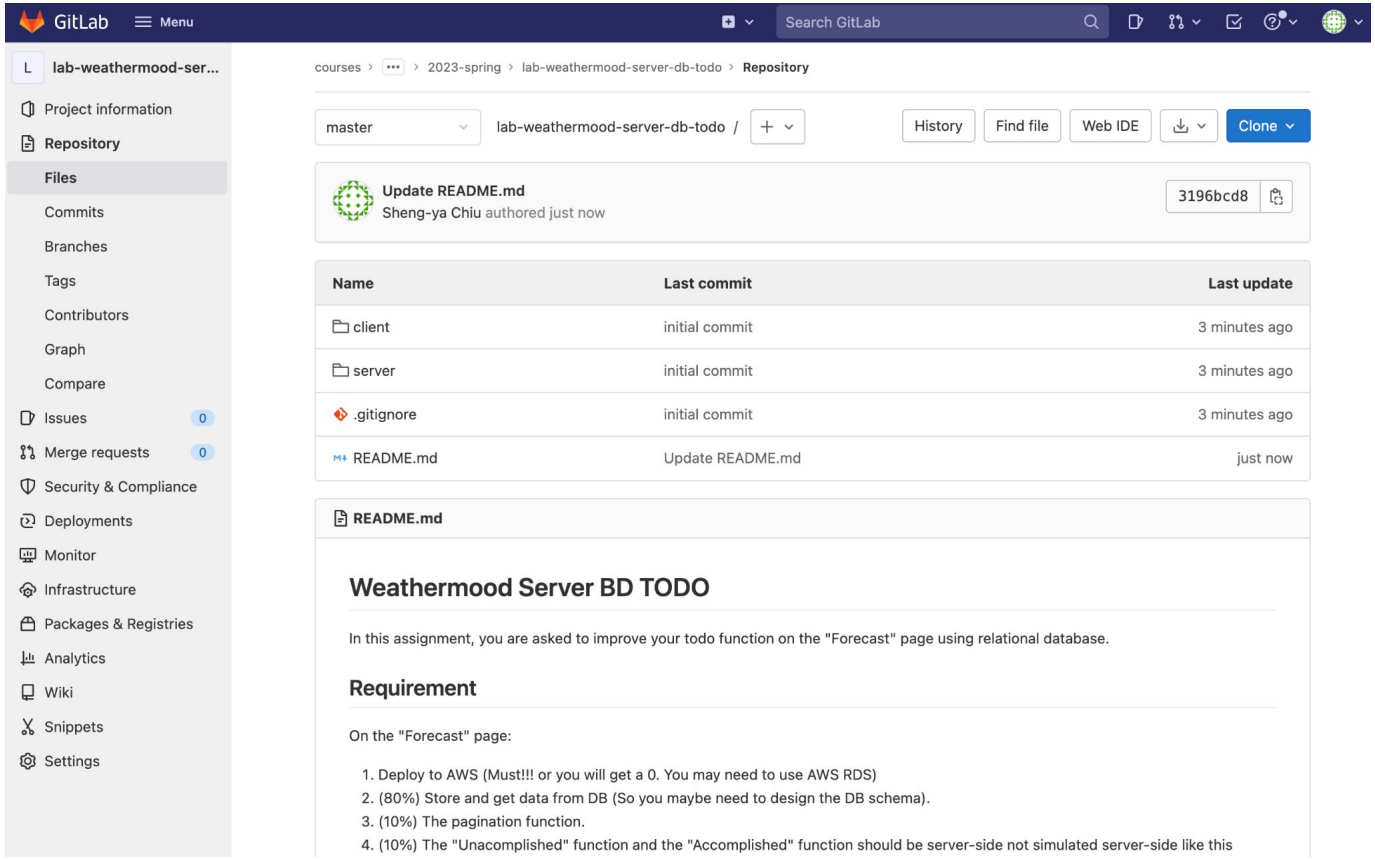
Lab 7
AWS RDS

Software Studio
Datalab, CS, NTHU
2023

Gitlab

lab-weathermood-server-db-todo



From local - file - db system

```
function list(searchText = '') {
    return new Promise((resolve, reject) => {
        if (!fs.existsSync('data-posts.json')) {
            fs.writeFileSync('data-posts.json', '');
        }

        fs.readFile('data-posts.json', 'utf8', (err, data) => {
            if (err) reject(err);

            let posts = data ? JSON.parse(data) : [];
            if (posts.length > 0 && searchText) {
                posts = posts.filter(p => {
                    return p.text.toLowerCase().indexOf(searchText.toLowerCase()) !== -1
                });
            }
            resolve(posts);
        });
    });
}
```

branch file

```
export function listPosts(searchText = '', start) {
    let url = `${postBaseUrl}/posts`;
    let query = [];
    if (searchText)
        query.push(`searchText=${searchText}`);
    if (start)
        query.push(`start=${start}`);
    if (query.length)
        url += '?' + query.join('&');

    console.log(`Making GET request to: ${url}`);

    return axios.get(url).then(function(res) {
        if (res.status !== 200)
            throw new Error(`Unexpected response code: ${res.status}`);

        return res.data;
    });
}
```

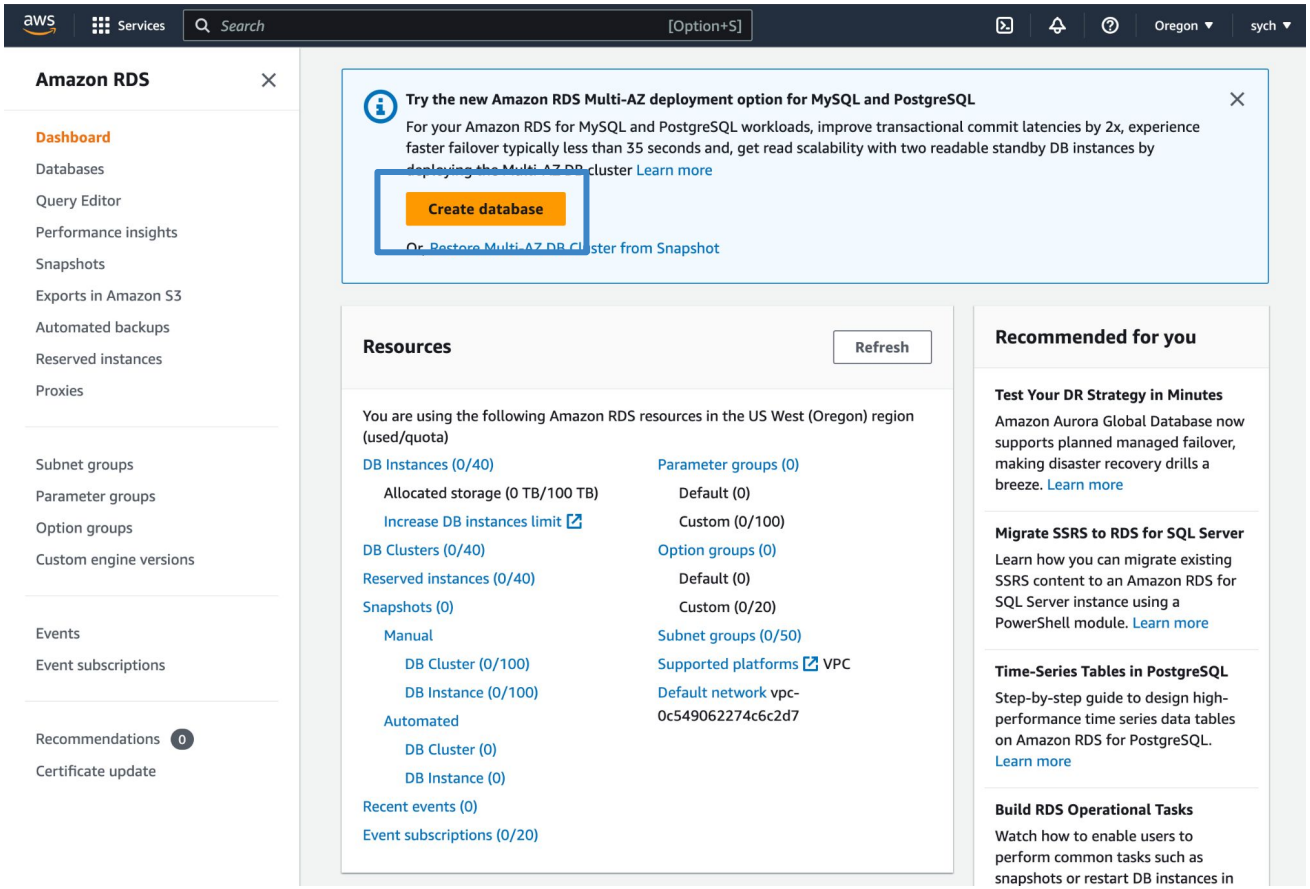
branch db

Outline

1. AWS RDS (relational database service)
2. EB / RDS connection
3. Setup weathermood db in RDS
4. Application setting and deploy

