

Configuration Manual

MSc Research Project Cloud Computing

Vinay Sriram Iyer

Student ID: X23203595

School of Computing National College of Ireland

Supervisor: Yasantha Samarawickrama

**National College of Ireland Project Submission Sheet School of Computing**

|  |  |
| --- | --- |
| **Student Name:** | Vinay Sriram Iyer |
| **Student ID:** | X23203595 |
| **Programme:** | Cloud Computing |
| **Year:** | 2024 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Yasantha Samarawickrama |
| **Submission Due Date:** | 12/12/2024 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | 4867 |
| **Page Count:** | 19 |

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:** | Vinay Sriram Iyer |
| **Date:** | 12th December 2024 |

**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): |  |

## AI Acknowledgement Supplement [MSc Research Project]

[Configuration Manual]

|  |  |  |
| --- | --- | --- |
| **Your Name/Student Number Course Date** | | |
| **Vinay Sriram Iyer/x23203595** | MSc in Cloud Computing | 12/12/2024 |

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click [here](https://libguides.ncirl.ie/useofaiinteachingandlearning/studentguide).

## AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

|  |  |  |
| --- | --- | --- |
| **Tool Name Brief Description Link to tool** | | |
| **N/A** | N/A | N/A |
|  |  |  |

## Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

|  |  |
| --- | --- |
| **[N/A]** | |
| [N/A] | |
| [N/A] | [N/A] |

## Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

## Additional Evidence:

[N/A]

## Additional Evidence:

[N/A]

Configuration Manual

Vinay Sriram Iyer X23203595

# Preprocessing GWA-T-12 Bitbrains

### Tools and Technologies

* **Operating System - Windows 11 Pro**
* **Processor - Intel(R) Core i5**
* **Physical Memory - 16 GB RAM**
* **Platform - Eclipse IDE 2024-03 (4.31.0)**
* **Language - Python 3.12.2**
* **Package - Pip 24.2, Pandas 2.2.3, numpy 2.1.1, python-dateutil 2.9.0.post(), pytz 2024.2, six 1.16.0, tzdata 2024.1, git-filter-repo 2.45.0**

### Environmental Setup:

1. Download and install Eclipse IDE from the link [here.](https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2024-09/R/eclipse-inst-jre-win64.exe&mirror_id=1285) Download and install Python

3.12.2 from the link [here.](https://www.python.org/ftp/python/3.13.1/python-3.13.1-amd64.exe) Ensure the ”Add Python to Path” box is checked before clicking install. Ensure the full path and Scripts folders are added to the Python installation directory and pip directory by the paths below:

C:\Users\<YourUsername>\AppData\Local\Programs\Python\Python39\ C:\Users\<YourUsername>\AppData\Local\Programs\Python\Python39\Scripts\

1. Go to the official Python website and download the get-pip.py script from the URL below:

https://bootstrap.pypa.io/get-pip.py

Save the file as get-pip.py to your home directory, preferably in your Downloads folder. Navigate to this directory in Command Prompt and run the below command to install pip 24.2:

C:*\*Users*\*<username>*\*AppData*\*Local*\*Programs*\*Python*\*Python312*\*python.exe get-pip.py

1. Verify the installation of Python 3.12.2 and Pip 24.2 by typing the command be- low in Command Prompt:

python --version pip --version

1. Install the Pydev Plugin from Eclipse Marketplace. Once installed, restart Eclipse. Configure Python 3.12.2 and select the location of the Python Interpreter installation. It is recommended to use your home directory on your machine for the Python 3.12.2 In- terpreter installation. Open the terminal on Eclipse IDE and install pandas with numpy by the following commands below in the project directory:

pip install pandas pip install numpy

Close Eclipse IDE. Reboot the machine and launch the Eclipse IDE once again.

### Preprocessing the Dataset:

1. Click on File *>*New *>*Python PyDev Project. Set the Python Interpreter as 3.12.2. In this configuration, the Pydev Project shall be titled ’GWA-T-12BitbrainsPreprocessing’. Ensure the Python 3.12.2 installation points correctly in the interpreter settings.

1. Navigate to File Explorer and locate the PyDev Module script(.py) that needs to be im- ported. In this case, the script shall be titled ’GWA-T-12BitbrainsPreprocessing script.py’. After dropping the script into your project, right-click on ’GWA-T-12BitbrainsPreprocessing’ and refresh. Ensure ’GWA-T-12BitbrainsPreprocessing script.py’ is placed in the ’GWA-

T-12BitbrainsPreprocessing’ Source Folder. Open the terminal in the navigated directory and import ’GWA-T-12BitbrainsPreprocessing script.py’ by the command below:

import GWA-T-12BitbrainsPreprocessing script.py

1. Replace the absolute paths of the variables ’fastStorage dir’ and ’rnd dirs’ with the path specific to the username on your machine:

C:\Users\<username>\<Folder>\fastStorage\2013-8 C:\Users\<username>\<Folder>\rnd\2013-7 C:\Users\<username>\<Folder>\rnd\2013-8 C:\Users\<username>\<Folder>\rnd\2013-9

1. After aggregating the data, the preprocessed results are saved as CSV files (prepro- cessed fastStorage.csv and preprocessed rnd.csv) in ’GWA-T-12BitbrainsPreprocessing’.

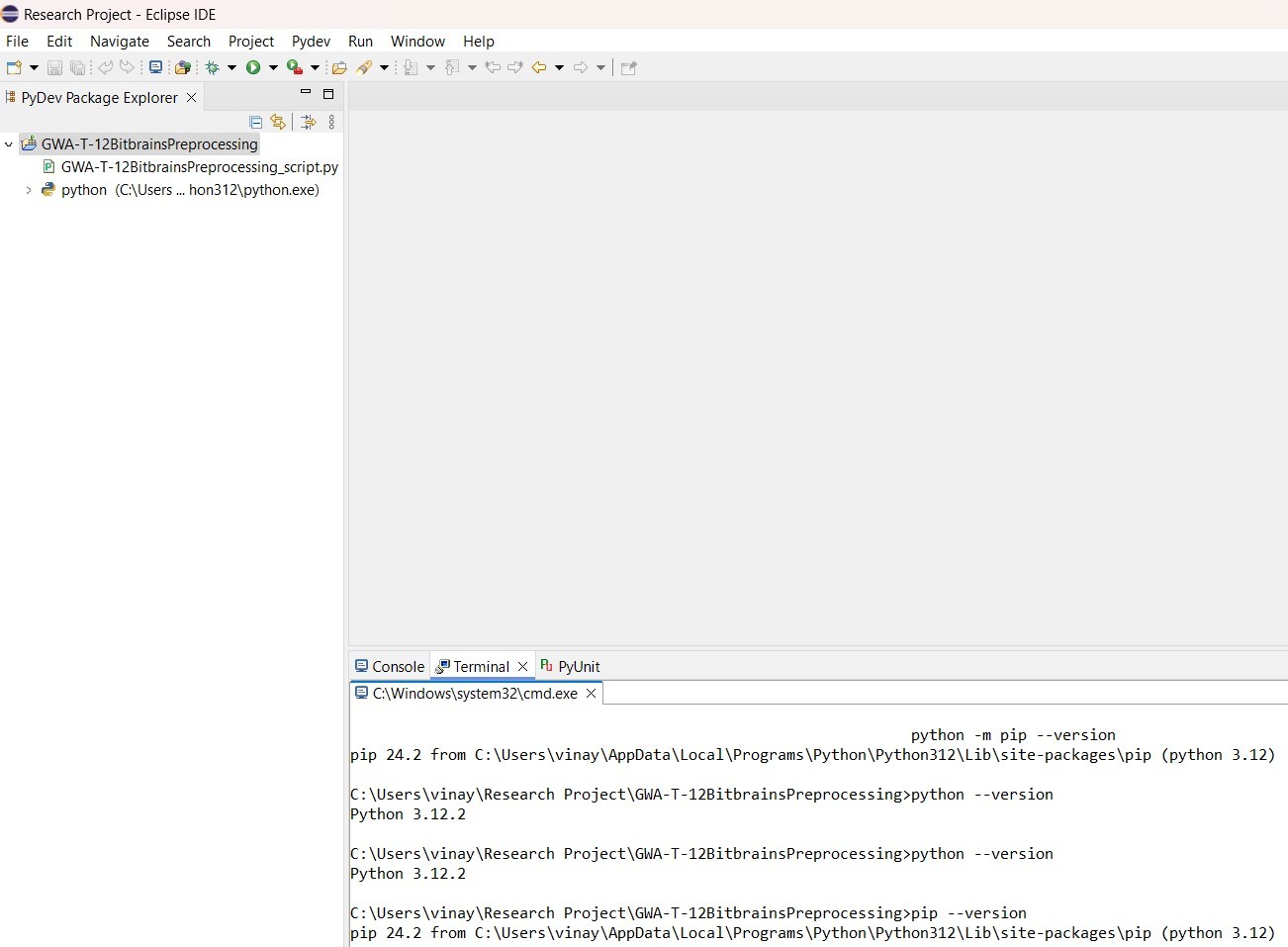


Figure 1: Final Environmental Setup for preprocessing the GWA-T-12 Bitbrains dataset on Eclipse IDE

