Logic used for Assignment – Databases

Table of Contents

Task-01: Data Loading – Logic

DriverManager from Java SQL package is used to establish connection with the Upgrad AWS RDS MySQL Database URL, using provided credentials.

MongoDB Java driver package is used to create client instance for establishing connection to AWS MongoDB NoSQL database, created in AWS Cloud using shared AWS account.

[Refer Appendix for AWS screenshots for VPC, EC2, Security group configurations & mongo db installation for reference.]

Created mongoDB "pgcdata" with collections named as "products".

Upon establishing connection with AWS RDS MySQL DB & AWS EC2 MongoDB, part of initialization steps, all the documents under "products" collection were deleted. Though this operation isn't mentioned as an explicit requirement part of the assignment, this is done to ensure that query results remain consistent upon multiple executions of the assignment application.

Same cleanup operation shall be done as last operation of this assignment application. But since it is possible for the assignment evaluators to verify the mongoDB collection from console, currently this cleanup operation at end of application execution is disabled/commented.

Considering the limited scope of applications evaluation rubrics, following assumptions were made –

1. MySQL table will contain all the fields of type VARCHAR / Strings.
2. List of records shall not be huge in size, yet restricted fetch size to 10, considering given MySQL dataset tables were having only 5 records.
3. Havent employed any other 3rd party libraries for ResultSet to JSON to MongoDB Import based data migration. Used simple methods only.

Created static stub methods to query individual tables & refactored common methods for better code reuse. Below methods are created for same purpose, which are called from main() method of Driver class –

1. importRDSVarcharTablesToMongoDb() - Method to import multiple RDS SQL table with only VARCHAR (String) fields to Mongo DB, uses below method.
2. insertVarcharTableToMongoDb() - Method to import records fetched by given sql query on a RDS SQL table with only VARCHAR (String) fields by adding a new category field referring to product category to Mongo DB. This method currently fetches records by paging 10 records at once. This shall be set as per needs. Uses below method.
3. getDocumentList() - Method to construct Mongo Document object for each records. This method uses to fetch individual fields of each records as present part of returned upon querying SQL table with only VARCHAR/String fields.

Refer Appendix – Sequence Diagram for method call flow.

Task-2: Queries – Logic

Query-01: Show the complete product inventory with common attributes. (Print the data only for the common attributes which are there for all three product categories)

Logic:

1. Build Query to find all documents
2. add (optionally) projection to include only the common fields & get MongoCursor.
3. Iterate cursor & Call PrintHelper.printSingleCommonAttributes() to display the common attributes on the Screen.

Additionally, usage of Projections here seems to be optional, as the PrintHelper.printSingleCommonAttributes() method is hardcoded to print the common attributes only, irrespective of whether other non-common / non-null columns are present or not part of the fetched result mongo cursor.

Query-02: Get the information about the first five products under the category 'Mobiles'.

Logic:

1. Build Query to find all documents with category as "Mobile"
2. Limit records to 5
3. get MongoCursor on returned documents
4. Iterate cursor & call respective PrintHelper.printAllAttributes() method to display output as required.

Query-03: Get the products ordered by category without their auto-generated '_id' column in MongoDB.

Logic:

1. Build Query to find all documents
2. Use Projections to exclude ID field &
3. Sort documents by $Category field in descending order
4. Get MongoCursor on returned documents
5. Iterate cursor & call respective PrintHelper.printAllAttributes() method to display output as required.

Query-04: Get the product count for every category.

