



VAGRANT-ANSIBLE MQTT + STACK TIG

ENTREGA - 3

12 Abril 2022

Escuela Superior de Informática

Mariola Zarco de Gracia

Índice de contenidos

Índice de contenidos	1
1. Enunciado	2
2. Tecnologías utilizadas.....	2
2.1. VirtualBox.....	2
2.2. Vagrant	2
2.3. Ansible	3
3. Distribución de Arquitectura.....	4
4. Configuraciones y documentación	4
4.1. Vagrant File	4
4.2. Configuración de los Hosts - Inventario	8
4.3. Configuración genérica – Ping, EPEL e Instalación de servicios	8
4.4. Configuración del Nodo 1 - Broker de Mosquitto	11
4.5. Configuración del Nodo 2 - Publisher de Mosquitto.....	11
4.6. Configuración del Nodo 3 - TIG	12
4.7. Configuración del Nodo 4 - Sensores	15
4.8. Desinstalación	16

1. Enunciado

Automatice el despliegue y configuración de un sistema que disponga de al menos tres actores / servicios que interactúen entre sí, mediante el uso de Vagrant y Ansible. El ejercicio queda abierto a cualquier servicio, por lo tanto es parte del / la estudiante elegir qué servicios serán configurados y desplegados en las máquinas descritas en Vagrant.

Se propone como ejemplo el stack TIG (Telegraf + InfluxDB + Grafana) visto en la asignatura de sistemas ciberfísicos, donde cada uno de los servicios puede ser desplegado y configurado en diferentes máquinas.

Incluso el bróker MQTT puede ser también virtualizado mediante Vagrant y Ansible.



Se deberá entregar una pequeña memoria con las decisiones tomadas y razonadas (recursos hardware asignados a cada nodo, configuraciones de red, ...), una imagen de la arquitectura TI implementada y el funcionamiento del mismo. Se deberá incluir un fichero adicional con los pasos a realizar para desplegar toda la arquitectura.

2. Tecnologías utilizadas

2.1. VirtualBox

Es un software de virtualización desarrollado por Oracle Corporation, soporta múltiples sistemas operativos como GNU/Linux, Mac OS X, Windows, además de ser una herramienta que tiene una versión de código abierto con licencia GPL versión 2.

VirtualBox permite virtualizar los sistemas operativos FreeBSD, GNU/Linux, OpenBSD, Windows, Solaris y muchos otros. En lo que respecta a los discos duros, utiliza emulación hardware en el que los discos de los sistemas invitados se guardan en el anfitrión como archivos individuales llamados discos duros virtuales.

2.2. Vagrant

Vagrant es una herramienta para la creación y configuración de entornos virtualizados en un solo flujo de trabajo. Es bastante simple, creada por HashiCorp puede ejecutarse tanto en Mac OS, Linux como en Windows. Además, es una herramienta de código abierto, con licencia MIT.

Esta desarrollada en Ruby y su configuración se basa en ficheros de configuración llamados Vagrantfiles. En estos ficheros se puede definir todo un entorno virtual que necesites desplegar, desde el sistema operativo a las interfaces de red.

También nos permite gestionar el aprovisionamiento de las instancias, compatible con orquestadores como Ansible, Docker y scripts de bash. La integración con estas tecnologías permite un control total sobre la configuración de las instancias.

Otra funcionalidad muy interesante es a lo que Vagrant denomina Boxes¹, son imágenes preinstaladas de diferentes sistemas operativos versionados. Hasicorp proporciona estas Boxes a través de un repositorio, donde se puede encontrar todos los sistemas en sus versiones oficiales. Y también se pueden encontrar Boxes subidas por usuarios con modificaciones y preinstalaciones de aplicaciones listas para ser usadas.

2.3. Ansible

Es una plataforma de software libre para aprovisionar y administrar infraestructura y aplicaciones. Está desarrollado principalmente por RedHat, ha sido categorizado como herramienta de orquestación. Gestiona los nodos a través de SSH y no necesita de ningún software adicional (excepto Python).

