MODUL

LKS PROVINSI JAWA TIMUR

CLOUD COMPUTING

2023

Description Project (6 Hours)

Modul Name: IoT Project

A project description for monitoring temperature and humidity using DHT22 and ESP32 Arduino, utilizing MQTT protocol connected to DynamoDB, AWS S3, and MySQL databases and implemented with Python, while visualization is done with Node-RED. In addition, each MQTT broker will be connected to EFS and REDIS.

The project for monitoring temperature and humidity using DHT22 and ESP32 Arduino is an environmental monitoring system that measures air temperature and humidity. The system utilizes the DHT22 sensor for measuring temperature and humidity, and the ESP32 Arduino microcontroller to send the measurement data via MQTT protocol to a server.

This server will be connected to several AWS services, including DynamoDB for data storage, AWS S3 for file storage, and MySQL for further data analysis. Additionally, the system will use the Python programming language to access and process data from the AWS server.

For data visualization, the system will utilize Node-RED, a visual programming platform that enables users to create dashboards and applications quickly and easily. This dashboard will display real-time temperature and humidity data using graphs and other visual displays.

Each MQTT broker in the system will be connected to Amazon Elastic File System (EFS) and REDIS. EFS is a shared file system, while REDIS is an open-source database used for session and cache data storage.

By combining cutting-edge technologies such as AWS and Node-RED, this system can provide accurate and real-time information about air temperature and humidity, making it very useful for monitoring environments such as server rooms, warehouses, or other storage areas.

Task

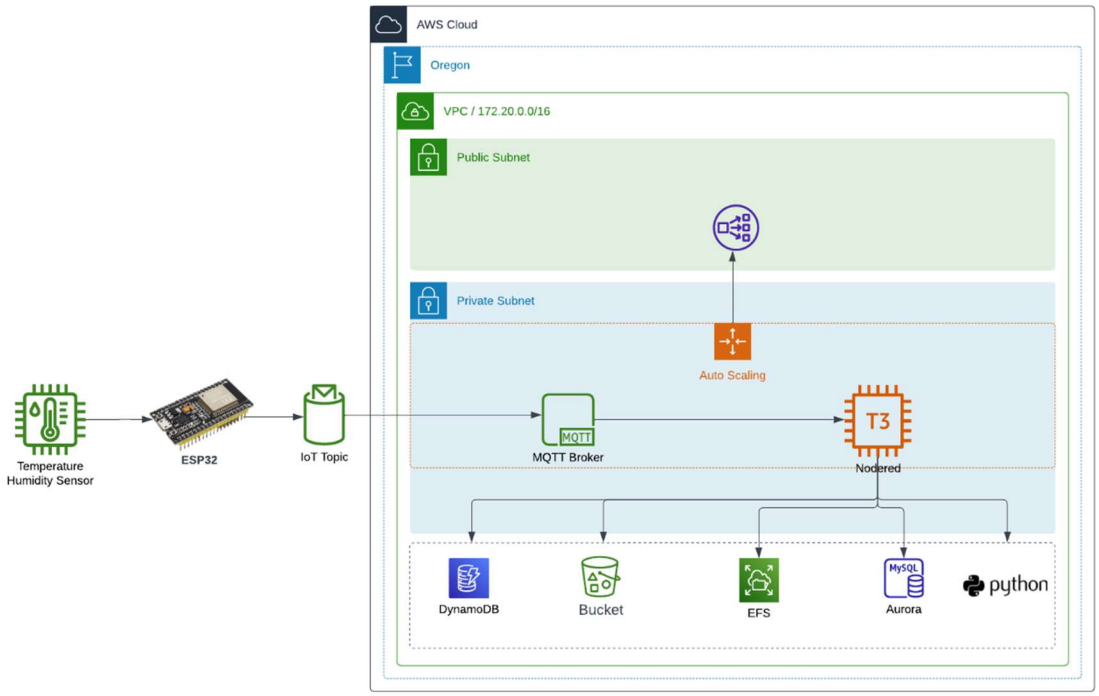
1. Share your AWS Professional on this link https://s.id/akunccjatim2023.
2. Read the documentation thoroughly (Outlined below).
3. Please view and understand the architecture section.
4. Log in to AWS Console
5. Set up your VPC Configuration. You can read VPC Configuration Details in the Architecture Details section.
6. Set up a security group. You can read the additional security rules in Security – Service Configuration
7. Set up your MQTT broker server; you can read MQTT Broker Configuration details.
8. Set up your Nodered Server. You can read Nodered Server Configuration details.
9. Set up the Python Programming. You can read the Pyhton Configuration details.
10. Set up your Datalog for the store data to DynamoDB and S3 Bucket. You can read Datalog Configuration details.
11. Set up your database; you can read Database Configuration details.
12. Set up the bucket; you can read Bucket Configuration details.
13. Setup your Auto Scaling and Load Balancer from Launch Template; you can read Auto Scaling and Load Balancer details.
14. Set up your sensor and microcontroller; you can read Internet of Things (IoT) Configuration details.
15. Test your MQTT Broker from the Microcontroller and Nodered.
16. Submit your MQTT account, password, MQTT Load Balancer and Nodered Load Balancer to http://gg.gg/cclb2023jatim

