# Introduction to NoSQL with MongoDB

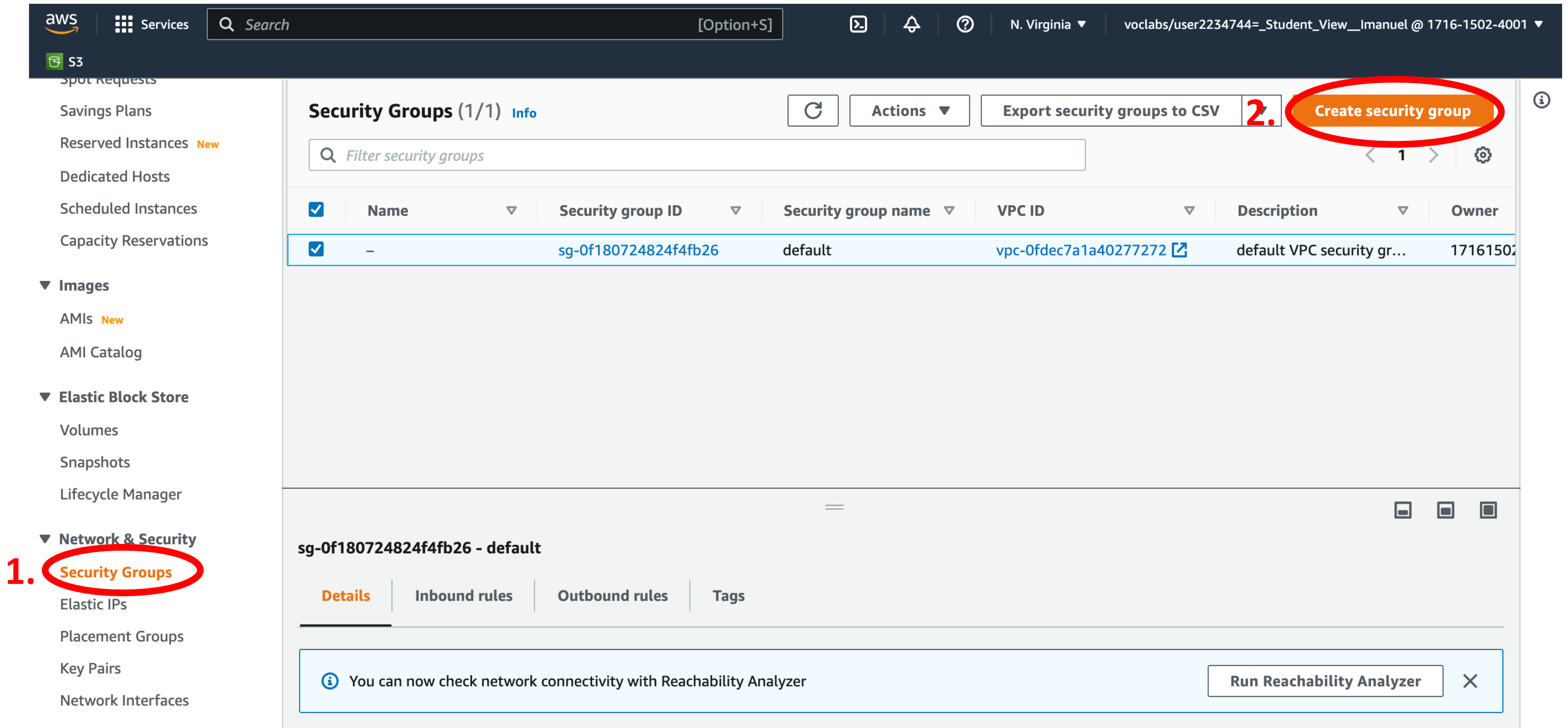
Imanuel Portalatin

## Prerequisites

Launch AWS EC2 instance with class data and dependencies

# Create Security Group (1/3)

- Open the Amazon EC2 console.
- Choose **Security Groups** from the menu to the left of the console.
- Click on **Create Security Group** at the top right of the console.

