MODUL 2

LKS PROVINSI JAWA TIMUR

CLOUD COMPUTING

2022

Description of project and tasks

This module is six hours - Day 2: Private and Public API.

The goal of this Project is to deploy and scale an API server services and exam service web application to support client services for Admission Apps. Today you will deploy a highly available, scalable, and efficient web application using the nodejs server provided. You also should be build required initial support services.

This nodejs application has service dependencies as outlined in the Technical Details section below. This application responds well to scale as horizontal scaling. This application will not work "out of the box". Instead, you will have six hours to get an infrastructure rolled out and application deployment. Throughout the day it is your responsibility to respond to any challenges that may present themselves. Once the day has concluded, additional load testing will be conducted to determine your architecture's resiliency and ability to self-repair.

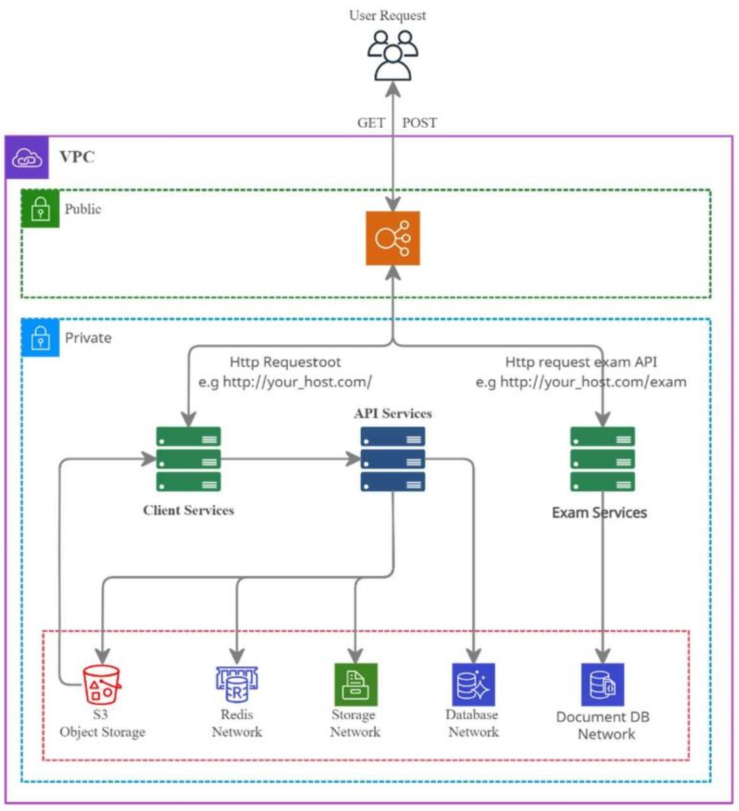
Task

1. Share you AWS pro account on this link https://s.id/akunlksprovjatim2022.
2. Read the documentation thoroughly (Outlined below).
3. Please view and understanding the application architecture in the architecture section.
4. Please carefully read the technical details section.
5. Please carefully read the application details.
6. Log into the AWS console.
7. Create your key pair and uploaded to this link https://s.id/keyday2 . Use this key pair to each instance you created.
8. Re-install client service. You can read detail instruction in Application Detail section.
9. Setup your VPC configurations. You can read VPC configuration details in Networking - Service Details section.
10. Setup a security group. You can read security additional rules in Security – Service Details section.
11. Setup document db. More details about it in Details Service section.
12. Deploy the API and exam services. More details instruction in Application section. (Please read technical details for more information's).
13. Setup ELB for exam and API services. Please Read Technical details for more information's.
14. Re-setup Public ELB to provide URL Routing.
15. Configure the client, API, and exam services to handle increasing load (See technical detail section, what auto scale config to use for client, API, and exam services).
    (a) Create and/or update the user data configuration correctly, including download locations of the javascript code and server configuration file.
    (b) Create and/or update the user data configuration to include any required local dependencies.
16. Verifying all of services are running well and the admission apps can work well.
17. Configure any server dependencies as outlined in the technical details.
18. Configure necessary application monitoring and metrics in CloudWatch.

