CARRENTAL APP DOCUMENTATION

Cloud Engineer Internship

by: Mark Murithi Kibara

Learn to Cloud

Remote
Work from anywhere

Deploying A Dynamic Carrental-App

Introduction

The main objective was to orchestrate a comprehensive and modernized deployment pipeline for a carrental web application, incorporating key DevOps and cloud engineering practices. The journey began with the establishment of a robust version control system using Git, followed by the containerization of the application using Docker. Subsequently, cloud infrastructure was provisioned on AWS, featuring a Linux-based virtual machine instance accessible via SSH and equipped with a web server. The introduction of Infrastructure as Code (IaC) with Terraform allowed for the programmable and consistent deployment of cloud resources. I then integrated a Continuous Integration/Continuous Deployment (CI/CD) pipeline using GitHub Actions, streamlining development workflows. And finally, addressing Site Reliability Engineering (SRE) principles, emphasizing monitoring, alerting, and incident response procedures to ensure the reliability and availability of the deployed application. Using AWS Fargate simplified the deployment of containers, automating the management of infrastructure resources and allowing for seamless scalability, cost efficiency, and enhanced operational agility in the deployment pipeline.

Version Control Integration

Version control System: Github
Repository URL: https://github.com/Topsideboss2/carrental-project
Language: Javascript, TypeScript
Database: PostgreSQL
Cache: Redis
Steps and Implementation:

- Created a git repository for the web application on github.
- Initialized a repository on my local machine.
- Sync local repo and remote repository.

Challenges Faced:

- None

Containerization with Docker

Environment Setup: Docker & Docker-compose file.
Steps and Implementation:

- Created a dockerfile for each of the microservice applications. i.e Web, API, Redis, PostgreSQL
- Created a docker-compose file to build and run containers simultaneously.
- Check proper volume, network, security and accessibility of containers.

Challenges Faced:

- None

Cloud Infrastructure (Cloud Engineering)

Cloud Platform: AWS
Account: IAM user with Admin privileges
Method: AWS Management Console
Steps and Implementation:

- Created an instance on AWS using the management console.
- Security group allowed port 80 and 22.
- User data was a bash script that was used to install:
  - Nginx Web Server
  - Docker
  - Certbot
  - PostgreSQL Client
- Create AMI from this instance.

File Content:

```
#!/usr/bin/env bash
# Bash script that installs nginx, certbot, postgres client, docker and docker-compose
sudo apt update
sudo apt upgrade -y
sudo apt-get install nginx -y
sudo apt install python3-certbot-nginx -y
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl \
    gnupg-agent \
    software-properties-common
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
sudo echo -e '\n' | add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
sudo apt install -y docker-compose
sudo curl -fsSL https://www.postgresql.org/media/keys/ACCC4CF8.asc|sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/postgresql.gpg
```

```
sudo echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" |sudo tee
/etc/apt/sources.list.d/pgdg.list
sudo apt update
sudo apt install -y postgresql-client-13 libpq-dev
```

Challenges Faced:

- -

Setup Cloud Storage and Networking & Infrastructure as Code (IaC) (DevOps and Cloud Engineering)

Cloud Platform: AWS
IaC Tool: Terraform
Repository URL: https://github.com/Topsideboss2/carrental-project/tree/main/iac
Language: HCL
Method: AWS CLI
Components: Virtual Machine Instance, VPC, Subnets, Security Groups, Secrets Manager, Amazon Certificate Manager, ALB, ECS, NAT-Gateway, RDS, Route 53, S3
Steps and Implementation:

- Created a directory in our git repository known as iac for the terraform configuration and sync with remote repository.
- Created S3 bucket to store Terraform State
- Create DynamoDB Table to lock the Terraform state
- Create Secrets in Amazon Secrets Manager
- Register Domain name in Route53

Challenges Faced:

- Integrating ECS with Terraform was quite technical.
- Configuring PostgreSQL to run on RDS

CI/CD Pipeline with Cloud Integration (DevOps and Cloud Engineering)

