One of the largest distributors for mobile consumer brands in the UAE and Saudi Arabia

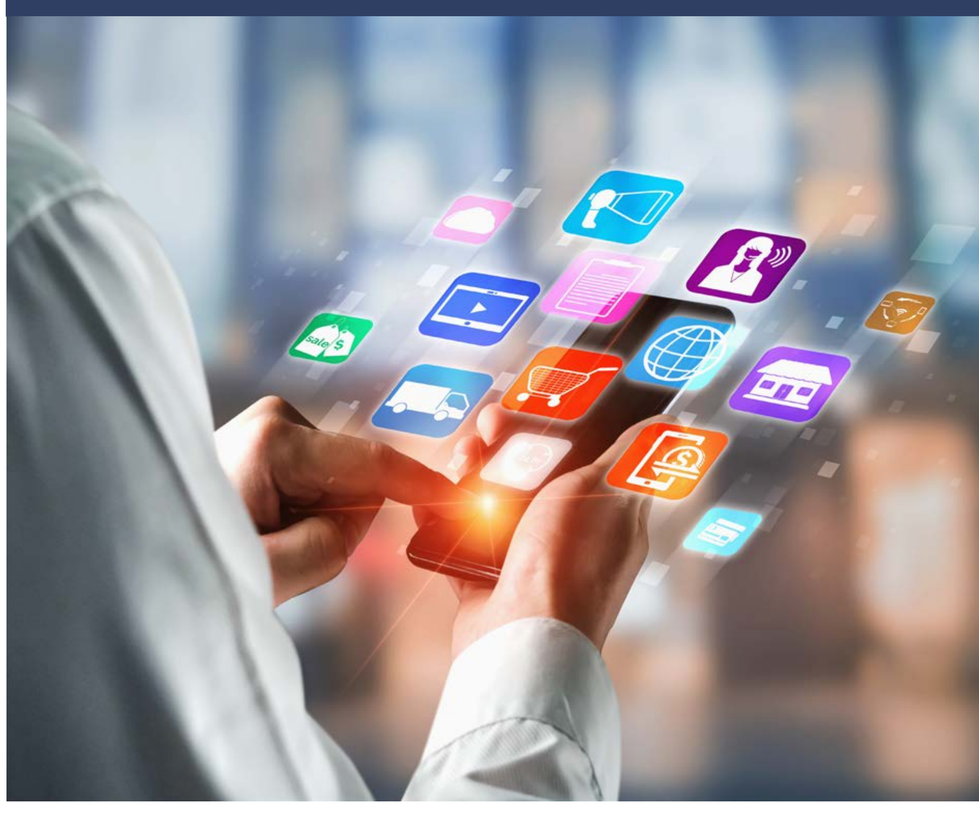
Engagement Overview

Axiom engaged TO THE NEW to build an end to end Next Generation Distribution Platform solution across various distribution business supply chains in order to increase the customer satisfaction and improve the efficiency.

The Client

Axiom Telecom is the largest distributor & retailer of some of the world's most recognizable Telecom Products brands in the Middle East. Founded in 1997, Axiom operations range from wholesale, retail, and value-added services to after-sales care of Telecom Products. Axiom comprehensively holds 65% of the Market share in UAE & KSA.

Industry: Ecommerce



Business Challenge

Axiom Telecom wanted to develop a Data Platform that complements the Axiom and Hyke Distribution platforms to streamline the operating model by agile capabilities and culture that drives new revenue streams to Axiom group.

- An Infrastructure as a code that can help to create an End-to-End infrastructure Automation and is flexible enough to integrate with different tools and cloud providers.
- The provisioning of IT infrastructure should be fast and simple with minimal efforts.
- Same Infrastructure code should be able to serve different environments.
- There should be configuration consistency with minimal human error involved.
- If the host machine crashes then infrastructure states shouldn't be lost.
- Ability to deploy ad-hoc environments on the fly with easy disposal.
- Manually changed resources should be overridden when Infrastructure code configuration is redeployed and this shouldn't change the existing infrastructure.
- Same configuration language should be easily adaptable on multiple cloud platforms.
- Deployment should be handled easily with the same configuration language.
- There should be minimum risk and human dependency for provisioning an environment.

Solution Overview

- Terraform manages different environments through terraform workspace. TTN selected and created a new terraform workspace and configured the environment specific parameters in terraform code for respective environments.
- TTN ran a “terraform plan” to verify the resources that were created by Terraform. Once verified the terraform code was applied to provision the infrastructure on AWS.
- Retrieved infra-credentials from AWS secret-manager as per environment and used it to provision and configure the resources.
- Created the VPC in AWS along with various network components like subnets, routes, nat gateways, etc.
- Provisioned the resources like EC2, RDS, S3, EKS cluster, Lambda, etc using various terraform AWS modules.
- Used ansible with the help of “local-exec” provisioner provided by Terraform to further perform various installations and configurations in EC2, RDS, EKS and other resources.
- Installed various tools like Kong, kube2iam, fluent-bit, apm, istio service-mesh, spot rescheduler, spot termination handler, etc. in EKS cluster through helm via ansible playbooks using terraform local-exec provisioner.
- Terraform output provided the endpoints to be used by the end user.
- All terraform states were configured to be stored on the S3 backend.
- A terraform lock was used using AWS DynamoDB service to ensure the terraform code was applied once at a time and hence preventing the corruption of terraform state files.

Business Benefits

The infrastructure as a code automated the provisioning and configuring of IT infrastructure in about less than an hour with minimal human intervention required. This reduced the time and human errors during provisioning of the infrastructure. The maintenance of terraform code using version control helps in tracking the changes and enables us to rollback to a previous version easily whenever required.

- Workspaces in terraform have enabled the same code to be deployed in dev, staging and production based on the locals for each environment.
- Terraform state files have been stored in the remote S3 Backend.
- Terraform applied helped to deploy an ad hoc environment and destroy it instantly when not needed.
- Terraform handled and maintained the state of our infrastructure and when some resources were found to be deviating from the state files.
- Terraform has multiple providers for AWS, Azure, Kubernetes, Helm etc and supports multiple platforms through that.
- Terraform lock has been configured using aws dynamodb service to prevent terraform to be applied once at a time.
- Ansible playbooks are used to configure the resources provisioned by Terraform using the “local-exec” provisioner provided by Terraform.
- Dynamic ansible playbooks, ansible hosts file and other files are created dynamically using the “local_file” resource of Terraform
- Using the Helm provider given by Terraform, Kubernetes API objects are deployed right after the creation of the infrastructure.

AWS Services used

We used various AWS services as below:

- AWS Route53

- AWS Secrets Manager
- AWS Elastic Kubernetes Service (EKS)
- AWS Elastic Cloud Compute (EC2)
- AWS Elastic Load Balancing
- AWS Aurora - MySQL (RDS)
- AWS ElastiCache Service
- AWS Simple Storage Service (S3)
- AWS Cloudfront
- AWS DynamoDB
- AWS Lambda
- AWS Cloudwatch
- AWS Transit Gateway

Know more about our offerings

www.tothenew.com

Talk to Our Experts

Connect with us