



The 1 API on New AWS Deployment Stack #1

Ankit - Long - Nam

Growth Session #31 - October 16 2020

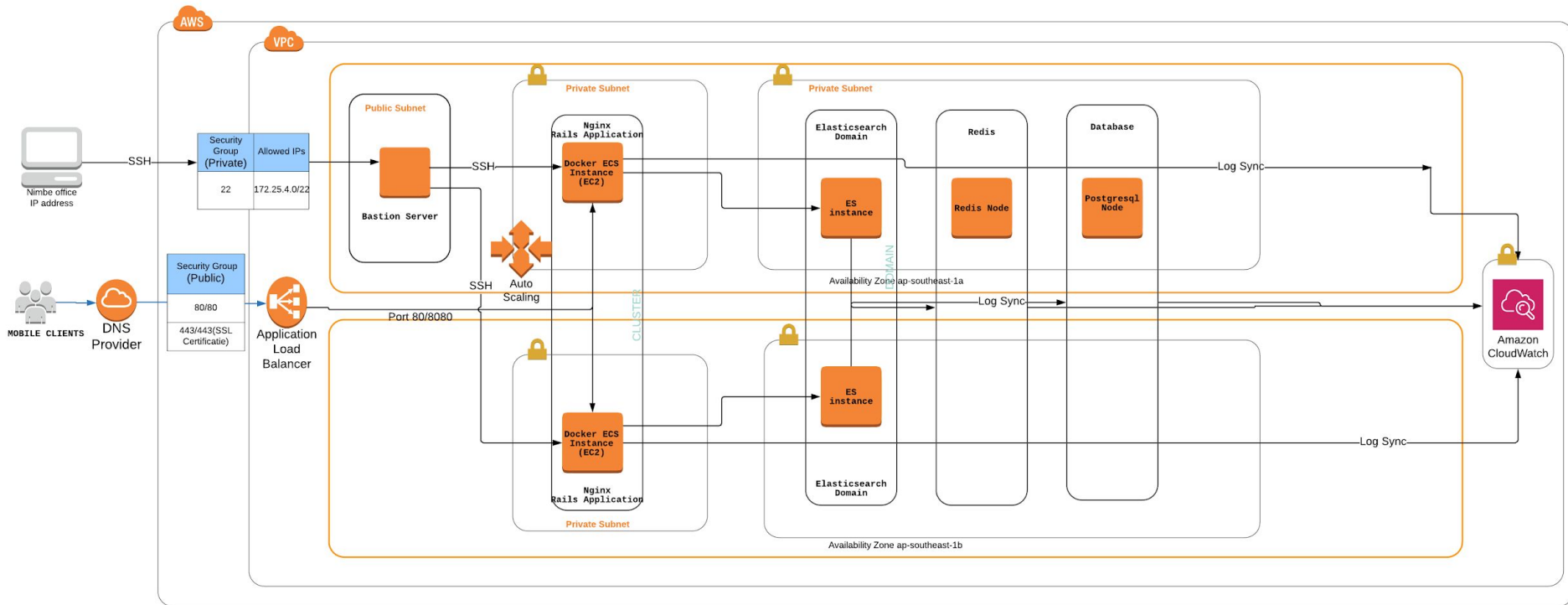
Objectives

- To deploy the1-api in serverless environment using AWS Fargate.
- Use nimble AWS account instead of The 1 AWS account.
- Explore CI/CD stack of AWS (CodeBuild, CodePipeline, Codedeploy)
- Use terraform for managing the infrastructure.