CI/CD Pipeline Tool: GitHub Actions
Repository URL: https://github.com/Topsideboss2/carrental-project/tree/main/.github/workflows
Language: YML
Steps and Implementation:

- Created a directory in our git repository known as .github/workflows to store deployment scripts and synced with remote repository
- Created GitHub Actions Secrets to store sensitive env variables
- The pipeline includes steps to:
  - I. Configure AWS credentials.
  - II. Build the AWS infrastructure using Terraform
  - III. Create an ECR repository
  - IV. Start a self-hosted EC2 runner using the AMI created above
  - V. Build and push the docker images to ECR
  - VI. Create an env file and export to an AWS S3 bucket
  - VII. Stop self-hosted EC2 runner
  - VIII. Create a new task definition revision
  - IX. Restart ECS Fargate Service

Challenges Faced:

- Re-architecting the docker-compose file into a task definition file
- Re-architecting Redis to run as an ECS container.
- Configuring containers to use PostgreSQL RDS
- I came to the realization that creating only one ECS task definition family for the entire application prevents copies of each application scaling independently. As a best practice, I'd use a separate task definition family for each of these pieces of containerized code.

Site Reliability Engineering (SRE)

Monitoring Tool: AWS Cloudwatch
Components: Alarms, Incidence Response Plan
Steps and Implementation:

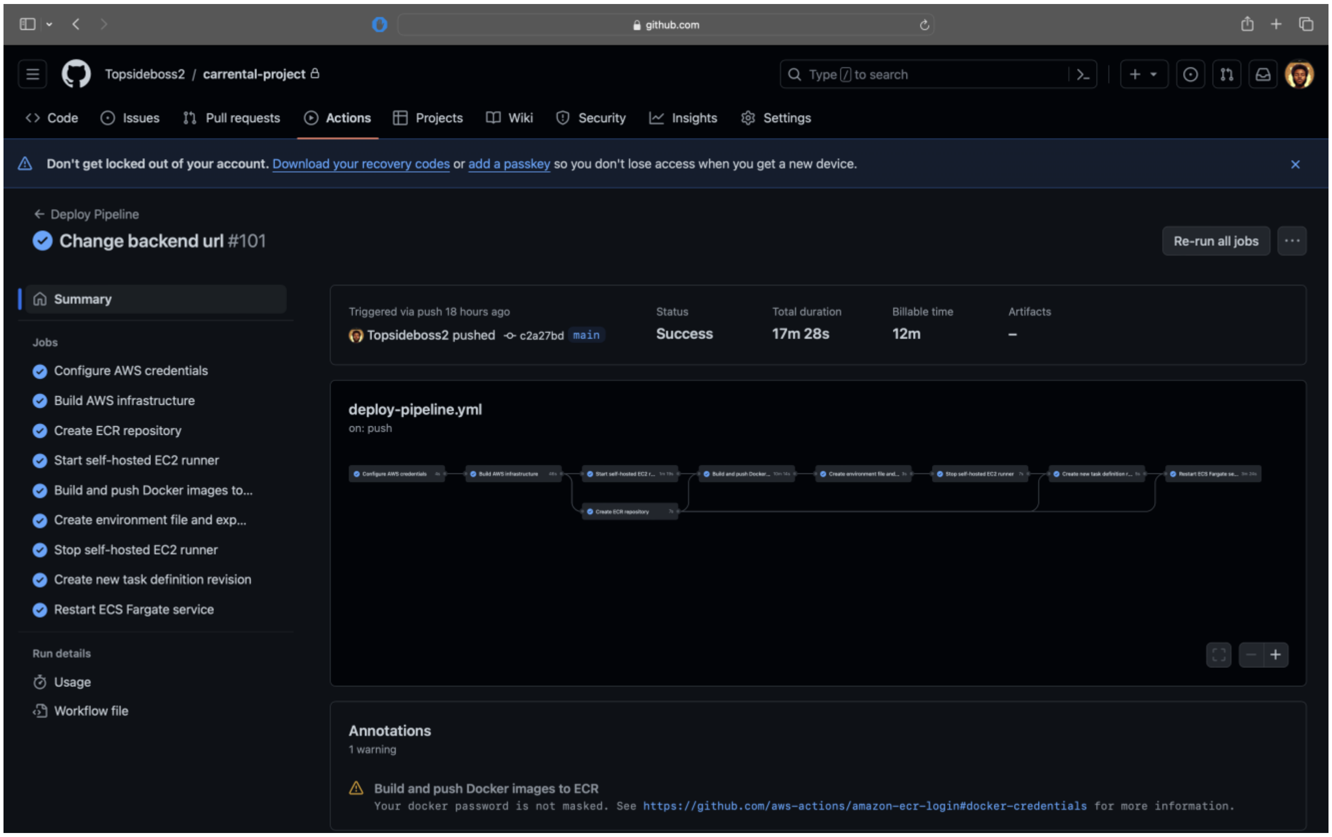
- Setup Monitoring with AWS CloudWatch
- Configure Alerts for Critical Metrics
- Implement Incidence response procedures

