

ACTIVIDAD 2 – AWS IoT – Node-Red

INFORME DE ACTIVIDAD

- 1) Se deberá entregar un documento en formato PDF, en el que se explique detalladamente, los pasos que se han seguido para resolver las actividades propuestas. Dicho documento, debe incluir las capturas de pantalla necesarias en las que se pueda ver el trabajo del alumno. Además, se deberá entregar un VIDEO MP4 mostrando el funcionamiento del sistema, junto con el flujo de Node-RED con nodos “debug” para mostrar el funcionamiento.
- 2) Plazo de entrega: Viernes 18/10/2024 hasta las 24hs.

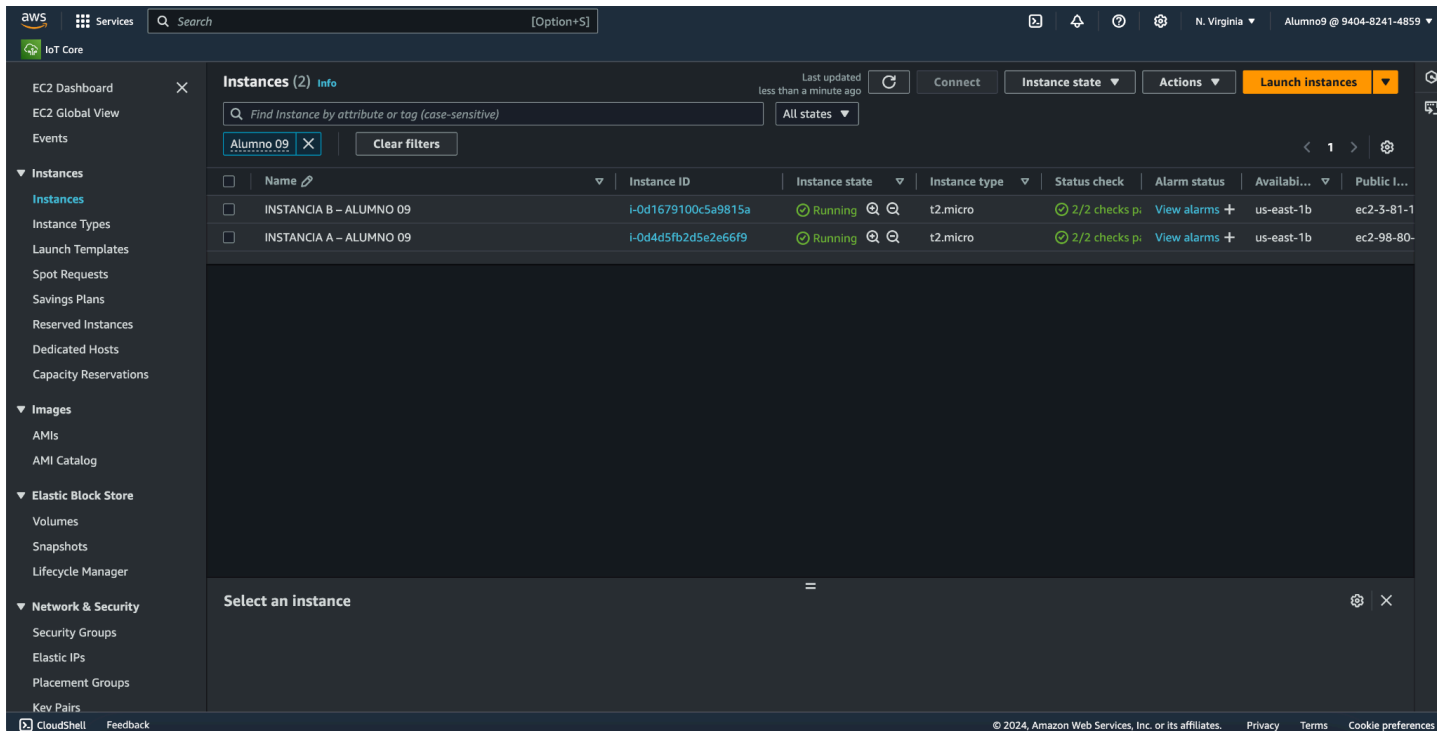
ACTIVIDAD PROPUESTA

- 1) Desplegar dos instancias t2.micro con Linux Debian 12 en AWS EC2: Instancia A e Instancia B.

Siguiendo lo aprendido de la actividad 1, levantamos las dos instancias denominadas como:

INSTANCIA A – ALUMNO 09

INSTANCIA B – ALUMNO 09



- 2) Instalar “Node-RED” en ambas instancias.

Nos conectamos vía SSH e instalamos node-red en ambas instancias:

```
ssh -i G9-Ej1.pem admin@ IP instancia A
ssh -i G9-Ej1.pem admin@ IP instancia B
```

en cada una de las instancias ejecutamos estos comandos:

```
sudo apt update
sudo apt install snapd
sudo snap install core
sudo snap install node-red
```

```
Cloud Computing — admin@ip-172-31-38-0: ~ — ssh -i G9-Ej1.pem admi...
...-i G9-Ej1.pem admin@98.80.229.121 ...9-Ej1.pem admin@3.81.162.154 +
ener.service → /lib/systemd/system/snapd.aa-prompt-listener.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.apparmor.servi
ce → /lib/systemd/system/snapd.apparmor.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.recovery-choos
er-trigger.service → /lib/systemd/system/snapd.recovery-chooser-trigger.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.seeded.service
→ /lib/systemd/system/snapd.seeded.service.
Created symlink /etc/systemd/system/cloud-final.service.wants/snapd.seeded.servi
ce → /lib/systemd/system/snapd.seeded.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.service → /lib
/systemd/system/snapd.service.
Created symlink /etc/systemd/system/sockets.target.wants/snapd.socket → /lib/sys
temd/system/snapd.socket.
Setting up gpg-wks-client (2.2.40-1.1) ...
Setting up gnupg (2.2.40-1.1) ...
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for dbus (1.14.10-1~deb12u1) ...
Processing triggers for libc-bin (2.36-9+deb12u7) ...
[admin@ip-172-31-38-0:~$ sudo snap install core
2024-10-16T18:41:06Z INFO Waiting for automatic snapd restart...
core 16-2.61.4-20240607 from Canonical✓ installed
admin@ip-172-31-38-0:~$ sudo snap install node-red
node-red 3.1.0 from Node-RED-Team (noderedteam✓) installed
admin@ip-172-31-38-0:~$
```

Luego instalamos net-tools

```
sudo apt install net-tools
```

EC2 > Security Groups > sg-0fe311eff35a6b688 - SegGrupo9

sg-0fe311eff35a6b688 - SegGrupo9

Actions

Details

Security group name SegGrupo9	Security group ID sg-0fe311eff35a6b688	Description launch-wizard-8 created 2024-09-25T18:10:58.234Z	VPC ID vpc-0e560130e5ddb730f
Owner 940482414859	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Tags

Inbound rules (1/2)

