Tugas Tambahan IF4031

Pengembangan Aplikasi Terdistribusi

Pembuatan Nginx ECS Cluster dengan Application Load Balancer menggunakan Terraform

Disusun oleh:

Malik Akbar Hashemi Rafsanjani 13520105

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

Deskripsi Tugas

Pada tugas ini, kami diminta untuk membuat Nginx Cluster di belakang aplikasi load balancer. Pembuatan tersebut menggunakan Fargate Launch Type pada Terraform. Selain itu, terdapat bonus untuk mengimplementasikan auto scaling untuk me-maintain 10 request per target (task).

Source Code

Source code dari tugas ini dapat diakses pada link berikut.
https://github.com/malikrafsan/Terraform-AWS-ECS-Auto-Scale

Implementasi

Pada implementasi tugas ini, digunakan referensi utama dari artikel berikut.
https://www.architect.io/blog/2021-03-30/create-and-manage-an-aws-ecs-cluster-with-terraform/

Pendefinisian Versi Provider Terraform AWS

Buat file `versions.tf`. yang berisi kode berikut.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}
provider "aws" {
  region = var.aws_region
  access_key = var.aws_access_key
  secret_key = var.aws_secret_key

  default_tags {
    tags = {
      Name = var.tag_name
    }
  }
}
```

Buat file `keys.tfvars`. yang berisi kode berikut.

```
aws_access_key = <access-key>
```

```
aws_secret_key = <secret-key>
```

ganti `<access-key>` dan `<secret-key>` menggunakan credential yang Anda miliki

Buat file `variables.tf` yang berisi kode berikut.

```
variable "aws_access_key" {
  type = string
  default = ""
}

variable "aws_secret_key" {
  type = string
  default = ""
}

variable "aws_region" {
  type = string
  default = "ap-southeast-2"
}

variable "tag_name" {
  type = string
  default = "PAT-demo-ecs-terraform"
}
```

Pada file `versions.tf`. tersebut, didefinisikan versi provider AWS yang digunakan. Didefinisikan pula region beserta credential yang dipakai, dan juga tag yang diberi. File `keys.tfvars` berisi credential yang dimiliki. Pemisahan file ini dilakukan agar project dapat dibuat publik dengan aman, tanpa khawatir nantinya credential key dipakai oleh pihak yang tidak berwenang. Selain itu, variabel-variabel yang dapat dibuat public didefinisikan pada file `variables.tf` agar memudahkan maintenance.

Setelah itu, jalankan perintah `terraform init` untuk menginisialisasi project terraform.

```
D:\KULIAH-AKBAR\SEMESTER-5\pat\mirror\Terraform-AWS-ECS-Auto-Scale>terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Installing hashicorp/aws v4.48.0...
- Installed hashicorp/aws v4.48.0 (signed by HashiCorp)

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Referensi

- https://developer.hashicorp.com/terraform/tutorials/aws-get-started/aws-build

Pendefinisian Konfigurasi Network

Buat file `networking.tf` yang berisi kode berikut.

```
resource "aws_vpc" "ecs_vpc" {
  cidr_block = var.cidr_block
}

resource "aws_subnet" "ecs_public_subnet" {
  vpc_id                  = aws_vpc.ecs_vpc.id
  count                   = length(var.public_cidr_block)
  cidr_block              = element(var.public_cidr_block, count.index)
  availability_zone       = element(var.availability_zones, count.index)
  map_public_ip_on_launch = true
}

resource "aws_subnet" "ecs_private_subnet" {
  vpc_id            = aws_vpc.ecs_vpc.id
  count             = length(var.private_cidr_block)
  cidr_block        = element(var.private_cidr_block, count.index)
  availability_zone = element(var.availability_zones, count.index)
}

resource "aws_internet_gateway" "ecs_internet_gateway" {
```

```
  vpc_id = aws_vpc.ecs_vpc.id
}

resource "aws_route" "ecs_ia" {
  route_table_id         = aws_vpc.ecs_vpc.main_route_table_id
  destination_cidr_block = var.destination_cidr_block
  gateway_id             = aws_internet_gateway.ecs_internet_gateway.id
}

resource "aws_eip" "ecs_nat_eip" {
  count = length(var.public_cidr_block)
  vpc   = true
  depends_on = [
    aws_internet_gateway.ecs_internet_gateway
  ]
}

resource "aws_nat_gateway" "ecs_nat_gateway" {
  count         = length(var.public_cidr_block)
     subnet_id             =   element(aws_subnet.ecs_public_subnet.*.id,
count.index)
  allocation_id = element(aws_eip.ecs_nat_eip.*.id, count.index)
}

resource "aws_route_table" "ecs_private_rt" {
  count  = length(aws_nat_gateway.ecs_nat_gateway)
  vpc_id = aws_vpc.ecs_vpc.id

  route {
    cidr_block     = var.cidr_block_route_table
        nat_gateway_id  =  element(aws_nat_gateway.ecs_nat_gateway.*.id,
count.index)
  }
}

resource "aws_route_table_association" "subnet_route_private" {
  count          = length(var.private_cidr_block)
    subnet_id             =  element(aws_subnet.ecs_private_subnet.*.id,
count.index)
      route_table_id    =   element(aws_route_table.ecs_private_rt.*.id,
count.index)
}
```