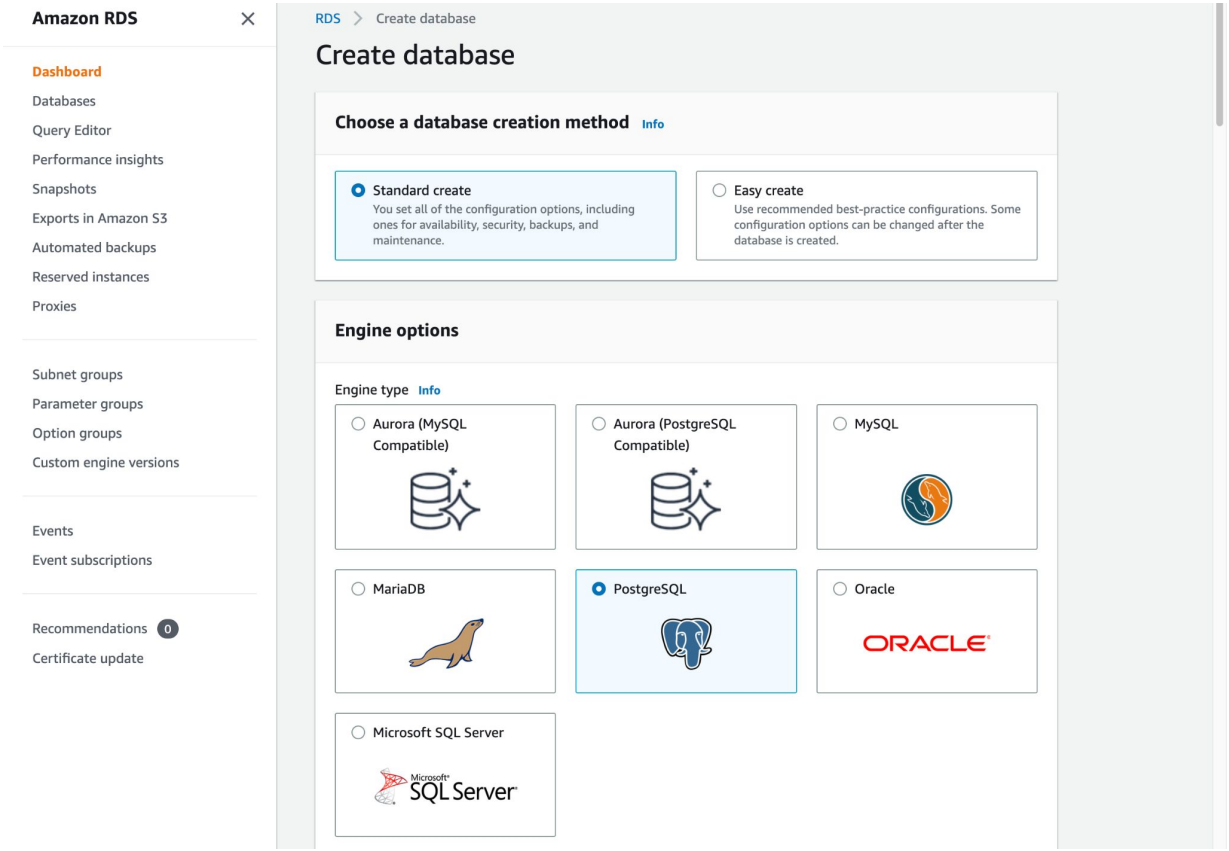
Find RDS

Click create database



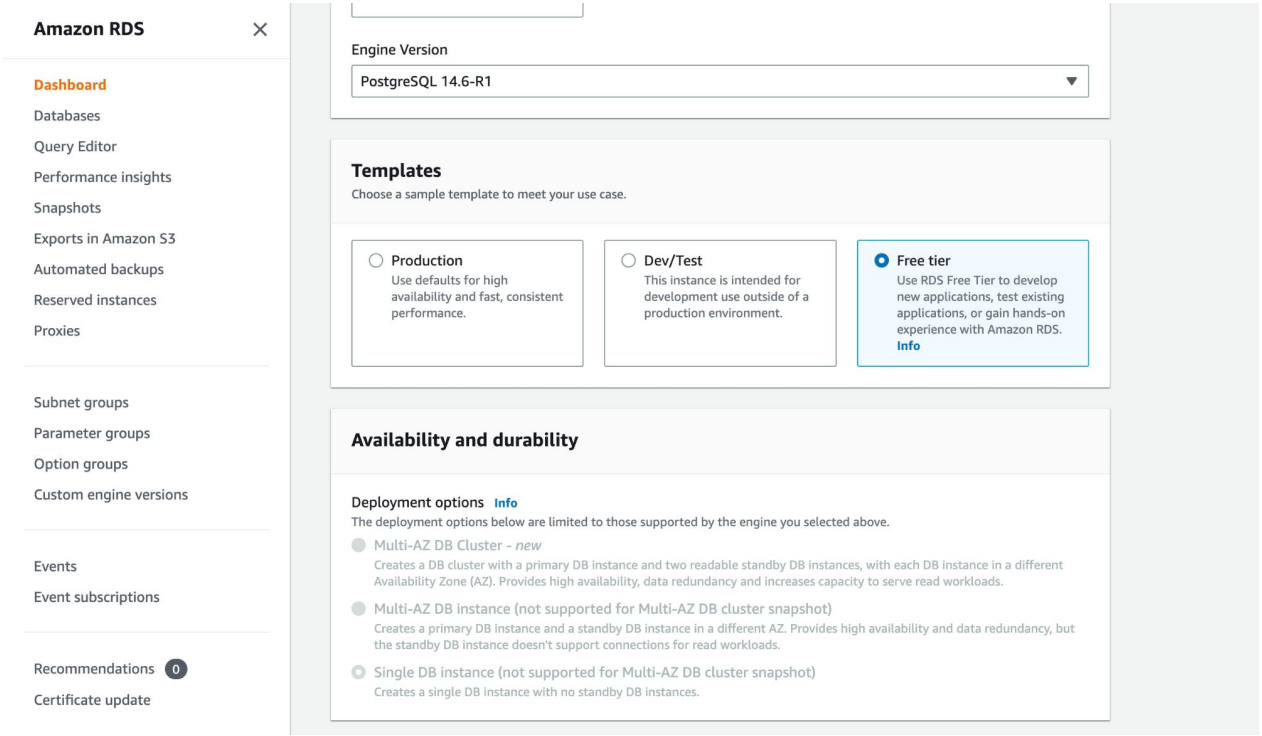
Create database

Standard create; PostgreSQL



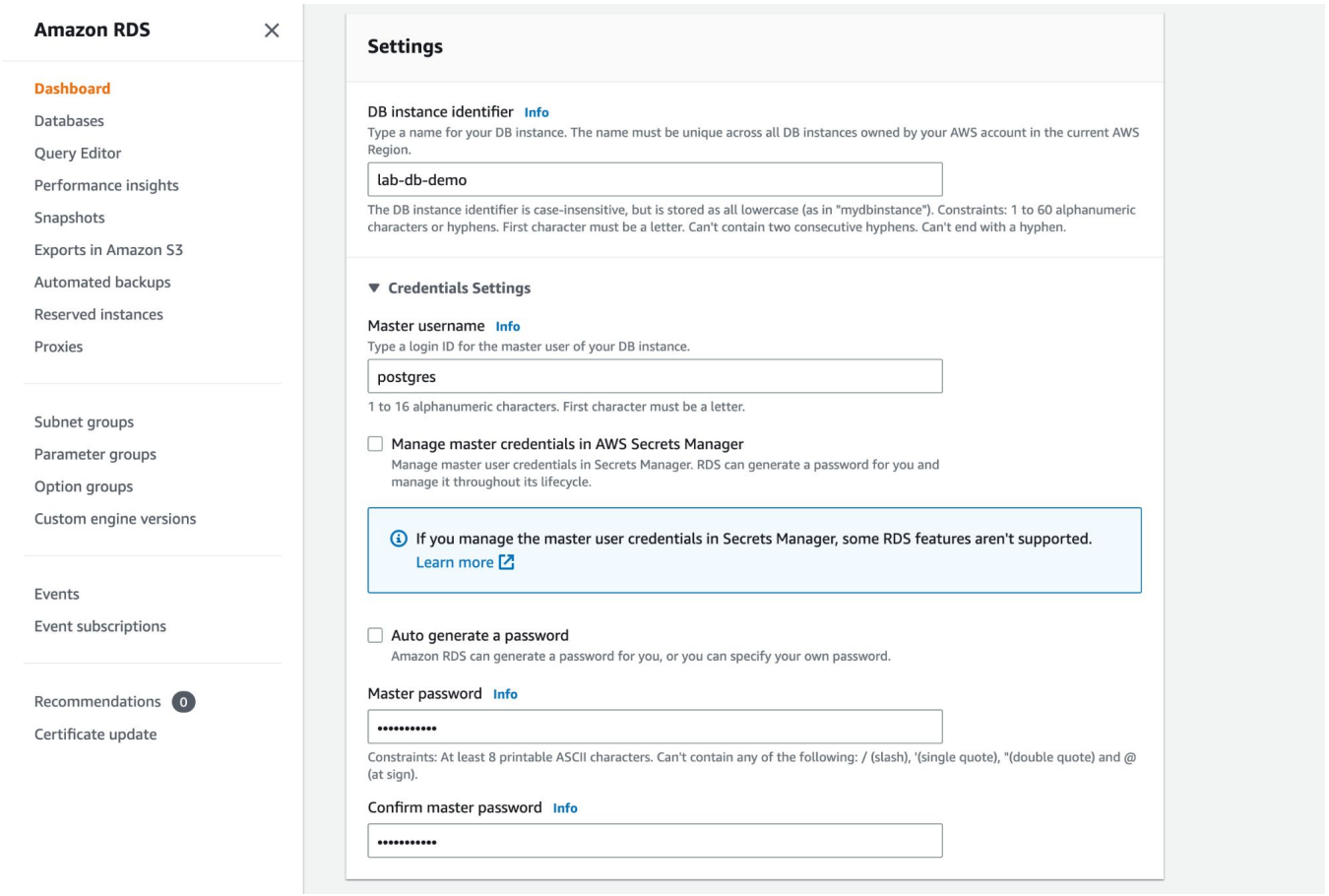
Create database

Engine version (default); Free tier



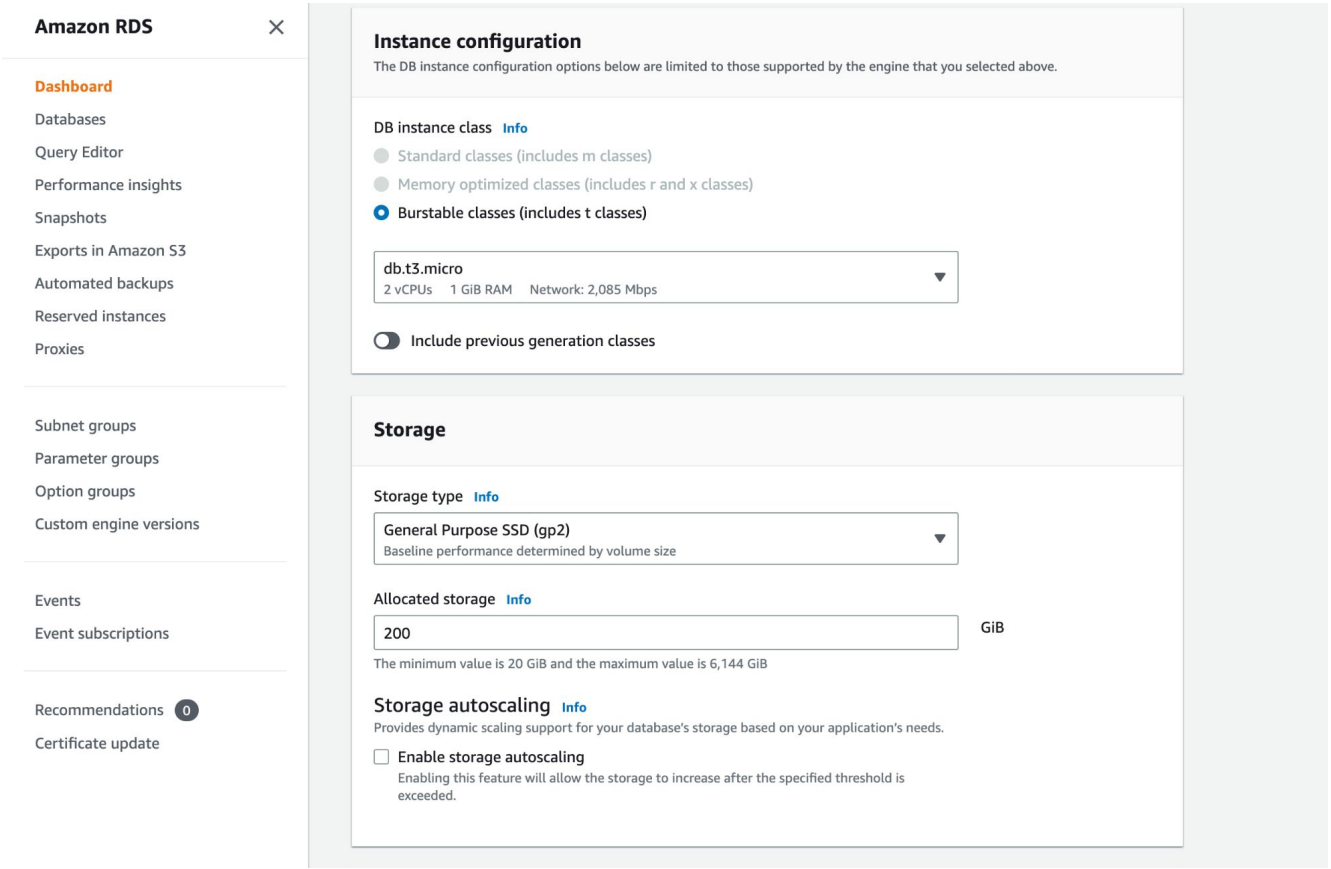
Create database

DB instance (name is up to you);
Master username & password (up to you too but must keep it)



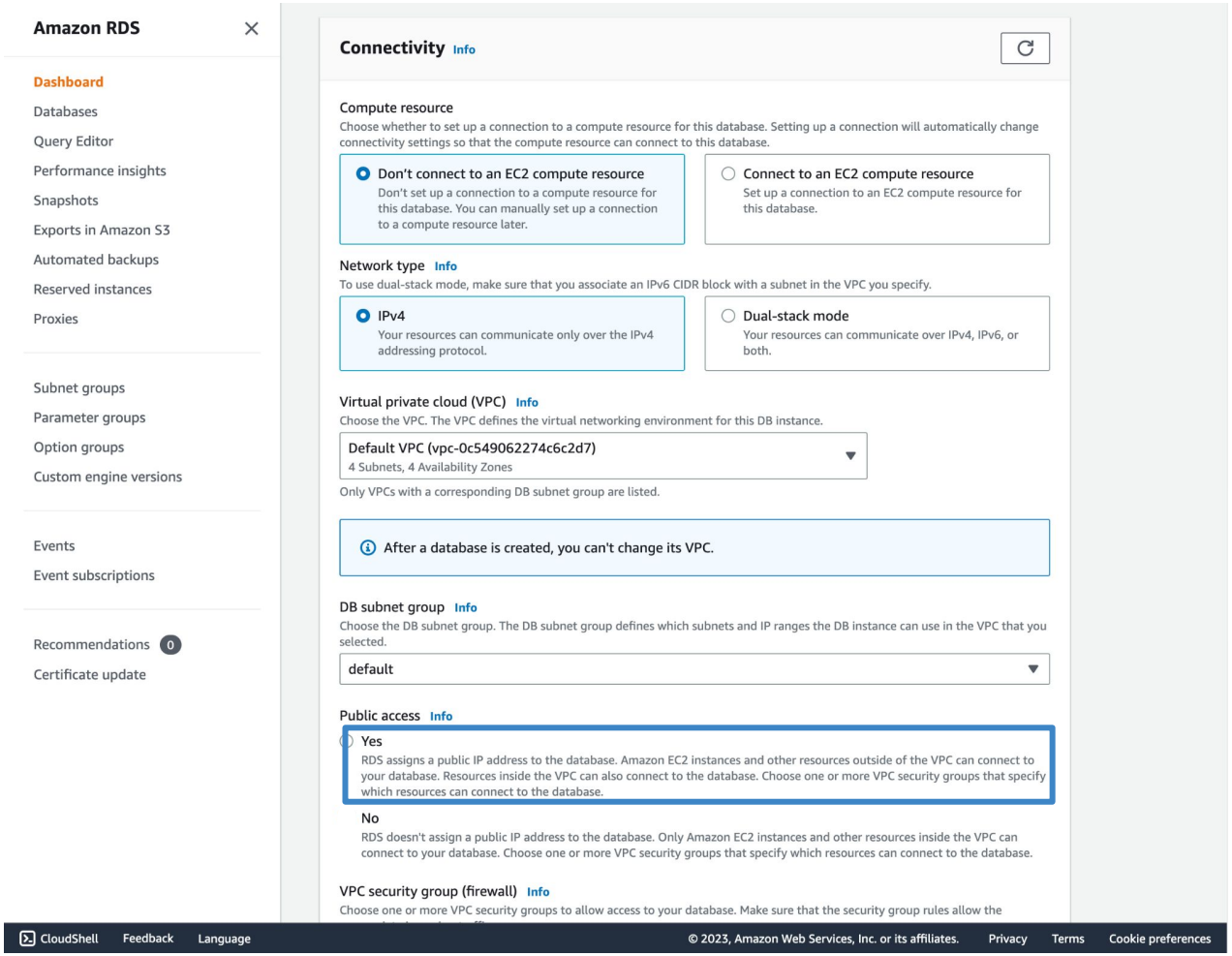
Create database

Default; Uncheck “Enable storage autoscaling”