Logic:

1. Build Query to aggregate documents, by grouping documents by "$Category"

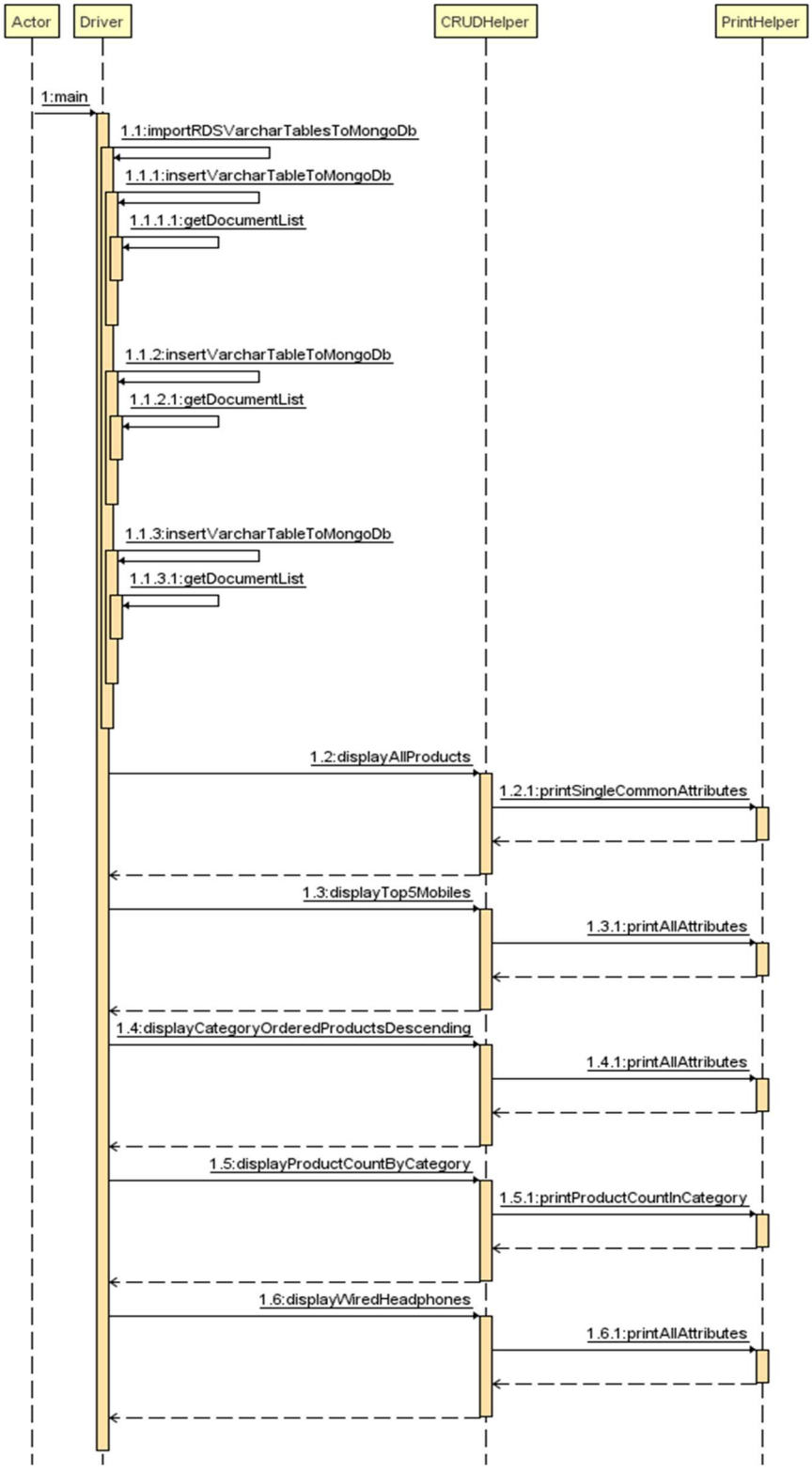
2. Accumulate the number of records using Accumulator.sum to keep adding 1 for every record grouped by "$Category" field
3. Sort the list & get MongoCursor on returned documents
4. Iterate cursor & call respective PrintHelper.printProductCountInCategory() method to display output as required.

Query-05: Get the information about all the wired headphones.