Technical Details

1. The IoT project can connect to the MQTT broker.
2. The visualization is deployed as Node-red.
3. The MQTT, Log, and Application server must be deployed in the t3.micro instance type, and you can't access the remote server port SSH, Session Manager, and other consoles. If your remote server uses port SSH, Session Manager and other consoles will be grounds for **disqualification**.
4. You should be prepared for 1000 concurrent requests to the Application server.
5. Public load balancer must contain a dual-stack (IPV4 and IPV6) for availability zone subnet using the internet-facing for MQTT Broker, Nodered.
6. While you create an MQTT Broker, Log, and Nodered Server service for Auto Scaling, you must use the launch template and user data; if the configuration does not follow the instructions, you will lose the point.
7. The log server requires a DynamoDB and S3 Bucket connection. As with the previous deployment, you can set up the database and a file storage directory on each instance with the application deployed. However, that will be far less efficient than creating a centralized solution. The more efficiently you run the infrastructure, the faster your servers will respond to requests and the more points you earn. You can read service instructions in the Service Detail section.
8. The Nodered requires a connection to the MySQL database. As with the previous deployment, you can set up the database and a file storage directory on each instance that has the application deployed. However, that will be far less efficient than creating a centralized solution. The more efficiently you run the infrastructure, the faster your servers will respond to requests and the more points you earn. You can read service instructions in the Service Detail section.
9. Following the security service details section, you should create a different security group.
10. IoT devices (ESP32 and DHT22) must be connected to MQTT Broker, and Nodered.
11. Remember to label every service you have created, like VPC, security group, EC2 Instance, and everything else you have made except those created automatically like an instance created automatically by auto-scaling. The more attention to the little things, the more point you will earn.

12. Remember to fill in each description and tag of the services.
13. You must set the region using **Oregon**.
14. The base OS chosen is **Amazon Linux 2** (https://aws.amazon.com/amazon-linux-ami/). This distribution was selected for its broad industry support, availability of support, and excellent integration with AWS.
15. The visualization uses Nodered. You can read the documentation from https://nodered.org/docs.

Architecture



VPC Configuration

Building a network infrastructure before starting all the tasks would be best. You should create VPC with CIDR 172.20.0.0/16. As shown in the following topology, you must set up VPC with the following condition:

1. Public Subnet must have two availability zone.
2. Public Subnet must contain IPV4 and IPV6 subnet.
3. IPv6 public subnet prefix is /64for each AZ.
4. You should create a public subnet for 30 portions of IP Addresses for AZ from IP 172.20.1.0/24.
5. You should create an MQTT private subnet for 35 portion IP Addresses for AZ from IP 172.20.2.0/25.
6. You should create a Nodered private subnet with 10 portion IP Addresses for AZ from 172.20.5.0/26
7. You should create a database private subnet for 25 portions of IP Addresses from IP 172.20.4.0/24.
8. You should create an EFS private subnet for 50 portions of IP Addresses for AZ from IP 172.20.3.0/24.

Security Configuration

Security is essential and should be kept from being passed down while building infrastructure. You must include security planning on your infrastructure. The admission apps have sensitive data such as student personal data, so you must create a security group. You can follow the security rules below to secure the app data.

