To change RDS Password:

Go to rds console, click on “thedb” cluster, click on update and change master password,

Go to ecs, task definition, click on backend task definition, update the password from there.

Allow specific IP address for local RDS access,

Go to security groups, click on edit inbound, add postgress port and your public ip as source.

CDK Stacks:

There are two primary stacks that are included in the application.

1. Network Stack
2. ComputeStack

Also, for isolation of dev and production, we have dev and prod environment. The configurations for respective environments can be found under config folder in the cdk repository.

To install cdk in your local system;

Install nodejs 16-18:

```
sudo apt install curl
curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash
nvm install 16
```

Install Cdk

```
npm install -g aws-cdk
```

Initialize CDK:

Pip install -e .

Pip install -r requirements.txt

Install CDK Application:

```
make synth
make deploy (Deploys NetworkStack for dev environment)
make deploy STACK=ComputeStack (Deploys ComputeStack for dev environment)
make deploy STAGE=prod (Deploys Network for prod environment)
make deploy STACK=ComputeStack STAGE=prod (Deploys ComputeStack for dev
environment)
```

Add new Service:

Go to Ecs config:>

- Dev
    - Add Ecs configuration under compute > ecs > new-service
- Prod
    - Add Ecs configuration under compute > ecs > new-service

Go to Ecs > compute_stack folder > ecs.py > add new function and call it in init (copy existing services configuration and update it as required).

RDS Configuration:

Please make sure to add the private Ip of the rds instance instead of the public endpoint (because we want our backend to privately connect with vpc via internal network).

Frontend/Backend Communication:

The frontend/backend communication is established via public endpoints. The backend endpoints in future should be secure via jwt based authentication mechanism for each session (user).

Extending ECS Services:

- ➔ Add new service configuration in dev/prod configuration (dev.yaml or prod.yaml)
- ➔ duplication existing service implementation from either backend or frontend service.
- ➔ Create any task role if the new service requires access to other aws services (dynamodb, s3 etc) Add it in task definition.
- ➔ Add any environment variables
- ➔ Open up ports on both container and service level:
    - add port mapping for contaiener:

```
        backend_container.add_port_mappings(ecs.PortMapping(container_port=80
00))
```

- add allow rules for service in the security group:

```
        self._backend_service.connections.allow_from(

            self._client_webapp_service, ec2.Port.tcp(8080)
        )
```

➔ Create service.

```
        # Create standard Fargate service for backendserver
        self._backend_service = ecs.FargateService(
            self,
            "backend-service",
            cluster=self._cluster,
            desired_count=self._config["compute"]["ecs"]["backend"][
                "minimum_containers"
            ],
            service_name="backendserver-" + self._config["stage"],
            task_definition=backendserver_taskdef,
            assign_public_ip=True,
            capacity_provider_strategies=capacity,
            cloud_map_options={
                "name": "backendserver-" + self._config["stage"],
                "cloud_map_namespace": self.namespace,
            },
        )
```

Note the cloud_map_options parameter. This would create a service discovery private endpoint for the service that would allow the communication between two services internally (within the vpc) . Here is how you can pass the private endpoint to another service's environment variable in the container definition

```
 environment={
                "REACT_APP_BACKEND_URL":
f"http://backendserver.{self.namespace.namespace_name}:8000",
             },
```