info@apriorit.com
www.apriorit.com

Case study

SaaS Growth and CI/CD Process Support with Smart AWS Infrastructure

Our client is a US-based SaaS vendor that provides a cloud platform for gathering, consolidating, analyzing, and presenting data received from user devices.

The platform collects information from monitored mobile devices and desktops using installed agents and provides users with web-based dashboards to track geolocations, online activities, sent and received data, and other details.

Apriorit was hired to enhance our client's existing solution by gradually improving its performance while extending its functionality and scalability.

Starting point

At the start of this project, our client had a SaaS solution with 30,000 active users and a back end hosted on 12 Windows servers run by a US-based third-party provider.

From the technical point of view, the purpose of the platform is to gather unstructured data from multiple sources, process that data according to special rules, and then show the results to the end user. Part of this data comes from Android devices via pull requests from an agent service. Other data comes from the web via push requests made through a custom C++ library, which encapsulates a private protocol.

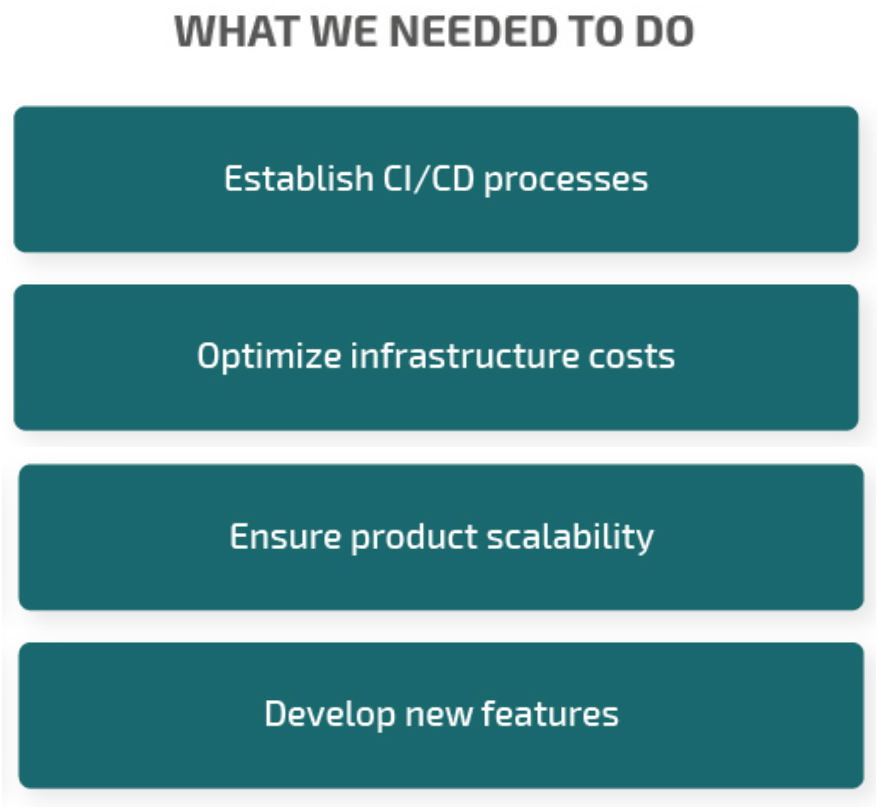
When we joined the project, the back end was a web service written in C# and managed by Internet Information Services (IIS) on Windows Server 2008 R2. The front end was built with PHP (Laravel) and a custom JavaScript framework. Some user data was stored in a MySQL database, while other data was stored as SQLite files in the server file system. Configuration data was partially stored in a MySQL database and partially distributed among web services.

The challenge

The project was rapidly increasing its number of active users, so our client wanted to increase the scalability and performance of their SaaS solution. They also had ambitious plans for the project's development but found it challenging to make improvements while providing uninterrupted service for the platform's end users.

The scope of work

To resolve these challenges, the Apriorit team was assigned four main tasks. The first task was to establish the right DevOps processes and structure to support the client's ambitious development plans and create continuous integration / continuous development (CI/CD) processes. The second task was to accommodate the growing number of users by providing an adequate architecture and infrastructure for the improved platform. The third task was to optimize infrastructure costs. The last (and the trickiest) task was to accomplish all three previous tasks in parallel with the development of new platform features.



