Fabio Ibanez (fabioi)
Sid Potti (sidpotti)
CS40
Mar 17, 2024

Overview

For our project we decided to deploy Redmine, which is an open-source project management system. Redmine is a flexible project management system that users, like ourselves, can adapt to their platform and database needs. In our case we're using a PostgreSQL DB. We deployed this project using the following code: https://github.com/sameersbn/docker-redmine/tree/master. Our deployment can be found at: redmine.sidpotti.infracourse.cloud/.

Architecture

Services

Compute Resources

Our backend is deployed using Docker on AWS ECS, and we use Fargate to elastically scale to meet compute requirements. We also use an **application load balancer** (ALB) to effectively distribute load and requests, particularly during times of high traffic.

Data Resources

Redmine is compatible with both MySQL and PostgreSQL databases. We decided to use PostgreSQL over MySQL because it has full ACID compliance, which is something we were looking for, and multiversion concurrency control (MVCC)[1] since in production it's likely that multiple users will be using the project management software at the same time. So we thought that for our use case, on an architecture level, PostgreSQL made more sense. Not to mention, our application is very write-heavy so PostgreSQL is also a better choice on that front.

Network Resources

We provisioned an AWS VPC deployed in `‘us-west-2’` and `‘us-west-2a’`. The application load balancer is in the public subnet, whereas our database is in a private isolated subnet with our ECS clusters being in a private subnet with egress. This is generally good practice in order to limit where communications to the database and ECS can come from. We also used AWS Secrets Manager to securely encrypt the authentication information for our database to be used by the Fargate cluster. We used AWS Route 53 to handle DNS records and also took further security into consideration in our network resources. We used AWS Certificate Manager to

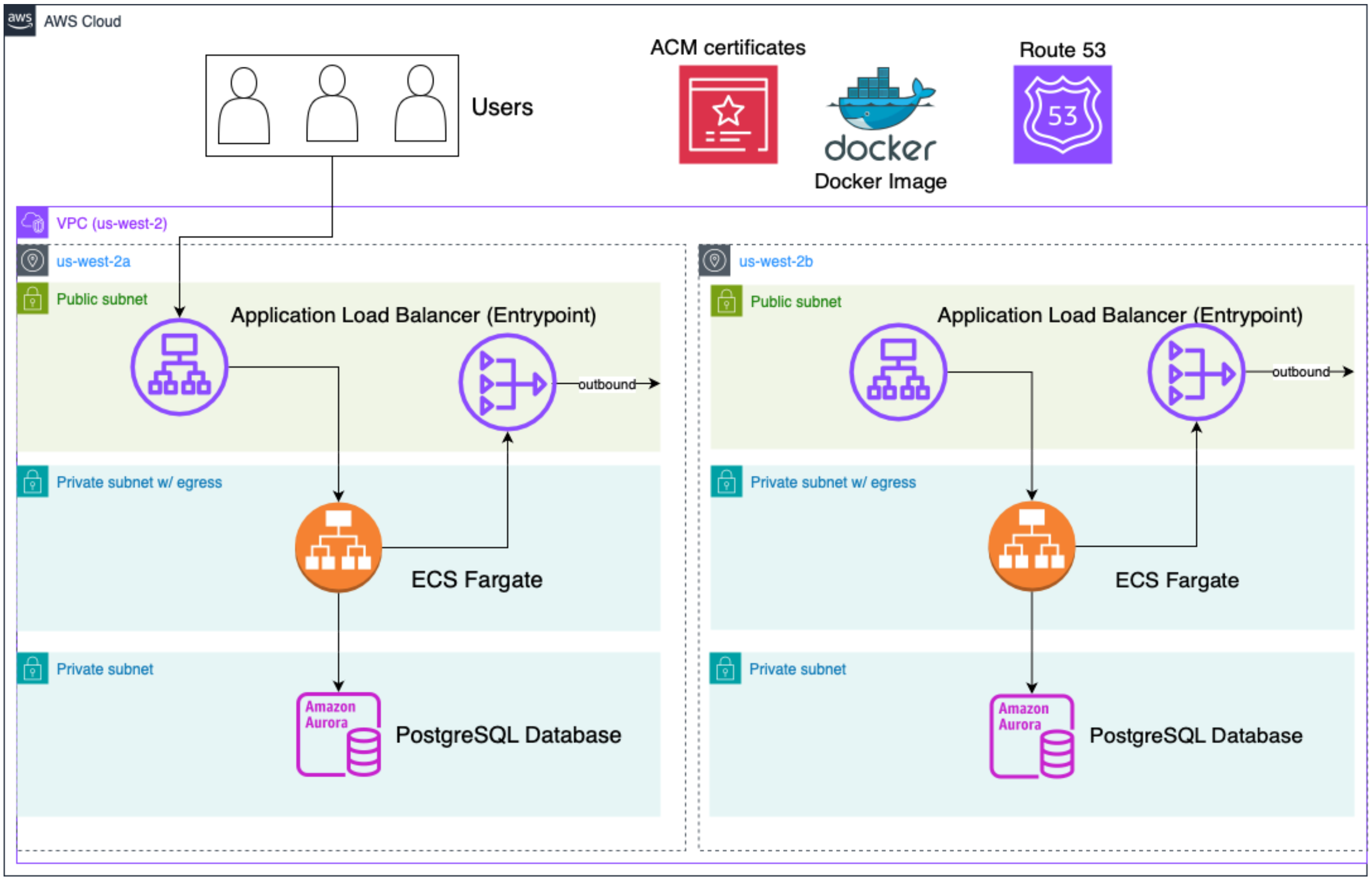
[1] https://aws.amazon.com/compare/the-difference-between-mysql-vs-postgresql/