Tambahkan kode berikut ke file `variables.tf`

```
variable "availability_zones" {
```

```
  type    = list(any)
  default = ["ap-southeast-2a", "ap-southeast-2b"]
}

variable "cidr_block" {
  type    = string
  default = "10.32.0.0/16"
}

variable "public_cidr_block" {
  type    = list(any)
  default = ["10.32.0.0/26", "10.32.1.0/26"]
}

variable "private_cidr_block" {
  type    = list(any)
  default = ["10.32.2.0/26", "10.32.3.0/26"]
}

variable "destination_cidr_block" {
  type    = string
  default = "0.0.0.0/0"
}

variable "cidr_block_route_table" {
  type    = string
  default = "0.0.0.0/0"
}
```

Pada kode di atas, didefinisikan konfigurasi VPC AWS. VPC (Virtual Private Cloud) merupakan layanan yang memungkinkan untuk meluncurkan sumber daya AWS pada jaringan virtual yang terisolasi. Hal ini membuat resource yang digunakan pada project ini seolah-olah terisolasi dari project lain. Selain itu, didefinisikan 4 subnet, 2 publik dan 2 privat. Subnet publik dan subnet private memiliki konfigurasi yang mirip. Perbedaannya ialah pada subnet public didefinisikan konfigurasi `map_public_ip_on_launch` bernilai `true`.

Subnet tersebut dipecah menjadi publik dan private untuk memisahkan resource yang dapat diakses secara publik dan resource yang hanya dapat diakses secara private. Resource seperti load balancer yang harus dapat diakses secara publik akan dimasukkan ke subnet publik. Sementara resource yang seharusnya hanya dapat diakses secara private seperti service nantinya akan dimasukkan ke dalam subnet private.

Selain itu, didefinisikan 6 resource lain untuk mengatur jaringan dan komunikasi dari dan keluar internet pada VPC, yaitu `aws_internet_gateway`, `aws_route`, `aws_eip`, `aws_nat_gateway`, `aws_route_table`, dan `aws_route_table_association`. internet

gateway berfungsi sebagai penyedia resouce untuk membuat VPC Internet Gateway. `aws_route` digunakan untuk membuat *routing table entry* pada *VPC routing table*. `aws_eip` merupakan sumber daya yang menyediakan elastic IP. `aws_nat_gateway` menyediakan sumber daya untuk membuat VPC NAT Gateway yang berfungsi untuk memperbolehkan sumber daya di VPC berkomunikasi dengan internet namun mencegah komunikasi dari luar ke VPC. `aws_route_table` menyediakan sumber daya untuk membuat VPC routing table. `aws_route_table_association` menyediakan sumber daya untuk mengelola *routing table* utama dari VPC.

Referensi

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/security_group

Pendefinisian Konfigurasi Security

Buat file `security.tf` yang berisi kode berikut.

```
resource "aws_security_group" "ecs_lb_sg" {
  vpc_id = aws_vpc.ecs_vpc.id
  name   = "lb-security-group"

  ingress {
    protocol    = "tcp"
    from_port   = 80
    to_port     = 80
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    protocol    = "-1"
    from_port   = 0
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_security_group" "ecs_task_sg" {
  vpc_id = aws_vpc.ecs_vpc.id
  name   = "task-security-group"

  ingress {
    protocol        = "tcp"
    from_port       = 80
    to_port         = 80
    security_groups = [aws_security_group.ecs_lb_sg.id]
```

```
  }

  egress {
    protocol    = "-1"
    from_port   = 0
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

Kode di atas berisi konfigurasi security pada project ini, terutama pada ecs load balancer dan ecs task. Pada konfigurasi load balancer, didefinisikan bahwa traffic yang boleh masuk hanya bisa dari dan ke port 80 dengan protokol TCP, sesuai pada blok `ingress`. Sementara itu, pada blok `egress` didefinisikan bahwa traffic keluar load balancer dapat kemana saja, dari port apa saja dan ke port apa saja. Selain itu, didefinisikan juga konfigurasi untuk ecs task dengan konfigurasi yang mirip dengan load balancer.

Referensi

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/security_group

Pendefinisian Konfigurasi Load Balancer

Buat file `load-balancer.tf` yang berisi kode berikut.

```
resource "aws_lb" "ecs_lb" {
  name               = "ecs-lb-13520105"
  internal           = false
  subnets            = aws_subnet.ecs_public_subnet.*.id
  security_groups    = [aws_security_group.ecs_lb_sg.id]
  load_balancer_type = "application"
}

resource "aws_lb_target_group" "ecs_lb_tg" {
  name        = "ecs-lb-tg-13520105"
  port        = 80
  protocol    = "HTTP"
  vpc_id      = aws_vpc.ecs_vpc.id
  target_type = "ip"
}

resource "aws_lb_listener" "ecs_lb_listener" {
  load_balancer_arn = aws_lb.ecs_lb.id
  port              = "80"
```

```
  protocol          = "HTTP"

  default_action {
    target_group_arn = aws_lb_target_group.ecs_lb_tg.arn
    type             = "forward"
  }
}
```

Pada kode di atas, `aws_lb` mendefinisikan load balancer dan di-attach ke public subnet dengan security group yang telah didefinisikan sebelumnya. Resource `aws_lb_target_group` mendefinisikan target group yang digunakan untuk resource load balancer. Sementara itu, `aws_lb_listener` digunakan untuk mendefinisikan listener pada load balancer.

Referensi

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/lb
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/lb_target_group
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/lb_listener

Pendefinisian Konfigurasi Utama

Buat file `main.tf` yang berisi kode berikut.

