

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

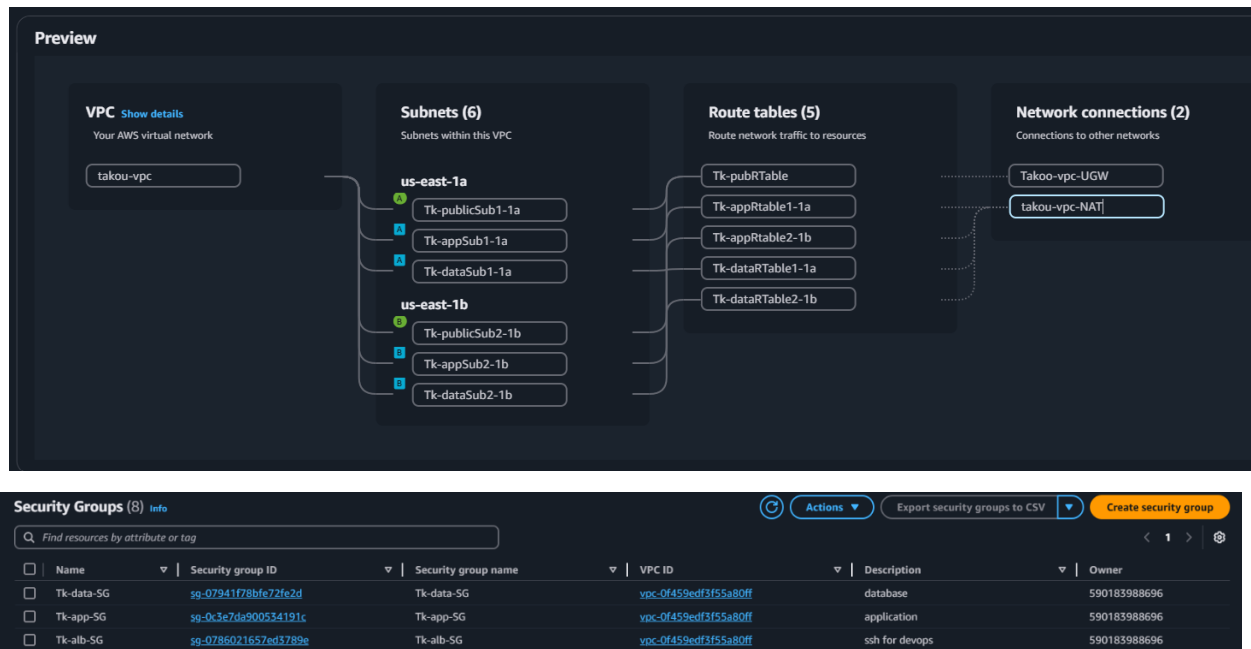
Step-by-Step Guide to Setting Up a Fully Serverless 3-Tier Architecture for Hosting a Dynamic Docker Web Application for a Business

Key:

- ECS: Fargate
- Lambda
- RDS
- IAM
- Secret manager
- CloudFormation
- NAT, VPC, Elastic IP, ALB
- Docker image

Step 1:

- Reproduce the infrastructure below from VPC to Security group



You will need three SG and NACLs.

- **Alb SG public:** Http and Https from anywhere
- **App-SG:** http and Https from “alb-SG”

Documented by **N.T Samuel**

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

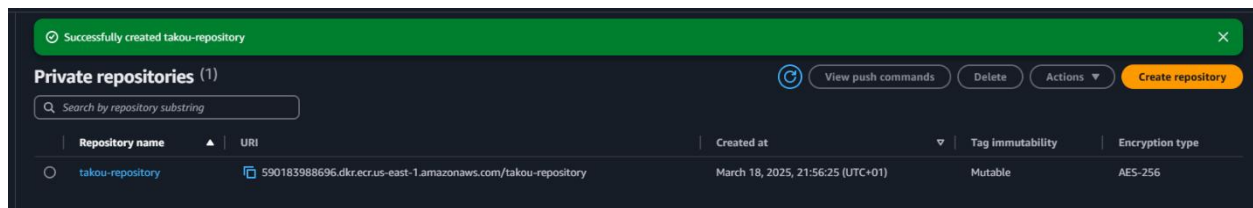
- **Data-SG:** MySQL and MySQL from app-SG & from data-SG itself (to be used by a lambda function in the database subnet.)

You will also need a NAT as seen above

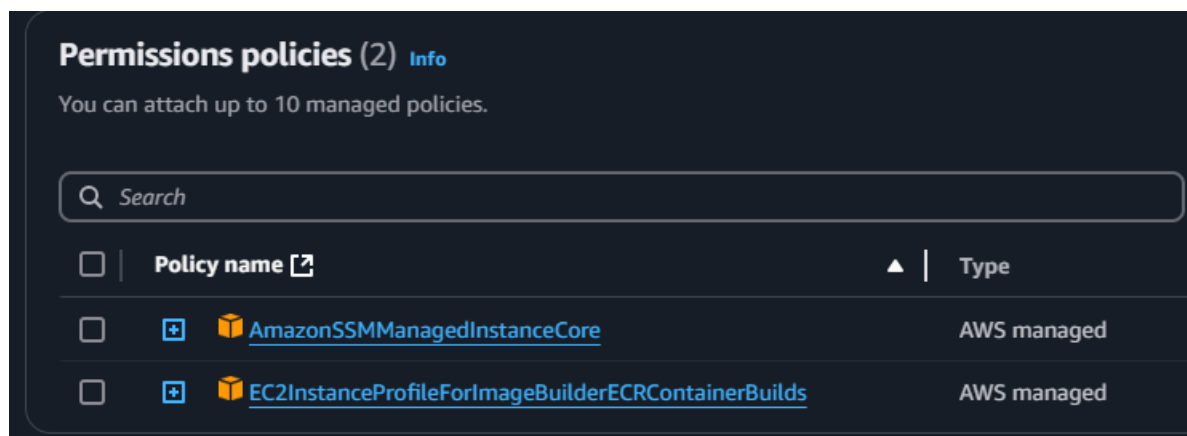
- NB: An Elastic IP will be allocated for the NAT above. Remember to release the Elastic IP at the end of this project.

