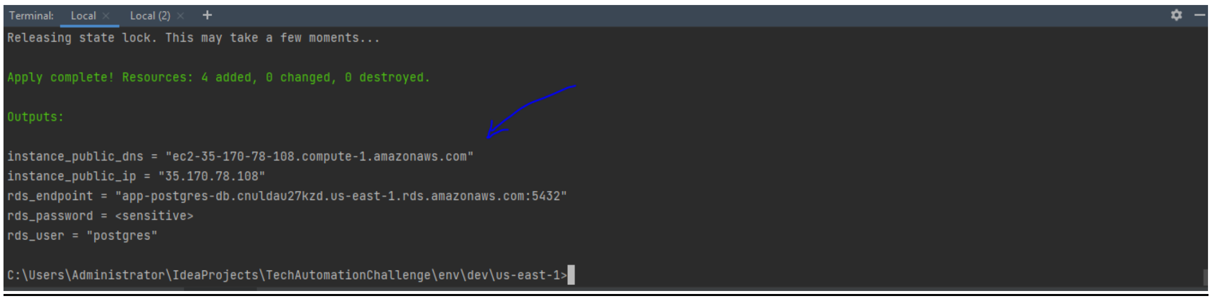
Servian Test challenge Application

Link to Source code for the Test Challenge application:

https://github.com/Akshaychiruguri/Serviantestchallange.git



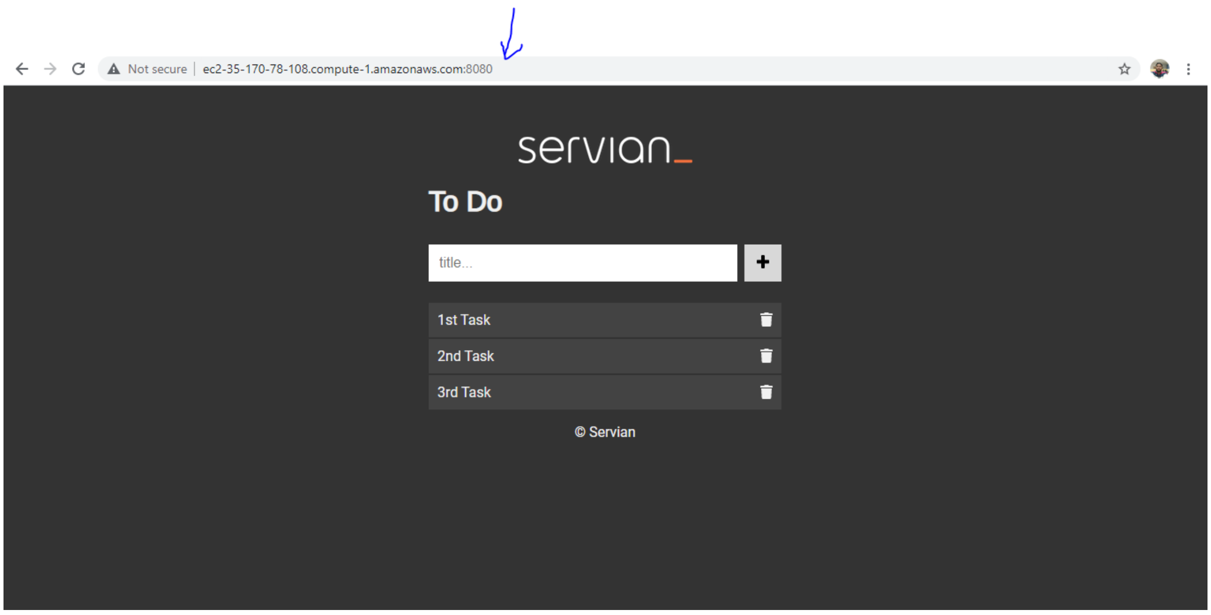
```
Terminal:  Local ×   Local (2) ×   +
Releasing state lock. This may take a few moments...

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

instance_public_dns = "ec2-35-170-78-108.compute-1.amazonaws.com"
instance_public_ip = "35.170.78.108"
rds_endpoint = "app-postgres-db.cnuldau27kzd.us-east-1.rds.amazonaws.com:5432"
rds_password = <sensitive>
rds_user = "postgres"

C:\Users\Administrator\IdeaProjects\TechAutomationChallenge\env\dev\us-east-1>
```



