National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Anilgovind Kokkoori Anilkumar
Student ID: x23176458

School of Computing
National College of Ireland

Supervisor: Dr Giovani Estrada

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anilgovind Kokkoori Anilkumar
Student ID:	x23176458
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Dr Giovani Estrada
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	2243
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	28th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Anilgovind Kokkoori Anilkumar
x23176458

1 Setting Up The AWS USER

The first step is to create a AWS user getting the Access key and secret key for accessing the cloud from local. The following steps has to be followed.

- Login to AWS console using root account.
- Navigate to IAM Dashboard and select users.
- Create a new user, input a username and click on "Provide user access to the AWS Management Console".
- Click on "I want to create an IAM user" in the popup and give a temporary password for your account.
- In the next tab click on "Attach policies directly" and select "AdministratorAccess" from the below listed policies.
- Click on next and then create user. Then store the credential details for sign in. Now the user will be listed in AWS console at IAM user tab.
- Click on the user and click on create access key. Now please select Local Code option from the pop up window.
- Click next after clicking the confirmation box.
- Select a tag for the credential that we are creating.
- Then click on create access key. Download the csv and click on Done. Keep the csv safe for future.

Use the created user credentials for login to aws account and it will prompt to change password for the first time. Create a strong password to access AWS console. Always keep the region to be Ireland(eu-west-2) for the following steps.

2 Setting Up The VPC, Subnets and routing tables

- Navigate to AWS VPC console. Then click on cretae VPC.
- Select VPC and more option this way we can create VPC subnets and routing tables.

- Keep the default configurations. select the NAT gateways 1 per Availability zone.
- Then click on create VPC
- This step will create VPC and routing tables with 2 private subnets one in each AZ, two public subnets one in each AZ. and 4 routing tables.

The created VPC has to be used for redis and redis accessing cloud resources.

3 Creating A Security Group

Security Group to allow necessary traffic from redis and its accessing resource is necessary. Security group can be created via the following steps.

- Navigate to VPC.
- Under security panel on the left side select Security groups as a initial step.
- Then click on create security group for creating a security group.
- Then select the new VPC that we are created in section 1 after filling name and description.
- Now create in inbound rule as in table 1.
- Then click on create security group.

Name	**Protocol**	**Port**	**Source**	**IP Version**
Custom TCP	TCP	6379	0.0.0.0/0	IPv4
SSH	TCP	22	0.0.0.0/0	IPv4
HTTP	TCP	80	0.0.0.0/0	IPv4

Table 1: Network Configuration Table

4 Creating DynamoDB

The DynamoDB for CRUD functions can be created by following the below steps.

- Navigate to AWS DynamoDB console.
- Click on tables. Then click on create table.
- Give table name. This project named the table as "RIC-EMPLOYEEE-TABLE". this is for dummy employee details for CRUD functions.
- Give the employeeId (String) as partition key and ppsNumber (String) as sort key.
- Keep default configuration and create table.
- Now the table will create in few minutes. This table has to be used by the CRUD functions.

5 Creating Elastic Cache - Redis

Elastic cache can be created by the following steps.

- Navigate to AWS ElastiCache.
- Click on Dashboard and then click on Redis OSS caches in Resource overview tab.
- Select Create Redis OSS cache. Then click on "Design your own cache" and make sure the selection is on "Easy create" then select Demo configuration. This configuration will have cache.t4g.micro, 0.5 GiB memory, Up to 5 Gigabit network performance.
- In the Cluster info section give a name for the cluster we are creating.In this project we gave "ric-rediscluster" as cluster info name.
- In the Connectivity section select "Create a new subnet group"
- After giving name and description for the subnet. Choose VPC created in section 2.
- Mange the subnet groups to have only 2 private subnets from the each of the availability zone.
- Now click on create and the redis cache will be created with engine version 7.1.0.
- Now we need to attach the security group that we created on section 3 to this redis.
- After the redis creation is completed. select the redis and navigate to Network and security section of it. from there check the Security groups. make sure to add the creted security group in section 3.
- Select the redis and navigate to bottom. then click on symbol of redis displayed under section "Shards and nodes". The details will have endpoint details which need to be used for establishing connection with redis.

6 Creating and configuring EC2

6.1 Creating EC2

The light weight load balancer will be running on a ec2 machine. We need to create this EC2 machine. It can be done through the following steps.

- Navigate to AWS Ec2 console.
- Click on launch instance.
- In the Launch an instance configuration page select the Ubuntu image for creating the cloud machine.
- Select t2.micro instance.
- Then create a key pair for remote access. If have one select it.

- In the Network settings select the VPC created in section 2 and security group created in section 3.
- Keep other details as default and create instance.

Next step is to assign Elastic Ip to the newly created Instance.

- The instance has to have a elastic-ip. for this select to Network & Security in the left side of the window and select the elastic IP.
- Select on "Allocate Elastic IP Address" then click on allocate. This will create a new elastic ip.
- To allocate the newly created ip to the ec2 instance we created, click on the ip from ip listing dash board and click on Action.
- From the drop down select associate elastic ip address.
- Select the newly created ec2 instance for associating this ip and proceed. The instance will be associated with this ip address.