Step 2:

- **Set up ECS repository:** Access **ECR** on your management console and create a private repository with AES-256 encryption type.



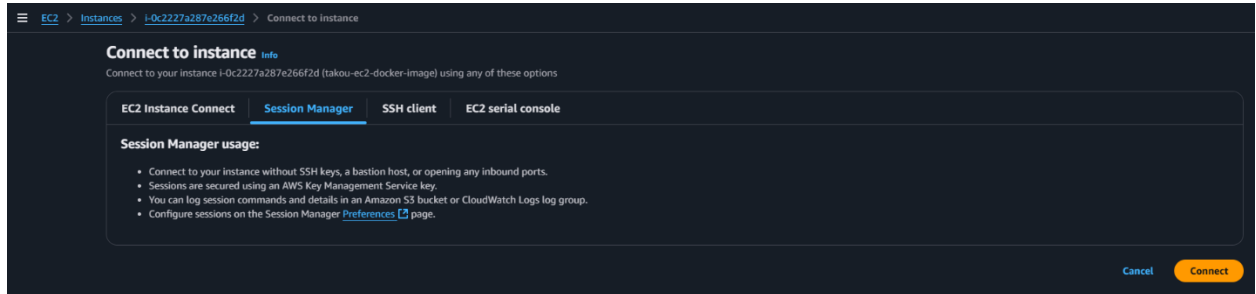
- **Set up docker image:** Create an IAM role allowing the following permission. (A docker image can be gotten from your developers or anywhere. A docker is a container holding something (An application). For this case our docker will be a dynamic website image.)



- Launch a Linux EC2 to setup our docker image: Attached the created role above
 - Use Linux 2023 or Linux 2
 - No keypair
 - Put it in the “appSubnet”

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

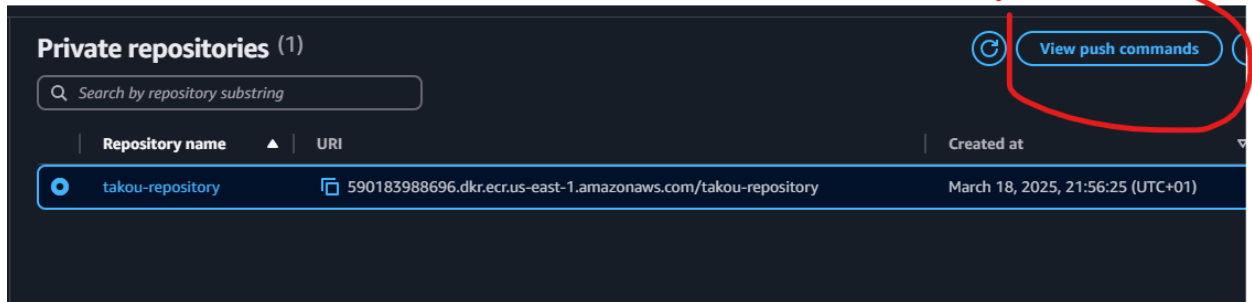
- **No public IP**
- **Assign to it the default SG for your VPC**
- **NB: Make sure to attached the IAM role**
- Connecting to our EC2 we will use “session manager” so SSH is not allowed.



