Summary

Nykaa engaged TO THE NEW to manage and overhaul their infrastructure and build processes so as to ensure better usage and ownership of the teams. TO THE NEW setup entire CI/CD pipeline, speeding up the deployment and improving the build quality. This engagement helped Nykaa to bring down the monthly AWS spend from $135K to $70K within a short span of four months.

The Client

Founded in 2012, Nykaa is a premier online beauty store in India selling fashion products to more than half a million customers across 900 cities. The products are 100 percent genuine and authentic, sourced directly from the brands.

The Goal

Nykaa had an extremely complex setup on AWS running 200+ servers. The entire infrastructure was managed by different development team members without having any standard deployment processes or best practices in place. The prevailing infrastructural assets were provisioned, configured and managed manually leading to lot of delays and was highly error prone. They were looking for an AWS expert who could help them standardize deployment & monitoring processes, setup CI/CD pipeline automation for faster integration and deployment along with 24/7 monitoring support.

Highlights

- Reduced AWS spend from $135K to $70K within a short span of four months
- Provided 24/7 monitoring & stringent SLAs to ensure 99.99% availability
- Implemented one click rollback and deployment with zero downtime
Key Features

- Improved the deployment procedures and reduced the time to deploy
- Infrastructure control and provisioning was done following industry standards and best practices
- Automated complete delivery pipeline using Jenkins to enable the team to promote code from DEV to UAT to PRODUCTION within few minutes
- Containerized applications using Docker & Amazon Elastic Container Service (ECS) to ensure that all environments have exactly same setup and can be easily replicated whenever required
- Leveraged auto scaling and spot instances to launch the application at a very low cost but at the same time ensured readiness to scale with an increase in traffic
- Reduced time to provision and deploy new services from 90 minutes to 10 minutes
- Audited & fixed gaps identified against AWS best security practices, OS hardening & operational checklist

Technical Excellence

- Moved the complete infrastructure from Amazon EC2 Classic to Amazon VPC. Separate Amazon VPCs were created for Production & Non-Production environments
- Implemented Blue-Green deployment strategy with zero downtime
- Implemented ELK for centralized logs monitoring & Chef for configurations management.
- Integrated Nagios & NewRelic for applications & infrastructure monitoring
- Created separate AWS Identity & Access Management (IAM) groups & users for tracking user actions
- Created AWS Identity & Access Management (IAM) roles for managing server permissions

Use of AWS Services

- Used Amazon EC2 to increase compute capacity in cloud
- Used Amazon VPC to launch AWS resources in a virtual network
- Used Amazon Elastic Container Service for quickly managing the containers
- Used Amazon RDS to setup, operate and scale a relational database
- Used Amazon S3 & Amazon Glacier for online file storage and data archiving
- Implemented Amazon ElastiCache (Redis) to deploy, operate, & scale an in-memory data store or cache in the cloud.
- Used Amazon Simple Notification Service (SNS) to coordinate and manage the delivery or sending of messages to subscribing endpoints or clients
- Leveraged Amazon Simple Email Service (SES) to increase the effectiveness of email marketing in cloud
- Implemented Amazon Classic Load Balancer (ELB) to automatically distribute incoming application traffic across multiple targets
- Leveraged Amazon Application Load Balancer (ALB) to ensure content-based routing
- Used Amazon Route 53, a scalable domain name system (DNS) service to direct end users to applications
- Used Amazon Elastic Block Store (EBS) to store persistent data
- Used AWS Identity & Access Management (IAM) for securely controlling access to AWS services
- Used Amazon CloudWatch to collect information, track metrics & monitor infrastructure on AWS cloud constantly
## Technology Stack

### Frontend
- React
- Varnish
- CDN

### Backend
- Node
- Redis
- Solr
- Apache Tomcat
- Django

### Database
- MongoDB
- MySQL

### Cloud and DevOps
- EFK
- CHEF
- Jenkins
- Docker
- Nagios
- Spotinst
- PagerDuty
- New Relic
- Pingdom
- Captrio

### AWS
- VPC
- Amazon EC2
- Amazon S3
- RDS
- ElastiCache
- Load Balancer