- You should create a security group allowing only TCP ports 1883 to access public devices and public Load Balancer. The security group doesn't allow anyone to access other ports from outside.
- You should create a security group that only allows TCP ports 80 and 443 to be accessed from outside and doesn't allow anyone to access other ports from outside. This security group will be used for public load balancers.
- You should create a security group for public load balancers Nodered, use TCP port 1880 can be accessed from outside, and doesn't allow anyone to access other ports from outside.

- You should create a security group that only allows TCP port 2049 for EFS can be accessed from MQTT Broker Server and Log Server and doesn't allow anyone to access other ports from outside.
- You should create a security group that only allows the public to access TCP port 3306 for the Database and doesn't allow anyone to access other ports from outside.

MQTT Broker Server

MQTT uses a publisher-subscriber architecture, where a publisher is a device that sends messages, and a subscriber is a device that receives messages. When a message is sent, devices that subscribe to the same topic will receive the message. In this case, the MQTT used is mosquitto.

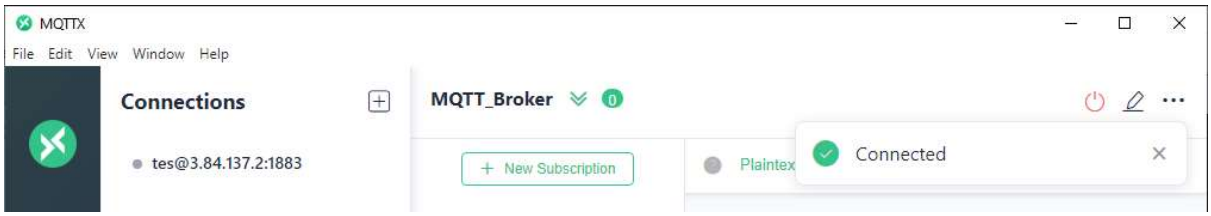
You can configure mosquitto in the epel feature on Amazon Linux 2 using userdata.

```
sudo amazon-linux-extras install epel
sudo yum install -y mosquitto
```

MQTT cannot be accessed by users who do not have an account or are anonymous. Therefore, you must configure to disable the anonymous feature in the **/etc/mosquitto/mosquitto.conf** file, input the command **allow_anonymous false** to file **mosquitto.conf** and must create a user and password for MQTT access. The command to create a user and password is as follows:

```
sudo mkdir -p /etc/mosquitto/passwd
sudo mosquitto_passwd -b /etc/mosquitto/passwd <username> <password>
```

Next, please test the MQTT configuration using the MQTTX software. If successful, you will be able to see a successful notification.



Nodered Server

This node-red server displays data results from MQTT sourced from the DHT22 sensor before you use visualization. The thing you do most do the installation stage of Nodered.

The Nodered was developed with JavaScript with one service running in it.

Both services were developed with JavaScript and running with node.js for the runtime. Firstly, you need node.js and NPM packages installed on your system. You can follow this LKS documentation for node.js installation, or you can use AWS documentation installation in AWS Docs; for LKS documentation, you can follow the below.

First, you must enable the node.js yum repository in your system provided by the Node.js official website. You also need development tools to build native add-ons to be installed on your system.

```
sudo yum install -y gcc-c++ make
curl -fsSL https://rpm.nodesource.com/setup_16.x | sudo bash -
```

NPM will also be installed with node.js. If you have done with the repository, next you can run **yum install -y nodejs** and **npm install -g –unsafe-perm node-red** command. Node-red can be run automatically. If successful, check the node-red via browser for the example **http://yourloadbalancer:1880.**

Node-red Dashboard

If you successfully for the installation node-red, after that, you must config the dashboard. Access the dashboard via the address **http://yourloadbalancer/ui**. Install library via pallet for MQTT, dashboard and MYSQL.

Setting Node-red must be connected to MQTT, dashboard using gauge and store data to MySQL. If you don't understand, check the documentation node-red on this link: **https://nodered.org/docs**/.