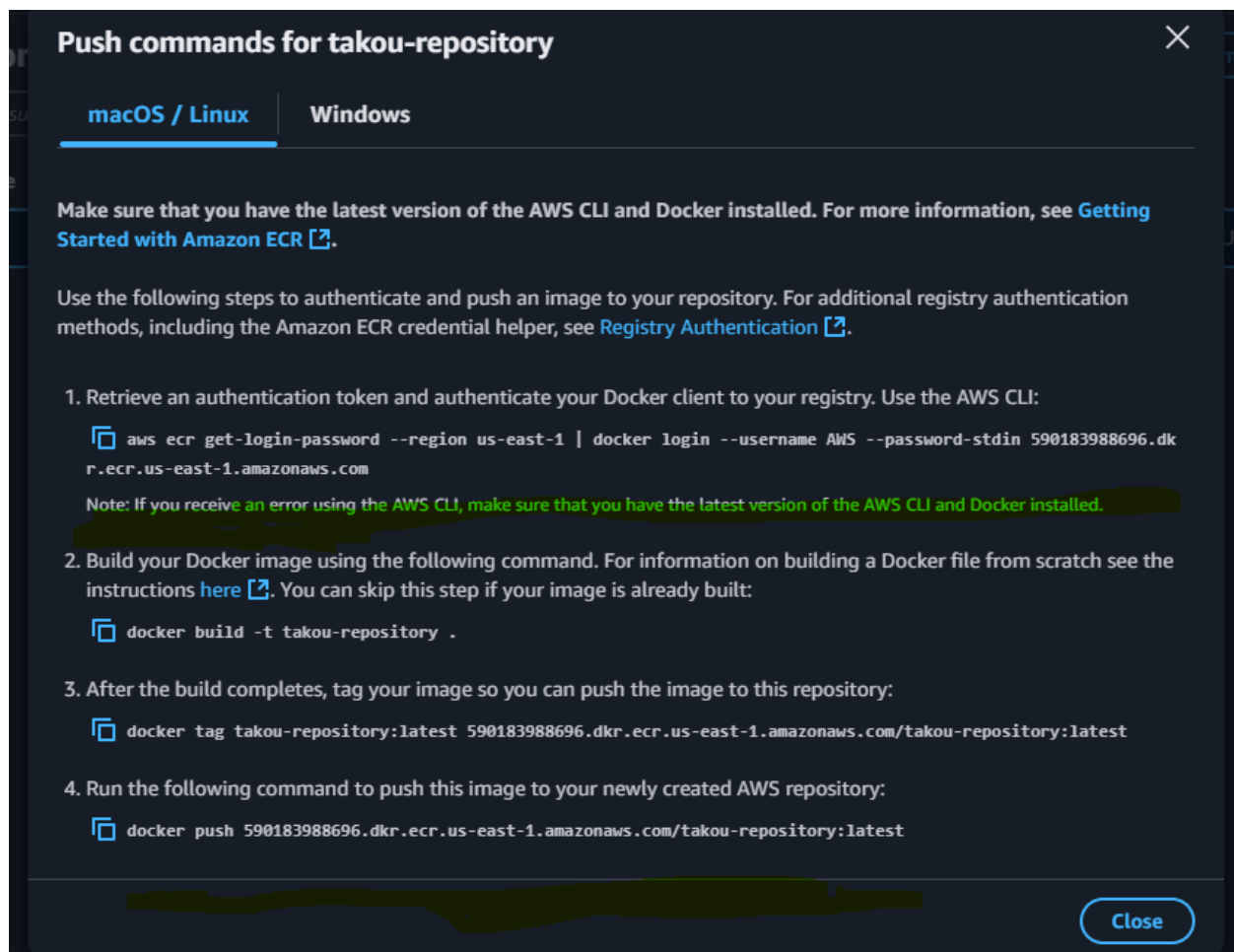
- issue the following command to install docker
 - `sudo su – ec2-user`
 - Make sure you are working in the: /home/ec2-user by using the “`pwd`” command.
 - Update instance: “`sudo yum update`”
 - Install docker: “`sudo yum install docker -y`”
 - Start docker: “`sudo service docker start`”
 - Add current user to docker group: “`sudo usermod -a -G docker ec2-user`”
 - Exit and connect back to EC2
 - Change user to: “`sudo su – ec2-user`” the default linux user is “ec2-user”. You can create a new one and use.
 - Verify your directory: (/home/ec2-user) “`pwd`”
 - Download docker image from my repository: “`wget https://github.com/synthetico/miniprojects/raw/refs/heads/main/amazon-ecs-mini-project/ritual-roast-code.zip`”
 - The link above is my repository holding the docker image (dynamic website). You can get any dynamic website and use.
 - Verify it downloaded using: `ls`
 - Unzip file: `unzip filename`
 - `cd` into the unzipped directory
 - `ls` to view files
 - Build docker image to your ECR: “`docker build -t ecr-url .`”
 - **NB: the is a dot at the end of the command above**
 - It will also install MySQL, copy source code t html, export image and layers.
 - When it's done very image: “`docker images`”

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

- Push docker into ECR
 - You can get these commands on your ECR dashboard



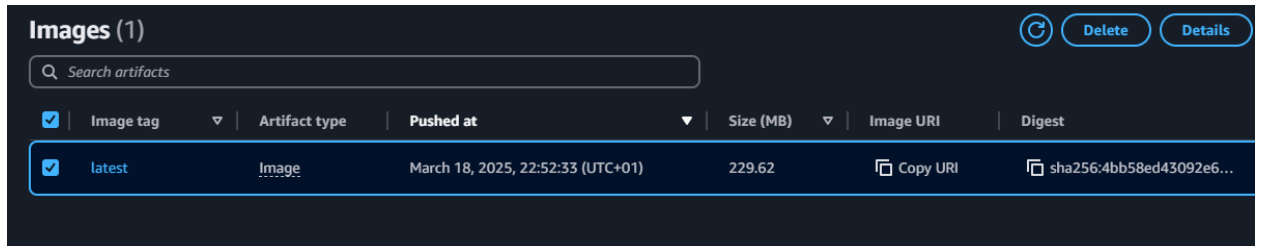
- From the image below, run the commands in green (1 and 4)



Documented by [N.T Samuel](#)

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

- The second command above was done already and the third is just tagging.
- If no IAM role was attached, the two commands above won't work
- Access your ECR and verify docker image in it.

