LOMBA KOMPETENSI SISWA (LKS) SEKOLAH MENENGAH KEJURUAN TINGKAT PROVINSI JAWA BARAT TAHUN 2024

INFORMASI DAN KISI-KISI

Bidang Lomba

CLOUD COMPUTING

PEMERINTAH DAERAH PROVINSI JAWA BARAT

DINAS PENDIDIKAN

Jalan Dr. Radjiman No. 6 Telp. (022) 4264813 Fax. (022) 4264881
Website : disdik.jabarprov.go.id
e-mail: disdik@jabarprov.go.id/sekretariatdisdikjabar@gmail.com
BANDUNG - 40171

LOMBA KOMPETENSI SISWA (LKS) SEKOLAH MENENGAH KEJURUAN TINGKAT PROVINSI JAWA BARAT TAHUN 2024

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

Bidang Lomba
CLOUD COMPUTING

PEMERINTAH DAERAH PROVINSI JAWA BARAT
DINAS PENDIDIKAN
Jalan Dr. Radjiman No. 6 Telp. (022) 4264813 Fax. (022) 4264881
Website : disdik.jabarprov.go.id
e-mail: disdik@jabarprov.go.id/sekretariatdisdikjabar@gmail.com
BANDUNG - 40171

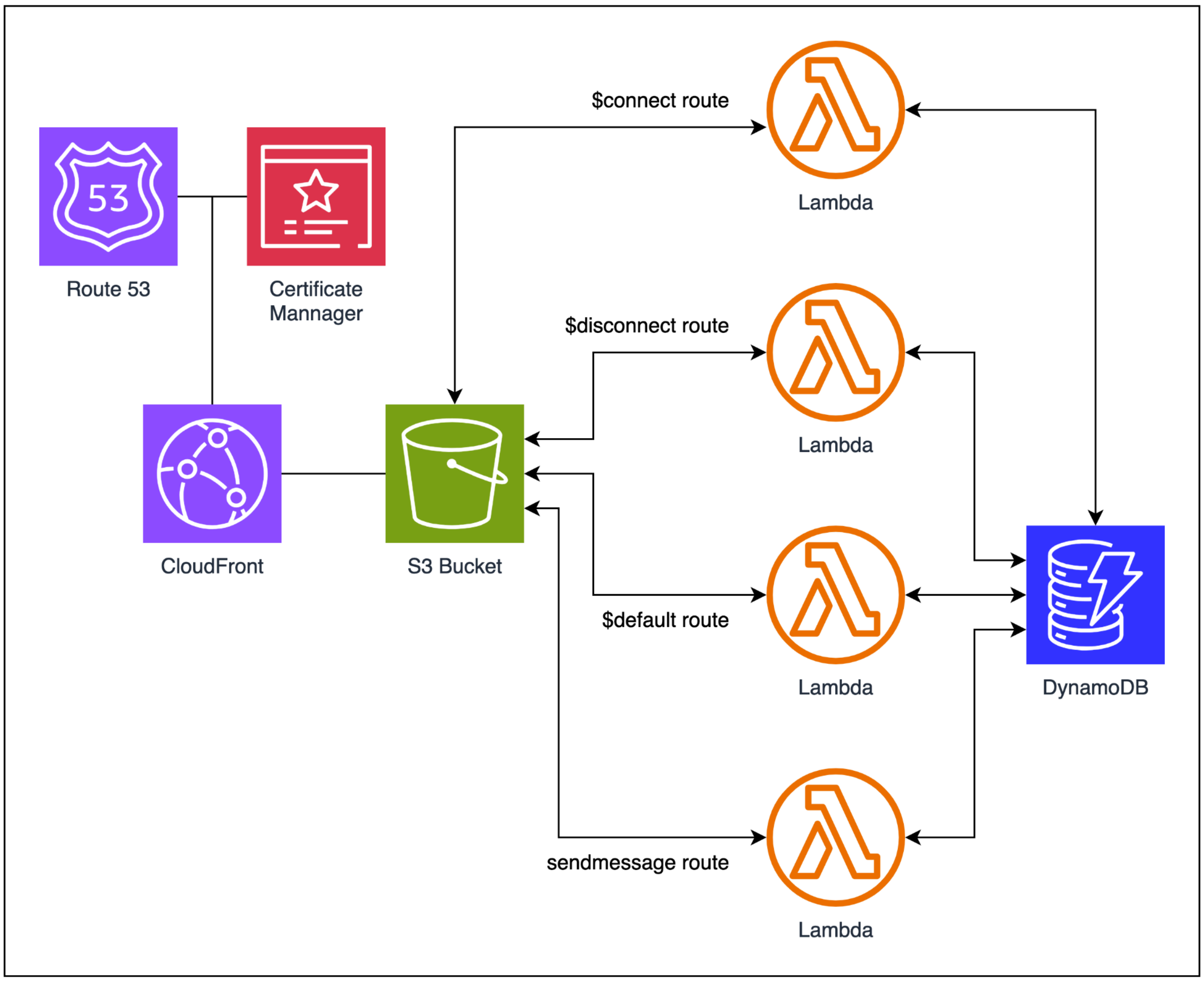
1. Overview

Building a chat application leveraging API Gateway WebSocket, DynamoDB, and AWS Lambda offers a robust and scalable solution. API Gateway WebSocket facilitates real-time bidirectional communication between clients and servers, while Lambda functions handle the application's logic, including message storage and retrieval in DynamoDB. This architecture ensures efficient handling of WebSocket events like connections, disconnections, and message exchanges. Leveraging Node.js enhances compatibility and stability, enabling seamless integration with AWS services. With this setup, developers can create a highly responsive and scalable chat platform, suitable for various real-time communication needs. This time we will create Lambda using Node.js 16 legacy code.

2. General Rules

1. Failure to comply with the rules will result in immediate disqualification.
2. You have 4 hours to finish the tasks.
3. You may use AWS Console and AWS CLI to deploy the solutions. You may not use SAM, CloudFormation or CDK.
4. Between and after the event, you may not access your account. Any activity on AWS during this period is not allowed.
5. During the event, multiple login is not permitted.
6. If you have any questions, do not hesitate to ask.

3. Architecture



1. A website should be hosted on S3 Website Hosting
2. Through provided website user can connect to chat server and chat with other users
3. When connecting, the server will store the connection id in DynamoDB table
4. When disconnecting, the server will remove the connection id from table
5. When user send messages the server will send the messages to other users based on their respective connection id

