clearscale

aws partner network

Premier Consulting Partner

Boingo's Wi-Fi Sponsorship and Advertising Platform AWS Migration Powered by DevOps Under the Hood

boingo

Executive Summary

Boingo Wireless, Inc. (NASDAQ: WIFI) helps the world stay connected. Their vast footprint of small cell networks covers more than a million DAS and Wi-Fi locations and reaches more than 1 billion consumers annually in places as varied as airports, stadiums, universities, and military bases. For more information about the Boingo story, visit boingo.com.

Boingo's Wi-Fi ad network delivers a unique way for mobile marketers to engage with consumers on the go. Marketers are guaranteed full engagement with their customers through offering free Wi-Fi service. Boingo Media sponsorship campaigns offer several screens of exclusive, high-impact interactions for their customers' brand. Boingo has optimized its networks and promotional offers for the smartphone audience, supporting all major mobile device platforms.

The Challenge

Over a period of 12 months, Boingo Wireless experienced a massive expansion of their international footprint. Boingo Media needed to expand their infrastructure capabilities to provide scalability and increased service levels of their Wi-Fi Sponsorship & Advertising platform to meet these growing demands. Boingo needed infrastructure that would be elastic, scalable, and more cost effective than their existing deployment at RackSpace. Boingo Media had previously acquired C9, who had migrated a portion of their infrastructure to AWS. This basic deployment gave Boingo Media insight into the AWS value proposition. Boingo Media needed an experienced partner that could migrate their application to an infrastructure designed to take full advantage of AWS services and technology.

> "ClearScale focused on the heavy-lifting and applied best practices to ensure that our systems scaled with our growing business. They implemented automation and provided responsive support after our engagement had completed. We were spared the problem of splitting finite resources and this allowed us to focus on running our business and get to the cloud at the same time."
>
> **Brian Shields**, Director of Software Engineering, Web & Advertising, Boingo Wireless

The ClearScale Solution

Boingo Wireless partnered with ClearScale to optimize their architecture for AWS, automate based on DevOps practices, and migrate their entire platform from RackSpace to AWS. ClearScale worked closely with Boingo Wireless to develop an in depth understanding of their business and technical requirements. ClearScale performed a full audit on the Rackspace environment and designed a solution fitting AWS best practices. Boingo Wireless agreed on a complete cut-over to the new infrastructure versus a phased migration. Data synchronization was implemented between the existing deployments in RackSpace and AWS to the new environment to minimize downtime during the cut-over.

Architecture

Designing architecture for scalability, availability, resiliency, and security is no easy task. It is a balancing act between complex technology and simplification of design. AWS delivers complex features and functionality in their services while providing ease of use and management. ClearScale designed a robust and highly scalable architecture for Boingo's specific requirements. AWS CloudWatch service is used to track performance and utilization on all AWS components to ensure that the solution is meeting specified requirements.

The production Virtual Private Cloud (VPC) spans three AWS Availability Zones (AZs). Three subnets exist in each AZ: Public Subnet, Private Application Subnet, and Private Data Subnet. This provides Boingo Wireless with flexibility and redundancy in the AWS region.

All external traffic is directed to AWS Elastic Load Balancers (ELBs), the mobile connection manager, or HAProxy in the Public Subnet. The ELBs route whitelisted traffic to Boingo application(s) running on AWS Elastic Compute (EC2) instances. The mobile connection manager application is used to track and manage mobile application downloads and writes data back to AWS ElastiCache (Redis) for reporting. HAProxy handles requests that are not whitelisted. The mobile connection manager EC2 instances and HAProxy EC2 instances are deployed across multiple AZs for load distribution and high availability (HA). Only infrastructure components that handle direct Internet connections are placed in the Public Subnet to limit security exposure.

Components in the public subnet communicate with Boingo application(s), RabbitMQ, and Celery in the Private Application Subnet that does not allow any direct connections from the Internet. The Boingo application(s) manage campaigns, reporting, and analytics. Boingo application EC2 instances are setup as an Auto Scaling Group across all AZs. This allows for automatic scaling of the Boingo application depending on traffic load. Celery is used to run jobs and aggregate all event data that is written to ElastiCache (Memcache). RabbitMQ and Celery are setup as application clusters across multiple AZs for redundancy and availability.

ElastiCache(Memcache) service, ElastiCache(Redis) service, MySQL RDS, and MongoDB are deployed in the Private Data Subnet. The MySQL RDS implementation supports the CMS system and is deployed across all three AZs. MongoDB, deployed on EC2 instances, is the primary database for all applications. Data in MongoDB is split into three shards across three replica sets, each in a different AZ. Components in the Private Data Subnet have been optimized for redundancy, performance, and scalability.