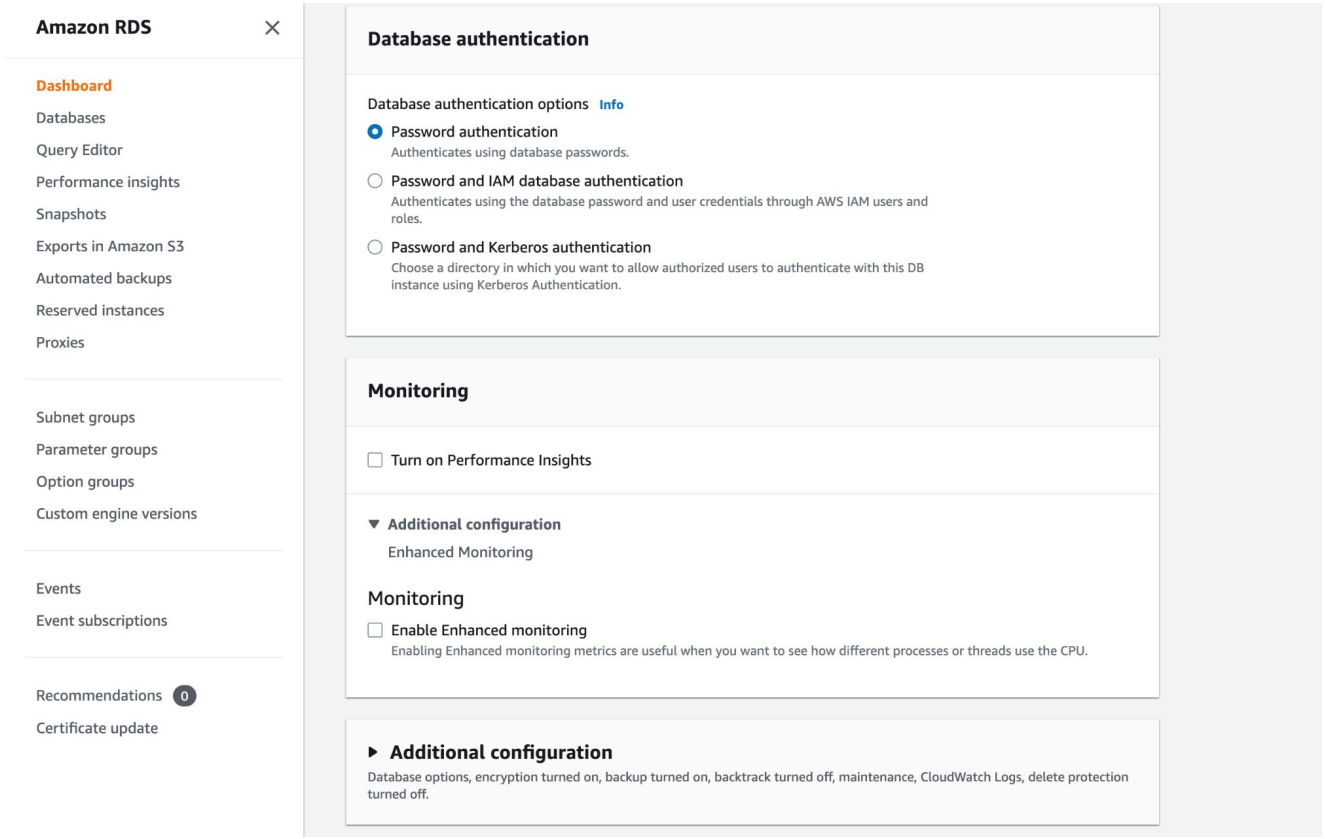
Create database

Public access Yes; other default



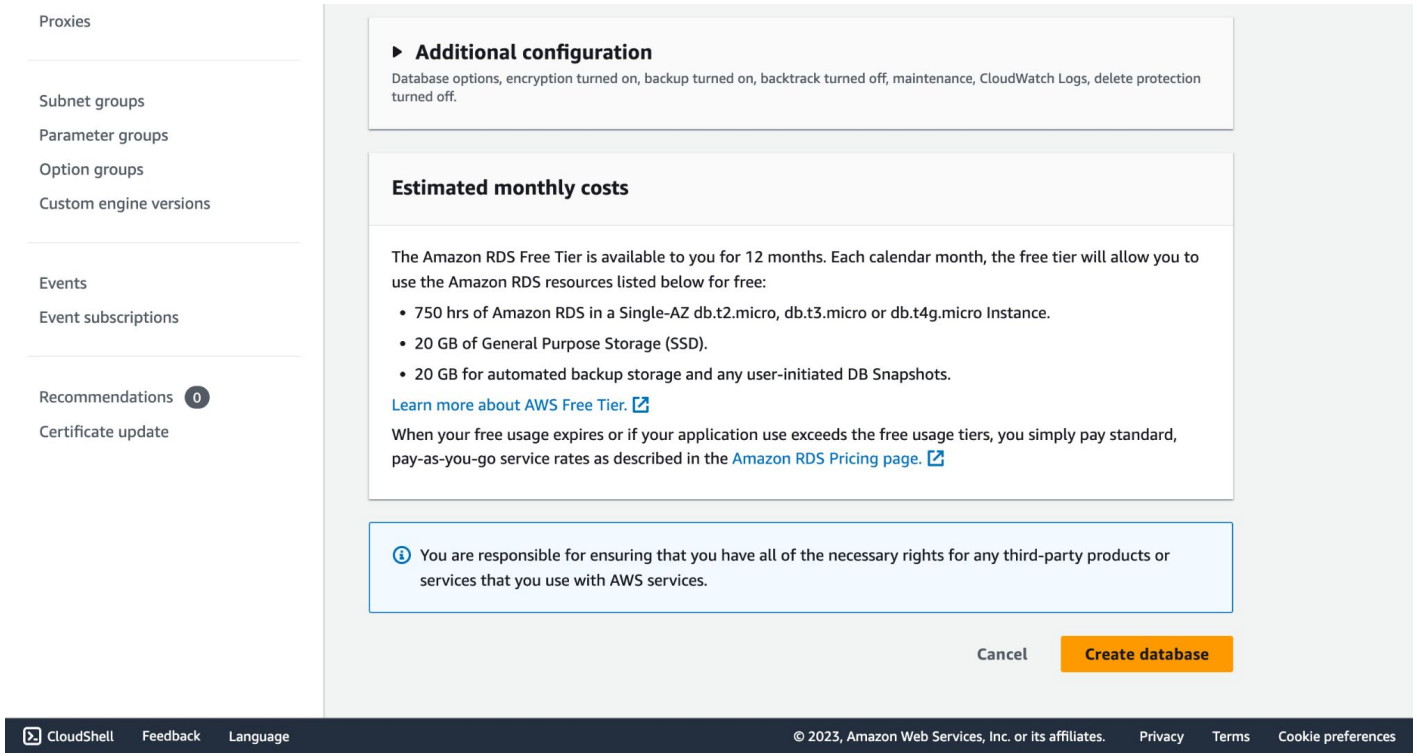
Create database

Password authentication

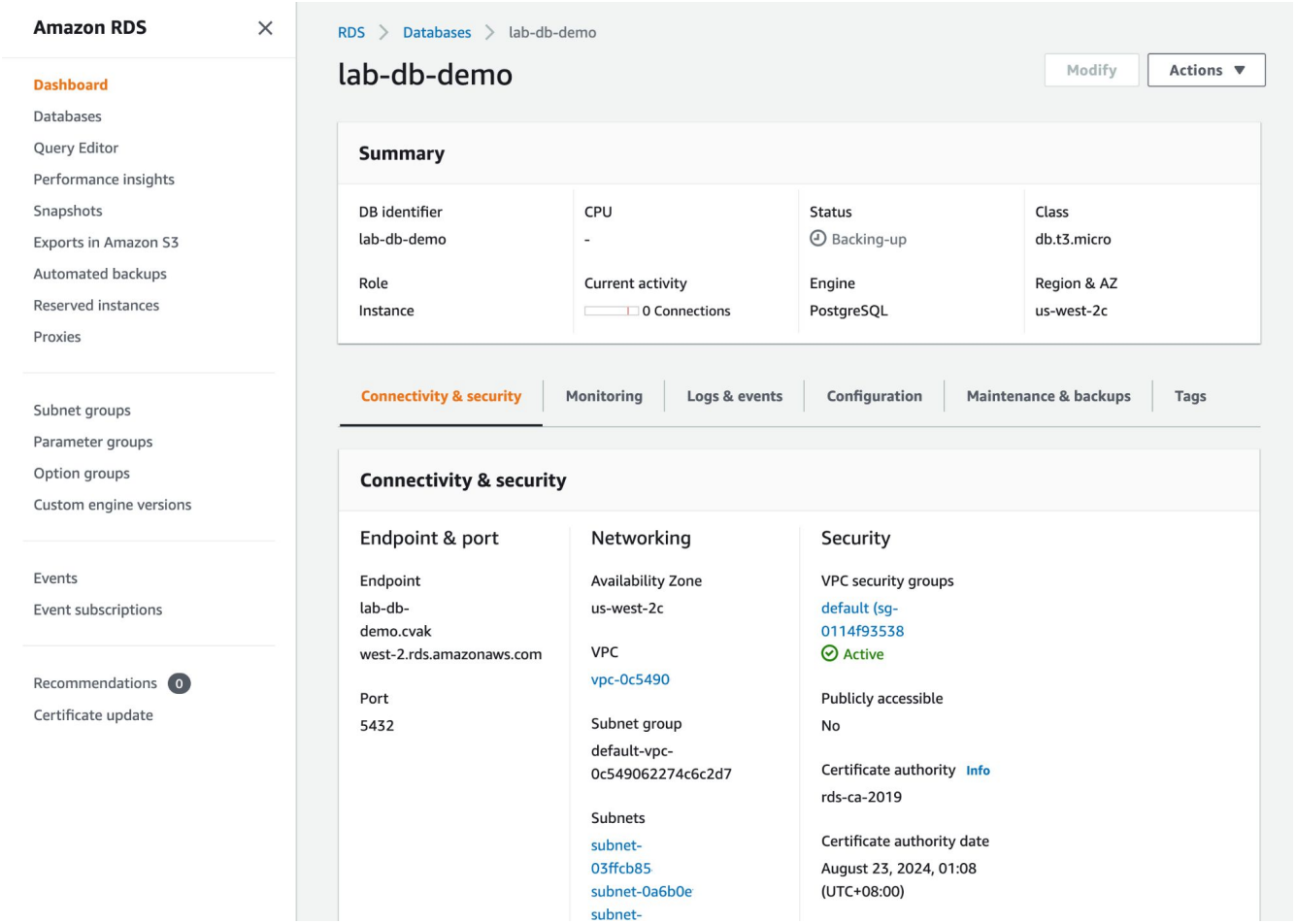


Create database

Make sure it is free tier; create database



Database

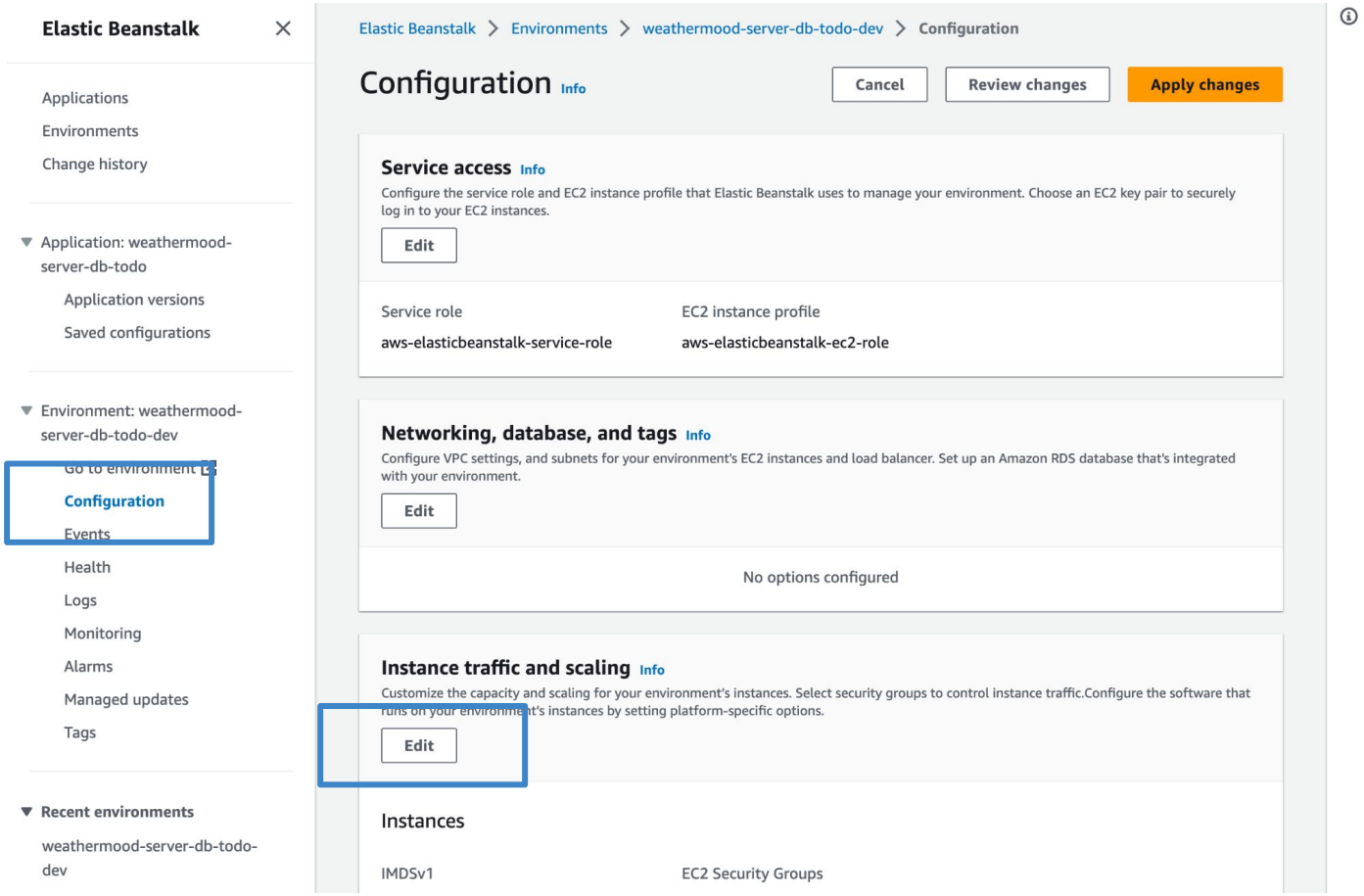

Amazon RDS
Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
Subnet groups
Parameter groups
Option groups
Custom engine versions
Events
Event subscriptions
Recommendations 0
Certificate update
RDS > Databases > lab-db-demo
lab-db-demo
Modify
Actions
Summary
DB identifier
lab-db-demo
CPU
-
Status
Backing-up
Class
db.t3.micro
Role
Instance
Current activity
0 Connections
Engine
PostgreSQL
Region & AZ
us-west-2c
Connectivity & security
Monitoring
Logs & events
Configuration
Maintenance & backups
Tags
Connectivity & security
Endpoint & port
Endpoint
lab-db-demo.cvak west-2.rds.amazonaws.com
Port
5432
Networking
Availability Zone
us-west-2c
VPC
vpc-0c5490
Subnet group
default-vpc-0c549062274c6c2d7
Subnets
subnet-03ffcb85
subnet-0a6b0e
subnet-
Security
VPC security groups
default (sg-0114f93538
Active
Publicly accessible
No
Certificate authority Info
rds-ca-2019
Certificate authority date
August 23, 2024, 01:08 (UTC+08:00)

Outline

1. AWS RDS (relational database service)
2. EB / RDS connection
3. Setup weathermood db in RDS
4. Application setting and deploy