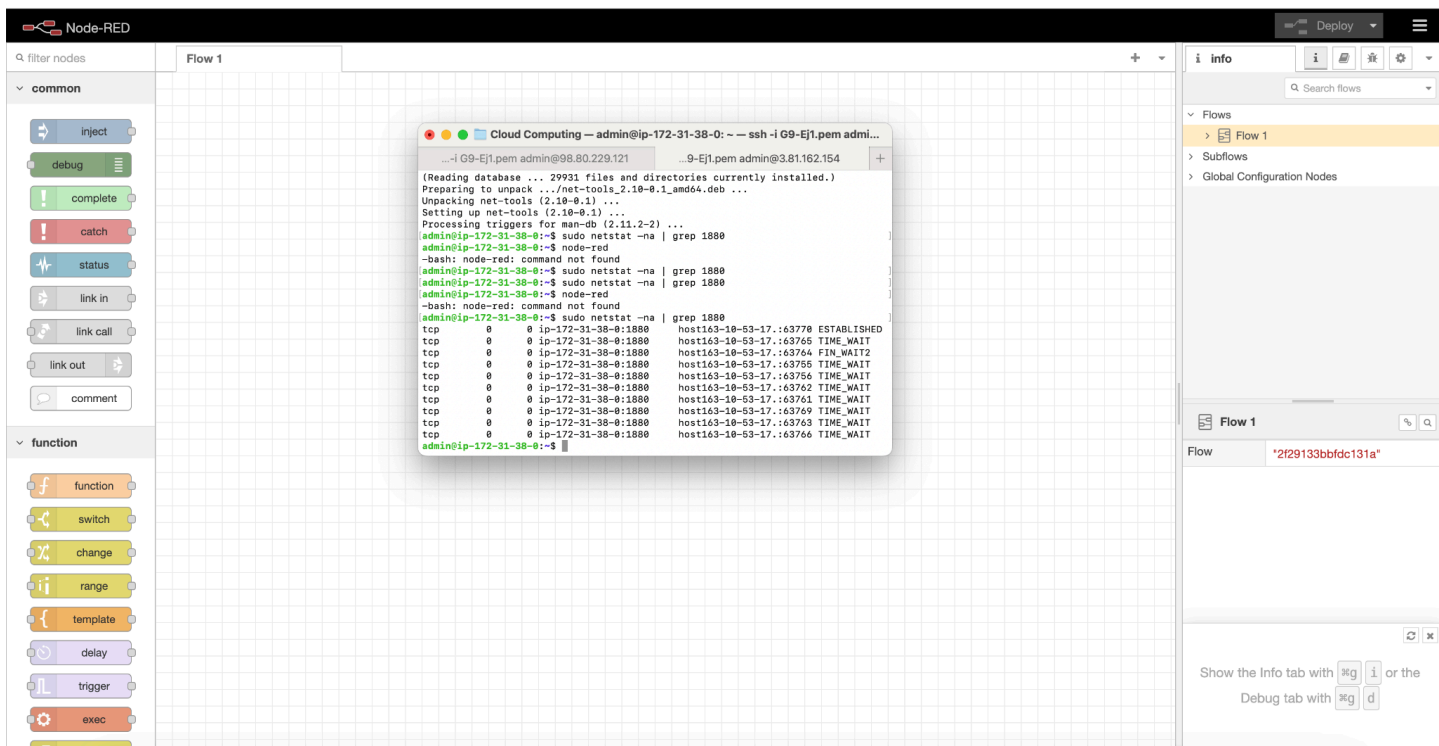
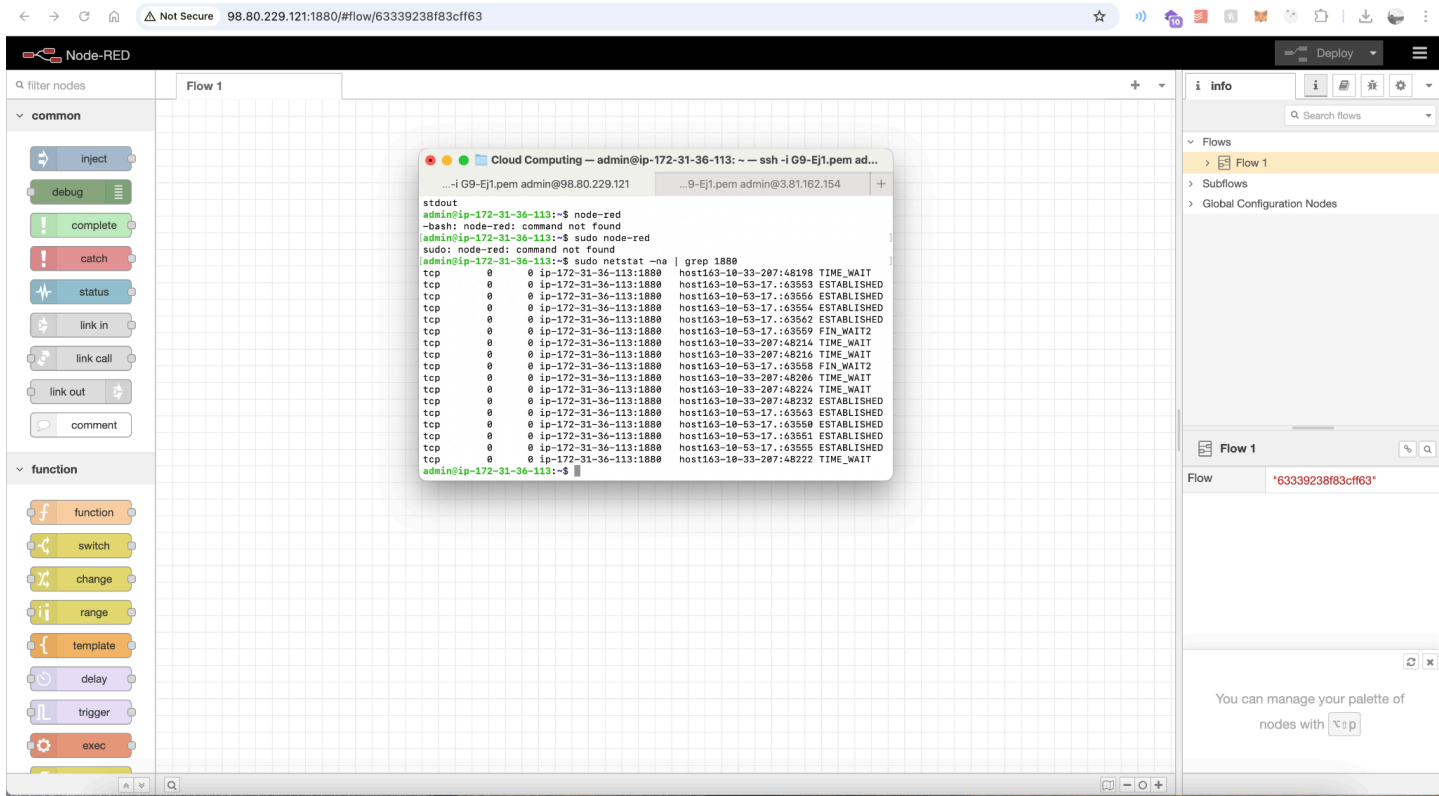
Search