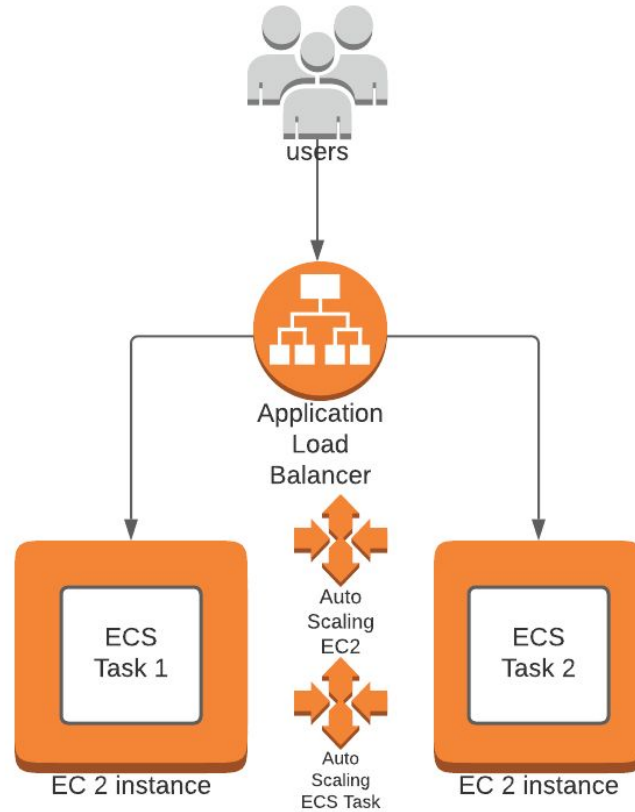
Current Infrastructure Diagram

THE1-API-INFRASTRUCTURE

Team Nimble | June 24, 2020

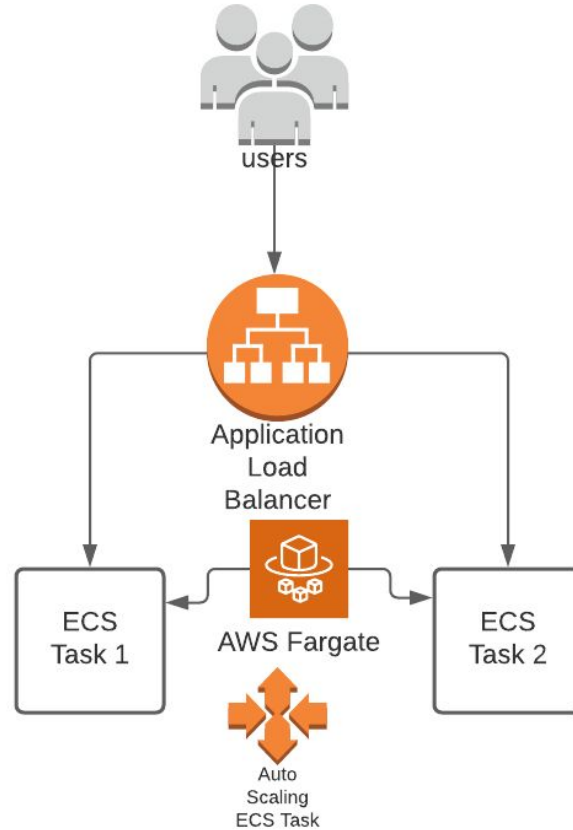


Infrastructure Diagram: ECS and EC2



- Autoscaling is based on metrics, such as
 - ECSAverageCPUUtilization (Per task)
 - ASGAverageCPUUtilization (Per EC2 Instance)
- Since the metrics for Autoscaling the task and the instances are different, we encounter several issues with autoscaling.
- Often it occurs in values calculated by AWS that:
 - Desired number of instances \neq Desired number of tasks

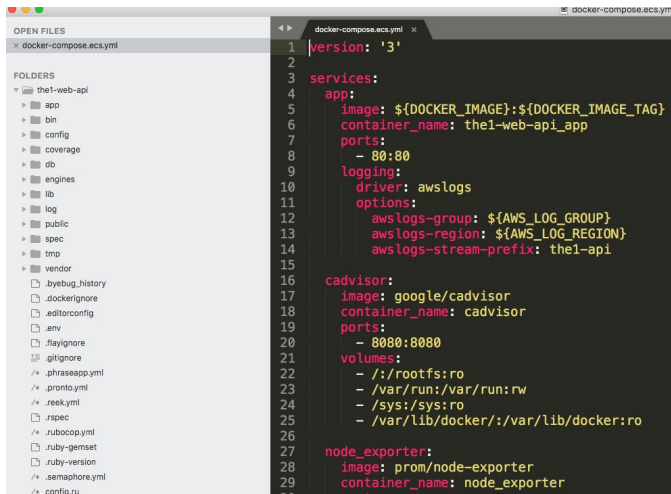
New Infra with Fargate



- Technology that you can use with Amazon ECS to run **containers** without having to manage servers.
- No longer have to provision, configure, or scale clusters of virtual machines to run containers.
- This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing.
- It is not specific to ECS, can be used with EKS (Elastic Kubernetes Service) as well