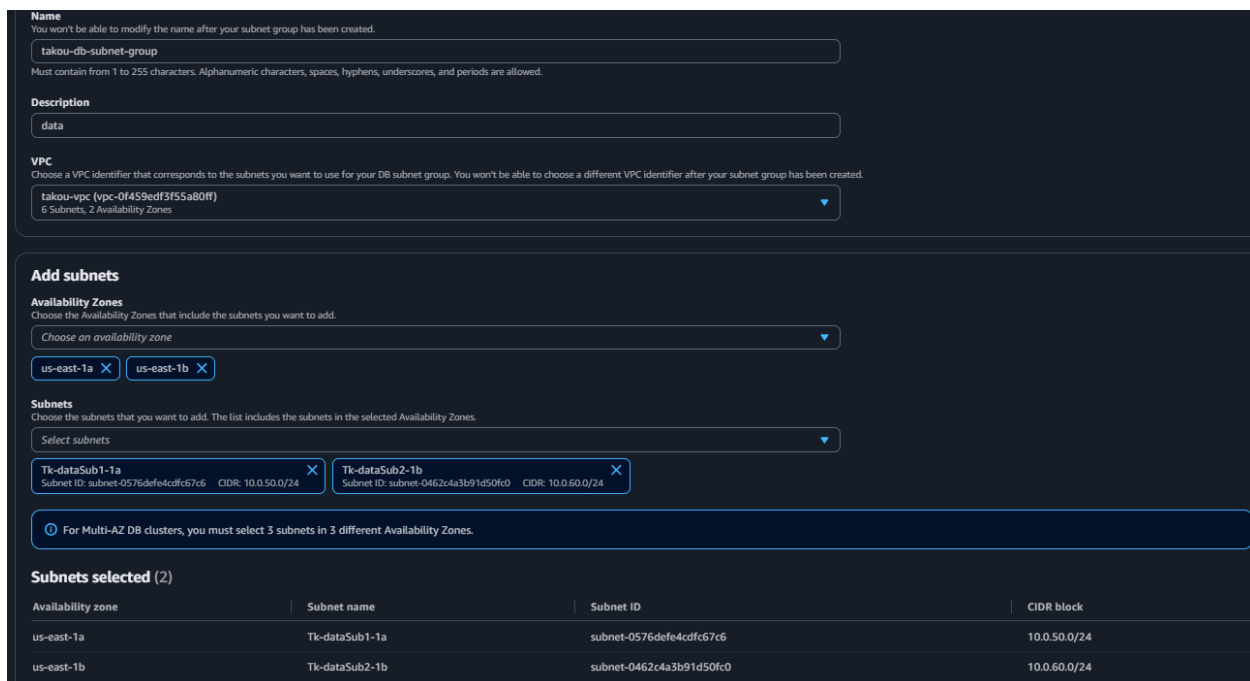


<input checked="" type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input checked="" type="checkbox"/>	latest	Image	March 18, 2025, 22:52:33 (UTC+01)	229.62	Copy URI	sha256:4bb58ed43092e6...

NB: the EC2 can be deleted as it's no more needed.

Step 3:

- Create a MySQL database: In production use **multi-AZ** but for this lab, we will use free tier
 - Create db subnet group



Name
You won't be able to modify the name after your subnet group has been created.
takou-db-subnet-group
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.

Description
data

VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
takou-vpc (vpc-0f459edf3f55a80ff)
6 Subnets, 2 Availability Zones

Add subnets

Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
us-east-1a X us-east-1b X

Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
Tk-dataSub1-1a Subnet ID: subnet-0576defe4cdfc67c6 CIDR: 10.0.50.0/24 X
Tk-dataSub2-1b Subnet ID: subnet-0462c4a3b91d50fc0 CIDR: 10.0.60.0/24 X

ⓘ For Multi-AZ DB clusters, you must select 3 subnets in 3 different Availability Zones.

Subnets selected (2)

Availability zone	Subnet name	Subnet ID	CIDR block
us-east-1a	Tk-dataSub1-1a	subnet-0576defe4cdfc67c6	10.0.50.0/24
us-east-1b	Tk-dataSub2-1b	subnet-0462c4a3b91d50fc0	10.0.60.0/24

- Create database using **MySQL engine-free tier-data-SG**.

Documented by **N.T Samuel**

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

- Name your database and give it a password

Create database info


Choose a database creation method


☒ **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)



☒ **MySQL**



☐ MariaDB


☐ Microsoft SQL Server


☐ Aurora (PostgreSQL Compatible)


☐ PostgreSQL


☐ Oracle


☐ IBM Db2


Edition
☒ MySQL Community

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

The screenshot shows the 'Settings' page for an Amazon RDS database instance. The 'DB instance identifier' is set to 'takoudb'. Under 'Credentials Settings', the 'Master username' is 'admin'. In the 'Credentials management' section, 'Self managed' is selected, indicating that the user will create their own password. The 'Master password' field shows a strength indicator of 'Very strong'. The 'Confirm master password' field is also visible.

- make sure to select your VPC, subnet group, database SG and set database name under “**Additional Configuration**” and create

The screenshot shows the 'Additional configuration' page. Under 'Database options', the 'Initial database name' is 'takoudb'. The 'DB parameter group' is set to 'default:mysql8.0' and the 'Option group' is 'default:mysql-8-0'. In the 'Backup' section, 'Enable automated backups' is checked, with a note that it creates a point-in-time snapshot of the database.

Step 4:

- We will set up secret manager to dynamically retrieve database secret information. This will help in secret authentication.
- Configure AWS secret manager
 - Access **secret manager** and store a new secret

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

Choose secret type

Secret type [Info](#)

☒ Credentials for Amazon RDS database ☐ Credentials for Amazon DocumentDB database ☐ Credentials for Amazon Redshift data warehouse

☐ Credentials for other database ☐ Other type of secret
API key, OAuth token, other.