Manage tags Edit inbound rules

Security group rule...	IP version	Type	Protocol	Port range	Source	Description
sgr-06eb8b28a1c117f26	IPv4	Custom TCP	TCP	1880	163.10.0.0/16	-
sgr-061f05097344ec884	IPv4	SSH	TCP	22	0.0.0.0/0	-

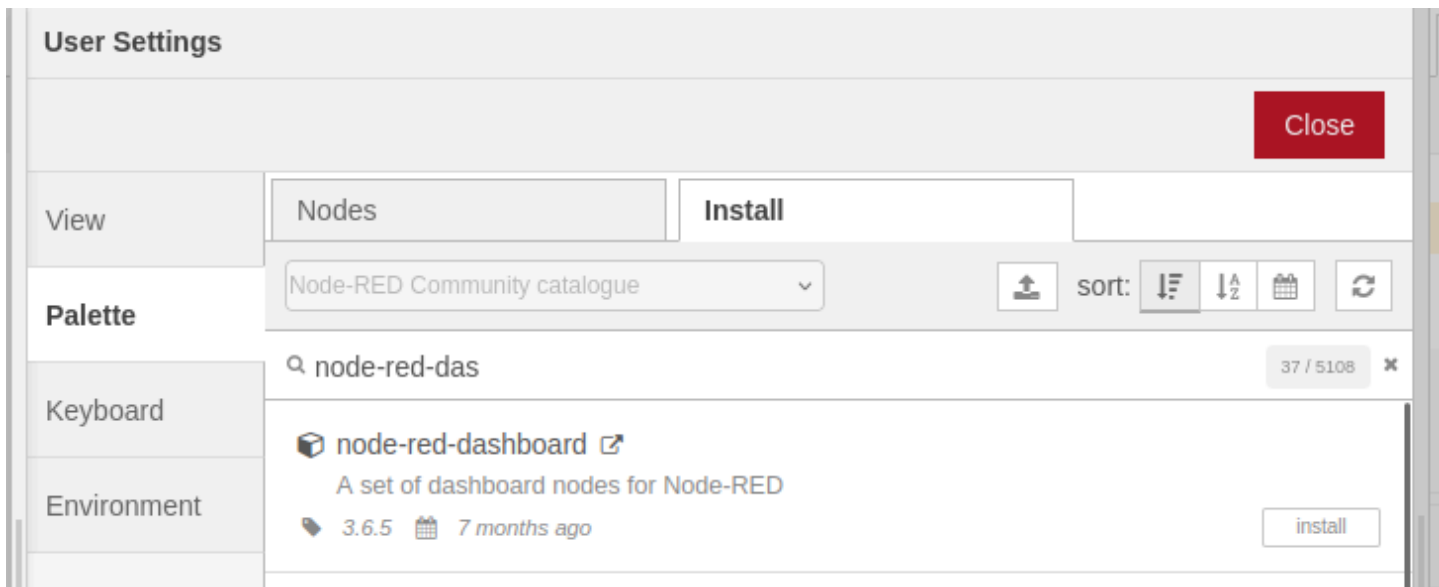
Agregamos regla de entrada para el puerto TCP 1880

Usando net-tools verificamos la instalación de Node-Red en cada una de las instancias.



3) Instalar el nodo “node-red-dashboard”.

Para instalar node-red-dashboard en ‘Manage Palette’ instalamos el paquete de nodos

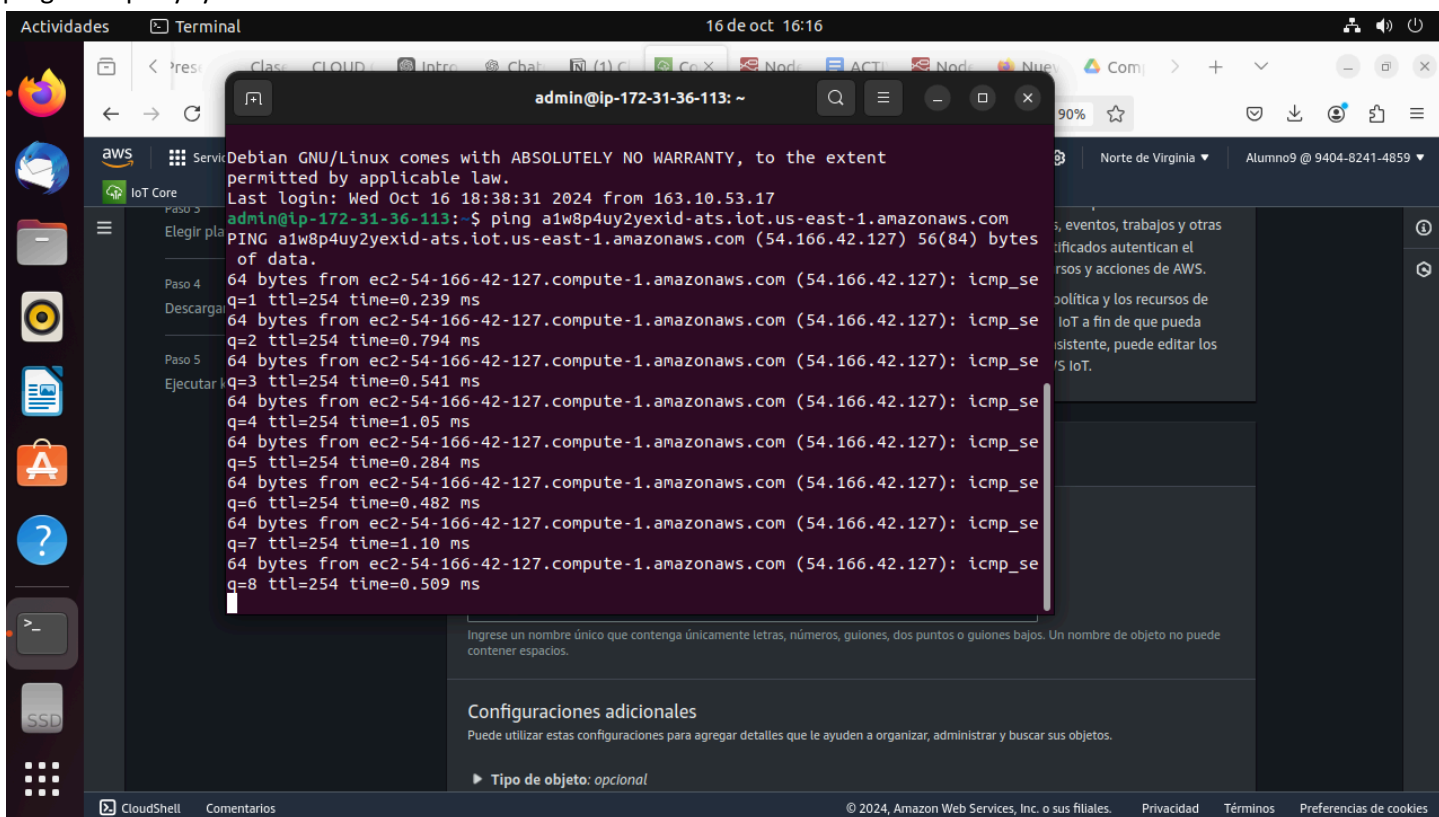


4) Registrar ambas instancias como objetos/dispositivos en AWS IoT para poder operar mediante el protocolo MQTT.