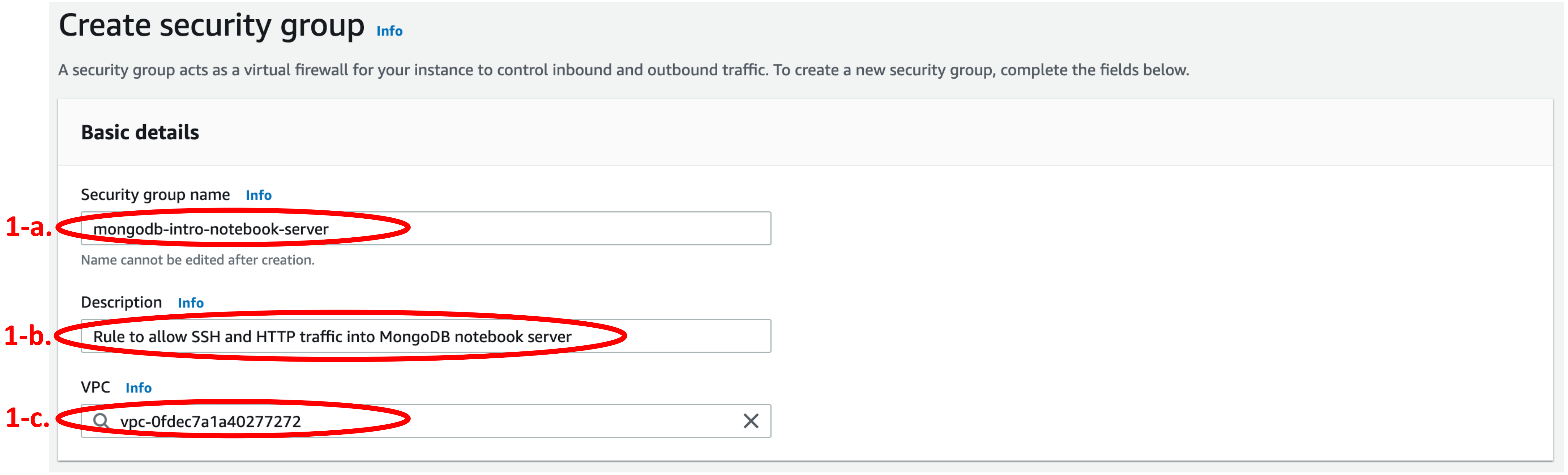


| Name | Security group ID | Security group name | VPC ID | Description | Owner |
|---|---|---|---|---|---|
| – | sg-0f180724824f4fb26 | default | vpc-0fdec7a1a40277272 | default VPC security gr... | 17161502 |

You can now check network connectivity with Reachability Analyzer

## Create Security Group (2/3)

1. On the **Create Security Group** page:
   a. Enter `mongodb-intro-notebook-server` as the **Security Group Name**.
   b. Enter a description for the new security group.
   c. Leave the default entry on the “VPC” field.



Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

Security group name Info

1-a. mongodb-intro-notebook-server

Name cannot be edited after creation.

Description Info

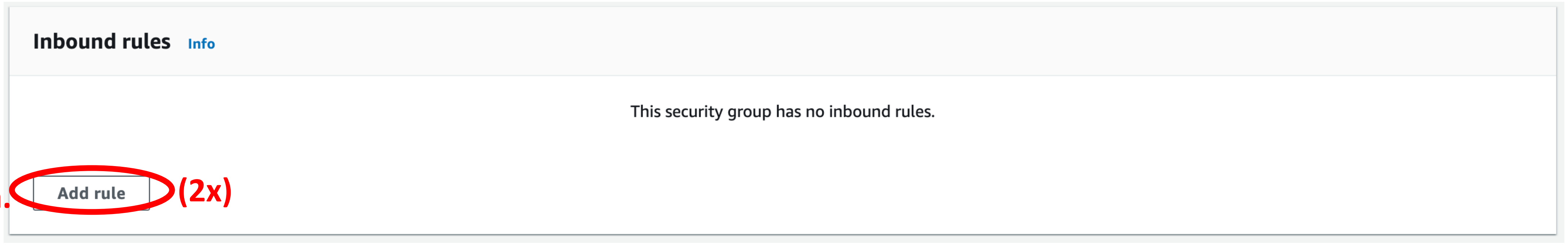
1-b. Rule to allow SSH and HTTP traffic into MongoDB notebook server

VPC Info

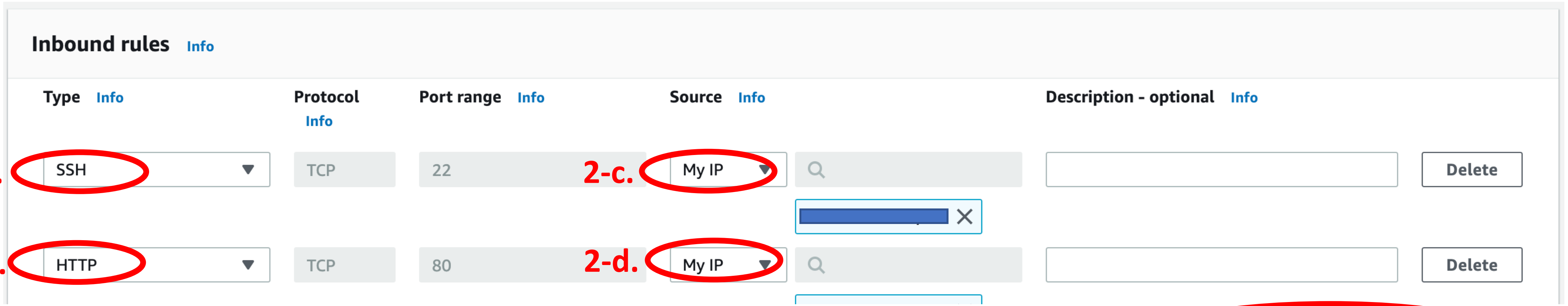
1-c. vpc-0fdec7a1a40277272

# Create Security Group (3/3)

1. Scroll down to the **Inbound rule** section:
   a. Click "Add rule" **twice (2x)** to create two new rule configuration rows.

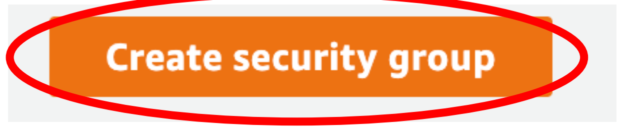


2. Configure the two rules as follows:



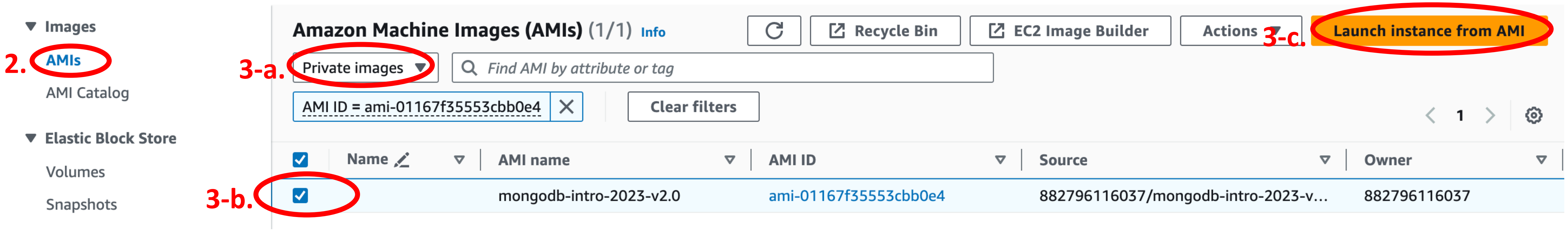
| Type | Protocol | Port range | Source | Description - optional |
|---|---|---|---|---|
| SSH | TCP | 22 | My IP | |
| HTTP | TCP | 80 | My IP | |

3. Click **Create Security Group**.



# EC2 Deployment (1/3)

1. Open the Amazon EC2 console.
2. Select **AMIs** from the **Images** section on the menu to the left of the console.
3. On the **Amazon Machine Image (AMI)** page:
   a. Make sure the “Private images” is selected from the drop-down menu on the upper left.
   b. Click the checkbox on the **“mongodb-intro-2023”** image (ami-00bd19240356b5ebd).
   c. Clock the **Launch Instance from AMI** button.



AMI ID = ami-01167f35553cbb0e4

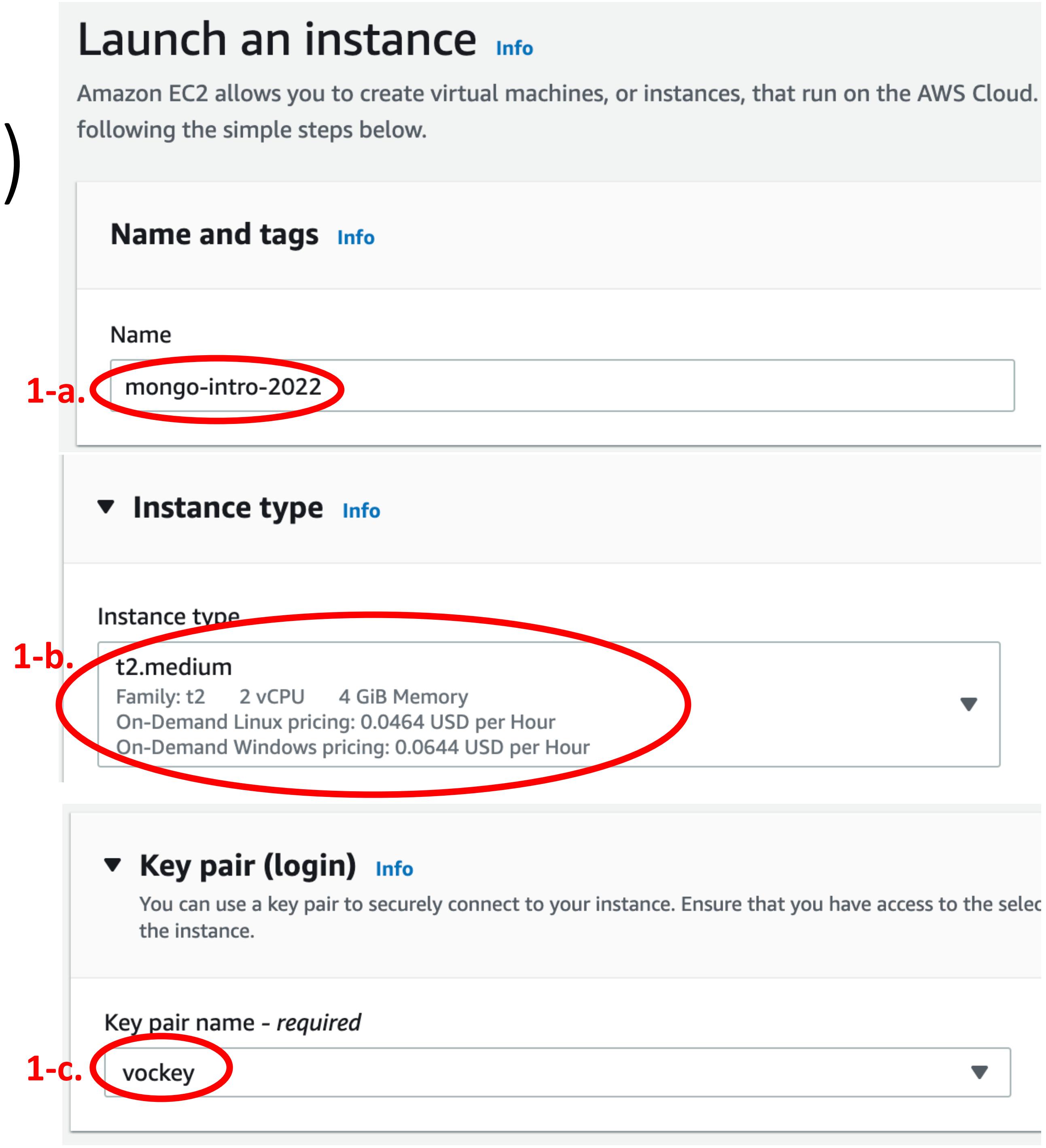
| ☑ | Name | AMI name | AMI ID | Source | Owner |
|---|---|---|---|---|---|
| ☑ | | mongodb-intro-2023-v2.0 | ami-01167f35553cbb0e4 | 882796116037/mongodb-intro-2023-v... | 882796116037 |

# EC2 Deployment (2/3)

1. On the **Launch an Instance** page:
   a. Enter “monogb-intro-2022” into the “Name” field under **Name and tags**.

   b. Scroll down to the **Instance Type** section and select the **`t2.medium`**.

   c. Select “vockey” on the “Key pair name” field under **Key pair (login)**.



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. following the simple steps below.

