# CASE STUDY

## Custom CI/CD Solution Deployed on AWS Helps Meal Delivery Service Transform Microservices Architecture

softserve

## Client Background

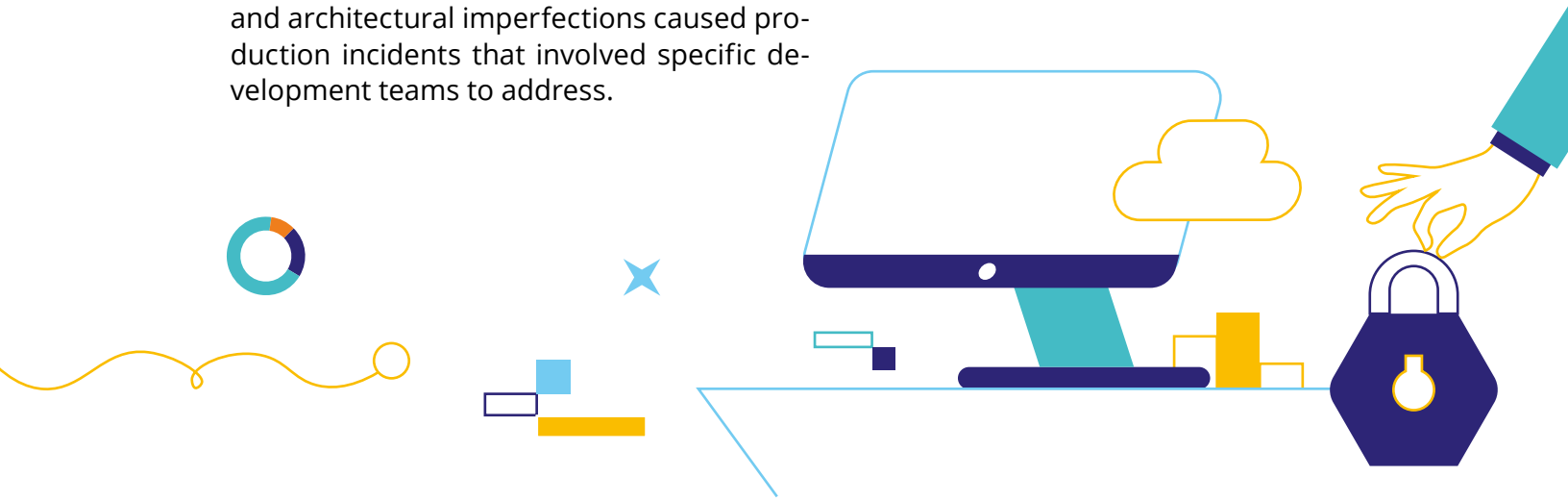
Our client is a top US-based meal delivery service.