4. Information

1. This solution must be deployed in ap-southeast-1 region

2. Any deployment outside the designated region will result in major point reduction.
3. Make sure you register your Certificate in the us-east-1 region.

5. Task

5.1 Create DynamoDB Table

Create DynamoDB Table with following configuration

- Name : `lks-jabar-2024-connection`
- Partition key :
  - Name : `connectionId`
  - Type : `String`
- Sort Key : leave empty
- Table settings : choose `Customize Settings`
- Table class : `DynamoDB Standard`
- Read/write capacity settings : `On-demand`
- Tags :
  - Key : LKS-CC-2024
  - Value : lks-chat-table

5.2 Create Lambda Functions

5.2.1 Connect Handler

This lambda will be used to connect to chat server

1. Create Lambda Function with following configuration
   - Option : `Author from scratch`
   - Function name : `api-ws-connect-handler`
   - Runtime : `Node.js 16.x`
   - Architecture : `x86_64`
2. Upload `connect-handler.zip` as code for created Lambda Function
3. Add tag :
   - Key : LKS-CC-2024
   - Value : lks-chat-connect-handler
4. Add **Environment variable** :
   - Key : `table`

    - Value : `lks-jabar-2024-connection`
5. Open execution role and do the following :
    - Add new policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Action": [
        "dynamodb:BatchWriteItem",
        "dynamodb:PutItem",
        "dynamodb:DescribeTable",
        "dynamodb:DeleteItem",
        "dynamodb:UpdateItem"
      ],
      "Resource": "[arn_of_lks-jabar-2024-connection_table]"
    }
  ]
}
```

    - Save policy with name `api-ws-connect-handler-db-policy`

5.2.2 Disconnect Handler

This lambda will be used to disconnect from chat server

1. Create Lambda Function with following configuration
    - Option : `Author from scratch`
    - Function name : `api-ws-disconnect-handler`
    - Runtime : `Node.js 16.x`
    - Architecture : `x86_64`
2. Upload `disconnect-handler.zip` as code for created Lambda Function
3. Add **Tag** panel, add tag :
    - Key : LKS-CC-2024
    - Value : lks-chat-disconnect-handler
4. Add **Environment variable** :
    - Key : `table`
    - Value : `lks-jabar-2024-connection`
5. Open execution role and do the following :
    - Add new policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Action": [
        "dynamodb:BatchWriteItem",
        "dynamodb:PutItem",
        "dynamodb:DescribeTable",
        "dynamodb:DeleteItem",
        "dynamodb:UpdateItem"
      ],
      "Resource": "[arn_of_lks-jabar-2024-connection_table]"
    }
  ]
}
```