Challenges Faced:

- Limited to only AWS monitoring tools.
- Unable to use Grafana or Prometheus on AWS ECS Fargate because there isn't a common underlying infrastructure for all of the containers.

OUTPUTS:

1. Successful Github Actions Pipeline



github.com

Topsideboss2 / carrental-project

Don't get locked out of your account. Download your recovery codes or add a passkey so you don't lose access when you get a new device.

← Deploy Pipeline

Change backend url #101

Summary

Jobs

- Configure AWS credentials
- Build AWS infrastructure
- Create ECR repository
- Start self-hosted EC2 runner
- Build and push Docker images to...
- Create environment file and exp...
- Stop self-hosted EC2 runner
- Create new task definition revision
- Restart ECS Fargate service

Run details

- Usage
- Workflow file

Triggered via push 18 hours ago	Status	Total duration	Billable time	Artifacts
Topsideboss2 pushed -o- c2a27bd main	Success	17m 28s	12m	–

deploy-pipeline.yml

on: push

Configure AWS credentials 4s
Build AWS infrastructure 48s
Start self-hosted EC2 r... 1m 19s
Create ECR repository 7s
Build and push Docker... 10m 14s
Create environment file and... 3s
Stop self-hosted EC2 runner 7s
Create new task definition r... 5s
Restart ECS Fargate se... 3m 24s