Implementation Plan

- Create a new terraform repo <https://github.com/nimblehq/the1-api-infra-demo>
- Replicate existing infra such as VPC, Elasticsearch, RDS, ElastiCache, but leave out autoscaling groups and EC2 instances
- Implement ECS cluster using Fargate
- Move ECS Task Definition from `compose` file to `terraform repo` (Currently the task definition lies inside project, and we initialize infra using ecs-cli instead of terraform)

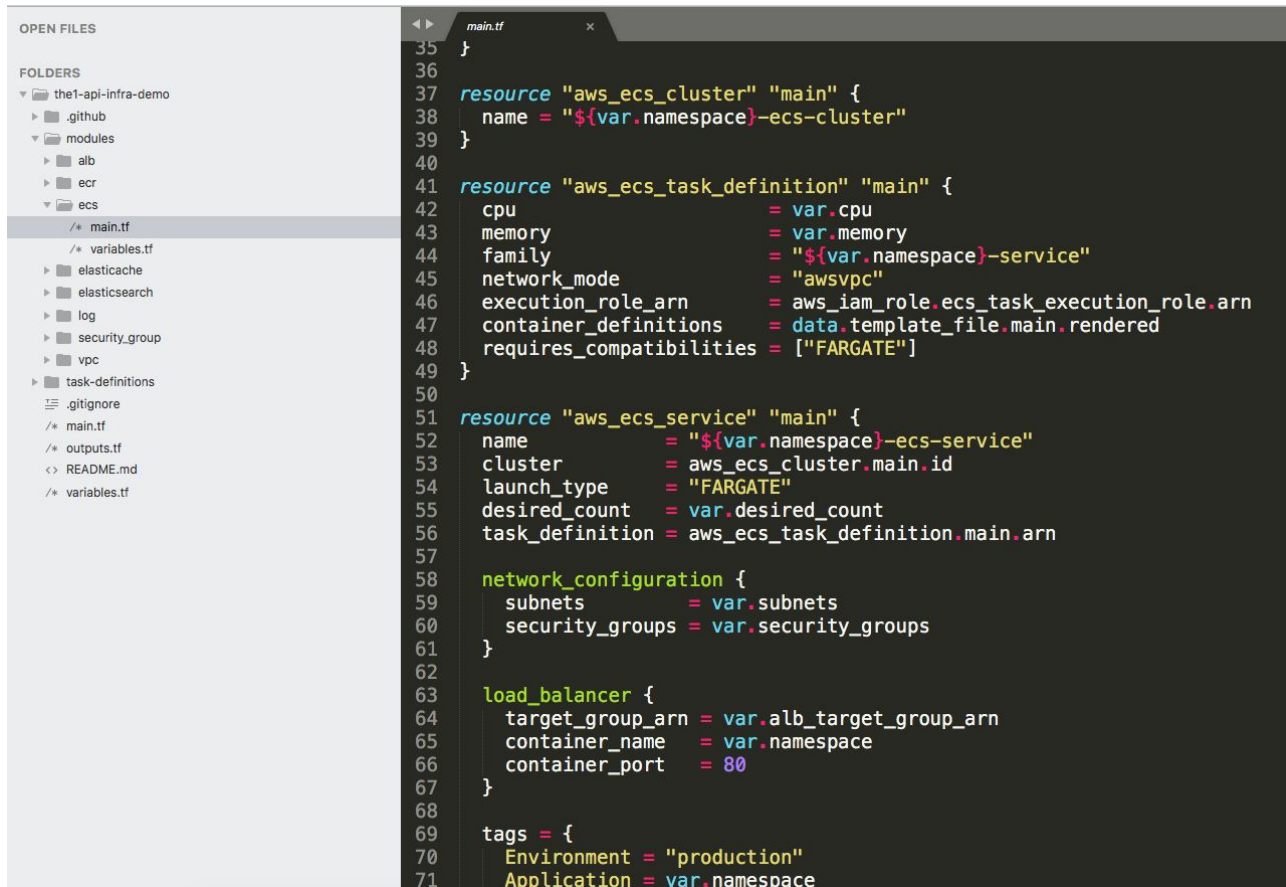


The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'the1-web-api', 'app', 'bin', 'config', 'coverage', 'db', 'engines', 'lib', 'log', 'public', 'spec', 'tmp', 'vendor', and files like '.byebug_history', '.dockerignore', '.editorconfig', '.env', '.flagignore', '.gitignore', 'phrasapp.yml', 'pronto.yml', 'reek.yml', 'rspec', 'rubocop.yml', 'ruby-gemset', 'ruby-version', 'semaphore.yml', and 'config.ru'. The code editor shows a Docker Compose file named 'docker-compose.ecs.yml' with the following content:

```
1 version: '3'
2
3 services:
4   app:
5     image: ${DOCKER_IMAGE}:${DOCKER_IMAGE_TAG}
6     container_name: the1-web-api_app
7     ports:
8       - 80:80
9     logging:
10      driver: awslogs
11      options:
12        awslogs-group: ${AWS_LOG_GROUP}
13        awslogs-region: ${AWS_LOG_REGION}
14        awslogs-stream-prefix: the1-api
15
16   cadvisor:
17     image: google/cadvisor
18     container_name: cadvisor
19     ports:
20       - 8080:8080
21     volumes:
22       - /:/rootfs:ro
23       - /var/run:/var/run:rw
24       - /sys:/sys:ro
25       - /var/lib/docker:/var/lib/docker:ro
26
27   node_exporter:
28     image: prom/node-exporter
29     container_name: node_exporter
```

→ To be moved to
Terraform repository

Terraform Code and Demo



OPEN FILES

FOLDERS