Security groups are configured for each application based on their unique access and communication requirements. Security groups also follow strict naming conventions to accommodate DevOps automation practices. This has proven to be an effective approach, balancing granularity of control with manageability and automation. AWS CloudTrail service is used to record every AWS operation performed in the environment for audit logging and compliance.

Automation

DevOps practices are a huge factor to the overall scalability of an environment. Automation and managing infrastructure as code are key components of these practices. AWS CloudFormation service was used to automate infrastructure deployment. Chef was chosen as the configuration management tool for this infrastructure as Chef has proven itself as a platform to create versioned, repeatable configurations. With CloudFormation and Chef it takes a few clicks to provision a new infrastructure block or migrate/redeploy an existing one. Chef is also a key to scalability of the solution, providing means to deploy new instances quickly and efficiently. Every instance deployed is fully automated through Chef recipes and cookbooks.

CloudFormation templates were segregated into four functional areas:

- *VPC and Base Components Template* — where networking and security components are defined
- *Applications Template* — where all application instances and load balancers are defined
- *MongoDB Template* — where MongoDB clustering components are defined
- *RDS and ElastiCache Template* — where MySQL RDS and caching layers are defined

Hosted Chef Configuration Management

What is Chef?

Chef is a powerful automation platform that transforms complex infrastructure into code. Chef automates how applications are configured, deployed, and managed across the network. Chef is built around simple concepts: achieving desired state, centralized modeling of IT infrastructure, and resource primitives that serve as building blocks. Hosted Chef is the SaaS offering of this powerful platform.

Numerous Chef recipes and cookbooks were produced to automate configuration of the servers and applications. Some were shared cookbooks for common tasks, such as updating Route53 DNS records and managing EBS storage volumes. Individual cookbooks were developed for the following applications: HAProxy, Mobile Connection Manager, Boingo Application, RabbitMQ, Celery, and the MongoDB cluster.

Here is an overview of one of the many Chef cookbooks (comprised of several recipes) developed for Boingo Wireless. This particular cookbook automates the deployment of MongoDB clusters. This is an in-depth look at how it is done.

Recipe for adding MongoDB repository:

- Installs MongoDB packages
- Installs MongoDB ruby gem

Recipe to setup MongoDB as configuration server for the cluster:

- Creates (or restores) EBS volume, attach, format, and mount
- Create (or update) DNS record for configuration server in Route53
- Configure MongoDB as configuration server for the cluster

Recipe to setup MongoDB as data server in the cluster:

- Creates (or restores) EBS volume, attach, format, and mount
- Configure MongoDB as data server for the cluster
- Configure replica set
- Connect to configuration servers and add configured replica set as a shard

Recipe to create backup of cluster:

- Stops Balancer process in MongoDB cluster
- Creates snapshots from EBS volume one of configuration server (and creates tags on this snapshot)
- Creates snapshots from EBS volume each of shard servers (and creates tags on this snapshots)
- Starts Balancer process in MongoDB cluster

Recipe to schedule daily backups:

- Creates cronjob for daily backup, if this server is first configuration server

Automation is a huge factor when it comes to scalability and efficiency of cloud infrastructure operations. Ultimately, successful automation practices equate to better availability and resiliency of the infrastructure and services. There is a winning combination when you combine the capability to deploy AWS infrastructure through CloudFormation and the power to configure the services with Chef.

A Successful Migration Methodology

ClearScale used a repeatable migration methodology developed over years of experience. An infrastructure audit was performed and, based on Boingo's requirements, the architecture was redesigned with AWS components in place. The infrastructure was converted to code and full automation of infrastructure and server configuration was developed. CloudFormation and Chef automation was used to deploy the staging environment. As the staging environment was tested, all changes required were integrated back into the automation tools. When the staging environment received a stamp of approval, the same automation was used to deploy the production environment. Data replication was setup from Rackspace to AWS in order to keep MySQL and Mongo databases in sync. A "mock" cutover was performed to further reduce migration risks. Every component and process (with exception of propagating DNS changes) had been tested iteratively before the actual migration took place. This iterative testing practice is extremely effective in delivering successful migrations.

The Benefits

Boingo Wireless is able to meet the growing needs of their customers on demand with the flexibility and power of the AWS Cloud. With automation and auto scaling, engineers are able to focus more effort on developing new products and features and less on operations and capacity build-outs. Boingo Wireless is achieving the operational efficiencies provided by the AWS Cloud, bringing down their total cost of ownership of infrastructure. More importantly, they have increased the availability and robustness of services provided to their customers.