```
resource "aws_ecs_task_definition" "ecs_task" {
  family                   = "ecs-nginx-service-13520105"
  network_mode             = "awsvpc"
  requires_compatibilities = ["FARGATE"]
  cpu                      = var.cpu_capacity
  memory                   = var.memory_capacity
  container_definitions    = <<DEFINITION
  [
    {
      "name" : "nginx",
      "image" : "nginx:1.23.1",
      "cpu" : 256,
      "memory": 1024,
      "network_mode": "awsvpc",
      "essential" : true,
      "portMappings": [
        {
          "containerPort" : 80,
          "hostPort" : 80
        }
```

```
      ]
    }
  ]
  DEFINITION
}

resource "aws_ecs_cluster" "main" {
  name = "nginx-cluster-13520105"
}

resource "aws_ecs_service" "ecs_service" {
  name            = "ecs-nginx-service-13520105"
  cluster         = aws_ecs_cluster.main.id
  task_definition = aws_ecs_task_definition.ecs_task.arn
  desired_count   = var.app_count
  launch_type     = "FARGATE"

  network_configuration {
    security_groups = [aws_security_group.ecs_task_sg.id]
    subnets         = aws_subnet.ecs_private_subnet.*.id
  }

  load_balancer {
    target_group_arn = aws_lb_target_group.ecs_lb_tg.id
    container_name   = "nginx"
    container_port   = 80
  }

  depends_on = [
    aws_lb_listener.ecs_lb_listener
  ]
}

output "aws_lb_hostname" {
  value = aws_lb.ecs_lb.dns_name
}
```

Tambahkan kode berikut ke file `variables.tf`

```
variable "cpu_capacity" {
  type = number
  default = 256
}

variable "memory_capacity" {
  type = number
```

```
  default = 1024
}

variable "app_count" {
  type = number
  default = 2
}
```

Kode di atas mendefinisikan task, kluster, service dan output yang digunakan pada project ini. `aws_ecs_task_definition` mendefinisikan task yang digunakan, yaitu nginx. Idenfifier `requires_compatibilities` menspesifikkan platform yang digunakan adalah Fargate, bukan EC2. Pada container tersebut, digunakan image nginx dengan port yang telah ditentukan yaitu 80 dengan resource CPU sebesar 256 (1024 CPU unit ekuivalen dengan 1 vCPU) dan memory 1024 MB. Selain itu didefinisikan nama dari ecs cluster, yaitu `nginx-cluster-13520105`.

`aws_ecs_service` mengumpulkan konfigurasi yang telah ditentukan sebelumnya, seperti task, cluster, konfigurasi network, dan load balancer. Digunakan tipe Fargate sehingga tidak diperlukan manajemen dari EC2 instance. Sesuai dari konfigurasi network, resource ini akan berjalan di atas subnet private. Lalu didefinisikan pula atribut array `depends_on` berisi load balancer. Hal ini membuat service tidak perlu dibuat sampai load balancer meminta.

Selain itu, didefinisikan output dari terraform. Hal ini berfungsi agar `aws_lb.ecs_lb.dns_name` menjadi value output. Hal ini membuat URL berupa nama DNS dari ecs load balancer akan ditampilkan ke layar setiap konfigurasi terraform di-apply.

Referensi

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/ecs_task_definition
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/ecs_cluster
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/ecs_service
- https://developer.hashicorp.com/terraform/language/values/outputs

Bonus: Pendefinisian Konfigurasi Auto Scaling

Buat file `auto-scale.tf` yang berisi kode berikut

```
resource "aws_appautoscaling_target" "ecs_target" {
  min_capacity       = var.min_capacity
  max_capacity       = var.max_capacity
           resource_id                                                =
"service/${aws_ecs_cluster.main.name}/${aws_ecs_service.ecs_service.name
```

```
}"
  scalable_dimension = "ecs:service:DesiredCount"
  service_namespace  = "ecs"
}

resource "aws_appautoscaling_policy" "ecs_target_request_count" {
  name               = "ecs-target-request-count"
  policy_type        = "TargetTrackingScaling"
  resource_id        = aws_appautoscaling_target.ecs_target.resource_id
                                    scalable_dimension                  =
aws_appautoscaling_target.ecs_target.scalable_dimension
                              service_namespace                         =
aws_appautoscaling_target.ecs_target.service_namespace

  target_tracking_scaling_policy_configuration {
    predefined_metric_specification {
      predefined_metric_type = "ALBRequestCountPerTarget"
                         resource_label                                 =
"app/${aws_lb.ecs_lb.name}/${basename("${aws_lb.ecs_lb.id}")}/targetgrou
p/${aws_lb_target_group.ecs_lb_tg.name}/${basename("${aws_lb_target_grou
p.ecs_lb_tg.id}")}"
    }
    target_value = var.target_value_request_count
    scale_in_cooldown  = var.scale_in_cooldown
    scale_out_cooldown = var.scale_out_cooldown
  }
}
```

Tambahkan kode berikut ke file `variables.tf`