Verificamos para poder conectar el dispositivo a AWS IoT en cada una de las instancias,

para eso siguiendo los pasos, realizamos un ping desde cada una de las terminales:

ping a1w8p4uy2yexid-ats.iot.us-east-1.amazonaws.com



Tanto como para la instancia A y para la instancia B creamos el recurso y generamos el connection Kit

IoT Core

✓ AWS IoT successfully created thing resource Grupo9B and generated your connection kit.

[Choose platform and SDK](#)

Step 4
Download connection kit

Step 5
Run connection kit

created a connection kit that includes the resources in a zip file that you need to install on your device. The resources in the connection kit are listed below. In this step, you'll install them on your device.

Connection kit

Certificate Grupo9B.cert.pem	Private key Grupo9B.private.key	AWS IoT Device SDK Node.js
Script to send and receive messages start.sh	Policy Grupo9B-Policy View policy	

Download

If you are running this from a browser on the device, after you download the connection kit, it will be in the browser's download folder.

If you are not running this from a browser on your device, you'll need to transfer the connection kit from your browser's download folder to your device using the method you tested when you prepared your device in step 1.

[Download connection kit](#)

Unzip connection kit on your device

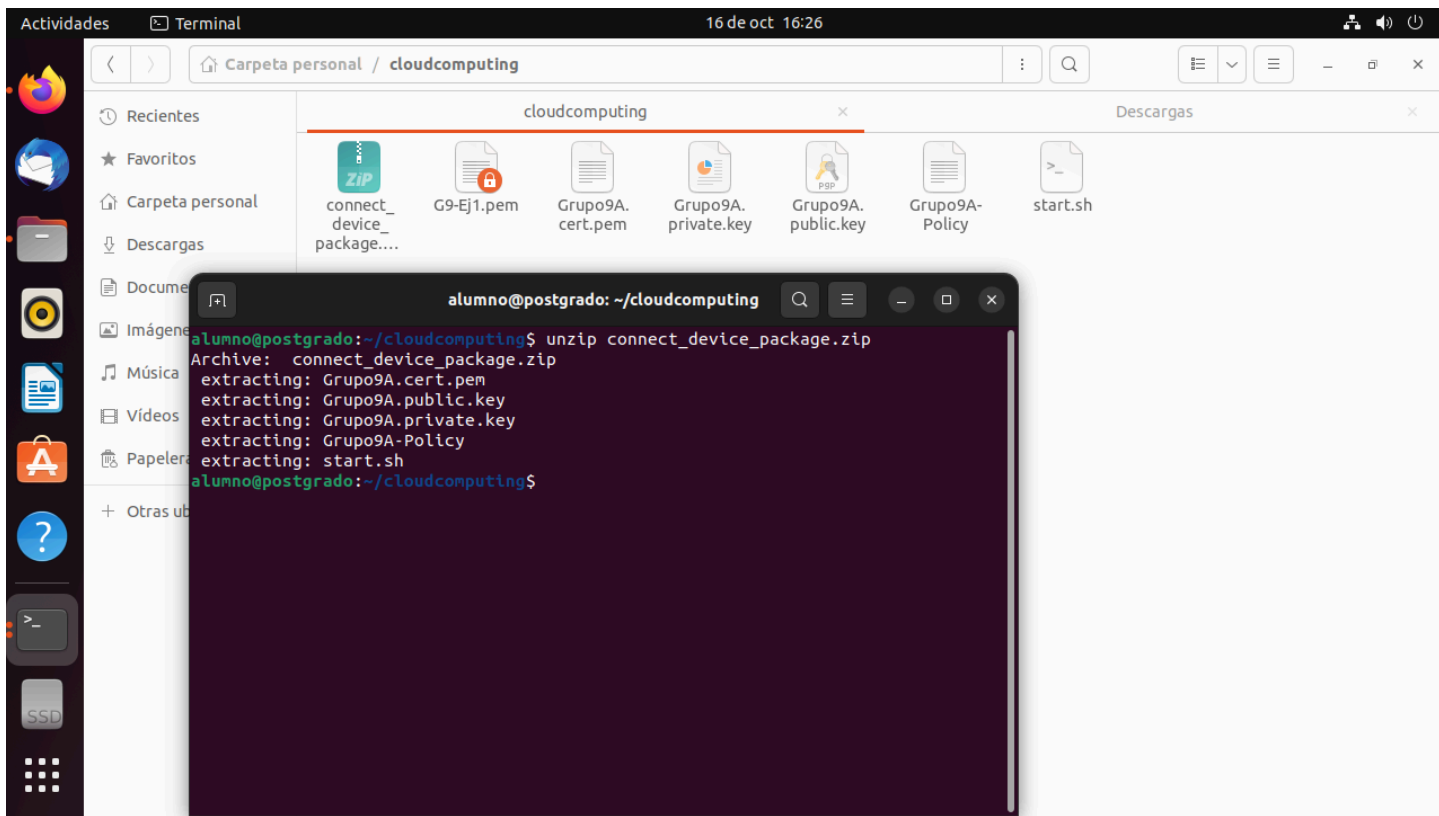
After the connection kit is on your device, unzip it using this command:

```
unzip connect_device_package.zip
```

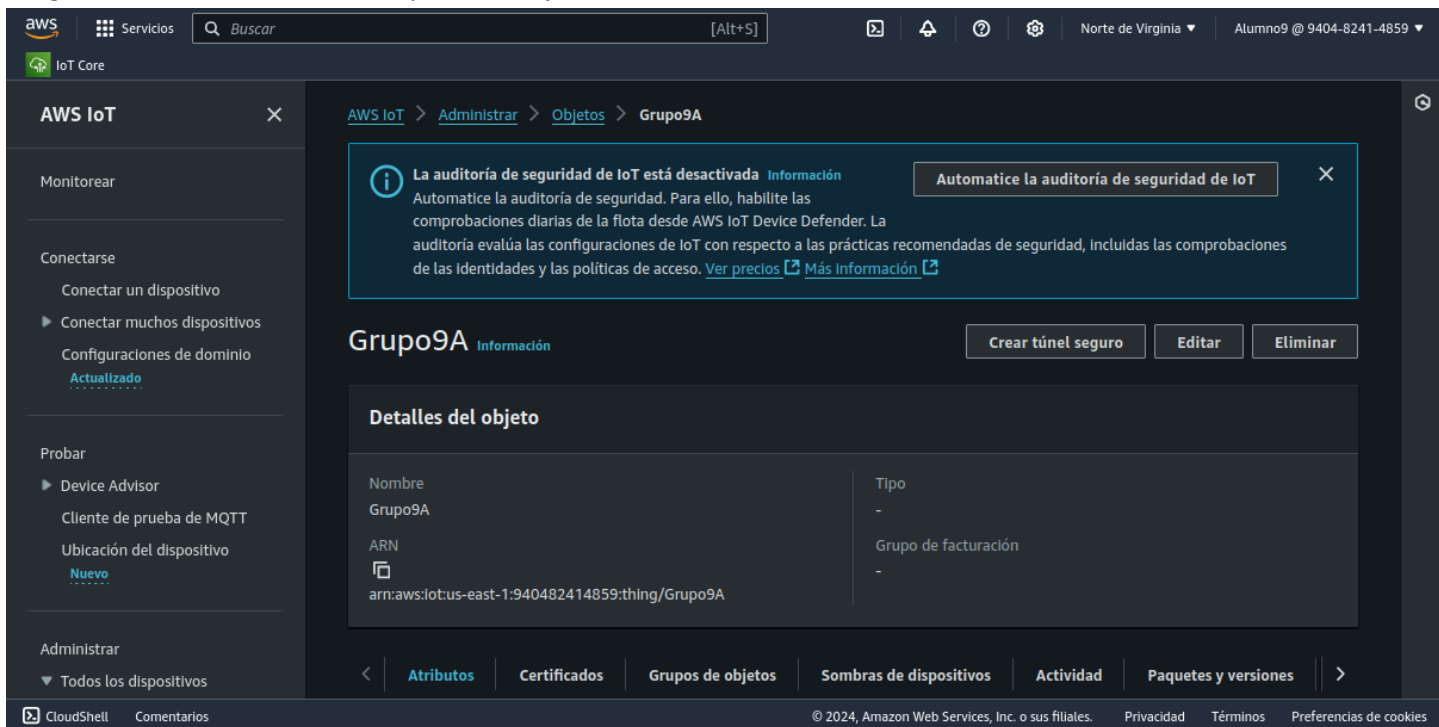
[Copy](#)

