

# AWS Bean Stalk Deploy MONO PERN Stack

1. Login to AWS Console create a free account if you don't have one already



Sign in

☒ Root user

Account owner that performs tasks requiring unrestricted access. [Learn more](#)

☐ IAM user

User within an account that performs daily tasks. [Learn more](#)

Root user email address

username@example.com

Next

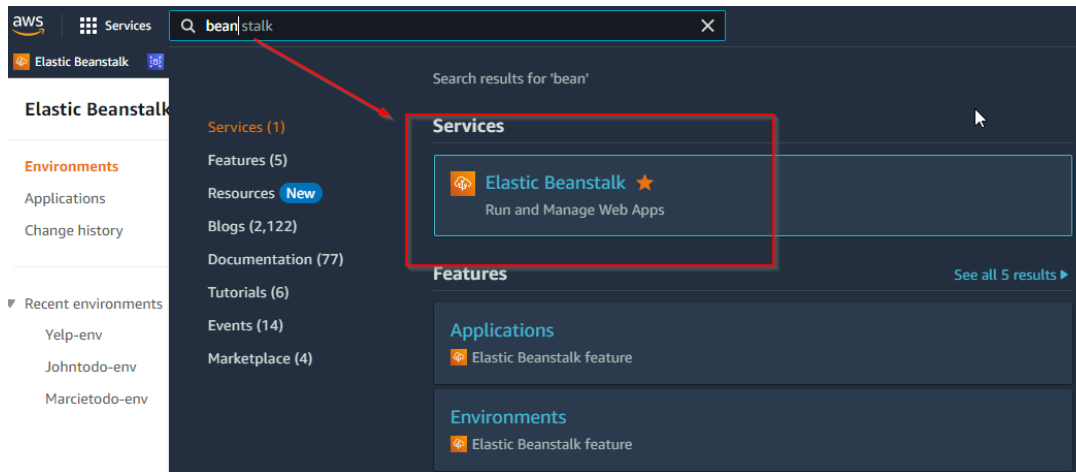
By continuing, you agree to the [AWS Customer Agreement](#) or other agreement for AWS services, and the [Privacy Notice](#). This site uses essential cookies. See our [Cookie Notice](#) for more information.

New to AWS?

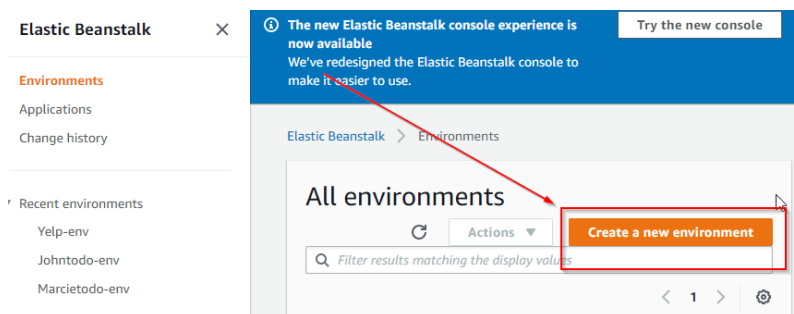
Create a new AWS account



2. Once logged in, you want to search and select the "Elastic Beanstalk" service.



3. Select "Create a new environment"



#### 4. Select “Web Server Environment”

### Select environment tier

Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications. Web servers are standard applications that listen for and then process HTTP requests, typically over port 80. Workers are specialized applications that have a background processing task that listens for messages on an Amazon SQS queue. Worker applications post those messages to your application by using HTTP.

☒ **Web server environment**  
Run a website, web application, or web API that serves HTTP requests.  
[Learn more](#)

☐ **Worker environment**  
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule.  
[Learn more](#)

Cancel Select

5. Give your application a name then scroll down and select “**Node.js**” as a platform  
Accept the other prefilled defaults for Node

### Application information

#### Application name

rest\_rant

Up to 100 Unicode characters, not including forward slash (/).

#### ► Application tags (optional)

### Platform

☒ **Managed platform**  
Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#)

☐ **Custom platform**  
Platforms created and owned by you.

#### Platform

Node.js

#### Platform branch

Node.js 16 running on 64bit Amazon Linux 2

#### Platform version

5.6.4 (Recommended)

6. Scroll down and select “upload your code”

Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Version label

Unique name for this version of your application code.

rest\_rant-source

Source code origin

Maximum size 512 MB

Local file

Public S3 URL

Choose file

No file uploaded

Application code tags

Cancel

Configure more options

Create environment

7. At this point you must prepare your local code to upload.  
Zip up all the code shown here in your backend folder.  
You can name the zip whatever you like.

