

Lab 7

AWS RDS

Software Studio
Datalab, CS, NTHU
2023

Gitlab

lab-weathermood-server-db-todo

The screenshot shows the GitLab interface for the repository 'lab-weathermood-server-db-todo'. The left sidebar contains navigation links for Project information, Repository, Files, Commits, Branches, Tags, Contributors, Graph, Compare, Issues (0), Merge requests (0), Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, Wiki, Snippets, and Settings. The main content area shows a commit history for the 'master' branch. One commit, 'Update README.md' by Sheng-ya Chiu, was just now authored. Below the commit history is a table listing files with their last commit and update times. The 'README.md' file is shown with its content, which includes a section titled 'Weathermood Server BD TODO' and requirements for improving the 'Forecast' page.

Name	Last commit	Last update
client	initial commit	3 minutes ago
server	initial commit	3 minutes ago
.gitignore	initial commit	3 minutes ago
README.md	Update README.md	just now

README.md

Weathermood Server BD TODO

In this assignment, you are asked to improve your todo function on the "Forecast" page using relational database.

Requirement

On the "Forecast" page:

1. Deploy to AWS (Must!!! or you will get a 0. You may need to use AWS RDS)
2. (80%) Store and get data from DB (So you maybe need to design the DB schema).
3. (10%) The pagination function.
4. (10%) The "Unaccomplished" function and the "Accomplished" function should be server-side not simulated server-side like this

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

The screenshot shows the Amazon RDS Dashboard. On the left, there's a sidebar with links like 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), and Certificate update. The main content area has a large blue callout box at the top with the text: "Try the new Amazon RDS Multi-AZ deployment option for MySQL and PostgreSQL. For your Amazon RDS for MySQL and PostgreSQL workloads, improve transactional commit latencies by 2x, experience faster failover typically less than 35 seconds and, get read scalability with two readable standby DB instances by deploying the Multi-AZ DB cluster. Learn more". It contains two buttons: "Create database" and "Or, Restore Multi-AZ DB Cluster from Snapshot". Below this, there are sections for Resources and Recommended for you.

Resources

You are using the following Amazon RDS resources in the US West (Oregon) region (used/quota)

DB Instances (0/40)	Parameter groups (0)
Allocated storage (0 TB/100 TB)	Default (0)
Increase DB instances limit <input type="button" value=""/>	Custom (0/100)
DB Clusters (0/40)	Option groups (0)
Reserved instances (0/40)	Default (0)
Snapshots (0)	Custom (0/20)
Manual	Subnet groups (0/50)
DB Cluster (0/100)	Supported platforms <input type="button" value=""/> VPC
DB Instance (0/100)	Default network vpc-0c549062274c6c2d7
Automated	
DB Cluster (0)	
DB Instance (0)	
Recent events (0)	
Event subscriptions (0/20)	

Recommended for you

- Test Your DR Strategy in Minutes**
Amazon Aurora Global Database now supports planned managed failover, making disaster recovery drills a breeze. [Learn more](#)
- Migrate SSRS to RDS for SQL Server**
Learn how you can migrate existing SSRS content to an Amazon RDS for SQL Server instance using a PowerShell module. [Learn more](#)
- Time-Series Tables in PostgreSQL**
Step-by-step guide to design high-performance time series data tables on Amazon RDS for PostgreSQL. [Learn more](#)
- Build RDS Operational Tasks**
Watch how to enable users to perform common tasks such as snapshots or restart DB instances in

Create database

Standard create; PostgreSQL

Amazon RDS

RDS > Create database

Create database

Choose a database creation method Info

Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

Aurora (MySQL Compatible) 

Aurora (PostgreSQL Compatible) 

MySQL 

MariaDB 

PostgreSQL 

Oracle 

Microsoft SQL Server 

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

Create database

Engine version (default); Free tier

The screenshot shows the 'Create database' wizard in the Amazon RDS console. The left sidebar lists various options: Dashboard (highlighted in orange), 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 (with a '0' badge), and Certificate update.

The main area starts with an 'Engine Version' dropdown set to 'PostgreSQL 14.6-R1'. Below it is a 'Templates' section with three options: 'Production', 'Dev/Test', and 'Free tier'. The 'Free tier' option is selected, indicated by a blue outline and a checked radio button. A tooltip for 'Free tier' states: 'Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.' Below this is a 'Availability and durability' section with a 'Deployment options' subsection. It lists three options: 'Multi-AZ DB Cluster - new', 'Multi-AZ DB instance (not supported for Multi-AZ DB cluster snapshot)', and 'Single DB Instance (not supported for Multi-AZ DB cluster snapshot)'. The first option is selected.

Create database

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

The screenshot shows the 'Amazon RDS' settings interface for creating a new DB instance. The left sidebar lists various RDS features: 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), and Certificate update. The main 'Settings' page has a 'DB instance identifier' field containing 'lab-db-demo'. Below it is a note about case-insensitivity and character constraints. Under 'Credentials Settings', the 'Master username' is set to 'postgres'. There's a note that master user credentials can be managed in AWS Secrets Manager. A warning box states that managing credentials in Secrets Manager may disable some RDS features. The 'Auto generate a password' checkbox is unchecked. The 'Master password' field contains masked text, and the 'Confirm master password' field also contains masked text.

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

Settings

DB instance identifier [Info](#)
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Credentials Settings

Master username [Info](#)
Type a login ID for the master user of your DB instance.

1 to 16 alphanumeric characters. First character must be a letter.

Manage master credentials in AWS Secrets Manager
Manage master user credentials in Secrets Manager. RDS can generate a password for you and manage it throughout its lifecycle.

If you manage the master user credentials in Secrets Manager, some RDS features aren't supported. [Learn more](#)

Auto generate a password
Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).

Confirm master password [Info](#)

Create database

Default; Uncheck “Enable storage autoscaling”

The screenshot shows the 'Instance configuration' section of the Amazon RDS console. In the 'Storage' section, the 'Allocated storage' field is set to 200 GiB. Below it, the 'Storage autoscaling' section is visible, containing a checkbox labeled 'Enable storage autoscaling' which is currently unchecked.

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

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class [Info](#)

- Standard classes (includes m classes)
- Memory optimized classes (includes r and x classes)
- Burstable classes (includes t classes)**

db.t3.micro
2 vCPUs 1 GiB RAM Network: 2,085 Mbps

Include previous generation classes

Storage

Storage type [Info](#)

General Purpose SSD (gp2)
Baseline performance determined by volume size

Allocated storage [Info](#)

200 GiB

The minimum value is 20 GiB and the maximum value is 6,144 GiB

Storage autoscaling [Info](#)

Provides dynamic scaling support for your database's storage based on your application's needs.

Enable storage autoscaling

Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

Create database

Public access Yes; other default

Amazon RDS X

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

Connectivity [Info](#)

Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

Network type [Info](#)
To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

IPv4
Your resources can communicate only over the IPv4 addressing protocol.

Dual-stack mode
Your resources can communicate over IPv4, IPv6, or both.

Virtual private cloud (VPC) [Info](#)
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-0c549062274c6c2d7)
4 Subnets, 4 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

DB subnet group [Info](#)
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

default

Public access [Info](#)

Yes
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

No
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) [Info](#)
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the

[CloudShell](#) [Feedback](#) [Language](#)

© 2023, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Create database

Password authentication

The screenshot shows the 'Create database' wizard in the Amazon RDS console. The left sidebar lists navigation options like Dashboard, Databases, and Query Editor. The main area is titled 'Database authentication' and contains three options:

- Password authentication**
Authenticates using database passwords.
- Password and IAM database authentication**
Authenticates using the database password and user credentials through AWS IAM users and roles.
- Password and Kerberos authentication**
Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

Below this, there are sections for 'Monitoring' (with a checkbox for 'Turn on Performance Insights') and 'Additional configuration' (with a checkbox for 'Enable Enhanced monitoring'). At the bottom, there's a summary section for 'Additional configuration' with a note about database options, encryption, backups, and CloudWatch Logs.

Create database

Make sure it is free tier; create database

The screenshot shows the 'Create database' wizard on the AWS RDS console. On the left, a sidebar lists various AWS services: Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Events, Event subscriptions, Recommendations (0), and Certificate update. The main content area is divided into sections:

- Additional configuration**: Describes database options, encryption turned on, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, and delete protection turned off.
- Estimated monthly costs**: Explains the Amazon RDS Free Tier availability for 12 months and lists free usage tiers:
 - 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance.
 - 20 GB of General Purpose Storage (SSD).
 - 20 GB for automated backup storage and any user-initiated DB Snapshots.

[Learn more about AWS Free Tier.](#)

When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the [Amazon RDS Pricing page](#).
- Important note**: A callout box states: "You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services."

At the bottom right are 'Cancel' and 'Create database' buttons.

Page footer:

- CloudShell
- Feedback
- Language
- © 2023, Amazon Web Services, Inc. or its affiliates.
- Privacy
- Terms
- Cookie preferences

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	CPU	Status	Class
lab-db-demo	-	Backing-up	db.t3.micro
Role	Current activity	Engine	Region & AZ
Instance	0 Connections	PostgreSQL	us-west-2c

Connectivity & security Monitoring Logs & events Configuration Maintenance & backups Tags

Connectivity & security

Endpoint & port	Networking	Security
Endpoint lab-db-demo.cvak west-2.rds.amazonaws.com	Availability Zone us-west-2c	VPC security groups default (sg-0114f93538) Active
Port 5432	VPC vpc-0c5490	Publicly accessible No
	Subnet group default-vpc-0c549062274c6c2d7	Certificate authority Info rds-ca-2019
	Subnets subnet-03ffccb85 subnet-0a6b0e subnet-	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

The screenshot shows the AWS Elastic Beanstalk Configuration page for the environment "weathermood-server-db-todo-dev". The left sidebar lists applications, environments, and recent environments. The main content area shows the "Configuration" tab selected under the "Environment: weathermood-server-db-todo-dev" section. The configuration is divided into several sections:

- Service access**: Configures service role and EC2 instance profile. It shows "aws-elasticbeanstalk-service-role" for the service role and "aws-elasticbeanstalk-ec2-role" for the EC2 instance profile.
- Networking, database, and tags**: Configures VPC settings and subnets. It indicates "No options configured".
- Instance traffic and scaling**: Customizes capacity and scaling. It shows "IMDSv1" for IMDS version and "EC2 Security Groups" for security groups.

Buttons for "Edit", "Cancel", "Review changes", and "Apply changes" are present at the top right of each section. A blue box highlights the "Configuration" link in the left sidebar and the "Edit" button in the "Instance traffic and scaling" section.

Database

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

The screenshot shows the Amazon RDS console interface. On the left, there's a sidebar with navigation links like Dashboard, Databases, Query Editor, etc. The main area shows the 'lab-db-demo' database details:

DB identifier	CPU	Status	Class
lab-db-demo	-	Backing-up	db.t3.micro
Role	Current activity	Engine	Region & AZ
Instance	0 Connections	PostgreSQL	us-west-2c

Below this, there are tabs for Connectivity & security, Monitoring, Logs & events, Configuration, Maintenance & backups, and Tags. The Connectivity & security tab is selected. It shows the following details:

Endpoint & port	Networking	Security
Endpoint lab-db-demo.cvak-west-2.rds.amazonaws.com	Availability Zone us-west-2c	VPC security groups default (sg-0114f93538) Active
Port vpc-0c5490	VPC Publicly accessible	

EB + RDS

Find EB - configuration - instances traffic edit

Elastic Beanstalk

Applications

Environments

Change history

Application: weathermood-server-db-todo

Application versions

Saved configurations

Environment: weathermood-server-db-todo-dev

Go to environment

Configuration **Configuration** (highlighted)

Events

Health

Logs

Monitoring

Alarms

Managed updates

Tags

Recent environments

weathermood-server-db-todo-dev

Elastic Beanstalk > Environments > weathermood-server-db-todo-dev > Configuration

Configuration

Cancel Review changes Apply changes

Service access Info

Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.

Edit

Service role: aws-elasticbeanstalk-service-role

EC2 instance profile: aws-elasticbeanstalk-ec2-role

Networking, database, and tags Info

Configure VPC settings, and subnets for your environment's EC2 instances and load balancer. Set up an Amazon RDS database that's integrated with your environment.

Edit

No options configured

Instance traffic and scaling Info

Customize the capacity and scaling for your environment's instances. Select security groups to control instance traffic. Configure the software that runs on your environment's instances by setting platform-specific options.

Edit

Instances	EC2 Security Groups
IMDSv1	

EB + RDS

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

The screenshot shows the AWS Elastic Beanstalk console for the environment 'weathermood-server-db-todo'. On the left, there's a sidebar with links for Applications, Environments, Change history, Application: weathermood-server-db-todo, Environment: weathermood-server-db-todo-dev, Recent environments (weathermood-server-db-todo-dev, weathermood-server-no-redux-dev), and Capacity Info.

The main content area is titled 'Instance metadata service (IMDS)'. It shows that both IMDSv1 and IMDSv2 are active. There's a checkbox labeled 'Deactivated' which is currently unchecked.

Below this is a section titled 'EC2 security groups' with the sub-section 'EC2 Security groups (3)'. A search bar is present above the table. The table lists three security groups:

	Group name	Group ID	Name
<input type="checkbox"/>	awseb-e-cvpe3krkzy-stack-AW...	sg-0371d34191e5f2abc	weathermood-server-db-todo...
<input type="checkbox"/>	awseb-e-cvpe3krkzy-stack-AW...	sg-07cf3784e309c4999	weathermood-server-db-todo...
<input checked="" type="checkbox"/>	default	sg-0114f935381120dcc	

At the bottom of the page, there's a 'Capacity' section with an 'Info' link and a note about configuring compute capacity and auto scaling settings.

Configure security group

RDS -> click VPC security groups

The screenshot shows the Amazon RDS console interface. On the left, there's a sidebar with navigation links like Dashboard, Databases, Query Editor, etc. The main area shows a database named 'lab-db-demo' with its summary details: DB identifier 'lab-db-demo', CPU ' - ', Status 'Backing-up', Class 'db.t3.micro', Role ' - ', Current activity '0 Connections', Engine 'PostgreSQL', and Region & AZ 'us-west-2c'. Below the summary, there are tabs for Connectivity & security, Monitoring, Logs & events, Configuration, Maintenance & backups, and Tags. The Connectivity & security tab is selected. It displays information about the endpoint and port, networking (Availability Zone 'us-west-2c', VPC 'vpc-0c5490'), and security (VPC security groups 'default (sg-0114f93538) Active'). A blue box highlights the 'Active' status of the security group.

Configure security group

Inbound rules -> edit inbound rules

The screenshot shows the AWS Security Groups console. At the top, there is a search bar with the placeholder "Filter security groups" and a search input field containing "search: sg-0114f935381120dcc". Below the search bar are buttons for "Actions", "Export security groups to CSV", and a prominent orange "Create security group" button. The main table lists one security group:

Name	Security group ID	Security group name	VPC ID	Description	Owner
-	sg-0114f935381120dcc	default	vpc-0c549062274c6c2d7	default VPC security gr...	785490623

Below the table, there are tabs for "Details", "Inbound rules", "Outbound rules", and "Tags". The "Inbound rules" tab is selected and highlighted with a blue box. A tooltip message "You can now check network connectivity with Reachability Analyzer" is displayed above the "Run Reachability Analyzer" button. The "Edit inbound rules" button is also highlighted with a blue box. The "Inbound rules" section shows one rule:

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-0225eae5fbe30c7ac	-	All traffic	All	All

Configure security group

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

EC2 > Security Groups > sg-0114f935381120dcc - default > Edit inbound rules

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
sgr-0225eae5fbe30c7ac	All traffic	All	All	Custom ▼ <input type="text"/> Q sg-0114f935381120dcc	Delete

[Add rule](#)

[Cancel](#) [Preview changes](#) **Save rules**

EB + RDS

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

The screenshot shows the 'Amazon RDS' interface with the 'Databases' tab selected. On the left, a sidebar lists various RDS features: Dashboard, Databases (selected), Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Events, and Event subscriptions.

The main content area is titled 'Security group' and shows a dropdown menu with 'Choose security groups' and a selected item 'default'. Below this is a 'Certificate authority' section with a dropdown menu containing 'rds-ca-2019'. Under 'Additional configuration', there is a 'Public access' section with two radio button options: 'Publicly accessible' (selected) and 'Not publicly accessible'. The 'Publicly accessible' option includes a note about EC2 instances connecting to the VPC. The 'Database port' section specifies the port number as '5432'.

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:
```

The screenshot shows the 'Connectivity & security' tab of an AWS RDS instance configuration page. On the left, there's a sidebar with links like Subnet groups, Parameter groups, Option groups, Custom engine versions, Events, and Event subscriptions. The main content area has tabs for Connectivity & security, Monitoring, Logs & events, Configuration, Maintenance & backups, and Tags. Under the Connectivity & security tab, there are three sections: Endpoint & port, Networking, and Security. The Endpoint & port section displays the endpoint URL: lab-db-demo.cvakngwhzvjq.us-west-2.rds.amazonaws.com. This URL is highlighted with a blue box. In the Networking section, the availability zone is listed as us-west-2c. In the Security section, it shows VPC security groups with one group named 'default (sg-0114f93538)' marked as Active.

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
```

Options:

-h: host

-p: port

-U: username

-d: dbname

The one you connected to

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

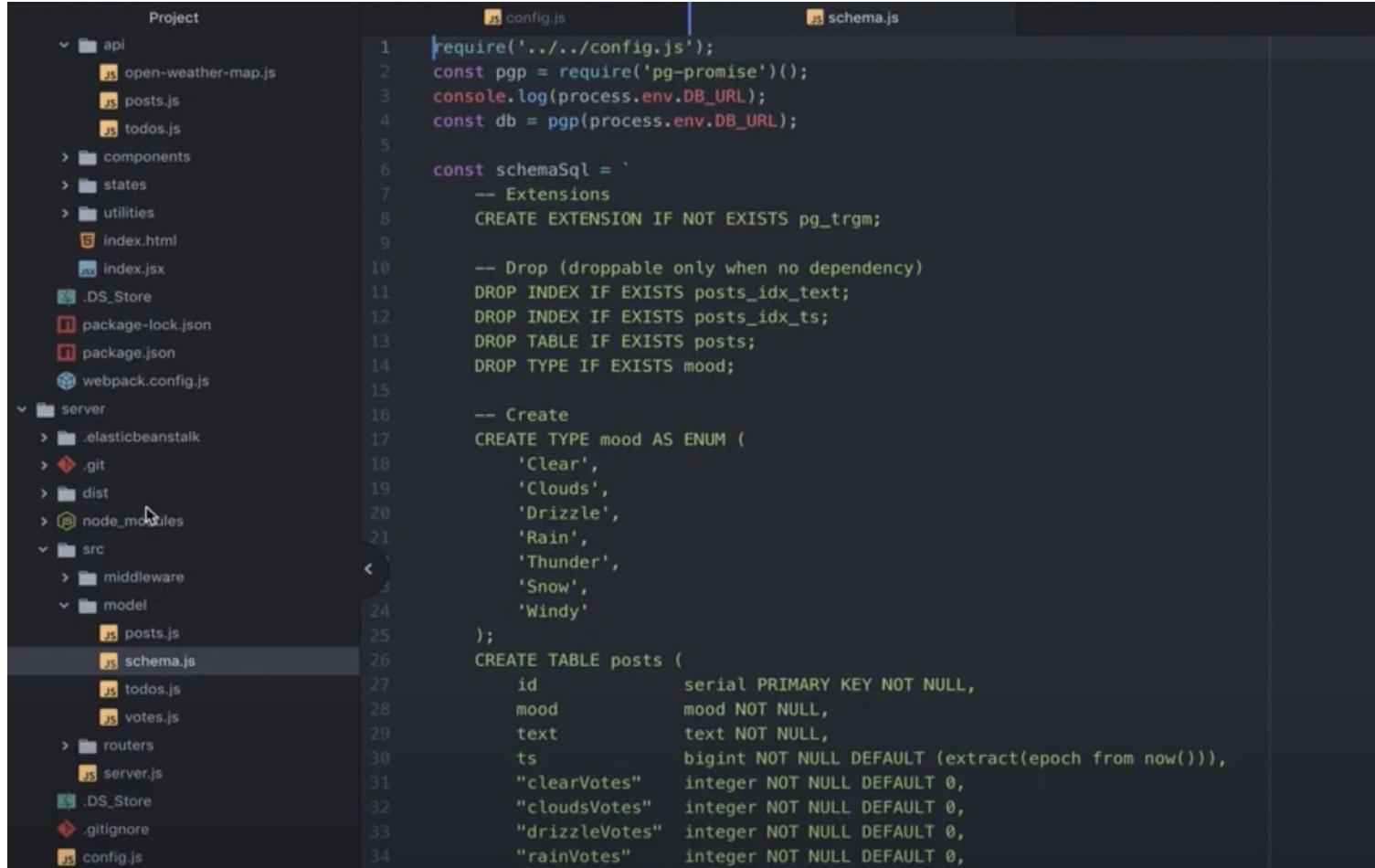
```
weathermood=> CREATE TABLE posts (
weathermood(>     id          serial PRIMARY KEY NOT NULL,
weathermood(>     mood         mood NOT NULL,
weathermood(>     text         text NOT NULL,
weathermood(>     ts           bigint NOT NULL DEFAULT (extract(epoch from now())),
weathermood(>     "clearVotes" 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



The image shows a code editor interface with a sidebar on the left displaying a project structure and two tabs on the right containing code snippets.

Project Structure:

- api
 - open-weather-map.js
 - posts.js
 - todos.js
- components
- states
- utilities
 - index.html
 - index.jsx
- .DS_Store
- package-lock.json
- package.json
- webpack.config.js

File Tabs:

- config.js
- schema.js

schema.js Content:

```
1  require('....config.js');
2  const pgp = require('pg-promise')();
3  console.log(process.env.DB_URL);
4  const db = pgp(process.env.DB_URL);
5
6  const schemaSql = `
7    -- Extensions
8    CREATE EXTENSION IF NOT EXISTS pg_trgm;
9
10   -- Drop (droppable only when no dependency)
11   DROP INDEX IF EXISTS posts_idx_text;
12   DROP INDEX IF EXISTS posts_idx_ts;
13   DROP TABLE IF EXISTS posts;
14   DROP TYPE IF EXISTS mood;
15
16   -- Create
17   CREATE TYPE mood AS ENUM (
18     'Clear',
19     'Clouds',
20     'Drizzle',
21     'Rain',
22     'Thunder',
23     'Snow',
24     'Windy'
25   );
26   CREATE TABLE posts (
27     id          serial PRIMARY KEY NOT NULL,
28     mood        mood NOT NULL,
29     text        text NOT NULL,
30     ts          bigint NOT NULL DEFAULT (extract(epoch from now())),
31     "clearVotes" integer NOT NULL DEFAULT 0,
32     "cloudsVotes" integer NOT NULL DEFAULT 0,
33     "drizzleVotes" integer NOT NULL DEFAULT 0,
34     "rainVotes" integer NOT NULL DEFAULT 0,
```

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

The screenshot shows the AWS Lambda console interface. On the left, there's a sidebar with navigation links: 'weathermood-server-dev', 'Go to environment', 'Logs', 'Health', 'Monitoring', 'Alarms', 'Managed updates', 'Events', and 'Tags'. The 'Configuration' link is highlighted in orange. The main area has a search bar at the top labeled 'Search for an option name or value'. Below it is a table with three columns: 'Category', 'Options', and 'Actions'. A single row is visible under the 'Software' category. The 'Options' column contains the following text:
Environment properties: NODE_ENV, RDS_DB_NAME, RDS_HOSTNAME, RDS_PASSWORD, RDS_PORT,
RDS_USERNAME
Log streaming: disabled
Proxy server: nginx
Rotate logs: disabled
X-Ray daemon: disabled

Application setting and deploy

A screenshot of a code editor window titled "config.js". The code is written in JavaScript and uses the dotenv library to handle environment variables. It attempts to read the NODE_ENV variable and switch between development, staging, or production environments based on its value. It also includes a catch block for errors and logs the configuration URL to the console.

```
1 require('dotenv').config();
2
3 try {
4     switch (process.env.NODE_ENV) {
5         case 'development':
6             process.env.DB_URL = `postgres://${process.env.PG_USERNAME}@${process.env.PG_HOST}:${process.env.PG_PORT}/${process.env.PG_DB}`;
7             break;
8         default:
9             // 'staging' or 'production'
10            process.env.DB_URL = `postgres://${process.env.RDS_USERNAME}:${process.env.RDS_PASSWORD}@${process.env.RDS_HOST}:${process.env.RDS_PORT}/${process.env.RDS_DB}`;
11            break;
12        }
13        // only used for debugging
14        console.log(`==DEBUG== process.env.DB_URL = ${process.env.DB_URL}`);
15    } catch (err) {
16        console.log(
17            err,
18            '\n\nError configuring the project. Have you set the environment variables?');
19    };
20}
21|
```

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

The screenshot shows the AWS Elastic Beanstalk configuration interface. On the left, there's a sidebar with navigation links: Applications, Environments, Change history, Application: s_weathermood-server_2021-db (with sub-links Application versions and Saved configurations), Environment: s-weathermood-server-2021-db-dev (with sub-links Go to environment, Configuration, Events, Health, Logs, Monitoring, Alarms, Managed updates, Tags), and a large blue box highlighting the Configuration link. The main content area is titled 'Load balancer' and contains settings for visibility (public), type (application), and shared status (false). Below this is a section titled 'Updates, monitoring, and logging' with an 'Edit' button highlighted by a blue box. The 'Monitoring' section shows system configuration (enhanced) and log streaming (Deactivated). The 'Updates' section shows managed updates (Deactivated) and deployment batch size (30). A vertical scrollbar is visible on the right side of the main content area.

Load balancer

Load balancer visibility	Load balancer type	Load balancer is shared
public	application	false

Store logs
Deactivated

Updates, monitoring, and logging Info

Define when and how Elastic Beanstalk deploys changes to your environment. Manage your application's monitoring and logging settings, instances, and other environment resources.

Edit

Monitoring

System	Cloudwatch custom metrics - instance	Cloudwatch custom metrics - environment
enhanced	—	—

Log streaming
Deactivated

Retention
7

Lifecycle
false

Updates

Managed updates	Update batch size	Deployment batch size
Deactivated	1	30

Application setting and deploy

Method 2: EB > config > edit > add properties

The screenshot shows the AWS Elastic Beanstalk configuration interface. On the left, a sidebar titled "Configuration" lists various management options: Events, Health, Logs, Monitoring, Alarms, Managed updates, and Tags. Below this is a section for "Recent environments". The main area is titled "Environment properties" and contains a table with five rows, each representing a property key and its value.

Key	Value
RDS_DBNAME	weat
RDS_HOSTNAME	lab-
RDS_PASSWORD	post
RDS_PORT	5432
RDS_USERNAME	post

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