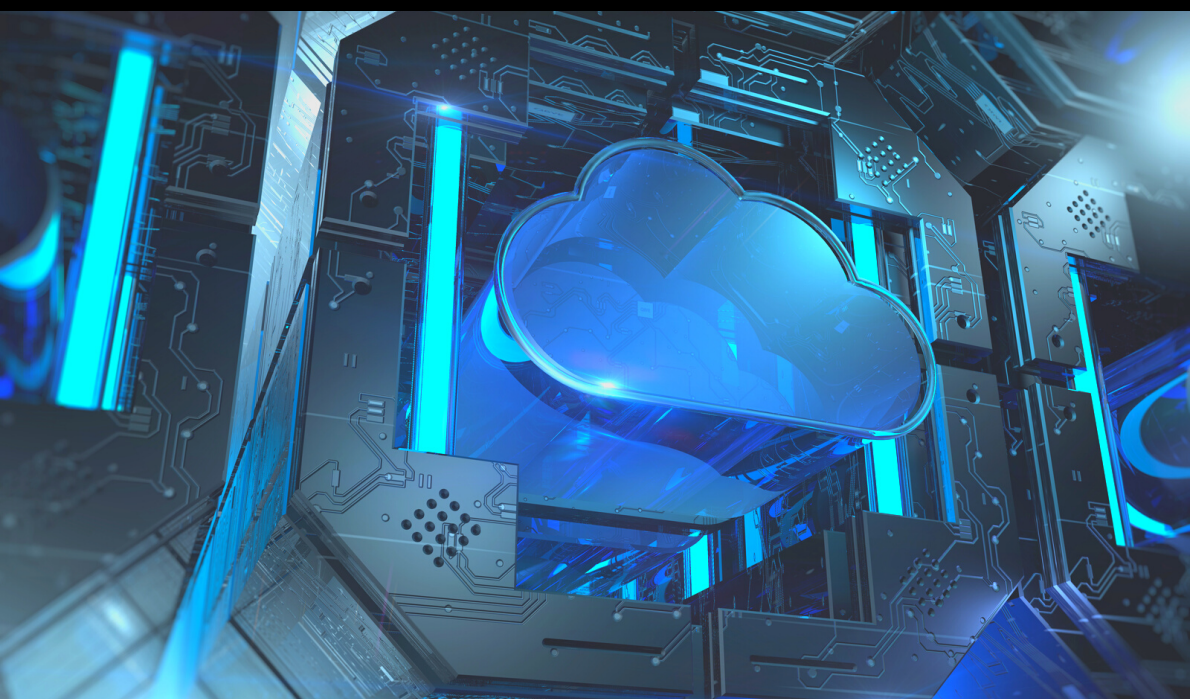
SOFTWAREMILL

Migrating SMS Gateway service to the AWS Cloud

CASE STUDY



CLIENT PROFILE

Intelli Messaging is a Tier 1 SMS Gateway Service provider in Australia, also offering Direct Carrier Connection routing to New Zealand. The service is designed and built for enterprise performance and reliability using geographically redundant systems and commercial carrier-grade technology. The solution lets users cost-effectively include mobile communication in their business processes for either marketing or operations.

PROJECT OVERVIEW

INDUSTRY
Telecom

GOAL OF THE PROJECT
To adopt the existing solution to the AWS cloud environment while maintaining continuous availability of the service.

CHALLENGE
A successful cloud migration and modernization of an SMS gateway system.

PROJECT DURATION
1 year

TECHNOLOGIES
AWS Cloud, Kubernetes, Helm, Terraform, MongoDB, MySQL, Prometheus, Scala, Java, Akka, Grafana, Graylog, Docker, Jenkins

BENEFITS

- Gaining agility and flexibility to quickly adapt and scale.
- Increased system resiliency and high availability due to built-in infrastructure redundancy provided by the AWS cloud.
- Easy on-demand system scalability provided by the combination of cloud-based infrastructure and the Kubernetes platform.
- Introduction of cost-effective resource usage.
- Simplified project maintenance resulting from passing the physical infrastructure management responsibility to the cloud provider.