Name and tags Info

Name

1-a. mongo-intro-2022

Instance type Info

Instance type

1-b. t2.medium
Family: t2 2 vCPU 4 GiB Memory
On-Demand Linux pricing: 0.0464 USD per Hour
On-Demand Windows pricing: 0.0644 USD per Hour

Key pair (login) Info

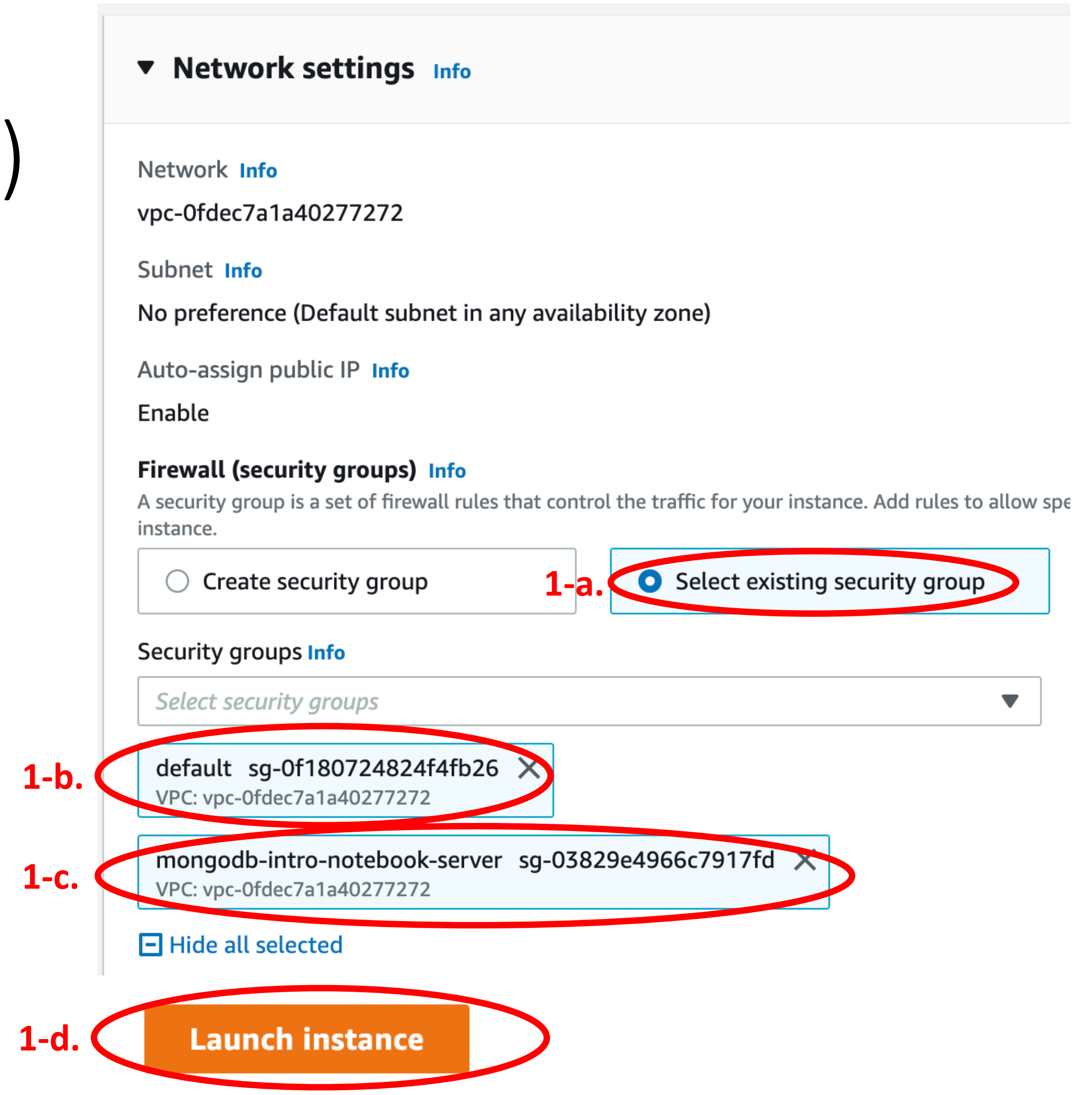
You can use a key pair to securely connect to your instance. Ensure that you have access to the selec the instance.

