NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Coaching - ECS (IaC + Automated app deployment)

Cloud Infrastructure Engineering

Nanyang Technological University & Skills Union - 2022/2023

Overview

By now, you would have had some experience with the following things:

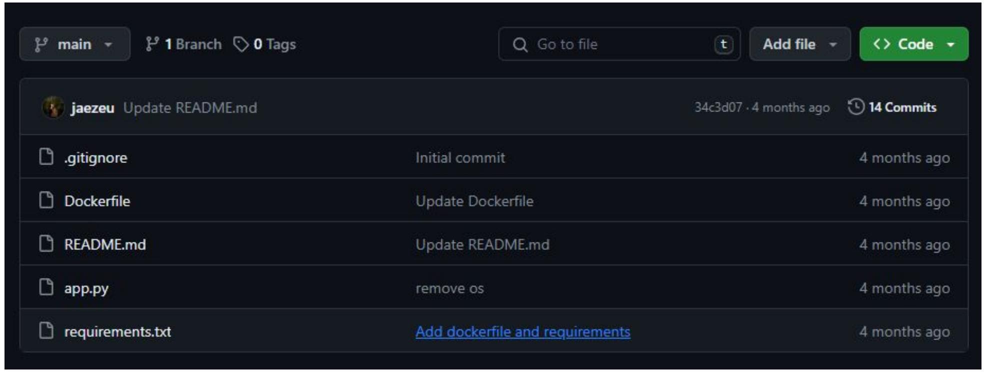
- Creating a ECR Repository (Console)
- Building and pushing an image to ECR
- Creating a ECS Cluster + Task Definition + Service (Console)

Therefore, in today’s activity we would look into performing the above actions through Terraform and Github Actions.

Pre-Requisites

If you do not already have, create your own git repository to store your app code (If the links below are moved, you can refer to next slide for the contents as well).

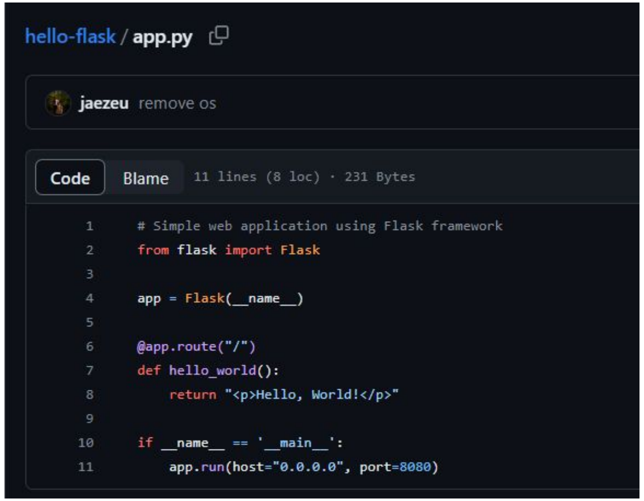
- app.py - hello-flask/app.py at main · jaezeu/hello-flask · GitHub
- requirements.txt - hello-flask/requirements.txt at main · jaezeu/hello-flask · GitHub
- Dockerfile - hello-flask/Dockerfile at main · jaezeu/hello-flask · GitHub



jaezeu Update README.md 34c3d07 · 4 months ago 14 Commits

.gitignore	Initial commit	4 months ago
Dockerfile	Update Dockerfile	4 months ago
README.md	Update README.md	4 months ago
app.py	remove os	4 months ago
requirements.txt	Add dockerfile and requirements	4 months ago

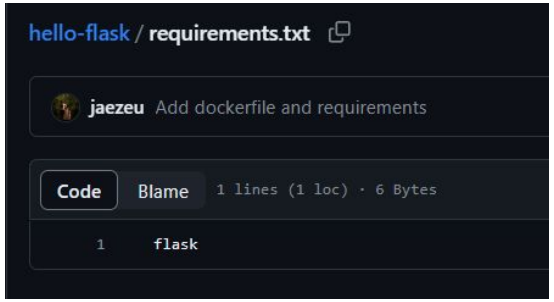
File Contents


hello-flask / app.py

jaezeu remove os

Code Blame 11 lines (8 loc) · 231 Bytes

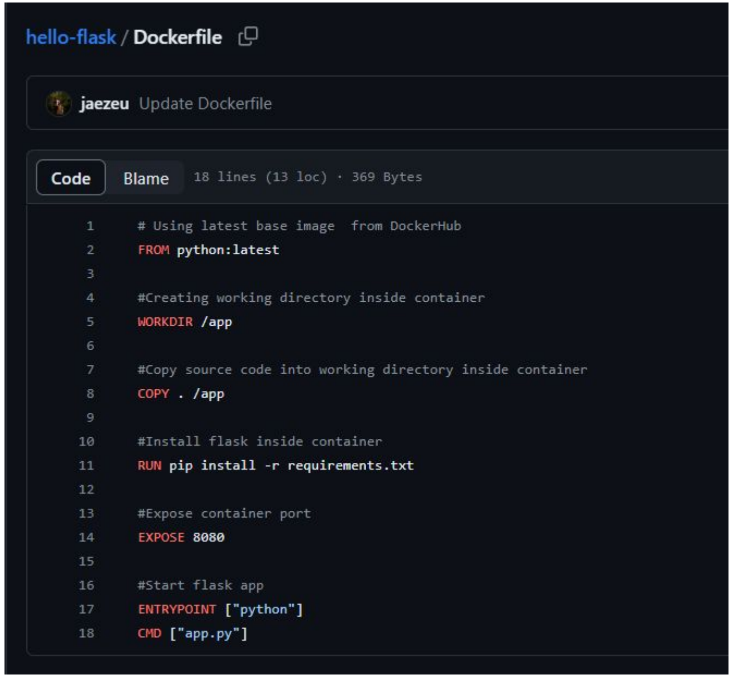
```python
# Simple web application using Flask framework
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"

if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080)
```