Technical Details

1. The admission apps is deployed as a javascript application. Do not alter the source code both of client and API service in any way as that will be grounds for disqualification.
2. The admission apps have three services that are must be deploy to the different server setup and you should deploy two (API and Exam service) of three services today.
3. All of application services such as client, API, and exam must be deploy in t2.small instance type.
4. You should be prepared for 1000 concurrent request to Admission apps. Be careful about overloading and watch for HTTP 503 or 404 responses from the client, API and exam service.
5. You should use existing public load balancer for setting up the URL routing of public exam API. Don’t create a new public load balancer or you will lose the points.
6. API server cannot be accessible from outside as a private API.
7. Exam service can be accessible from outside as a public API.
8. The committee may send you a link to get your public load balancer address.
9. An additional load testing may be sent in the middle of the competition to your public load balancer address, and you may have to make performance improvements at that time. Better performance and less errors will make you earn more point.
10. While you create API service for Auto-Scaling, you are only allowed to use the launch configurations for the API server services. If the configuration is not following the instruction, then you will lose the point.
11. While you create Exam service for Auto-Scaling, you are allowed to use the launch configurations or launch template for the Exam services. If the configuration is not following the instruction, then you will lose the point
12. This API server requires a connection to a Redis database, a MySQL database, and a location to store data cache and log files. You can use the service you have created on the day one.
13. This exam service requires a connection to a mongo database as a document db. More details about document db instruction at service details section.
14. You should be creating an additional security group follows in security Service Details section.
15. Remember to label every service you have created like vpc, security group, ec2 instance and everything else you have created except those created automatically like instance who has creating automatically by auto scaling. The more attentions to the little things, more point you will earn.
16. Remember to fill each description and tag of the services to get more points.
17. The base OS that has been chosen is Amazon Linux (https://aws.amazon.com/amazon-linux- ami/). This distribution was selected for its broad industry support, stability, availability of support and excellent integration with AWS.

Architecture



The above example illustrates one possible architectural design for the deployment of the application. This shows all required segments needed to server requests. The Url Routing "/" to access client services and url path "/exam" to access exam services, all of them path can be accessible from outside (public).

Application Details

The Admission Apps was developed with javascript with three services running in it. Those services are Client services, API Server services and the additional service is Exam services, those services have their respective functions. The Client service is used for the front end which is to provide views and interaction between applications and users. The API Server is used for back-end purpose which served to provide data to user through client service. The last one is Exam service is use for public exam API. Installation instructions of those service are below.

Pre-Requirement

All services were developed with javascript and running with nodejs for the runtime. Firstly, you need nodejs and NPM package installed to 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 follows below.

First of all, you need to enable 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 -sL https://rpm.nodesource.com/setup_14.x | sudo -E bash -
```

NPM will also be installed with node.js. If you have done with the repository, next you can install the nodejs packages. The NPM was already installed when node js was installed.

You should clone a Admission Apps source code from git repository. Make sure you have git package installed in your system. The git repo URL is https://github.com/betuah/lks-apps-one.git.

Once complete, there are three folders on it with some file in the root folder, one of some files is README.md file. You can carefully read the configuration instruction and environment key and value of each service in README.md file. There has three folders on it, the client, server and exam folders. Each folder represents an application service.

Note: make sure your instance is connected to the internet for running all these instructions.

Client Services

You may have deployed this service yesterday but there has been a slight change in the source code to support the exam service. you have to re-clone or pull service to get the latest updates. There is an additional environment to be able to connect with the Exam public API. detailed instructions are below:

1. You should install all required dependencies with npm or yarn command in the client folder.
2. You should pull request to git repository to get the latest updates.
3. Create new version of client launch template.
4. You should create the .env file in server folder like below.

   ```
   API_URL=YOUR_API_BACKEND_SERVER_HOST
   API_EXAM_URL=YOUR_API_BACKEND_SERVER_HOST
   ```

   More detail you can follows the client config Setup instruction in README.md file.

5. Use 2nd option deployment in README.md file to re-generate static source code. Remember don’t use first option deployment or you will lose the points. This Gives us the ability to host our web application on any static hosting.
6. The static source code will be generated in dist folder.
7. This client service must be running on web server, please install httpd as web server.
8. If all done you can access the client apps with http://YOUR_DOMAIN in web browser. The functionality of client service may not run properly. You should deploy the API and Exam service to work with this client service.

Note: If you have Auto-Scaling Group for this services, use launch template while build it.

API Server Services

The API server services was developed with express js, which is a node js framework. This API server services require all of dependencies in server/package.json file in server folder of Admission Apps.

This service requires a connection to redis as cache, s3 to store assets data, serverless database and storage to store log and file cache. You can't start deploying this service if those all services are not available or not running well. More details for installation of those service at the Service Details section. You can follow these instructions below for the API server deployment.

1. You should install all required dependencies with npm or yarn command in the client folder.
2. You should create the .env file in server folder like below.