Personal-Coding-Projects > test > rest-rant-monorepo-predeploy > backend >

Name	Date modified	Type	Size
config	2/7/2023 10:43 AM	File folder	
controllers	2/7/2023 10:43 AM	File folder	
migrations	2/7/2023 10:43 AM	File folder	
models	2/7/2023 10:43 AM	File folder	
node_modules	2/7/2023 11:41 AM	File folder	
public	2/7/2023 11:50 AM	File folder	
seeders	2/7/2023 10:43 AM	File folder	
.env	2/7/2023 11:53 AM	ENV File	1 KB
deploy-version-1	2/7/2023 12:19 PM	Compressed (zipp...	7,346 KB
index	2/7/2023 11:43 AM	JS File	1 KB
package	2/7/2023 11:48 AM	JSON File	1 KB
package-lock	2/7/2023 11:41 AM	JSON File	356 KB

8. Now browse to select that new zip file of your backend code with the AWS GUI and select **“Create environment”**

This may take several minutes to build out now.

• Upload your code

Upload a source bundle from your computer or copy one from Amazon S3.

Version label

Unique name for this version of your application code.

rest\_rant-source

Source code origin

Maximum size 512 MB

• Local file

○ Public S3 URL

Choose file

File name : **deploy-version-1.zip**

✓ File successfully uploaded

► Application code tags

Cancel

Configure more options

Create environment

9. Once it's complete don't worry about any status error in the console.

You still have to setup your database and environment variables.

Restrant-env

Restrant-env.eba-8imzj98s.us-east-1.elasticbeanstalk.com (e-ghphyev62j)

Application name: **rest\_rant**

Refresh

Action

Health



Severe

Causes

Running version

rest\_rant-source

Upload and deploy

Platform



Node.js 16 running on 64bit  
Amazon Linux 2/5.6.4

Change

## 10. Now create Your AWS database

Select Configuration and scroll down to **“Database”** and select **edit**

The screenshot shows the AWS Elastic Beanstalk console. On the left, the 'Configuration' tab is selected in the sidebar. The main content area lists various configuration options for the environment, including Load balancer, Rolling updates and deployments, Security, Monitoring, Managed updates, Notifications, and Network. At the bottom, the 'Database' option is highlighted with a red box, and a red arrow points to its 'Edit' button.

- Select **“Postgres”** as the engine
- Select **“db.t3.micro”** as the instance
- (Very important being if you don't select this instance type you may see some high charges on your AWS bill. This is the “free level” instance)
- Select a **username** and **password** for your AWS database
- Select **“delete”** as a detention policy. (again, to save on possible charges not included in the “free tier”)
- Select **“Apply”** this will take some time to re-build again

The screenshot shows the 'Database' configuration page in the AWS Elastic Beanstalk console. The form fields are as follows:

- Engine:** postgres
- Engine version:** 13.7
- Instance class:** db.t3.micro (labeled as 'Very Important')
- Storage:** 5 (Choose a number between 5 GB and 1024 GB)
- Username:** postgres
- Password:** \*\*\*\*\*
- Availability:** Low (one AZ)
- Database deletion policy:** Delete (selected)

The 'Delete' option under the deletion policy is highlighted with a red box. A red arrow points to the 'Apply' button at the bottom right.

11. Now you have to setup access to the database from any IP address

- Click **Configuration** again and scroll down to the database section.
- Click on the link to the new database **“Endpoint”**

Configuration

Endpoint: [awseb-e-ufidyzy6mq-stack-awsebrdsdatabase-pjhxrjvir3u8.cgo2aecz6xww.us-east-1.rds.amazonaws.com:5432](#)

Next you will see your list of databases. Click on the one you just created

awseb-e-ufidyzy6mq-stack-awsebrdsdatabase-pjhxrjvir3u8

- Scroll down to you see **security groups** and click on the **“inbound”** one

Security group rules (2)

Security group	Type	Rule
rds-awseb-e-daju4jqnaw-stack-awsebrdsdbsecuritygroup-bb9g1wq7ttg1-hwub (sg-0e4288aa84f4db6b5)	EC2 Security Group - Inbound	sg-07ed12b07a9f251da
rds-awseb-e-daju4jqnaw-stack-awsebrdsdbsecuritygroup-bb9g1wq7ttg1-hwub (sg-0e4288aa84f4db6b5)	CIDR/IP - Outbound	0.0.0.0/0