hello-flask / requirements.txt

jaezeu Add dockerfile and requirements

Code Blame 1 lines (1 loc) · 6 Bytes

```
flask
```


hello-flask / Dockerfile

jaezeu Update Dockerfile

Code Blame 18 lines (13 loc) · 369 Bytes

```dockerfile
# Using latest base image  from DockerHub
FROM python:latest

#Creating working directory inside container
WORKDIR /app

#Copy source code into working directory inside container
COPY . /app

#Install flask inside container
RUN pip install -r requirements.txt

#Expose container port
EXPOSE 8080

#Start flask app
ENTRYPOINT ["python"]
CMD ["app.py"]
```

Activity Part 1(Terraform with base image)

- Create a new Git Repository with a .gitignore file(**Terraform**) and a README.md file. **<u>This would be your repository to store your Infra-as-Code only</u>**.
- Spend about 45 - 60 mins to do the following in Terraform (You can use the default VPC and public subnets):
  - Use ap-southeast-1 as your provider region
  - Create a ECR repository using aws_ecr_repository | Resources | hashicorp/aws | Terraform | Terraform Registry
    - Remember to add **<u>force_delete = true</u>** in your code, so that you can run terraform destroy later.
  - Create a ECS Cluster + Task Definition + Service using the following module: terraform-aws-modules/terraform-aws-ecs: Terraform module to create AWS ECS resources 🇺🇦 (github.com)
    - Image: `public.ecr.aws/docker/library/httpd:latest`
    - cpu: 512
    - memory: 1024
    - deployment_minimum_healthy_percent: 100
  - Create the security group for your ECS Service (port 80 if you're using the httpd image). You can use the following module:
    - terraform-aws-modules/terraform-aws-security-group: Terraform module to create AWS Security Group resources 🇺🇦 (github.com)
- Run terraform apply to create your infra and then see if you can view your task in the browser using its public IP.

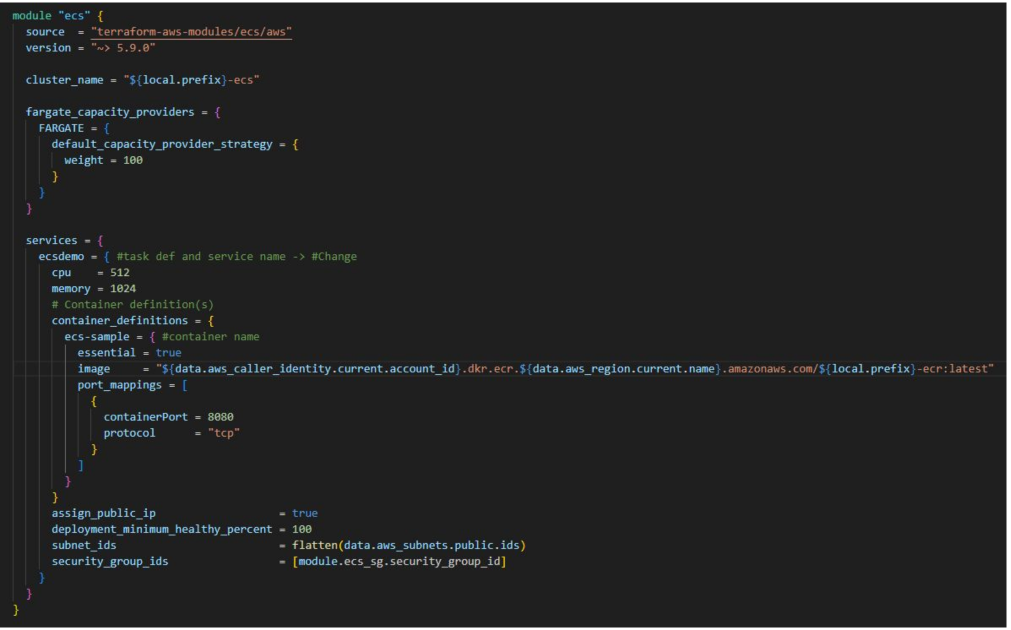
Activity Part 2(Terraform with custom image)