Credentials [Info](#)

User name
admin

Password
Bushinfo326
☒ Show password

Encryption key [Info](#)
You can encrypt using the KMS key that Secrets Manager creates or a customer managed KMS key that you create.
aws/secretsmanager [Add new key](#)

Database [Info](#)

Search instances

DB instance	DB engine	Status	Creation date (UTC)
takoudb	mysql	available	March 18, 2025 at 22:06:01

- name secret
- Set up rotation period

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

- Since the rotation will be managed by a lambda function, create a rotation function as seen below and store secret

Configure AWS Secrets Manager to rotate this secret automatically.

☒ Automatic rotation

Rotation schedule [Info](#)

☒ Schedule expression builder ☐ Schedule expression

Time unit **Days**

Days 5

Window duration - optional

4h

Enter the time in hours.

☒ Rotate immediately when the secret is stored. The next rotation will begin on your schedule.

Rotation function [Info](#)

☒ Create a rotation function ☐ Use a rotation function from your account

Lambda rotation function

Secrets Manager adds the prefix 'SecretsManager' to your function name.

SecretsManager

Function name is required. Rotation function name including prefix must be maximum 64 alphanumeric characters, hyphens, and underscores.

Rotation strategy [Info](#)

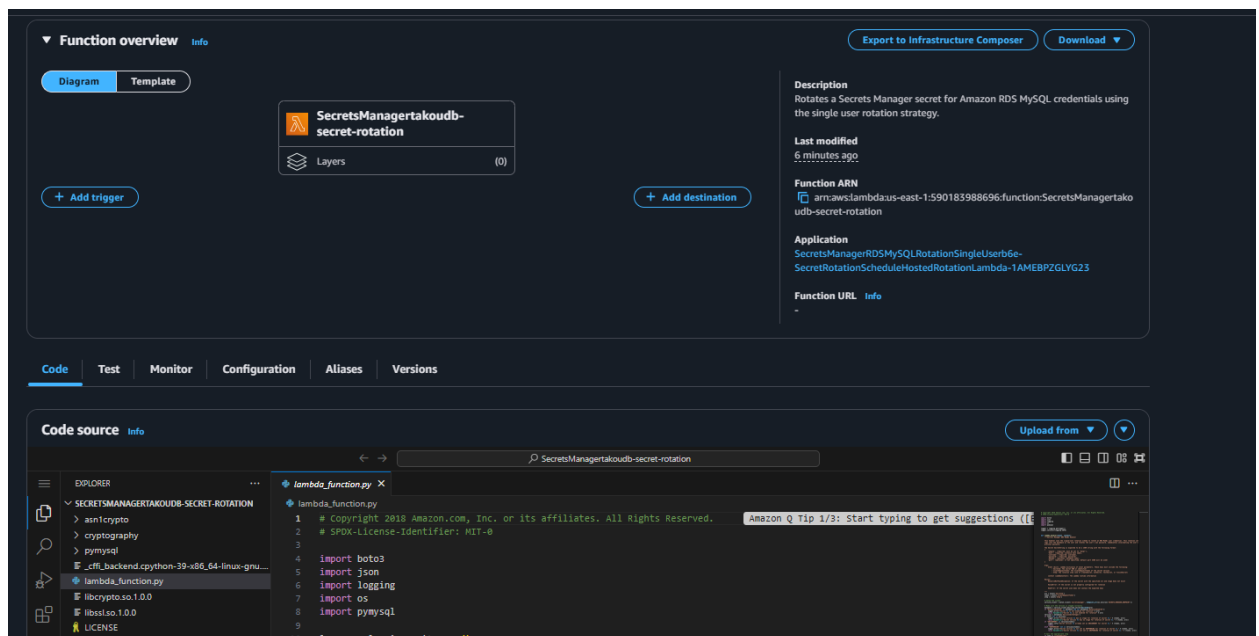
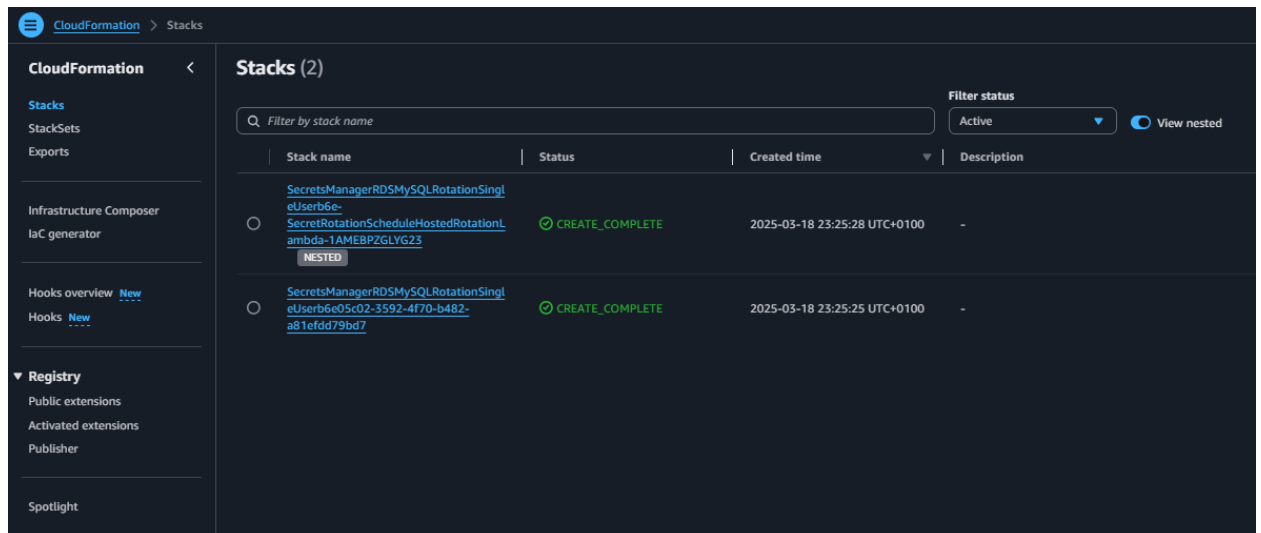
☒ Single user
The user must have permission to update their password.