Con Ansible podemos actuar contra plataformas en la nube, como host virtualizados, hipervisores e incluso dispositivos de red o servidores físicos. Se distribuye más comúnmente en arquitecturas GPU/Linux a través de los paquetes EPEL o PIP.

Arquitectura de ansible:

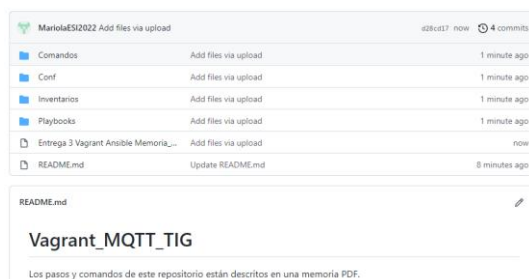
Ansible tiene dos tipos de servidores:

Automatización de la creación de escenarios para protocolos de red.

- Controlador: Servidor desde el que se lanza la orquestación y tiene el código fuente a ejecutar. Tiene Ansible instalado. En nuestro caso será la máquina anfitriona.
- Nodo: Al que se conecta el controlador mediante SSH y es aprovisionado. No necesita de ningún software aparte de conectividad SSH como ya hemos comentado.

Los ficheros descritos en el documento para las consiguaciones se encuentran en:

https://github.com/MariolaESI2022/Vagrant_MQTT_TIG/upload/main



¹ Vagrant boxes: <https://app.vagrantup.com/boxes/search>

3. Distribución de Arquitectura

Hemos decidido proceder con los datos recogidos de nuestro anterior ejercicio de la asignatura de IoT, que contenía información de sensores de temperatura, sonido y nivel de líquido, que reemplazarían los de presencia en este dibujo, y quedaría la distribución por nodos como sigue:

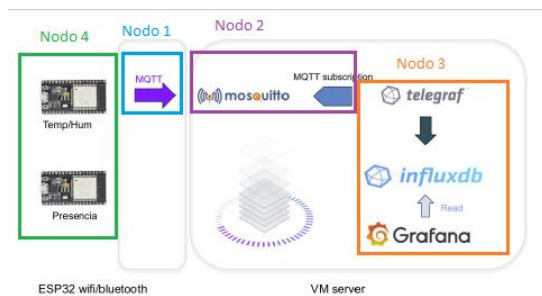


Figura 3.1 – Arquitectura y distribución de Nodos.

4. Configuraciones y documentación

4.1. Vagrant File

Para la configuración del Vagrant file se han tenido en cuenta las siguientes cuestiones:

COMANDO: `$ vagrant init`

- En primer lugar, hemos revisado el listado de Boxes de Vagrant, filtrando por los conceptos de mosquitto, mqtt, grafana y ansible y aunque hemos encontrado algunas boxes ya desarrolladas para algunas de las aplicaciones, no se correspondían 100% con nuestro ejercicio, por lo que hemos decidido dejar la box genérica de Ubuntu.²
- Hemos decidido trabajar con la distribución de Ubuntu. Se podría haber elegido la distribución CentOS por ofrecer mayor estabilidad y haber aplicado el comando yum para la descarga e instalación de paquetes, pero hemos preferido mantener Ubuntu por disponer de una comunidad de desarrollo e informaciones más amplia en red para esta etapa inicial ya que los datos en la aplicación tampoco son abundantes.

```
Vagrant.configure("2") do |config|  
  config.vm.box = "ubuntu/xenial64"
```

² <https://app.vagrantup.com/vadoc2020/boxes/mosquitto>

<https://app.vagrantup.com/codeyourinfra/boxes/monitor>

https://app.vagrantup.com/superbogiuseppe/boxes/grafana_server

- Ya que son distintas las aplicaciones que van a necesitar de acceso a la red y la red de las máquinas debe ser la misma que las del PC anfitrión, hemos optado por configurar una red pública, creando un puente entre redes para que cada máquina aparezca como otro dispositivo físico y tener una mejor identificación. Esto lo definimos dentro de cada nodo.

```
N1.vm.network "public_network", bridge: "enp_03"
```

- Se debe redirigir el puerto 80 de las máquinas invitadas o SO invitado a un puerto de la máquina que las alberga (anfitrión).