```
variable "min_capacity" {
  type = number
  default = 2
}

variable "max_capacity" {
  type = number
  default = 15
}

variable "target_value_request_count" {
  type = number
  default = 10
}

variable "scale_in_cooldown" {
```

```
  type = number
  default = 150
}

variable "scale_out_cooldown" {
  type = number
  default = 150
}
```

Pada kode di atas, didefinisikan konfigurasi untuk target autoscaling dan juga policy yang digunakan dalam autoscaling. `aws_appautoscaling_target` mendefinisikan resource mana yang akan dijadikan target autoscaling, beserta minimal dan maksimal kapasitas. `aws_appautoscaling_policy` mendefinisikan pada keadaan apa aplikasi ini akan di-scale. Pada tugas ini, aplikasi akan di-scale berdasarkan jumlah resource yang diterima oleh target, dengan jumlah request sesuai pada target value. Selain itu juga didefinisikan cooldown dari scaling, baik ketika scale in maupun scale out.

Kesulitan yang Dihadapi

Terdapat beberapa kesulitan yang saya hadapi dalam pengerjaan tugas ini, antara lain:

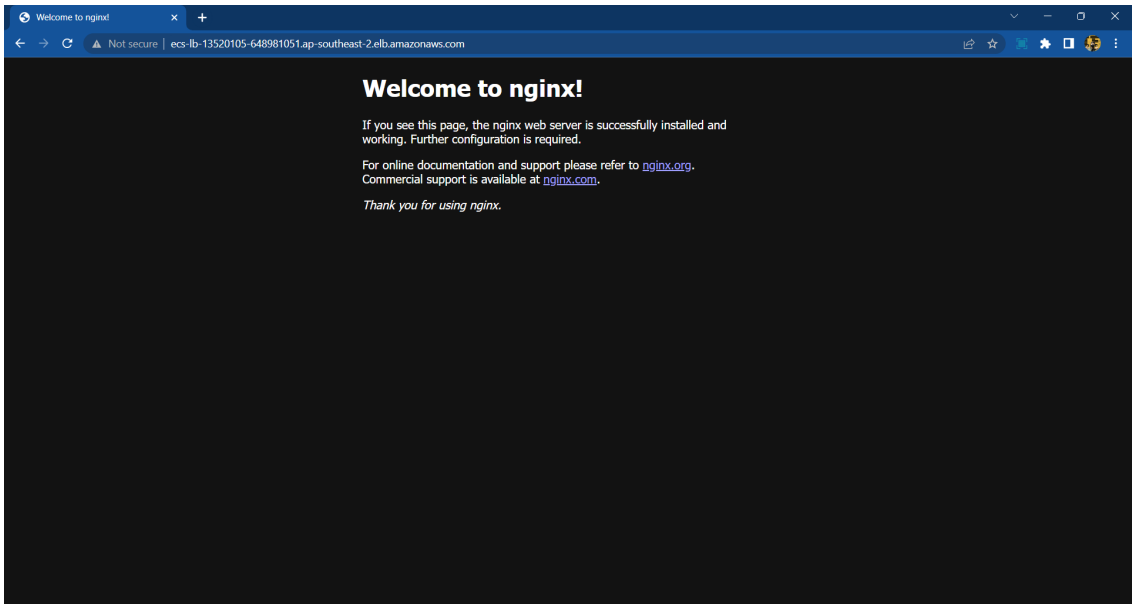
- Banyak istilah-istilah yang tidak familiar pada AWS dan terraform sehingga perlu banyak mempelajari makna istilah-istilah tersebut.
- Konfigurasi jaringan pada cluster yang memerlukan pengetahuan cukup detail baik dari sisi networking maupun dari infrastruktur AWS.
- Mengkonfigurasikan ulang project terraform dari yang awalnya memakai key AWS milik akun pribadi menjadi menggunakan key AWS yang disediakan pada tugas kali ini.

Pengujian yang Dilakukan

