Lab 09

ASW RDS Review

Software Studio

DataLab, CS, NTHU

2021 Spring

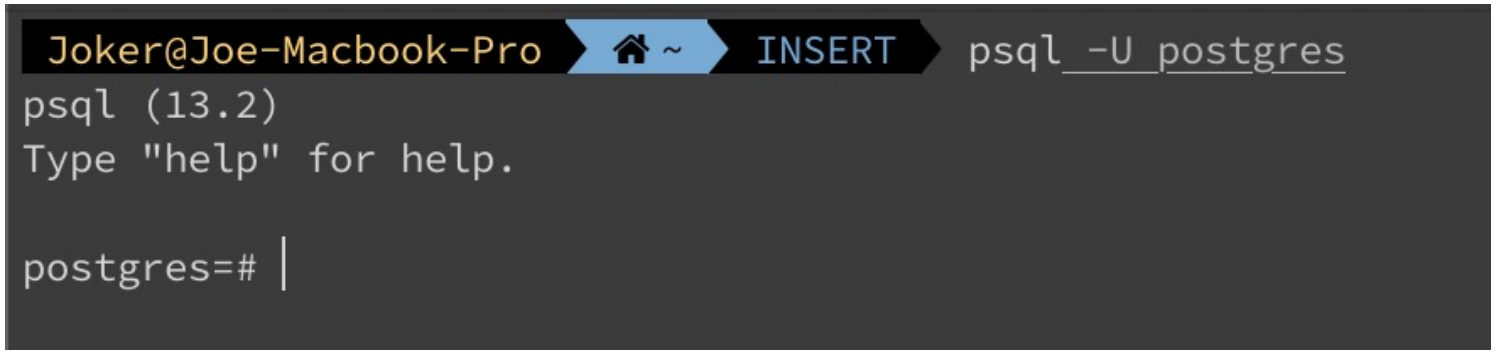
Get content from GitLab

- Pull and checkout to the new branch from GitLab
  - weathermood_no_redux -> server-db branch
  - Weathermood-server_no_redux -> db branch

0. PostgreSQL Tutorial
– Set Up a Local Database

PostgreSQL Tutorial – Set Up a Local Database

- **Step 1: Connect to PostgreSQL Database Server**
  - psql –U postgres : connect to db server as User *postgres*



PostgreSQL Tutorial – Set Up a Local Database

- **Step 2: Create Database *weathermood***
  - CREATE DATABASE weathermood

```
postgres=# CREATE DATABASE weathermood;
CREATE DATABASE
postgres=# \l
                                  List of databases
    Name     |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-------------+----------+----------+-------------+-------------+-----------------------
 Joker       | Joker    | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
             |          |          |             |             | postgres=CTc/postgres
 template1   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
             |          |          |             |             | postgres=CTc/postgres
 weathermood | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(5 rows)
```

PostgreSQL Tutorial – Set Up a Local Database

- **Step 3: Connect to the database**
  - \c <dbname> <username> : connect to the database with username
  - \dt : List available tables

```
postgres=# \c weathermood
You are now connected to database "weathermood" as user "postgres".
weathermood=# \dt
Did not find any relations.
weathermood=#
```

PostgreSQL Tutorial – Set Up a Local Database

- **Step 4: Create a table**
  - \e: edit command in our own editor
  - DROP TABLE IF EXISTS <table name>
  - CREATE TABLE <table name>

```
weathermood=# \e
NOTICE:  table "posts" does not exist, skipping
DROP TABLE
NOTICE:  type "mood" does not exist, skipping
DROP TYPE
CREATE TYPE
CREATE TABLE
```

psql.edit.50250.sql

var > folders > q5 > f0bgxj6n1fg42922n6vxd9mh0000gn > T > psql.edit.50

Run on active connection | Select block

```
-- Drop (droppable only when no dependency)
DROP TABLE IF EXISTS posts;
DROP TYPE IF EXISTS mood;
-- Create
CREATE TYPE mood AS ENUM (
  'Clear',
  'Clouds',
  'Drizzle',
  'Rain',
  'Thunder',
  'Snow',
  'Windy'
);
CREATE TABLE posts (
  id serial PRIMARY KEY NOT NULL,
  mood mood NOT NULL,
  text text NOT NULL,
  ts bigint NOT NULL DEFAULT (
    extract(
      epoch
      from now()
    )
  ),
  "clearVotes" integer NOT NULL DEFAULT 0,
  "cloudsVotes" integer NOT NULL DEFAULT 0,
  "drizzleVotes" integer NOT NULL DEFAULT 0,
  "rainVotes" integer NOT NULL DEFAULT 0,
  "thunderVotes" integer NOT NULL DEFAULT 0,
  "snowVotes" integer NOT NULL DEFAULT 0,
  "windyVotes" integer NOT NULL DEFAULT 0
);
```

PostgreSQL Tutorial – Set Up a Local Database

- **Step 5: Check the table**
  - \d <table name>: Describe a table such as column, type, etc.

```
weathermood=# \dt
         List of relations
 Schema | Name  | Type  |  Owner
--------+-------+-------+----------
 public | posts | table | postgres
(1 row)

weathermood=# \d posts
                                  Table "public.posts"
    Column     |  Type   | Collation | Nullable |              Default
---------------+---------+-----------+----------+-----------------------------------
 id            | integer |           | not null | nextval('posts_id_seq'::regclass)
 mood          | mood    |           | not null |
 text          | text    |           | not null |
 ts            | bigint  |           | not null | date_part('epoch'::text, now())
 clearVotes    | integer |           | not null | 0
 cloudsVotes   | integer |           | not null | 0
 drizzleVotes  | integer |           | not null | 0
 rainVotes     | integer |           | not null | 0
 thunderVotes  | integer |           | not null | 0
 snowVotes     | integer |           | not null | 0
 windyVotes    | integer |           | not null | 0
Indexes:
    "posts_pkey" PRIMARY KEY, btree (id)
```

PostgreSQL Tutorial – Set Up a Local Database

- **Step 6: INSERT some test data**
  - INSERT INTO <table name> (column 1, column 2, …)

    VALUES (value1, value2, …)

```
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
```

PostgreSQL Tutorial – Set Up a Local Database

- **Step 7: Check the data**
  - SELECT * FROM posts ORDER BY id DESC LIMIT 5;

```
weathermood=# SELECT * FROM posts ORDER BY id DESC LIMIT 5;
 id  | mood  |          text           |     ts     | clearVotes | cloudsVotes | drizzleVotes | rainVotes | thunderVotes | snowVotes | windyVotes
-----+-------+-------------------------+------------+------------+-------------+--------------+-----------+--------------+-----------+------------
 100 | Clear | word100 word101 word102 | 1620837606 |          0 |           0 |            0 |         0 |            0 |         0 |          0
  99 | Clear | word99 word100 word101  | 1620837606 |          0 |           0 |            0 |         0 |            0 |         0 |          0
  98 | Clear | word98 word99 word100   | 1620837606 |          0 |           0 |            0 |         0 |            0 |         0 |          0
  97 | Clear | word97 word98 word99    | 1620837606 |          0 |           0 |            0 |         0 |            0 |         0 |          0
  96 | Clear | word96 word97 word98    | 1620837606 |          0 |           0 |            0 |         0 |            0 |         0 |          0
(5 rows)
```

PostgreSQL Tutorial – Set Up a Local Database

- **Step 8: Create Index**
  - CREATE INDEX <idx_name> ON <table_name> USING [method]

```
-- Extensions
CREATE EXTENSION IF NOT EXISTS pg_trgm;
-- Drop (droppable only when no dependency)
DROP INDEX IF EXISTS posts_idx_text;
DROP INDEX IF EXISTS posts_idx_ts;
CREATE INDEX posts_idx_ts ON posts USING btree(ts);
CREATE INDEX posts_idx_text ON posts USING gin(text gin_trgm_ops);
```

