Create an IAM User

Navigate to IAM in AWS Console:

- Log in to the AWS Management Console.
- Search for and select IAM.

Create a New User:

- Click on "Users" in the left sidebar.
- Click "Add user".

Set User Details:

- Set up a username and click Next.
- Permission options: Select "Attach policies directly".
- Permissions policies: Select “AdministratorAcces”.
- Click Next and Create user.

Set Acces key:

- Go to IAM in AWS Console.
- Click on "Users" in the left sidebar.
- Click on your user.
- Click on “Create access key”.
- Use Case: Select “Command Line Interface (CLI)”.
- Check on the confirmation box.
- Click Next and Create access key.
- **IMPORTANT**: Save both the Acces key and Secret acces key.

Create a Public RDS PostgreSQL Instance

Navigate to RDS in AWS Console:

- Click "Create database".
- Select "Standard Create".
- Engine type: Select PostgreSQL.
- Select "Free tier" for development/testing.
- Set a DB instance identifier.
- Credentials management: Select ”Self managed”.
- Set a master username and password .
- Virtual Private Cloud (VPC): Select the default VPC.
- VPC security groups: Create a new VPC security group and set the name.

Additional Configuration:

- Set the initial database name.
- Click "Create database" and wait for the database status to become.

VPC security group configuration:

- Navigate to RDS in AWS Console:
- Click on "Databases" in the left sidebar.
- Click on your Database.
- Go to the “Connectivity & security” section of your database.
- Click on your VPC security group.
- Click on your Security group ID.
- Go to the “Inbound rules” section of your security group.
- Click on “Edit inbound rules”.
- Click on “Add rule”:
    - Type: PostgreSQL
    - Protocol: TCP
    - Port Range: 5432
    - Source: 0.0.0.0/0
- Click on “Save rules”.

Configure AWS CLI:

- Open your terminal or command prompt anywhere:
- Run:
    - aws configure
- Enter the following when prompted:
    - AWS Access Key ID: From the IAM user you created.
    - AWS Secret Access Key: From the IAM user.
    - Default region name: us-east-1.
    - Default output format: json.

Install kubectl:

Download and install kubectl:

- Visit https://kubernetes.io/docs/tasks/tools/ and follow instructions for your operating system.

Install eksctl

Download and install eksctl:

- Visit https://eksctl.io/installation/#for-windows_1, go to the “Third-Party Installers” section and follow instructions for your operating system.
- **IMPORTANT:**
    - For Windows, you will need chocolatey. You can visit https://chocolatey.org/install and follow the instructions to install it.
    - For MacOS, you will need Homebrew. You can visit https://brew.sh and follow the instructions to install it.

Create and Configure an EKS Cluster

Create EKS Cluster Using eksctl:

- Open Terminal or Command Prompt anywhere.

- Run the following command to create a cluster named library-cluster with a node group of 2 t3.medium instances:

  eksctl create cluster --name library-cluster --region us-east-1 --nodegroup-name standard-workers --node-type t3.medium --nodes 2 --nodes-min 2 --nodes-max 4 --managed

Update kubeconfig:

- Run:
  - aws eks --region us-east-1 update-kubeconfig --name library-cluster

Deploy Your Application to EKS

First, go to the deployment.yaml file and change the DATABASE_URL value to yours:

postgresql://<username>:<password>@<RDS Database endpoint>:5432/<database_name>

You can find the RDS Database endpoint here:

- Navigate to RDS in AWS Console:
- Click on "Databases" in the left sidebar.
- Click on your Database.
- Copy the value in “Endpoint & port”

”

Navigate to the application directory in your command line and deploy the manifests:

- Run:
  - kubectl apply -f kubernetes/deployment.yaml
- Runt:
  - kubectl apply -f kubernetes/service.yaml

Deploy the Frontend to S3

First, we need to change the endpoint on the frontend:

- Run:
  - kubectl get services
- Copy the “EXTERNAL-IP” value of library-backend-service.
- Navigate to the App.js file on /frontend/src.
- Set “API_BASE_URL” to the endpoint we just copied.
- Remember to add “http://” at the beginning.

Navigate to S3 on the AWS Console.

- Click on the “Create bucket” button.
- Set a name for the bucket.
- Uncheck the “Block all public access” checkbox.

Edit bucket properties:

- Navigate to S3 on the AWS Console.
- Click on your bucket.

- Click on the “Properties” section.
- On “Static website Hosting” click on the “Edit” Button.
- Static website hosting: Select “Enable”.
- Index document: “index.html”.

Edit bucket permissions:

- Navigate to S3 on the AWS Console.
- Click on your bucket
- Click on the “Permissions” section.
- Bucket Policy: Click on the “Edit” Button.
- Paste the following:

```
        {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "PublicReadGetObject",
        "Effect": "Allow",
        "Principal": "*",
        "Action": "s3:GetObject",
        "Resource": "<REPLACE WITH YOUR BUCKET ARN>/*"
      }
    ]
}
```

- Note: You can find your Bucket ARN just above the Policy textbox.
- Click on “Save changes”.

Create build file::

- Navigate to the frontend folde on your command line.
- Run:
    - npm install
- Run the following command:
    - On Unix-based systems (macOS/Linux)
      export NODE_OPTIONS=--openssl-legacy-provider
      npm run build
    - On Windows (Command Prompt)
      set NODE_OPTIONS=--openssl-legacy-provider
      npm run build
    - On Windows (PowerShell)
      $env:NODE_OPTIONS="--openssl-legacy-provider"
      npm run build

Upload files:

- Navigate to S3 on the AWS Console.
- Click on your bucket.
- Click on the upload button
- Click on add files.
- Select the files created on the build folder (asset-manifest.json and index.html)r.
- Click on add folder.
- Select the folder created on the build folder (static).
- Click on upload.

Acces the app:

- Navigate to S3 on the AWS Console.
- Click on your bucket.
- Select the “Properties” section.
- Static website endpoint: Click on the Bucket website endpoint.

NOTE: FOR ANY CONFIGURATIONS NOT MENTIONED ON THE GUIDE USE THE DEFAULT VALUES.