David Rex

DevOps Engineer

Contact

Address
Colorado Springs, United States, 80909

Phone
719-310-2858

E-mail
drex@idevops.io

Skills

Solutions deployment

Program testing software

Testing and maintenance

Code reviews

DPSM

Dedicated and persistent history of meeting company goals. Skilled in working under pressure and consistently working until a job is completed or in it's needed state.

Experienced in designing, building, documenting, supporting, automating, and optimizing infrastructure and deployments in AWS, Azure, Kubernetes, Terraform and more.

Education

2007-08 - 2011-08

High School Diploma

Golden Eagle High - Eagle Rock, MO

2022-05 - Current

DevOps Engineer

IDevOps.io - Golden, CO

Certifications

20221	CKA: Certified Kubernetes Administrator
20222	HashiCorp Certified: Terraform Associate (002)

Work History

2016-03 - 2022-05

Food and Beverage Service Technician

Denver Beverage, Denver, CO

- Maintained effective supply levels by monitoring and reordering food stock and dry goods.
- Developed team communications and information for meetings.
- Used critical thinking to break down problems, evaluate solutions and make decisions.
- Worked flexible hours across night, weekend and holiday shifts.
- Actively listened to customers' requests, confirming full understanding before addressing concerns.

DevOps Tasks Performed

•

- Created bash script to audit Kubernetes pods for liveness probes and report which pods in which namespace had probes in their spec.
- Created bash script to audit Kubernetes pods for readiness probes and report which pods in which namespace had probes in their spec.
- Created bash script to audit Kubernetes pods for resources requests and report which pods in which namespace had requests in their spec.
- Created bash script to audit Kubernetes pods for resources limits and report which pods in which namespace had limits in their spec.
- Created python script to produce audit report of Kubernetes pod report of if liveness and readiness probes are enabled.
- Created python script to produce audit report of Kubernetes pod report of if resource requests and limit are enabled.
- Created Jenkins Dockerfile from Scratch.
- Setup simple Hello-world pipeline in Jenkins to test Docker deployment of Jenkins.
- Created Redis Dockerfile from Scratch.
- Setup Redis test script in python to test Docker deployment of Redis.
- Wrote terraform to deploy s3 buckets in AWS with lifecycle options.
- Wrote terraform to build and deploy virtual machine for nginx server in AWS.
- Wrote terraform to setup IAM role and policy for reading and writing from s3 bucket.
- Created an AWS SQS Queue using Terraform
- Installed Redis Via Helm and Tested it worked through Kubernetes port forwarding.
- Wrote python script to test password protected Redis utilizing Kubernetes port forwarding through local host.
- Created and modified values.yaml file to change Redis helm chart service type from ClusterIP to LoadBalancer in order to test through public internet.
- Wrote python script to test password protected Redis utilizing Kubernetes load balancer.

- Wrote Terraform to deploy AWS VPC with 3 subnets: 1 public, 2 private, with a layer networking setup routing traffic through internet gateway to setup segregated networking layer for application stack.
- Wrote Terraform to deploy pay_per_request DynamoDB table for test application in AWS.
- Wrote Terraform to deploy custom ECR's on demand for test application docker images in AWS.
- Built a django based we site that allows users to login sign up and post thoughts like twitter.
- Built docker image for Django site.
- Built Kubernetes deployment manifest for Django site and Docker image then deployed into Kubernetes cluster.
- Created MYSQL sidecar for django project on local network database with persistent storage.
- Created separate deployment with persistent data of MYSQL to reference using K8S DNS in Django project.
- Created new pipeline steps to deploy MYSQL for above Django project.
- Setup Django to use Redis for session caching in load balanced Django services using K8S DNS to reference Redis.
- Created deployment and service for Redis and added steps to pipeline for Django project so it will deploy Redis to utilize cache.
- Troubleshoot why MYSQL wasn't starting correctly, and resolved issue with deployment manifest referencing wrong port.
- Created Django unit tests for Django app to have 100% code coverage
- Add functionality to CI/CD pipeline to deploy Django app that starts ephemeral test infrastructure using docker in the git action pipeline to start MYSQL and Redis for the Django unit tests to utilize before building Docker image.
- Added ZAProxy Scan to Django deployment pipeline to test OWASP vulnerabilities and fail pipeline on failed scans.
- Added automatic Route53 domain name updating in our CI/CD pipeline to deploy Route53 DNS for our

Django project

- Added ingress controller your our Kubernetes deploy for Django project
- Added automatic SSL generation using cert-manager to our ingress for Django project to serve site on https endpoint.
- Implement regression/UE testing for Django site functionality
- Troubleshooting errors in pipeline, ephemeral environments, and UE regression testing to make full CI pipeline work.
- Created a script to pull a password for use in a pipeline using bash. Using passwords.idevops.io and saved to variable.
- Created a bash script to template replace variables in config files using environment variables.
- Created AWS command to list out all AWS resources using AWS CLI and JQ
- Delete AWS cloud 9 resource using AWS CLI (environment cleanup)
- Created bash script that dynamically creates Terraform to deploy AWS ec2 instances with prompt based on inputs.
- Created bash script to monitor disk usage and alert when over 75%
- Created file terraform to deploy ec2 instance with security group that allows port 80 and 22
- Create ansible-playbook that configures ec2 instance to run python api by installing python, cloning repository and setting up nginx reverse proxy to server gunicorn on port 80.
- Deployed and validated python api application with ansible and terraform.
- Add load balancer and target group to ec2 instance to can access via elb load balancer with terraform.
- Add route53 block in terraform to automatically assign domain ec2 load balancer which is your name.