6.2 connecting to EC2

Next we need to establish a connection from the VS code to this machine.

- First get the the connection details to the ec2. this will be available when you select the connect option after selecting the instance from dashboard.
- Get the ssh connection details.
- Execute chmod 400 "secret access file.pem" on the file if its newly created when we create the ec2.
- Follow the steps in "Connecting VSCode to your EC2 Instance" of (Cheng; 2023). This will establish connection to the ec2 instance from local machine via VScode.

6.3 Setting up the project in EC2

Next step is to setup the project in EC2. After establishing the connection to the cloud machine. follow the below steps.

- Make sure the Ubuntu version is Ubuntu 24.04.1 LTS. use hostnamectl command to make sure the machine is similar to the following one that we used.
  Static hostname: ip-10-0-7-166
  Icon name: computer-vm
  Chassis: vm
  Machine ID: xxxxxxxxxx
  Boot ID: xxxxxxxxxx
  Virtualization: xen
  Operating System: Ubuntu 24.04.1 LTS
  Kernel: Linux 6.8.0-1018-aws
  Architecture: x86-64

Hardware Vendor: Xen
Hardware Model: HVM domU
Firmware Version: 4.11.amazon
Firmware Date: Thu 2006-08-24
Firmware Age: 18y 3month 2w 2d

- Execute sudo apt update to update all the update the list of available packages in local.
- Next upgrade all the installed packages to its latest by sudo apt upgrade -y
- Now remove unwanted packages and all by "sudo apt autoremove -y" and "sudo apt autoclean" commands.
- Install curl for downloading the node package

- Execute the following step to install node. detailed steps are in (freecodecamp; 2023).
- Test installation by the following commands.
  node –version
  npm –version
  For this project we had node version of v18.19.1 and npm version of 9.2.0.
- Normally the image have git installed if not install git.
- Pull the load balancer code from git by the following command
  git clone https://github.com/anilgovind-nci/RIC-LOAD-BALANCER.git
- Navigate to RIC-LOAD-BALANCER directory then install the packages to run the project by
  "npm install"
- Next step is to download AWS CLI and configure AWS CLI. for that follow the below commands. detailed steps are in (services; 2024).
- Check the installed awscli version by
  aws –version
- Next configure the awscli to access other cloud resources from the ec2 machine.run the following command.
  aws configure
- Fill the details with the credentials that we created in section 1.
  AWS Access Key ID [None]: xxxxxxxx
  AWS Secret Access Key [None]: xxxxxxxxx
  Default region name [None]: eu-west-2
  Default output format [None]: json

- Now we need to set the redis with initial data. for that navigate to setup folder then run the command// node setRedis.js// This will create initial dummy data in redis.
- Now the machine is completely ready for running the load balancer. Run the following command to run the code.
  npm run app
- The endpoints can now access from the local machine through REST methodes. The url is given below.
  http://localhost:3000/ric

7 Setting Up Secret Manager

Secret manager has to be setup for storing the credentials.

- Navigate to aws secret manager then click on store a new secret.
- Select "Other type of secret" from Choose secret type section.
- Now configure the key value pair with the following details.

  RedisHost: The endpoint we created for redis in section 5.
  RedisPort: 6379
  getFunctionResourcesRediskey: getLambdaDetailskey
  postFunctionResourcesRediskey: postLambdaDetailskey
  putFunctionResourcesRediskey: putFunctionResourcesRediskey
  deleteFunctionResourcesRediskey: deleteLambdaDetailskey
- Then click next and give name "ric-credentials" for the secrets. keep everything default and store the secret.

8 Creating Lambda Layer

The common packages that the project using is created as a layer and uplod into lambda layer. The received arn is using for configuring the lambda functions in section 9.

- The code for lambda layer can be downloaded from git by the following command.
  git clone https://github.com/anilgovind-nci/Lambda-Layer.git
- Now zip the downloaded folder and rename it to nodejs.
- Navigate to AWS lambda and select layers. Then click on create layer.
- Give name and description then upload the zip file cretaed earlier.
- Now click on create to create the layer.
- Once the layer is created get its ARN. This one we need for configuring serverless file.
- The layer ARN can be seen when we click the created layer.

9 Setting up Serverless And Uploading CRUD Functions

All the configurations of CRUD functions and other test function with its events are configured via a serverless file.

There is four different serverless files for deployment.

1. One for the Redis Updated and Lambda warming
2. One for CRUD function deployment
3. One for SNS lambda warming architecture
4. One for Event Bridge warming architecture

9.1 Setting up Serverless

- Make sure local machine have node version 18 and above. If install node version 18 or above (Node; 2022)
- The serverless can be installed locally by npm (serverless; 2024) install -g serverless
- Check node version installed.