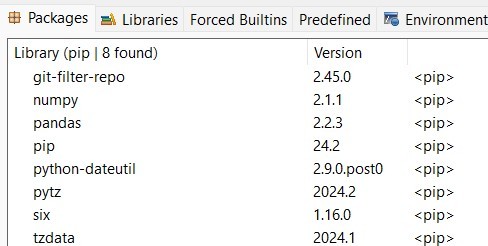


Figure 2: Libraries for the ’GWA-T-12BitbrainsPreprocessing’ Project Folder on Eclipse IDE

# Incorporated Aggregated Algorithmic Programs

### Tools and Technologies

* **Operating System - Windows 11 Pro**
* **Processor - Intel(R) Core i5**
* **Physical Memory - 16 GB RAM**
* **Platform - Eclipse IDE 2024-03 (4.31.0)**
* **Language - Java (jdk-17)**
* **Framework - CloudSim-3.0.3 Toolkit**

### Environmental Setup

1. Download and install CloudSim jar files with all related dependencies from the link [here.](https://jar-download.com/?search_box=cloudsim) Extract the files to a folder on your local machine, preferably the Downloads folder.
2. Create a new Eclipse IDE Workspace Directory. For this configuration, the Workspace Directory shall be titled ’Cloudsim ResearchProject’. Open ’Cloudsim ResearchProject’ and go to ’File > New > Java Project. For the configuration of this Project Directory, the directory shall be titled as ’cloudsimresearchprojectfinal’. Check the box that says ”Use default location” to point to ’cloudsimresearchprojectfinal’.
3. Right click on ’cloudsimresearchprojectfinal’ and go to ’Properties’. Go to ’Java Build Path’ > ’Libraries’. Click on ’Add External JARs’. Select all the .jar files in the extracted CloudSim jar folder from the Downloads folder. Click on ’Apply and Close’.
4. Click on ’src’ inside the project folder ’cloudsimresearchprojectfinal’ and click on ’Remove from Build Path’. Delete the ’src folder’. Right Click on ’cloudsimresearch- projectfinal’ and select ’New’ > ’Source Folder’. Create java/src/main as the new java source folder for ’cloudsimresearchprojectfinal’. Navigate to File Explorer and loc- ate the Java classes that needs to be imported in ’cloudsimresearchprojectfinal’. For this configuration, the aggregated algorithmic programs shall be utilised for evaluating scalability on CloudSim-3.0.3 in this section and AWS in the next section. It is recom- mended to keep the aggregated algorithmic program classes in a folder on your machine, preferably the Downloads Folder. In this configuration, the folder containing the aggreg- ated algorithmic program classes shall be titled ’cloudsimresearchprojectfinalnew’. Drag and drop the aggregated algorithmic program classes titled ’AggregatedIterativeHeurist- icNonConvexEnergyAware’, ’AggregatedHeuristicAugment’ and ’AggregatedPrioritySe- lectionOffloading’.
5. The aggregated algorithmic programs that shall be implemented on CloudSim-3.0.3 and on AWS us-east-1(North Virginia) Region in the next section will incorporate the last iteration on simulation. For each aggregated algorithmic program, change the absolute file path for the simulation output according to the lines below. Note that the backward slashes ’*\\*’ shall be represented as forward slashes ’//’ here:

**Iterative Heuristic Energy-Aware Non-Convex Algorithmic Program:**

PrintStream out = new PrintStream(new FileOutputStream( ”C://Users//*<*username

*>*//Cloudsim ResearchProject//

simulation output aggregatediterativeheuristicnonconvexenergyawareupdatednewtwo.txt”));

writeCloudletResultsToFile(newList, ”C://Users//*<*username *>*// Cloudsim ResearchProject

simulation output aggregatediterativeheuristicnonconvexenergyawareupdatednewtwo.txt”);

cloudletList = loadCloudletsFromCSV(brokerId, ”C:/Users/*<*username *>*/Research Project/GWA-T-12BitbrainsPreprocessing/preprocessed fastStorage.csv”);

cloudletList.addAll(loadCloudletsFromCSV(brokerId, ”C:/Users/*<*username *>*/Research Project/GWA-T-12BitbrainsPreprocessing/preprocessed rnd.csv”));

**Heuristic AUGMENT Non-Convex Algorithmic Program:**

PrintStream out = new PrintStream(new FileOutputStream(”C://Users//*<*username

*>*//Cloudsim ResearchProject

//simulation output aggregatedheuristicaugmentupdatednewthree.txt”));

cloudletList = loadCloudletsFromCSV(brokerId, ”C:/Users/*<*username *>*/Research Project/GWA-T-12BitbrainsPreprocessing/preprocessed fastStorage.csv”);

cloudletList.addAll(loadCloudletsFromCSV(brokerId, ”C:/Users/*<*username *>*/Research Project/GWA-T-12BitbrainsPreprocessing/preprocessed rnd.csv”));

writeEnergyResultsToFile(”C://Users//*<*username *>*//Cloudsim ResearchProject

//simulation output aggregatedheuristicaugmentupdatednewthree.txt”, totalEnergy, cool- ingPower, totalEnergyWithCooling);

writeCloudletResultsToFile(newList, ”C://Users//*<*username*>*// Cloudsim ResearchProject

//simulation output aggregatedheuristicaugmentupdatednewthree.txt”);

**Priority Selection Offloading Algorithmic Program:**

String outputFilePath = ”C://Users//*<*username*>*//Cloudsim ResearchProject

//simulation output aggregatedpriorityselectionoffloadingalgorithmnewfive.txt”;

cloudletList = loadCloudletsFromCSV(brokerId,”C:/Users/*<*username*>*/Research Project/GWA-T 12BitbrainsPreprocessing/preprocessed fastStorage.csv”);

cloudletList.addAll(loadCloudletsFromCSV(brokerId, ”C:/Users/*<*username*>*/Research Project/GWA-T-12BitbrainsPreprocessing/preprocessed rnd.csv”));

1. Run each algorithmic program for the CloudSim simulation. CloudSim will execute each simulation and the simulation output file will be stored in the respective directories for each algorithmic program.

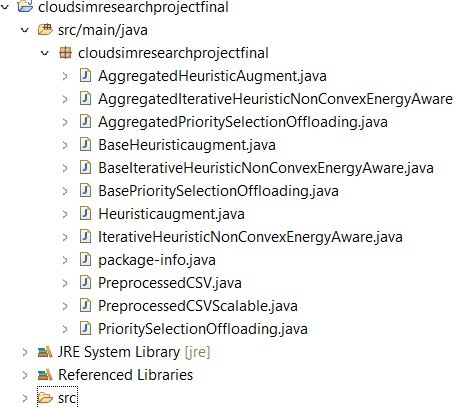


Figure 3: Final Environmental Setup for the Incorporated Aggregated Algorithmic Programs on Eclipse IDE.

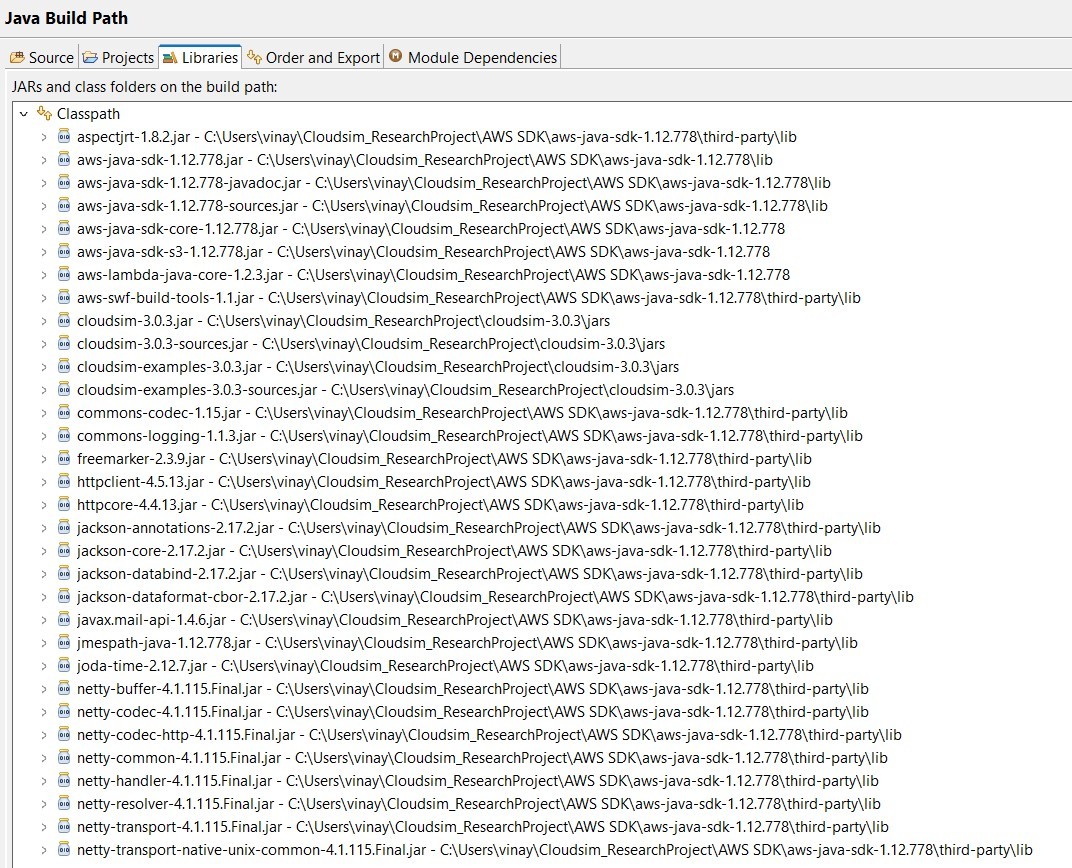


Figure 4: Libraries for the ’cloudsimresearchprojectfinal’ Project Folder on Eclipse IDE. The AWS SDK library is included here as well which shall be described and utilised in the next section.