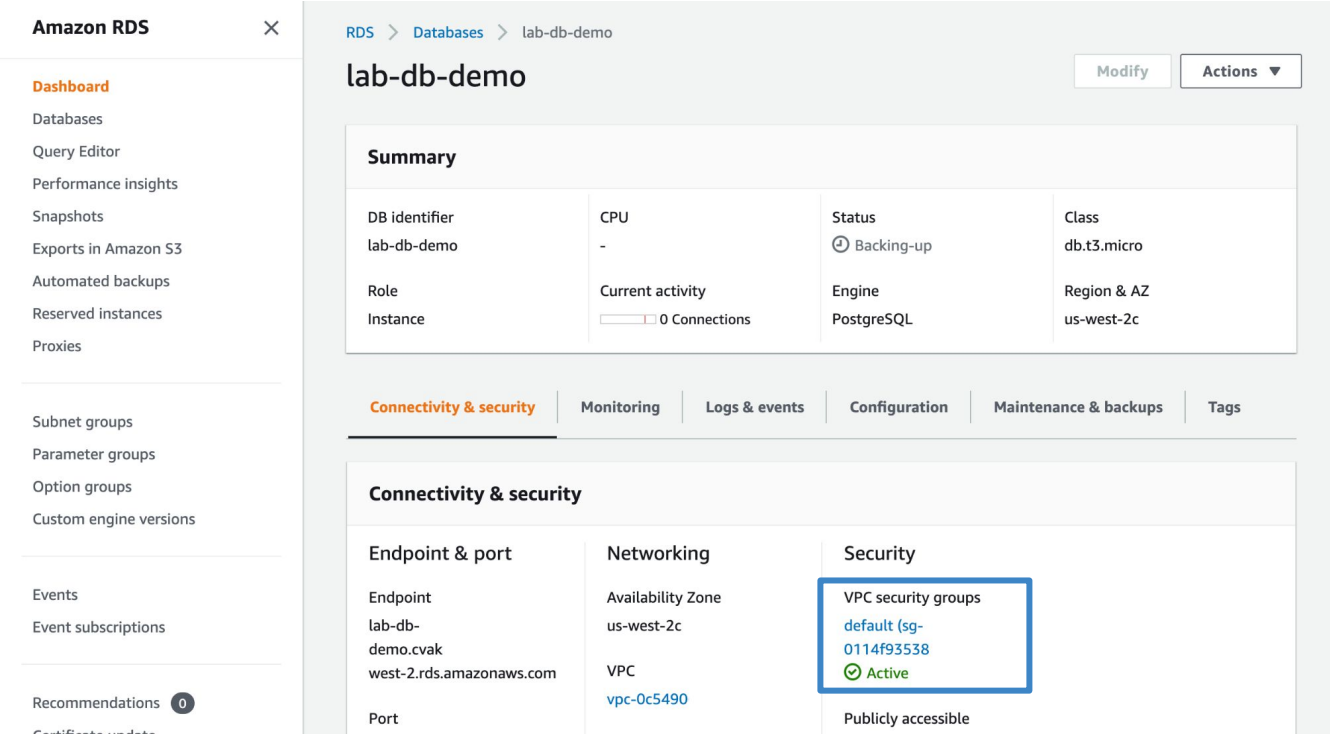
EB + RDS

Find EB - configuration - instances traffic edit



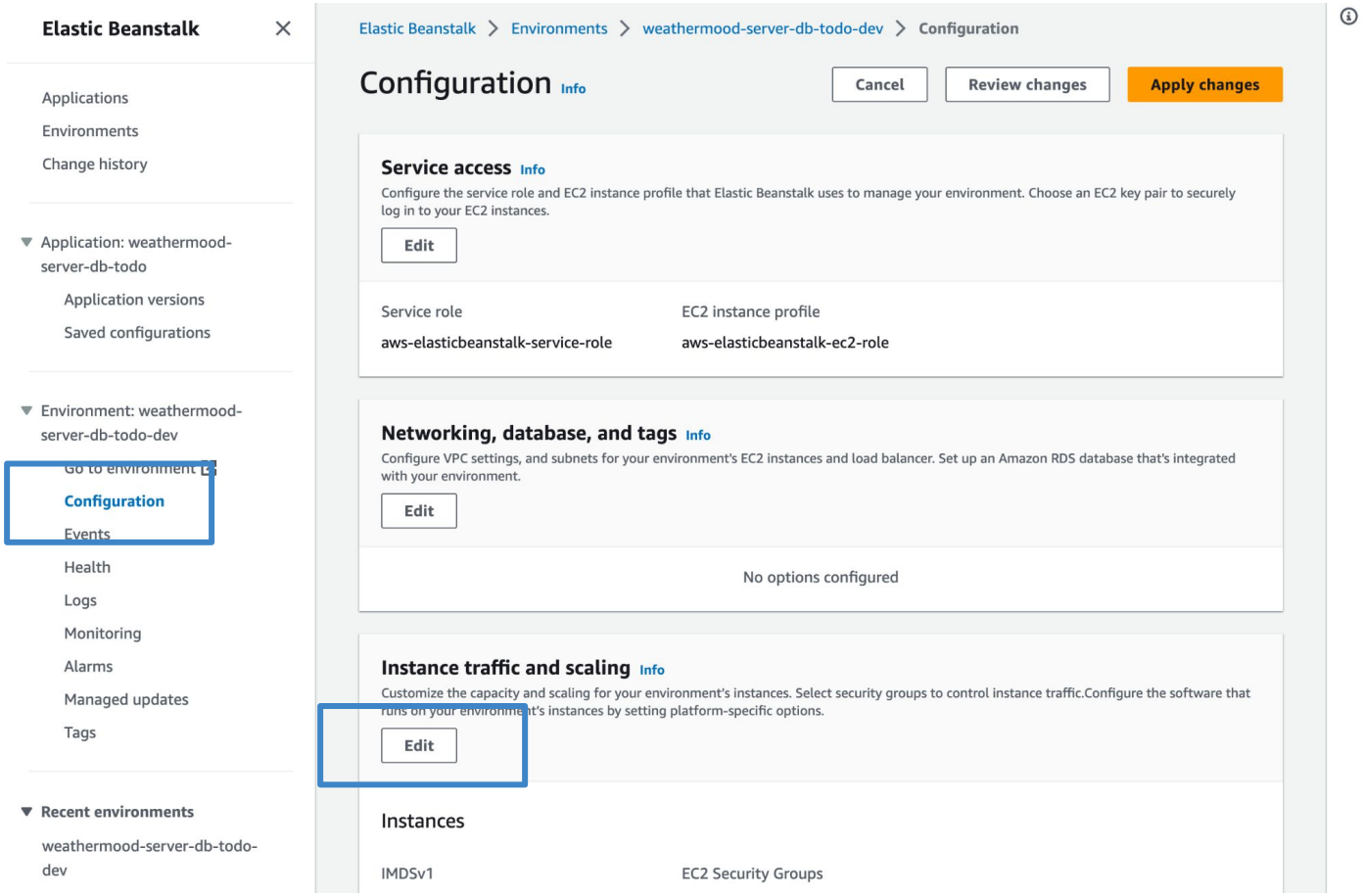
Database

Setup RDS security groups to allow ingress from machines in the same group. Go check out your DB security group first.