BACKGROUND

Intelli Messaging SMS gateway is a carrier-grade messaging technology for the enterprise and application providers. The system is designed to process large volumes of SMS traffic and is able to accept client submissions via various entry points such as SMPP connections, REST API, or email. These submissions are converted into SMPP messages and routed via a set of configurable routes to the other SMS gateways. The system is also capable of receiving delivery receipts from those gateways and pushing notifications about them via different notification channels to the configured recipients. Another type of supported traffic includes Mobile Originated messages that can be received from the SMS gateways and pushed towards customer-defined endpoints.

For the past 10 years, SoftwareMill has been engaged in developing a new bulk-messaging gateway that met the client's expectations for messaging volume, throughput, and availability. This included software design and development, testing, deployment, support, and project management for the new gateway system.

In this business domain, high availability and resilience to failure are absolute must-haves. Failing to provide those may result in damaging the company's reputation and subsequently losing customers' trust. With the passing years, it had become clear that trying to provide a highly available (HA) and resilient system while running on a local data center was becoming a constantly increasing challenge, mostly due to the effort required to maintain physical infrastructure and because of the way the deployment process compatible with that infrastructure was designed. Thus, about a year ago, we were presented with a task to replace the on-premises technology with flexible, scalable, and cost-effective computing power in the cloud.

CHALLENGES

By the time the client started thinking about the cloud migration process, the Intelli Messaging system had been running for roughly 10 years on-premises. The old infrastructure was not flawless. This setup had various **low-level infrastructure issues** and **didn't allow to scale the system cost-effectively**.

System services were tied to specific machines based on the predicted resource usage, which meant they could not be easily spun up elsewhere in case of failure. These services were deployed as sets of Java jars, meaning they were not properly isolated when running on the same machine. The deployment process itself was cumbersome in case something went wrong, and rollbacks were done by manually modifying symbolic links to point them to the previous deployment directory. Deploying some of the critical services that are supposed to be "always" online was very stressful to anyone participating in the process.

The event store was running on a truly outdated version of MongoDB. Upgrading that instance was something everyone was extremely reluctant about, given the potential risk in case something went wrong.

In order to migrate the system to the cloud, it was necessary to identify the required cloud capabilities and limitations in the context of Intelli Messaging. On top of that, it was very important to address the topic of migrating the data from the data center to the cloud as well as minimizing the system downtime window. An additional set of requirements around security features based on VPN was added by some of the customers who were aware of the migration plan.

SOLUTION

Once the decision has been made, it was time to analyze the existing setup and explore the possibilities offered by the cloud. The primary acceptance goal was to have the system providing the same set of capabilities, but with **improved availability, resilience, scalability, and less maintenance overhead**. Internally, the development team has set up expectations on **reducing the complexity of the deployment process and also reducing the level of software maintenance effort**.

The first step was to research the available cloud solutions on the market. It was necessary to compare the pricing, features, and capabilities, as well as support options. An additional factor was that our client already had some other system running on AWS, which meant they were familiar with the service. And so, Amazon AWS was chosen.

Once the provider has been selected, it was time to start thinking about the best way of using available cloud services. **Amazon Virtual Private Cloud service** made it possible to create completely isolated environments for the production, staging, and internal tools, while the **EC2 service** provided the team with the capability to spin up and terminate as many instances as needed by the system. An additional benefit was that AWS takes care of the creation and maintenance of load balancing building blocks, providing such resources as **ALB and NLB load balancers** out of the box.

During this stage, it became clear that it was necessary to **automate the process of creating and managing** whatever infrastructure required by our system on top of the AWS offering. To that end, the **Terraform** tool was used to define infrastructure as a code for all environments that we needed. Combined with **Amazon DynamoDB and S3** to store the Terraform state, it became a very useful tool for maintaining our cloud environments.