- Save policy with name `api-ws-disconnect-handler-db-policy`

5.2.3 Sendmessage Handler

This lambda will be used to send message to chat server

1. Create Lambda Function with following configuration
   - Option : `Author from scratch`
   - Function name : `api-ws-sendmessage-handler`
   - Runtime : `Node.js 16.x`
   - Architecture : `x86_64`
2. Upload `sendmessage-handler.zip` as code for created Lambda Function
3. Add **Tag** panel :
   - Key : LKS-CC-2024
   - Value : lks-chat-sendmessage-handler
4. Add **Environment variable** :
   - Key : `table`
   - Value : `lks-jabar-2024-connection`
5. Open execution role and do the following
   - Add new policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Action": [
        "dynamodb:BatchGetItem",
        "dynamodb:GetRecords",
        "dynamodb:GetShardIterator",
        "dynamodb:Query",
        "dynamodb:GetItem",
        "dynamodb:Scan",
        "dynamodb:ConditionCheckItem",
        "dynamodb:DescribeTable"
      ],
      "Resource": "[arn_of_lks-jabar-2024-connection_table]"
    },
    {
      "Sid": "Statement2",
      "Effect": "Allow",
      "Action": "execute-api:ManageConnections",
      "Resource":
"arn:aws:execute-api:[region]:[account-id]:*/*/POST/@connections/*"
    }
  ]
}
```

- Save policy with name `api-ws-sendmessage-handler-db-policy`

5.2.4 Default Handler

1. Create Lambda Function with following configuration
   - Option : `Author from scratch`
   - Function name : `api-ws-default-handler`
   - Runtime : `Node.js 16.x`
   - Architecture : `x86_64`
2. Upload `default-handler.zip` as code for created Lambda Function.
3. Add **Tag** :
   - Key : LKS-CC-2024
   - Value : lks-chat-default-handler
4. Open execution role and do the following :
   - Add new policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Action": "execute-api:ManageConnections",
      "Resource":
"arn:aws:execute-api:[region]:[account-id]:*/*/POST/@connections/*"
    },
    {
      "Sid": "Statement2",
      "Effect": "Allow",
      "Action": "execute-api:ManageConnections",
      "Resource":
"arn:aws:execute-api:[region]:[account-id]:*/*/GET/@connections/*"
    }
  ]
}
```

- Save policy with name `api-ws-default-handler-db-policy`

5.3 Create WebSocket API

1. Open API Gateway Console.
2. Create **WebSocket API** :
   - **API name** : `ws-chat-app`.
   - **Route selection expression** : `request.body.action`.
3. **Predefined routes** :
   - **$connect**
   - **$disconnect**
   - **$default**.
4. **Custom routes**
   - **Route key** : `sendmessage`
5. **Attach integrations** settings :
   - Integration for **$connect**
     - Integration type : `Lambda`
     - AWS Region : your region
     - Lambda function : **Connect Handler** lambda you created previously
   - Integration for **$disconnect**
     - Integration type : `Lambda`
     - AWS Region : your region
     - Lambda function : **Disconnect Handler** lambda you created previously
   - Integration for **$default**
     - Integration type : `Lambda`
     - AWS Region : your region
     - Lambda function : **Default Handler** lambda you created previously
   - Integration for **sendmessage**
     - Integration type : `Lambda`
     - AWS Region : your region
     - Lambda function : **Sendmessage Handler** lambda you created previously
6. **Stages** setting :

    - Stage name : `production`
7. **Tags** :
    - Key : LKS-CC-2024
    - Value : lks-chat-websocket-api

5.4 Create S3 Website Hosting

5.4.1 Create S3 Bucket

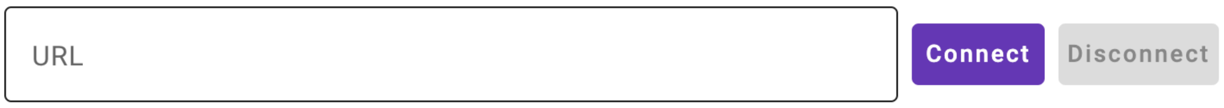
1. **Bucket name** : `ws-chat-web-[account_id]`
2. **Object Ownership**:
    - ACLs enabled.
    - Bucket owner preferred
3. **Block *all* public access** : disabled
4. **Tag** :
    - Key : LKS-CC-2024
    - Value : lks-chat-ws-chat-bucket
5. **Static website hosting** settings :
    - **Static website hosting** : enable
    - **Hosting type** : Host a static website
    - **Index document** : `index.html`

5.4.3 Uploading website to S3 bucket

1. Download `web.zip` and extract it on your computer.
2. Upload extracted files to your bucket with following settings :
    - **Permission / Access control list (ACL)** : Grant public-read access

5.4.4 Testing your website

1. Open S3 bucket you have created previously.
2. Choose **Properties** tab.
3. Under **Static website hosting** copy your website URL.
4. Open URL in browser.
5. You should see something like th