☐ Alternating users
This strategy clones the initial user and stores both sets of credentials in one secret. One set of credentials is always valid. You must provide admin credentials in a separate secret.

- ***NB: Based on the rotation period configured, lambda will rotate your secrets respecting that period for you and your team. So a lambda function will be created.***

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

- At this stage **CloudFormation** will come up and set up the lambda, setup the secret manager and establish communication via the data SG between lambda and database.



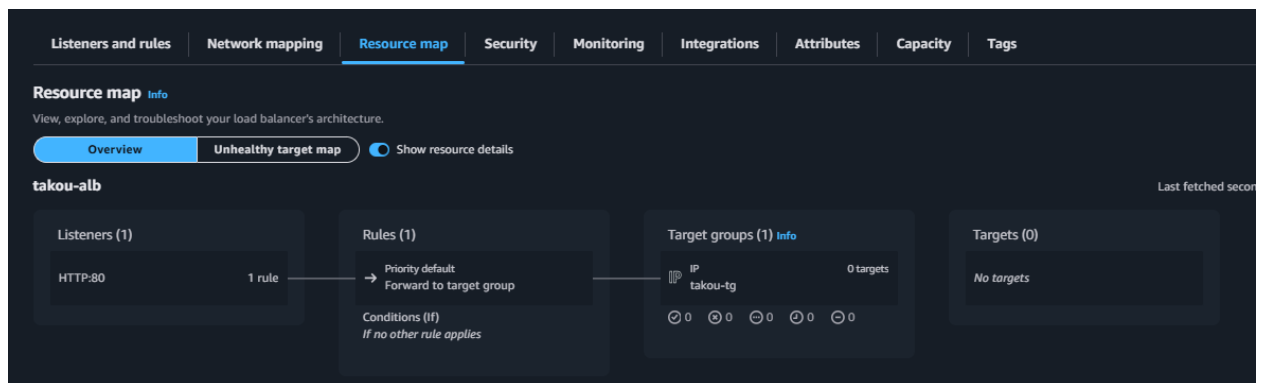
Step 5:

- Create an ALB
 - Create a target group
 - **IP add** as target because we are using EC2 Farget
 - Name target group and select Http as protocol
 - Select VPC

Documented by **N.T Samuel**

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

- Assign health check file: “health.html”
- Click next
- Make sure your VPC is selected, click remove on the subnet Ip address below
- Review and create
- **Create ALB**
- Name it
- Select “**internet facing**”
- Select VPC and public subnets
- Select **alb-SG**
- Set listener to **https**
- Select your SSL certificate
- Add a second listener with **http** as protocol **redirecting** it to https port 443.
- Create alb

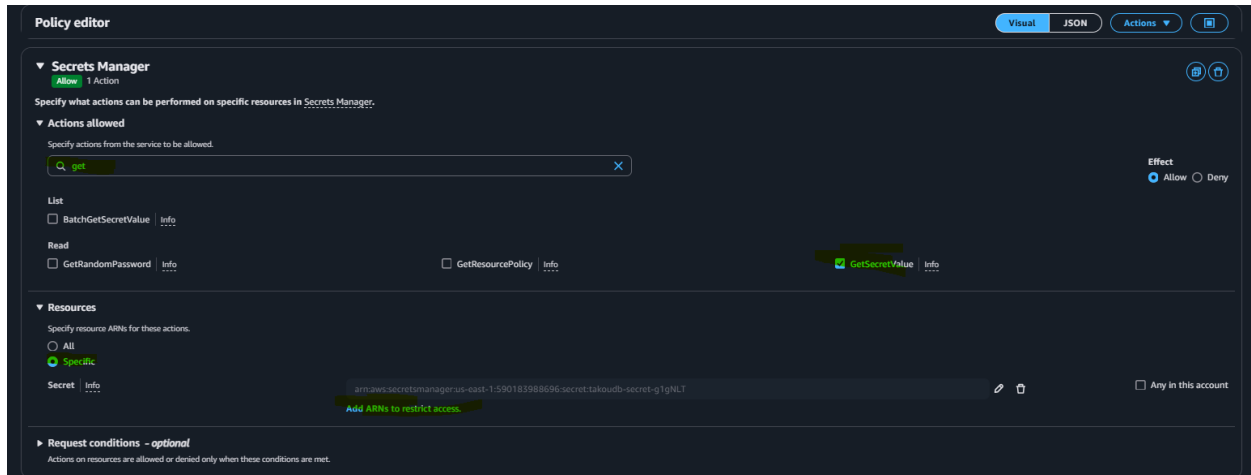


- Above is the ALB movement.
- Create an IAM role that has multiple policies to allow **secret manager** reference, **CloudWatch logging** and **docker image copying**.
 - Access **IAM dashboard**
 - Create a policy for secret manager
 - Click on policies
 - For Action, search for get and select “**Getsecretvalue**”
 - Under resources be specific and attach the **Secret Manager’s ARN** you created on secret manager
 - Access your secret manager and copy the ARN

Documented by **N.T Samuel**

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

- Click on “Add ARN” and paste the copied ARN
- Create.