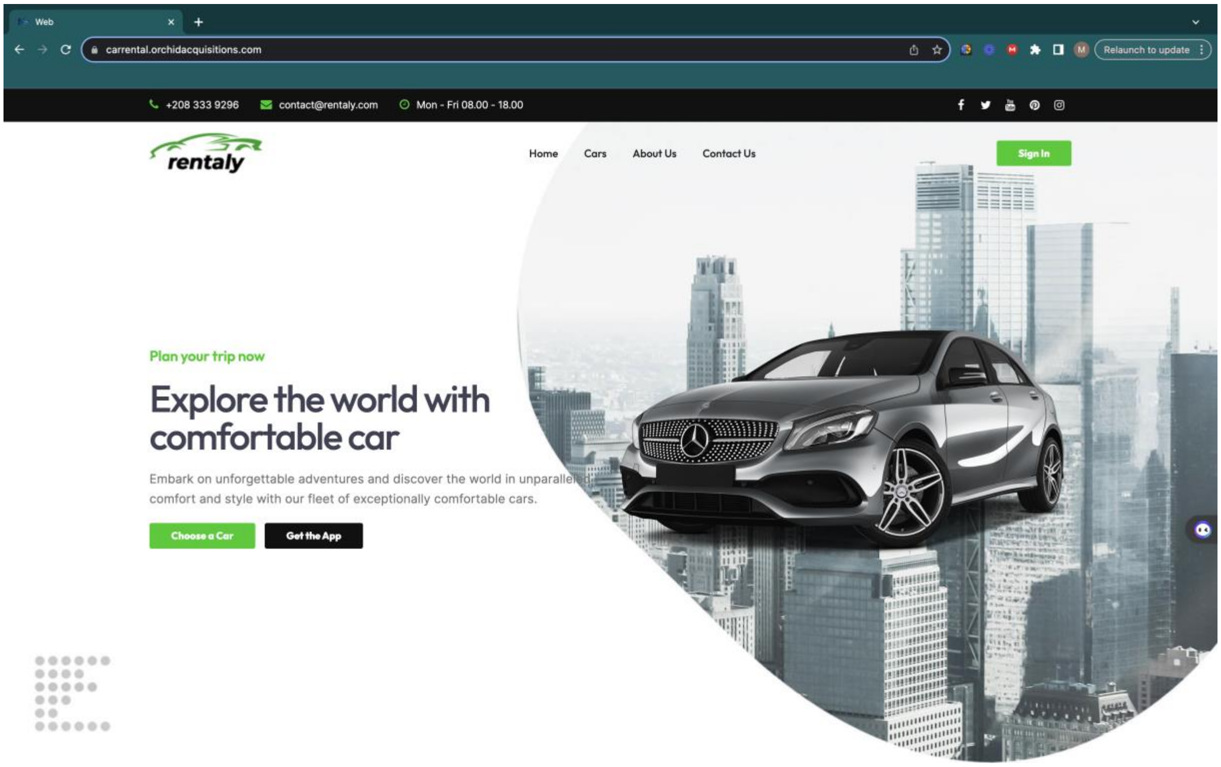
Annotations

1 warning

Build and push Docker images to ECR

```
Your docker password is not masked. See https://github.com/aws-actions/amazon-ecr-login#docker-credentials for more information.
```