5.5 Create distribution

5.5.1 Create Cloudfront distribution

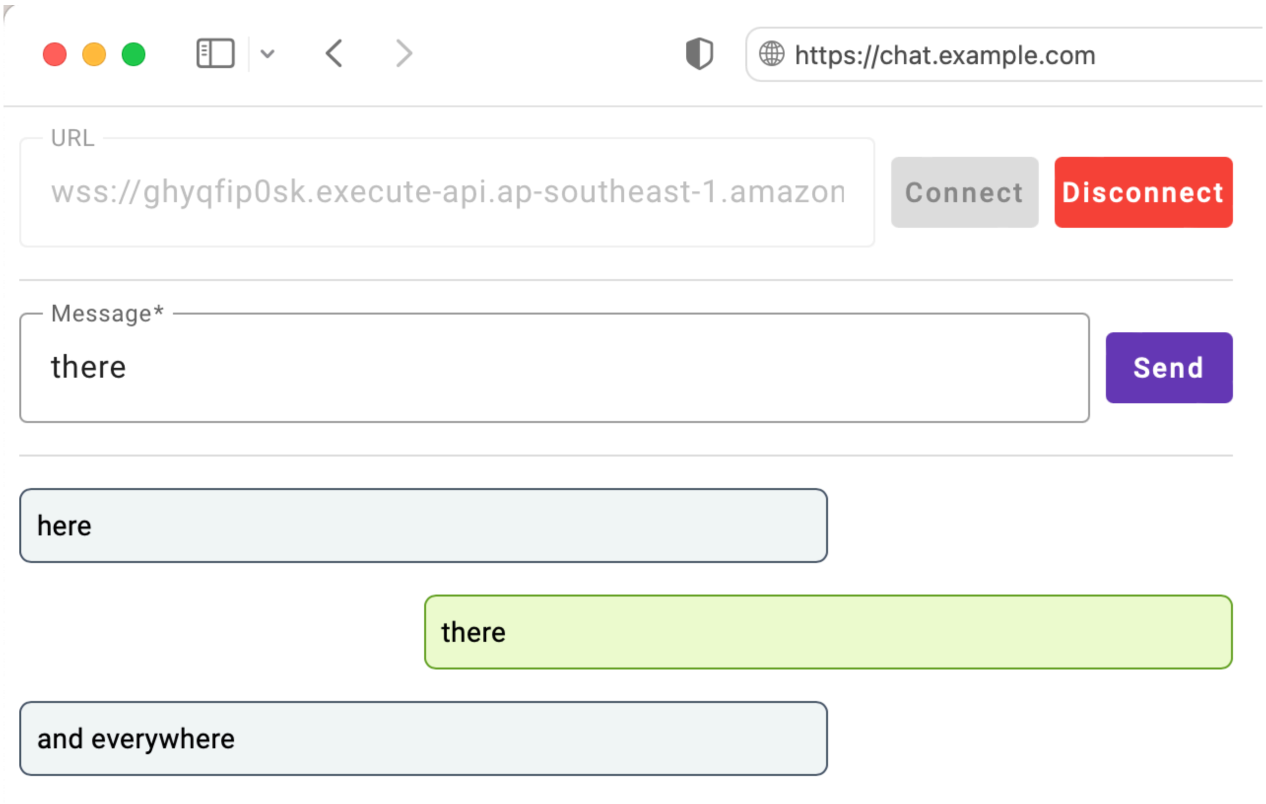
1. Create a new distribution.
2. **Origin** :
    - Domain : the domain part of your S3 Website URL.
    - **HTTP Port** : `80`
    - **Name** : enter `ws-chat-distribution`
3. **Default cache behavior** :
    - Viewer protocol policy : **Redirect HTTP to HTTPS**
4. **Web Application Firewall** : Do not enable protections.
5. **Settings**
    - **Alternate domain name** `chat.<your_domain>`
6. Tag :
    - Key : LKS-CC-2024
    - Value : lks-chat-ws-chat-distribution

5.5.2 Create DNS Record

Create a record that matches with your **Alternate domain name** of your **Cloudfront distribution**.

5.6 Chat using your hosted website

1. Copy the alternate domain of your Cloudfront distribution and use it as your website URL.
2. Open your website URL in two or more browser tabs/windows.
3. Try to communicate between those tabs/windows :
    - For **URL**, enter your **WebSocket URL**.
    - Click **Connect**.
    - Enter your message and click Send


https://chat.example.com

URL

wss://ghyqfip0sk.execute-api.ap-southeast-1.amazon

Message*

there

here

there

and everywhere

6. Reference

- [DynamoDB]
- [Lambda]
- [API Gateway]
- [Simple Storage Service (S3)]
- [CloudFront]
- [Route 53]