# AWS Testing

### Tools and Technologies

* **Region - us-east-1 (N. Virginia)**
* **Platform - Eclipse IDE 2024-03 (4.31.0), AWS**
* **AWS Services - Key Management Service us-east-1-KMS-Requests, Elastic Compute Cloud T3CPU, EBS General Purpose (gp3), Elastic Compute Cloud NatGateway, Internet Gateway, Virtual Private Cloud EC2 Free Tier, Virtual Private Cloud VpcEndpoint, Virtual Private Cloud VP- CPeering, CloudWatch, Lambda, Simple Storage Service GeneralPur- poseBuckets**
* **Language - Java (jdk-17)**
* **Framework - CloudSim-3.0.3 Toolkit**
* **Libraries - AWS S3 SDK 2.20.16, AWS Lambda SDK 2.20.16, AWS**

**Lambda Java Core 1.2.1, AWS S3 (Legacy SDK) 1.12.507, Amazon CloudWatch SDK 2.20.16, AWS SDK Core 2.20.16, CloudSim 3.0.3, SLF4J API 2.0.9, SLF4J Simple 2.0.9, Maven Shade 3.4.0, Maven Com- piler 3.10.1**

### Environmental Setup

1. Sign in to the AWS Management Console and navigate to the ’VPC dashboard’. Click on ’Create VPC’. For this configurational setup, the first VPC shall be titled ’NetworkSlicing1 VPC’. Specify the IPv4 CIDR block as 10.1.0.0/16 and select ten- ancy as default. Repeat this step for four more VPC’s and accordingly name them as ’NetworkSlicing2 VPC’, ’NetworkSlicing3 VPC’, ’NetworkSlicing4 VPC’ and ’Network- Slicing5 VPC’. Accordingly assign the IPv4 CIDR blocks respectively as 10.2.0.0/16, 10.3.0.0/16, 10.4.0.0/16, 10.5.0.0/16. Click on ’Edit VPC settings’ for each VPC and ensure ’Enable DNS resolution’ and ’Enable DNS hostnames’ are enabled and saved for each VPC.
2. Navigate to the ’Route Table’ dashboard. Click on ’create route table’. For this configurational setup, the first route table shall be titled ’RouteTable1 Subnet’. Select ’NetworkSlicing1 VPC’ and create the route table. Repeat this step for ’NetworkSli- cing2 VPC’, ’NetworkSlicing3 VPC’, ’NetworkSlicing4 VPC’ and ’NetworkSlicing5 VPC’ for ’RouteTable2 Subnet’, ’RouteTable3 Subnet’, ’RouteTable4 Subnet’ and ’RouteTable5 Subnet’.
3. Navigate to the ’Subnets’ dashboard. Click on ’Create subnet’. Select ’NetworkSli- cing1 VPC’. For this configurational setup, the the subnet shall be titled as ’Network- Slice1 Subnet’. Assign IPv4 CIDR block as 10.1.1.0/24 and create the subnet. Repeat this step for for four more subnets and accordingly name them as ’NetworkSlice2 Subnet’, ’NetworkSlice3 Subnet’, ’NetworkSlice4 Subnet’, ’NetworkSlice5 Subnet’. Accordingly assign the IPv4 CIDR blocks respectively as 10.2.1.0/24, 10.3.1.0/24, 10.4.1.0/24 and 10.5.1.0/24.
4. Go back to the ’Route Table’ dashboard. Click on ’RouteTable1 Subnet’. Go to ’Sub- net associations’ *>* ’Add subnet associations’. Select ’NetworkSlice1 Subnet’ and click on ’Save associations’.

Repeat this step for ’RouteTable2 Subnet’, ’RouteTable3 Subnet’, ’RouteTable4 Subnet’ and ’RouteTable5 Subnet’.

1. Go to the ’Security Groups’ dashboard. Click on ’Create Security Group’. For this configurational setup, the security group shall be titled as ’RingFence1 SecurityGroup’. Select ’NetworkSlicing1 VPC’ and create the security group. Go to ’Inbound Rules’ and click on ’Edit inbound rules’. Add an inbound rule for for ’Type’ HTTPS and select ’RingFence1 SecurityGroup’ as the source. Click on ’Save Rules’. Repeat step 5 for ’RingFence2 SecurityGroup’ with ’RingFence2 SecurityGroup’ as an inbound rule. Repeat step 5 for nine more security groups till ’RingFence11 SecurityGroup’. For ’Ring- Fence2 SecurityGroup’ to ’RingFence11 SecurityGroup’, do not add an inbound rule for the same security group.
2. Go to the Instances Dashboard. Click on ’Launch instance’. For this configura- tional setup, the first instance shall be titled ’NetworkResource0 VM’. Select the ’In- stance Type’ as ’t2.nano’. Select ’NetworkSlicing1 VPC’, ’NetworkSlice1 Subnet’ and ’RingFence1 SecurityGroup’ as the VPC, subnet and security group respectively. Create the first instance. Create 29 more instances with ’Instance Type’ set as ’t2.nano’ till ’NetworkResource29 VM’ with the following configuration as per the table below:

Table 1: Configuration of Network Resources, VPC’s, Subnets, and Security Groups

|  |  |  |  |
| --- | --- | --- | --- |
| **Network Resource (VM)** | **VPC** | **Subnet** | **Security Group** |
| NetworkResource1 VM | NetworkSlicing1 VPC | NetworkSlice1 Subnet | RingFence1 SecurityGroup |
| NetworkResource2 VM | NetworkSlicing1 VPC | NetworkSlice1 Subnet | RingFence1 SecurityGroup |
| NetworkResource3 VM | NetworkSlicing1 VPC | NetworkSlice1 Subnet | RingFence2 SecurityGroup |
| NetworkResource4 VM | NetworkSlicing1 VPC | NetworkSlice1 Subnet | RingFence2 SecurityGroup |
| NetworkResource5 VM | NetworkSlicing2 VPC | NetworkSlice2 Subnet | RingFence3 SecurityGroup |
| NetworkResource6 VM | NetworkSlicing2 VPC | NetworkSlice2 Subnet | RingFence3 SecurityGroup |
| NetworkResource7 VM | NetworkSlicing2 VPC | NetworkSlice2 Subnet | RingFence4 SecurityGroup |
| NetworkResource8 VM | NetworkSlicing2 VPC | NetworkSlice2 Subnet | RingFence4 SecurityGroup |
| NetworkResource9 VM | NetworkSlicing2 VPC | NetworkSlice2 Subnet | RingFence4 SecurityGroup |
| NetworkResource10 VM | NetworkSlicing2 VPC | NetworkSlice2 Subnet | RingFence5 SecurityGroup |
| NetworkResource11 VM | NetworkSlicing2 VPC | NetworkSlice2 Subnet | RingFence5 SecurityGroup |
| NetworkResource12 VM | NetworkSlicing3 VPC | NetworkSlice3 Subnet | RingFence6 SecurityGroup |
| NetworkResource13 VM | NetworkSlicing3 VPC | NetworkSlice3 Subnet | RingFence6 SecurityGroup |
| NetworkResource14 VM | NetworkSlicing3 VPC | NetworkSlice3 Subnet | RingFence6 SecurityGroup |
| NetworkResource15 VM | NetworkSlicing3 VPC | NetworkSlice3 Subnet | RingFence6 SecurityGroup |
| NetworkResource16 VM | NetworkSlicing4 VPC | NetworkSlice4 Subnet | RingFence7 SecurityGroup |
| NetworkResource17 VM | NetworkSlicing4 VPC | NetworkSlice4 Subnet | RingFence7 SecurityGroup |
| NetworkResource18 VM | NetworkSlicing4 VPC | NetworkSlice4 Subnet | RingFence7 SecurityGroup |
| NetworkResource19 VM | NetworkSlicing4 VPC | NetworkSlice4 Subnet | RingFence8 SecurityGroup |
| NetworkResource20 VM | NetworkSlicing4 VPC | NetworkSlice4 Subnet | RingFence8 SecurityGroup |
| NetworkResource21 VM | NetworkSlicing4 VPC | NetworkSlice4 Subnet | RingFence8 SecurityGroup |
| NetworkResource22 VM | NetworkSlicing4 VPC | NetworkSlice4 Subnet | RingFence8 SecurityGroup |
| NetworkResource23 VM | NetworkSlicing5 VPC | NetworkSlice5 Subnet | RingFence9 SecurityGroup |
| NetworkResource24 VM | NetworkSlicing5 VPC | NetworkSlice5 Subnet | RingFence9 SecurityGroup |
| NetworkResource25 VM | NetworkSlicing5 VPC | NetworkSlice5 Subnet | RingFence10 SecurityGroup |
| NetworkResource26 VM | NetworkSlicing5 VPC | NetworkSlice5 Subnet | RingFence10 SecurityGroup |
| NetworkResource27 VM | NetworkSlicing5 VPC | NetworkSlice5 Subnet | RingFence10 SecurityGroup |
| NetworkResource28 VM | NetworkSlicing5 VPC | NetworkSlice5 Subnet | RingFence11 SecurityGroup |
| NetworkResource29 VM | NetworkSlicing5 VPC | NetworkSlice5 Subnet | RingFence11 SecurityGroup |