Aplikasi dapat diakses melalui link berikut: http://ecs-lb-13520105-648981051.ap-southeast-2.elb.amazonaws.com/

1. Pengujian melalui Pengaksesan Web Browser

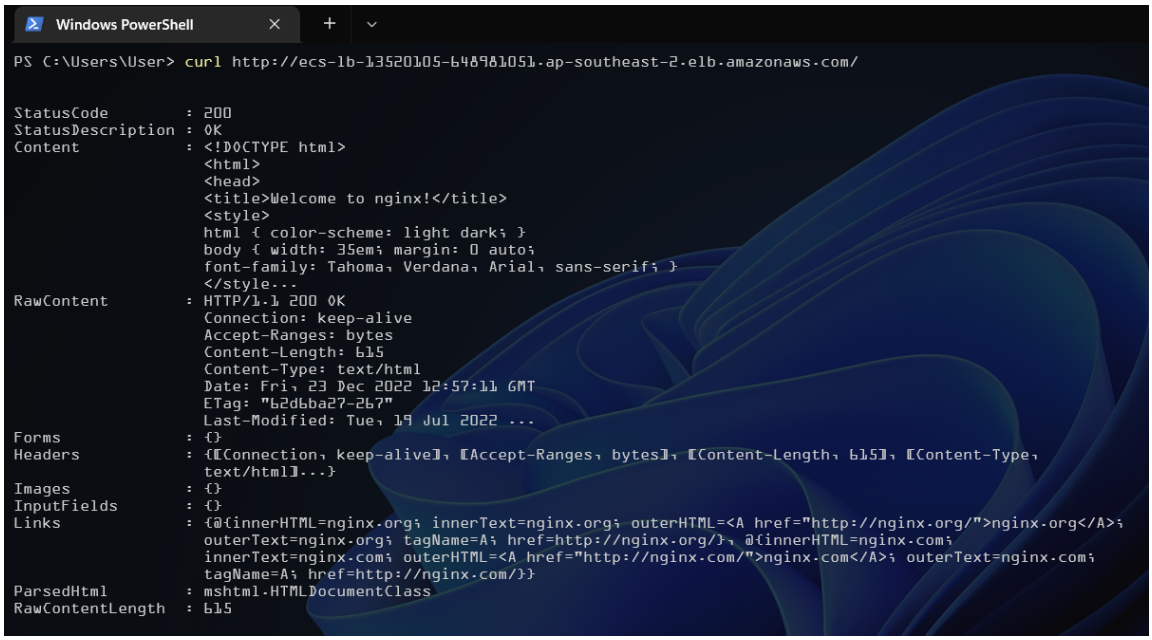
Aplikasi dapat diuji dengan mengakses link yang telah tertera



Pada pngujian ini, aplikasi mengembalikan response berupa default page dari nginx seperti pada gambar di atas. Hal ini menunjukkan bahwa nginx telah bekerja dengan baik pada aplikasi ini.

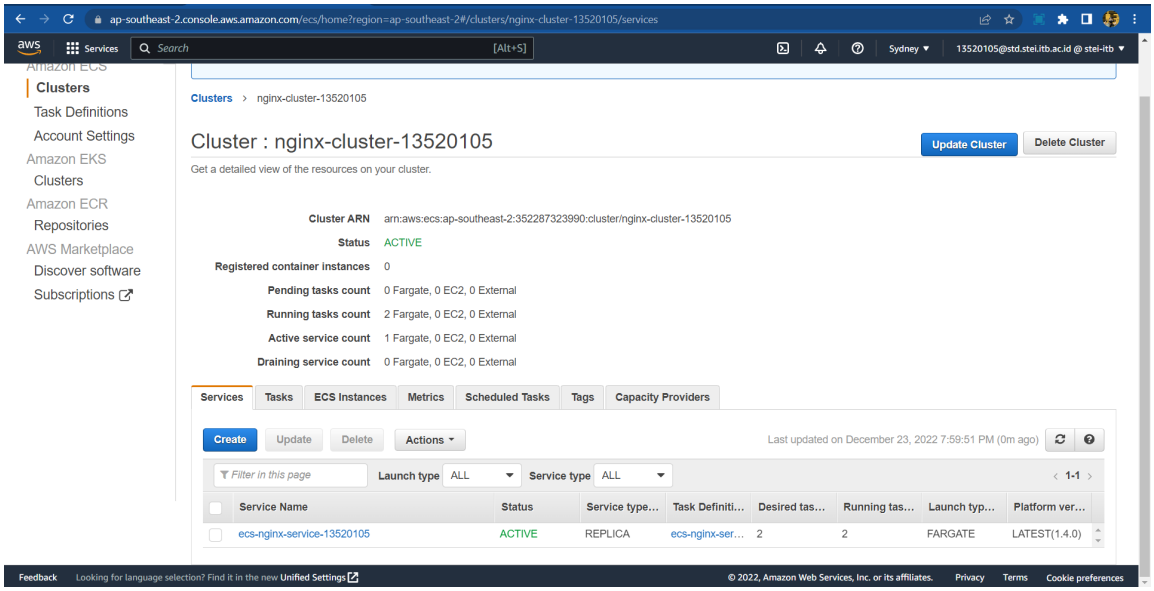
2. Pengujian melalui Pengaksesan dengan Curl

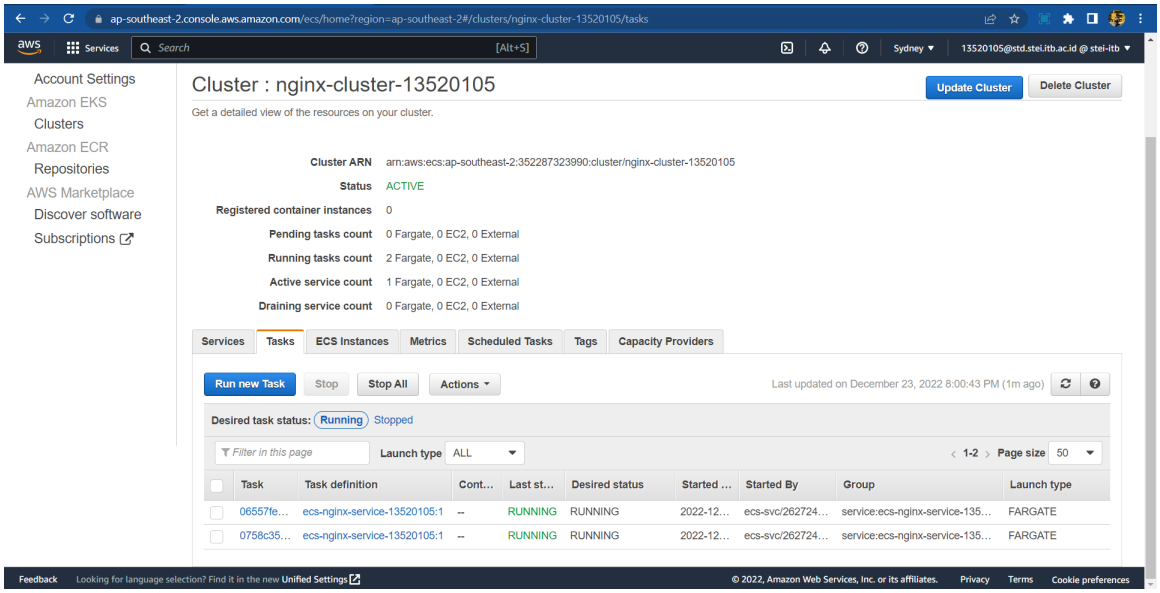
Aplikasi ini juga dapat diuji dengan melakukan GET request menggunakan curl, dengan hasil sebagai berikut.



Pada pengujian ini juga didapat hasil sukses serupa dengan pengujian sebelumnya.

3. Pengecekan pada AWS