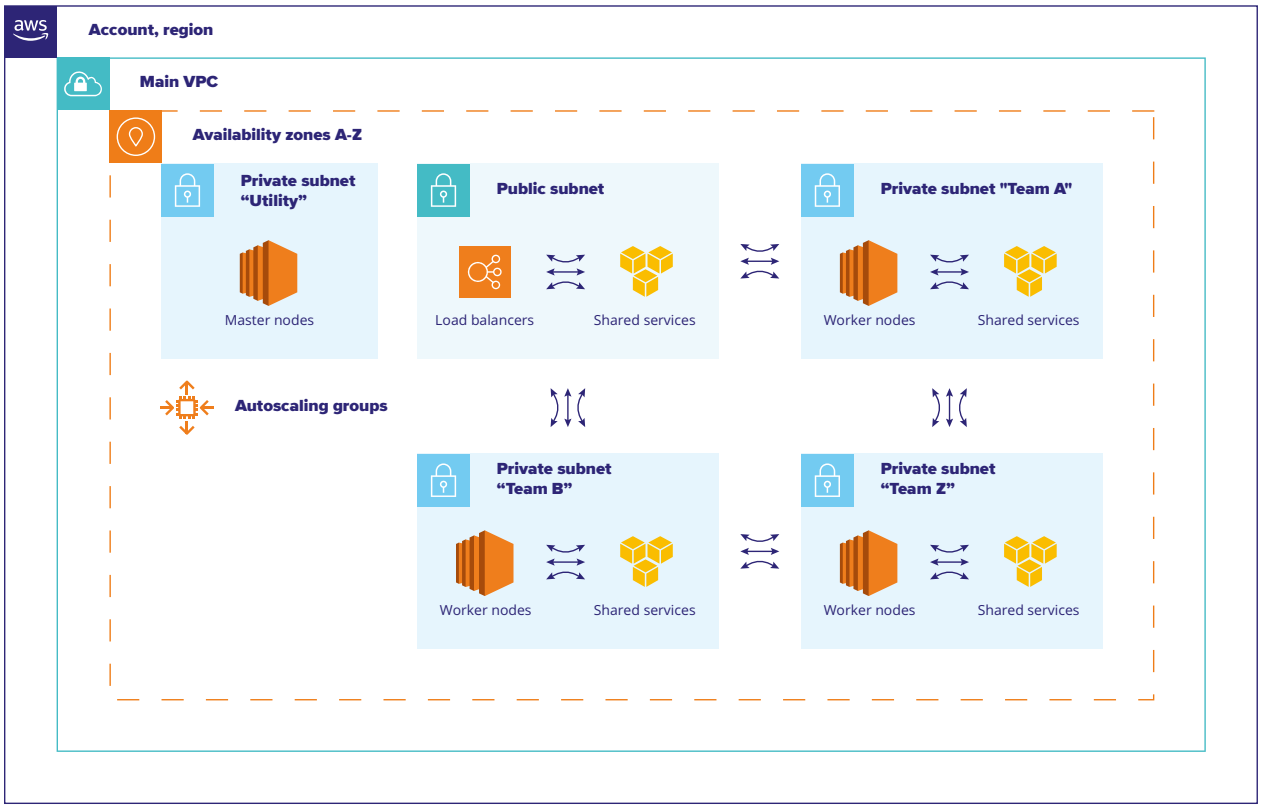
## Business Challenge

Our client owns a web platform based on microservice architecture. Portions of the platform have to communicate with on-prem devices and software. The main part of the solution leveraged a Kubernetes cluster semi-manually deployed into AWS using a mix of an open-source tool and Terraform configurations. The business has strict reliability requirements that blocked the possibility of updating the cluster version and its core components in runtime. There was significant risk of losing the cluster because of an update failure, and together with other limitations, was impeding desired infrastructure improvements. At the time the project with SoftServe began, our client had lost most of its associates who had deep knowledge on the infrastructure history, architectural drivers, and decisions made. Outdated infrastructure components and architectural imperfections caused production incidents that involved specific development teams to address.

Furthermore, the Chef orchestration tool was used to provision new instances, including Kubernetes instances. Its configuration had critical issues that caused the launch of new nodes to be successful only half the time.

Additionally, regarding the infrastructure, a custom CI/CD solution was created. The CI solution imposed significant restrictions on application delivery, making unusual CI use cases nearly impossible to implement. The CI tool, despite its good quality, prevented developers from gaining actual knowledge about Docker, Kubernetes, and related open-source tools. To meet the growing internal development and business needs, our client needed to update the Kubernetes cluster deployment architecture and prepare a flexible CI/CD solution that meet current industry standards.






## Project Description

The project included several parts. Some were extracted into separate projects, such as improvements in the logging system, which could be done in earlier stages than other larger planned architecture changes. The proposed solution included improvements in several areas.

The first and most important part was the networking architecture change. Two additional network segments with similar settings were prepared in the production environment for future Kubernetes installations to make future migrations between different cluster configurations smoother and safer. This change made it possible to do architectural experiments in production without the risk of affecting all users at once. This was done not only to update the Kubernetes cluster version, but also to perform significant changes to any internal components without interrupting all running applications at once.

Secondly, the cloud-based Kubernetes service AWS EKS was chosen for future use. The decision shifted the responsibility of maintaining the cluster's core components to the cloud provider. This change also brought another networking extension in Kubernetes, which was deeply integrated with AWS VPC, improving the overall networking performance.

Thirdly, all the configurations of the EKS installation were reflected in code, which included a combination of Terraform configurations, CI/CD jobs, and Kubernetes resources in the form of Helm charts or plain YAML configurations, stored under a source-control repository. This guaranteed transparency, disaster-recovery opportunities, and eliminated worry of any accidental infrastructure or cluster component changes in any environment, including production.

