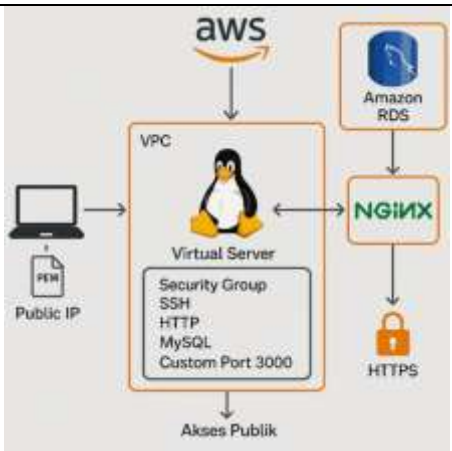
LOMBA KOMPETENSI SISWA (LKS)
SEKOLAH MENENGAH KEJURUAN
TINGKAT KAB. TASIKMALAYA
TAHUN 2026

NASKAH SOAL
*(Terbuka/~~Tertutup~~)

Bidang Lomba

CLOUD COMPUTING

SOAL PRAKTIKUM (Topik I)

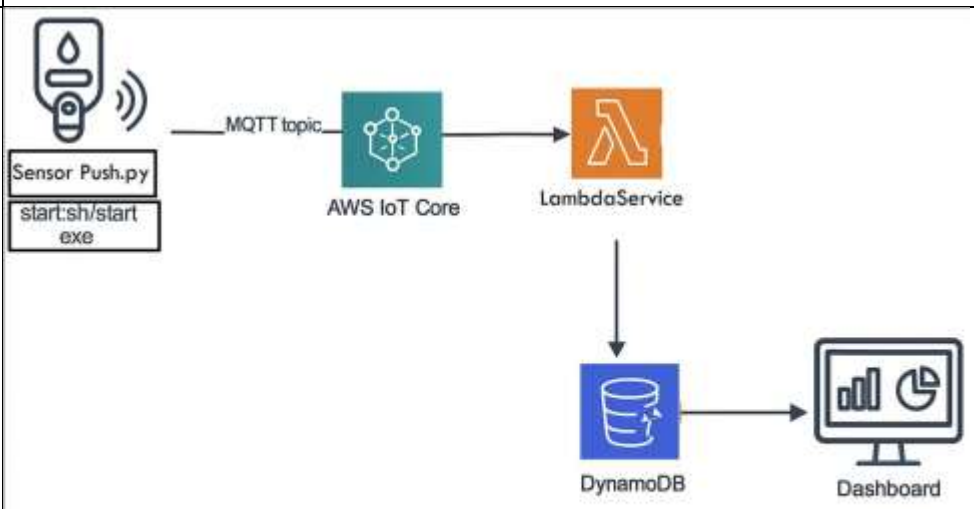
Topic 1	**Deploy Dynamic Web Applications on AWS**
Duration	**4 Hours**
Need	1. AWS IAM Accounts 2. Web Browser 3. Terminal for SSH (Putty/Termius/etc)
Task	 A company wants to migrate their Node.js web application to AWS to run data management processes and serve large numbers of users. You are assigned to: 1. Create a Virtual Server with type t2.micro with Linux operating system (you can choose any) 2. For development purposes, you are required to enable PORT for **SSH, HTTP, HTTPS, MySQL** and Custom Port **3000 services.** 3. Make sure you have the Public IP Address information and .pem file for the SSH process from your laptop. 4. If you need an Access Key, you are allowed to ask the judge 5. **Checkpoint I :** Up to here please send the Public IP to the jury to do the DNS Pointing process. Also make sure the jury **has confirmed** the DNS information to you 6. After you have successfully logged in to the server, you must install nodeJS and NginX. 7. Once successful, you are tasked with creating a MySQL database on the RDS service with the format: db_kota/kabupaten (eg: db_kotajakarta) **8. You don't need to create any table/schema.** 9. Make sure you have the database credentials that will be used in the connection configuration at the application layer later.

	10. You can clone the NodeJS-based Web application at the following link: **https://github.com/adhimaswisnuyudo/lks-jabar-2025-cloud-computing-crud-app.git** 11. Please save the project in **/home/your-username/** 12. Configure the Database connection in the file: **/home/your-username/your-project/config/db.js** according to the connection endpoint that was created previously. 13. Run Web Server with PM2 (ProcessManager) so that it can run as a background process. **14. Checkpoint II** Make sure that the website can be accessed publicly by accessing: **http://your-public-ip:3000** and can perform **CRUD activities** 15. Then you are assigned to configure an NginX web server with the server_name **city/district.domainname.xyz (as given by the jury)** 16. Do a reverse proxy in NginX that points to the NodeJS Server (so that when there is a request on port 80 / http, the system will automatically respond with the NodeJS application) 17. **Checkpoint III** Make sure that when you access http://your-domain the server will respond with a NodeJS web application (not Hello World NginX) 18. Configure SSL for your website and make sure the URL https://your-domain can be accessed properly (SSL can use any service including certbot / Let's Encrypt) 19. **Checkpoint IV** Make sure that when accessing http://your-domain , the system will redirect to https://your-domain
Additional information	You are allowed to improvise/improve the task by implementing other features as long as: 1. Does not reduce the basic tasks above 2. Do not change/add roles & policies that have been determined 3. No overtime
Output	1. NodeJS-based applications can run CRUD functions and can be accessed via the URL: https://your-domain properly. 2. Take a screenshot of each configuration and provide an easy-to-understand description.