- the1-api-infra-demo
 - .github
 - modules
 - alb
 - ecr
 - ecs
 - main.tf
 - variables.tf
 - elasticache
 - elasticsearch
 - log
 - security_group
 - vpc
 - task-definitions
- .gitignore
- main.tf
- outputs.tf
- README.md
- variables.tf

```
35 }
36
37 resource "aws_ecs_cluster" "main" {
38   name = "${var.namespace}-ecs-cluster"
39 }
40
41 resource "aws_ecs_task_definition" "main" {
42   cpu           = var.cpu
43   memory        = var.memory
44   family        = "${var.namespace}-service"
45   network_mode  = "awsvpc"
46   execution_role_arn = aws_iam_role.ecs_task_execution_role.arn
47   container_definitions = data.template_file.main.rendered
48   requires_compatibilities = ["FARGATE"]
49 }
50
51 resource "aws_ecs_service" "main" {
52   name = "${var.namespace}-ecs-service"
53   cluster = aws_ecs_cluster.main.id
54   launch_type = "FARGATE"
55   desired_count = var.desired_count
56   task_definition = aws_ecs_task_definition.main.arn
57
58   network_configuration {
59     subnets = var.subnets
60     security_groups = var.security_groups
61   }
62
63   load_balancer {
64     target_group_arn = var.alb_target_group_arn
65     container_name = var.namespace
66     container_port = 80
67   }
68
69   tags = {
70     Environment = "production"
71     Application = var.namespace
72   }
73 }
```

- Replicated the existing infra of The 1 on Nimble account using terraform
- Setup the ECS cluster using fargate

Explore CI/CD stack of AWS:

- Codebuild
- Codepipeline
- Codedeploy

Thanks!

Contact Nimble

nimblehq.co

hello@nimblehq.co

Bangkok

399 Interchange 21 Sukhumvit Road, Unit
#2402-03, Klong Toei, Wattana, Bangkok
10110, Thailand

Singapore

28C Stanley St, Singapore 068737

Hong Kong

20th Floor, Central Tower
28 Queen's Road, Central, Hong Kong