PostgreSQL Tutorial – Set Up a Local Database

Step 9: Connect application to database – 1

- Set up environment variables
  - *NODE_ENV* = development
  - *PG_USERNAME* = postgres
  - *PG_HOSTNAME* = localhost
  - *PG_PORT* = 5432
  - *PG_DBNAME* = weathermood

Windows

```
SET NODE_ENV=development
```

Mac

```
export NODE_ENV=development
```

PostgreSQL Tutorial – Set Up a Local Database

Step 9: Connect application to database – 2

- DB Connection settings

- config.js

```
switch (process.env.NODE_ENV) {
  case 'development':
    process.env.DB_URL = `postgres://${process.env.PG_USERNAME}@${process.env.PG_HOSTNAME}:
    ${process.env.PG_PORT}/${process.env.PG_DB_NAME}`;
```

- model/posts.js

```
if (!global.db) {
  const pgp = require('pg-promise')();
  db = pgp(process.env.DB_URL);
}
```

PostgreSQL Tutorial – Set Up a Local Database

Step 10: Define API to interact with DB - 1

- Define model to access database using SQL
  - server/model/posts.js

```
function list(searchText = '', start) {
  const where = [];
  if (searchText) where.push(`text ILIKE '%$1:value%'`);
  if (start) where.push('id < $2');
  const sql = `
        SELECT *
        FROM posts
        ${where.length ? 'WHERE ' + where.join(' AND ') : ''}
        ORDER BY id DESC
        LIMIT 10
    `;
  return db.any(sql, [searchText, start]);
}
```

PostgreSQL Tutorial – Set Up a Local Database

Step 10: Define API to interact with DB - 2

- Set up router to receive request and query
  - server/router/posts.js

```
// List
router.get('/posts', function (req, res, next) {
  const { searchText, start } = req.query;
  postModel
    .list(searchText, start)
    .then((posts) => {
      res.json(posts);
    })
    .catch(next);
});
```

PostgreSQL Tutorial – Set Up a Local Database

Step 10: Define API to interact with DB - 3

- Client send request based on the defined API
  - client/api/posts.js

```
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) {
```

PostgreSQL Tutorial – Set Up a Local Database

- Step 1: Connect to PostgreSQL Database Server
- Step 2: Create database *weathermood*
- Step 3: Connect to the database
- Step 4: Create a table
- Step 5: Check the table
- Step 6: INSERT some test data
- Step 7: Check the data
- Step 8: Create Index
- Step 9: Connect application to database
- Step 10: Define API to interact with DB

PostgreSQL Tutorial – Set Up a Local Database

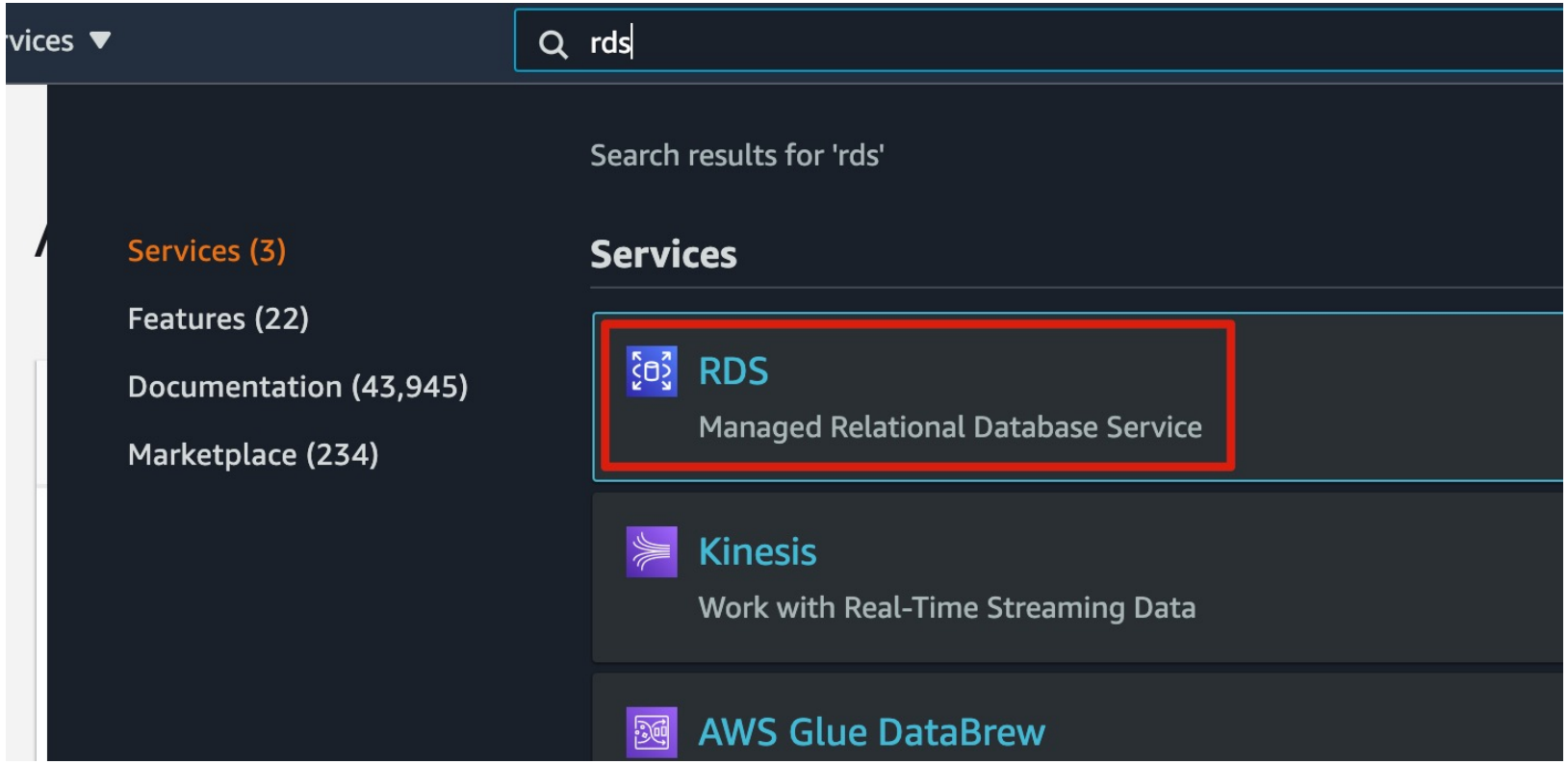
Some useful tricks:

1. Run 2 servers to avoid copy *dist* from client to server during development
2. Write a script to automate the setting of environment variables
    - Reminder: the setting would only be available in this shell