Key pair name - *required*

1-c. vockey

# EC2 Deployment (3/3)

1. On the **Network Settings** section:
   a. Choose “Select existing security group” under **Firewall (security groups)**.
   b. Expand the “Security groups” drop-down list and select the “default” group.
   c. Expand the drop-down list again and select the “mongo-intro-notebook-server” group. (After selecting both groups, click “Show all selected (+1)” to verify the groups as in the picture.)
   d. Click on **Launch Instance** under the **Summary** page to the right.



▼ **Network settings** Info

Network Info
vpc-0fdec7a1a40277272

Subnet Info
No preference (Default subnet in any availability zone)

Auto-assign public IP Info
Enable

**Firewall (security groups)** Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow spe
instance.

○ Create security group [✓ ● Select existing security group]

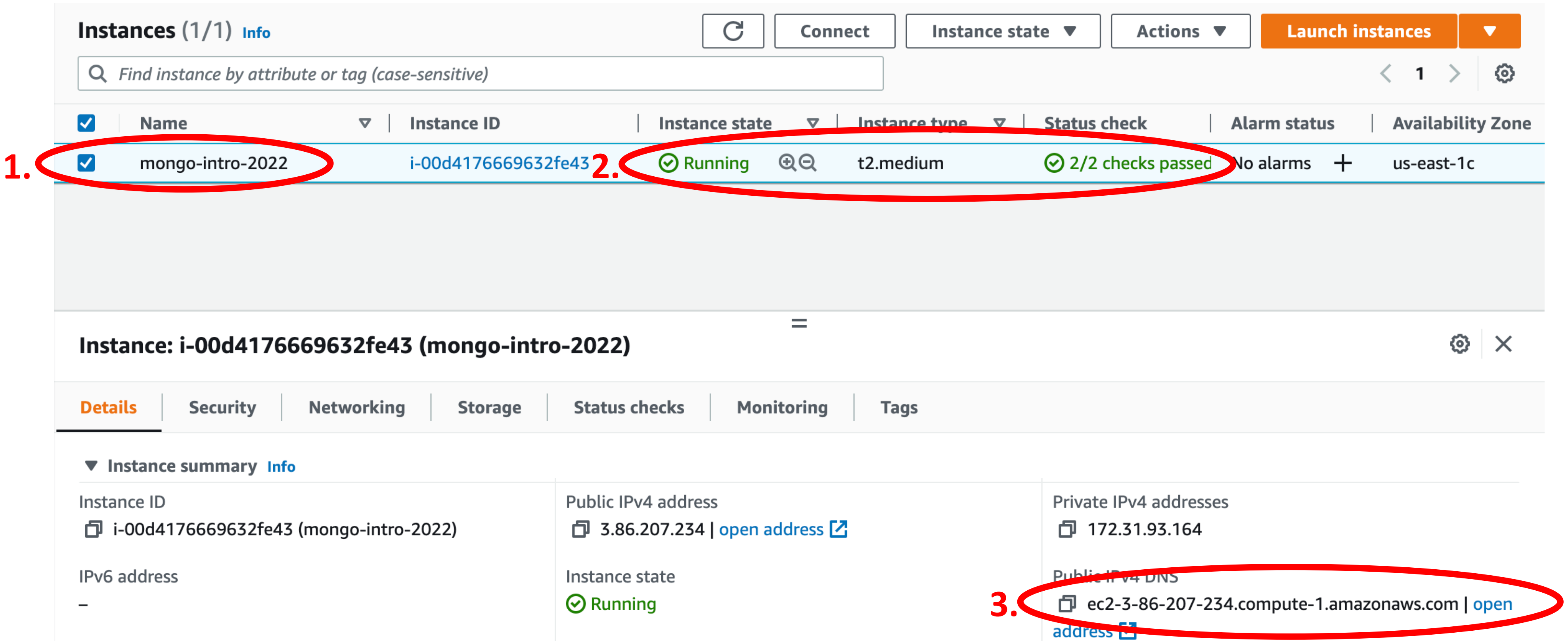
Security groups Info
Select security groups ▼

default sg-0f180724824f4fb26
VPC: vpc-0fdec7a1a40277272

mongodb-intro-notebook-server sg-03829e4966c7917fd
VPC: vpc-0fdec7a1a40277272

# Verify EC2 Deployment (1/2)

- On the **Instances** page of the Amazon EC2 console:
  1. Select the "mongodb-intro-2022" instance from the **Instances** list.
  2. Verify that the instance status is "Running" and that Status Checks have passed. (Status checks will take a few minutes to complete. Be patient!)
  3. Copy the **Public DNS (IPv4)** under the "Details" tab below.