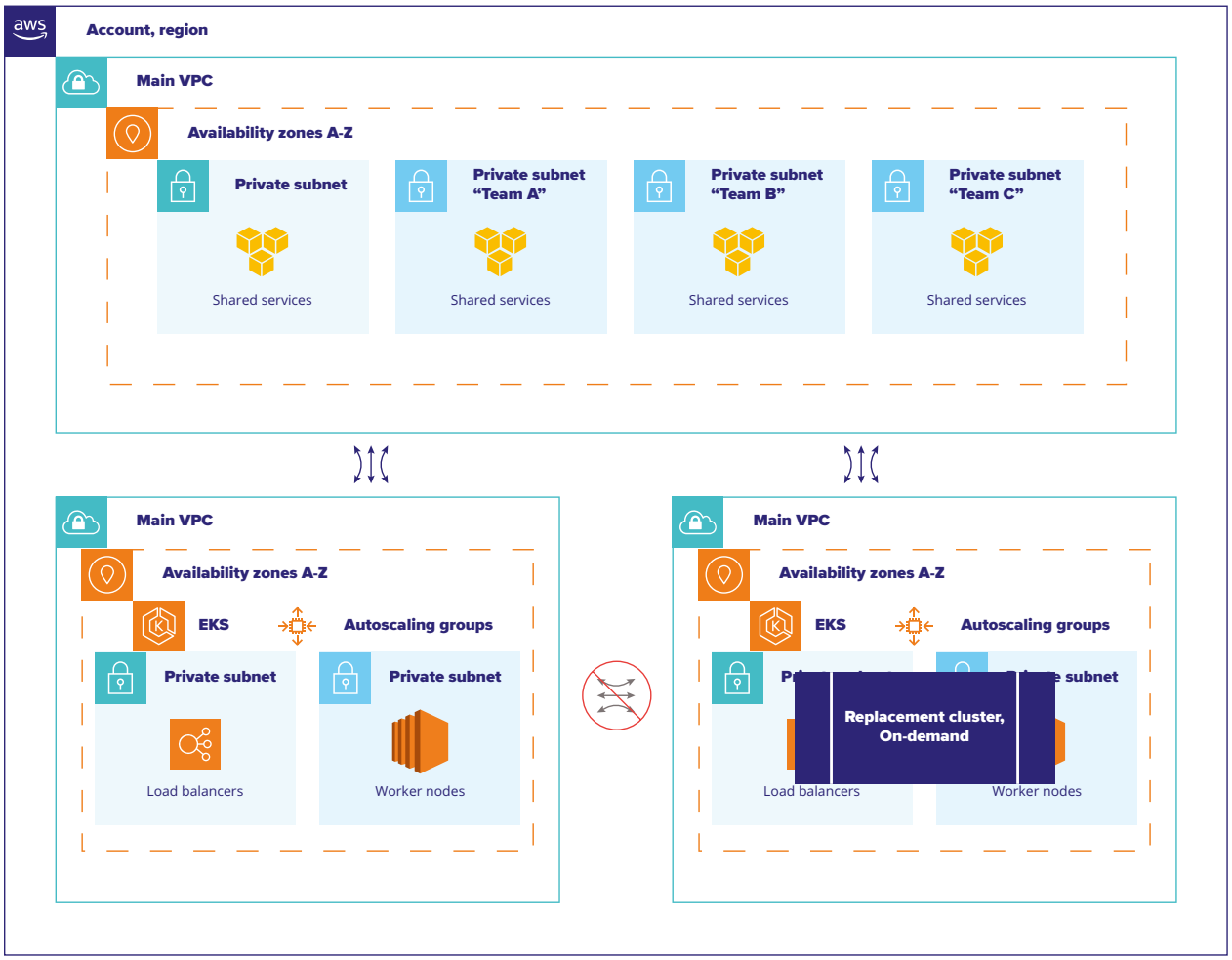
Additionally, a new role-based authentication was configured based on a combination of an existing SAML-provider with IAM roles and Kubernetes RBAC settings which increased the overall security of the cluster.

Another noteworthy improvement was the use of a different load balancing method. The Traefik Ingress Controller from the previous cluster was replaced with an ALB Ingress Controller in the new setup. Since the ALB Ingress Controller stays beyond the cluster itself, this step improved the overall reliability and network performance.

One of the principal requirements was to support gRPC communication between services. To make the communication stable and transparent, a service mesh component, Linkerd, was used in the previous Kubernetes installation. The new cluster setup also included Linkerd, but it was updated to the second generation, which is significantly different from the first version—it requires changes to the deployment process and included several new features and improvements.

The next key change was to replace the legacy CI tool with a new multifunctional solution based on top of Helm and Jenkins features. Helm is a popular package manager for Kubernetes and unlike many third-party offered packages allows for the creation of custom packages using a well-known standardized method. Jenkins is an old, but powerful automation server. A custom CI library was developed to make the new CI solution extremely flexible, transparent, repeatable, and scalable. The primary idea around this CI version is "conventions against restrictions" which allows developer teams to implement unusual CI/CD pipelines without waiting for DevOps team members to become available. Also, the solution makes possible the evolutionary development of CI/CD without the mandatory updating of all microservices when any new changes were introduced into the primary libraries.

