The 1 API on New AWS Deployment Stack #1

Ankit - Long - Nam

Growth Session #31 - October 16 2020

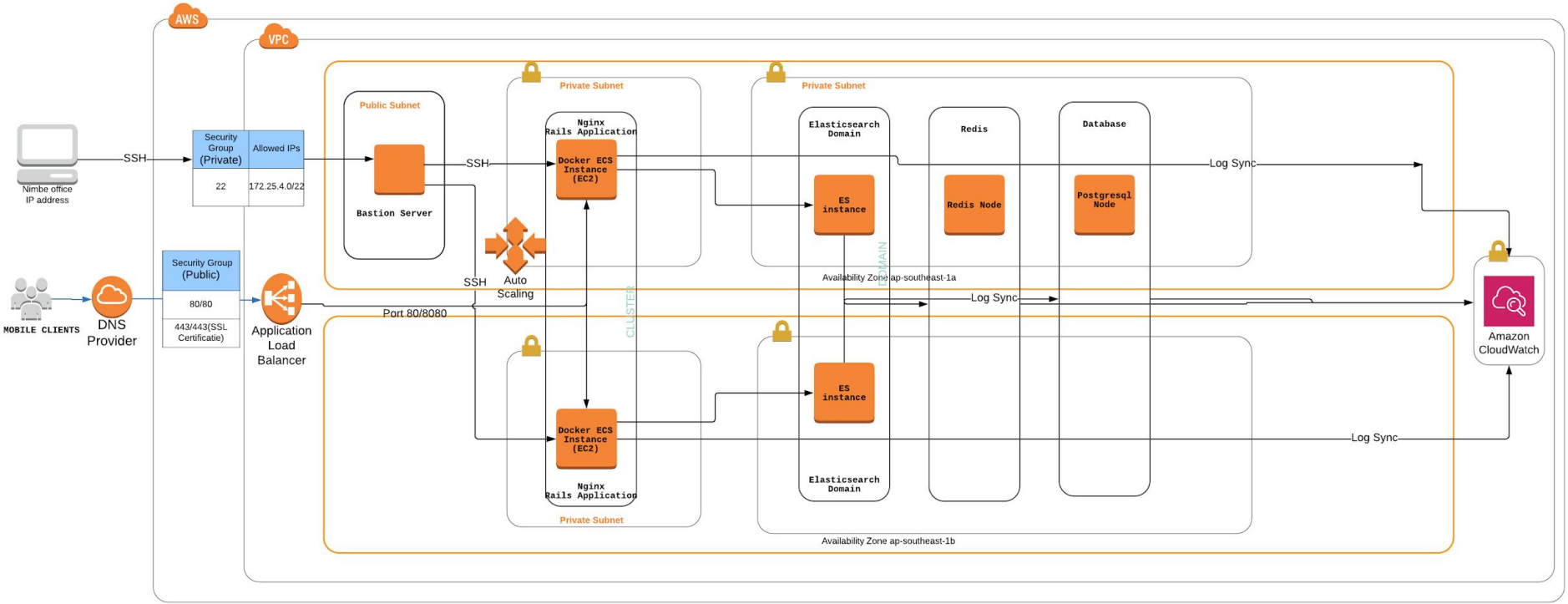
Objectives

- 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.