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "arn:aws:secretsmanager:us-east-1:590183988696:secret:takoudb-secret-g1gNLT"
    }
  ]
}
```

NB: The Json is for the policy created above.

- Create a role now
- Role will be used by “elastic container service” ECS
- Still on IAM dashboard, click on **create role**.
- Trusted entity is “AWS service”
- Use case “Elastic container service” and choose “Elastic container service task”

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

Select trusted entity Info

Trusted entity type

- ☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ☐ **Web identity**
Allow users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Elastic Container Service

Choose a use case for the specified service.

Use case

- ☐ **Elastic Container Service**
Allows ECS to create and manage AWS resources on your behalf.
- ☐ **Elastic Container Service Autoscale**
Allows Auto Scaling to access and update ECS services.
- ☒ **Elastic Container Service Task**
Allows ECS tasks to call AWS services on your behalf.
- ☐ **EC2 Role for Elastic Container Service**
Allows EC2 instances in an ECS cluster to access ECS.
- ☐ **Elastic Container Service for VPC Lattice**
Allows access to create and manage AWS service resources required to manage VPC Lattice feature in ECS workloads.

- Search the following permissions: ***AmazonECSTaskExecutionRole*** and the **custom policy created above**

Step 6:

- Create ECS cluster and task definition, then services
- Access your ECS dashboard.
 - **Create a cluster**
 - Name it
 - Choose Fargate
 - Create
 - **Now create a task definition**
 - Name it
 - Choose Fargate as launch type
 - Select created ECS role for both **task role** and **execution role**
 - Name container and input ECR URL as image URL

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