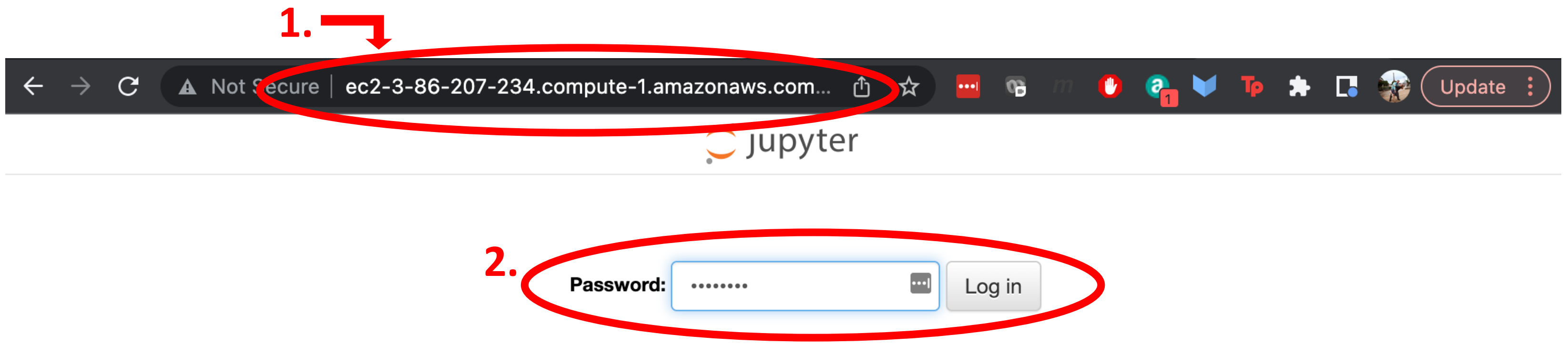
| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone |
|---|---|---|---|---|---|---|
| mongo-intro-2022 | i-00d4176669632fe43 | Running | t2.medium | 2/2 checks passed | No alarms | us-east-1c |

Instance: i-00d4176669632fe43 (mongo-intro-2022)

| Instance ID | Public IPv4 address | Private IPv4 addresses |
|---|---|---|
| i-00d4176669632fe43 (mongo-intro-2022) | 3.86.207.234 \| open address | 172.31.93.164 |
| **IPv6 address** | **Instance state** | **Public IPv4 DNS** |
| – | Running | ec2-3-86-207-234.compute-1.amazonaws.com \| open address |

# Verify EC2 Deployment (2/2)

- On a web browser window:
  1. Paste the **Public DNS** of the instance on the address bar and press <Enter>.
  2. Type the "P@ssword" in the **Password** field and click "Log In".



3. Once logged into Jupyter, you should able to successfully run all cells in the provided "mongodb_intro.ipynb" notebook.
   (NOTE: The PyMongo installation code in cell #1 is provided for reference. PyMongo is already installed on this instance and does not need to be installed again to run the rest of the notebook.)

# Lab 1

Access MongoDB through Jupyter notebook

# Query MongoDB with Jupyter

Use the Jupyter notebook server you deployed earlier to:

1. Install and initialize PyMongo in Jupyter:
   a. Install the PyMongo Python package from a Jupyter notebook
   b. Import PyMongo package and get MongoClient for the notebook
2. Connect to MongoDB server on the `‘database’` host using standard port
3. Connect to the `‘test’` database on the server from Step 2.
4. Get a list of the collections contained in the `‘test’` database
5. Query a single document from the `‘restaurants’` collection and save the return value to a variable
6. Use `pprint` to display document from Step 5. as “nicely” formatted JSON

## Lab 2

Create documents and issue simple queries using mongo shell

## Part A: Create a simple document

- Select the test database and Jupyter to insert the following document into the `users` collection:

```
{
        name: “sue”,
        age: 26,
        status: “pending”
}
```

  - Hint: If the collection does not exist, it will be created on first insert
- Query all documents in the `users` collection to make sure your insert was successful
  - Hints:
    - On mongo shell you can use the `pretty()` cursor modifier to format output for easier reading.
    - In Jupyter notebooks, use the `pprint` Python package to display a single document as nicely formatted JSON.
    - For multiple results in Jupyter, iterate through cursor and print each one separately.