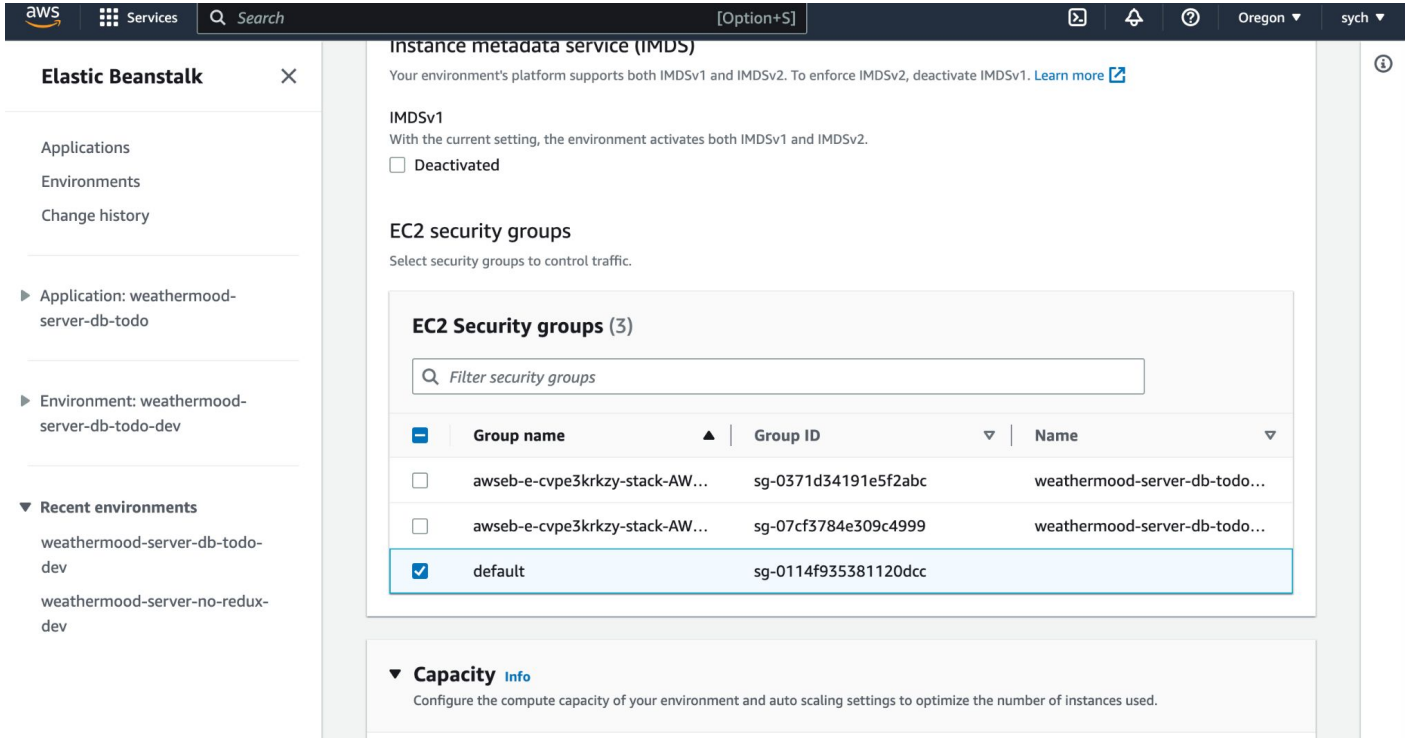
EB + RDS

Find EB - configuration - instances traffic edit



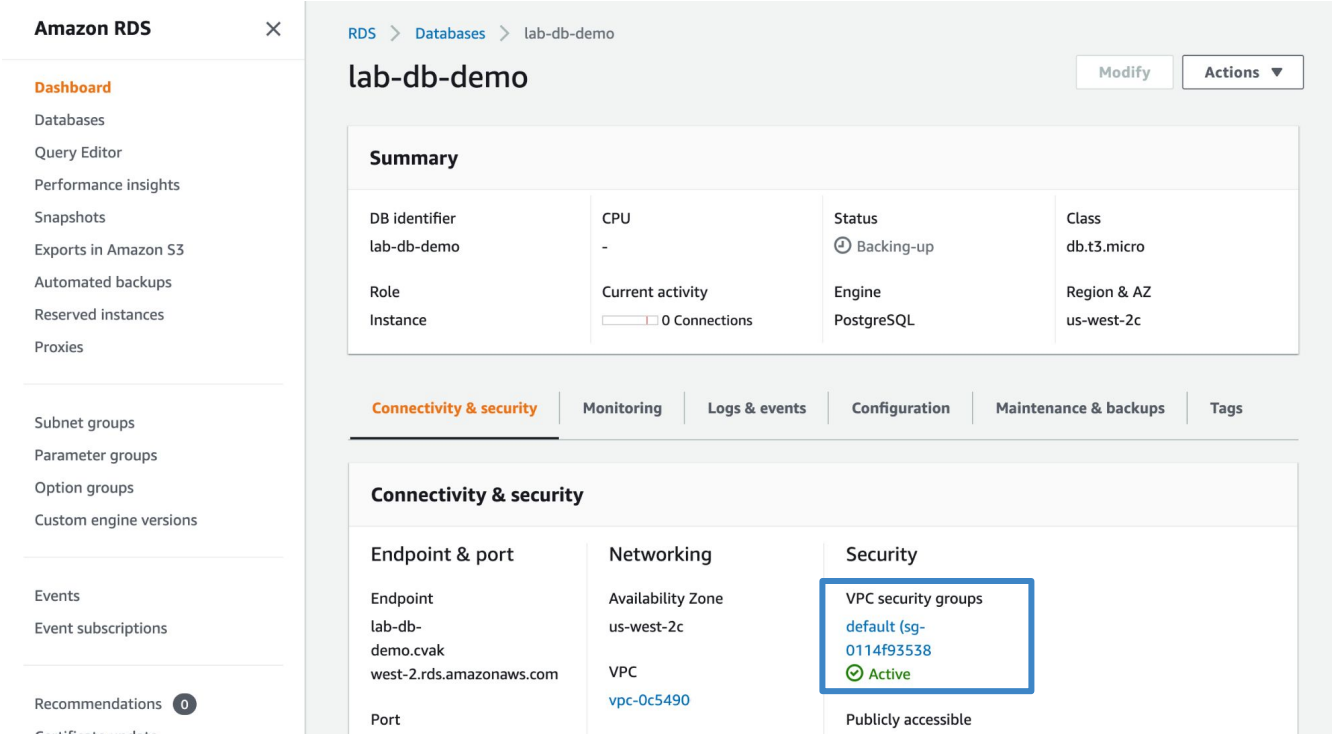
EB + RDS

Click the security group from RDS then apply (button is at the bottom)



