

Migrate your Ruby on Rails App to AWS Fargate using AWS Rails Provisioner

Jingyi Chen @ AWS SDK for Ruby

Oct. 21st, 2019

Agenda

- What is “aws-rails-provisioner”? Current Status?
- How to use “aws-rails-provisioner”?
- How does “aws-rails-provisioner” work?
- Why use “aws-rails-provisioner”?
- Q & A



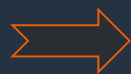
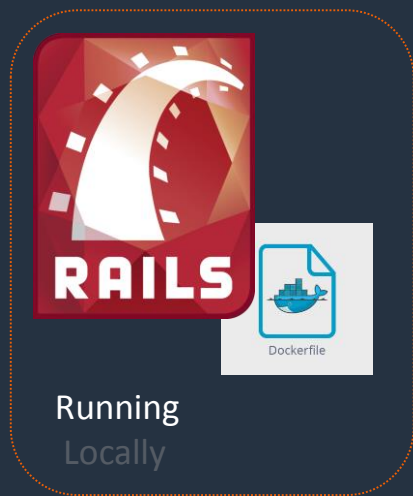
'aws-rails-provisioner'

under developer preview,
supporting AWS Fargate

A tool to define and deploy containerized
Ruby on Rails applications on AWS

'aws-rails-provisioner'

under developer preview,
currently supporting AWS Fargate



aws-rails-provisioner



Why AWS Fargate?

Run containers without having to manage servers or clusters

What's not covered

- Why and how to containerize a Rails Application
- Different compute platform comparison (EC2, EKS, ECS, Elastic Beanstalk ...)
- Deep dive on each AWS Services used
- Deep dive on AWS Cloud Development Kit (CDK)



One Configuration File
Two Commands
Three AWS CDK Stacks

One configuration file

aws-rails-provisioner.yml

```
vim examples/
1 multi_service.yml
1 | Version: '0'
2
3 vpc:
4   max_azs: 2
5   enable_dns: true
6 services:
7   rails_foo:
8     source_path: /path/to/rails_foo
9     fargate:
10      desired_count: 1
11      public: true
12      envs:
13        PORT: 80
14      db_cluster:
15        engine: aurora-mysql
16        db_name: app_development
17      scaling:
18        max_capacity: 2
19        on_cpu:
20          target_util_percent: 80
21          scale_in_cool_down: 300
22      rails_no_db:
23        source_path: /path/to/no_db
24        fargate:
25          desired_count: 1
26          public: true
27          envs:
28            PORT: 80
~
~
~
~
NORMAL master multi_service.yml 3% 1:1
:1
```


Two Commands

aws-rails-provisioner build
aws-rails-provisioner deploy

Two Commands

aws-rails-provisioner -h

```
chejngy@4c327596e395: ~  
chejngy@4c327596e395 ~$ aws-rails-provisioner -h  
Usage: aws-rails-provisioner [command] [options]  
  
Commands:  
  build: generate CDK stacks from aws-rails-provisioner.yml  
  deploy: deploy generated CDK stacks  
  
Options:  
  -s, --service SERVICE_NAME      Service name. Changes apply to all services configured under :services if not provided  
  -p, --profile AWS_PROFILE_NAME  Used the indicated AWS profile  
  -f, --file YAML_FILE_PATH       File path to `aws-rails-provisioner.yml`, default to `aws-rails-provisioner.yml` at current directory  
  -d, --cdk-directory DIR_PATH    Directory where generated code lives, defaults to `cdk-sample`  
  --with-cicd                      Enable CICD stack generation, use with `build`  
  --init                           Deploy the stack that creates VPC and ECS cluster  
  --fargate                         Deploy the stack that creates Fargate service and DBCluster (if specified)  
  --cicd                           Deploy the stack that builds CICD flow  
  -h, --help                       Show help  
chejngy@4c327596e395 ~$
```

Three AWS CDK Stacks

Initial Stack & Fargate Stack

CI/CD

```
aws-rails-provisioner build --with-cicd  
aws-rails-provisioner deploy --cicd
```