COMANDO: `$ sudo python3 -m http.server 80`

- Hemos configurado los nodos como sigue, teniendo en cuenta que la mayor cantidad de flujo vendrá desde el Broker y el Publisher de Mosquitto, los hemos puesto en nodos independientes 1 y 2, y el resto de aplicaciones TIG, las hemos unificado en un único nodo 3, y por último los sensores en un nodo 4, balanceando así las cargas entre los nodos y manteniendo el acceso a la información lo más cerca posible entre ellos.

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/xenial64"
  config.vm.define "nodo1" do |n1|
    N1.vm.network "public_network", bridge: "enp_03"
    N1.vm.hostname = "Mosquitto_Broker"
    N1.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"
    N1.vm.provider "virtualbox" do |vb|
      vb.cpus = "2"
      vb.Memory = "1024"
    end
  end

  config.vm.define "nodo2" do |n2|
    N2.vm.network "public_network", bridge: "enp_03"
    N2.vm.hostname = "Mosquitto_Publisher"
    N2.vm.network "forwarded_port", guest: 80, host: 8081, host_ip: "127.0.0.1"
    N2.vm.provider "virtualbox" do |vb|
      vb.cpus = "2"
      vb.Memory = "512"
    end
  end
```

```
  config.vm.define "nodo3" do |n3|
    N3.vm.network "public_network", bridge: "enp_03"
    N3.vm.hostname = "TIG"
    N3.vm.network "forwarded_port", guest: 80, host: 8082, host_ip: "127.0.0.1"
    N3.vm.provider "virtualbox" do |vb|
      vb.cpus = "2"
      vb.Memory = "512"
    end
  end

  config.vm.define "nodo4" do |n4|
    N4.vm.network "public_network", bridge: "enp_04"
    N4.vm.hostname = "Sensores"
    N4.vm.network "forwarded_port", guest: 80, host: 8083, host_ip: "127.0.0.1"
    N4.vm.provider "virtualbox" do |vb|
      vb.cpus = "2"
      vb.Memory = "512"
    end
  end
```

Nota: Hemos puesto el hostname con el fin de ilustrar y referenciar visualmente la distribución de los nodos, pero se mantendrán sus hostnames como nodo1, nodo2, nodo3 y nodo4.

- Por último hemos definido las rutas para los archivos que deben sincronizarse tanto en el anfitrión como en la máquina invitada como sigue:

```
config.vm.synced_folder "../data/nodo1", "service/ansible/nodo1"
config.vm.synced_folder "../data/nodo2", "service/ansible/nodo2"
config.vm.synced_folder "../data/nodo3", "service/ansible/nodo3"
config.vm.synced_folder "../data/nodo4", "service/ansible/nodo4"
```

- En este caso únicamente tenemos un hypervisor con lo que lo necesitaremos configuración adicional en este sentido.
- Por último, aquí describimos los comandos³ que se van a utilizar tanto para la configuración, como para el inicio o parada de la máquina virtual, y para la configuración de SSH para tener acceso a la terminal, **especificando el nombre de la máquina virtual**.