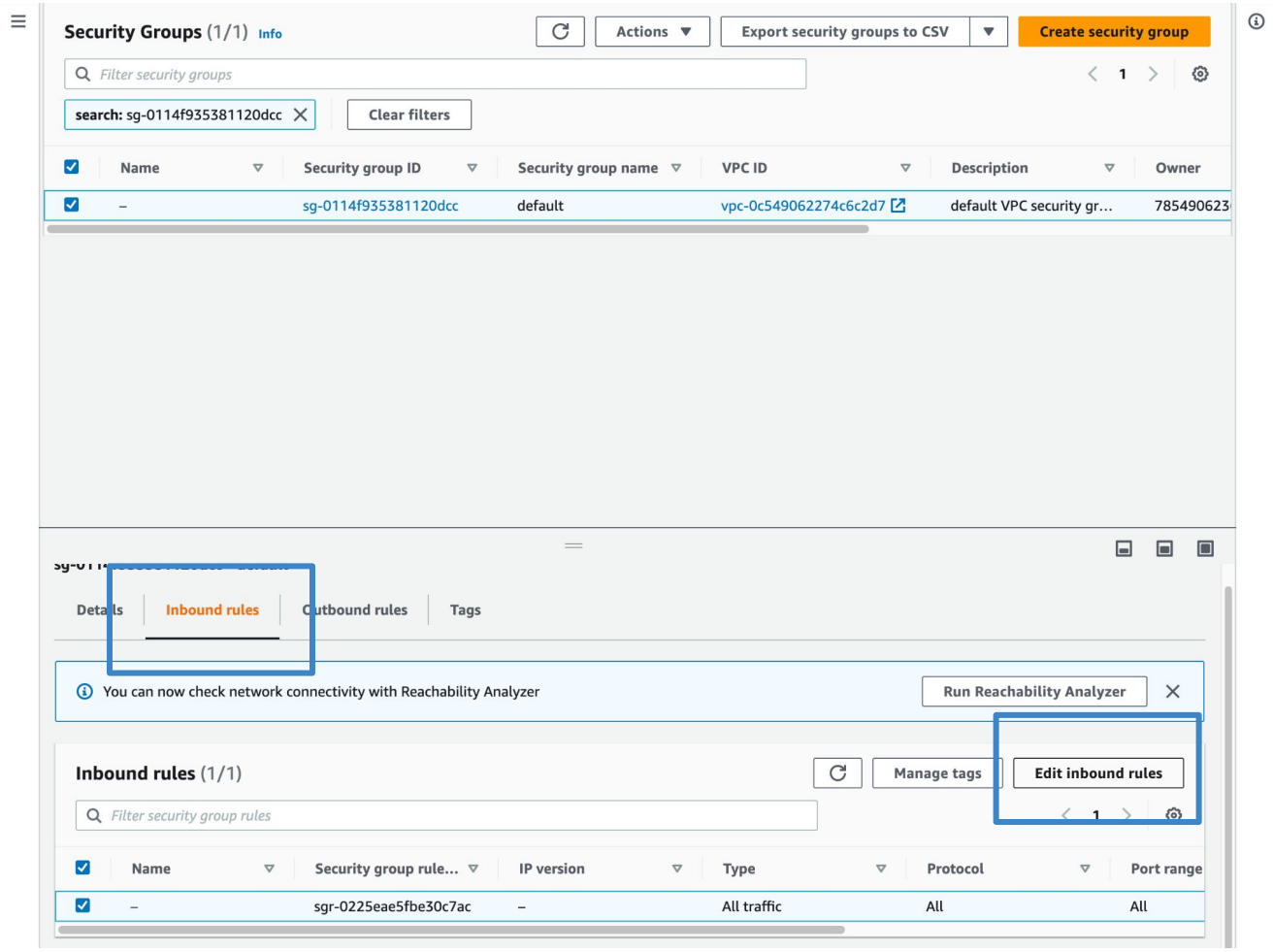
Configure security group

RDS -> click VPC security groups



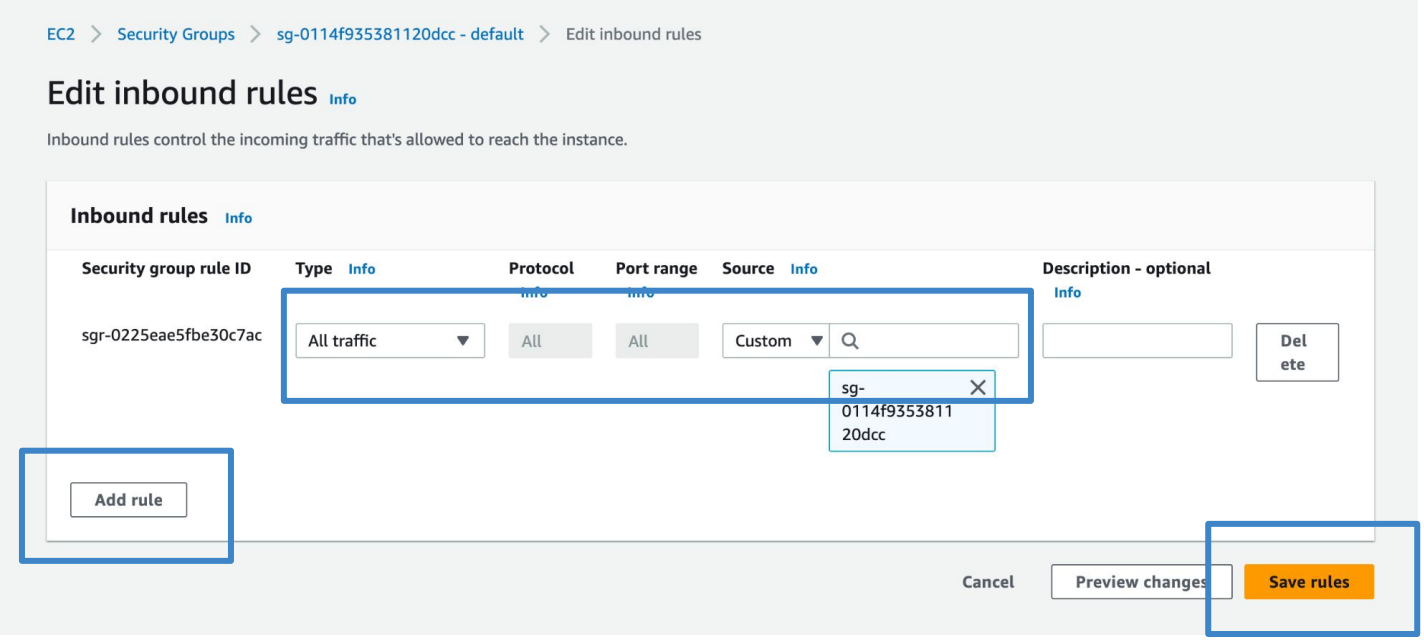
Configure security group

Inbound rules -> edit inbound rules



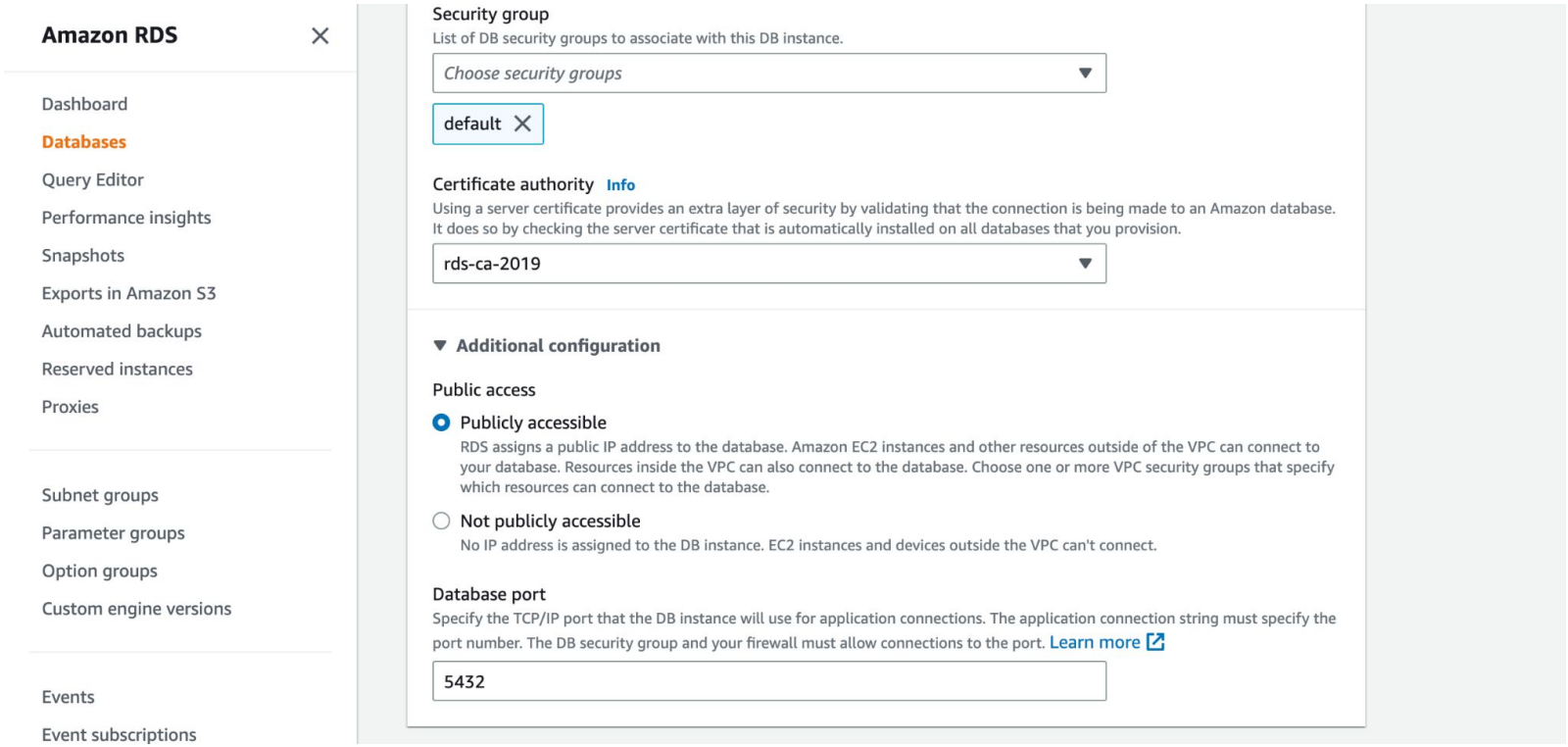
Configure security group

1. Type postgreSQL; source custom -> choose the security group from RDS
   Add rule
2. Type postgreSQL; source my ip
   Save rules



EB + RDS

RDS > database > public accessibility should be Yes, modify if No:
From RDS > Databases > modify > public access



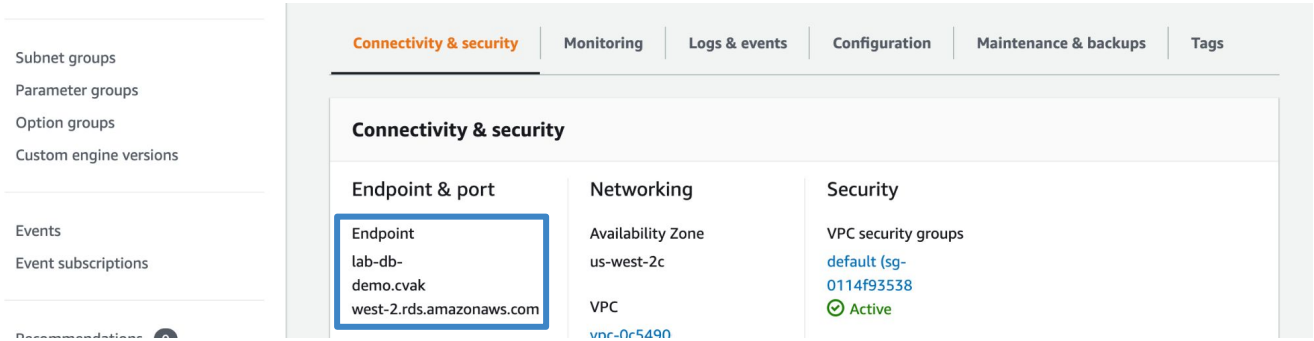
Outline

1. AWS RDS (relational database service)
2. EB / RDS connection
3. Setup weathermood db in RDS
4. Application setting and deploy

Create database in RDS

Server cmd: `createdb -h <RDS_endpoint> -p 5432 -U <RDS_master_name> weathermood`
Then enter password
To create weathermood db

```
(base) sasaya@sasayadeMBP server % createdb -h lab-db-demo.cvakngwhzvjq.us-west-2.rds.amazonaws.com -p 5432 -U postgres weathermood
Password:
Password:
```



Find endpoint in RDS

Create database in RDS

Server cmd: `psql -h <RDS_endpoint> -U <RDS_master_name>`
Then enter password
Then `\c weathermood` if you are in other db
Make sure you are connected to weathermood db

```
postgres=> \dt
         List of relations
 Schema | Name  | Type  |  Owner