1. Creating RDS Database

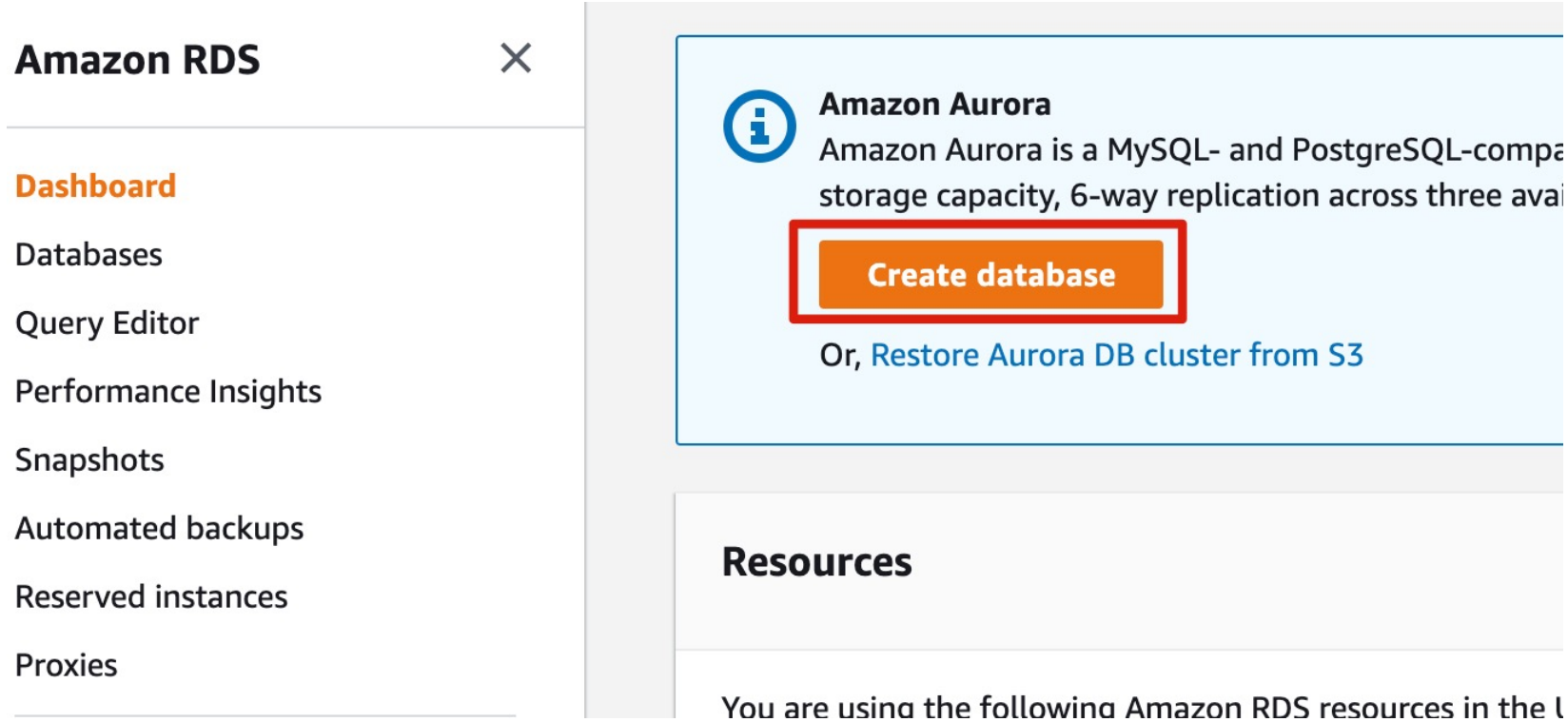
Creating RDS Database

Step 1: Find RDS service



Creating RDS Database

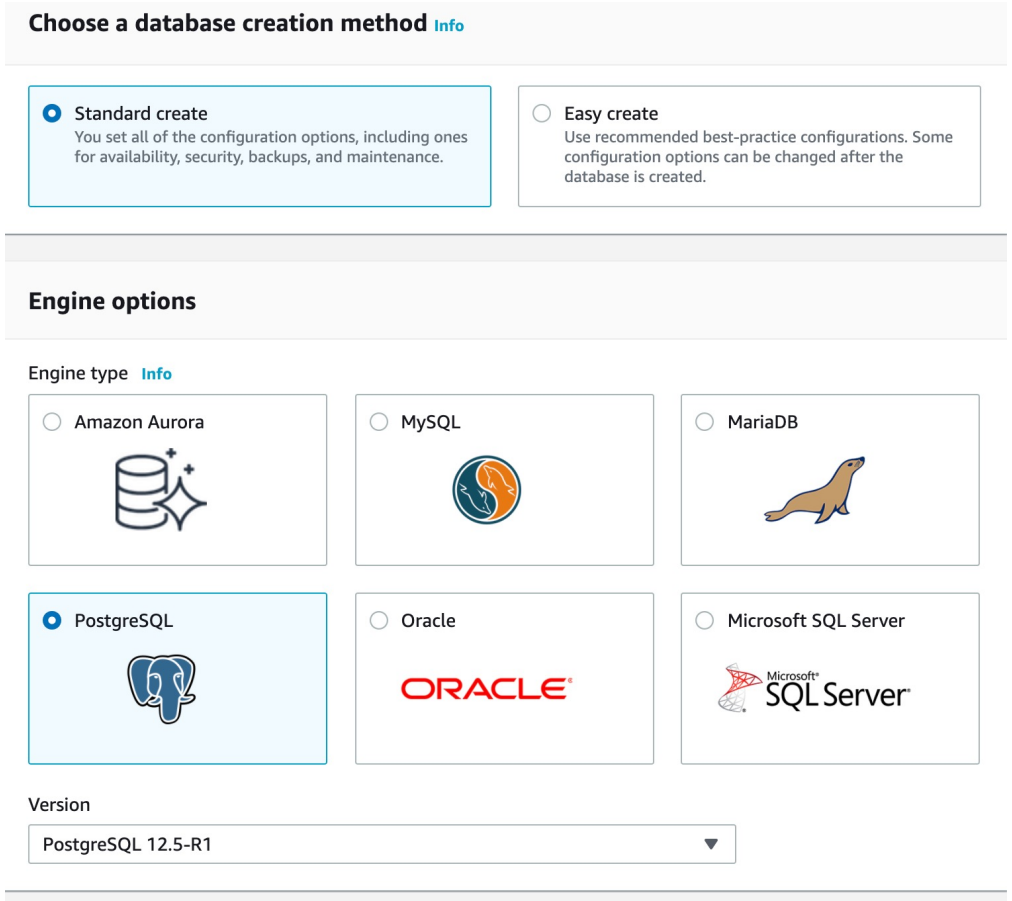
Step 2: Start creating database



Creating RDS Database

Step 3: Database settings - 1

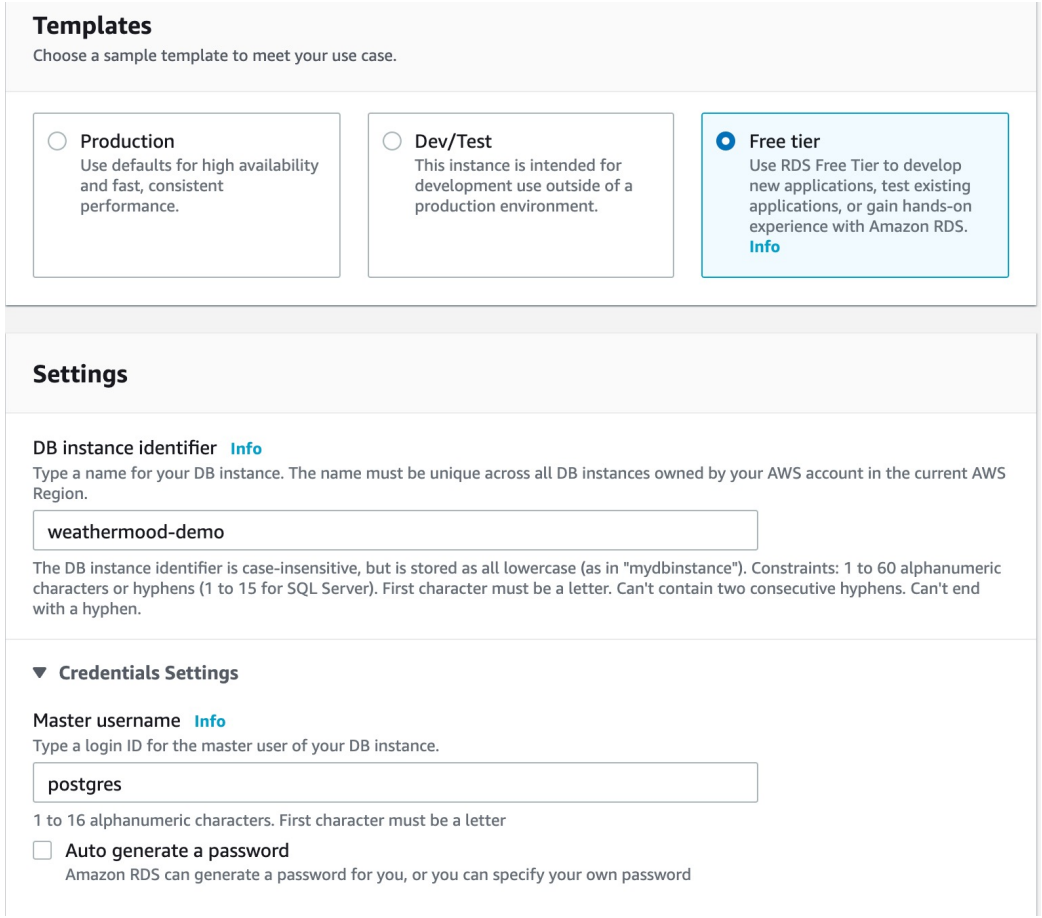
- Method: Standard create
- Engine type: PostgreSQL
- Version: 12.5-R1 (default for free tier)