- In the **“actions”** drop down select **“Edit inbound rules”**

Security Groups (1/1) Info

Actions

Edit inbound rules

Name	Security group ID	Security group	Description
-	sg-0e4288aa84f4db6b5	rds-awseb-e-daju4jqna...	vpc-06f6bf47f652c6d20

- Select “add rule”, add “custom TCP” set the port as needed, set it to “Anywhere ipv4” and select “Save Rules”

Edit inbound rules [Info](#)

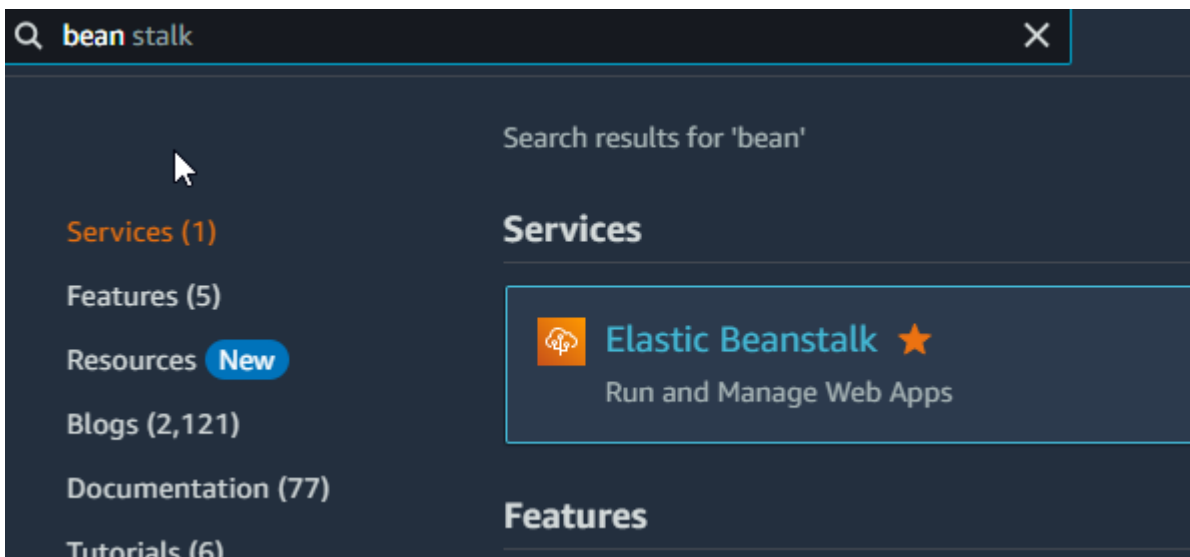
Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	
sgr-063e52fbb02361915	PostgreSQL	TCP	5432	Custom		Delete
sgr-0b359b161c50a7dcd	Custom TCP	TCP	5432	Custom		Delete

[Add rule](#)

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

- Now search and go back to the “bean stalk” settings.



## 12. Time to add your system environment variables

- First you need to copy that database endpoint that you created under **configuration** (You will need this link to enter as a variable but **remove the :5432 port number from the end**)

▼ Restrant2-env

[Go to environment](#) [Configuration](#) [Logs](#) [Health](#) [Monitoring](#) [Alarms](#) [Managed updates](#) [Events](#) [Tags](#)

▼ Recent environments

Restrant2-env

Security	EC2 key pair: -- IAM instance profile: aws-elasticbeanstalk-ec2-role Service role: arn:aws:iam::873403394234:role/aws-elasticbeanstalk-service-role
Monitoring	CloudWatch Custom Metrics-Environment: CloudWatch Custom Metrics-Instance: Health event log streaming: disabled Ignore HTTP 4xx: disabled Ignore load balancer 4xx: disabled System: Enhanced
Managed updates	Managed updates: disabled
Notifications	Email: --
Network	This environment is not part of a VPC.