--------+-------+-------+----------
 public | posts | table | postgres
 public | todos | table | postgres
(2 rows)

postgres=> \q
```

The one you connected to

Options:

-h: host

-p: port

-U: username

-d: dbname

Create database in RDS

Define schema / create table

- Method 1: Migrate schema
  - *pg_dump -h <dev-server> –U <dev-user> --no-owner --schema-only -c weathermood > db.dump*
  - *psql –h <rds-endpoint> -U <res-user> weathermood < db.dump*
- Method 2: Connect to remote psql server first and manually create

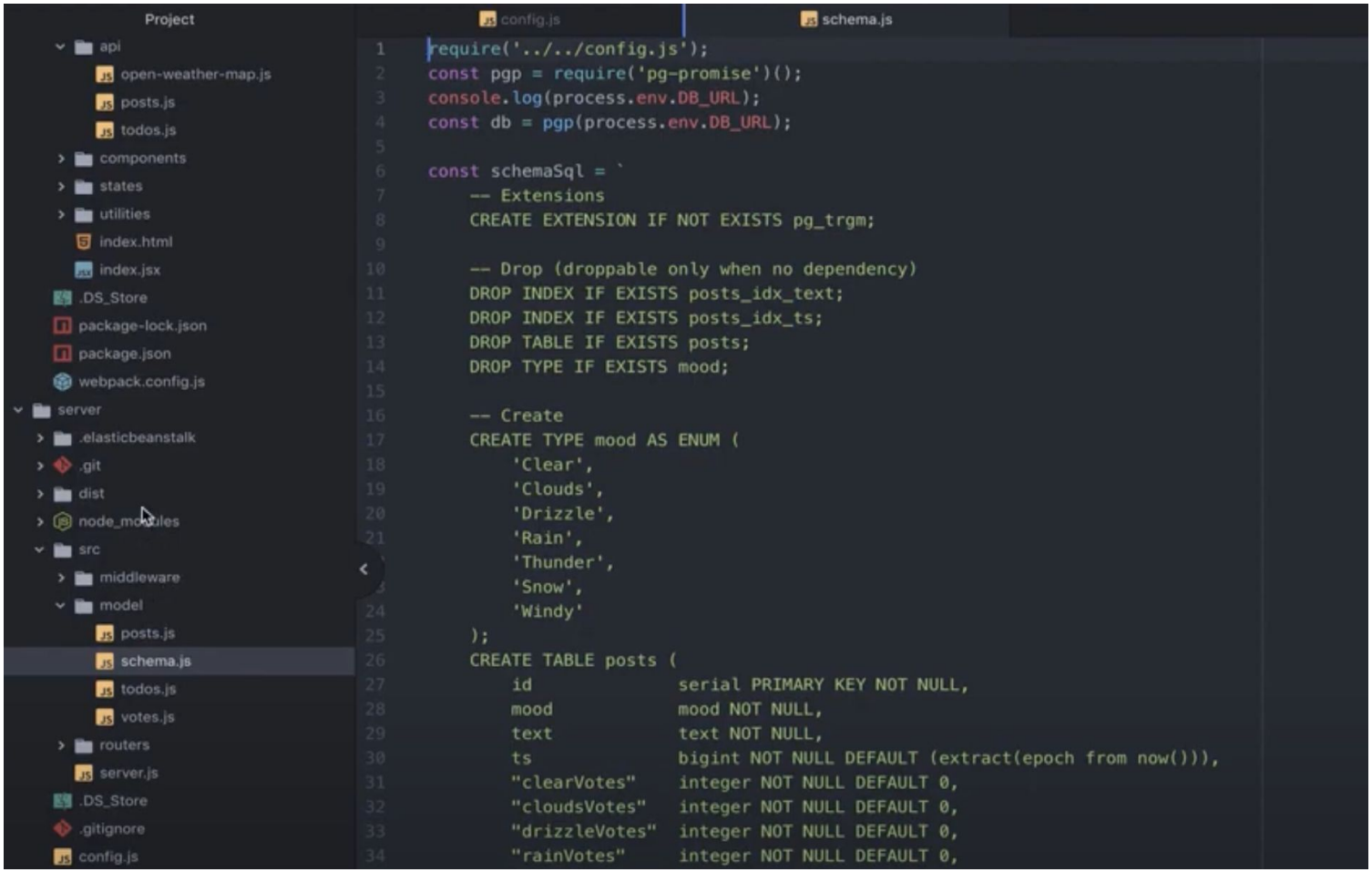
```
weathermood=> CREATE TABLE posts (
weathermood(>         id              serial PRIMARY KEY NOT NULL,
weathermood(>         mood            mood NOT NULL,
weathermood(>         text            text NOT NULL,
weathermood(>         ts              bigint NOT NULL DEFAULT (extract(epoch from now())),
weathermood(>         "clearVotes"    integer NOT NULL DEFAULT 0,
weathermood(>         "cloudsVotes"   integer NOT NULL DEFAULT 0,
weathermood(>         "drizzleVotes"  integer NOT NULL DEFAULT 0,
weathermood(>         "rainVotes"     integer NOT NULL DEFAULT 0,
weathermood(>         "thunderVotes"  integer NOT NULL DEFAULT 0,
weathermood(>         "snowVotes"     integer NOT NULL DEFAULT 0,
weathermood(>         "windyVotes"    integer NOT NULL DEFAULT 0
weathermood(>     );
```