1. Go back to the ’Security Groups’ dashboard. Click on ’RingFence1 SecurityGroup’ and go to ’Inbound Rules’ *>* ’Edit inbound rules’. Configure three inbound rules as type ’HTTP’, ’HTTPS’ and ’custom TCP’ (1024-65535) for each ip of ’NetworkRe- source4 VM’. For each respective security group till ’RingFence11 SecurityGroup’, ensure the following configuration is met as per the table:

Table 2: Ring Fence Security Group Configurations

|  |  |
| --- | --- |
| **Ring Fence Security Group** | **Configuration Details** |
| RingFence2 SecurityGroup | Configured with the IPs of NetworkResource1 VM, NetworkResource6 VM, and  NetworkResource8 VM. |
| RingFence3 SecurityGroup | Configured with the IPs of NetworkResource3 VM and NetworkResource22 VM. |
| RingFence4 SecurityGroup | Configured with the IPs of NetworkResource1 VM, NetworkResource10 VM, and  NetworkResource4 VM. |
| RingFence5 SecurityGroup | Configured with the IPs of NetworkResource8 VM, NetworkResource18 VM, and  NetworkResource24 VM. |
| RingFence6 SecurityGroup | Configured with the source as RingFence6 SecurityGroup. |
| RingFence7 SecurityGroup | Configured with the IPs of NetworkResource10 VM, NetworkResource19 VM,  NetworkResource22 VM, and NetworkResource23 VM. |
| RingFence8 SecurityGroup | Configured with the IPs of NetworkResource5 VM, NetworkResource17 VM, and  NetworkResource18 VM. |
| RingFence9 SecurityGroup | Configured with the IPs of NetworkResource10 VM, NetworkResource16 VM,  and NetworkResource26 VM. |
| RingFence10 SecurityGroup | Configured with the IP of NetworkResource24 VM. |
| RingFence11 SecurityGroup | Configured with the source as RingFence11 SecurityGroup. |

1. The security groups for the instances for their respective subnets in their VPC’s are now configured. Attached below are images specific to the configuration in this manual.

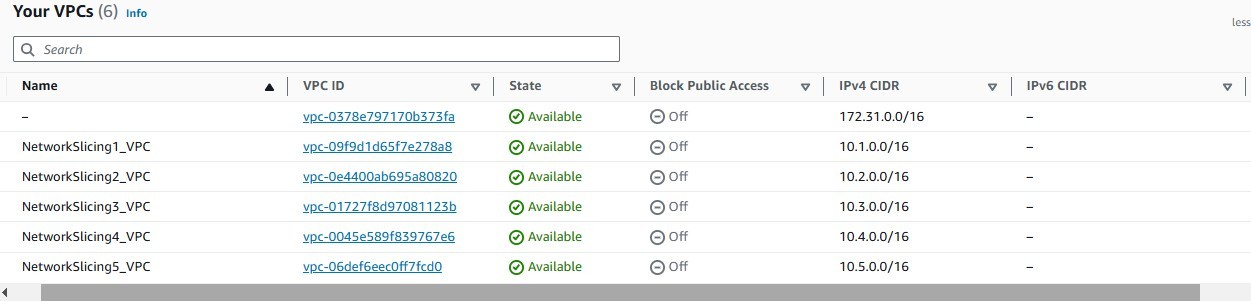


Figure 5: VPC Dashboard for the created VPC’s

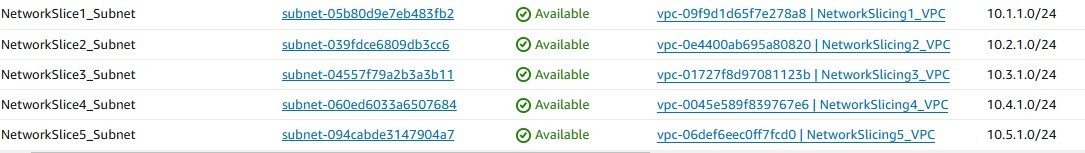


Figure 6: Subnet Dashboard for the created Subnets



Figure 7: Route Table Dashboard for the created Route Tables

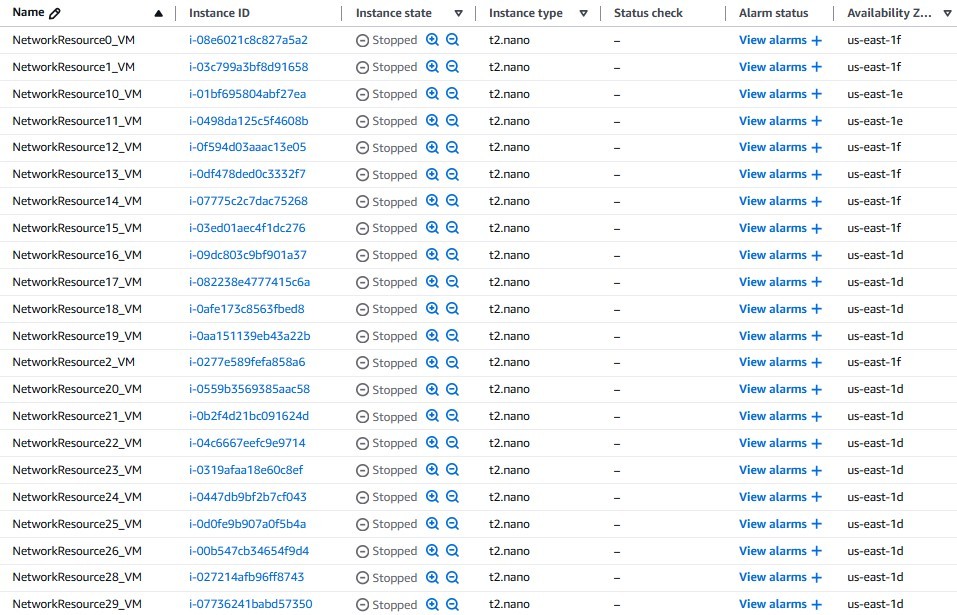


Figure 8: EC2 Instance Dashboard for the created Instances

1. Go to the ’VPC Dashboard’ and click on ’create endpoint’. For this configura- tional purpose, the first VPC endpoint shall be titled ’EP NetworkSlicing1 VPC. Select ’AWS services’ and ’com.amazonaws.us-east-1.s3’ as the service with Type ’Gateway’. Se- lect ’NetworkSlicing1 VPC’ as the VPC and select ’RouteTable1 Subnet’ as the Subnet.

Click on ’Custom’ in Policy and copy the JSON policy from the text document labelled ’JSON for VPC Endpoint EP NetworkSlicing1 VPC to EP NetworkSlicing5 VPC’ to this ’Custom’ Policy. Create the endpoint. Repeat step 9 for ’EP NetworkSlicing2 VPC’ to ’EP NetworkSlicing5 VPC’. Then create another VPC endpoint with the title

’EP NetworkSlicing1 VPC CloudWatchLogs’ for this configurational purpose. Select ’AWS services’ and ’com.amazonaws.us-east-1.s3’ as the service with Type ’Interface’. Se- lect ’NetworkSlicing1 VPC’ as the VPC and select the subnet that is available. En- sure ’Enable DNS name’ is checked and select the available subnet. Select the secur- ity groups ’RingFence1 SecurityGroup’ and ’RingFence2 SecurityGroup’. Click on ’Cus- tom’ in Policy and copy the JSON policy from the text document labelled ’JSON for VPC Endpoint CloudWatchLogs and CloudWatchMonitoring’. Create the VPC end- point ’EP NetworkSlicing1 VPC CloudWatchLogs’. Create another VPC endpoint with the title ’EP NetworkSlicing1 VPC CloudWatchMonitoring’ with the same procedure as the VPC endpoint ’EP NetworkSlicing1 VPC CloudWatchLogs’.

1. Go to the ’VPC Peering’ dashboard. Click on ’Create peering connection’. For this configurational purpose, the first VPC Peering endpoint shall be titled ’Network-

Slice1 NetworkSlice2 VPCPeering’. Select ’NetworkSlicing1 VPC’ and

’NetworkSlicing2 VPC’ as the Requester and Acceptor. Click on ’Create peering connec- tion’. Go to Actions *>* ’Accept request’ to accept the VPC peering connection.

Repeat Step 10 for the following VPC Peering connections titled

’NetworkSlice2 NetworkSlice5 VPCPeering’, ’NetworkSlice2 NetworkSlice4 VPCPeering’ and ’NetworkSlice4 NetworkSlice5 VPCPeering’ for this configurational example. Note that the names of the ’NetworkSlice VPCPeering’ VPC peering endpoints mentioned above shall correspond to that Requester and Acceptor.