# Part B: Create complex documents

- Use Jupyter to insert a new document into the `users` collection for the following user:

| | | |
|---|---|---|
| **Name** | | Ned McDodd |
| **Address** | **Street** | 1 Courthouse Drive |
| | **City** | Whoville |
| | **State** | New Seuss |
| | **Zip** | 11111 |

  - Hint: Make the value of the `address` field an <u>object</u> of the form:

    { street: <value>, city: <value>, state: <value>, zip: <value> }

  - Note: You do <u>not</u> have to include an `age` or `status` field; MongoDB documents on the same collection can have different schemas!
- Query all documents in the `users` collection to make sure your insert was successful

## Part C: Create documents with arrays

- Use Jupyter to insert a document for a user as follows:
  - Name of your choosing (preferably a Who from Whoville!)
  - Make the status field an array with values “approved” and “account pending”
- Query all documents in the `users` collection to make sure your insert was successful

# Part D: Use cursor modifiers

- Use Jupyter to get the number of documents in the `restaurants` collection
  - Hint: In mongo shell, this is the same as querying all documents in the collection but with the `count()` cursor modifier added; however, in Jupyter, we must use the PyMongo `count_documents()` method since `count()` is not defined for the cursor object. Also, `count_documents()` requires a filter argument, so we pass an empty object (`{}`) to count all documents. See: https://pymongo.readthedocs.io/en/stable/tutorial.html#counting
- Use mongo shell to query the first 5 documents in the `restaurants` collection

# Lab 3

Query MongoDB documents using conditions on fields, arrays and objects

## Part A: Query on equality conditions

- Find the document in the `users` collection for Ned McDodd, Mayor of Whoville
- Find all documents in the `users` collection that have a status of “pending” <u>or</u> “approved”
- Find all users that live in Whoville

## Part B: Queries with operators

- Use the `restaurants` collection to answer the following questions:
  1. How many of the restaurants are Pizzerias?
  2. How many restaurants from question 1 have at least one grade lower than C, or a score greater than 19, regardless of grade?
  3. How many restaurants from question 1 have at least one A with score 10 or less?

# Lab 4

Customize MongoDB query results using projections

## Customize query results

- Add a projection to the queries for questions 2 and 3 of Lab 2 Part B to customize the results as follows:
  - Instead of the count, display only the name and address of the first five (5) results

## Lab 5

Find restaurants using text and geospatial queries

## Part A: Find restaurants using text search

- Use the `restaurants` collection to answer the following questions:
  1. How many restaurants have names that contain the words “Vinny”, “Vinnie” or “Famous”?
  2. How many of the restaurants from question 1 are Pizzerias?

# Part B: Find restaurants by location

- Create `2dsphere` indexes as follows:
  - On the `restaurants` collection use the `coord` field of the `address` object
    - Hint: You may have to use quotes when using “dot notation” to define field names
  - On the `neighborhoods` collection use the `geometry` GeoJSON object
- How many pizzerias are there in Little Italy?
  - Hint: In mongo shell, you can use Javascript syntax to store a document returned from a query for use in future queries:

```
var my_neighborhood = db.neighboorhoods.findOne({ name: /foo/ })
```

  - Once you find the target neighborhood, you can use the GeoJSON geometry on geospatial queries against the `restaurants` data set

# Lab 6

Update MongoDB documents

## Update Users

Update documents in the `users` collection as follows:

A. Replace the `status` field on the document with name “sue” with an array with the value `[ “approved” ]`
B. Make sure the `status` array of all documents has the value “approved”
C. Change the zip code for Ned McDodd, Mayor of Whoville, to 11112
D. Delete all documents except for those with name “sue” and “Ned McDodd”

# Lab 7

Build a MongoDB aggregation pipeline

# Aggregate restaurants health grades

- Build an aggregation pipeline to find the proportion of restaurants with “bad” (I.e. 19.5 or less) health inspection scores for each borough