unzippeamos en cada instancia el kit de conexión que esto nos va a servir luego para el MQTT.

```
Cloud Computing — gastonginestet@host163-10-53-17 — -zsh — 152x24
...mputing — admin@ip-172-31-38-0: ~ — ssh -i G9-Ej1.pem admin@3.81.162.154
➔ Cloud Computing unzip connect_device_package\ Grupo9B.zip
Archive: connect_device_package Grupo9B.zip
  extracting: Grupo9B.cert.pem
  extracting: Grupo9B.public.key
  extracting: Grupo9B.private.key
  extracting: Grupo9B-Policy
  extracting: start.sh
➔ Cloud Computing █
```



Imágenes de la creación de los respectivos objetos:



AWS IoT

Monitor

Connect

Connect one device

Connect many devices

Domain configurations Updated

Test

Device Advisor

MQTT test client

Device Location New

Manage

All devices

Things

Thing groups

Thing types

Fleet metrics

Greengrass devices

LPWAN devices

Software packages New

Remote actions

Miscellaneous routing

CloudShell Feedback

AWS IoT successfully created thing resource Grupo9B and generated your connection kit.

AWS IoT > Manage > Things > Grupo9B

IoT security audit is off Info

Automate IoT security audit

Automate your security audit by enabling daily checks on your fleet from AWS IoT Device Defender. The audit evaluates your IoT configurations against security best practices, including checks for identities and access policies. [View pricing](#) [Learn more](#)

Grupo9B Info

Create secure tunnel Edit Delete

Thing details

Name

Grupo9B

Type

-

ARN

arn:aws:iot:us-east-1:940482414859:thing/Grupo9B

Billing group

-

Attributes

Certificates

Thing groups

Device Shadows

Activity

Packages and versions

Jobs

Alarms

Defender metrics

Attributes (0) Info

Attributes are key-value pairs that can be searchable or non-searchable. Searchable attributes can be used to filter lists of things without using fleet indexing. Non-searchable attributes can be used to find things, but only when fleet indexing is turned on.

Key

Value

Type

No attributes

Editamos el policy del grupo9a y grupo9b para permitir todo tipo de acción/recurso asi evitamos problemas:

AWS IoT

Monitor

Connect

Connect one device

Connect many devices

Domain configurations Updated

Test

Device Advisor

MQTT test client

Device Location New

Manage

All devices

Greengrass devices

LPWAN devices

Software packages New

Remote actions

Message routing

Retained messages

Security

Intro

Certificates

CloudShell

Feedback

Edit policy: Grupo9B-Policy (Version 1)

Policy statementsPolicy examples

Policy document Info

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

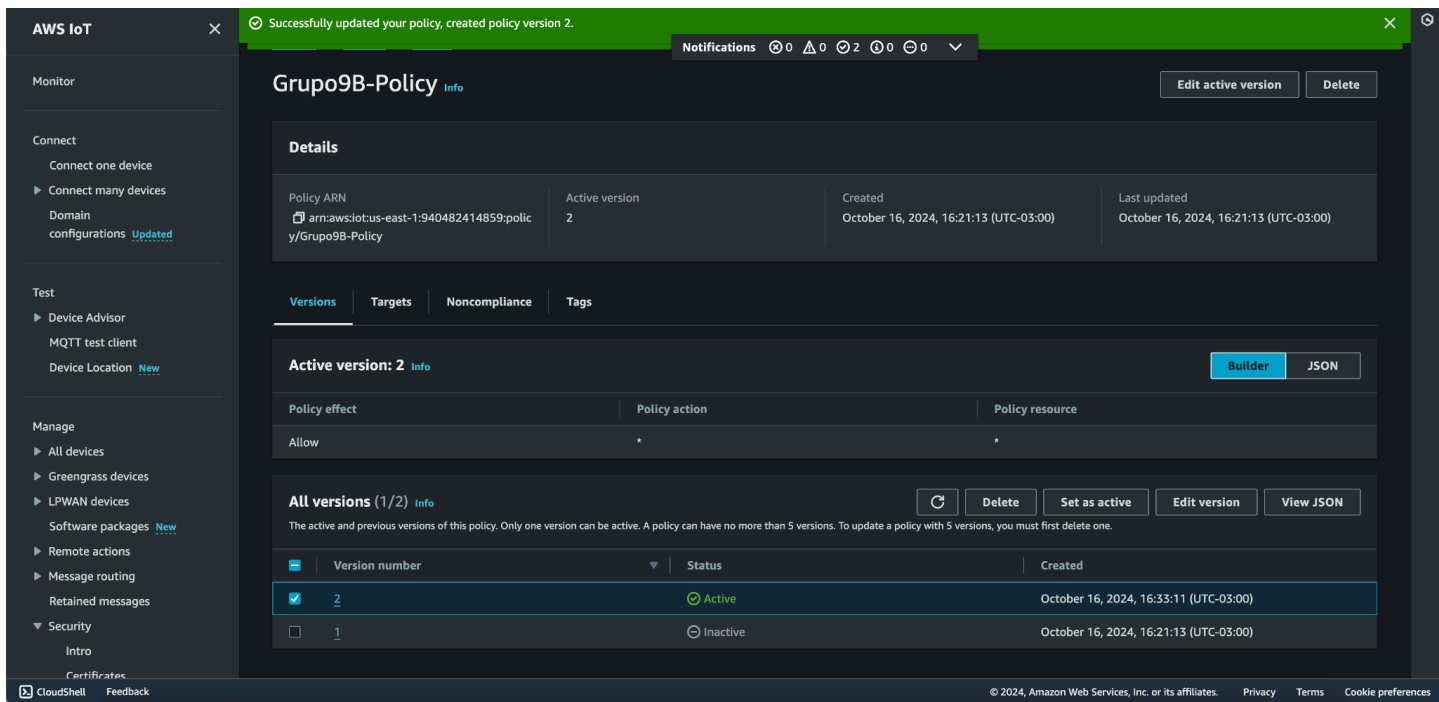
BuilderJSON

Policy document

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "*",
7       "Resource": "*"
8     }
9   ]
10 }
```

JSONLn 10, Col 2Errors: 0Warnings: 0

y activamos la versión actualizada de cada uno:



5) Desarrollar, en la instancia A, la solución de un “CONTROL REMOTO” para gestionar la iluminación de un ambiente: ´botón “ENCENDIDO/ON”, botón “APAGADO/OFF”, informe de estado de la luminaria y funcionalidad “TIMER de 5s” para que activada dicha funcionalidad, al encender la luminaria se apague a los 5 segundos.

