Take Home Challenge

Objective

This is a take home challenge to test your knowledge of Cloud Engineering. The challenge is divided into two parts, each with a different objective but relying on each other. Plan and design the infrastructure for real world deployment and roll out.

Part 1

Create an automated pipeline that consumes customer records via a CSV file and uploads them into a dynamodb database. The customer will be uploading these files on a daily basis. The pipeline should be able to handle multiple files in one go. Make sure to handle multitennancy as well as error handling. People make mistakes and can upload the same file multiple times, so make sure to handle this as well.

CSV File Fields

These are the minimum fields that are required for the CSV file. Feel free to add more fields if needed.

- medical_record_number
- first_name
- last_name
- date_time (in local time)
- doctors_notes

On the day of the demo, we will provide you with a few CSV files to try to upload. Then we will ask you to query certain data from the database.

Part 2

You are to take 2 microservices frontend microservice which connects to the middle-tier API microservice which will interact with the database that you created in Part 1.

Deploy thes microservices on AWS/LocalStack using the services that you are most comfortable with.(ECS, EKS, Fargate, etc). Keep in mind of cost to serve the microservices and the database. You may have to modify the microservice code to make it work for deployment.

You should also create a ci/cd pipeline using the services of your choice (GitHub Actions, CircleCI, Jenkins, etc). Create additional pipelines/promotion techniques to move from dev, stage and prod environments.

Requirements

- Create a design doc of your infrastructure for both parts of the challenge. Pictures speak volumes. Walk us through your thought process.
- Use Infrastructure as Code (IaC). Pick either Terraform or CDK as your IaC.
- Write unit tests to validate that your code works as expected.
- Do your best to implement as much security as possible.

- Please try to check all your work into some source control system as you go. GitHub, Bitbucket, CodeCommit, etc. We don't want you to lose all your hard work.
- Make sure at the end you have way to destroy the infrastructure so as not to incur any costs.

Questions

Feel free to email me

- joe@tendo.com

Conclusion

Lastly have fun with this and be as creative as possible with the challenge. We are not looking for a perfect solution but rather an example of your thought process and how you would approach this problem in real life.

Environment

This will require AWS access:

- **Recommended:** If you already have an AWS account, feel free to use that or sign up for a free account.
- **Alternative:** Sign up for a 14-day trial account of LocalStack. Make sure you schedule your demo prior to trial expiration 😀 .
  - LocalStack does **NOT** support `CodeBuild` and `CodePipeline` and so you will have to add additional layers if you chose this path.

Quick Guide for LocalStack

LocalStack Setup

- Install Container Runtime: Docker(Subscription Needed Now), Podman, Rancher. Pick one.
  - Docker Install Doc
  - `brew install podman`
  - Rancher Install Doc
- Install LocalStack:
  - `brew install localstack/tap/localstack-cli`
  - `export LOCALSTACK_AUTH_TOKEN=<your_license`
  - `localstack license activate`
  - `localstack start`
- Set the license
  - `export LOCALSTACK_AUTH_TOKEN=<your_token>`
  - `localstack license activate`
- Startup localstack
  - `localstack start`
- LocalStack Desktop

LocalStack Terraform Setup

- Docs
- `brew install terraform`
- `pip install terraform-local`
- `tflocal init; tflocal plan; tflocal apply`

LocalStack CDK Setup

- Docs
- AWS CLI Local
  - `brew install awscli-local`
- NVM:
  - `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash`
- Place this in your shell rc file. Eg. ~/.zshrc ~/.bashrc, etc.
  - `export NVM_DIR="$([ -z "${XDG_CONFIG_HOME-}" ] && printf %s "${HOME}/.nvm" || printf %s "${XDG_CONFIG_HOME}/nvm")" [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm`
- Re-load your shell file
  - `exec $SHELL`
- NodeJS v22.0.0
  - `nvm install v22.0.0`
- CDK App Config
  - `env=cdk.Environment(account='000000000000', region='us-east-1')`
- AWS Config
  - `aws configure --profile localstack`
  - When prompted, enter the following:
    - AWS Access Key ID: mock_access_key
    - AWS Secret Access Key: mock_secret_key
    - Default region name: us-east-1
    - Default output format: json
- Then, you can set the AWS_PROFILE environment variable to use this profile:
  - `export AWS_PROFILE=localstack`

Debugging

Some commands to help with debugging.

- `cdklocal -vvv —debug bootstrap`