DevOps Solution to Effectively Manage IT Infrastructure

About the Client

This customer is a leading Badge printing and technology service provider from New Jersey, USA. They are also experts in developing attendance tracking and lead retrieval systems. Along with onsite installation facilities they provide consultation, quality assurance and extended support to all their products.

> *I want to mention how much we enjoyed InApp's help as our application transitioned to AWS. We look forward to working with InApp in the future.*
>
> — Vice President

Business Challenge

- The customer's original system used two standalone servers hosted on Rackspace.
- Manual intervention was required for every code change and deployment which eventually delayed the release.
- This setup was limited to very minimal changes during a given time frame.
- A long-term scalable solution was necessary to effectively manage their IT infrastructure.
- Application downtime was mandatory to make any changes.

InApp's solution

InApp proposed a plan to migrate the standalone servers and it's functionalities to an AWS based architecture which enables continuous integration & deployment. To accumulate all the code changes from different collaborators we used Github as a version control software. The functionalities were created as separate auto-scaling instances which could be accessed through custom APIs. With the help of Jenkins & MSBuild Continuous Integration was introduced and this enabled the developers to make frequent code changes to the system. Independent load balancers were associated with each auto-scaling groups which help in distributing the incoming load. AWS CodeDeploy was used for Continuous Deployment which picked up the build file from S3 and deployed them to the required instances automatically.

The idea behind the DevOps approach was to make the application more scalable and capable of withstanding frequent code changes and enhancements. The auto-scaling instances allowed the application to be available on-demand for the customer and prevented system downtime during deployment.

Highlights

Technology:

The applications and functionalities were built as EC2 instances in independent auto scaling group. If any instances were stopped the auto scaling feature created a new instance thereby maintaining the application stability.

Automation:

Jenkins was used to pull in any latest code changes from the repository which would then automatically create a build file with the help of MSBuild. After all the necessary approvals, using AWS CodeDeploy, the final build file was then triggered to be deployed in the respective instances. In case of any issues, the rollback steps were triggered automatically.

Storage:

Amazon S3 containers were used to store the intermediate build file before deployment. During a release, the CodeDeploy service automatically picked up the build file from the S3 container and deployed it to the existing instances as well as the newly created instances due to the auto-scaling feature.

Security:

Access to the Virtual Machines were restricted using Security Group Policies. Applications such as AWS Inspector was used to assess any incoming threats to the system. Amazon GuardDuty, Inspector and WAF were also used to monitor the traffic for any malicious attempts to access the system.

Monitoring & Alerting:

AWS CloudWatch was employed to monitor the application infrastructure. There were options to check the resource utilization such as CPU, Network parameters, Disk, etc periodically which helped in managing the auto scalable architecture. Notifications could also be configured to monitor the billing progress based on current usage.

Business Benefit

- A scalable architecture that requires zero downtime during application deployment.
- Each functionality built as separate instances which can be accessed through custom APIs.
- An automated environment which makes the code build and deployment faster than the conventional methods.
- Advanced security layers to monitor and alert in case of any malicious attempts to the system.
- Cloud storage for convenient access of build files during deployment.
- Improved deployment frequencies which could accumulate frequent code changes and enhancements.
- More efficient monitoring and alerting mechanism to keep track of all the changes happening to the system.