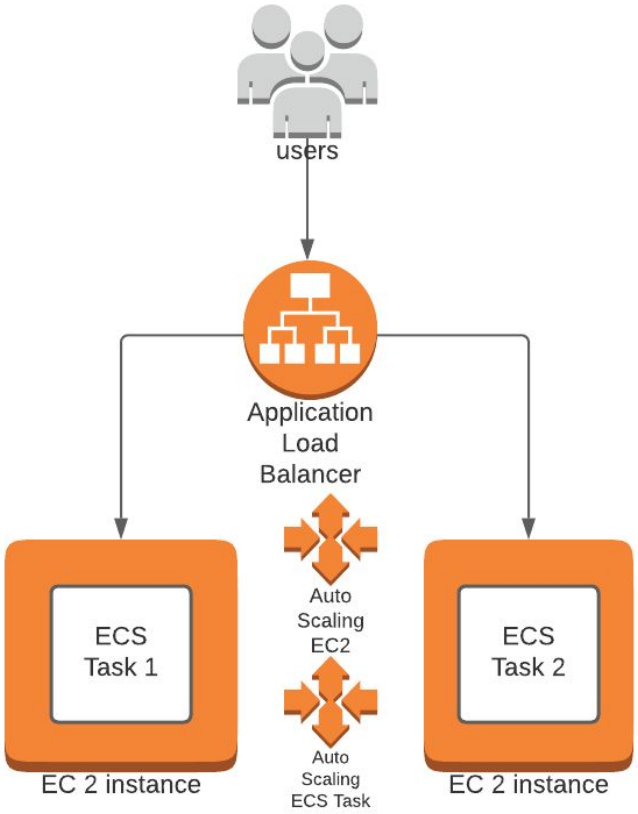
Current Infrastructure Diagram

THE1-API-INFRASTRUCTURE

Team Nimble | June 24, 2020



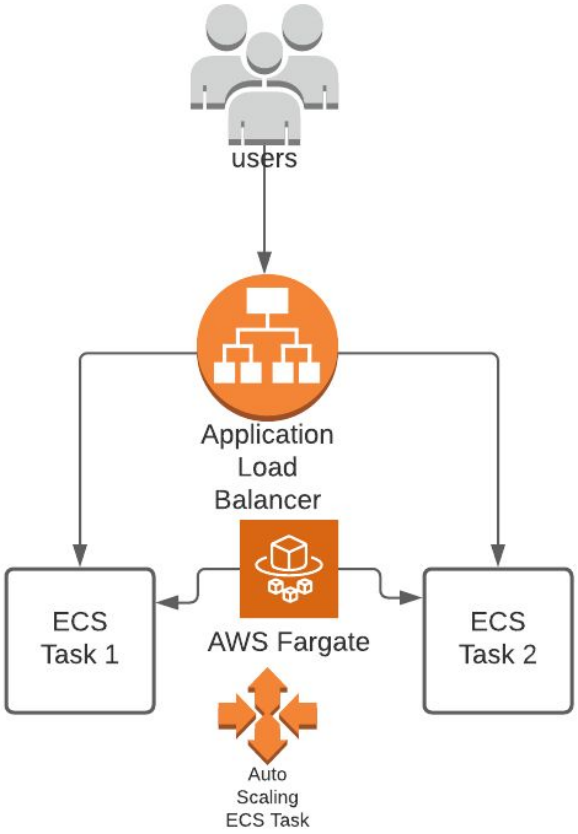
Infrastructure Diagram: ECS and EC2



Issues with Autoscaling

- 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 ≠ Desired number of tasks

New Infra with Fargate



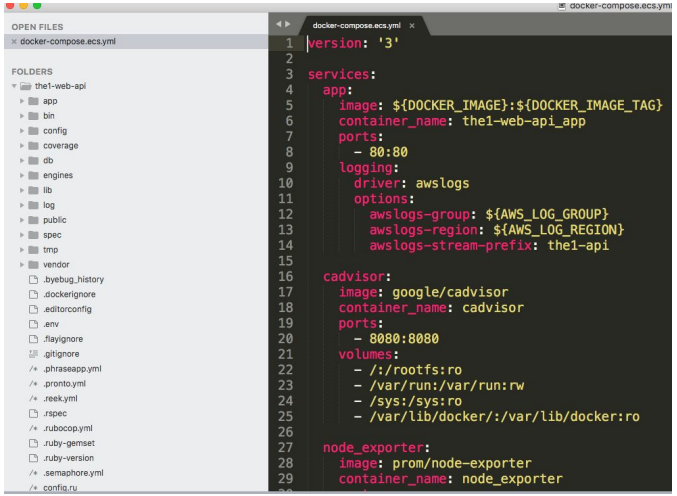
AWS Fargate

- 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

AWS Fargate

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)



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

main.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
```

Progress

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

To Do

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

nimble