyecto a produccion

Introduccion a Vagrant

Existen muchas herramientas que nos permiten crear un entorno de produccion como el que nosotros esperamos tener. Esto lo usamos ya que cuando estamos probando tenemos que correr por un lado la base de datos, los microservicios, los servidores y asi, para hacer este proceso mas agradable vamos a utilizar una herramienta llamada Vagrant.

<https://www.vagrantup.com/>

Vagrant maneja proveedores de maquinas virtuales, Este maneja VirtualBox que nos permite crear maquinas virtuales con diferentes sistemas operativos para poder montar nuestro entorno de desarrollo como gustemos.

Nosotros vamos a hacer despliegue a maquina virtual tipo Ulinux que en este caso va a ser Ubuntu.

Entonces lo que hace Vagrant es entregarnos una interfaz para crear una maquina virtua sin tener que crearla manualmente y poder gestionarla facilmente.

Ahora vamos a crear una maquina virtual con Vagrant para amenizar nuestro ambiente de desarrollo y para simular todas las condiciones que vamos a tener en nuestro ambiente de produccion.

Primero vamos a instalar Vagrant (Este se instala como un comando para nuestro PC) y VirtualBox que funciona como el backend de Vagrant.

<https://www.vagrantup.com/downloads.html>

<https://www.virtualbox.org/wiki/Downloads>

Luego en la consola podemos corroborar que si esten instalados corriendo el comando vagrant –version para el caso de vagrant y para VirtualBox podemos ver si tenemos el acesso directo en el escritorio.

Creando nuestra primer maquina virtual

Vamos a crear nuestra maquina virtual en la que vamos a simular un ambiente de produccion similar al que esperamos obtener. Primero tenemos que crear una carpeta en la raiz del proyecto que se llame platzigram-deploy y es aquí donde tendremos todas las referencias a nuestros scripts y ademas vamos a crear un archivo de configuracion para *Vagrant*.

En vagrant podemos hacer algo similar que en npm podemos hacer uso del siguiente comando para que este nos cree un archivo de configuracion:

**Vagrant init**

Vagrant corre en Ruby, entonces al abrir nuestro editor de texto configuremos el lenguaje en Ruby para ver la sintaxis bien

Ahora podemos decirle a Vagrant que sistema operativo queremos manejar, podemos ver que sistema operativo es el que necesitamos instalar nosotros que en este caso sera un Linux ya que vamos a usar Digital Ocean que corre en sistemas Linux. En esta pagina encontramos todas las imágenes y dependencias que podemos instalar con Vagrant:

https://vagrantcloud.com/search

Vamos a usar una version LTS de ubuntu llamada trusty64, entonces lo que hacemos es poner su nombre en la configuracion de nuestro archivo de Vagrant.

config.vm.box = "ubuntu/trusty64"

Ahora vamos a cambiar la configuracion de la ram que le asignaremos a nuesra maquina virtual que en este caso sera de 2GB.

config.vm.provider "virtualbox" do |vb|

# # Display the VirtualBox GUI when booting the machine

# vb.gui = true

#

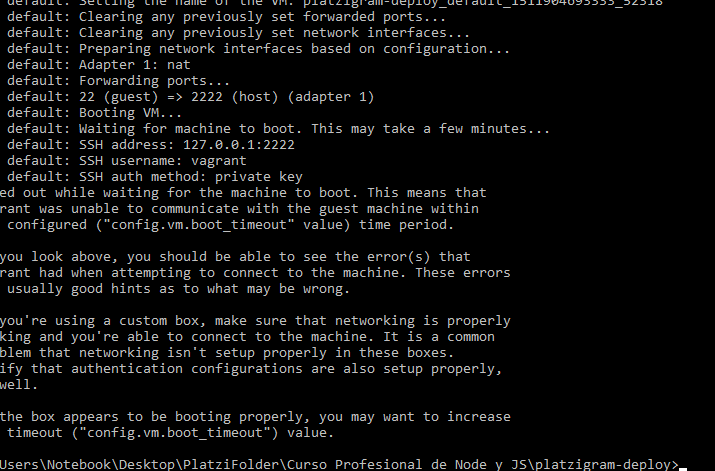
# # Customize the amount of memory on the VM:

vb.memory = "2048"

end

Ahora ya tenemos listo las dos configuraciones que vamos a necesitar para poder trabajar, siguiente a esto ya podemos ir a la terminal de comandos y podemos correr el comando **vagrant up**. (Este proceso puede tardar un tiempo ya que no tenemos descargada la imagen y luego la va a instalar)