Create database in RDS

Generate dummy data

```
weathermood=# INSERT INTO posts (mood, text)
weathermood-#     SELECT
weathermood-#         'Clear',
weathermood-#         'word' || i || ' word' || (i+1) || ' word' || (i+2)
weathermood-#     FROM generate_series(1, 100) AS s(i);
INSERT 0 100
```

Create database in RDS



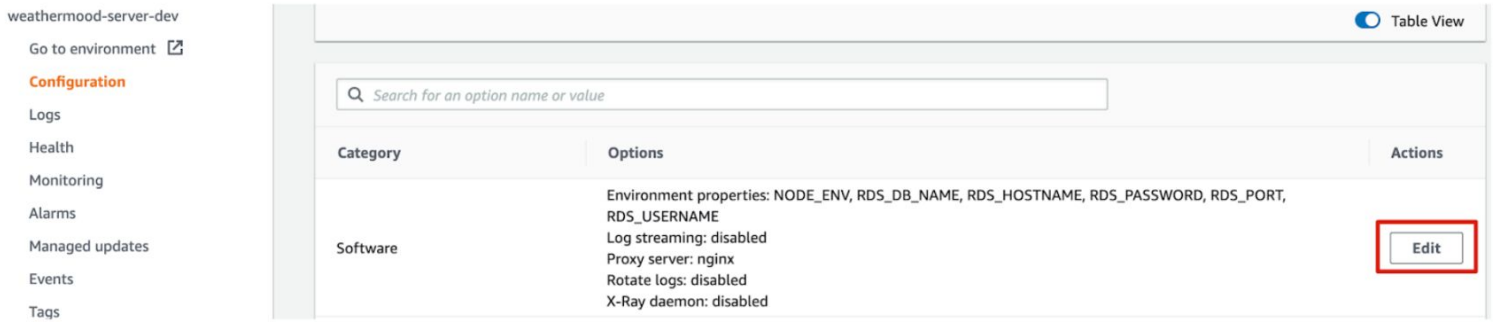
Outline

1. AWS RDS (relational database service)
2. EB / RDS connection
3. Setup weathermood db in RDS
4. Application setting and deploy

Application setting and deploy

Step 1: Add environments variables on EB environments

- **Method 1: Using EB CLI**
  - **eb setenv NODE_ENV=production, RDS_HOSTNAME=…**
  - Reminder: you also need RDS_PASSWORD for the DB_URL, check *server/config.js*
- **Method 2: Using AWS console:** EB -> Environments -> Configuration -> Software



Application setting and deploy

config.js

```
require('dotenv').config();

try {
  switch (process.env.NODE_ENV) {
    case 'development':
      process.env.DB_URL = `postgres://${process.env.PG_USERNAME}@${process.env.PG_H
      break;
    default:
      // 'staging' or 'production'
      process.env.DB_URL = `postgres://${process.env.RDS_USERNAME}:${process.env.RDS_
      break;
  }
  // only used for debugging
  console.log(`==DEBUG== process.env.DB_URL = ${process.env.DB_URL}`);
} catch (err) {
  console.log(
    err,
    '\n\nError configuring the project. Have you set the environment veriables?'
  );
}
```

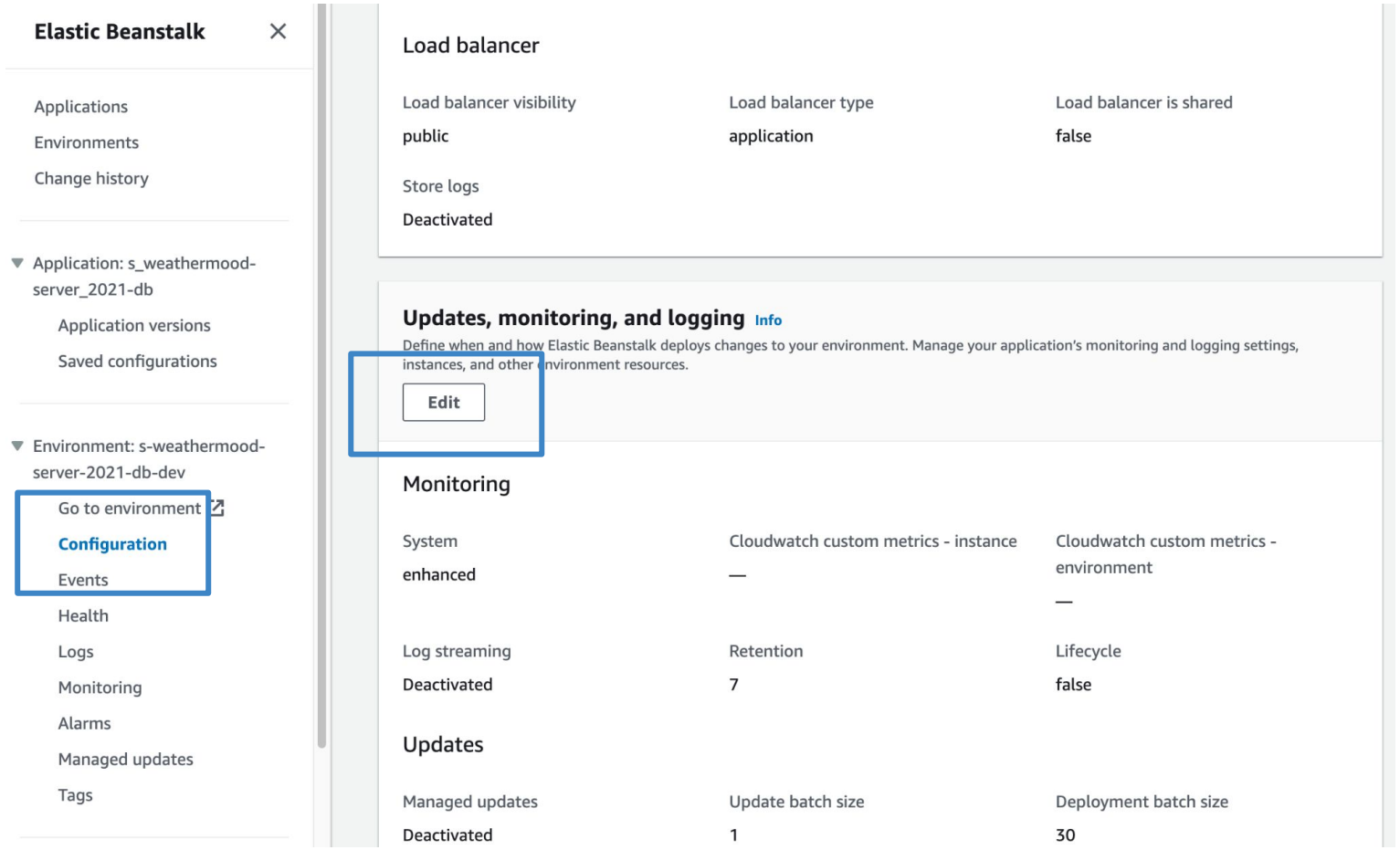
Application setting and deploy

Method 1: cmd

```
$ eb setenv NODE_ENV=production  \
  RDS_HOSTNAME=<rds-endpoint>  RDS_PORT=5432  \
  RDS_USERNAME=<user>  RDS_PASSWORD=<password>  \
  RDS_DB_NAME=weathermood
```

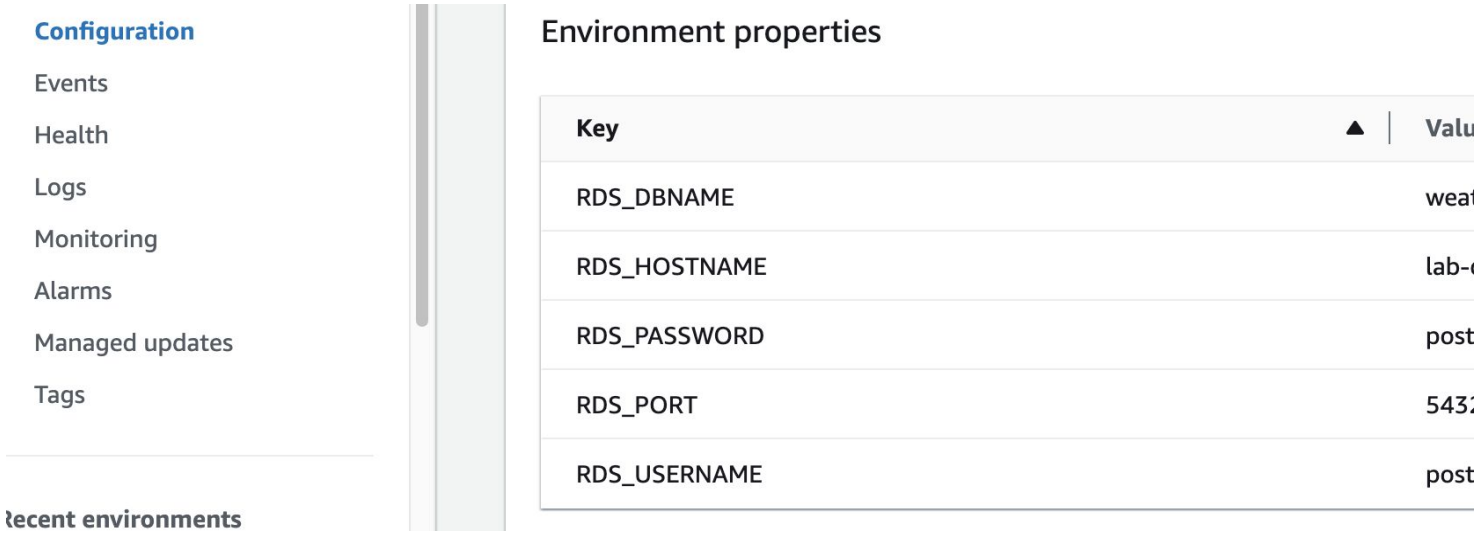
Application setting and deploy

Method 2: EB > config > edit > add properties



Application setting and deploy

Method 2: EB > config > edit > add properties



Application setting and deploy

Step 2: Configuration before deploy

- Change postBaseUrl to your server
- Change OpenWeatherAPI Key to your key
- Build client project and copy *dist* to the server project

Step 3: Deploy to EB

- Commit before deploy
- eb deploy <environment>
- Reminder: You need to specify the environment this time

Fork then clone project

1. Client side code in the client folder
2. Server side code in the server folder
3. **npm install** both first to get all the packages

// if you have other instances on EB, you may consider terminate it