```
#Comando para configuración de la máquina virtual
$ vagrant init ubuntu/xenial64

#Comandos para iniciar o detener la máquina virtual
$ vagrant up xenial64
$ vagrant halt

#Comandos para ver el listado y creación
$ VBoxManage list vms | grep xenial64
"xenial64" {a507ba0c-...24bb}

#Configuración de SSH para tener acceso a la terminal
$ vagrant ssh
$ vagrant ssh default
$ vagrant ssh node1

#Comandos para ver el status
$ vagrant status
$ vagrant global-status
```

- Para el tamaño, hemos visitado las páginas con guías sobre el tamaño, como InfluxDB hardware sizing en su versión 1.9.⁴ donde se nos aconseja que si tenemos un único nodo, será válido pero no tendremos redundancia de datos, por lo cual si aplicamos el factor de réplica n-1, no podríamos permitirnos perder ningún nodo. Otras consideraciones⁵ a tener en cuenta son:
 - Número de escrituras y consultas por segundo
 - Cardinalidad de la serie
 - Complejidad de las consultas
 - Almacenaje de tipo SSD (sino al menos 2000 IOPs para recuperación rápida)

³ <https://www.vagrantup.com/docs/cli/ssh>

⁴ https://docs.influxdata.com/enterprise_influxdb/v1.9/guides/hardware_sizing/

- Bytes y comprensión - Según sean las medidas, *tags keys*, *field keys*, *tag values* y *consideración de que los field values y timestamps* se guardan en cada punto. Los valores no string, requieren 3 bytes y el resto depende de la compresión.
- Separación de los directorios wal y data en diferentes dispositivos de almacenamiento.

En nuestro proyecto teníamos la recepción de 3 sensores diferentes (agua, temperatura, sonido) en diferentes habitaciones de una casa con la siguiente estructura que define el tamaño de los tags utilizados:

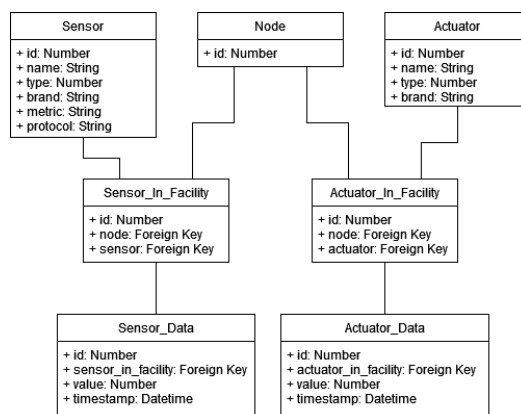


Figura 4.1 – Campos y etiquetas recogidos por cada sensor de la ESP32.

La estimación fue de 82 mensajes a la hora, con lo que tenemos un ratio inferior a 1 mensaje por segundo, con un total de 612 KB estimados al día, con lo que sincronizando bien los tiempos de escritura de los distintos dispositivos y teniendo en cuenta la simplicidad de las escrituras propuestas, tenemos un total de 252,68 KB al año.

- ➔ Por lo tanto, aun teniendo en cuenta las consideraciones anteriores, estimamos que con dos CPU para evitar la redundancia ya que las mediciones se realizan a la vez y es difícil sincronizarlas y con una capacidad de 512Mb⁵, tenemos más que suficiente para el nodo 3 TIG.

El nodo 1 y 2, en un principio tampoco requerirían de mayor capacidad de los 512Mb y con un VirtualBox de 60Gb deberíamos tener suficiente, según los cálculos arriba expuestos ya que no se prevé el aumento de número de sensores o medidas, las consultas a realizar no serían superiores a 2 por hora por cada tipo de dato y siempre podríamos redimensionar. Hemos preferido no limitarla a la baja por si necesitamos realizar la instalación de otras aplicaciones o instalaciones que puedan ocupar espacio en nuestras máquinas.

⁵ https://docs.influxdata.com/influxdb/v1.8/guides/hardware_sizing/

4.2. Configuración de los Hosts - Inventario

Configuramos los hosts a través del uso de ssh y los private keyfiles que a través del comando `ssh-keygen -lf /path/to/key` hemos encontrado ⁶en:

```
ansible_ssh_private_key_file=/home/mariola/vagrant_xenia164/.vagrant/machines/nodo1/virtualbox/private_key
```

El contenido del `Inventario_Hosts` es cada nodo con cada private key más su IP correspondiente y nos permite el acceso a través de los comandos ssh, o bien a los puertos de los nodos (forwarded del dispositivo NAT) o a través de la IP.