```
NODE_ENV=production
PORT=8000
DB_TYPE=YOUR_DATABASE_TYPE
MYSQL_DB=YOUR_MYSQL_DATABASE_NAME
MYSQL_USERNAME=YOUR_MYSQL_USERNAME
MYSQL_PASSWORD=YOUR_MYSQL_PASSWORD
MYSQL_HOST=YOUR_MYSQL_HOST
MYSQL_PORT=YOUR_MYSQL_PORT
REDIS_HOST=YOUR_REDIS_HOST
REDIS_PORT=YOUR_REDIS_PORT
REDIS_PASSWORD=YOUR_REDIS_PASSWORD
AWS_ACCESS_KEY=YOUR_AWS_ACCESS_KEY
AWS_SECRET_ACCESS_KEY=YOUR_AWS_SECRET_ACCESS
AWS_BUCKET_NAME=YOUR_AWS_BUCKET_NAME
LOG_PATH=YOUR_LOG_FOLDER_LOCATION
CACHE_PATH=YOUR_CACHE_PATH_FILE_LOCATION_STORE
```

More detail you can follows the Server config Setup instruction in README.md file. Run "npm run start-prod" to start API server with PM2. You can use "npm run stop-prod" to stop it if is needed.

3. Then you can access the API endpoint at http://YOUR_API _HOST:8000/check.

Note: If you have Auto-Scaling Group for this services, use launch configurations while build it.

Exam API Services

Same as API server the exam services was developed with express js. This exam service also require all of dependencies in exam/package.json file in exam folder of Admission Apps. This service needs to connect with mongodb as a document database. You can use document db service on aws instead of mongodb. You can't run this service if it fails to connect to document db. More details about document db in Service details section. For deploying exam service, you can follows these instruction below.

1. You should install all required dependencies with npm or yarn command in the client folder.
2. You should create document db and download the ssl into exam/ssl folder and set the cert name in environment file.
3. You should create the .env file in exam folder like below.