2. Running Application



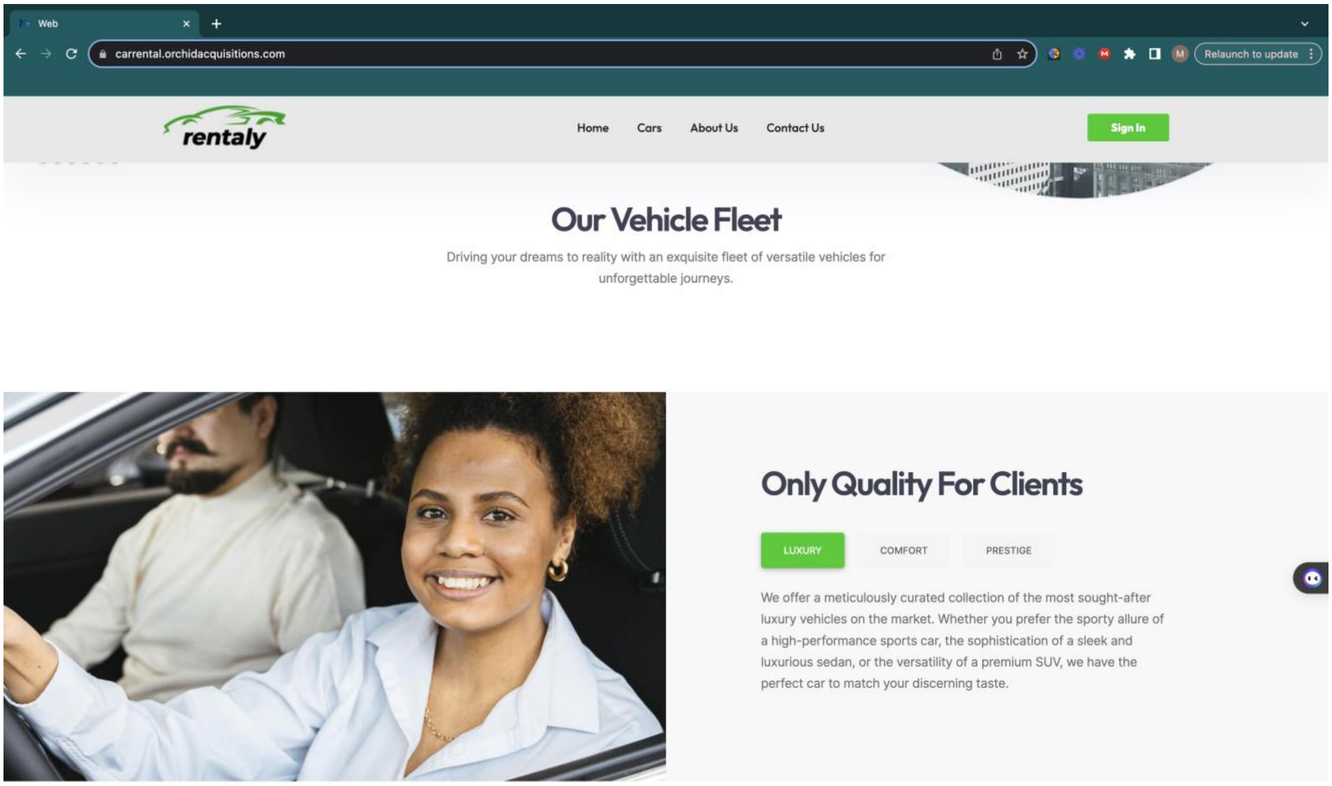
carrental.orchidacquisitions.com

Plan your trip now

Explore the world with comfortable car

Embark on unforgettable adventures and discover the world in unparalleled comfort and style with our fleet of exceptionally comfortable cars.