```
$ ssh -p 2200 \
> -i /home/mariola/vagrant_xenia164/.vagrant/machines/nodo1/virtualbox/private_key \
> vagrant@localhost hostname
xenia164

$ ssh -i /home/mariola/vagrant_xenia164/.vagrant/machines/nodo1/virtualbox/private_key \
> vagrant@192.168.18.50 hostname
xenia164
```

Comentado [MZDG1]:

Comentado [MZDG2R1]:

Comentado [MZDG3]:

4.3. Configuración genérica – Ping, EPEL e Instalación de servicios

Algunas notas previas a considerar antes de realizar el siguiente paso.

Nota 1

Las aplicaciones TIG las tenemos ya previamente instaladas a través del comando `wget` con la última versión de cada una de ellas en nuestro folder `/etc/aplicación`. Si este no fuera el caso, al no estar utilizando centOS tendríamos que incluir una opción `shell`: `wget [ruta destino] [link descarga]`.

Nota 2

Para ejecutar los playbooks utilizamos el comando:

COMANDO: `$ ansible-playbook-i hosts (nameofplaybook.yml)`

Nota 3

Para verificar que todos los nodos están activos ejecuto el siguiente comando

COMANDO: `$ ansible -i hosts all -m ping`

Ahora continuamos con la instalación de EPEL cada uno de los nodos e instalamos las correspondientes aplicaciones en caso de que no existan, con el `become:true` me convierto en superusuario y después procedo reiniciando cada uno de los servicios siendo el contenido del archivo `playbook_all.yml` el siguiente (página siguiente):

⁶ <https://martincarstenbach.wordpress.com/2019/12/12/tipsntricks-finding-the-injected-private-key-pair-used-in-vagrant-boxes/>

```

C: > Users > Usuario > Desktop > Entrega_3_Vagrant > ! playbook_all.yml
1  ---
2  - hosts: all
3    tasks:
4      - name: Ping to all nodes
5        ping:
6      - name: Install EPEL
7        become: true
8        name: epel-release
9        state: present
10
11
12  - hosts: nodo1
13    become: true
14    tasks:
15      - name: Install mosquitto
16        become: true
17        name: mosquitto
18        state: present
19      - name: Copy config
20        copy:
21          src: broker.conf
22          dest: /etc/mosquitto/mosquitto.conf
23          notify: Restart Broker
24    handlers:
25      - name: Restart Broker
26        service:
27          name: mosquitto
28          state: started
29
30  - hosts: nodo2
31    become: true
32    tasks:
33      - name: Install mosquitto
34        become: true
35        name: mosquitto
36        state: present
37      - name: Copy config
38        copy:
39          src: pub.conf
40          dest: /etc/mosquitto/mosquitto.conf
41          notify: Restart Publisher
42    handlers:
43      - name: Restart Publisher
44        service:
45          name: mosquitto
46          state: started
47

```

```
C: > Users > Usuario > Desktop > Entrega_3_Vagrant > ! playbook_all.yml

48 - hosts: nodo3
49   become: true
50   tasks:
51     - name: Install Telegraf
52       become: true
53       name: telegraf
54       state: present
55     - name: Copy config
56       copy:
57         src: telegraf.conf
58         dest: /etc/telegraf/telegraf.conf
59       notify: Restart Telegraf
60     - name: Install Influx DB
61       become: true
62       name: influxDB
63       state: present
64     - name: Copy config
65       copy:
66         src: influxDB.conf
67         dest: /etc/influxDB/influxDB.conf
68       notify: Restart InfluxDB
69     - name: Install Grafana
70       become: true
71       name: grafana
72       state: present
73     - name: Copy config
74       copy:
75         src: influxDB.conf
76         dest: /etc/grafana/grafana.conf
77       notify: Restart Grafana
78   handlers:
79     - name: Restart Telegraf
80       service:
81         name: telegraf
82         state: started
83     - name: Restart InfluxDB
84       service:
85         name: influxdb
86         state: started
87     - name: Restart Grafana
88       service:
89         name: grafana
90         state: started
```

4.4. Configuración del Nodo 1 - Broker de Mosquitto

En este caso hemos seguido la configuración del ejercicio anterior practicado en clase, siendo el resultado del contenido del archivo *m_bro.conf*:

```
C: > Users > Usuario > Desktop > Entrega_3_Vagrant > broker.conf
1 listener 2883 192.168.18.53 #IP de sensores
2 allow_anonymous true
```

Por otro lado creamos el playbook del broker *playbook_m_bro.yml* con las tareas correspondientes de comprobación e inicio asignadas, como sigue:

```
1 ---
2 hosts: broker_mqtt
3 tasks:
4   - name: Copy config
5     become: true
6   copy:
7     src: broker.conf
8     dest: /etc/mosquitto/mosquitto.conf
9   - name: Check Broker
10    service:
11      name: mosquitto
12      state: started #le digo que inicie mosquito
13
```

4.5. Configuración del Nodo 2 - Publisher de Mosquitto

En este caso hemos seguido con la configuración⁷ del archivo *m_pub.conf*:

```
C: > Users > Usuario > Desktop > Entrega_3_Vagrant > pub.conf
1 listener 2883 192.168.18.50 #IP del broker
2 allow_anonymous true
```

Por otro lado creamos el playbook del publisher *playbook_m_pub.yml* con las tareas de publicación asignadas, como sigue:

⁷ https://mosquitto.org/man/mosquitto_pub-1.html

https://apimirror.com/ansible~2.9/modules/mqtt_module

```

1  ---
2  - hosts: pub_mqtt
3  tasks:
4      - name: Install Paho-MQTT
5        become: true
6        yum:
7          name: python-paho-mqtt
8          state: present
9      - name: Publish a message on an MQTT topic - Temperature
10        mqtt:
11          topic: 'service/ansible/nodo4'
12          payload: 'Temperature at {{ ansible_date_time.iso8601 }}'
13          qos: 0 # calidad el servicio
14          retain: False # si queremos que se retenga el mensaje
15          server: 192.168.18.51. #IP del nodo 2, que es donde publica.
16          port: 2883 #(por defecto es el 1883)
17      - name: Publish a message on an MQTT topic - Sound Level
18        mqtt:
19          topic: 'service/ansible/nodo4'
20          payload: 'Sound_level at {{ ansible_date_time.iso8601 }}'
21          qos: 0
22          retain: False
23          server: 192.168.18.51.
24          port: 2883
25      - name: Publish a message on an MQTT topic - Water Level
26        mqtt:
27          topic: 'service/ansible/nodo4'
28          payload: 'Water_level at {{ ansible_date_time.iso8601 }}'
29          qos: 0
30          retain: False
31          server: 192.168.18.51.
32          port: 2883

```

4.6. Configuración del Nodo 3 - TIG

Configuración de los archivos de configuración de cada una de las aplicaciones *playbook_tel_conf*, *playbook_inf_conf* y *playbook_graf_conf*:

```

C: > Users > Usuario > Desktop > Entrega_3_Vagrant > ! playbook_tel_conf.yml
1  ---
2  hosts: nodo3 #suscriptor y publicador telegraf
3  tasks:
4      - name: Copy config
5        become: true
6        copy:
7          src: telegraf.conf
8          dest: /etc/telegraf/telegraf.conf
9      - name: Comprobar si suscriptor y publicador de telegraf activo
10        service:
11          name: telegraf
12          state: started #le digo que inicie telegraf
13

```