Logic:

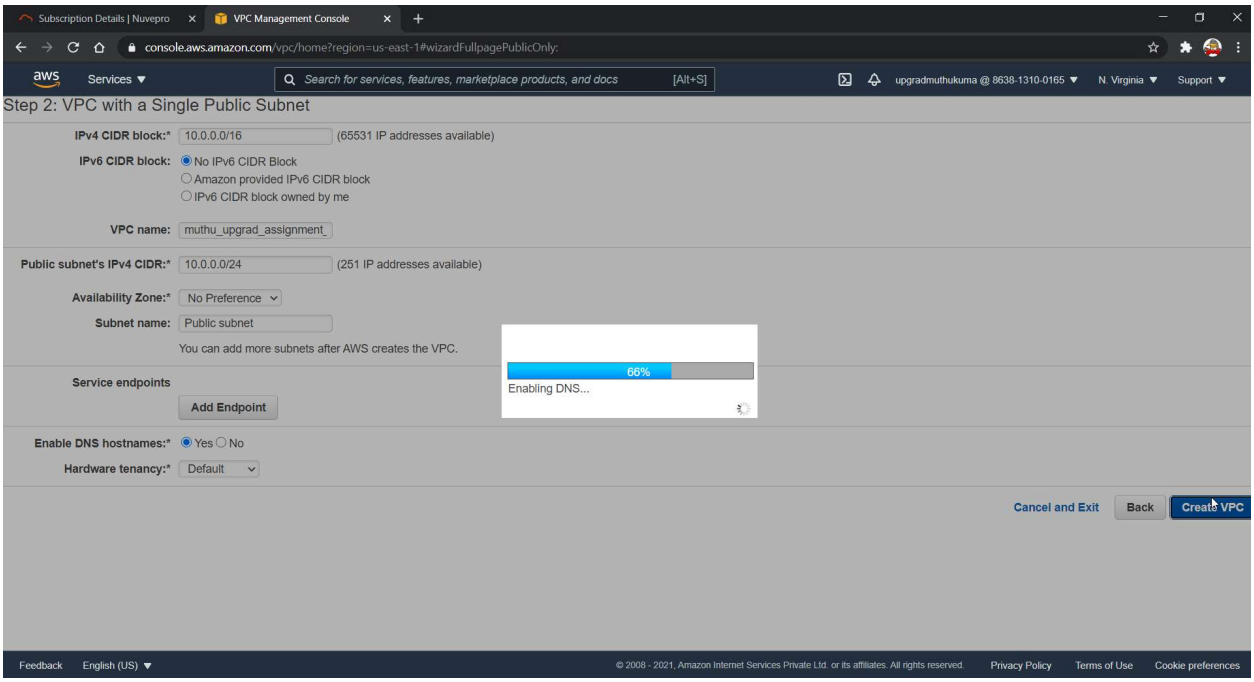
1. Create BasicDBObject to define criterias, such that “Category” key contains value “Headphone” & “ConnectorType” key contains value “Wired”
2. Build query to find all documents matching criteria set by BasicDBObject
3. Get MongoCursor on returned documents.
4. Iterate cursor & call respective PrintHelper.printAllAttributes() method to display output as required.

APPENDIX – Sequence Diagram (Method Call Flow)