Endpoint: [awsb-e-daju4jqnaw-stack-awsebrdsdatabase-yh8vmuuvotak.cgo2aecz6xww.us-east-1.rds.amazonaws.com:5432](#)

Availability: Low (one AZ)

Engine: postgres

- Next go to **configuration “software”** and select **“edit”**

**Elastic Beanstalk** X

Environments  
Applications  
Change history

Rest Rant 2  
Application versions  
Saved configurations

Restrant2-env  
Go to environment

**Configuration**

Logs  
Health  
Monitoring  
Alarms  
Managed updates  
Events

**Configurations** Cancel Review changes Apply all changes

Table View

Search for an option name or value

Category	Options	Actions
<b>Software</b>	Log streaming: disabled Proxy server: nginx Rotate logs: disabled X-Ray daemon: disabled	<b>Edit</b>
Instances	EC2 security groups: awseb-e-daju4jqnaw-stack-AWSEBSecurityGroup-5SXPC70MYQN3 IMDSv1: disabled IOPS: container default Monitoring interval: 5 minute Root volume type: container default Size: container default Throughput: container default AMI ID: ami-0917af17d9ee3d7d4 Availability Zones: Any Breach duration: 5 Capacity rebalancing: disabled Environment type: load balancing, auto scaling	<b>Edit</b>

- Scroll down to Environment properties and enter your env variables for production
- It's important that you include that **NODE\_ENV=production** setting as well as that long link from your newly created database. *Note that is case sensitive match what you used in your code*
- The other settings should match what you used previously when creating the DB on AWS
- The DB name on the AWS side (RDS\_DB\_NAME should be **“ebdb”**, use that no matter what you named it locally for development

This will now trigger another Beanstalk rebuild over several minutes

**Environment properties**

The following properties are passed in the application as environment properties. [Learn more](#)

Name	Value
<b>RDS_HOSTNAME</b>	awseb-e-daju4jqnaw-stack-awsebrdsdatabase-yh8vmuuvotak.cgo
<b>NODE_ENV</b>	Production
RDS_USERNAME	postgres
RDS_PASSWORD	postgres
RDS_DB_NAME	Rest_Rant
RDS_PORT	5432

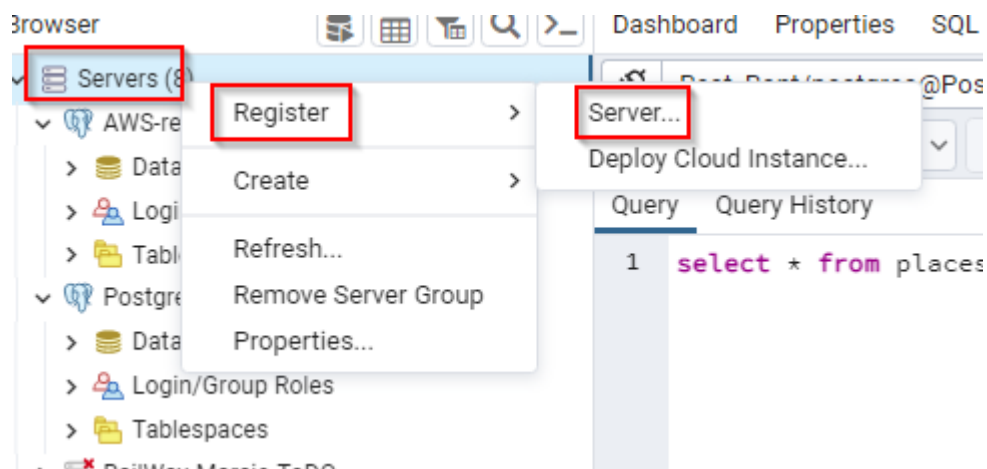
Cancel Continue **Apply**



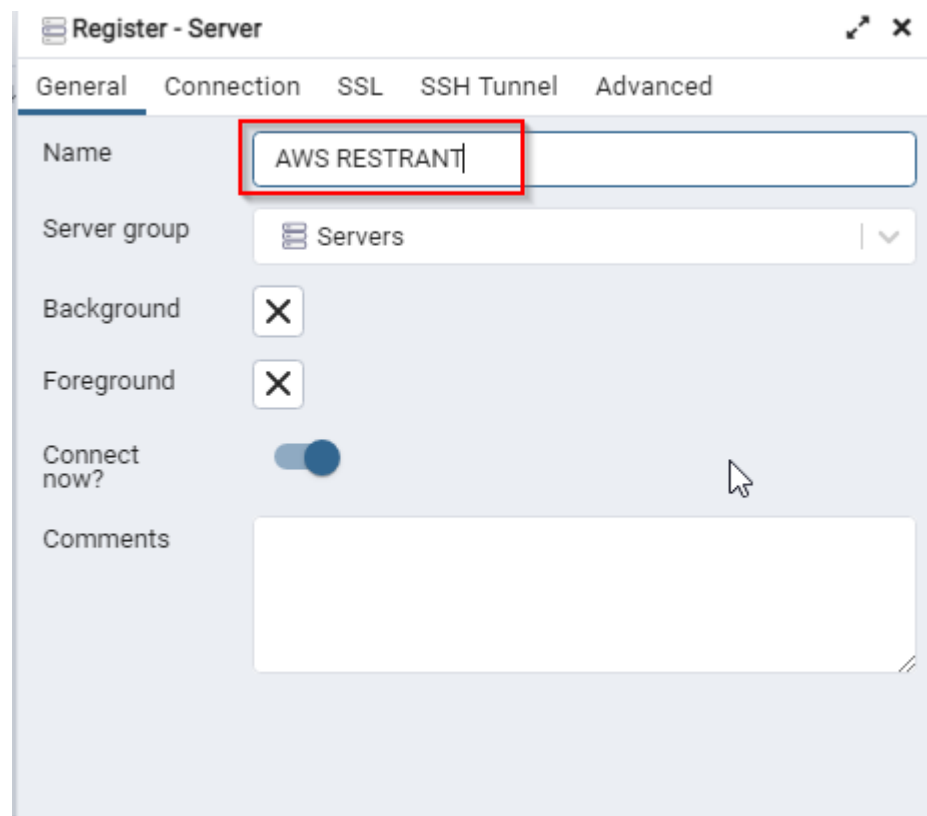
## Setup the Database on AWS with your backup file

In the local PGADMIN tool connect to your AWS database

- First Right click on servers and select “**add new server**”



- Give the server a name



- Click **“Connection”** and enter The Server Details to connect remember to remove the port from the end again (:5432)

**Register - Server**

General **Connection** SSL SSH Tunnel Advanced

Host name/address: awseb-e-daju4jqnaw-stack-awsebrdsdatabase-yh8vm

Port: 5432

Maintenance database: postgres

Username: postgres

Kerberos authentication? ☐

Password: .....

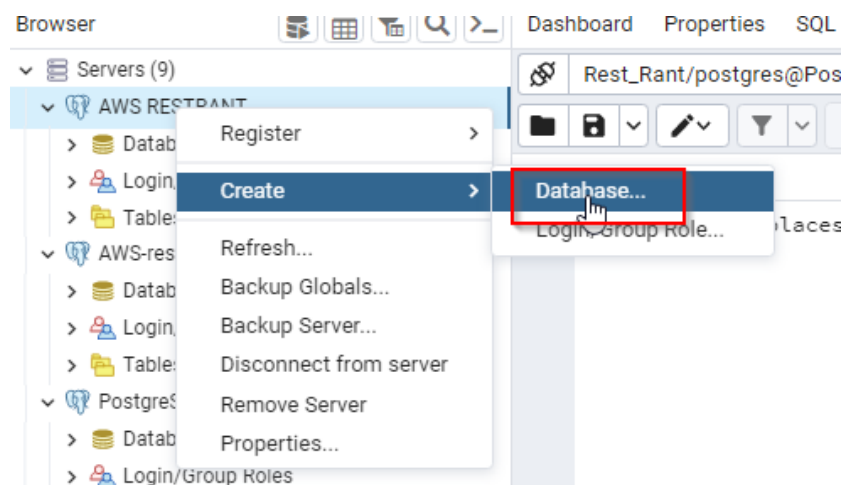
Save password? ☐

Role:

Service:

Close Reset Save

- Next right click on the newly connected server and select **“create database”**




- The database name in PGAdmin should match the AWS Environment entry of **“ebdb”**