Utilizando las herramientas que nos provee node-red y node-red-dashboard insertamos unos botones de node-red-dashboard, lo configuramos para que se pueda mostrar y que tengan el funcionamiento requerido

imagen del proceso de creación, estabamos probando que se muestre el estado del led:

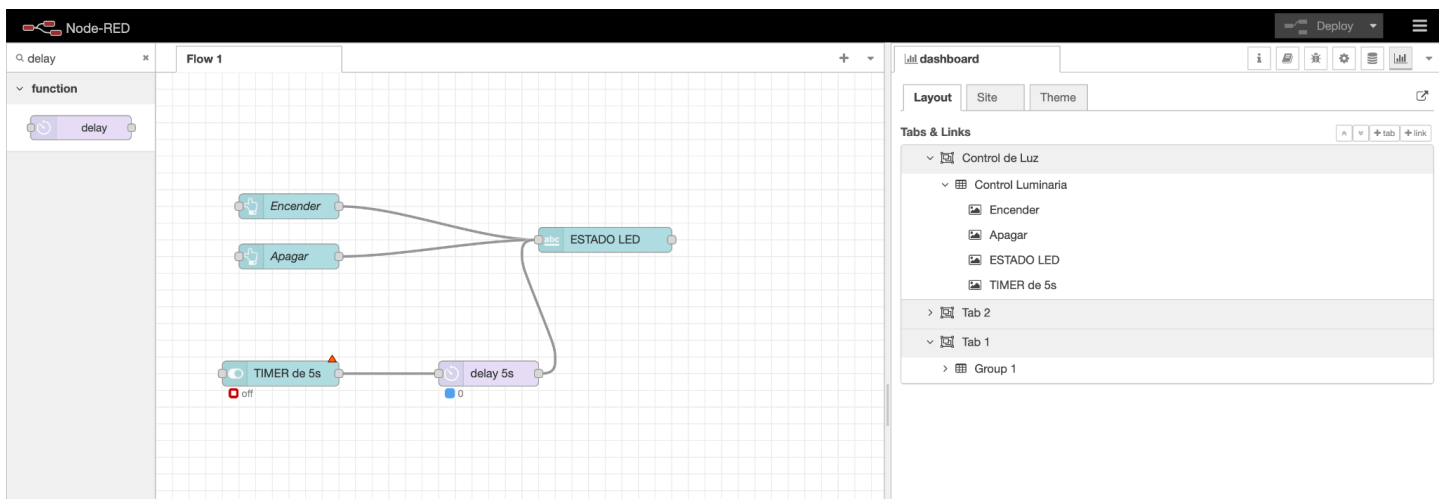
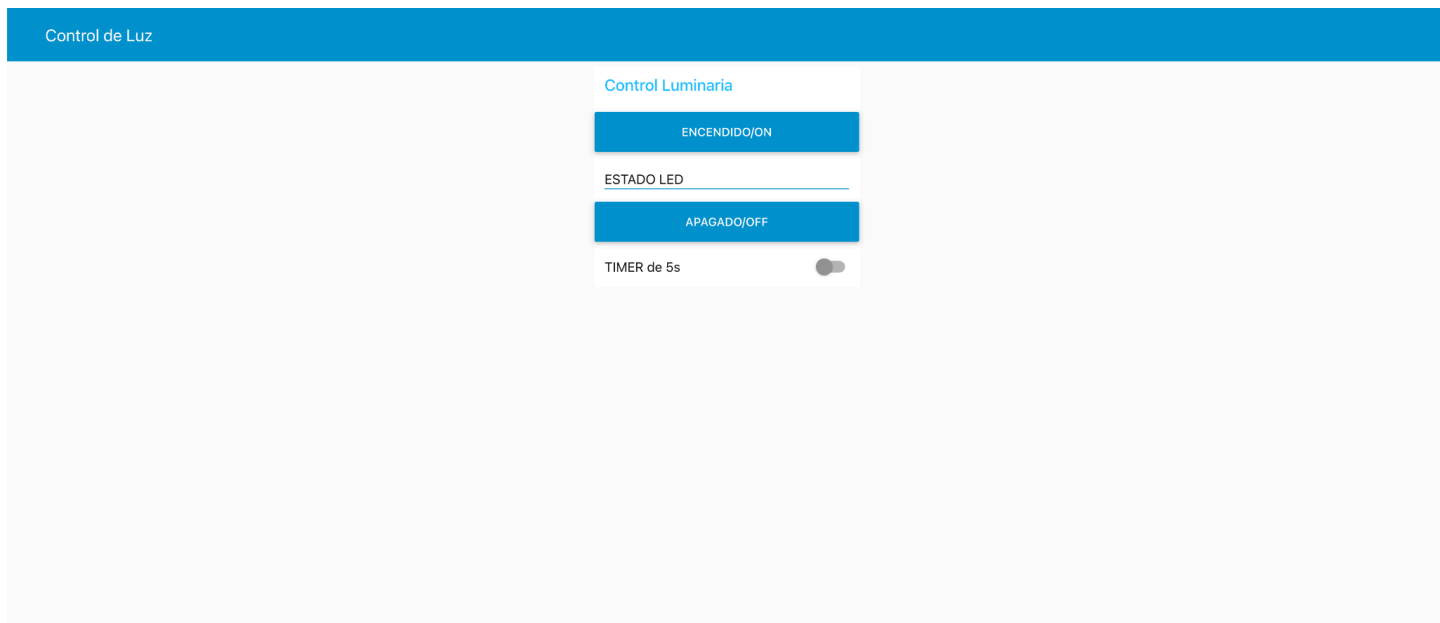
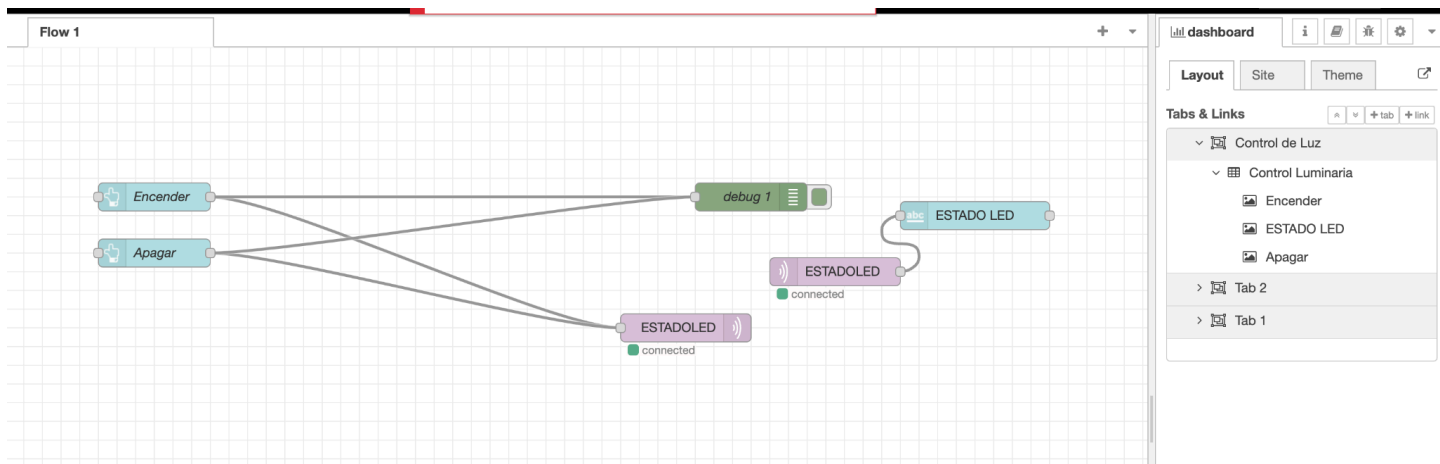
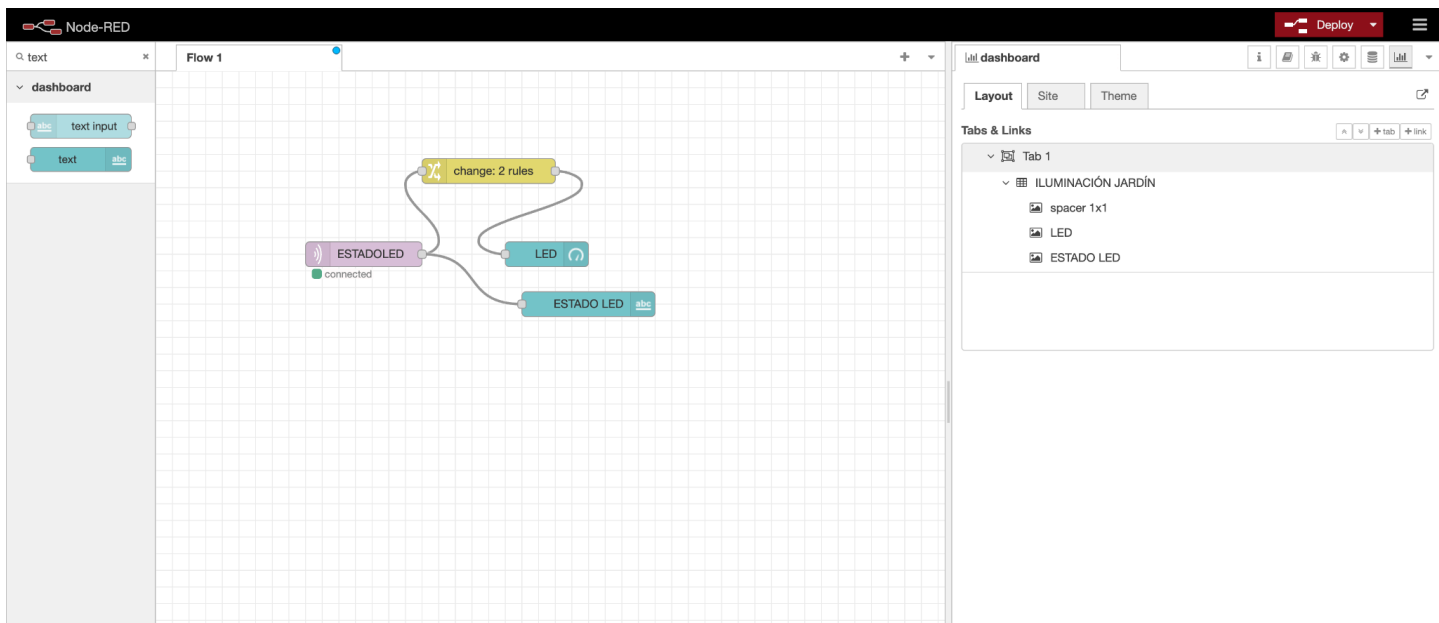


imagen del flujo terminado.



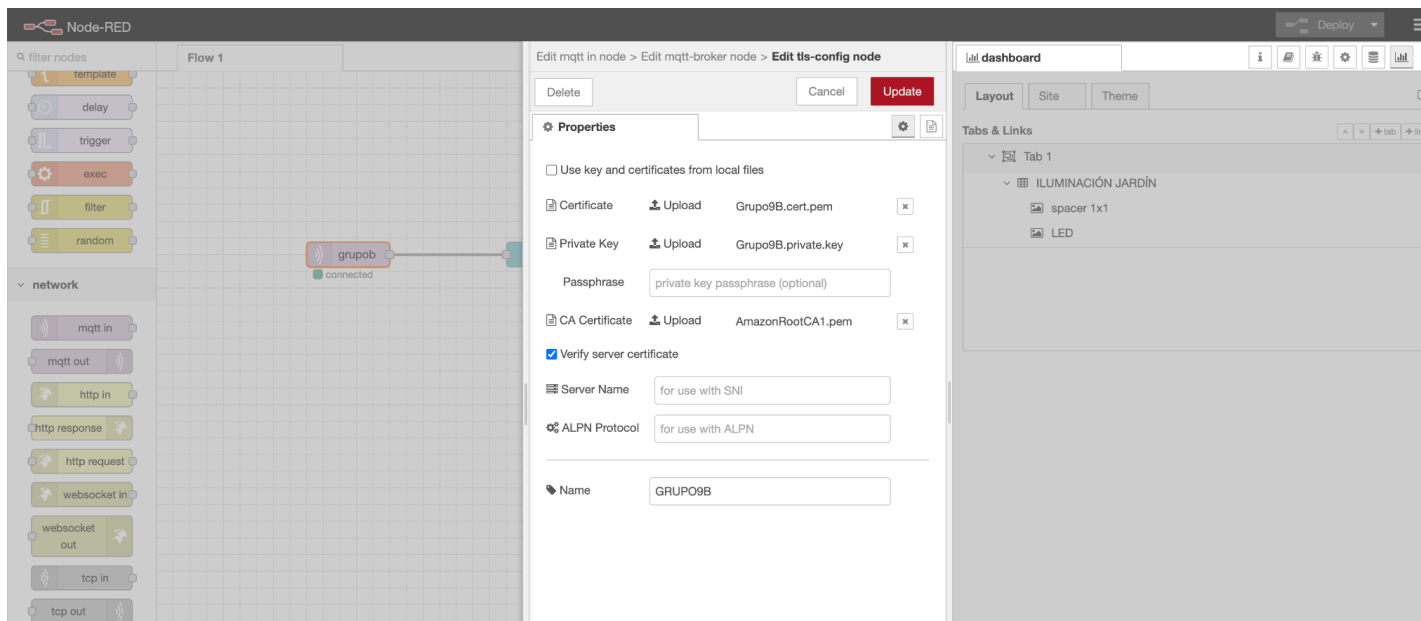
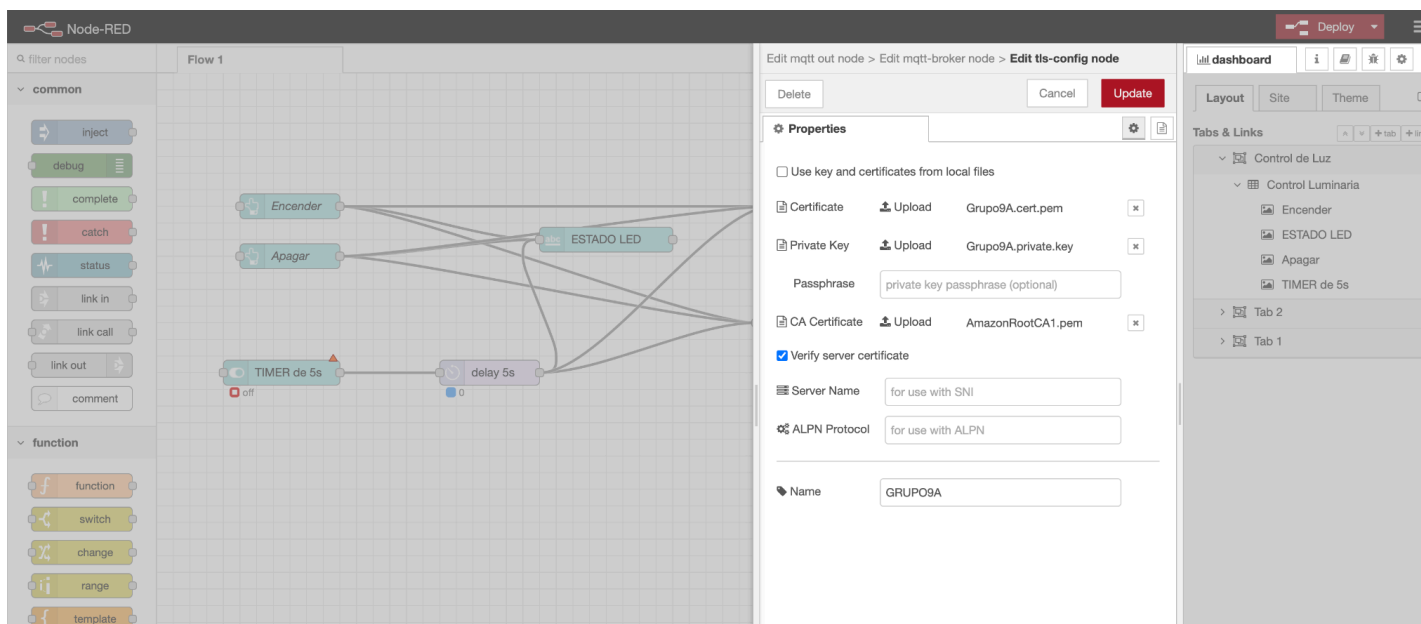
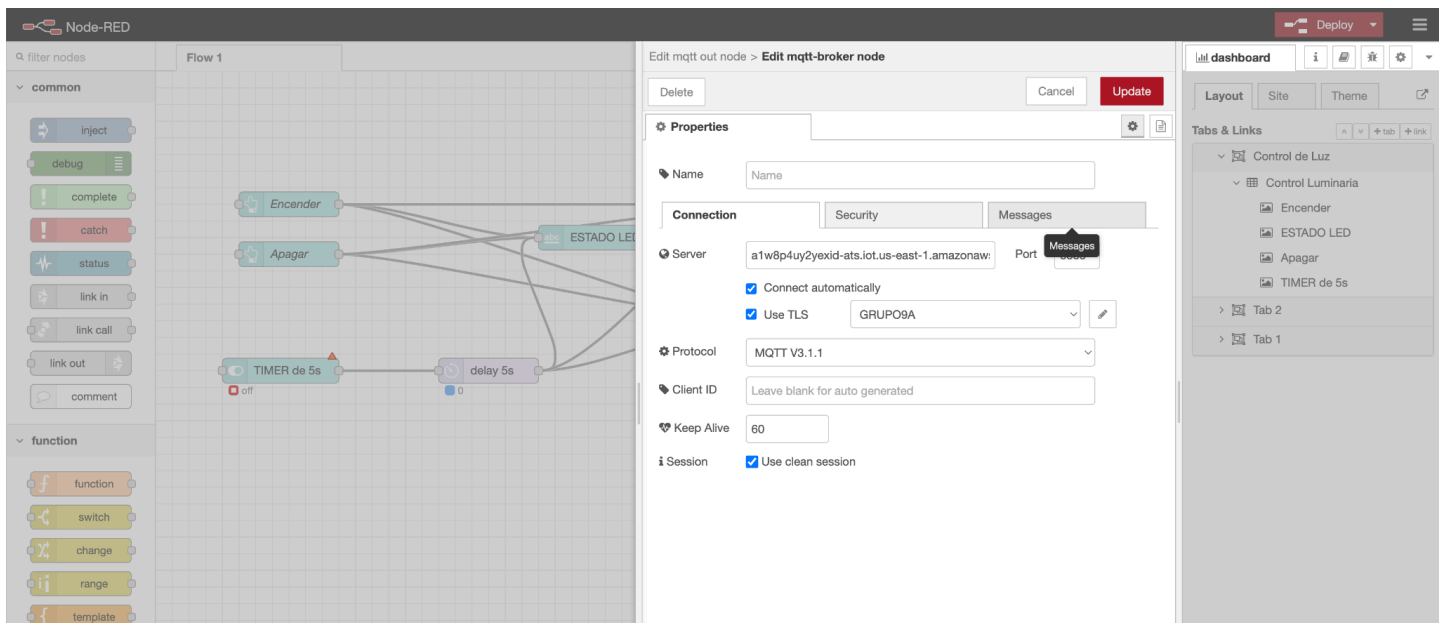
6) Desarrollar, en la instancia B, la solución para controlar la “ILUMINACIÓN” de un ambiente simulando el “ENCENDIDO/APAGADO” de la luminaria, informando el estado de la “LUMINARIA” y el estado del “TIMER”.

configuramos un nodo que recibe como string un “ENCENDIDO” / “APAGADO” y al grafico le devuelve un 1 o 0 respectivamente, así logramos mostrar el gráfico.



7) Las instancias A y B deben comunicarse por medio de MQTT utilizando el bróker de AWS IoT.

con lo generado en el paso 4, configuramos los MQTT requeridos utilizando la clave privada, el .pem y el CA certificate de Amazon.



DEMOSTRACIÓN DEL FLUJO: <https://youtu.be/fh9rDHTiixY>