1. Go to the ’Internet Gateway’ dashboard. Click on ’Create internet gateway’. For this configurational purpose, the first internet gateway shall be titled as ’IG NetworkSlicing 1’. Go to ’Actions’ *>*’Attach to VPC’ and select ’NetworkSlicing1 VPC’. Click on ’At- tach internet gateway’. Repeat Step 11 for internet gateways ’IG NetworkSlicing 2’ to ’IG NetworkSlicing 5’.
2. Go to the ’Elastic IP’s’ dashboard. Click on ’Allocate Elastic IP address’ and then click on ’Allocate’. Repeat this step four times for five elastic IP’s.
3. Go to the ’NAT Gateways’ dashboard. Click on ’Create NAT gateway’. For this con- figurational purpose, the first ’NAT gateway’ shall be titled ’NG NetworkSlicing 1’. Select ’NetworkSlice1 Subnet’, allocate an elastic ip and create the NAT gateway. Repeat step 13 for four more NAT gateways, namely ’NG NetworkSlicing 2’ to ’NG NetworkSlicing 5’ for ’NetworkSlice2 Subnet’ to ’NetworkSlice5 Subnet with an elastic ip each.
4. Go to the ’Route Tables’ dashboard and click on ’RouteTable1 Subnet’. Go to ’Routes’ and click on ’Edit routes’. Configure three routes for

’NetworkSlice1 NetworkSlice2 VPCPeering’, ’NG NetworkSlicing 1’ and

’EP NetworkSlicing1 VPC’ as targets with 10.2.0.0/16, 0.0.0.0/0 and ’com.amazonaws.us- east-1.s3’ (Endpoint type - Gateway) set as ’Destination’ respectively. Similarly, set the ’Route Table’ configuration for the route tables below.

Table 3: Route Table Configurations for Subnets

|  |  |  |
| --- | --- | --- |
| **Route Table** | **Target** | **Destination** |
| RouteTable2 Subnet | NetworkSlice1 NetworkSlice2 VPCPeering | 10.1.0.0/16 |
| NetworkSlice2 NetworkSlice4 VPCPeering | 10.4.0.0/16 |
| NetworkSlice2 NetworkSlice5 VPCPeering | 10.5.0.0/16 |
| NG NetworkSlicing 2 | 0.0.0.0/0 |
| com.amazonaws.us-east-1.s3 (Gateway) | com.amazonaws.us-east-1.s3 |
| RouteTable3 Subnet | NG NetworkSlicing 3 | 0.0.0.0/0 |
| com.amazonaws.us-east-1.s3 (Gateway) | com.amazonaws.us-east-1.s3 |
| RouteTable4 Subnet | NetworkSlice2 NetworkSlice4 VPCPeering | 10.2.0.0/16 |
| NetworkSlice4 NetworkSlice5 VPCPeering | 10.5.0.0/16 |
| NG NetworkSlicing 4 | 0.0.0.0/0 |
| EP NetworkSlicing4 VPC | com.amazonaws.us-east-1.s3 |
| RouteTable5 Subnet | NetworkSlice2 NetworkSlice5 VPCPeering | 10.2.0.0/16 |
| NetworkSlice4 NetworkSlice5 VPCPeering | 10.4.0.0/16 |
| NG NetworkSlicing 5 | 0.0.0.0/0 |
| EP NetworkSlicing4 VPC | com.amazonaws.us-east-1.s3 |

1. The inter-VM connections and connection to the internet cloud are now estab- lished for the instances for their respective subnets in their VPC’s. Attached below are images specific to the configuration in this manual.

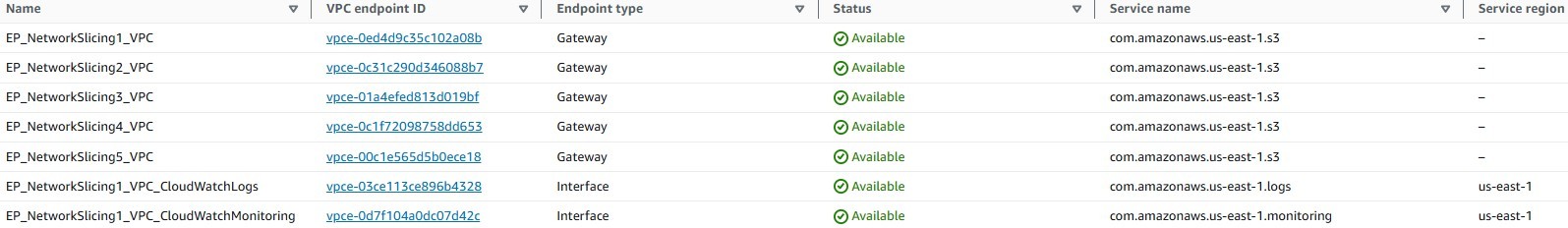


Figure 9: VPC Endpoints Dashboard for the created VPC Endpoints



Figure 10: VPC Peering Dashboard for the active VPC Peering Connections

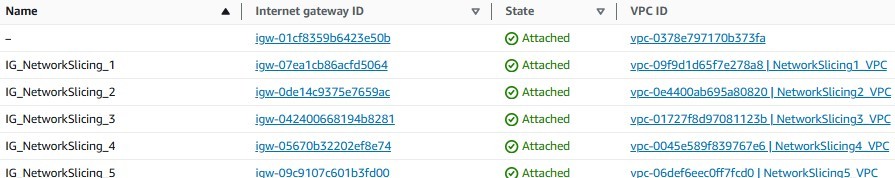


Figure 11: Internet Gateway Dashboard for the created Internet Gateways



Figure 12: Nat Gateway Dashboard for the created NAT Gateways

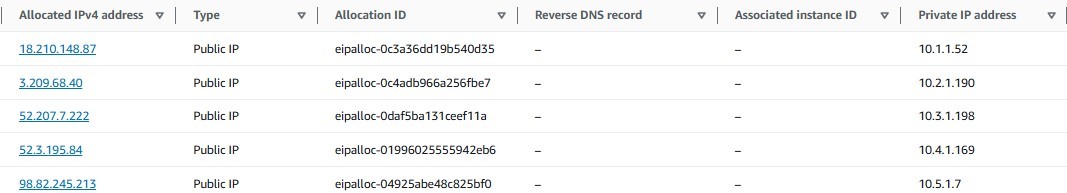


Figure 13: Elastic IP Dashboard for the created Elastic IP’s

1. Go to the IAM Dashboard and click on ’Users’. Click on ’Add users’. For this configurational purpose, the IAM user shall be titled ’Aggregated-Lambda-Cli-User’. Click on Next and then click on ’Attach policies directly’. Attach the permission policies ’AWSLambda FullAccess’ and ’IAMReadOnlyAccess’. Click on Next. Click on ’Add new tag’ and set ’Environment’: ’Development’ for the newly created tag. Create the user ’Aggregated-Lambda-Cli-User’ and click on ’Aggregated-Lambda-Cli-User’. Go to ’Security Credentials’ and click on ’Create access key’. Click on ’Command Line Inter- face(CLI)’, check the ’Confirmation’ and click on ’Next’.

Provide a suitable description to ’Aggregated-Lambda-Cli-User’ and click on ’Next’. Download the .csv file containing the IAM Access Key and Secret Access Key. Note down the Secret Access Key as this secret access key shall be used to configure the AWS CLI from the terminal in Eclipse IDE. Go to ’Tags’ and click on ’Manage Tags’. Ensure your ’Secret Access Key’ with the suitable description is added there as a tag along with ’Environment’: ’Development’.

1. Go to Policies in the IAM Dashboard and click on ’Create policy’. Click on ’JSON’ and open the text document titled ’JSON for AggregatedLambdaBucketPolicy’. Copy the JSON policy from ’JSON for AggregatedLambdaBucketPolicy’ to ’Policy Editor’ un- der ’JSON’ and click on ’Next’. For this configurational purpose, the policy shall be titled ’AggregatedLambdaBucketPolicy’. Provide a suitable description to ’AggregatedLambd- aBucketPolicy’ and click on ’Create policy’. Repeat Step 17 with the policies titled ’Ag- gregatedLambdaNetworkSlicing’ and ’AggregatedMetricsMonitor’ with the JSON policies from ’JSON for AggregatedLambdaNetworkSlicing’ and ’JSON for AggregatedMetric- sMonitor’ respectively.
2. Go to Roles in the IAM dashboard and click on ’Create role’. For the ’Trusted en- tity type’, choose ’AWS service’ and select ’Lambda’ for ’Service or use case’. Click on ’Next’. Attach ’AggregatedLambdaBucketPolicy’, ’AggregatedLambdaNetworkSli- cing’ and ’AggregatedMetricsMonitor’ as ’Permission policies’ and click on ’Next’. For this configurational purpose, the IAM role shall be titled ’AggregatedLambdaExecution- Role’. Create ’AggregatedLambdaExecutionRole’ and navigate to the ’Trust relation- ships’ tab. Verify that the JSON policy matches the policy as per the text document ’JSON Trust Relationships for AggregatedLambdaNetworkSlicing’. Go back to Policies in the IAM Dashboard and attach ’Aggregated-Lambda-Cli-User’ with ’Aggregated- LambdaExecutionRole’ for ’AggregatedLambdaBucketPolicy’, ’AggregatedLambdaNet- workSlicing’ and ’AggregatedMetricsMonitor’. Go back to Users in the IAM Dashboard and click on ’Aggregated-Lambda-Cli-User’. Go to the ’Permissions’ tab and click on ’Add permissions’ *>*’Attach policies directly’. Attach the policies ’AggregatedLambd- aBucketPolicy’, ’AggregatedLambdaNetworkSlicing’ and ’AggregatedMetricsMonitor’ to ’Aggregated-Lambda-Cli-User’. Click on ’Add permissions’.
3. Go to the Lambda Dashboard and click on ’Create function’. For this configura- tional purpose, the Lambda function shall be titled ’AggregatedLambdaNetworkSlicing’.