https://carrental.orchidacquisitions.com

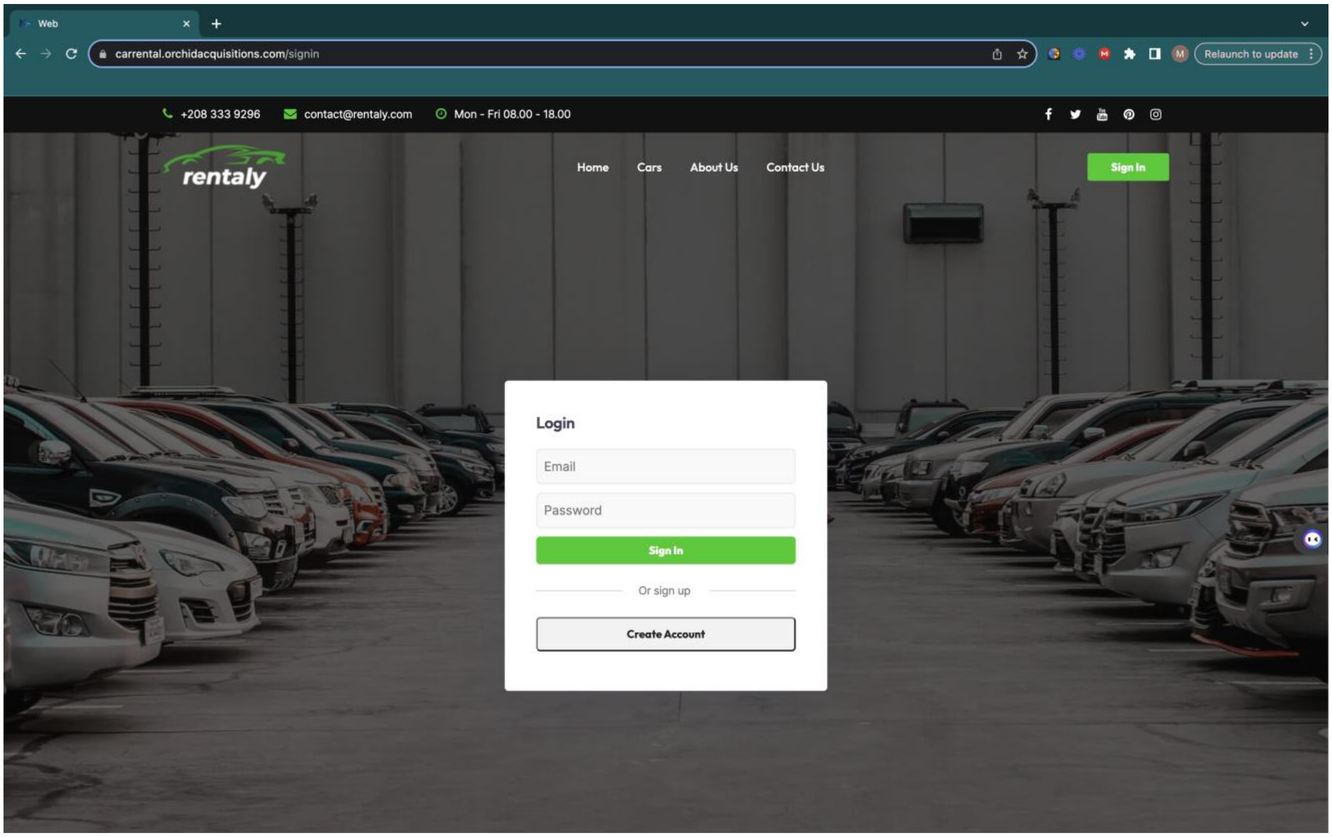


Our Vehicle Fleet

Driving your dreams to reality with an exquisite fleet of versatile vehicles for unforgettable journeys.

Only Quality For Clients

We offer a meticulously curated collection of the most sought-after luxury vehicles on the market. Whether you prefer the sporty allure of a high-performance sports car, the sophistication of a sleek and luxurious sedan, or the versatility of a premium SUV, we have the perfect car to match your discerning taste.