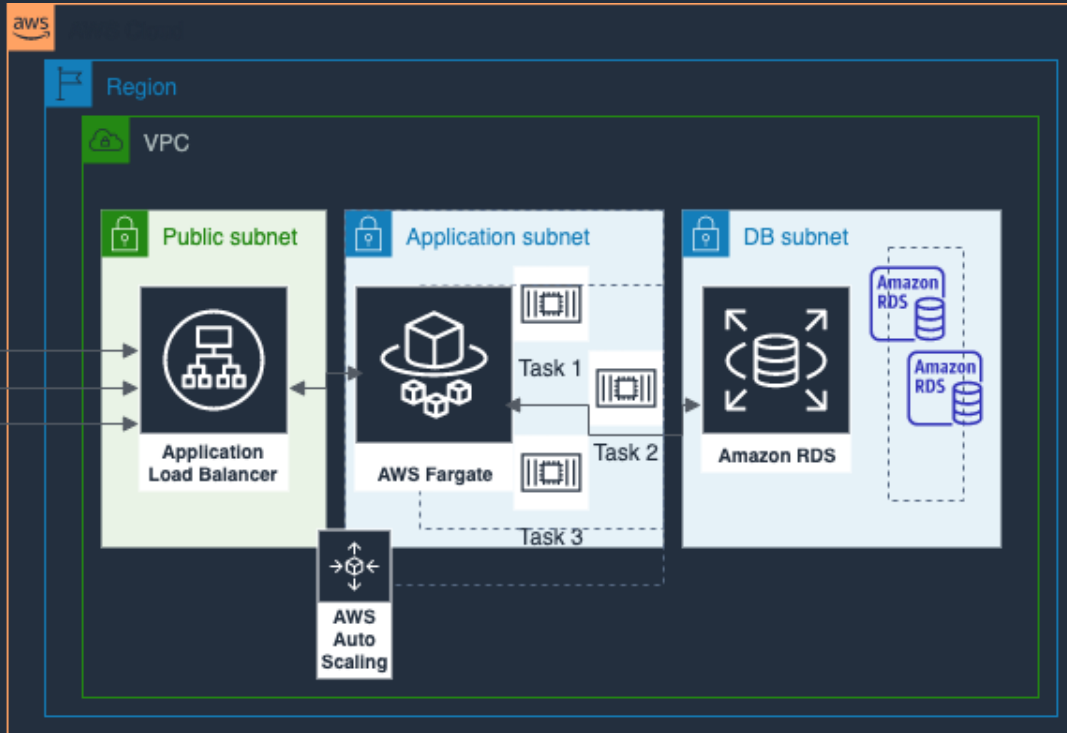


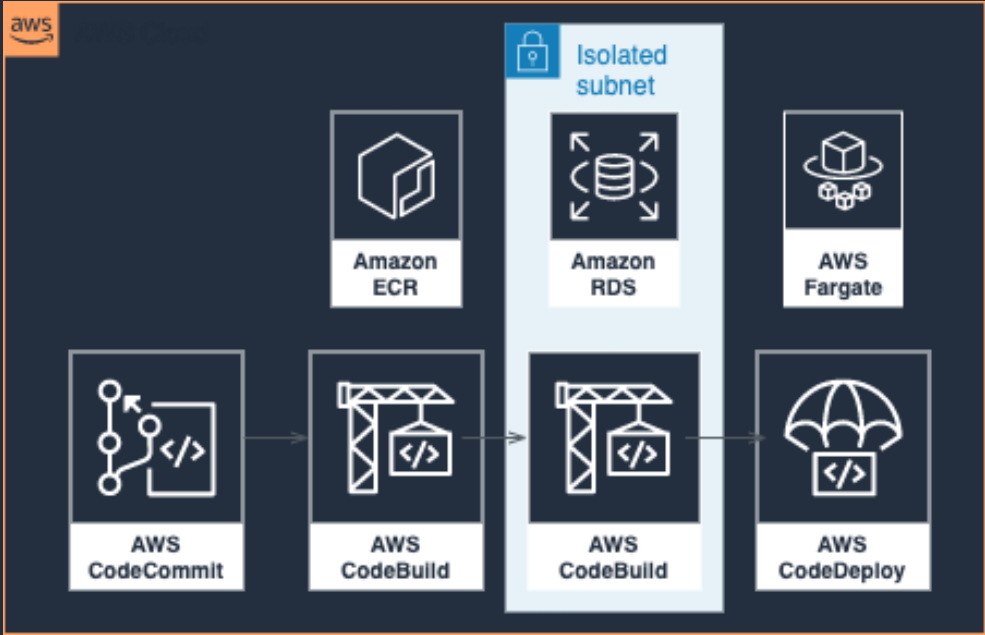
Three AWS CDK Stacks

CI/CD Stack



What have been built so far?





Infrastructure as Code on AWS

AWS Blog Post:

<https://aws.amazon.com/blogs/developer/introducing-the-aws-rails-provisioner-gem-developer-preview/>

Github Repo:

<https://github.com/awslabs/aws-rails-provisioner>

Thank you!

Q & A

<https://github.com/awslabs/aws-rails-provisioner>

<https://rubygems.org/gems/aws-rails-provisioner>