Select ’Java 17’ as the runtime and click on ’Additional Configurations’. Check ’En- able VPC’ and select ’NetworkSlicing1 VPC’ with ’NetworkSlice1 Subnet’ as the VPC and Subnet. Select ’RingFence1 SecurityGroup’ and ’RingFence2 SecurityGroup’ as the security groups. Click on ’Create function’. Go to the ’Configuration’ tab and click on ’General Configuration’ in ’AggregatedLambdaNetworkSlicing’. Click on ’Edit’ and adjust ’Memory’ to 2048 MB, ’Timeout’ to 7 minutes and ’Existing role’ to ’Aggregated- LambdaExecutionRole’. Click on ’Save’.

1. Go to the Amazon S3 Dashboard and click on ’create bucket’. For this configurational purpose, the S3 bucket shall be titled ’aggregatedlambdabucketoutput’. Click on ’Create bucket’. Upload the ’preprocessed fastStorage.csv’ and ’preprocessed rnd.csv’ from the ’Research Project/GWA-T-12BitbrainsPreprocessing’ folder in your machine.
2. The IAM user, role and policies is now configured to use the newly created Lambda function to upload each CloudSim Project as a packaged fat JAR/Zip file using AWS CLI from the terminal in Eclipse IDE. The simulation outputs for each CloudSim Pro- ject shall be stored in the S3 Bucket respectively. Attached below are images specific to the configuration in this manual.

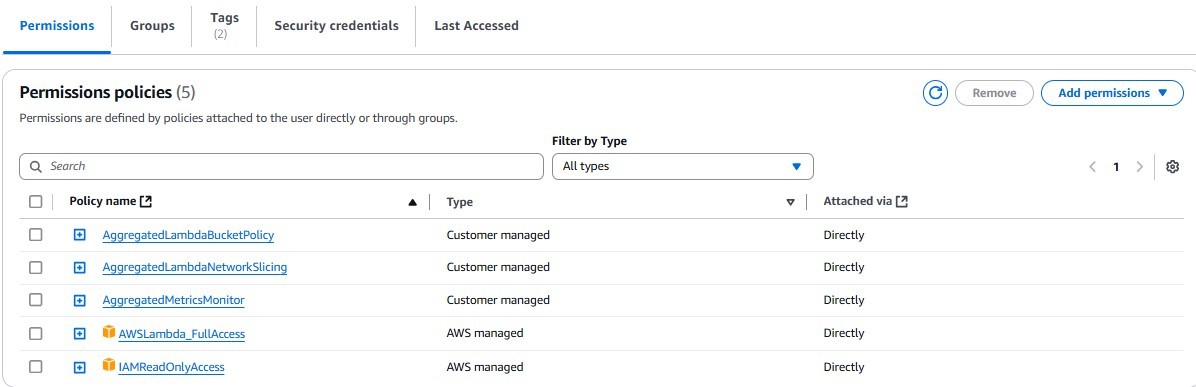


Figure 14: IAM User Dashboard for the newly created user to access IAM-related details, Lambda, S3 Bucket and CloudWatch.

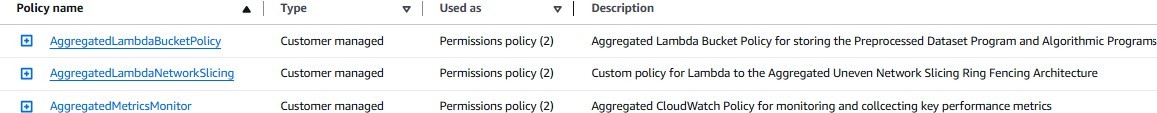


Figure 15: IAM Policies Dashboard for the newly created policies facilitating access to Lambda, S3 Bucket and CloudWatch by the user.

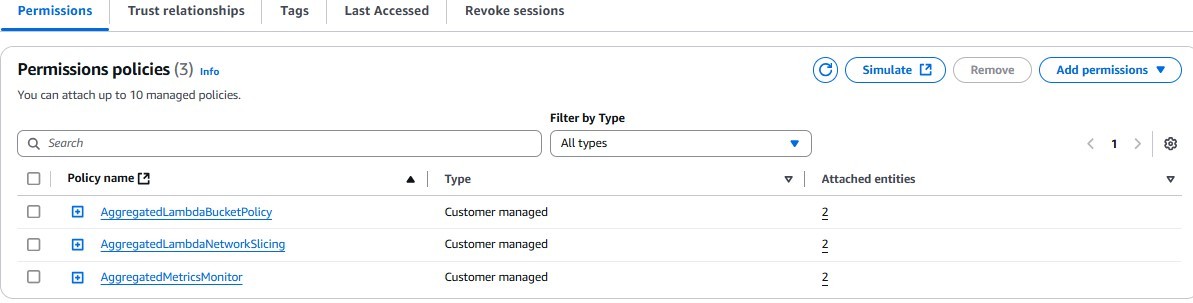


Figure 16: IAM Role Dashboard for the newly created IAM role on the IAM user for S3 Bucket and CloudWatch.



Figure 17: IAM Policies Dashboard for the newly created IAM policies for S3 Bucket and CloudWatch used by the IAM user.

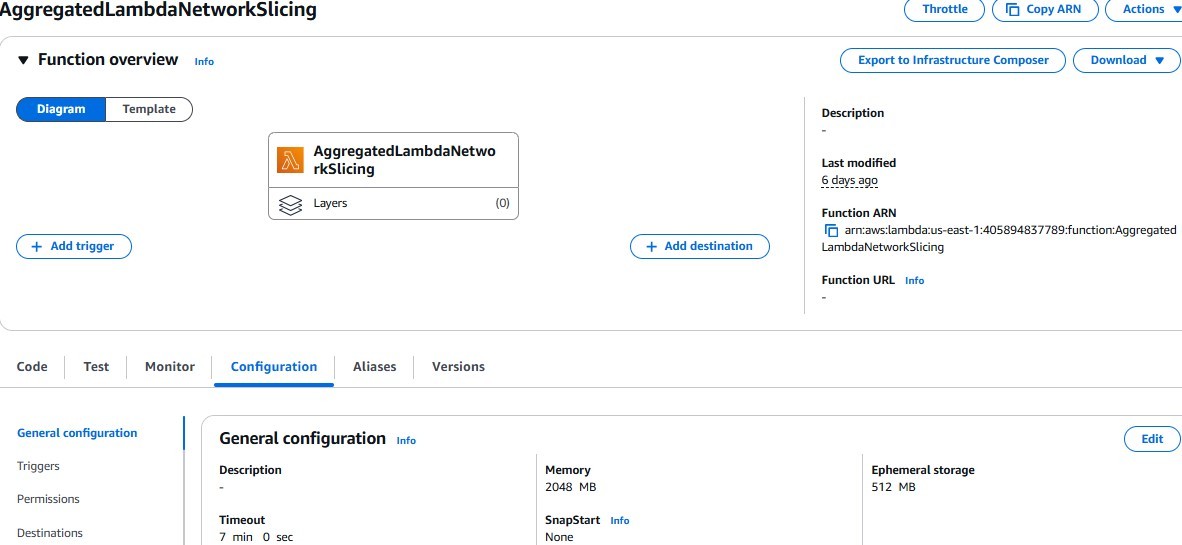


Figure 18: Lambda Dashboard for the newly Lambda function to be used by the IAM user.

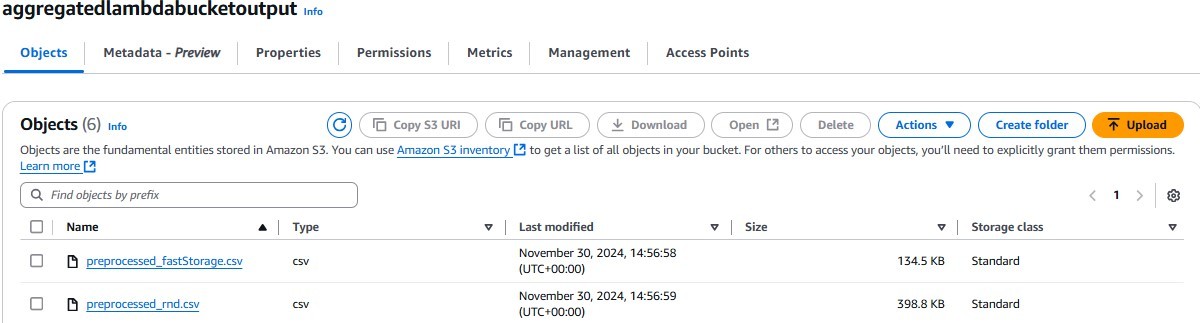


Figure 19: S3 Bucket Dashboard for the logs to store the simulation outputs for aggregated algorithmic program.