9.2 Deployment using serverless

- Pull the code from git using
  git clone https://github.com/anilgovind-nci/ric-crud-application.git
- Before going to the next step configure every serverless file with the created resources. following are the areas that might need to edit.
  1. **The lambda layers ARN in serverless file**. It configured below lambdas which need the the layer that we created in section 8 check layer configured for lambda in Figure 1.
  2. **Region** if deploying in other than Ireland(eu-west-2)
  3. **Configure the redisTimeAdjustmentsForEndpoints** with securityGroupIds, subnetIds under its VPC configuration in serverless. This file can be obtained from redisUpdateFunctions directory. check Figure 2 for reference.
  4. Similarly make sure to configure **secret manager ARN** with yours in the resource section of the above said serverless file.
  5. **Rename org, app** and service if needed.
  6. Configure **event bridge** execution times if need to try with other invocation pattern.
- Install the packages by running command
  npm install
  root, redisUpdateFunctions, keepLambdaWarm/viaEventBridge and keepLambdaWarm/viaSNS has to be updated with the above command.

```yaml
functions:
  ricGet:
    handler: functions/ric-get.handler
    environment:
      CENTRALISED_LOG_GROUP_NAME: 'RIC-CRUD-log-group'
    layers:
      - arn:aws:lambda:eu-west-1:557690584148:layer:dependencies-layer:13
```

Figure 1: lambda layer configuration

```yaml
functions:
  redisTimeAdjustmentsForEndpoints:
    handler: ./redisTimeAdjustmentsForEndpoints.handler
    layers:
      - arn:aws:lambda:eu-west-1:557690584148:layer:dependencies-layer:13
    memorySize: 128
    timeout: 30
    events: []
    vpc:
      securityGroupIds:
        - sg-0307564d9956a71fa
      subnetIds:
        - subnet-04bdf4b9558e72dee
        - subnet-03adaef5d4afc6b48
    role: RedisFunctionRole
```

Figure 2: Configured VPC and subnets via serverless file

- Next run
  serverless deploy The above command has to be run at every folder level above mentioned.
- Running the command will deploy the code and configure the AWS resources automatically.

After the deployment Serverless will give the API gateway endpoints. The same can be obtained from AWS console too.

9.3 Testing of CRUD Functions

The test scripts are in the path test/scripts

- The random time generating functions is names as random_time_generator.js we can run it using command
  node random_time_generator.js
  We can configure number of divisions needed for a specific amount of time.
- The generated array should be kept in file called sporadicTests/sporadicHitTimimgs.json as json object for testing.

```
const jsonFilePath = path.join(__dirname, 'sporadicHitTimimgs.json');
// const apiEndpoint = 'https://vpsllm3pah.execute-api.eu-west-1.amazonaws.com/dev/ric-delete';
const apiEndpoint = 'http://localhost:3000/ric'
```

Figure 3: Configure Endpoint for Destination

```
    if (i < invokeGroups.length - 1) {
        const waitMessage = 'Waiting for 15 minutes before the next group...';
        console.log(waitMessage);
        writeLog(waitMessage);
        await new Promise(resolve => setTimeout(resolve, 900000));
    }
}
```

Figure 4: Configure Sporadic Interval

- Adding multiple key and value will result different waves in sporadic test.
- The test sporadic test files are named getFunctionColdStartIterativeTest.js, post-FunctionColdStartIterativeTest.js, putFunctionColdStartIterativeTest.js and delete-FunctionColdStartIterativeTest.js.
- Run these code by command
  node "filename"
- Check Figure 3 and Figure 4 for reference.
- The sporadic test files can be configured to hit either load balancer via local host or api gateway.
- There is configurable time and endpoint in each of this file.
- Hard code these values according to the need.

9.4 Evaluation of test results

We need two log files for every test evaluation. One from local creted by running the above said sporadic test functions. And another one from cloud watch logs.

- Collect the respective log file from local after running the test.
- run the google collab [1] with this file

  This will give the failed request numbers and total user experienced response time.
- next navigate to cloudwatch.
- click on log groups.
- select the invoked lambda log group.

[1] https://colab.research.google.com/drive/1IKOHrAhAQ6fqGVY8Jezio-1jvvhgQZ4q#scrollTo=eqQ9VAQj5TVO

- click on "search all log streams".
- scroll down to the bottom until every logs are collected by AWS.
- at the very top on actions click download as csv option to download the code in csv format.
- use this downloaded log to run the google collab [2].

- upload the cloudwatch file and complete execution of above given collab will give total billed duration and total memory used.

References

Cheng, B. (2023). Detailed guide to connect ec2 with vscode — medium.
URL: *https://medium.com/@bobbycxy/detailed-guide-to-connect-ec2-with-vscode-2c084c265e36*

freecodecamp (2023). How to install node.js on ubuntu – node linux installation guide.
URL: *https://www.freecodecamp.org/news/how-to-install-node-js-on-ubuntu/*

Node (2022). Node.js — node v18.12.0 (lts).
URL: *https://nodejs.org/en/blog/release/v18.12.0*

serverless (2024). Setting up serverless framework with aws.
URL: *https://www.serverless.com/framework/docs/getting-started*

services, A. (2024). Installing or updating the latest version of the aws cli - aws command line interface.
URL: *https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html*

[2] `https://colab.research.google.com/drive/12qSmyoA2hMMqz0Hyw2vP6Bf3MDaVqcrK#scrollTo=f6LMCeEQGJxS`