generate a **backend TLS certificate** to encrypt and secure communications. We didn't need a frontend TLS certificate since the HTML was generated just-in-time (JIT). So there was no "proper" frontend code to create a TLS certificate for.

Observability

We don't have any observability for our project.

Diagram



Operation

The users of our Redmine instance directly communicate with the application load balancer, which is the entrypoint to our application. This was enforced by the nature of our application, which is written in Ruby on Rails which performs JIT HTML generation which prohibits us from making use of Cloudfront CDN and similar services because there's nothing that we can usefully cache for our users.

When a client makes a request to our ALB, the ALB communicates without an ECS container in the private subnet which then makes any necessary changes (adding task, deleting task, adding project, deleting project, etc.) state changes to our PostgreSQL database.

Process

Difficulties encountered

When deploying our project we ran into a few difficulties. One of them was making sure that we opened up and assigned the right permissions to certain ports required by the database and the Redmine Docker image we used. We ran into a tricky situation with circular dependencies in creating a security group for the database in the `data_stack` the reason that this was tricky was because it created a circular dependency between the data_stack and the `compute_stack` since the security group required a fargate service security group. We tried to create a separate stack for creating the security group that also resulted in circular dependencies so we removed it. We ended up reasoning about this problem, and checking in with Cody. It turns out since our database is in the private subnet we can just allow connections from all ipv4 addresses since you can't access the database from the internet.

In general though, we had some initial misunderstandings with how deploying a docker image would even work. We didn't understand the interface between the database and the application and this took some time for us to understand so that we could then translate requirements into IaC code.

Lessons learned

Throughout this project we got a deep understanding of how cloud deployments work, and the various AWS services required (and those not necessary) to deploy a Redmine instance. Overall we feel very comfortable being able to deploy infrastructure on the cloud – and we understand now how to break down Docker image requirements into IaC code. One of the biggest learnings in this project is understanding how each stack builds on top of previously deployed stacks. In debugging this project we appreciated the separation of concerns that having stacks provided. This made debugging and identifying problems a ***<u>much</u>*** more pleasant experience than what it could have been had we not had different stacks.

Citations:

1. Redmine Docker image we used:
   https://github.com/sameersbn/docker-redmine/tree/master
2. Yoctogram Assignment 2 code
   a. We used the assignment 2 code as a boilerplate starting point for our implementation. We asked the teaching staff about this and were told this was ok.
3. ChatGPT-4
   a. Used GPT-4 for debugging syntax of IaC APIs.