1. Go to Eclipse IDE and right-click on Package explorer and click on *>*’Java Pro- ject’. For the ’iterative heuristic energy-aware non-convex’ aggregated algorithmic pro- gram in this configurational purpose, the ’CloudSim’ Project shall be titled ’lambdafina- literativeresearchprojectfinal’. Click on ’Next’ and then ’Finish’. Move into the directory ’lambdafinaliterativeresearchprojectfinal’ in Eclipse IDE and enter the following com- mand below:

aws configure

Provide the Access Key ID, Secret Access Key from the .csv file downloaded earlier. Since this configurational example’s architectural environment is in us-east-1, select ’us- east-1’ as the default region. Right-click on click on ’Configure’ *>*’Convert to Maven Project’. Click on ’Finish’. Enter the following command below in the terminal on Ec- lipse IDE:

mvn install:install-file -Dfile=C:/Users/<username>/Cloudsim Research Project/cloudsim-3.0.3/jars/cloudsim-3.0.3.jar -DgroupId=org.cloudbus. cloudsim -DartifactId=cloudsim -Dversion=3.0.3 -Dpackaging=jar -DpomFile= C:/Users/<username>/Cloudsim ResearchProject/cloudsim-3.0.3/jars/ cloudsim-3.0.3.pom

Click on the created pom.xml created in ’lambdafinaliterativeresearchprojectfinal’. Copy and paste the contents of the text document ’pom.xml for Aggregated Algorithmic Pro- grams on AWS’ into the newly created pom.xml in ’lambdafinaliterativeresearchproject- final’. Enter the following command below in the terminal on Eclipse IDE.

mvn clean install -U

Navigate to the src folder in ’lambdafinaliterativeresearchprojectfinal’ and delete module- info.java. Right-click on src/main/java in ’lambdafinaliterativeresearchprojectfinal’ and click on ’Build Path’ *>*’Remove from Build Path’. Delete the src folder. Right-click on the project folder ’lambdafinaliterativeresearchprojectfinal’ and click on ’New’ *>*’Source Folder’. Name the source folder as ’src/main/java’. Right-click on the project folder ’lambdafinaliterativeresearchprojectfinal’ and click on ’New’ *>*’Class’. Name the class ’FinalAggregatedIterativeHeuristicNonConvexEnergyAwareLambda’. Open the text doc- ument labelled ’FinalAggregatedIterativeHeuristicNonConvexEnergyAwareLambda.java’ and copy the packages from the text document ’FinalAggregatedIterativeHeuristicNon- ConvexEnergyAwareLambda.java’ into the algorithmic program on Eclipse. Then copy the rest of the program from the text document ’FinalAggregatedIterativeHeuristicNon- ConvexEnergyAwareLambda.java’ onto the algorithmic program on Eclipse. Save the program. Open the text document labelled ’Additional Dependencies for Aggregated Algorithmic Programs on AWS’ on your machine and copy the extra dependencies to your pom.xml created in ’lambdafinaliterativeresearchprojectfinal’. Run the following command below on the terminal in Eclipse IDE:

mvn clean package

1. Right-click on the project folder ’lambdafinaliterativeresearchprojectfinal’ *>*’Update Project’. Check ’Force Update of Snapshots/Releases’ and click on OK. Enter the fol- lowing commands on the terminal in the project folder lambdafinaliterativeresearchpro- jectfinal’ in Eclipse IDE:

aws lambda update-function-code –function-name AggregatedLambdaNetwork Slicing –zip-file fileb://C:/Users/*<*username*>*/Cloudsim ResearchProject

*<*projectfolder*>*/target/FinalAggregatedIterativeHeuristicNon ConvexEnergyAwareLambda-1.0-SNAPSHOT.jar

aws lambda update-function-configuration --function-name AggregatedLambda NetworkSlicing --handler lambdafinaliterativeresearchprojectfinal.Final FinalAggregatedIterativeHeuristicNonConvexEnergyAwareLambda::handleRequest

aws lambda get-function --function-name AggregatedLambdaNetworkSlicing

1. Go to the ’AggregatedLambdaNetworkSlicing’ Lambda function in your Lambda Dashboard and click on the configuration tab ’Test’. For this configurational purpose, the test event shall be titled ’AggregatedIterLambdaTest’. Open the text document ’JSON for AggregatedLambdaNetworkSlicing AggregatedIterLambdaTest’ and copy the JSON to the Event JSON in ’AggregatedLambdaNetworkSlicing’ Lambda on the Lambda dashboard. The Lambda function ’AggregatedLambdaNetworkSlicing’ is now ready to run. Run the test event ’AggregatedIterLambdaTest’ and click on ’Logs’ after the event is run. In the CLoudWatch Dashboard, click on ’All metrics’ and then click on the newly created CloudWatch custom namespace ’AggregatedMetricsMonitor’ *>*’Metrics with no dimensions’.

Click on ’Graphed Metrics’ by selecting the ’performance metrics’ specific to the graph- ical display and adjust the period if the ’AggregatedLambdaNetworkSlicing’ Lambda was created far back. Repeat steps 22 and 23 exactly as per the configuration below for the other aggregated algorithmic programs which are ’lambdafinalheuristicresearchprojectfi- nal’ and ’lambdafinalpriorityresearchprojectfinal’ as per the table in this configurational example below. Replace the project folder name and class name when the AWS CLI commands are run as per the terminal on Eclipse IDE. The aws configure command does not need to be run again.

Table 4: Project Folder and Class Details

|  |  |
| --- | --- |
| **’Project Folder’** | **’Class’** |
| lambdafinalheuristicresearchprojectfinal | AggregatedHeuristicAugmentLambda.java |
| lambdafinalpriorityresearchprojectfinal | FinalAggregatedPriorityLambda.java |

Table 5: ’JSON for AggregatedHeuristicAugmentLambda Lambda’ and Lambda Test Event Details

|  |  |
| --- | --- |
| **JSON for AggregatedHeuristicAugmentLambda Lambda** | **Lambda Test Event Details** |
| AggregatedLambdaNetworkSlicing AggregatedHeurLambdaTest | AggregatedHeurLambdaTest.java |
| AggregatedLambdaNetworkSlicing AggregatedPriLambdaTest | AggregatedPriLambdaTest.java |

Attached below are images specific to the configuration in this manual for the test event ’AggregatedIterLambdaTest’ run on ’AggregatedLambdaNetworkSlicing’ Lambda.

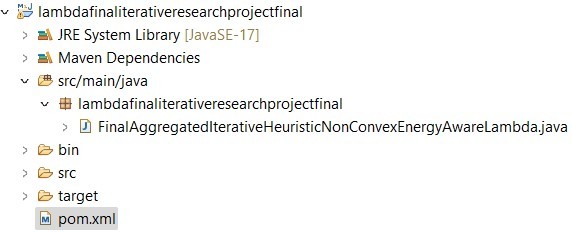


Figure 20: Project Explorer of the Iterative Heuristic Energy-Aware Non-Convex Algorithmic program on Eclipse IDE.

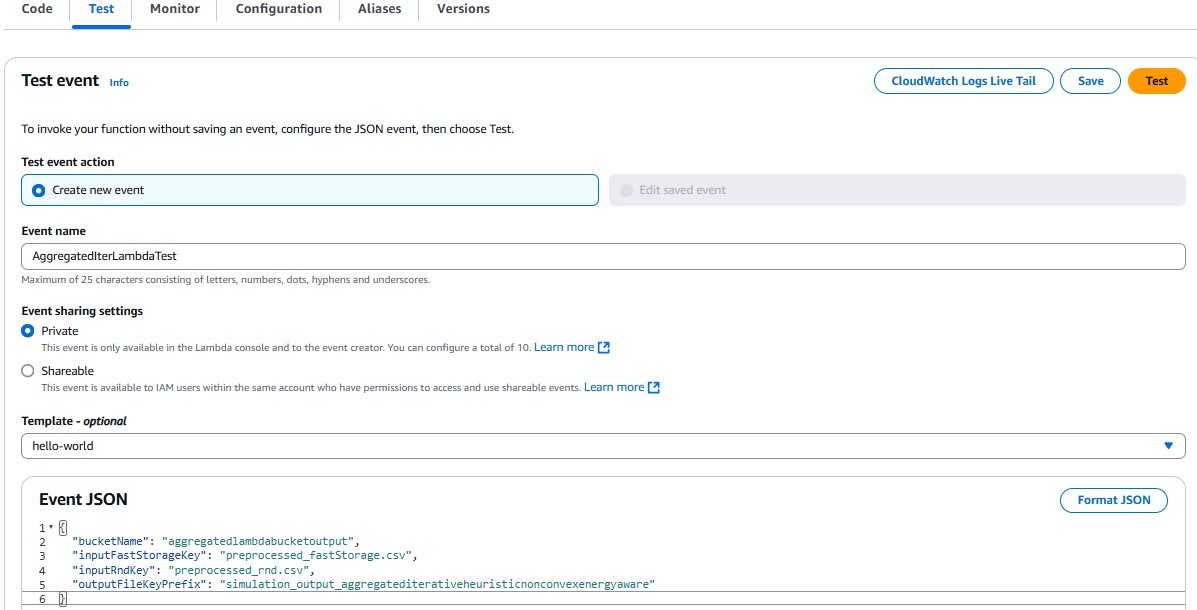


Figure 21: Lambda Test Event for the Iterative Heuristic Energy-Aware Non-Convex Algorithmic program on the Lambda function.