information :

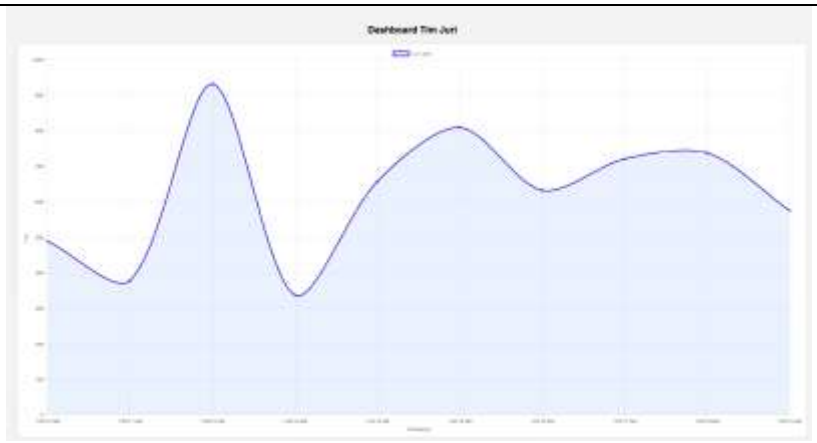
1. If there are any problems related to accounts, roles, policies, etc., please contact the jury.
2. Participants are allowed to improvise tasks taking into account time, ability and without changing the essence of the main task itself
3. Naming each component always uses the contingent name behind it. For example (sensor-light-db-cityJakarta)
4. Participants are allowed to view official AWS documentation and other sites **EXCEPT THOSE OF AI NATURE**
5. Information about Domain Names will be explained on the day of implementation.

SOAL PRAKTIKUM (Topik II)

Topics 2	IoT Sensor Data Monitoring System
Need	1. Installing Python 3.x on your laptop 2. Installing pip / other package managers (paho-mqtt==1.5.0) 3. Code Editor (VSCode / Sublime / etc)
Duration	5 Hours
Task	 1. For A Device with " **Connect One Device** ", give Name **"ThingSensor<City/District Name>** 2. Choose platform in accordance with OS Laptop You, use Python SDK 3. Download Connection Kit 4. Extract connection kit Which Already downloaded previously 5. Change permission files start.sh if you are using Linux/Mac 6. There is a file start.exe so that Can in Execution then run the file 7. Change the aws-iot-device-sdk-python-v2/samples/pubsub.py file with the pubsub.py file on this link https://github.com/adhimaswisnuyudo/lks-jabar-2025-cloud-computing-iot.git as well as change message - device_id in accordance with existing provisions 8. Change And use " **Active Version** " policy Which has been automatically generated with the following conditions: **a. Policy action : Iot:Connect, iot:Publish, Iot:Receive, Iot:subscribe** **b. Effect : Allow** **c. Policy Resources : ***

	9. Change parameter topic on files start.sh with format " **sdk/sensor/<city/district name>** " 10. Running Return start.sh file 11. **Checkpoint I :** Perform testing on the MQTT menu Test Client, And make sure You can subscribe topic which has been set in point 8 12. Make it A Table with DynamoDB with Name **DynamoDB<City/District Name>** with provision **partition_key = "device_id"** And **Sortkey = "time_stamp"** 13. Create a LambdaFunction with the name **LambdaSensor<City/District Name>** And Contents script with **The Lambda Function.py** file is located at https://github.com/adhimaswisnuyudo/lks-jabar-2025-cloud-computing-iot.git 14. Change configuration Name table according to with point 11

	14. Create a Policy with the name **LambdaPutDataIntoDynamoDB<City/District Name>** which aim For give authority in **PutItem** 15. Create an IoT Rules with the name **RuleIotToLambda<City/District Name>** with the statement " **SELECT * FROM '<Topic you created earlier>'** " and point action on Lambda Which Already made on point 12 16. In Page Lambda, make it A Trigger Which refers to the IoT Rules that have been created previously 17. **Checkpoint II:** Test whether Lambda can receive data from IoT And keep into the database using **CloudWatch features** 18. Create a Lambda with the name **LambdaApiGateway<City/District Name>** And Add Policy with the name **LambdaApiGetDynamoDB<City/District Name>** with the condition "can access data in the **DynamoDB Database** " 19. Edit Contents Lambda with LambdaAPI.py file on URL following This : https://github.com/adhimaswisnuyudo/lks-jabar-2025-cloud-computing-iot.git 20. For A Trigger on Lambda with provision type **API Gateway, type : REST API, Security : Open** 21. Trials with method access Endpoint URL on browser until the JSON format appears according to that generated by the Sensor Simulator. 22. Download files dashboard.html on GitHub on, And Point the Endpoint according to what was created in point 20 23. Trials whether dashboard walk with Good or not
Output	1. Simulator Sensor walk with Good 2. MQTT Client on AWS can accept data Which you publish 3. Data stored in a way real time on Dynamo DB 4. Data can published by FIRE Gateway 5. Data served with Good in Dashboard like below this

1. Mark plus If dashboard Can display results in a way real time And up to date
2. Mark plus If dashboard on hosting in platform Cloudfront AWS