Creating RDS Database

Step 3: Database settings - 2

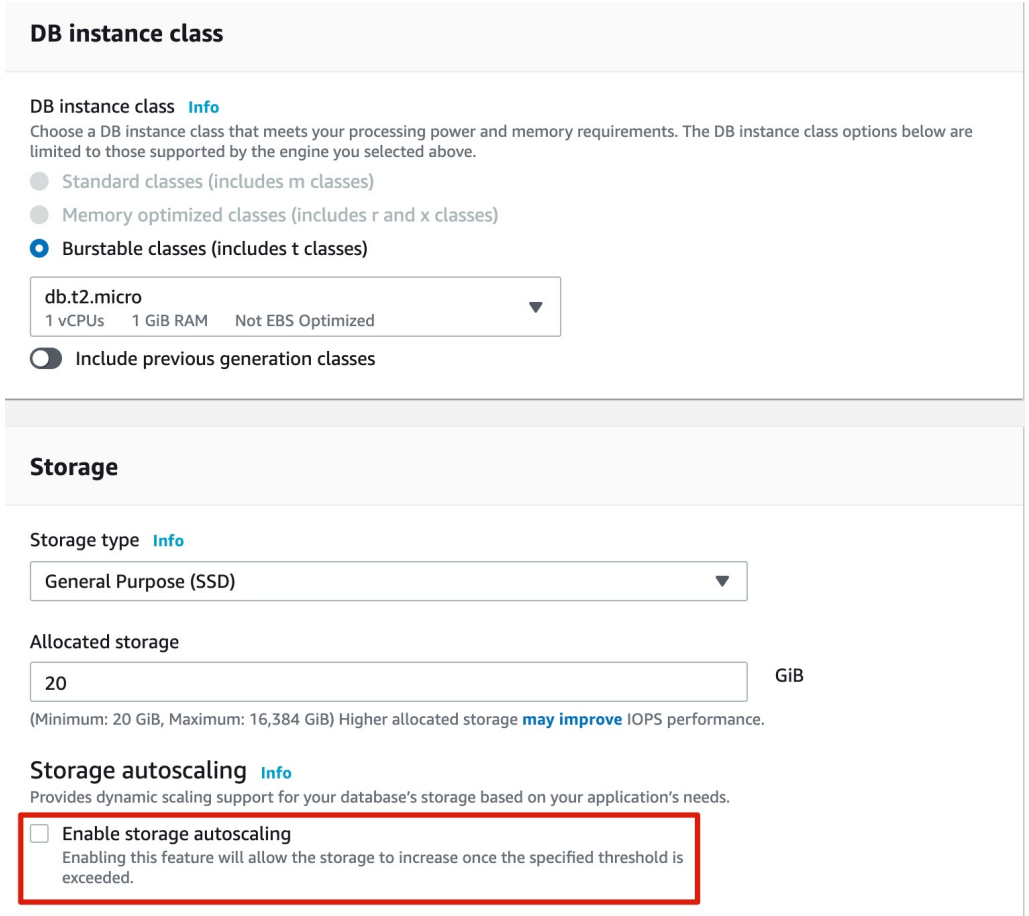
- Template: Free tier
- Master username: postgres (recommended)



Creating RDS Database

Step 3: Database settings - 3

- DB instance class: default
- Storage: default, only uncheck storage autoscaling

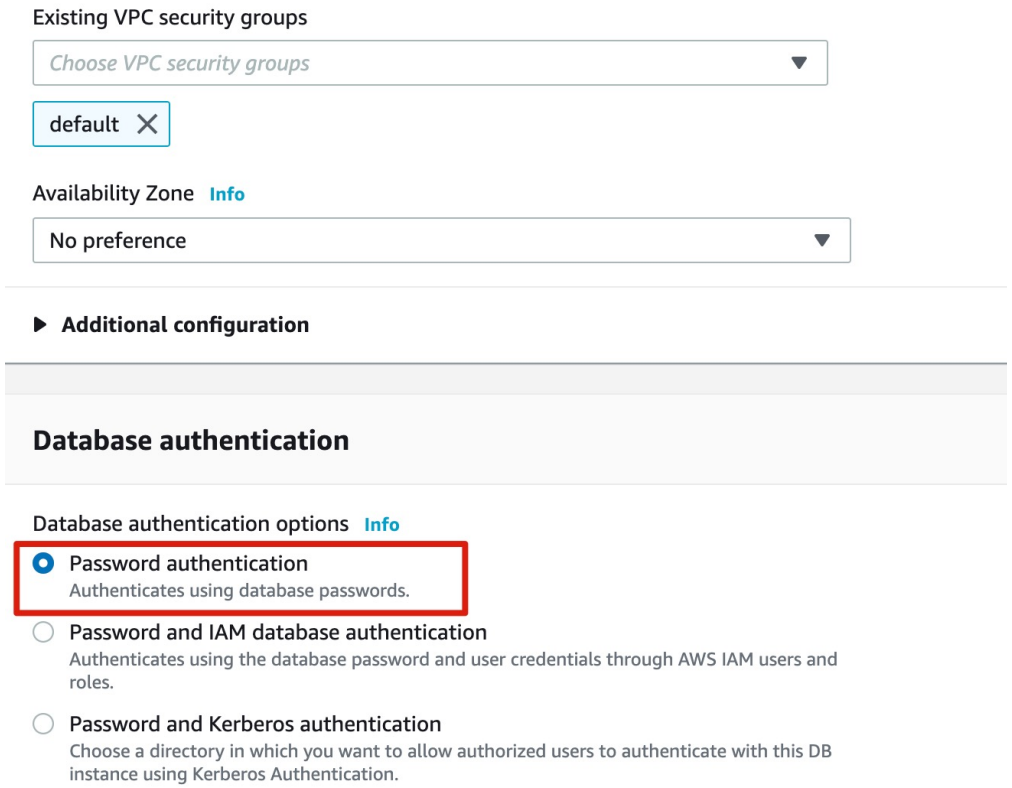


Creating RDS Database

Step 3: Database settings – 4

- Database authentication:

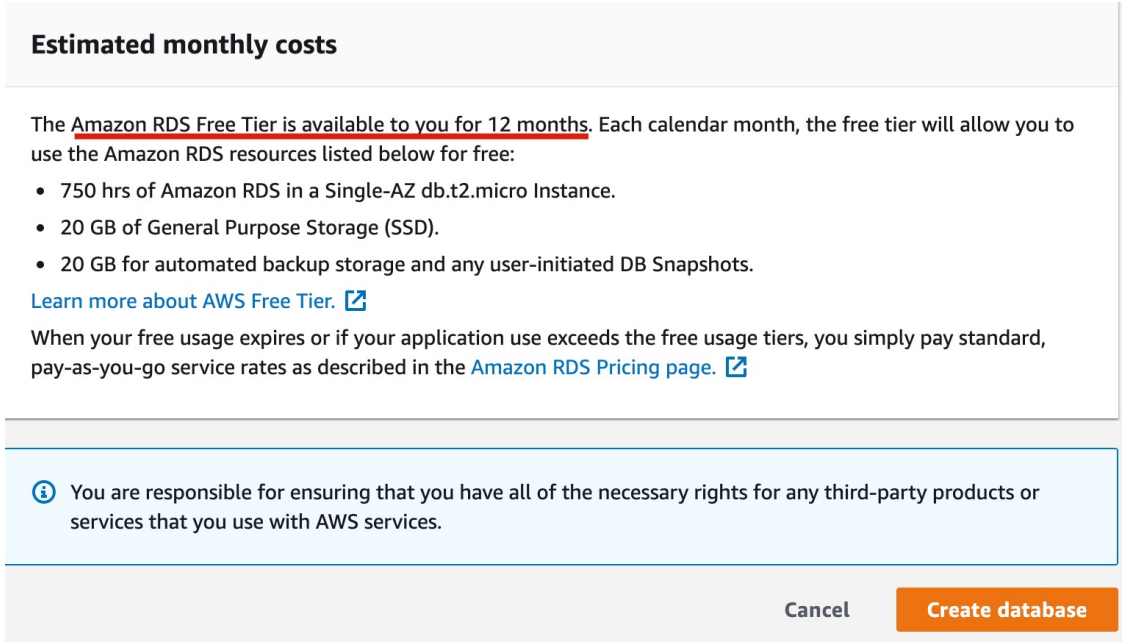
  Password authentication



Creating RDS Database

Step 3: Database settings – 5

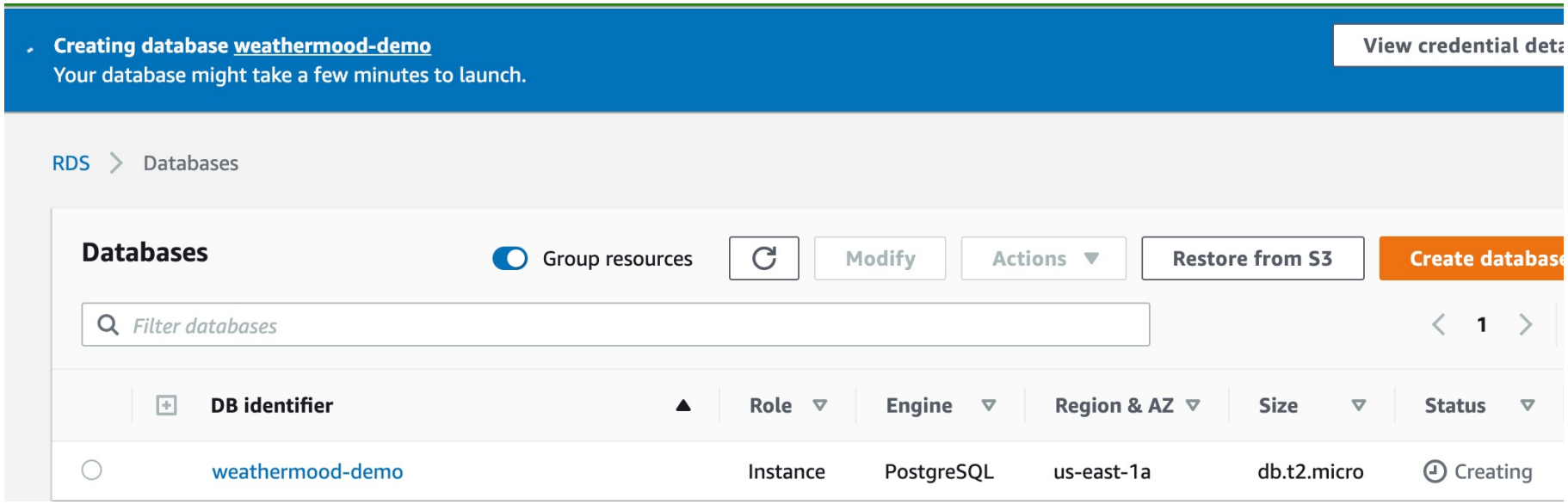
- Review: make sure you are using free tier



Creating RDS Database

Step 4: Check if DB successfully created

- Reminder: it might take few minutes to launch

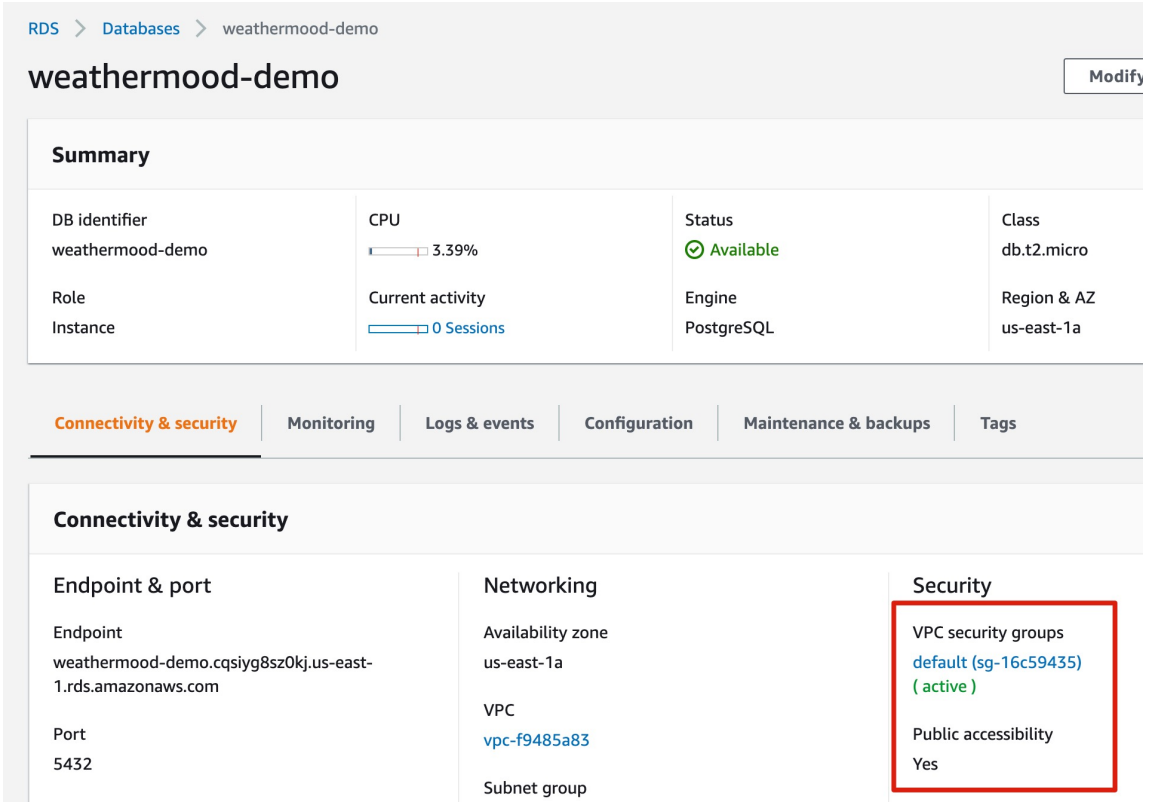


2. EB/RDS Connection – Security Group Setting

EB/RDS Connection – Security Group Setting

Step 1: Database details checking

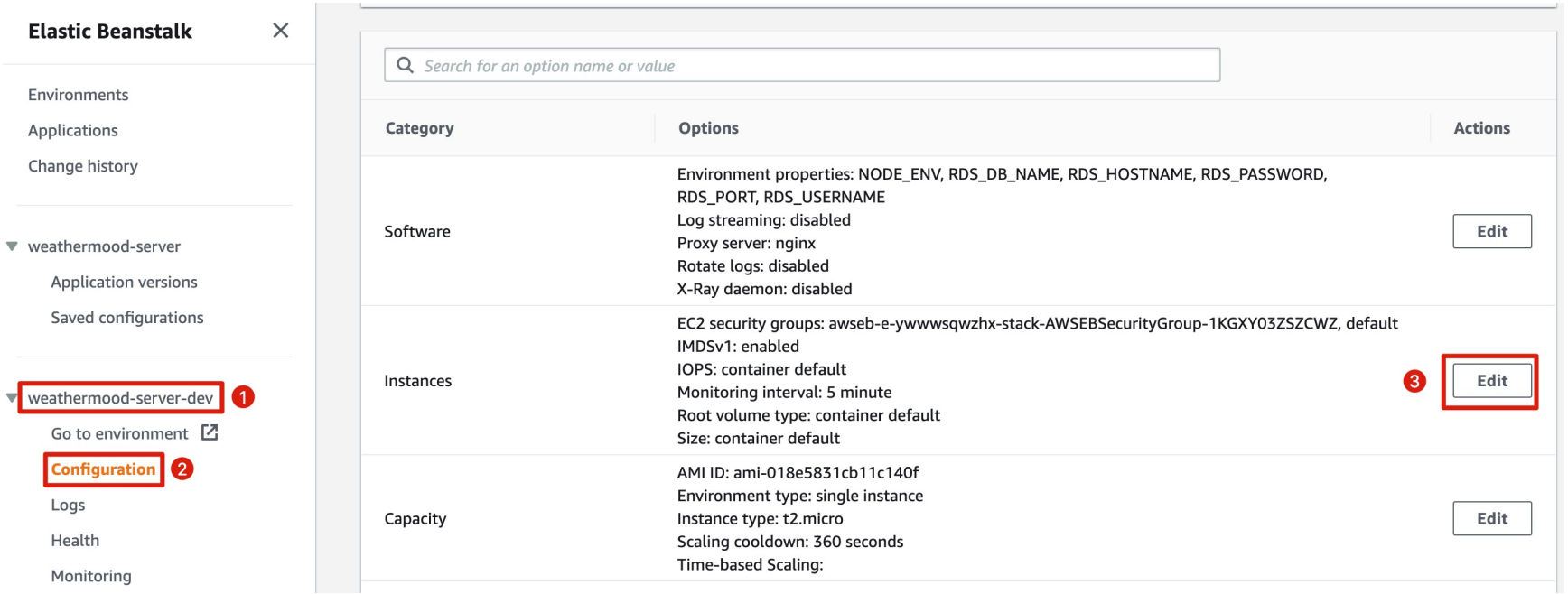
- Security groups
- Public Accessibility



EB/RDS Connection – Security Group Setting

Step 2: EB security group setting - 1

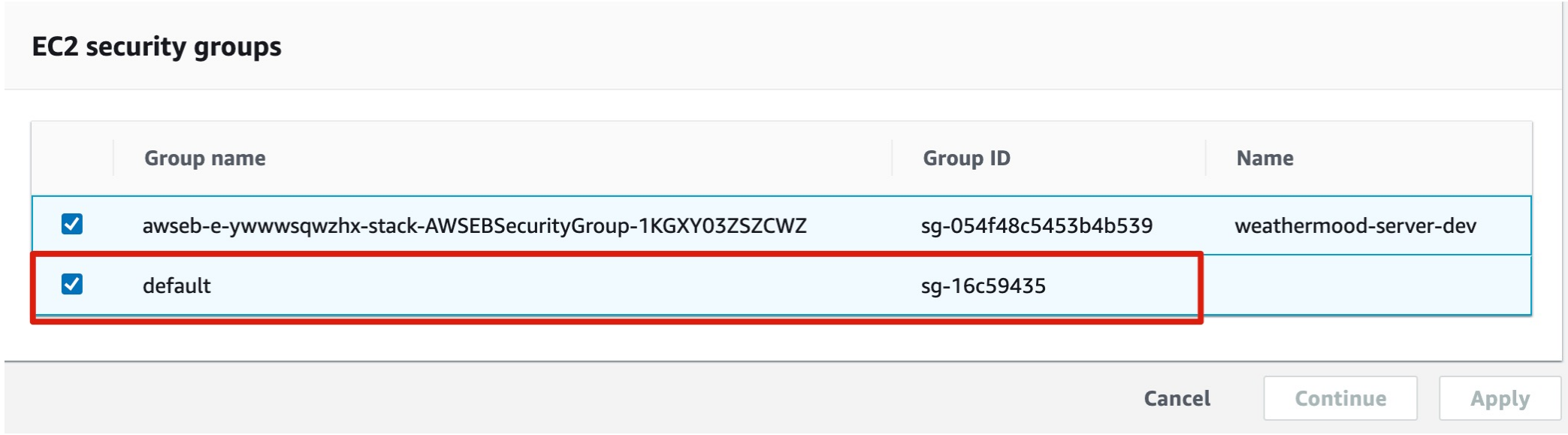
- EB -> Environments -> Configuration -> Instance



EB/RDS Connection – Security Group Setting

Step 2: EB security group setting - 2

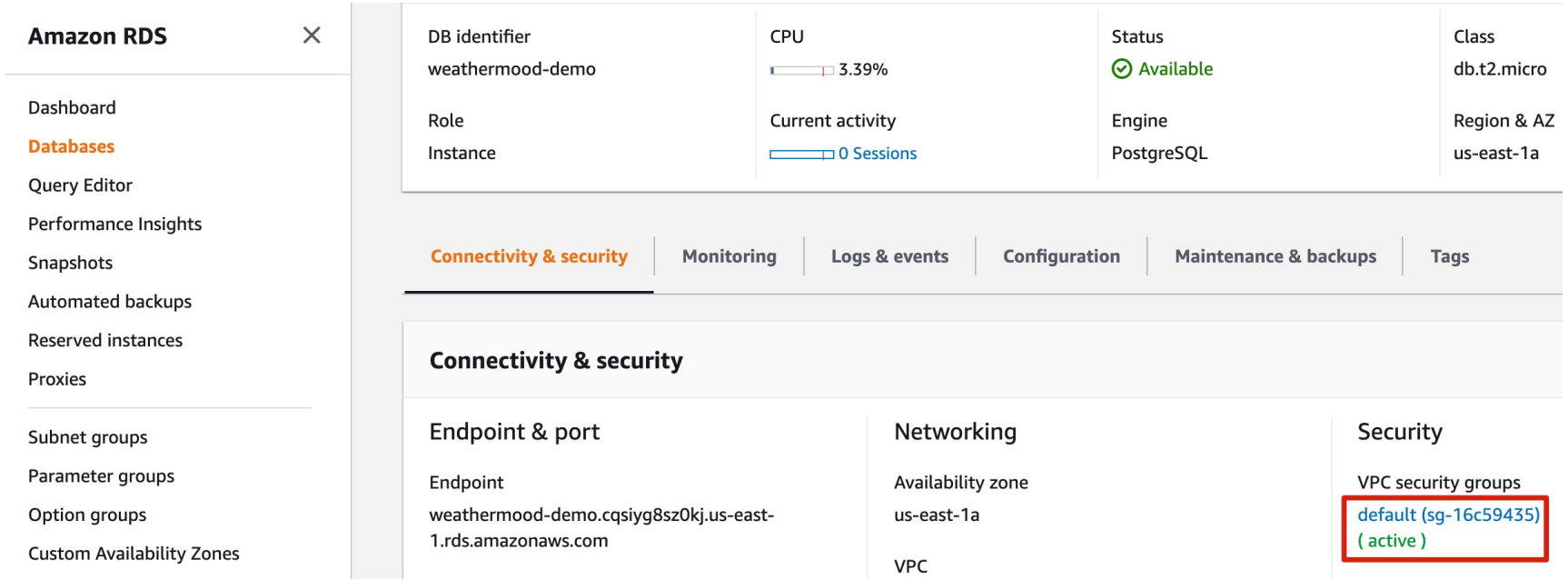
- EC2 security groups: add the one from your DB



EB/RDS Connection – Security Group Setting

Step 3: Security group inbound rules setting - 1

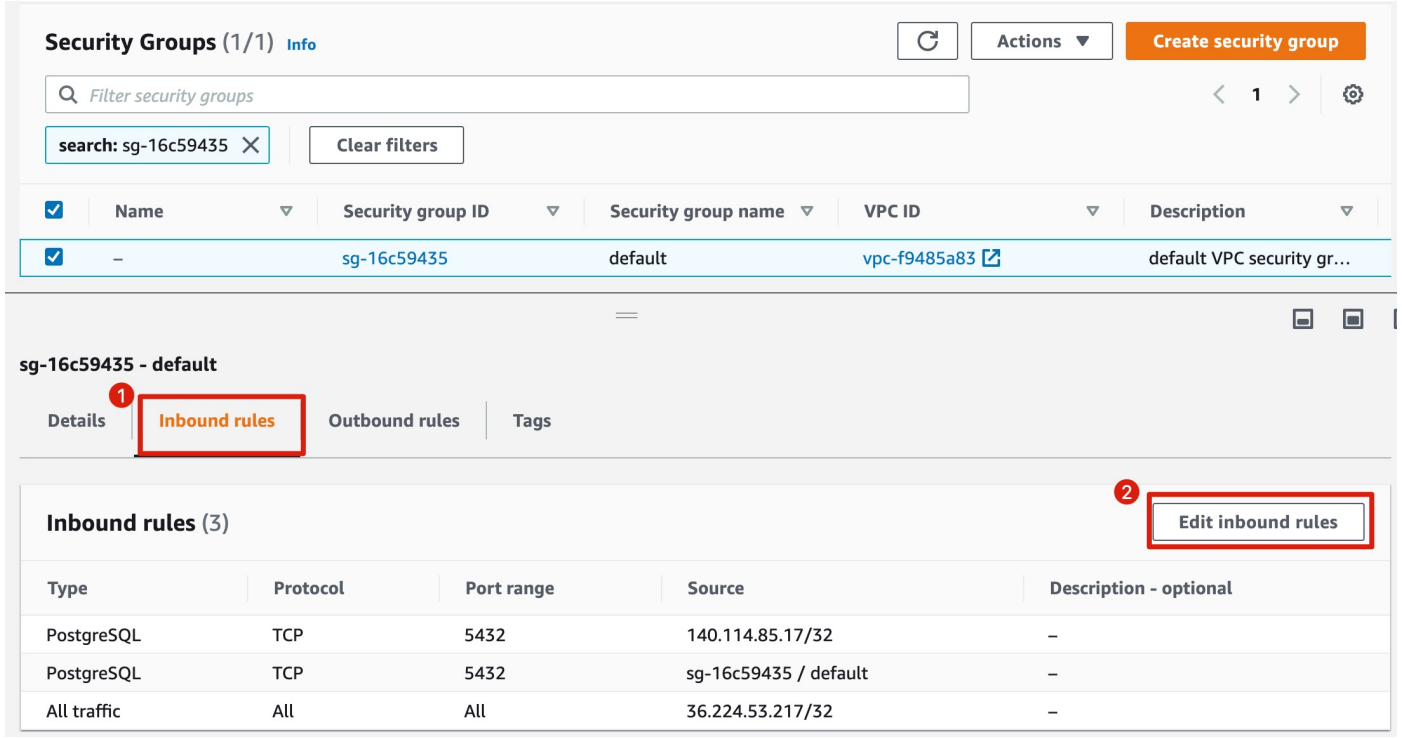
- RDS -> Databases -> Click security group



EB/RDS Connection – Security Group Setting

Step 3: Security group inbound rules setting - 2

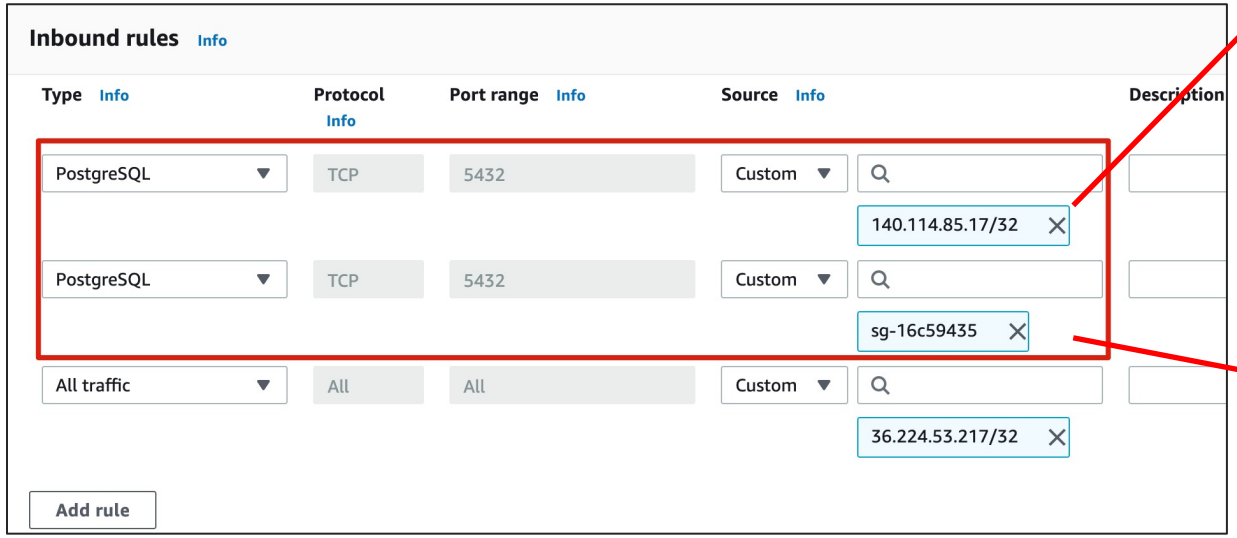
- Inbound rules -> Edit Inbound rules

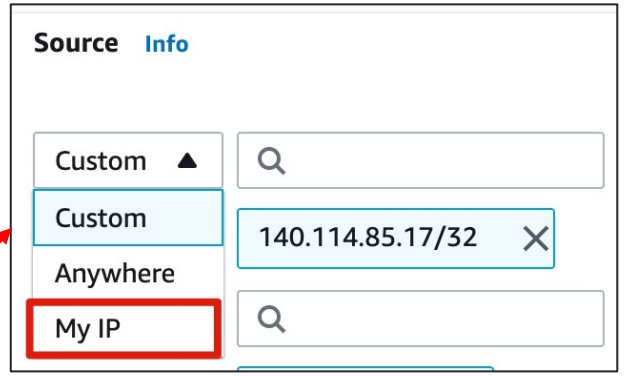


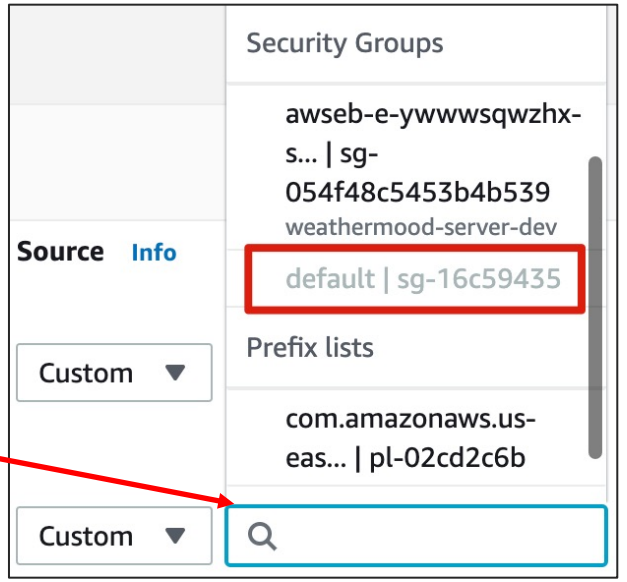
EB/RDS Connection – Security Group Setting

Step 3: Security group inbound rules setting - 3

- Add 2 rules:
  - Type: PostgreSQL
  - Source: My IP, Security Group







3. Set Up *weathermood* DB in RDS

Set Up *weathermood* DB in RDS

Step 1: Create *weathermood* DB

- *createdb –h <rds-endpoint> -p 5432 –U postgres <DB Name>*

Step 2: Connect to *weathermood* DB

- *psql –h <rds-endpoint> -p 5432 –U postgres –d <DB Name>*

Options:

-h: host

-p: port

-U: username

-d: dbname

```
Joker@Joe-Macbook-Pro ~ INSERT createdb -h weathermood-demo.cqsiyg8sz0kj.us-east-1.rds.amazonaws.com -p 5432 -U postgres weathermood
Password:
Joker@Joe-Macbook-Pro ~ INSERT psql -h weathermood-demo.cqsiyg8sz0kj.us-east-1.rds.amazonaws.com -p 5432 -U postgres -d weathermood
Password for user postgres:
psql (13.2, server 12.5)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

weathermood=>
```

Set Up *weathermood* DB in RDS

Step 3: 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(>     );
```

Set Up *weathermood* DB in RDS

Step 4: Generate dummy data

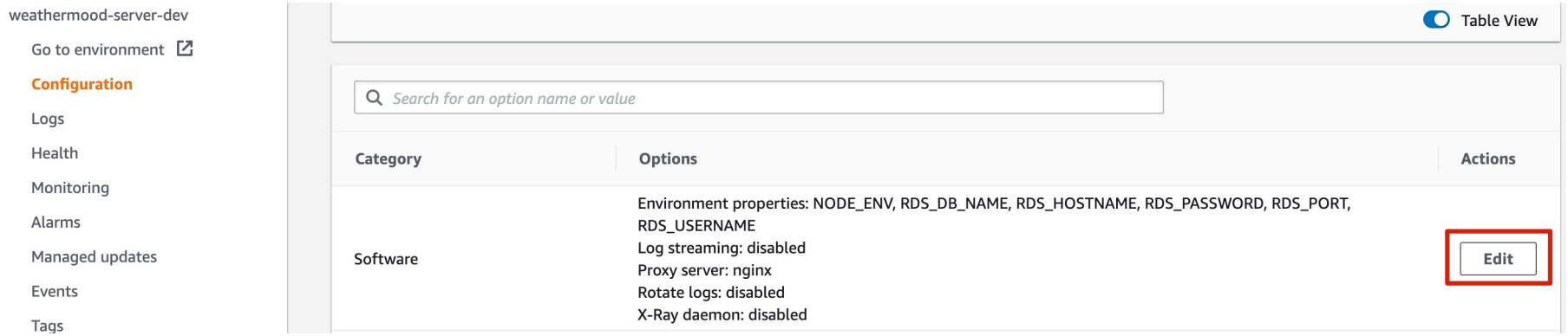
```
weathermood=# INSERT INTO posts (mood, text)
weathermood-#     SELECT
weathermood-#         'Clear',
weathermood-#         'word' || i || ' word' || (i+1) || ' word' || (i+2)
weathermood-#     FROM generate_series(1, 100) AS s(i);
INSERT 0 100
```

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 *config.js*
- **Method 2: Using AWS console:** EB -> Environments -> Configuration -> Software



Application Setting and Deploy

Step 2: Configuration before deploy

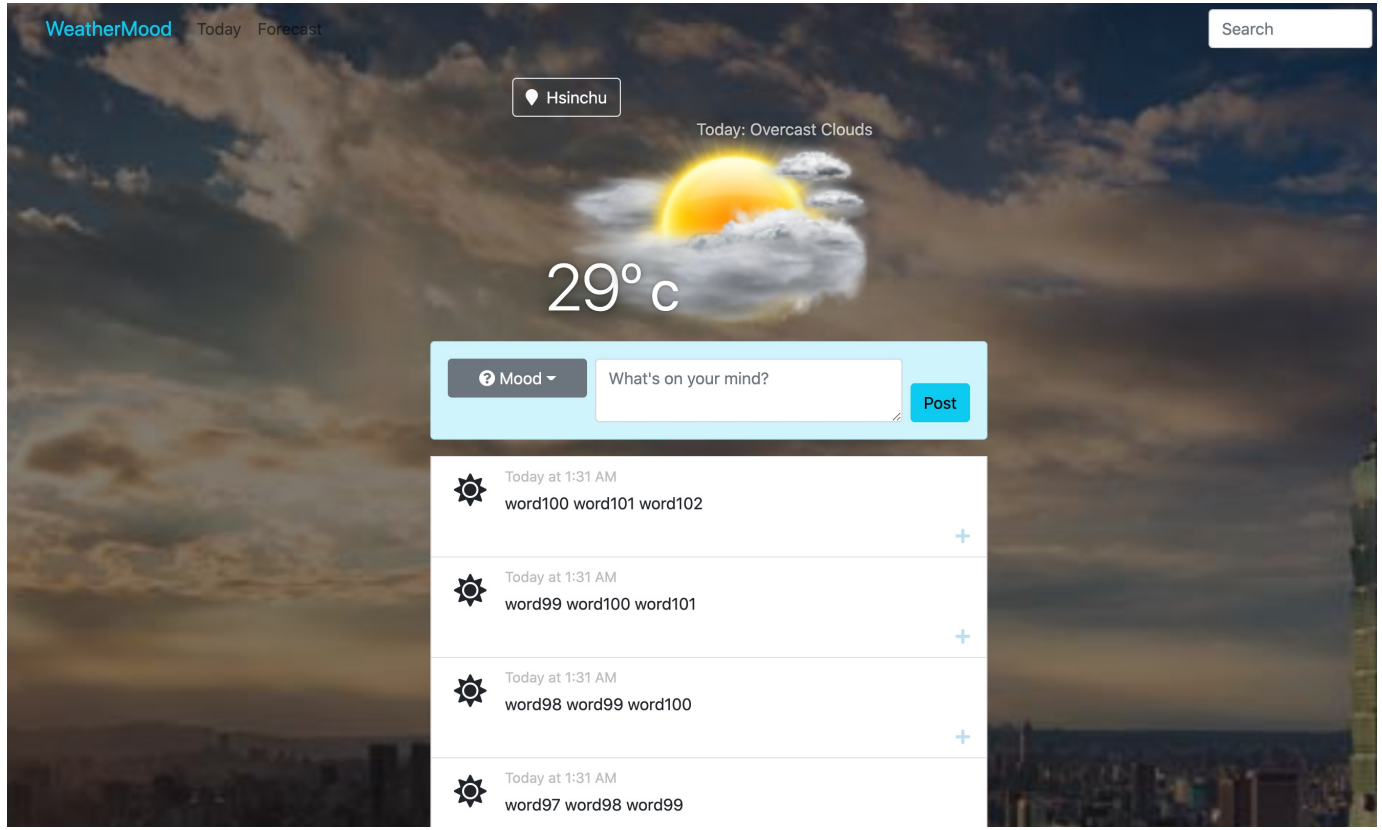
- Change postBaseUrl to your server
- Change OpenWeatherAPI Key to your key
- Build client project and copy *dist* to the server project

Step 3: Deploy to EB

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

Application Setting and Deploy

Step 4: Check in your browser



Application Setting and Deploy

Some useful tricks:

- eb console: Open the EB console in browser
- eb list: List all environments
- eb printenv: Show the environment variables