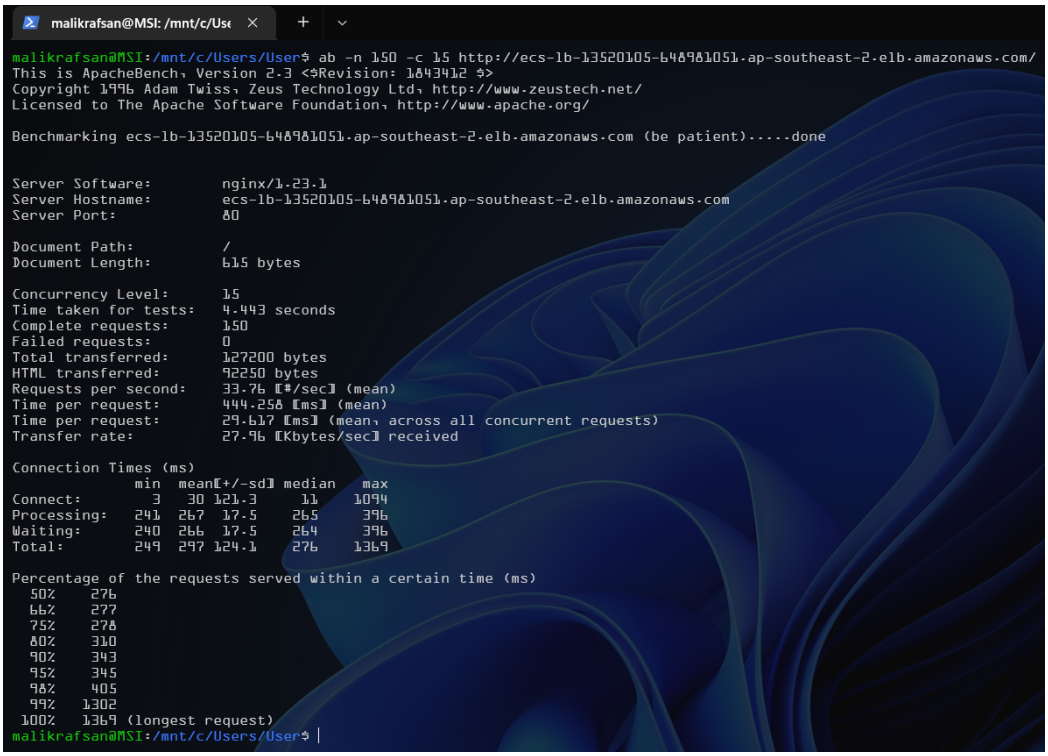


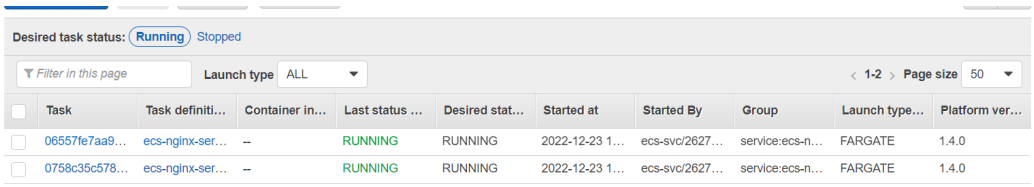
Pada kedua gambar di atas, terlihat tangkapan layar pada AWS console terkait detail konfigurasi service dan task yang berjalan. Konfigurasi tersebut sesuai dengan yang telah didefinisikan pada implementasi sebelumnya.

4. Pengujian Bonus Auto Scaling

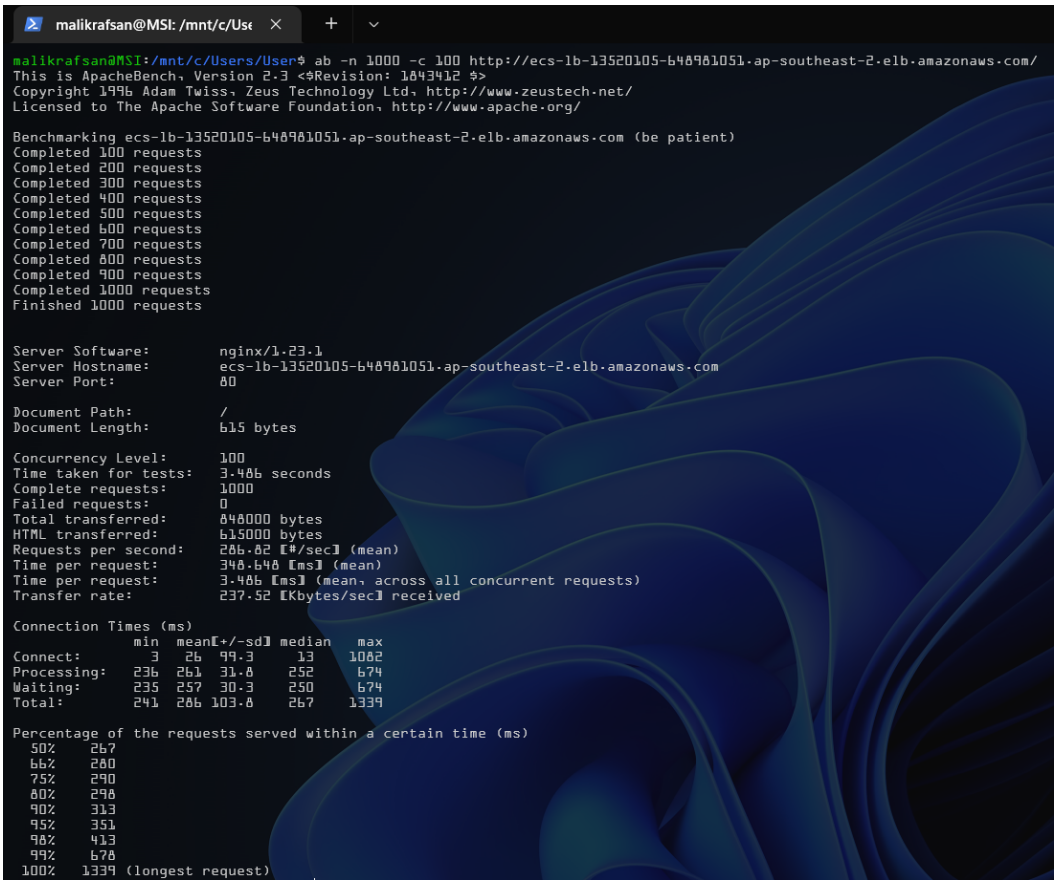
Pada project ini, diimplementasikan auto scaling untuk me-maintain 10 request per target, dalam kasus ini task, dengan minimal instance berjumlah 2 dan maksimal instance berjumlah 15. Untuk mesimulasikan request yang banyak secara konkuren, digunakan software Apache Bench (AB).

a. 150 Request dengan 15 Concurent Request

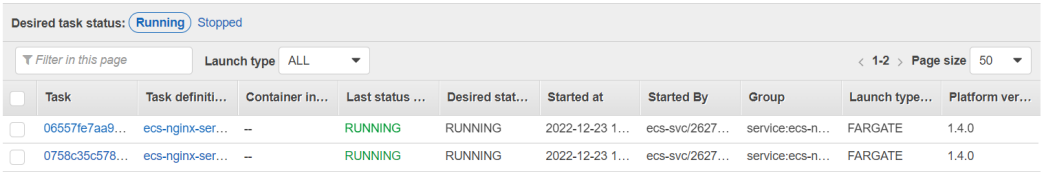




b. 1000 Request dengan 100 Concurent Request