- Update your image parameter to the following format:
  - <ACCOUNT-ID>.dkr.ecr.<REGION>.amazonaws.com/<YOUR-ECR-NAME>:latest
- Try to automatically interpolate the <ACCOUNT-ID> and <REGION> in the image parameter above using terraform data sources.
- Update your security group inbound to port 8080
- Then run terraform apply again

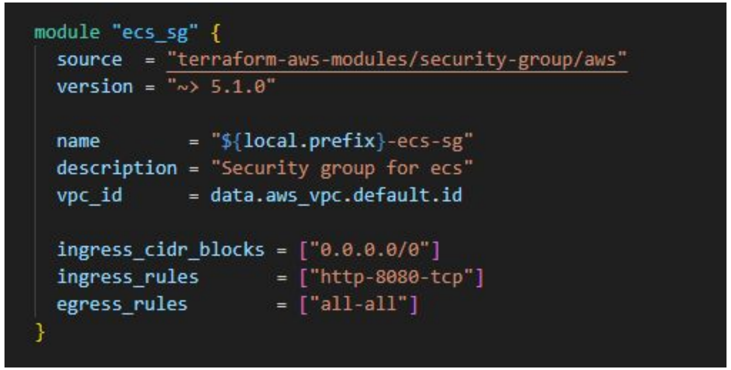
Your service should be unhealthy now. Why is it so?

Remember to also push your code to the remote repository

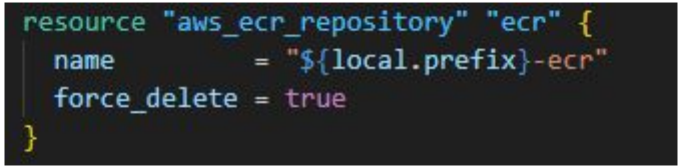
Activity Part 2(Terraform with custom image)



```
module "ecs" {
  source  = "terraform-aws-modules/ecs/aws"
  version = "~> 5.9.0"

  cluster_name = "${local.prefix}-ecs"

  fargate_capacity_providers = {
    FARGATE = {
      default_capacity_provider_strategy = {
        weight = 100
      }
    }
  }

  services = {
    ecsdemo = { #task def and service name -> #Change
      cpu    = 512
      memory = 1024
      # Container definition(s)
      container_definitions = {
        ecs-sample = { #container name
          essential = true
          image     = "${data.aws_caller_identity.current.account_id}.dkr.ecr.${data.aws_region.current.name}.amazonaws.com/${local.prefix}-ecr:latest"
          port_mappings = [
            {
              containerPort = 8080
              protocol      = "tcp"
            }
          ]
        }
      }
      assign_public_ip                   = true
      deployment_minimum_healthy_percent = 100
      subnet_ids                         = flatten(data.aws_subnets.public.ids)
      security_group_ids                 = [module.ecs_sg.security_group_id]
    }
  }
}
```



```
module "ecs_sg" {
  source  = "terraform-aws-modules/security-group/aws"
  version = "~> 5.1.0"

  name        = "${local.prefix}-ecs-sg"
  description = "Security group for ecs"
  vpc_id      = data.aws_vpc.default.id

  ingress_cidr_blocks = ["0.0.0.0/0"]
  ingress_rules       = ["http-8080-tcp"]
  egress_rules        = ["all-all"]
}
```



```
resource "aws_ecr_repository" "ecr" {
  name         = "${local.prefix}-ecr"
  force_delete = true
}
```

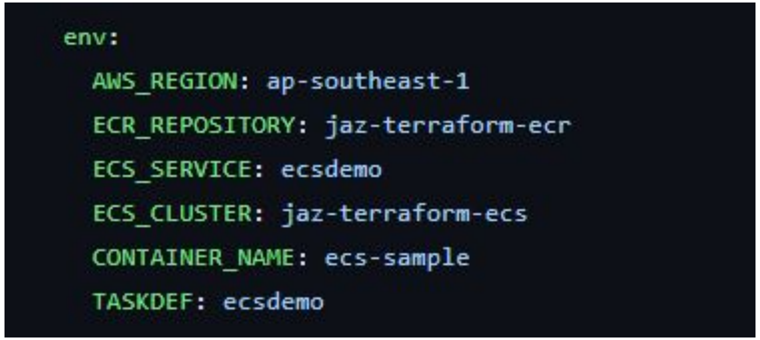
Activity Part 3 (Github Actions from App Repository)

In your application repository, Remember to create the following secrets for your workflow to be able to interact with your AWS account (Settings -> Secrets and Variables -> Actions -> New repository secret).

- AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY

Once done, clone your repository and add the following workflow yml file in the .github/workflows folder:

- https://github.com/jaezeu/hello-flask/blob/main/.github/workflows/deploy-to-ecs.yml
- Remember to update the following values based on your infra, before pushing your code to your remote repository:



```
env:
  AWS_REGION: ap-southeast-1
  ECR_REPOSITORY: jaz-terraform-ecr
  ECS_SERVICE: ecsdemo
  ECS_CLUSTER: jaz-terraform-ecs
  CONTAINER_NAME: ecs-sample
  TASKDEF: ecsdemo
```

Post-Activity

Remember to run terraform destroy at the end of class to destroy the infrastructure that you’ve created today.