```
NODE_ENV=production
PORT=9000
DB_TYPE=mongodb_aws
MONGO_DB=YOUR_MONGO_DATABASE
```

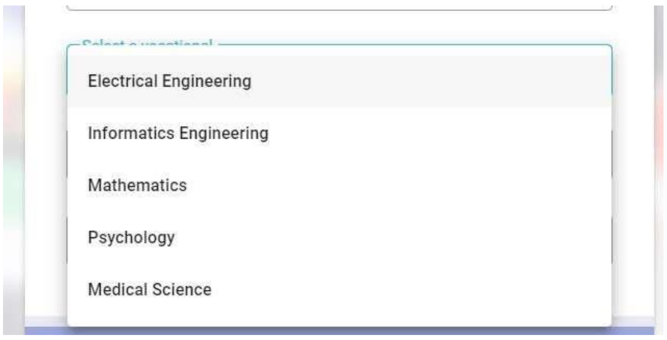
```
MONGO_USERNAME=YOUR_MONGO_USER
MONGO_PASSWORD=YOUR_MONGO_PASSWORD
MONGO_HOST=YOUR_MONGO_HOST
MONGO_PORT=YOUR_MONGO_PORT
MONGO_CERT=YOUR_MONGO_CERT
LOG_PATH=YOUR_LOG_FOLDER_LOCATION
```

Note: Don't change DB_TYPE variable. More detail you can follows the Exam config Setup instruction in README.md file.

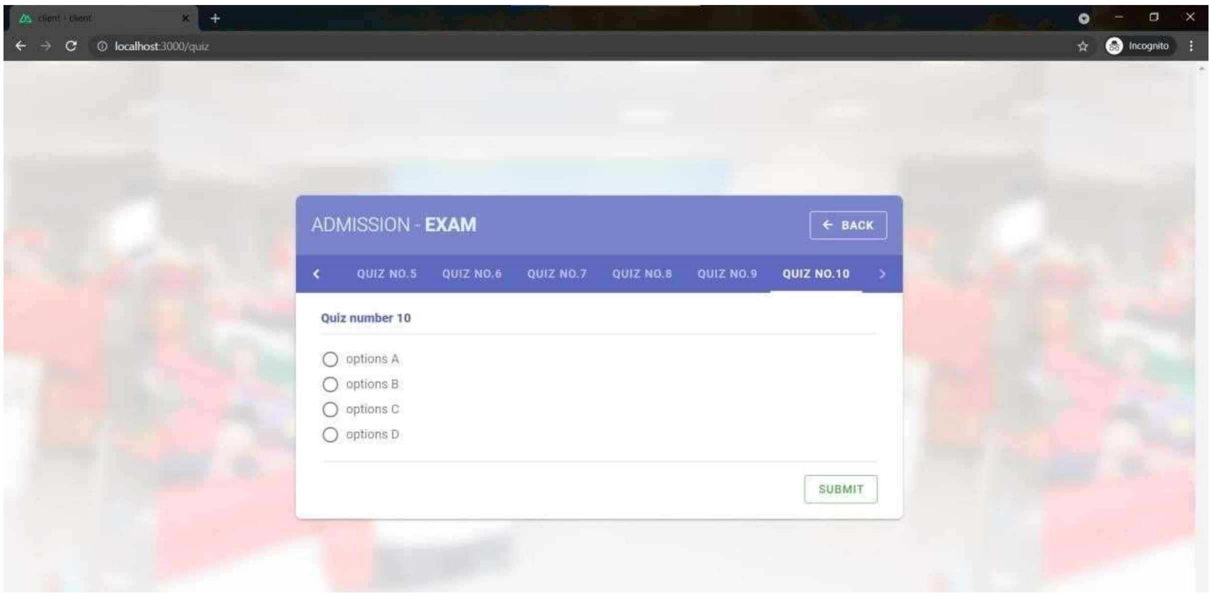
4. Run "npm run start-prod" to start API server with PM2. You can use "npm run stop-prod" to stop it if is needed.
5. Then you can access the Exam API endpoint using Postman software at http://YOUR_API _HOST:9000/.
6. You can create initial dummy data with Exam API Endpoint "GET http://YOUR_HOST:9000/exam/init". This endpoint is used to create dummy data. You can flush the dummy data with this Exam API endpoint "GET http://YOUR_HOST:9000/exam/flush". You can use "GET http://YOUR_HOST:9000/exam/quiz" to retrieve all dummy data.

Application Testing

You can make sure the application that you have deployed can run properly. you can try to register at root URL in registration menu. If the connection to your database is successful, the data will appear in the student data menu and vocational drop down will show the data on register form.

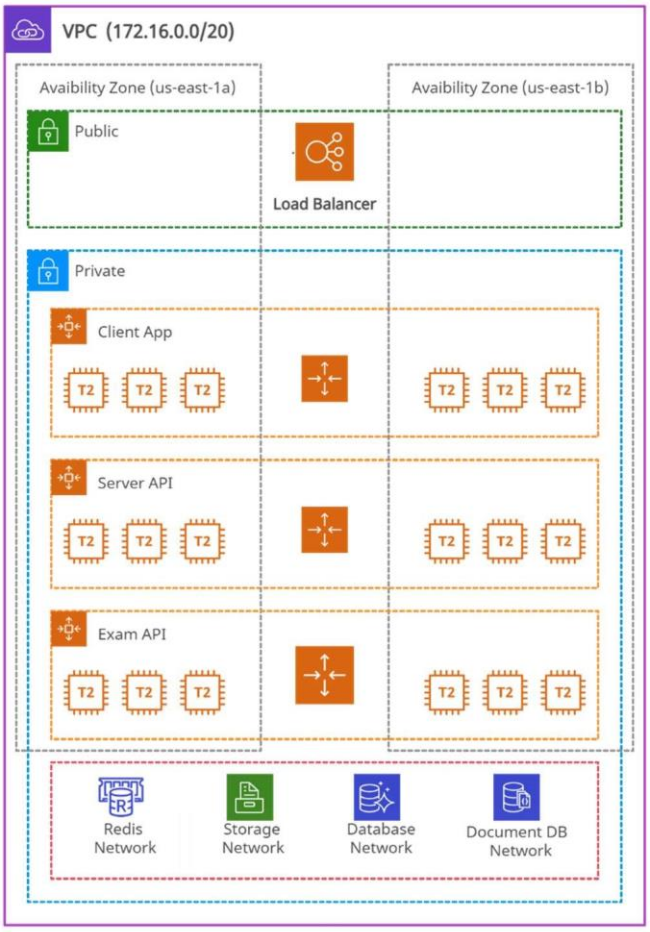


To find out if the client service is connected to the general exam API, you can check on the exam menu, if the dummy question data appears, it means it has been successful, but if the data is not found, it is possible that the init data is not running or your exam service is not running properly.



Service Details

Networking (VPC)



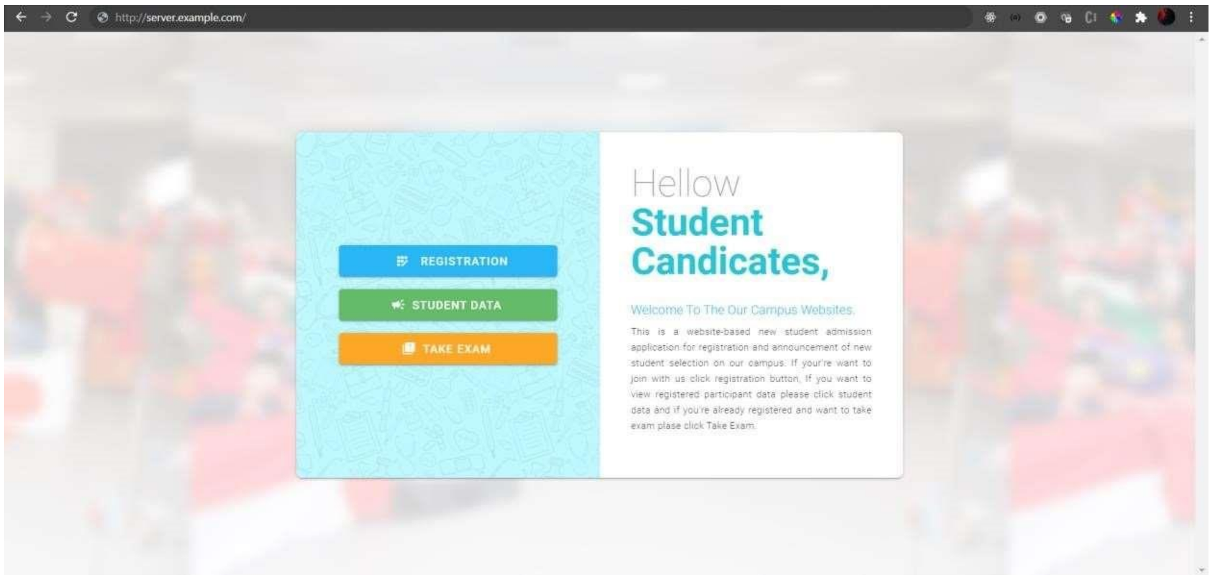
You have built a network infrastructure before and you will also use that infrastructure for today. However, there will be additional service today, so you have to create a few more subnets. As shown in the following topology, then you have to create a new subnet to provide exam service network. You should create subnet with 50 IP portion for each AZ from block IP 172.10.2.128/25 and should create subnet with 50 IP portion from block IP 172.10.0.0/23.