**SSH** es un protocolo de seguridad en el que se encripta todos los comandos corridos de consola. Esto es mas que todo para que podamos administrar servidores remotamente. <https://es.wikipedia.org/wiki/Secure_Shell>



* Este nos entrega el puerto en el que va acorrer, ademas de las credenciales con las que vamos autenticarnos. La llave privada la veremos mas adelante.

Ahora si queremos ya entrar a nuestra maquina virtual sin configurar nada mas podemos hacer uso del siguiente comando:

**Vagrant ssh**

* Este lo que hace es que nos conecta con la maquina virtual mediante **SSH** y nos permite ya interactuar con nuestra maquina virtual.

Ahora tenemos que crear las llaves que vamos a utilizar para asegurar nuestra conexión con nuestro servidor, para esto podemos usar dos metodos de seguridad, uno el de llave privada y el otro de llave publica/llave privada.

Apartir de este momento voy a escribir solo lo esencial y no todo en detalle como pedazos de codigo. Esto lo voy a hacer ya que por el sistema operativo no puedo hacer uso del **ssh** entonces ire anotando cosas estricatemente como definiciones, funciones de ciertos programas y demas flujos en un entorno de produccion.

Luego de ver como se implementan las llaves publicas y privadas en ssh para que estas sean requeridas por Vagrant y asi poder conectarse al servidor de despliegue lo que sigue en la lista es hacer estas tareas de proveer las llaves privadas automaticamente mediante un software de provisionamiento de servidores y entornos de desarrollo. Ansible.

Introduccion a Ansible

Ansible nos permite automatizar nuestras tareas de provisionamiento del servidor en un entorno de produccion que en este caso nosotros lo estamos emulando para crear ese feeling de entorno de produccion en nuestro entorno de desarrollo.

Este proyecto de ansible es OpenSource y esta escrito en Python. Este a diferencia de sus competidores no usa un agente instalador en el servidor para poder comunicarse con el, Ansible usa el protocolo **ssh** si la necesidad de instalar dependencia en el servidor para permitir la comunicación entre ellos.

Este maneja unas reglas de juego, por asi decirlo, mediante los archivos playbooks donde nosotros iremos declarando las variables y los procesos que se van a realizar dentro de nuestras tareas automatizadas.

Ansible nos permite desplegar de forma facil y rapida, y ademas nos permite obtener esa instancia de despliegue en cualquier computador con solo “typear” un comando en la linea de comandos.

<https://www.ansible.com/>

En el enlace de arriba podemos encontrar toda la documentacion de ansible y las formas de como instalarlo que de por si esta orientado mucho a sistemas operativos tipo UNIX.

Creando nuestro primer rol de Ansible

Se usan las playbooks como tareas generales que dependen de los roles, en este caso los roles en Ansible son simplemente tareas especificas. Por ejemplo: Instalar node en la maquina, Instalar NGNX, etc.

Para crear el rol, creamos en la carpeta del proyecto de despliegue un directorio llamado roles y dento de este creamos una carpeta con el nombre del proyecto. Es aquí donde tendremos que crear 3 carpetas basicas que maneja Ansible.

1. Files: Estos tendran todos los archivos que nosotros queremos copiar en nuestro entorno de produccion. Es decir todo los proyectos que hemos creado a lo largo del curso. Por ejemplo: platizgram-api, platzigram-client, etc.
2. Meta: Este contendra informacion sobre el rol que se creo. Por ejemplo si quisieramos publicar el rol para que otro lo use lo que hacemos es colocar en este directorio toda la informacion acerca del rol.
3. Tasks: Aca es donde iran todas las tareas que hara el rol. En este caso seria todos los comando de npm o los comando de node para poder instalar todas las dependencias. Tambien la base de datos que estamos usando y demas.

Ahora lo que se hace es crear los archivos de tareas, Ansible maneja los archivos con extension YAML que es muy popular y usado en los entornos de desarrollo de Python y Ruby. Dentro de estos archivos iremos colocando las dependencias que vamos a instalar (Podemos colocar como dependencia un role externo). Podemos hacer uso de Ansible Galaxy que es basicamente como el NPM de Ansible donde se publican los roles mas comunes y que pueden ser consumidos por cualquiera.

<https://galaxy.ansible.com/>

En el enlace de arriba podemos encontrar los roles que han publicado y ademas la documentacion de como instalarlos.

Buscaremos en este caso los roles para NGNX, Nodejs, RethinkDB y las demas dependencias que hagan parte de nuestro proyecto.

(Estas dependencias iran dentro de la carpe de meta)

Ahora dentro de nuestro folder de Task vamos a crear las tareas genereales. En algunas dependencias de npm tenemos que algunas necesitan los frameworks de C y C++ para poder compilar, tenemos que hacer uso de apt para poder correr estas, pero no vamos a definir el role de apt para correr estas, lo que haremos es usar uno que este creado en la pagina de Ansible Galaxy.

Otra cosa importante es que node necesita dependencias para correr en entornos tipo UNIX, para esto node mismo recomienda usar los paquetes de nodesource. En este caso nodesource nos provee de esto y ademas de un rol de Ansible para poder instalarlo de forma automatica.

Tenemos que decirle a Ansible que nos garantice que git este instalado y ademas del build-essentials que es el que contiene todo lo necesario para compilar todos los archivos de C y C++.

Creando nuestro primer playbook para despliegue

Aca lo que hacemos en estos archivos de configuracion es colocar cuales son los host a los que nosotros vamos a hacer deploy mediante Ansible.

Luego mediante un archivo inventory.ini lo que hacemos es declarar el tipo de usuario de ansible-ssh y luego el host o direccion de este servidor que tendriamos en la nube.

Luego ya podemos correr Ansible y ver que se instalen todas las dependencias correctamente. Luego despues de esto vamos a hacer el forward de los puertos para que cuando nosotros en nuestra maquina estemos corriendo por ejemplo la terminal de rethinkdb no lo corrra en el mismo puerto en el servidor. Entonces para eso hacemos el mapeo y asi evitamos errores al hacer deploy.

Crea tareas para Platzigram dentro de Ansible

Ahora vamos a instalar los demas proyectos que teniamos. Vamos a hacer uso del directorio de Files. Esto podemos hacerlo de varias maneras. Podemos usar repositorios privados con NPM con versiones actualizadas y luego desempaquetado y listo para usar.

Vamos a hacer uso del comando Pack de NPM que nos permite empaquetar un modulo con solo la informacion necesaria para ese modulo.

Para desempaquetar un modulo de NPM se usa el comando **tar xvfz**, Este lo que hace es desencriptar los archivos en los dos formatos que tiene NPM que son tar y gzip. Luego ya este nos expande los archivos y nos lo deja en una carpeta llamada package.

Luego ya podemos hacer uso del modulo de copy de Ansible pero primero tenemos que asegurarnos que antes de copiar los archivos deba estar configurada la base de datos. Luego despues de esto ya podemos correr las tareas de copiar los proyectos.

A este le tenemos que indicar el comando que tiene que correr, luego el directorio de donde tiene que copiar los archivos y la carpeta destino en nuestro servidor.

Crear tareas para el resto de los proyectos

Aca lo que vamos a hacer es copiar todos nuestros proyectos dentro de nuestro servidor usando ansible. Hacemos basicamente lo mismo que en el anterior paso solo que cambian los proyectos que vamos a copiar.

Podemos indicarle a Ansible que en lugar de hacer manual el copiado de cada proyecto pueda hacer una tarea y que esa tarea la ejecute por cada una de los proyectos que tengamos. Esto en Ansible se le conocen como Loops. Entonces le tenemos que pasar por parametro un item en el atributo de src y luego le indicamos con un nuevo atributo with\_items el parametro de proyecto que se esta ejecutando.

Esta lista de proyecto la tenemos que crear en el archivo main/playbook de nuestro Ansible y ahí crear una variable con el nombre de la lista y el contenido que en este caso seria los nombres de los proyectos.

Creacion de servicios de upstart

Aca vamos a hacer uso de herramientas que nos permitan levantar nuestro proyecto en caso de que se caiga. Por que nosotros estamos usando npm start y arranca, pero en caso de que este se volque, Quien nos va a levantar de nuevo nuestro proyecto?.

Dentro del ecosistema de node existen varias herramientas que nos permiten hacer este proceso de levantamiento de servidor como lo son supervisor y demas que nos permiten hacer processmonitoring. No deberiamos solucionar estos problemas de upstarting con node ya que en los sistemas operativos tipo UNIX nos proveen mecanimos para solucionar esto. El mas conocido se llama upstart.

Upstart me permite crear servicios dentro de mi sistema operativo y manejarlos mediante comandos que me permitan arrancar un proceso o terminar un proceso. Lo que vamos a necesitar a la primera seran nuestros microservicios de nuestra api, nuestro modulo de Realtime y nuestro platzigram frontend. Para cada uno de estos procesos tenemos que crear un archivo de configuracion.

Aca es donde vamos tambien a actualizar nuestras variables de entorno, todos los archivos de configuracion. Dentro de un archivo .config vamos a definir los codigos que usara **upstart** para correr el servicio o para detenerlo. Tambien podemos indicarle que nos levante el servicio de nuevo con la palabra respawn. Luego le seteamos el usuario que hara esto mas el directorio destino de esto.

Siguiente vamos a setear nuestras variables de entorno que en este proyecto eran las de produccion para la clase de base de datos y las llaves para los secrets de AWS y de JWT.

Estas variables de entorno tienen que estar actualizadas cada vez que nosotroso vayamos a hacer commit a produccion y luego podemos ya subir todos los cambios al servidor. Siempre tenemos que asegurarnos de que estas este actualizadas con el codigo que vayamos actualizando en el github o en el repositorio que estemos usando.

Luego de esto vamos a hacer que Ansible corra nuestros servicios creando una nueva variable y asignandole una lista con los nombres de los archivos de configuracion que creamos anteriormente. Luego en el main/playbook hacemos lo mismo que con la tarea anterior para hacer uso de parametros dentro de la tarea.

Tenemos que hacer esto para todos los proyectos, en este caso el de Realtime y el del frontend.

Ahora tenemos que ver que herramienta nos permite hacer un agrupamiento de todos estos servicios y que se vean todos juntos en un solo puerto HTTP. Para esto vamos a usar NGNX.

Introduccion a NGiNX

Tenemos 3 servicios corriendo en puertos distintos, Platzigram-api, Platzigram-Realtime y Platzigram-Frontend. Todos estos manejan un puerto HTTP distinto por lo que hace la conexión a estos servicios mas complicada, NGINX crea una capa que nos permite acceder a todos estos servicios atravez de un solo puerto (El puerto 80 es el que maneja por defecto para HTTP) y nadie vea como tenemos la arquitectura de nuestro proyecto. NGINX maneja proxys, lo que hace es generar proxys para cada uno de estos puertos y luego servir toda nuestra aplicación en un solo puerto.

En los siguiente videos vamos a ver como configurar cada uno de estos archivos que tendremos que crear para orquestar toda nuestra aplicación con NGINX.

<https://nginx.org/en/>

En el enlace de arriba tendremos toda la informacion de NGINX y su documentacion.

Creacion de rutas para nuestros microservicios

Es momento de crear nuestrra capa Front que se encargara de crear un servidor para nuestros proyectos, una capa para nuestro proyecto de API otro para el Realtime y otro para nuestro Frontend.

Primero creamos una carpeta en nuestro apartado Files de Ansible y la llamamos NGNIX, luego dentro de que este creada, creamos un archivo para la configuracion de nuestro servidor. NGNIX maneja un lenguaje parecido al JSON pero no es JSON. Entonces en este archivo lo que vamos a hacer es configurar el servidor que queremos levantar.

Ahora tenemos que decirle el puerto en el que va a escuchar, el nombre del servidor. Siguiente a esto le vamos definir una ruta que es la por defecto que seria la ‘/’ de nuestro servidor virtual. En este lo que vamos a hacer es usar un proxy\_pass que nos redirecciona la peticion que hagan a este puerto o servidor montado por NGNIX a nuestro servidor “real” montado en Node.js que es el que nos va hacer toda la logica de nuestra aplicación.

Tenemos que setear los headers de los requests que nos llegan a NGINX para poder pasarselos a Node.js, el primero sera el host al que le haran esta peticion, el segundo sera la ip real del sujeto que nos hizo la peticion (Esto por si necesitamos saber la ip en algun momento de nuestra aplicación), el tercero es a donde va dirijido esta peticion, el cuarto es el protocolo por el cual nos hacen la peticion. Y por ultimo la version de http (esta es opcional). Todos estos header seran mandados a nuestro servidor real de Node donde sera manipulada la informacion.

Este seria el archivo de configuracion para nuestro primer proyecto de Frontend. Tenemos que repetir el proceso para los demas proyectos.

Para nuestro proyecto de WebSockets solomente cambiamos el nombre para asi poder diferenciar los DNS cuando vayamos a hacer peticiones a este servicio y ademas Socket.io necesita dos headers adicionales para funcionar, El header de connection y el header de upgrade. Este header de upgrade lo que hace es recibir los cambios de estado en el protocolo de HTTP y el otro es la actualizacion de la conexión.

Para el microservicio mas complejo, lo que hacemos es definir locations haciendo match primero a cada microservicio, estos microservicios solo necesitan los header default, pero necesitamos obtener del request los parametros de busqueda para nuestra ruta como por ejemplo /pictures/tags/users. Para hacer esto NGNIX nos entrega una variable de request llamada $request\_uri que contiene toda la ruta de la peticion HTTP. Para acceder a estos campos vamos a capturarlos con una expresion regular y el uso de un IF que nos permite hacer NGINX. Luego de esto lo que vamos a hacer es dentro del if hacer el proxy\_pass y a la ruta donde esta corriendo nuestro microservicio de pictures le pasamos por parametro el resultado de la expresion regular. (Este resultado queda guardado en una variable seteada por NGNIX el cual es llamada $1).

Este proceso lo repetimos para los demas microservicios.

Ahora tenemos que hacer que Ansible nos cargue los archivos en nuestro servidor, para esto tenemos que ir a nuestro archivo main y crear una nueva tarea al cual le hacemos un copy con ruta nginx/{{ item}}, luego le indicamos la ruta de destino en nuestro servidor UNIX y seteamos los modos que van a ser solo de lectura para todos los usarios.

Luego vamos a hacer una iteracion de los elementos que tenemos pero si queremos podemos hacerlo directamente si crear una variable antes.

Por ultimo creamos una tarea para reiniciar NGINX en el que va a ejecutar un servicio para reiniciar nuestro servicio de NGINX.

Ya tenemos todo nuestro proyecto montado a NGINX, ahora tenemos que llevar nuestro proyecto a produccion ya no queremos mas tener nuestro proyecto en un ambiente de desarrollo simulado.

Llevando nuestro proyecto a Digital Ocean

Digital Ocean es un servicio de computacion en la nube que nos permite crear maquinas o instancias para hacer despliegue de proyectos a produccion pero en la nube. Entonces ya no vamos a hacer despliegue a nuestra maquina virtual en Vagrant si no que vamos a hacerlo a Digital Ocean que nos entregara una IP o DNS para poder conectar nuestro despiegue alli. Una caracteristica super buena de Digital Ocean es que tiene Discos SSD.

Digital Ocean le llama droplets a los servidores o maquinas en la nube para poder hacer despliegue. Cuando vamos a crear una este nos dice que sistema operativo queremos instalar en nuestro servidor. Tambien podemos crear One Click Apps por ejemplo si creamos una aplicación con el stack MEAN este lo que hace es configurarnos nuestra maquina virtual de una para este stack, pero en nuestro caso del curso nosotros creamos toda la configuracion desde 0 con Ansible asi que vamos a usar la imagen de ubuntu y la vamos a ir configurando.

Luego podemos escoger el plan para nuestra maquina. Ademas de escoger la region del datacenter, este lo vamos a escoger según nosotros decidamos la que se ajuste mejor a nuestra posicion geografica.

Tambien nos pedira la llave SSH que se habia creado para hacer deploy. Esta llave es la publica no la privada. La que usabamos para hacer el deploy a nuestro servidor local en una maquina virtual.

Luego ya podemos crear nuestro droplet y este empezara el proceso de creacion de la maquina virtual y luego nos entregara una IP que sera a la que vamos a acceder a nuestro servidor. Pero tenemos que cambiar en nuestro archivo de inventory.ini y ahí copiamos la IP que nos entrego Digital Ocean. (Tenemos que crear el rol con el nombre del droplet que creamos en Digital Ocean y le decimos a Ansible que somo usuario root y le pasamos la IP de nuestro Droplet). Luego de hacer el deploy con Ansible ya podemos hacer uso de nuestra IP en el navegador y veremos que nos entregara el archivo de bienvenidad de NGINX. Lo que faltaria seria dar un DNS a nuestra IP y crear los subdominios para los demas microservicios de nuestro proyecto.

<https://www.digitalocean.com/>

En el enlace de arriba tenemos la pagina de Digital Ocean.

Dominio y proyecto en produccion

Ahora que ya tenemos desplegado nuestro proyecto en nuestro servidor en Digital Ocean, vamos a hacer uso del protocolo de DNS (Domain Name Server) al cual le vamos a asignar un nombre a nuestra ip del servidor. En este caso el profesor Julian usa Namecheap el cual contiene dominios bastante baratos y la administracion de estos es realmente sencilla.

<https://www.namecheap.com/>

Luego de tener comprado nuestro dominio podemos ir a Advanced DNS y luego vemos los Host Records que tenemos. Ahora vamos a crear nuestros Records.

Existen dos dominios fundamentales. El primero es el que convierte una direccion IP en un nombre de dominio y el segundo son los Records CNAME que basicamente son los subdominios de nuestra IP.

Para crear el primero lo que hacemos es crear un nuevo record y darle que va a ser de tipo Record A, luego este nos pide el nombre del Host que le vamos a asignar (Siempre en Namecheap le vamos a poner “**@”**) y luego la IP que nos entrega Digital Ocean. Por ultimo le damos a salvar cambios y listo.

Para crear el segundo podemos actualizar el que tenemos por defecto por Namecheap. Este CNAME lo que hace es redireccionarnos cualquier peticion hecha en un www o en un host que queramos poner en ese campo y lo envia al nombre de dominio que nosotros le indiquemos, En este caso seria el dominio que acabamos de comprar. Por ultimo nos dice un campo TTL (Time To Live), basicamente es el tiempo en el que va a vivir este dominio de CNAME (Lo recomendado es dejarlo en automatico).

Luego como tenemos dos rutas mas que son las de api y la de ws, lo que hacemos es usar estos Records de CNAME y indicarle en el host el subdominio (ws) y luego en la redireccion la apuntamos al dominio principal. Luego le damos a guardar.

Ya por ultimo tenemos que esperar a que ocurrar el efecto de propagacion por todos los servidores de DNS. Este proceso puede tardar 48hrs, pero esto no es del tanto cierto, puede tomar de 30mins a 60mins. Esto debido a que tiene que ocurrir el efecto de cambio en todos los servidores de DNS del mundo de internet, entre mas cercano mas rapido ocurre el cambio. Luego de que este totalmente propagado ya podemos ver nuestro sitio del proyecto que subimos.

Tenemos herramientas en ciertos sistemas operativos atravez de la linea de comando, en MAC OS esta **dig** que nos permite hacer consultas a servidores de DNS para verificar si estos estan listos para ser usado. Si este no esta disponible nos va a entregar un registro antiguo de lo contrario nos dara registro con nuestro dominio actualizado e indicando que el dominio apunta a la IP que nos entrega Digital Ocean.