- Set environment variables as seen below customizing the values

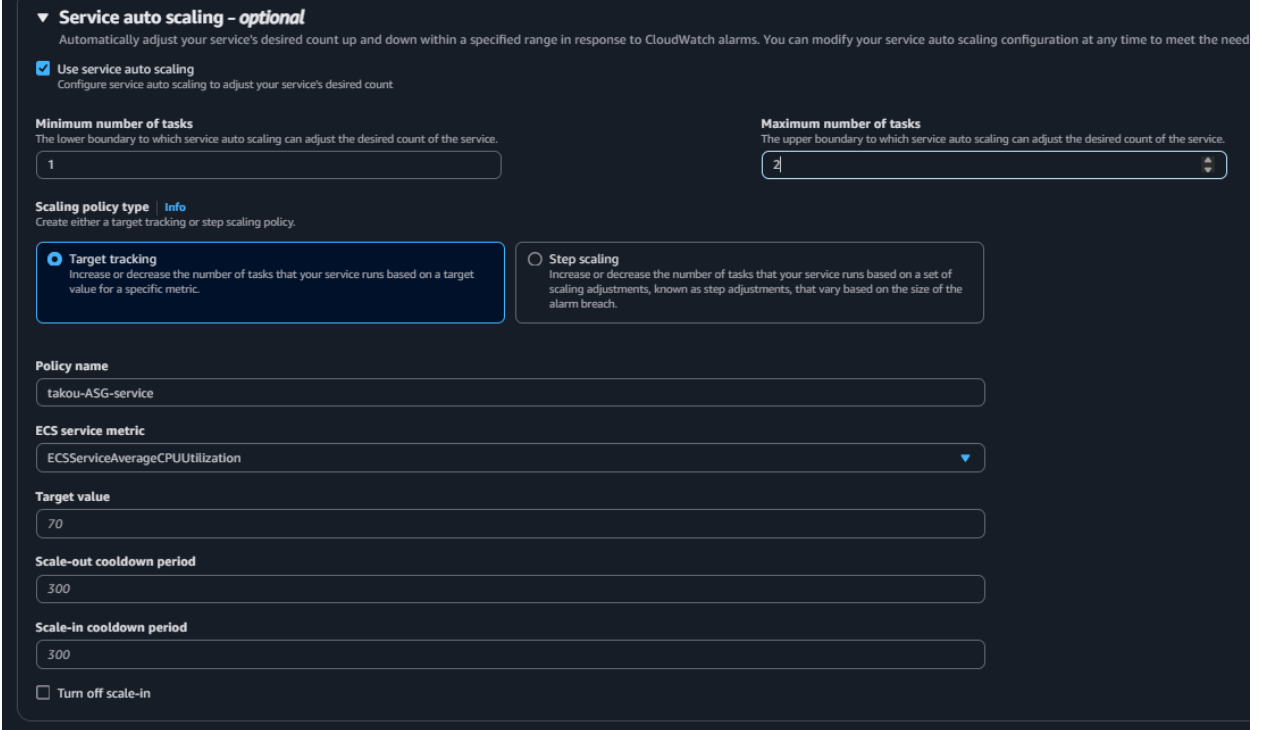
Key	Value type	Value	
DB_SERVER	Value	takoudb.cle80ms2ye5w.us-east-1.r	Remove
DB_DATABASE	Value	takoudb	Remove
SECRET_NAME	Value	takoudb-secret	Remove
AWS_REGION	Value	us-east-1	Remove

[Add environment variable](#)

- **Create a service**
- ***NB: From cluster creation, CloudFormation start working again at the background. Any failure, verify CloudFormation***
 - Under your cluster
 - Create a service
 - Launch type: Fargate
 - Application type: Service (Combination of various tasks)
 - Select task definition
 - Name your service
 - Screw down to “**networking**”
 - Select your **VPC** and **app subnets**
 - Choose **app subnet SG**
 - Access “**Load balancing**”
 - Select the load balancer and its component
 - ***NB: In the case were your ALB had SSL certificate, select the HTTPS listener.***

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV

- Enable auto scaling as seen below

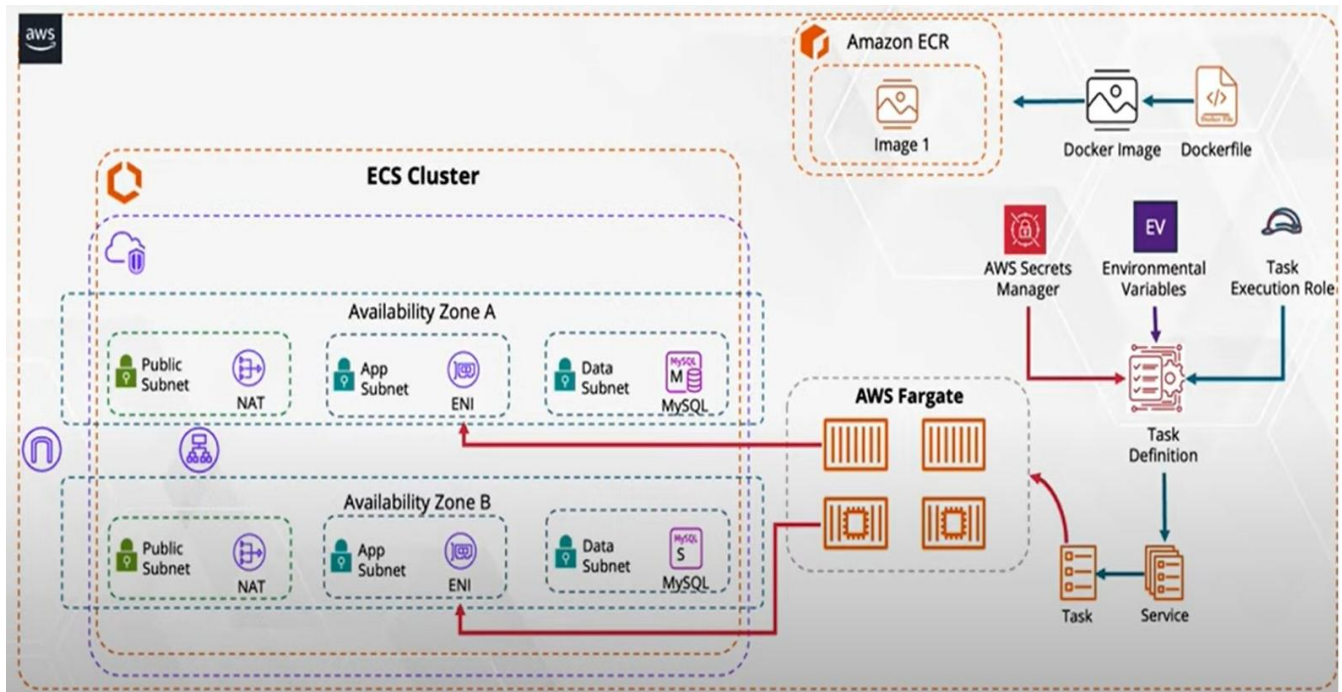
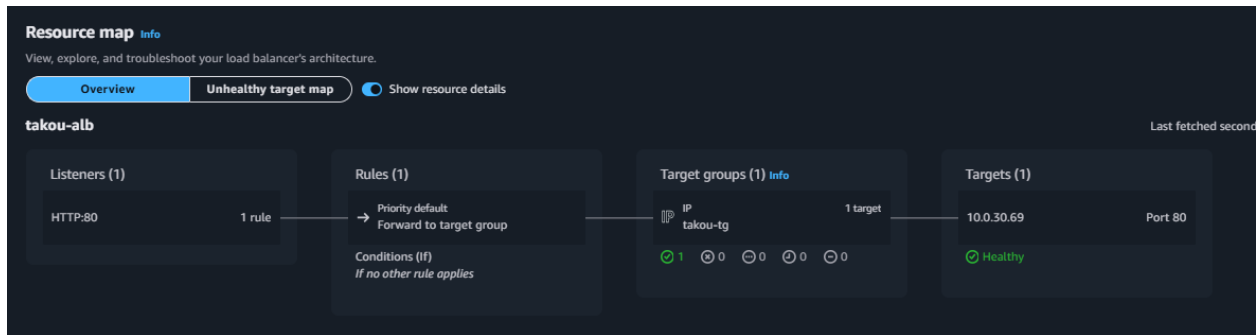
- 

Create and wait for CloudFormation to deploy your service.

- **Access Route 53** and create an alias record pointing to your ALB (Application Load Balancer).
- **View the result** in a browser using your custom DNS.
- **Test your RDS** by submitting data through your dynamic website.
- **Troubleshooting:**
 - If you encounter errors, test your site using the ALB DNS.
 - Verify ALB listeners and their attachments.
 - Ensure the correct environment database variables are used.
 - Check CloudFormation for any errors.

● **Note:** If this setup is not for production, delete CloudFormation and all related resources to avoid unnecessary costs.

https://youtu.be/C6v1GVHfOow?si=Kt_YvuKsCRvpkFmV



Documented by **N.T Samuel**