3.A method for correctly and securely provisioning the MONGODB\_URI and REDIS\_URL environment variables. This could potentially use Hashicorp Vault.

* We can either use Hashicorp vault or AWS secrets Manager for securing the environment variables. AWS secrets manager is a better choice for the solution because:
  + fully managed by AWS, hence zero administration from user side
  + Secured storage of secrets on AWS
  + Allow encryption of keys stored via KMS
  + Key rotation can be configured within a specific period
  + Privilege Access Management (IAM)

4.A writeup describing your solution.

The solution has two parts

1. Dockerize the ruby application

While creating the docker image we will have to do the following steps in Dockerfile.

1. Fetch Ruby image as base for ruby version used in application.
2. Install all the packages needed to run the rails application in container like binutils, curl, git, gnupg, cmake, python, python-dev, postgresql-client, supervisor, tar, tzdata etc.
3. Create an application directory in the container.
4. Copy application code into application directory in docker container.
5. Set work directory to application directory in docker container.
6. Run bundler install.
7. Run rake assets precompile bundle command.
8. Expose port 3000 of docker container.
9. Start rails server bind the server to 0.0.0.0

2. Use terraform to provision the AWS infrastructure required to deploy the application

* We use terraform templates to create the required resources for the application
  + Networking: VPCs and Subnets
  + Database: Redis
  + IAM roles for the relevant access
  + AWS secrets manager to host the secrets/credentials
  + Elastic container service to run the docker image of the ruby application