```
C: > Users > Usuario > Desktop > Entrega_3_Vagrant > ! playbook_inf_conf.yml
1 ---
2 hosts: nodo3 #suscriptor InfluxDB
3 tasks:
4 - name: Copy config
5   become: true
6   copy:
7     src: influxDB.conf
8     dest: /etc/influxdb/influxDB.conf
9 - name: Check Suscriptor
10  service:
11    name: influxDB
12    state: started #le digo que inicie InfluxDB
13
```

```
C: > Users > Usuario > Desktop > Entrega_3_Vagrant > ! playbook_graf_conf.yml
1 ---
2 hosts: nodo3 #suscriptor y publicador grafana
3 tasks:
4 - name: Copy config
5   become: true
6   copy:
7     src: grafana.conf
8     dest: /etc/grafana/grafana.conf
9 - name: Comprobar si suscriptor de grafana esta activo
10  service:
11    name: grafana
12    state: started #le digo que inicie telegraf
13
```

Nota: No he encontrado el módulo de Telegraf para ver las opciones de su configuración por tanto lo he configurado con la plantilla de mqtt para que al menos figurase el paso en la memoria, pero no es correcto.

Configuración del suscriptor de telegraf con el archivo *telegraf_sus.conf*:

```
C: > Users > Usuario > Desktop > Entrega_3_Vagrant > telegraf_sus.conf
1 listener 2883 192.168.18.51 #IP del mosquitto_publisher
2 allow_anonymous true
```

Configuración del publicador de telegraf *playbook_tel_pub.conf*:

```
C: > Users > Usuario > Desktop > Entrega_3_Vagrant > ! playbook_tel_pub.yml
1 ---
2 - hosts: nodo3 #pub_telegraf
3   tasks:
4     - name: Transfer the message on an MQTT topic - Temperature from node 2 into InfluxDB
5       telegraf:
6         topic: 'service/ansible/nodo2'
7         payload: 'Temperature at {{ ansible_date_time.iso8601 }}'
8         qos: 0 # calidad el servicio
9         retain: False # si queremos que se retenga el mensaje
10        server: 192.168.18.52. #IP del nodo 3, que es donde publica.
11        port: 2883 #(por defecto es el 1883)
```

```

13 - name: Transfer the message on an MQTT topic - Sound Level from node 2 into InfluxDB
14   telegraf:
15     topic: 'service/ansible/nodo2'
16     payload: 'Sound_level at {{ ansible_date_time.iso8601 }}'
17     qos: 0
18     retain: False
19     server: 192.168.18.52.
20     port: 2883
21 - name: Transfer the message on an MQTT topic - Water Level from node 2 into InfluxDB
22   telegraf:
23     topic: 'service/ansible/nodo2'
24     payload: 'Water_level at {{ ansible_date_time.iso8601 }}'
25     qos: 0
26     retain: False
27     server: 192.168.18.52.
28     port: 2883

```

Primero creo una base de datos, cuya lectura la vamos a realizar a nivel interno y en principio sin restricciones de user y password a los usuarios. En este archivo *playbook_inf_DB.yml* también le indicamos los data_points a recolectar.

```

C: > Users > Usuario > Desktop > Entrega_3_Vagrant > ! playbook_inf_DB.yml
1 ---
2 hosts: nodo3 #suscriptor InfluxDB
3 tasks:
4 - name: Create database
5   influxdb_database:
6     hostname: "192.168.18.53"
7     database_name: "DB_Vivienda_1"
8 - name: Write points into database
9   influxdb_write:
10     hostname: "192.168.18.53"
11     database_name: "DB_Vivienda_1"
12     data_points:
13       - measurement: Temperature
14         tags:
15           host: nodo4
16           sensor: 1
17         time: "{{ ansible_date_time.iso8601 }}"
18         fields:
19           value: 2000 #desconozco el rango
20       - measurement: Sound_Level
21         tags:
22           host: nodo4
23           sensor: 2
24         time: "{{ ansible_date_time.iso8601 }}"
25         fields:
26           value: 2000 #desconozco el rango
27       - measurement: Water_Level
28         tags:
29           host: nodo4
30           sensor: 3
31         time: "{{ ansible_date_time.iso8601 }}"
32         fields:
33           value: 2000 #desconozco el rango
34

```

Por último la configuración del contenido del archivo `graf_datasource.yml` para poder escuchar y representar los registros de influxDB sería el siguiente:

```
C: > Users > Usuario > Desktop > Entrega_3_Vagrant > ! playbook_graf_datasource.yml
1  ---
2  - name: Create influxdb datasource
3    grafana_datasource:
4      name: "datasource-influxdb"
5      grafana_url: "https://grafana.company.com"
6      grafana_user: "admin"
7      grafana_password: "xxxxxx"
8      org_id: "1"
9      ds_type: "influxdb"
10     ds_url: "https://influx.company.com:8086"
11     database: "telegraf"
12     time_interval: ">10s"
13     tls_ca_cert: "/etc/ssl/certs/ca.pem"
```

Si quisiéramos visualizar la información a través de un dashboard podríamos utilizar la siguiente configuración:

```
- hosts: localhost
  connection: local
  tasks:
    - name: Import Grafana dashboard foo
      grafana_dashboard:
        grafana_url: http://grafana.company.com
        grafana_api_key: "{{ grafana_api_key }}"
        state: present
        message: Updated by ansible
        overwrite: yes
        path: /path/to/dashboards/foo.json

    - name: Export dashboard
      grafana_dashboard:
        grafana_url: http://grafana.company.com
        grafana_user: "admin"
        grafana_password: "{{ grafana_password }}"
        org_id: 1
        state: export
        uid: "000000653"
        path: "/path/to/dashboards/000000653.json"
```

4.7. Configuración del Nodo 4 - Sensores

Aunque lo hemos puesto en último lugar, realmente deberíamos haber empezado por esta configuración para poder ir comprobando paso a paso que el resto de las configuraciones anteriores eran correctas conforme construíamos la arquitectura. La configuración del nodo 4 que contiene la publicación de información por parte de los sensores sería la siguiente `playbook_sens_pub.yml`:


```

C: > Users > Usuario > Desktop > Entrega_3_Vagrant > ! playbook_sen_pub.yml
1  ---
2  - hosts: nodo4 #sensores
3    tasks:
4      - name: Install Paho-MQTT
5        become: true
6        yum:
7          name: python-paho-mqtt
8          state: present
9      - name: Publish a message on an MQTT topic - Temperature
10       mqtt:
11         topic: 'service/ansible/nodo4'
12         payload: 'Temperature at {{ ansible_date_time.iso8601 }}'
13         qos: 0 # calidad el servicio
14         retain: False # si queremos que se retenga el mensaje
15         server: 192.168.18.50. #IP del nodo 1, que es donde publica.
16         port: 2883 #(por defecto es el 1883)
17      - name: Publish a message on an MQTT topic - Sound Level
18       mqtt:
19         topic: 'service/ansible/nodo4'
20         payload: 'Sound_level at {{ ansible_date_time.iso8601 }}'
21         qos: 0
22         retain: False
23         server: 192.168.18.50.
24         port: 2883
25      - name: Publish a message on an MQTT topic - Water Level
26       mqtt:
27         topic: 'service/ansible/nodo4'
28         payload: 'Water_level at {{ ansible_date_time.iso8601 }}'
29         qos: 0
30         retain: False
31         server: 192.168.18.50.
32         port: 2883

```

4.8. Desinstalación ⁸

Por último existen unos comandos útiles para la desinstalación del programa:

```

rm -rf /opt/vagrant
rm -f /usr/bin/vagrant

```

Y de los datos asociados:

En todas las plataformas, este directorio se encuentra en la raíz de su directorio personal, y se llama `vagrant.d`. Eliminamos el directorio `~/vagrant.d` para borrar los datos del usuario.

⁸ <https://www.vagrantup.com/docs/installation/uninstallation>