Comments for the source code:

- Default security group with open ssh will be created by the terraform here , we can create custom security groups and
- Custom VPC’s to further strengthen the security
- ingress also can be allowed from selected CIDR range or IPs to restrict attack surface.
- For increasing security an ALB in public zone can be attached to EC2 in private subnet , WAF and DDOS can be enabled on cloud front and a attached to LB as source.
- The RDS secrets values can be retrieved from vault or AWS Service store manager to further enhance the secrets management
- I have hosted web service using EC2 and docker compose to keep it simple, we can also use Load balancers, EKS or ECS to deploy. Using Fargate and containersation services will also be a good practice.

- As this is a test application, to keep it simple and cost effective I have chosen ec2 with docker compose and DynamoDB in the backend to run the webpage.

Possibility 2

Deploying application on cloud with CircleCI pipeline

Automating ECS deployment with CircleCI:

This helps to integrate and deploy our code to ECS

This process helps it more often, integrating and deploying small pieces of code constantly and as soon as possible.

Before creating this pipeline we need to create Microservice using ECS on AWS then using the pipeline we will deploy the code to cluster on AWS.

Firstly, we need to create a VPC with two public and private subnets in same availability zone. The main idea behind this is to keep our data in private subnet and expose them through the API will be deployed in the public Subnet using ECS

VPC helps to increase the security of our systems and we can create several VPC in a company for different departments. We need to specify the IP range address in VPC

Now we will create subnets inside VPC. All the instances present in public subnet can be accessed from Internet through internet gateway. Whereas private subnet is not open to internet. All the subnets inside VPC can communicate between them by creating a route table. Here we give access to private subnet from public subnet through NAT gateway.

By default subnets are private. To open to internet we need to setup internet gateway.

Setting up the RDS:

Now to save our data on system we will create a Postgres db on AWS.

Best practice is to keep our database in private subnet. Now database is setup in 2private subnets in 2availability zones we have created.

Why do we need two availability zone?

Because the DB is going to run in one AZ and the other AZ is used by AWS to save logs and backup for redundancy.

To keep DB secure we need to create a SG to limit traffic to DB and allow access from only public subnet

Now the Database, VPC and security groups are created

Now we will create a ECS cluster to execute our API using containers.

Best practice is to use Fargate because its easy to run containers without managing servers or cluster config.

Now we can create ECR where we will save our images and deploy it on cluster

Why we need ALB to run containers?

Because there will be more than one task running at the same time. In case during deployment process we will run both old and new containers simultaneously for a short periode of time This ALB will help to distribute traffic among containers using Listner, Listening rules and Target groups.

Why we need to setup security group to ALB?

Because if we are going to open our service to external traffic. We can improve our security by setting up a specify ports and protocols in our security group.

We have setup infrastructure on AWS with ECS stack, VPC, ALB and security groups.

Now we will create a pipeline to deploy cloud on AWS ECS. I will be using Circle CI tool.

Steps involved in this pipeline:

i. Build/Test the code
ii. push docker image
iii. Push the docker image to ECR we have created earlier on AWS
iv. Deploy it on ECS by updating task definition with the new image version

There are several environment variables we need to set up:

- AWS_ECR_Account_URL
- AWS_Access_key_ID
- AWS_secret_Access_Key
- AWS_region

With both CircleCi and AWS ECR configured we can start building images and pushing them to repo using docker file

AWS Certificate Manager: Its provisions and manages AWS SSL/TLS certificates for use with AWS services and internal resources. secure network communications and establish the identity of websites over the Internet as well as resources on private networks. deploy it on ACM-integrated AWS resources, such as Elastic Load Balancers, Amazon CloudFront distributions, and APIs on API Gateway. ACM automatically renews this certificates.

Fargate vs EC2:

It depends on the requirement of the company and applications managed. For this test challenge application I prefer fargate over EC2 . Fargate allocates the right amount of compute, eliminating the need to choose instances and scale cluster capacity. You only pay for the resources required to run your containers, so there is no over-provisioning and paying for additional servers.