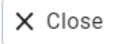


Create - Database

General Definition Security Parameters Advanced SQL

Database

Owner  postgres

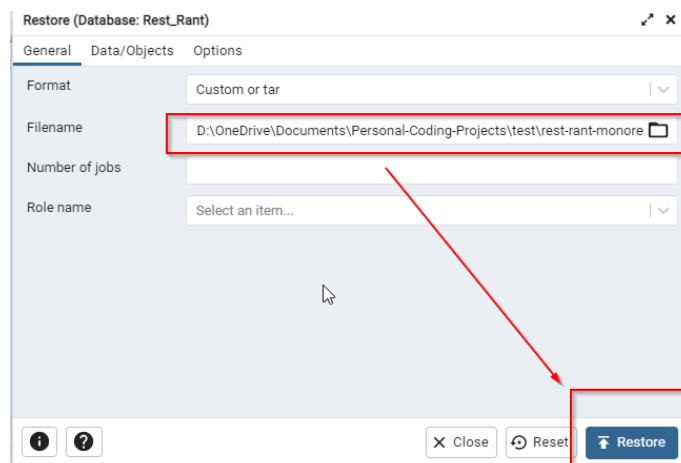
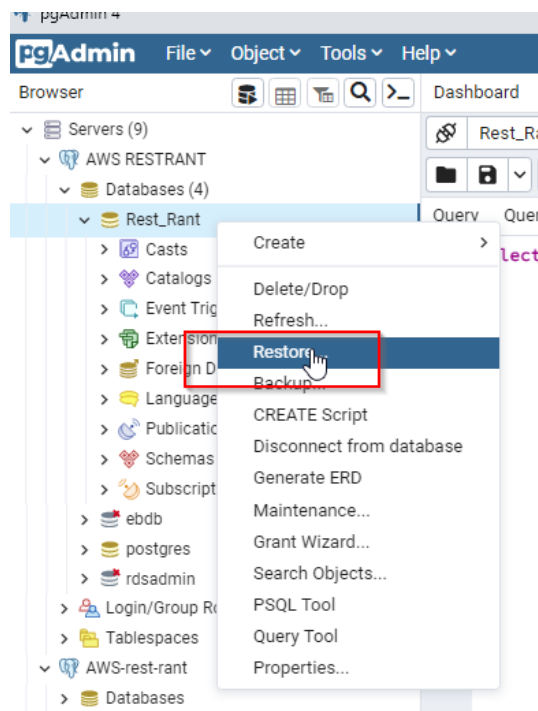
Comment

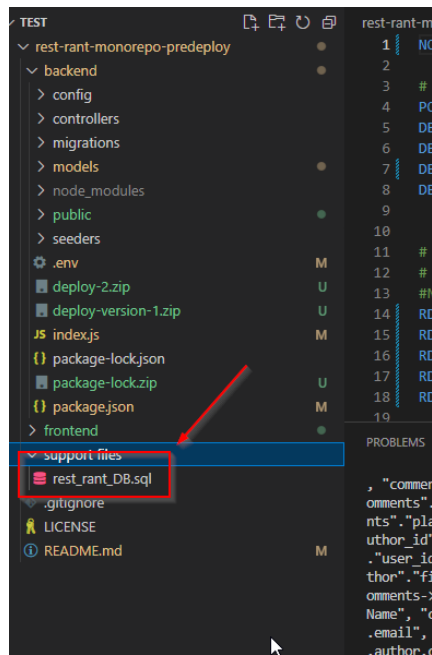
    

Name	Value
NODE_ENV	production
RDS_DB_NAME	ebdb
RDS_HOSTNAME	awseb-e-nbdvwm3tyg-stack-awsebrdsdatabase-tfrygq1xqgib.cgo2
RDS_PASSWORD	postgres
RDS_PORT	5432
RDS_USERNAME	postgres

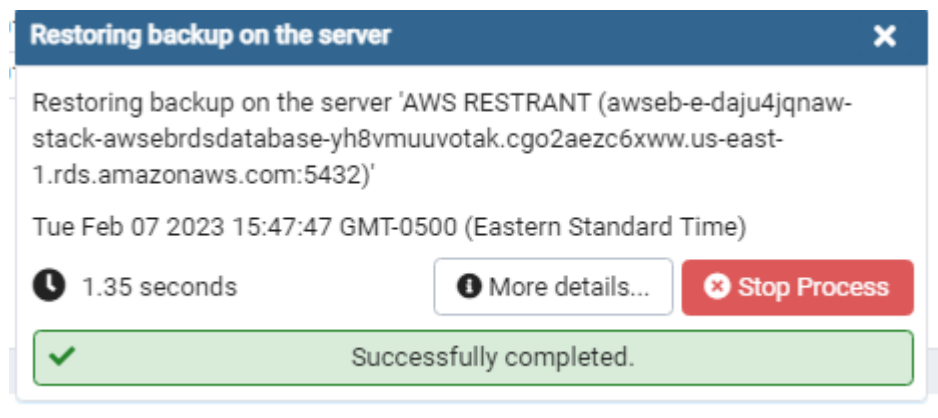
- Right Click on the newly created database and select **Restore**
- Browse to that same backup sql file we used earlier when creating the local database

- 





- You can go test your app now it should be ready for production



Click the link in AWS Beanstalk and test your app