Our solution

Apriorit created a dedicated team for this project that consisted of a business analyst, a project manager, four developers, a database engineer, a DevOps engineer, and two QA specialists. During the project, this team closely cooperated with a DBA specialist and several PHP developers from the client's side. To complete all tasks set by the client, we used the following team and technologies:

APRIORIT TEAM	KEY TECHNOLOGIES USED	
• Business analyst	• AWS EC2	• Django
• Project manager	• Docker	• RabbitMQ
• Database engineer	• Atlassian	• ECS
• Software developers	• MySQL	• CentOS
• DevOps engineers	• Python	
• QA specialists	• DynamoDB	

We started platform improvements by establishing CI/CD processes. After that, we moved to improving the product's scalability with AWS and optimizing infrastructure costs. The shift to the new platform architecture allowed us to quickly deploy newly developed features. Let's see how we did it step by step.

Establishing CI/CD processes

We built two additional environment levels: dev (for development testing) and staging (for QA testing before deployment to production). To automate the build and deployment processes, we used the Bamboo tool by Atlassian.

At first, the platform and this new scheme worked well with the existing hosting provider. But at some point, as the user base increased, it became hard for the client to manage the system's growth using the mechanism provided by their local hosting service provider. The Apriorit team analyzed the pros and cons of several solutions and proposed building a SaaS solution based on Amazon Web Services (AWS).

Migrating to AWS and optimizing costs

Once our client approved the plan to deploy the SaaS solution on AWS web hosting, we got to work.

Through the end of the migration process, our client was expecting significant growth in the number of platform users. This would potentially lead to an increase in the number of deployed servers as well as the associated support costs. We estimated that the new platform would require 28 Windows servers instead of the 12 Windows servers that the client was currently using. Given this, it was obvious that the new AWS-based scheme required cost optimizations right from the start.

We decided to shift from Windows servers to Linux servers to both increase performance and lower costs. To implement this server migration, our developers ported the custom C++ library, encapsulating the main data exchange protocol from Windows to Linux (CentOS). As a result, we managed to decrease the number of servers to only six.

The shift to six Linux servers also allowed us to provide a simple mechanism for onboarding new servers, enabling smooth system growth in the future. In addition, we rewrote the backend web service and implemented worker environments using the Django Python framework.

We developed a new system based on:

1. DynamoDB (to store all kinds of data)
2. MySQL on RDS (mostly to serve configs and logs)
3. ECS with two types of clusters (web services and workers)
4. An EC2 instance to host RabbitMQ

At first, we considered using SQS as a straightforward task queueing solution. But Amazon has some issues with long-running tasks on their workers, which is why we decided to use Celery. However, we then discovered that Celery has issues with the push model of serving queues imposed by the Simple Queue Service. So in the end, we decided to go with RabbitMQ.

In the new setup, all code is sitting in a Docker container, and shipping to the AWS environment is done using CodeBuild. Our team also used Docker Compose to set up a local development environment (with MySQL, Rabbit, Dynamo, etc.) which allowed us to optimize maintenance costs for the AWS-based SaaS platform.

The impact

Apriorit's first achievement was rebuilding the development and deployment environment and optimizing the corresponding processes. Besides the DevOps engineering on AWS, our team introduced Jira for task management and Bitbucket for code hosting. All these changes allowed us to ensure continuous deployment with AWS and release several features and improvements every week without having to worry about how the system would handle further growth.

The project still relies on Bamboo for CI/CD processes with AWS infrastructure, but we're thinking about trying Jenkins, which is faster in some cases and a bit more agile than Bamboo.

PROJECT RESULTS	
Established CI/CD	Successful release of several features and improvements every week
High system scalability	The number of active system users increased from 30,000 to 100,000, who perform over one million requests per day
Optimized infrastructure costs	40% savings compared with the project starting point and more than 300% compared with the projected cost of the old environment accounting for growth

The results in terms of cost savings have been impressive. We cut the platform maintenance costs **by about 40%** compared with the project starting point and by **more than 300%** compared with the costs of the old environment after accounting for estimated growth. We achieved this with a smart environment setup that uses fewer and less expensive resources: instead of an estimated 28 Windows Server 2008 R2 servers, we now use only six CentOS servers. We also replaced expensive file system storage with unified DynamoDB storage.

Our client's system now serves more than 100,000 active users who perform over one million requests per day — and this system meets all of the client's performance and UX requirements.

Get in touch with us to improve your SaaS solution and take your project to another level!