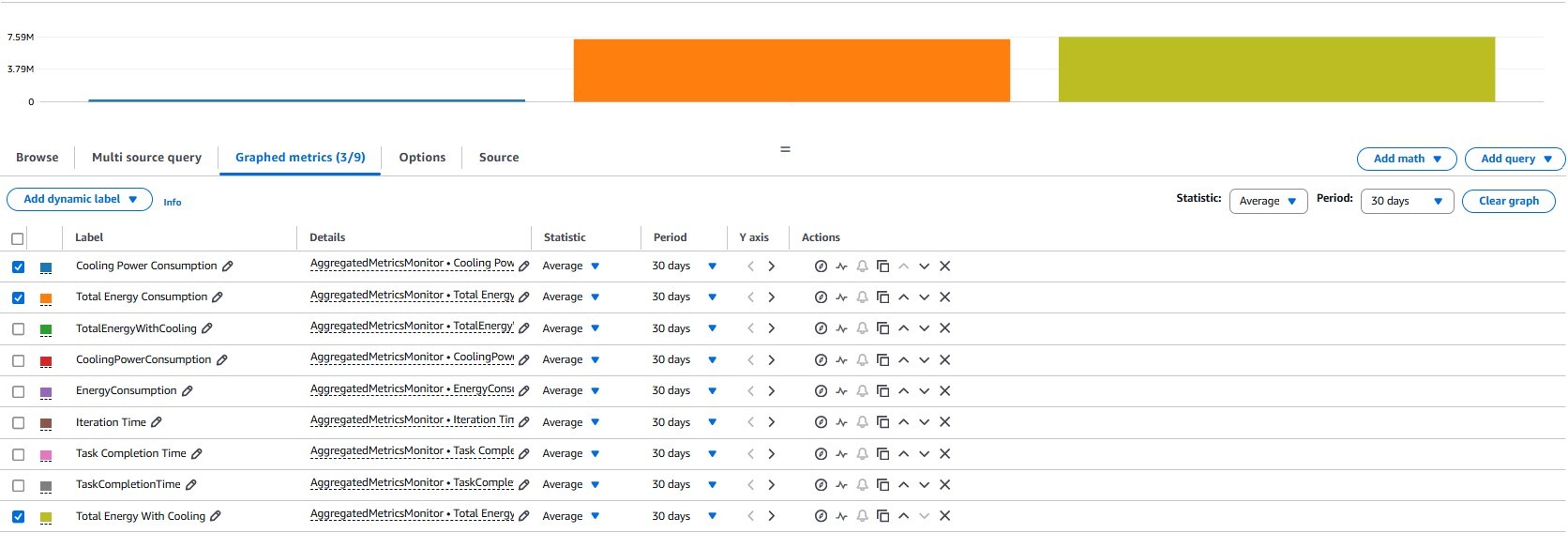


Figure 22: CloudWatch Monitoring for the Lambda function.

# Implementation Notes

### CloudSim Implementation and AWS Environmental Setup

* + 1. **CloudSim Environmental Setup**

**Creation of CloudSim Environment**: The CloudSim environment was initialised for the preprocessed dataset program defining the number of users.

**Creation of Datacenters, Virtual Machine Allocation and Cloudlet Submission**: Next, created datacenters that consisted of hosts configured with CPU cores, RAM, network bandwidth and storage were incorporated as network slices and hosts as ring fences. Created VM’s configured for MIPS(million instructions per second) as L3 computational network resource in ring fences were submitted to the Datacenter Broker.

Created workload tasks from preprocessed fastStorage.csv and preprocessed rnd.csv were simulated as workloads into VM’s.

**Inter-VM Communication and Total Energy Consumption**: VM-to-VM commu- nication links between the L3 computational network resources were incorporated us- ing the NetworkTopology functionality. The cooling constraint was indirectly simulated through a shared constraint on bandwidth or energy and the total energy consumption with all logical components was calculated.

**Simulation Execution and Printed Results**: With the simulation finished here, the total energy consumption including the performance metrics were printed to the output log file for analysis.

* + 1. **AWS Environmental Setup**

An architectural diagram representing the implementation of the aggregated uneven net- work slicing ring fencing architecture with associated components for assessing scalability is provided below:

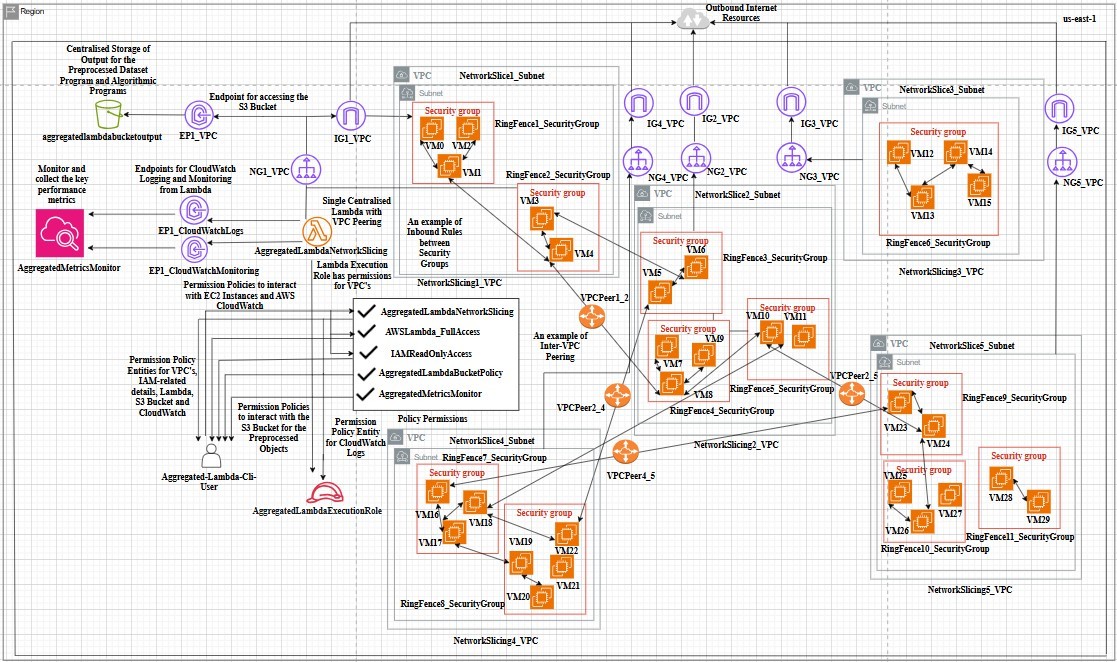


Figure 23: AWS Setup for the Aggregated Uneven Network Slicing Ring Fencing Architecture with respective requirements in us-east-1.

**Environmental Configuration**: The AWS environment above setup five VPC’s with corresponding subnets consisting of eleven security groups that contained thirty VM’s or EC2 instances. VPC’s with corresponding subnets, security groups and VM’s can be compared to network slices, ring fences and computational network resources. As an additional factor in uneven aggregation, the subnets were in different availability zones but defined in the same region. VPC peering setup inter-slice communication where cor- responding route tables with explicit subnet association for those subnets were configured with outbound rules for security groups to allow only required traffic between VM’s.

Likewise, Inter-Ring Fence communication was defined with inbound rules for security groups through VM-level access. To follow up on inter VM-VM connectivity between the respective security groups, each respective subnet ensured it’s own NAT gateway and Internet Gateway for access to outbound internet resources. Next, the lambda function ’AggregatedLambdaNetworkSlicing’ was utilised as a test event evaluating each algorithmic program against each other specifically for the last iteration. ’Aggregated- LambdaNetworkSlicing’ Lambda was associated with the first network slice titled ’Net- workSlicing1 VPC’ to ensure peering connections to all other VPC’s. After the test event is successfully run, the dataset based metrics such as CPU, memory, disk usage, network bandwidth are logged and monitored by ’AggregatedMetricsMonitor’ CloudWatch. ’AggregatedMetricsMonitor’ CloudWatch with it’s two associated VPC endpoints facilit- ated CloudWatch Logs and CloudWatch Monitoring with collected performance metrics respectively. The logged performance metrics specific to ’AggregatedLambdaNetworkSli- cing’ function execution were viewed in custom dashboards to benchmark each algorithmic program on the basis of energy consumption, iteration time and task completion time. A more detailed view of the performance metrics specific to each run algorithmic program such as CloudSim metrics can be downloaded from the ’aggregatedlambdabucket’ S3 Bucket as simulated output files. The AggregatedLambdaNetworkSlicing’ test event that ran the algorithmic programs collected from ’aggregatedlambdabucket’ was associated with a NAT gateway and attached VPC endpoint that routed private network resource traffic in their own private VPC’s.

**IAM-specific permissions:** ’AggregatedLambdaBucketPolicy’ and ’AggregatedMetric- sMonitor’ were attached policy permissions on ’AggregatedLambdaExecutionRole’ that stored the output of the algorithmic programs with object keys that facilitated Cloud- Watch Logs and Monitoring respectively. These objects keys were the cleaned datasets preprocessed fastStorage.csv and preprocessed rnd.csv. ’AggregatedLambdaExecution- Role’ was an IAM role that leveraged VPC peering for the remaining VPC’s and IAM- related details for ’AggregatedLambdaNetworkSlicing’ Lambda, ’aggregatedlambdabuck- etoutput’ S3 bucket and ’AggregatedMetricsMonitor’ CloudWatch. Attached to ’Aggreg- atedLambdaExecutionRole’ was a policy permission titled ’AggregatedLambdaNetwork- Slicing’ that fulfilled IAM-specific tasks to the Network Slicing Ring Fencing architecture. An IAM user titled ’Aggregated-Lambda-Cli-User’ ensured correct association with all the aforementioned policy permissions including the ’AggregatedLambdaNetworkSlicing’ policy execution role for all sources. For each incorporated program in Aggregated- LambdaNetworkSlicing Lambda, the updated function code and IAM-specific details to the AggregatedLambdaNetworkSlicing policy were reflected in AWSLambda FullAccess and IAMReadOnlyAccess for Aggregated-Lambda-Cli-User. AggregatedMetricsMonitor CloudWatch collected detailed performance metrics that was stored through the ’aggreg- atedlambdabucketoutput’ S3 Bucket as a means of centralised storage for the programs.