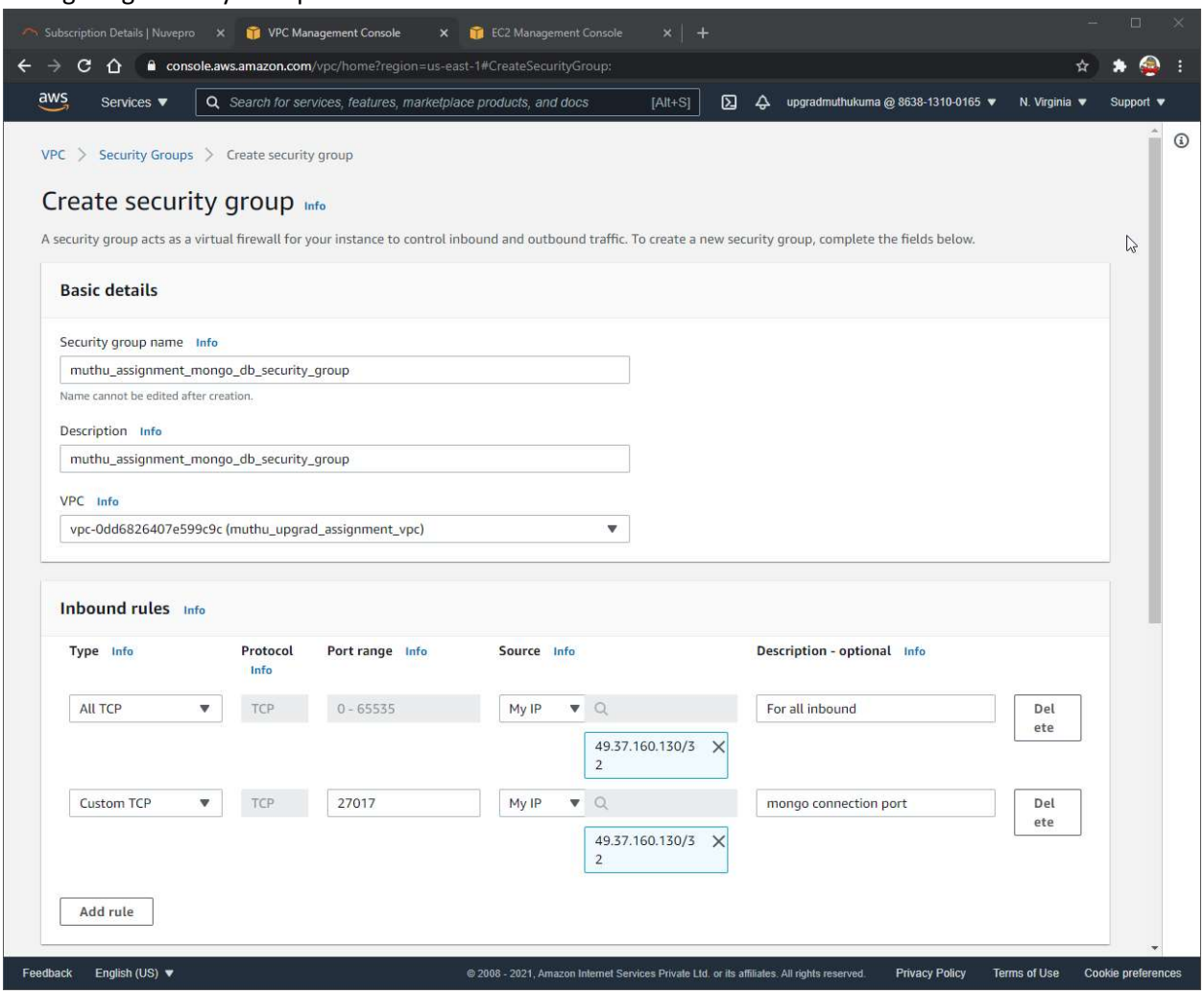


APPENDIX – AWS Screenshots

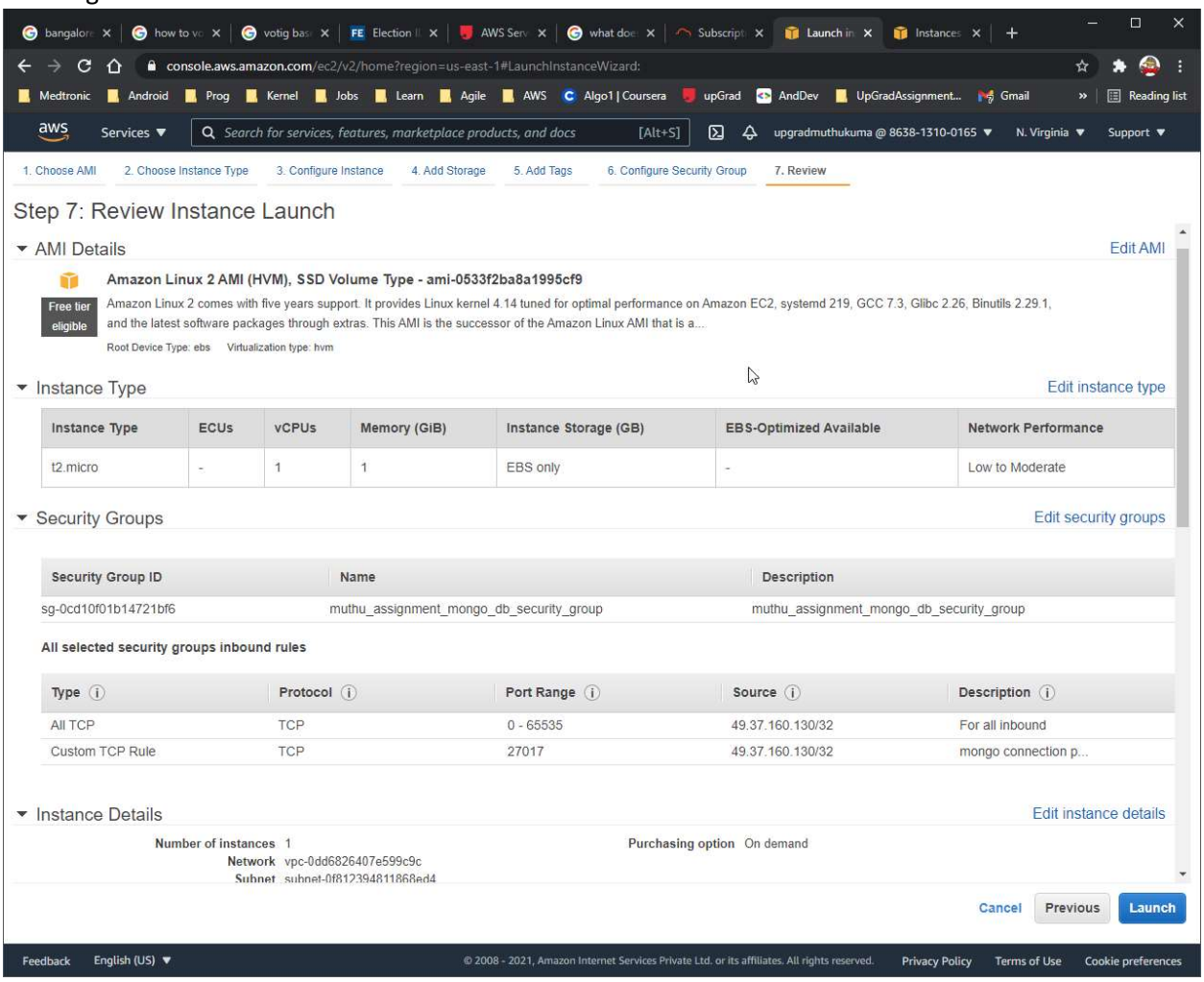
1. VPC Creation



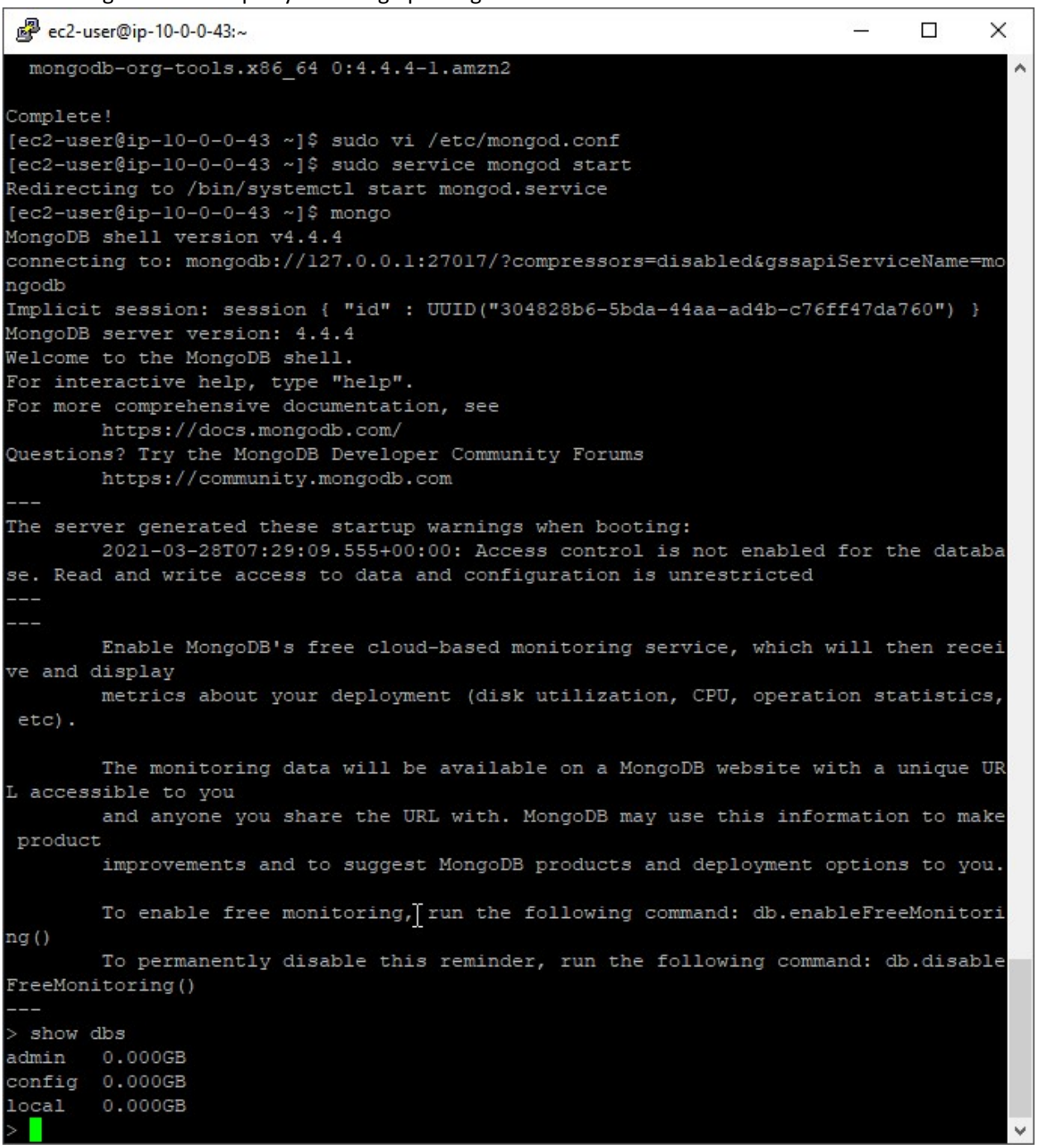
2. Configuring Security Group



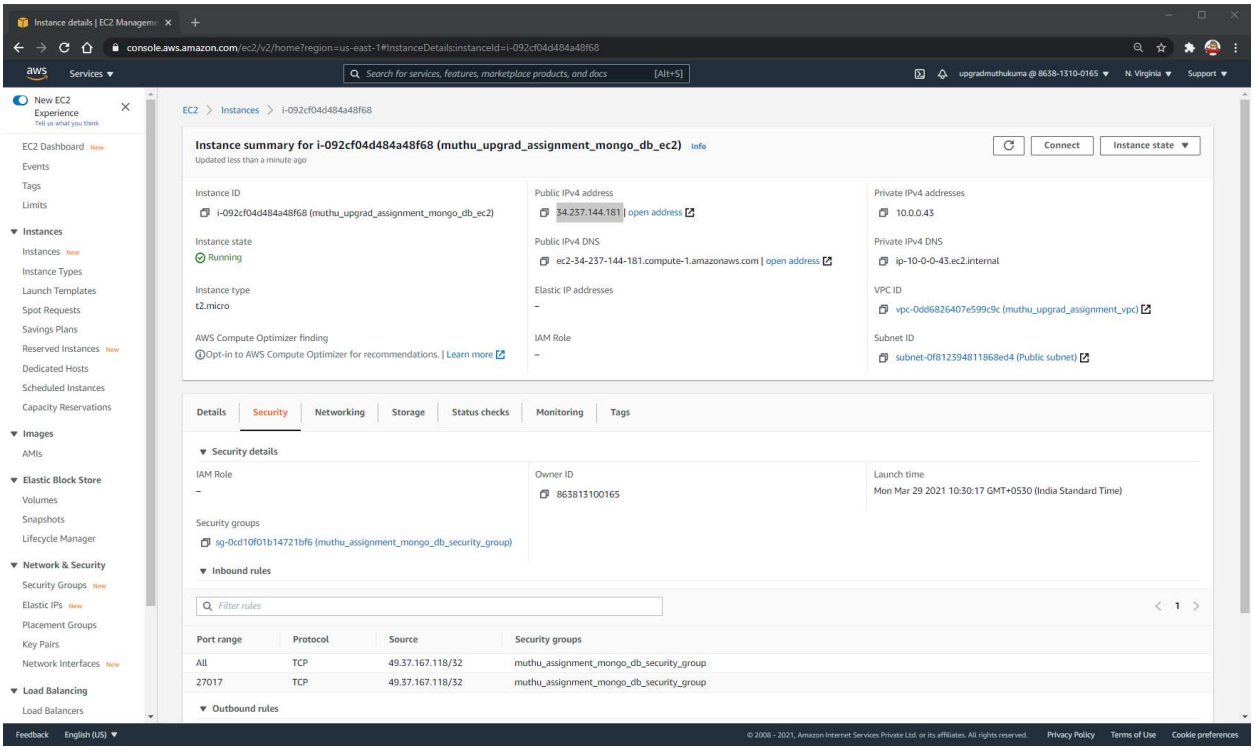
3. Creating EC2



4. Connecting to EC2 from putty & setting up mongo DB



5. Running EC2 instance with mongo DB installed.


Instance details | EC2 Management
console.aws.amazon.com/ec2/v2/home?region=us-east-1#InstanceDetails:instanceId=i-092cf04d484a48f68
Services