SOLUTION

Since the system has a number of publicly available entry points that clients can access via DNS name, it was required to be able to expose verified certificates for these clients. A task that is much easier when you work with the **AWS Certificate Manager service**. Amazon Route 53 service was also used to configure mappings between specific DNS names and respective load balancers.

One of the big pains was the way the system was deployed on the old infrastructure. With the cloud providing the foundation for the system, choosing containers to provide an isolated environment for each service deployment seemed like a natural choice. **Docker** was an obvious candidate there. **Amazon Elastic Container Registry** was used as a repository for the service images. These images are created via Jenkins builds - and the **Jenkins** server itself is running on one of the EC2 instances.

As with any non-trivial system, you need to address the problem of resources required by its services. The solution had to be easily scalable and resilient to failures related to resource usage, which meant **increased service availability**.

First, the resource usage on the old infrastructure was measured. With that data, it was possible to define the minimum expectations in terms of computing power that was needed from the cloud. This translated into the choice of **EC2** instance types that the system would run on.

The next thing to take care of was the scaling capability. Doing it manually was out of the question. **Amazon Elastic Kubernetes Service** (EKS) was chosen to run the **Kubernetes clusters** providing the environment to deploy the services. With the Auto Scaling groups managed by the EKS, the system automatically gets the resources it needs.

SOLUTION

When resources are no longer needed, they are automatically released. Based on that foundation, it was possible to define the conditions under which specific services would be scaled to handle incoming traffic peaks.

One of the most important topics to consider was the migration of the data that the system generates. Data integrity after the migration was a priority requirement.
The system works with two databases:

- event store (MongoDB)
- and reporting database (MySQL).

It was decided that in both cases it is best to set up a replication process that will ensure data is up-to-date the moment the system in the cloud starts running.

On the AWS end, the **AWS RDS service** was chosen to maintain the reporting database. That provided a resilient database service, with automatic backup capability as well as a set of configurable AWS CloudWatch alerts that could be defined to let the team know when things go the wrong way.

An important characteristic of the system is that some of its services are expected to run without interruption for a very long period of time. Failing to ensure this specific capability could be viewed as a sign of poor quality by some of the customers. But the services were supposed to run within the ever-changing Kubernetes cluster environment, where your pod could be re-deployed anytime depending on the overall system resources needs. **Dedicated Kubernetes eviction policies** have been implemented in order to reduce the risk of this happening for these critical services. Thanks to that, the risk of such a service having an interruption is much lower than it would be otherwise.

SOLUTION

Because of the security requirements, it was necessary to configure various secure channels of accessing the system for the team and the system users, as well as setting up a secure connection for the data migration process. That was resolved by using the **AWS Transient Gateway, AWS Site to Site VPN connections, AWS VPN Client service**, and **a custom VPN server** handling the connections where the outgoing traffic had to be sent with a public, constant IP address.

Last but not least, some time was spent on reviewing the available options in terms of upgrading the technology stack in the system. Among other things, the **MongoDB engine** was successfully upgraded to a newer version.

SOLUTION

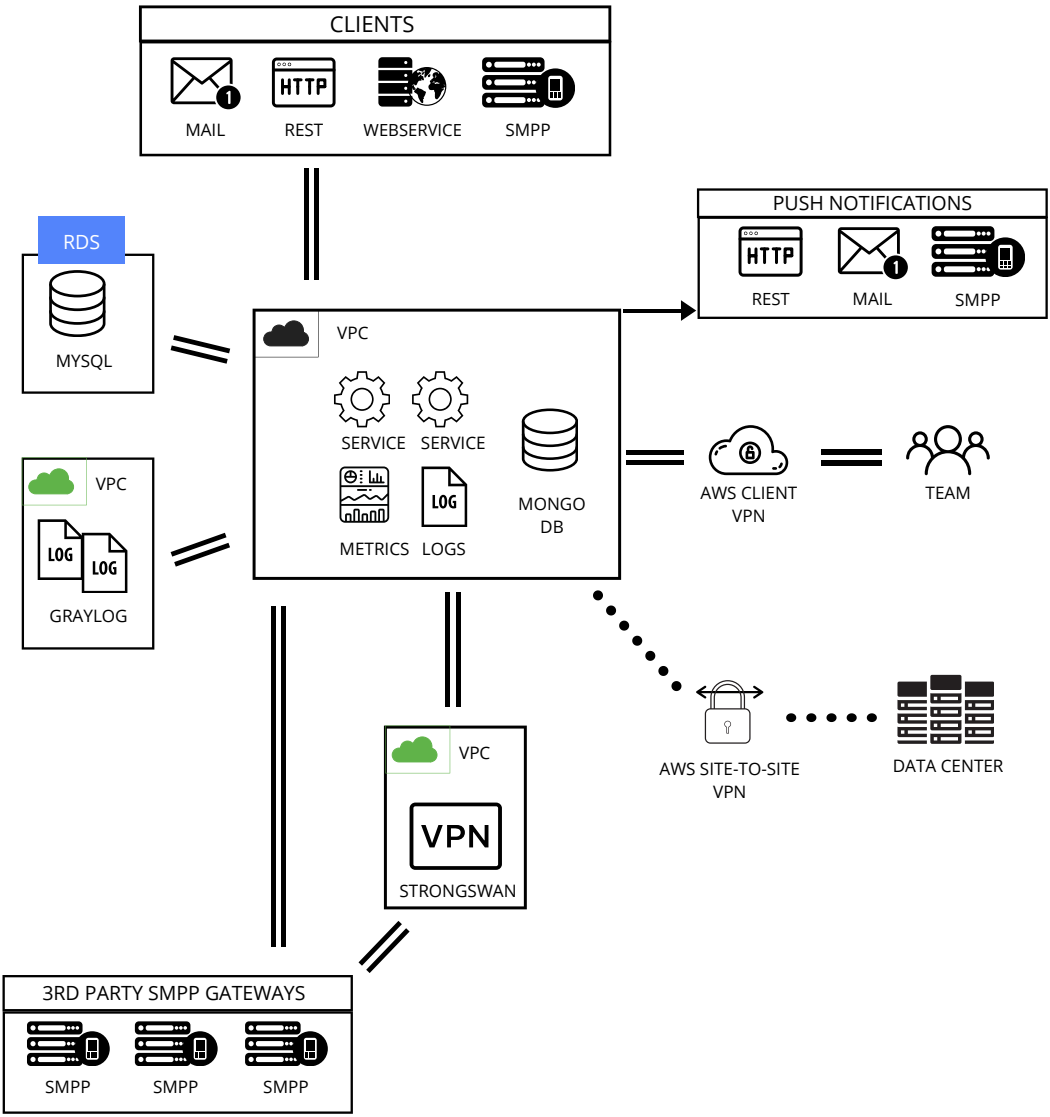


Fig 1. - Intelli Messaging Architecture Overview

RESULTS

As an outcome of this work, the Intelli Messaging system has **increased overall traffic handling capacity**. The solution already had a chance to prove its worth. Shortly after the official launch, it received traffic several times higher compared to what it was capable of handling before. It wasn't just a single spike but it lasted for a few hours. The client immediately felt **more confident about the stability and throughput of the system**.

The system can now **automatically scale out or scale in** based on the actual resource usage. No manual intervention needed, all that is required is to maintain the rules which define the scaling triggers. Thanks to the Kubernetes and EKS foundation, the system is much more **reliable and resistant to failures**. The number of incidents that cannot be resolved by automatic service redeployment has gone down significantly. **Service availability is higher**, giving the customers more confidence in the quality of service.

Physical infrastructure incidents are not a problem anymore. Things such as hard drive failures and equipment redundancy are managed by the cloud provider, taking away the stress and effort required to handle this kind of incident under the pressure of time.

System owners now have a much **better insight into the costs of running the system**. AWS provides clear and up-to-date reports about current billings, therefore it's easier to make predictions and plan the budget. The AWS Cost Explorer service can also provide some **recommendations on the most optimal AWS resource usage** based on historical data.

RESULTS

The service **deployment process is now faster, leaner, and more resilient to potential issues.** Rolling back a failed deployment is a matter of seconds instead of minutes. Releases can be performed more often, with a higher level of confidence and lower level of stress.

The complete migration from the on-premises data center to AWS cloud solutions took about a year to finish. With the system foundation based on the cloud, it became **easier to identify the right direction for future improvements of the overall system design and performance**.

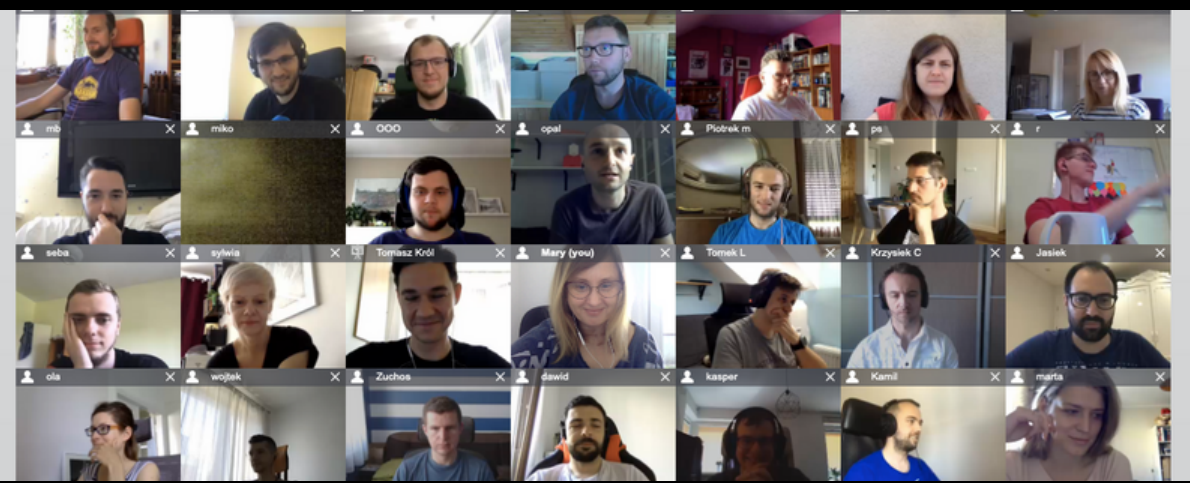
"We didn't want to import our existing problems into a new environment. SoftwareMill was pragmatic in determining what needed to be re-architected and what could be "lifted and shifted". Using Kubernetes, Helm, and Terraform on AWS, SoftwareMill was able to quickly test and validate different migration strategies with the aim of minimizing service disruption to our customers down to minutes, not hours or days.
As a result, we were able to confidently migrate our services from our on-premise environment to AWS and make improvements to our critical components and processes along the way."

Adam Lau, CIO, Intelli Messaging

SOFTWAREMILL

GOT AN IDEA?
WE'LL MAKE IT HAPPEN

contact@softwaremill.com
www.softwaremill.com



We are SoftwareMill, a Poland(EU)-based consulting & custom software development company, delivering services remotely, worldwide for over 10 years. Being experts in Scala (Akka, Play, Spark), Java, Kotlin we specialize in blockchain, distributed big data systems, machine learning, IoT, and data analytics.

We are a leading consultancy chosen for digital transformation. Integrity, versatility, understanding of the business, right soft skills, strong work ethic, rich experience and top notch mastery of technology makes us the perfect choice. Have a project in mind? We can proactively transform your business with technology