Login

Email

Password

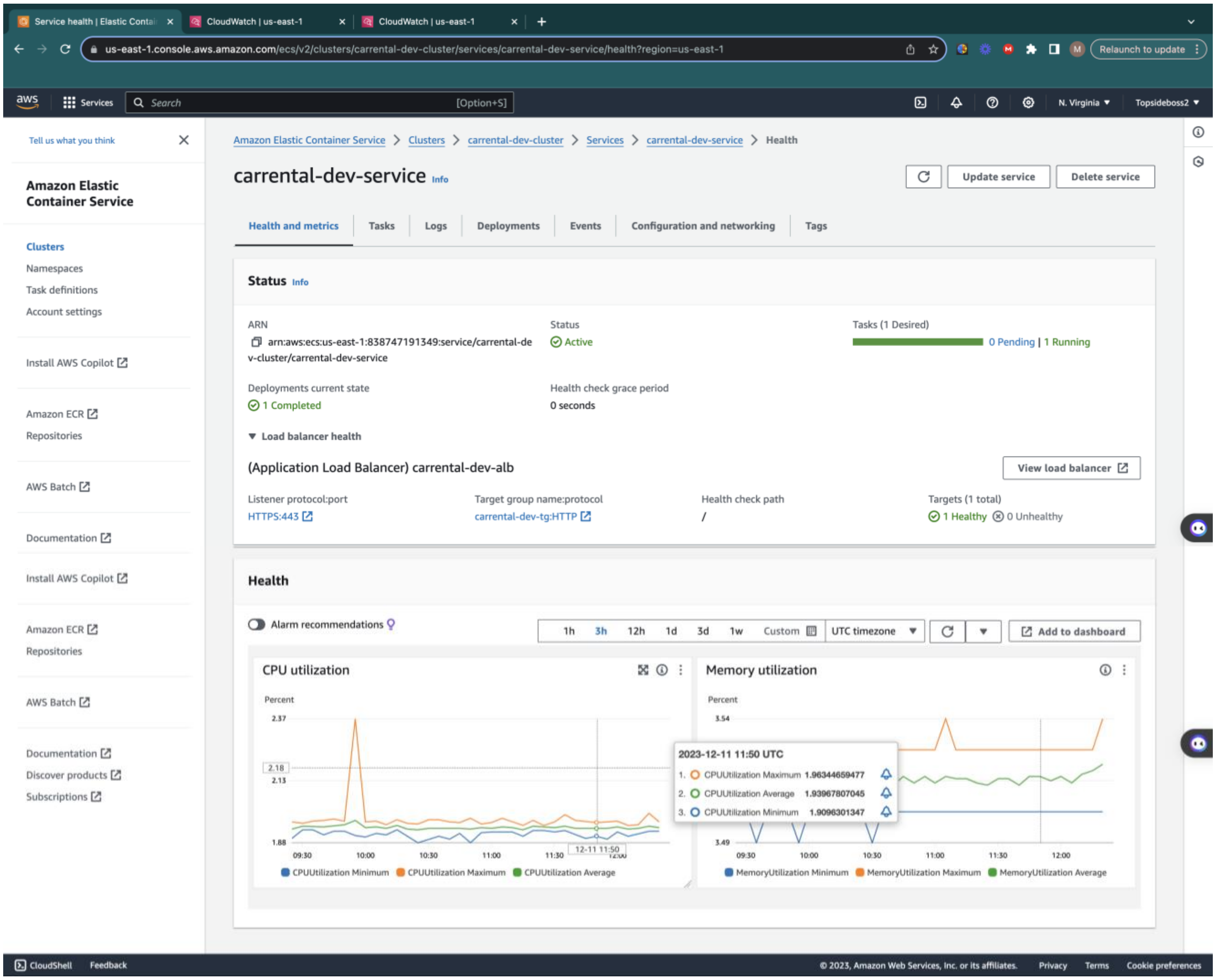
Sign In

Or sign up

Create Account

https://carrental.orchidacquisitions.com

3. AWS ECS Carrental Service Metrics



Service health | Elastic Contai × CloudWatch | us-east-1 × CloudWatch | us-east-1 × +

us-east-1.console.aws.amazon.com/ecs/v2/clusters/carrental-dev-cluster/services/carrental-dev-service/health?region=us-east-1

Relaunch to update

aws Services Search [Option+S] N. Virginia ▼ Topsideboss2 ▼

Tell us what you think ×

Amazon Elastic Container Service

- Clusters
- Namespaces
- Task definitions
- Account settings
- Install AWS Copilot
- Amazon ECR
- Repositories
- AWS Batch
- Documentation
- Install AWS Copilot
- Amazon ECR
- Repositories
- AWS Batch
- Documentation
- Discover products
- Subscriptions

Amazon Elastic Container Service > Clusters > carrental-dev-cluster > Services > carrental-dev-service > Health

carrental-dev-service Info

Update service | Delete service

Health and metrics | Tasks | Logs | Deployments | Events | Configuration and networking | Tags

Status Info

ARN
arn:aws:ecs:us-east-1:838747191349:service/carrental-dev-cluster/carrental-dev-service