Tree Structure of Project:

```
.
├── README.md
├── app.py
├── cdk
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-310.pyc
│   │   ├── compute_stack.cpython-310.pyc
│   │   ├── data_stack.cpython-310.pyc
│   │   ├── dns_stack.cpython-310.pyc
│   │   ├── network_stack.cpython-310.pyc
│   │   ├── security_stack.cpython-310.pyc
│   │   └── util.cpython-310.pyc
│   ├── compute_stack.py
│   ├── data_stack.py
│   ├── dns_stack.py
│   ├── network_stack.py
│   └── util.py
├── cdk.context.json
├── cdk.json
├── cdk.out
│   ├── asset.7f18a11296f35510ee16538afec983ed6312e12afbf81b777089a9f8e34e2474
│   │   ├── __entrypoint__.js
│   │   └── index.js
│   ├── asset.eefabeeb9435f22791074dea9e4e942aaa601c1f1fb1574d9bf54e4fe5fab075
│   │   ├── Dockerfile
│   │   ├── VERSION
│   │   ├── assets
│   │   │   ├── build
│   │   │   │   └── install.sh
│   │   │   ├── runtime
│   │   │   │   ├── config
│   │   │   │   │   ├── nginx
│   │   │   │   │   │   ├── redmine
│   │   │   │   │   │   └── redmine-ssl
│   │   │   │   │   └── redmine
│   │   │   │   │       ├── additional_environment.rb
│   │   │   │   │       ├── config.ru
│   │   │   │   │       ├── configuration.yml
│   │   │   │   │       ├── database.yml
│   │   │   │   │       ├── secret_token.rb
│   │   │   │   │       └── unicorn.rb
│   │   │   │   ├── env-defaults
│   │   │   │   └── functions
│   │   │   └── tools
│   │   │       ├── redmine-backup-create
│   │   │       ├── redmine-install-plugins
│   │   │       └── redmine-install-themes
│   │   ├── docker-compose-mariadb.yml
│   │   ├── docker-compose-memcached.yml
│   │   ├── docker-compose-mysql.yml
│   │   ├── docker-compose-sqlite3.yml
│   │   ├── docker-compose-ssl.yml
│   │   ├── entrypoint.sh
│   │   ├── make_release.sh
│   │   ├── support
│   │   │   └── rhel
│   │   │       └── etc
│   │   │           ├── rc.d
│   │   │           │   └── init.d
│   │   │           │       └── redmine
│   │   │           └── sysconfig
│   │   │               └── redmine
│   │   └── test
│   │       ├── Dockerfile
│   │       ├── index.js
│   │       └── package.json
│   ├── cdk.out
│   ├── final-project-redmine-compute-stack.assets.json
│   ├── final-project-redmine-compute-stack.template.json
│   ├── final-project-redmine-data-stack.assets.json
│   ├── final-project-redmine-data-stack.template.json
│   ├── final-project-redmine-dns-stack.assets.json
│   ├── final-project-redmine-dns-stack.template.json
│   ├── final-project-redmine-network-stack.assets.json
│   ├── final-project-redmine-network-stack.template.json
│   ├── final-project-redmine-security-stack.assets.json
│   ├── final-project-redmine-security-stack.template.json
│   ├── manifest.json
│   └── tree.json
├── docker-redmine
│   ├── Changelog.md
│   ├── Dockerfile
│   ├── LICENSE
│   ├── Makefile
│   ├── README.md
│   ├── VERSION
│   ├── assets
│   │   ├── build
│   │   │   └── install.sh
│   │   ├── runtime
│   │   │   ├── config
│   │   │   │   ├── nginx
│   │   │   │   │   ├── redmine
│   │   │   │   │   └── redmine-ssl
│   │   │   │   └── redmine
│   │   │   │       ├── additional_environment.rb
│   │   │   │       ├── config.ru
│   │   │   │       ├── configuration.yml
│   │   │   │       ├── database.yml
│   │   │   │       ├── secret_token.rb
│   │   │   │       └── unicorn.rb
│   │   │   ├── env-defaults
│   │   │   └── functions
│   │   └── tools
│   │       ├── redmine-backup-create
│   │       ├── redmine-install-plugins
│   │       └── redmine-install-themes
│   ├── docker-compose-mariadb.yml
│   ├── docker-compose-memcached.yml
│   ├── docker-compose-mysql.yml
│   ├── docker-compose-sqlite3.yml
│   ├── docker-compose-ssl.yml
│   ├── docker-compose.yml
│   ├── entrypoint.sh
│   ├── make_release.sh
│   ├── support
│   │   └── rhel
│   │       └── etc
│   │           ├── rc.d
│   │           │   └── init.d
│   │           │       └── redmine
│   │           └── sysconfig
│   │               └── redmine
│   └── test
│       ├── Dockerfile
│       ├── index.js
│       └── package.json
├── requirements-dev.txt
├── requirements.txt
├── source.bat
├── tests
│   ├── __init__.py
│   └── unit
│       └── __init__.py
└── tree.txt

36 directories, 99 files
```