Security

Like a networking section above, maybe you have also created a few securities group necessary. Security is an important thing that should not be pass when you are building an infrastructure. Today there is an additional service that must be deploy so you must create some additional security group to allow exam port just can be accessible from public subnet and client subnet. You also should create security group allow document db port can be accessible from exam service, the default port of document db is 27017. You can use the security group that you have created before according to your current service needs.

URL Routing

You will provide a single URL that the application will use to access both of your applications. In order to facilitate this, you will need to perform URL routing. The application identified as "root" should run on the root of your URL. Thus, if deployed properly, when you visit the public URL of your application, you should see the screen below:

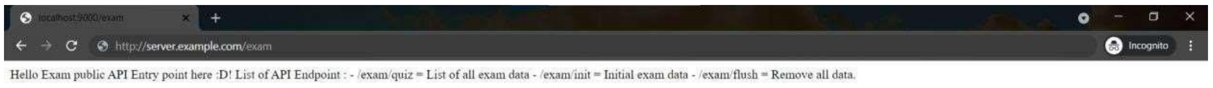


Hellow
Student
Candicates,

Welcome To The Our Campus Websites.

This is a website-based new student admission application for registration and announcement of new student selection on our campus. If your're want to join with us click registration button. If you want to view registered participant data please click student data and if you're already registered and want to take exam plase click Take Exam.

You will need to deploy the second application named, "exam" to the URL of "/exam" off of your main URL. If your main URL is http://server.example.com, then hitting that main site in your browser should return the image above. Going to http://server.example.com/exam should however show the page below:



```
Hello Exam public API Entry point here :D! List of API Endpoint : - /exam/quiz = List of all exam data - /exam/init = Initial exam data - /exam/flush = Remove all data.
```

You can accomplish this by also deploying the application named "exam" to the "/exam" suffix to your URL. The URL that you enter will receive requests on the root of your URL as well as the "/exam" endpoint.

Document DB

The Exam service uses Mongoose as an ORM library that connects the exam service to database services with No SQL language. Mongoose uses the document base database, so you need to deploy a document db cluster. You need to create a document db cluster with instance class db.r5.large with just one instance and 27017 as a ducment db port. Don't forget to setup user and password to be access by exam service.

Once complete, you will need to set database environment to exam services via ".env" files deployed to each instance.

```
DB_TYPE=mongodb_aws
MONGO_DB=YOUR_MONGO_DATABASE
MONGO_USERNAME=YOUR_MONGO_USER
MONGO_PASSWORD=YOUR_MONGO_PASSWORD
MONGO_HOST=YOUR_MONGO_HOST
MONGO_PORT=YOUR_MONGO_PORT
MONGO_CERT=YOUR_MONGO_CERT_NAME
```

Note: The db_type env variable must be mongodb_aws for use document db on aws cloud.

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 examples, you could look to create a centralized file storage solution with Elastic File Storage to store log and cache file from the application. You can use a local directory to save the log files but a shared storage solution or the ability to share files to each instance will allow you to have centralize log file which is it will be made maintenance process easier. File Cache need central file storage solution for consistency data. The relevant environment variable in the ".env" configuration file is below:

```
CACHE_PATH= YOUR_CACHE_PATH_FILE_LOCATION_STORE

LOG_PATH=YOUR_LOG_FOLDER_LOCATION
```

CACHE_PATH is the local directory for file cache on the instance where the server application is able to both read and write to the cache files. This can be a local directory on the server or a remote directory mounted as a local directory on the server. The default path storage if you not set cache environment path will store in <your_apps_path>/server/tmp.

LOG_PATH is the local directory for store applications log on the instance where the server application is able to both read and write to the cache files. This can be a local directory on the server or a remote directory mounted as a local directory on the server. The default path log storage if you not set cache environment path will store in <your_apps_path>/server/logs.