Search for services, features, marketplace products, and docs [Alt+S]
upgradmuthukuma @ 8638-1310-0165
N. Virginia
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard
Events
Tags
Limits
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Scheduled Instances
Capacity Reservations
Images
AMIs
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
EC2 > Instances > i-092cf04d484a48f68
Instance summary for i-092cf04d484a48f68 (muthu_upgrad_assignment_mongo_db_ec2) Info
Updated less than a minute ago
Connect
Instance state
Instance ID
i-092cf04d484a48f68 (muthu_upgrad_assignment_mongo_db_ec2)
Public IPv4 address
34.237.144.181 | open address
Private IPv4 addresses
10.0.0.43
Instance state
Running
Public IPv4 DNS
ec2-34-237-144-181.compute-1.amazonaws.com | open address
Private IPv4 DNS
ip-10-0-0-43.ec2.internal
Instance type
t2.micro
Elastic IP addresses
–
VPC ID
vpc-0dd6826407e599c9c (muthu_upgrad_assignment_vpc)
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role
–
Subnet ID
subnet-0f812394811868ed4 (Public subnet)
Details
Security
Networking
Storage
Status checks
Monitoring
Tags
Security details
IAM Role
–
Owner ID
863813100165
Launch time
Mon Mar 29 2021 10:30:17 GMT+0530 (India Standard Time)
Security groups
sg-0cd10f01b14721bf6 (muthu_assignment_mongo_db_security_group)
Inbound rules
Filter rules
1
Port range
Protocol
Source
Security groups
All
TCP
49.37.167.118/32
muthu_assignment_mongo_db_security_group
27017
TCP
49.37.167.118/32
muthu_assignment_mongo_db_security_group
Outbound rules
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences