Shan e Raza

Site Reliability/DevOps Engineer at TicketManager

03025022209 @ shan.e.raza@outlook.com Rawalpindi, Pakistan

Profile

An experienced systems/platform architect, Cloud operations engineer, Linux systems engineer, and hands-on leader. Significant experience in extremely large scale deployments, automation, metrics, pro-active monitoring, and information security, Advocates, and practice's infrastructure as code, eliciting useful application metrics, building fault-tolerance systems.

Specialties: Building sustainable server and serverless utilities, design and developing automated scripts for pro-active monitoring of distributed applications, OS.

Strengths & Skills

Monitoring and Logging

- ✔ Graphite
- ✔ ELK
- ✔ Logstash
- ✔ Elasticsearch
- ✔ Excellent Knowledge about File Systems
- ✔ Process Monitoring
- ✔ Collectd
- ✔ Incident Response
- ✔ Grafana
- ✔ Riemann
- ✔ Kibana
- ✔ InfluxDB
- ✔ Telegraf
- ✔ Zabbix
- ✔ Kapictor
- ✔ Chronograf
- ✔ Prometheus
- ✔ Nagios
- ✔ Automation

Networking

- ✔ SSL
- ✔ SSH
- ✔ Networking
- ✔ TLS
- ✔ Setting up caching server (Squid/Varnish/etc)
- ✔ Setting up Reverse Proxy (Nginx/etc)
- ✔ Setting up load balancer (Nginx/AWS/etc)
- ✔ Setting up firewall
- ✔ OSI Model TCP/IP/UDP
- ✔ SCP
- ✔ HTTPS

Automation

- ✔ Selenium
- ✔ SFTP
- ✔ AWS Step functions and State machine
- ✔ Ansible
- ✔ Chef
- ✔ AWS Cloud Formation
- ✔ Terraform
- ✔ ProtractorJS

Love for Terminal

- ✔ Network
- ✔ Vim/Nano
- ✔ Text manipulation
- ✔ Bash scripts
- ✔ System performance
- ✔ Compiling app from source (gcc/make/etc)

Database

- ✔ SPROC
- ✔ SQL
- ✔ Triggers
- ✔ Mysql
- ✔ Mongo DB

Production Web App

- ✔ Micro-Services based App Architecture
- ✔ Deployments
- ✔ Disaster Recovery Plan
- ✔ Backups Management

Cloud Providers

- ✔ Google Cloud Platform
- ✔ Azure
- ✔ AWS
- ✔ Digital Ocean

Cluster Management

- ✔ EKS
- ✔ ECS
- ✔ Docker Swarm
- ✔ Kubernetes

Runtimes/Languages

- ✔ NodeJs
- ✔ Python
- ✔ Clojure
- ✔ Java

Distributed Systems Platform

- ✔ Service-Level Indicators (SLI)
- ✔ Service-Level Objectives (SLO)
- ✔ Kafka

CICD

- ✔ ECR
- ✔ Docker
- ✔ Jenkins

Web Servers

- ✔ Apache
- ✔ Nginx
- ✔ IIS

Operating Systems

- ✔ Linux
- ✔ UNIX

Other

- ✔ Jira configuration/Developer

Education

BS in Software Engineering Foundation University , 2016 BS in Software Engineering Foundation University Rawalpindi, 2016 Rawalpindi

BS in Software Engineering Foundation University Rawalpindi, 2016

Certifications

HashiCorp Certified: Terraform Associate

May 2021 - May 2023

HashiCorp
https://www.credly.com/badges/c91aaf32-065b-4a28-9aed-18602b8d2eab/public_url

Site Reliability Engineering: Measuring and Managing Reliability

Starting October 2020

Google Cloud
https://www.coursera.org/account/accomplishments/certificate/NHS5C3TPVGVA

Experience (5 Years)

DevOps Consultant

February 2020 - Present

Scorpbit Technologies — Rawalpindi, Pakistan

Working on numerious projects with multi-cultural envoirment as Cloud Architect and DevOps Engineer.

Site Reliability/DevOps Engineer

July 2017 - Present

TicketManager — Islamabad, Pakistan

TicketManager Is an sports company, a SaaS, and they offer to orchestrate your sports tickets, Create loyalty programs, Make events easy, and provides reporting on ROI.

SDET/Automation Engineer

March 2015 - June 2017

Tipping Points Pvt Ltd. — Rawalpindi, Pakistan

Tipping Points offer a Social Streaming Platform Application to there huge numbers of users, Where users interact and contribute to various social events.

Projects

App Test and Delivery Automation Framework

Tipping Points Pvt Ltd.

Tools: Java, Jenkins, Dockers, Appium, Bahs scripting

At Tippings Points, We have various streaming Applications, for different regions, My responsibility was to design a framework, which will be used to Automate Test, deployment, package, and delivery to the app and play store.

API Test Automation and Continuous Integration to Production Running services

Tipping Points Pvt Ltd.

Tools: Jest, Jenkins, Dockers, Logstash, Elasticsearch, JS

At Tippings Points, I Automated the dev-test life-cycle of our Streaming app's mainly there API's using Jest (JS-based framework) i-e upon successful testing of our application, a script which I designed and written will package the API services at containers and tests the integration with our clone of running production system i-e Pre-Production, Upon full compliance, Script will merge and deliver of our packaged API's to the Production account using build pipelines as part of the workflow!

SFTP Data Intake Automation

TicketManager

Tools: S3, ECS, Lambda, Mysql, SPROC, Cloud-watch event rules, Cloud-watch logs, IAM, Build Scripts.

At TicketManager, we have highly reputed customer base, customers as Automobiles companies, communication companies, and Banking, etc. Our customer base gets increasing and within a customer company, their user data changes are fast-paced, thus we offered our customers an SFTP service to provide there profile files at schedules, which will get processed within seconds as soon as file drops at our server. For this I have designed and developed our DATA intake Automation from our FTP server, as soon our customer gets registered we onboard their user base in our customer instance in at our Application platform (TicketManager). Architecture designed by us have ARN rule monitoring the activity over our SFTP machine at ECS cluster, as soon as file gets dropped, cloud-watch event rule gets triggered by alarm, and in turns triggers our serverless Lambda function, which first tests the file authority/type/authenticity/scope, and provide our customer and stakeholder a live feedback of file current status(processing/accepted/operational/rejected), if file is accepted, Our Serverless function process it and calls upon our RDS of MYSQL to run a SPROC to update records at our Database, after successful execution system reports of final status of processed file with full details over email, and over webhooks to our internal channel, It is fun working with **serverless**!!

Monitoring health and Logging metrics

TicketManager

Tools: Riemann, Graphite, Grafana, collectd, Docker collectd plugin, Elasticsearch, Logstash, Kibana, StatsD, Prometheus

At this project, an individual, my goal was to ensure **Pro-Active** Monitoring health and logging metrics for our containerized application `TicketManager`, I have achieved this goal of centralized log management by segregating the process into the following parts which came to the resolved state in sequence;

1. **Monitoring Architecture** – What metrics we want to track in running Docker containers?
2. **Collecting a Metric Data** – Used Prometheus as a light-weight polling server running inside a Docker container and which saves the data/metrics into a time-series database – and its integration into the rest of the application.
3. **Types of Metrics** which were collected for processing:
    1. **Runtime Metrics** – Statistics collected by OS and Application Host e.g (memory usage, CPU load, and a number of web server requests), this gave us a view on how hard our containers are working.
    2. **Application Metrics** – Custom statistics relevant to our application, How many time a particular type of event has been processed, or how many API calls we are making, this empowers us with a more detailed view of what application was doing.
    3. **Container runtime metrics** – metrics which come from docker platform itself. (e.g Number of containers running? A number of containers health checks?) this elicited such information which might be hidden in a higher level aggregated metrics.
4. **Building Dashboard** – Plugging everything together into a dashboard which gave us a single view of our Application Health, like what the application is doing, what the containers are doing, and what Docker is doing to manage the containers.

DRP

TicketManager

Tools: Terraform, AWS, Shell scripting, Python, Js, AWS CLI

I **lead** our successful effort for **DRP**(Disaster Recovery Planning) by working with our Architecture and Support Team, I followed the idea of **Infrastructure as code** by performing **dependency inversion** of our 300+ **integrations** based Application Architecture and provided a production running DR env within the deadline of 2 months, i-e now our system is of monitoring and alter has already been designed by our team in way **Pro-Active manner**, that I with working with our Arch team and CTO, decided **SLI's** and **SLO's** of our product, and have cloud-watch event rules configured, that if series of incidents start happening at our **multi-services** based Application, our switch for terraforming gets triggered and within downtime to minimum we have our active DR region up and running, and will be a complete support as we were providing at our main production account, and ALB's will keep the users traffic to this region until our support team found and fix reasons's of incidents happening at production account, all reporting module was built during this Project to let our tech and stake-holders team know of the latest **insights**.

Industries

- Information Technology