```
malikrafsan@MSI: /mnt/c/Use
malikrafsan@MSI:/mnt/c/Users/User$ ab -n 1000 -c 100 http://ecs-lb-135201105-648981051.ap-southeast-2.elb.amazonaws.com/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ecs-lb-135201105-648981051.ap-southeast-2.elb.amazonaws.com (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        nginx/1.23.1
Server Hostname:        ecs-lb-135201105-648981051.ap-southeast-2.elb.amazonaws.com
Server Port:            80

Document Path:          /
Document Length:        615 bytes

Concurrency Level:      100
Time taken for tests:   3.486 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      848000 bytes
HTML transferred:       615000 bytes
Requests per second:    286.82 [#/sec] (mean)
Time per request:       348.648 [ms] (mean)
Time per request:       3.486 [ms] (mean, across all concurrent requests)
Transfer rate:          237.52 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        3   26  99.3     13    1082
Processing:   236  261  31.8    252     674
Waiting:      235  257  30.3    250     674
Total:        241  286 103.8    267    1339

Percentage of the requests served within a certain time (ms)
  50%    267
  66%    280
  75%    290
  80%    298
  90%    313
  95%    351
  98%    413
  99%    678
 100%   1339 (longest request)
```

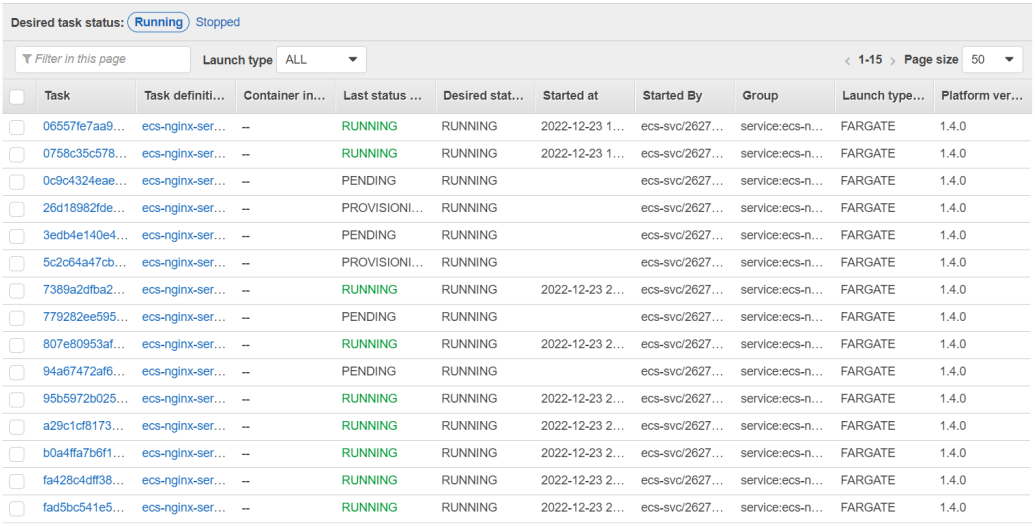


Desired task status: Running Stopped

Filter in this page | Launch type ALL | 1-2 | Page size 50

Task	Task definiti…	Container in…	Last status …	Desired stat…	Started at	Started By	Group	Launch type…	Platform ver…
06557fe7aa9…	ecs-nginx-ser…	--	RUNNING	RUNNING	2022-12-23 1…	ecs-svc/2627…	service:ecs-n…	FARGATE	1.4.0
0758c35c578…	ecs-nginx-ser…	--	RUNNING	RUNNING	2022-12-23 1…	ecs-svc/2627…	service:ecs-n…	FARGATE	1.4.0

c. 10.000 Request dengan 1000 Concurent Request



Desired task status: Running Stopped

Filter in this page | Launch type: ALL | 1-15 | Page size: 50

Task	Task definiti...	Container in...	Last status ...	Desired stat...	Started at	Started By	Group	Launch type...	Platform ver...
06557fe7aa9...	ecs-nginx-ser...	--	RUNNING	RUNNING	2022-12-23 1...	ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
0758c35c578...	ecs-nginx-ser...	--	RUNNING	RUNNING	2022-12-23 1...	ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
0c9c4324eae...	ecs-nginx-ser...	--	PENDING	RUNNING		ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
26d18982fde...	ecs-nginx-ser...	--	PROVISIONI...	RUNNING		ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
3edb4e140e4...	ecs-nginx-ser...	--	PENDING	RUNNING		ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
5c2c64a47cb...	ecs-nginx-ser...	--	PROVISIONI...	RUNNING		ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
7389a2dfba2...	ecs-nginx-ser...	--	RUNNING	RUNNING	2022-12-23 2...	ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
779282ee595...	ecs-nginx-ser...	--	PENDING	RUNNING		ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
807e80953af...	ecs-nginx-ser...	--	RUNNING	RUNNING	2022-12-23 2...	ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
94a67472af6...	ecs-nginx-ser...	--	PENDING	RUNNING		ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
95b5972b025...	ecs-nginx-ser...	--	RUNNING	RUNNING	2022-12-23 2...	ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
a29c1cf8173...	ecs-nginx-ser...	--	RUNNING	RUNNING	2022-12-23 2...	ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
b0a4ffa7b6f1...	ecs-nginx-ser...	--	RUNNING	RUNNING	2022-12-23 2...	ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
fa428c4dff38...	ecs-nginx-ser...	--	RUNNING	RUNNING	2022-12-23 2...	ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0
fad5bc541e5...	ecs-nginx-ser...	--	RUNNING	RUNNING	2022-12-23 2...	ecs-svc/2627...	service:ecs-n...	FARGATE	1.4.0

```
malikrafsan@MSI:/mnt/c/Users/User$ ab -n 10000 -c 1000 http://ecs-lb-133520105-648981051.ap-southeast-2.elb.amazonaws.com/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ecs-lb-133520105-648981051.ap-southeast-2.elb.amazonaws.com (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
apr_pollset_poll: The timeout specified has expired (70007)
Total of 9999 requests completed
malikrafsan@MSI:/mnt/c/Users/User$
```

5. Evaluasi dan Analisis

Pada pengujian yang telah dilakukan, dapat disimpulkan bahwa baik dari load balancer, service, task, dan network bekerja dengan baik. Dapat dilihat bahwa aplikasi dapat diakses dengan baik dan mengembalikan response default dari nginx. Selain itu, aplikasi melakukan auto scaling ketika diakses dengan request yang banyak secara bersamaan. Namun, aplikasi tidak melakukan auto scaling sesuai dengan target value yang telah ditentukan, yaitu 10 request per target. Hal ini dapat disebabkan oleh berbagai hal, antara lain:

- #### Adanya cooldown scaling

  Pada project ini, didefinisikan cooldown untuk scale in dan scale out selama 150 detik. Hal ini menyebabkan ketika aplikasi di-hit dengan request yang banyak di satu waktu, ia tidak akan secara langsung melakukan auto scaling.

- #### Request yang sederhana

  Pada pengujian dilakukan request yang sederhana yaitu GET request pada static HTML. Hal ini membuat server dapat menangani request dengan sangat mudah.

Selain itu, karena request berupa GET dan dilakukan pengaksesan static resource, mungkin saja terjadi caching sehingga level konkurensi turun.

Analisis di atas dapat dikatakan sebatas spekulasi. Untuk mendapatkan hasil eksak, diperlukan pengujian lebih lanjut, yang mana di luar scope dari tugas ini.

Pelajaran yang Didapatkan

Dari tugas ini, saya banyak belajar mengenai pengkonfigurasian infrastruktur secara umum, terutama menggunakan Terraform dan AWS. Saya juga belajar sangat banyak mengenai networking pada suatu infrastruktur cluster di mana digunakan subnet publik dan subnet privat, beserta bagaimana penggunaan routing melalui routing table. Saya juga belajar mengenai hands on load balancer dan auto scaling aplikasi, yang mana sangat bermanfaat pada aplikasi terdistribusi. Hal-hal ini merupakan hal-hal yang cukup baru untuk saya dan saya merasa sangat senang mendapatkan pengalaman baru ini.