Data log

The data log is a function that saves data to DynamoDB and S3 Bucket. Please do a configuration where you must create DynamoDB with the following structure: table

name structure **dynamodb_nourut**, partition key name is **sensor_id** and sort key using a **timestamp**. As for the bucket, please create a bucket first with the name **bucketlks_nourut**, where the data can be accessed by anyone and made into an archive after 30 days. After 1 year, the data will be permanently deleted.

Python Programming

In the Nodered Server, you must configure the Python because the case must run the application for the store datalog. Install Python 3.7 using the command sudo yum install python3 and install the library json, boto3, paho-mqtt using command pip3 install paho-mqtt boto3. After that, install Python and the library. You must be running the program from https://github.com/handipradana/lksprov.git using userdata and running the automatic program using a cronjob.

File Storage

A central file storage location is NOT a requirement for the application to operate. The application will serve some requests faster with a centralized file storage solution. As with the previous service example, you could create a centralized file storage solution with Elastic File Storage to store logs and cache files from the application. You can use a local directory to save the log files. Still, a shared storage solution or the ability to share files to each instance will allow you to have a centralized log file, making the maintenance process easier. You must set the EFS multiple AZ, and the file cache needs a central file storage solution for consistent data. For the lifecycle management, you must set the transition to IA for 30 days and transition out to of IA on first access, then mount EFS to Nodered Server using userdata.

DATABASE

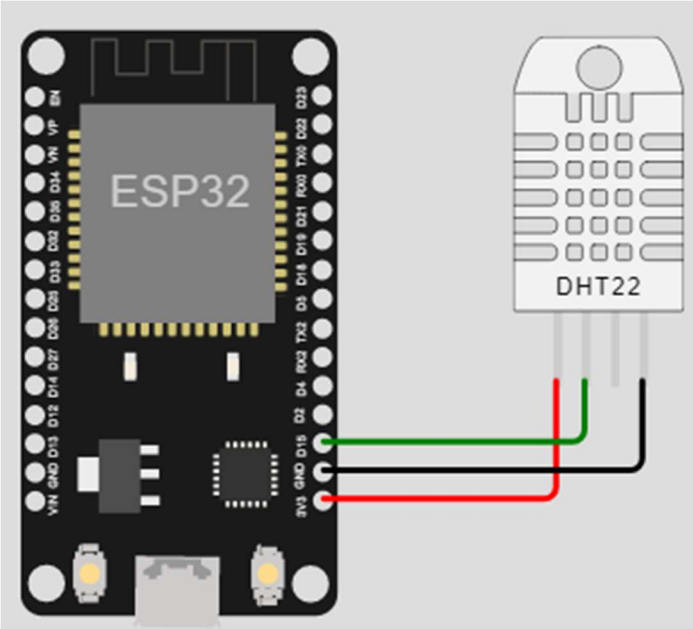
The application must use a reliable Relational Database Service (RDS) with a **serverless V2** concept with Aurora MySQL 3.0.2 (compatible with MySQL 8.0.23). The template type used is **Dev/Test**. The database name is **log_mqtt**, the username is **admin**, and the password is **lks2023!@**. Use serverless instance configuration. Increase the database unit capacity automatically according to the database load demand. Ensure the **minimum ACUs is 2GiB**, and the maximum unit capacity is no more than **16 GiB RAM** to save costs. Connectivity using **Dual-stack mode**. Once

your serverless is provisioned, you must create the necessary tables to serve the requests, the name of table is **tbl_data**. You can use the table definition below using query or HeidiSQL software and create fields as follows:
id INT(11), AUTO_INCREMENT, Primary Key
temperature VARCHAR(50), NULL
humidity VARCHAR(50), NULL
timestamp (TIMESTAMP)

After the create table, you must setting nodered store data from MQTT Broker to database, if you don't understand, you can check the documentation from this link https://flows.nodered.org/node/node-red-node-mysql

INTERNET OF THINGS

Connect the DHT22 sensor to the ESP32's GPIO pin, for example, to pin 15.



Begin to program the ESP32 and DHT22, and the program code can be accessed at the following address http://github.com/handipradana/lksprov.git Configure the program in the previous link, and replace the one marked with XXX, adjust the username, password, and MQTT broker link conditions with the one that was previously made.

```
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connected to WiFi
Connecting to MQTT...
Connected to MQTT
Suhu: 26.40||Kelembaban : 74.00
```

Testing