Status
Active

Tasks (1 Desired)
0 Pending | 1 Running

Deployments current state
1 Completed

Health check grace period
0 seconds

▼ Load balancer health

(Application Load Balancer) carrental-dev-alb

View load balancer

Listener protocol:port
HTTPS:443

Target group name:protocol
carrental-dev-tg:HTTP

Health check path
/

Targets (1 total)
1 Healthy ⊗ 0 Unhealthy

Health

Alarm recommendations

1h 3h 12h 1d 3d 1w Custom | UTC timezone ▼ | Add to dashboard

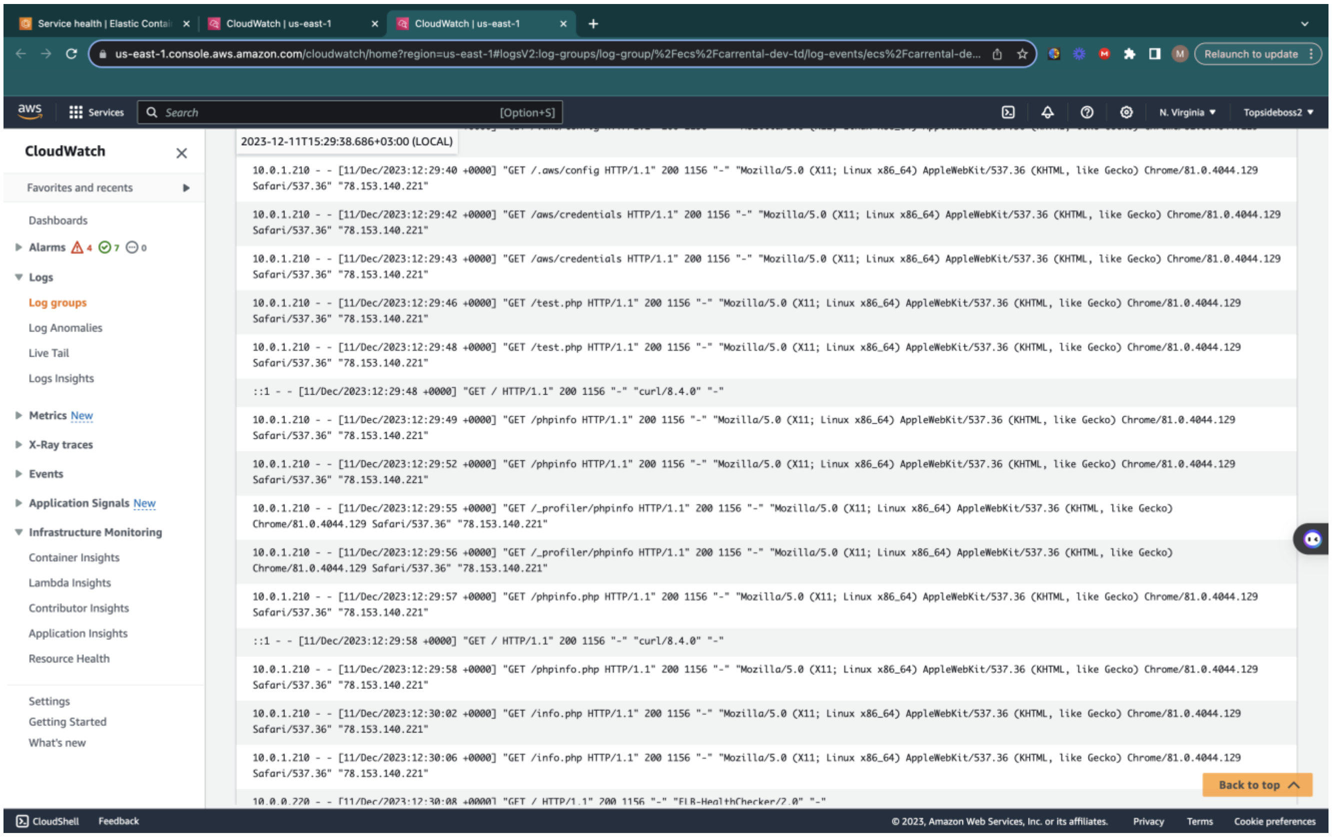
CPU utilization

Memory utilization

2023-12-11 11:50 UTC
1. CPUUtilization Maximum 1.96344659477
2. CPUUtilization Average 1.93967807045
3. CPUUtilization Minimum 1.9096301347

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

AWS CloudWatch Container Logs



2023-12-11T15:29:38.686+03:00 (LOCAL)

```
10.0.1.210 - - [11/Dec/2023:12:29:40 +0000] "GET /.aws/config HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:29:42 +0000] "GET /aws/credentials HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:29:43 +0000] "GET /aws/credentials HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:29:46 +0000] "GET /test.php HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:29:48 +0000] "GET /test.php HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

::1 - - [11/Dec/2023:12:29:48 +0000] "GET / HTTP/1.1" 200 1156 "-" "curl/8.4.0" "-"

10.0.1.210 - - [11/Dec/2023:12:29:49 +0000] "GET /phpinfo HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:29:52 +0000] "GET /phpinfo HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:29:55 +0000] "GET /_profiler/phpinfo HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/81.0.4044.129 Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:29:56 +0000] "GET /_profiler/phpinfo HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/81.0.4044.129 Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:29:57 +0000] "GET /phpinfo.php HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

::1 - - [11/Dec/2023:12:29:58 +0000] "GET / HTTP/1.1" 200 1156 "-" "curl/8.4.0" "-"

10.0.1.210 - - [11/Dec/2023:12:29:58 +0000] "GET /phpinfo.php HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:30:02 +0000] "GET /info.php HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

10.0.1.210 - - [11/Dec/2023:12:30:06 +0000] "GET /info.php HTTP/1.1" 200 1156 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129
Safari/537.36" "78.153.140.221"

10.0.0.220 - - [11/Dec/2023:12:30:08 +0000] "GET / HTTP/1.1" 200 1156 "-" "ELB-HealthChecker/2.0" "-"
```