The Kubernetes node bootstrapping process was improved. The Chef orchestration tool is still used to install complimentary utilities, improve monitoring, and add other tools to an instance. However, any configuration errors no longer block the instance from being added to the cluster and start processing its payload. This change considerably reduced the time required to add new instances to Kubernetes clusters.




## Migration

Each complex system has its history, and revolutionary changes can do these a disservice; not only because it's undesirable to interrupt production services under load, but because people have limited bandwidth. Spreading knowledge across a company requires time, patience, and effort. Some relationships between applications aren't visible until they break and sometimes code has issues and does not meet modern requirements, therefore it's not always easy to move into a new environment. Migrations take significantly longer time and include thousands of details to track. This project was no exception.

Around 150 separate micro-services were thoroughly analyzed, prepared, and converted into the new CI. The final migration into the new production environment required close control over all cross-dependencies, splitting all services into separate groups based on their public or private relationships (Strangler pattern). Every unexpected obstacle was analyzed, discussed with teams, and eventually overcome thanks to constant and close cooperation with developers and management. This made it possible to solve problems that required attention in completely different directions.

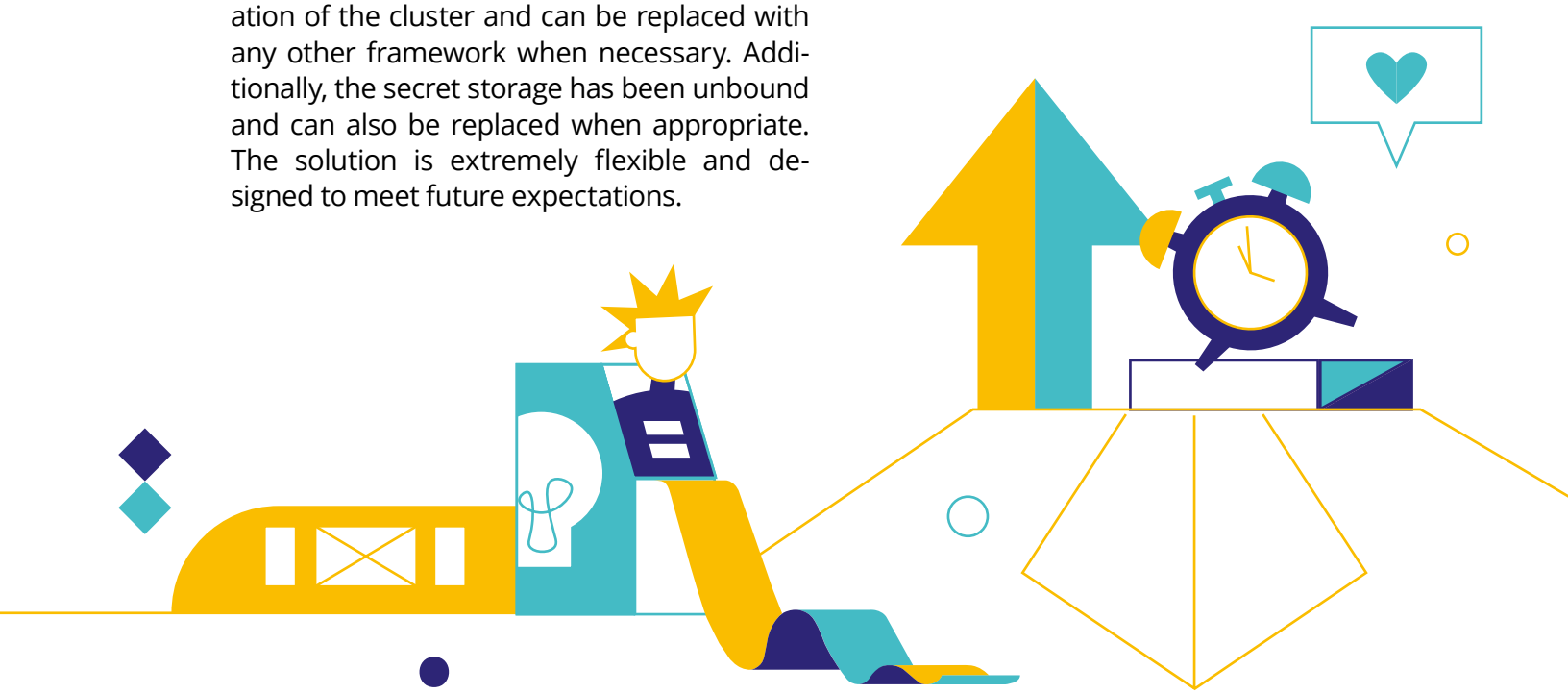
## Value Delivered

Despite the 'expected surprises' that appeared during project implementation, the new environment was built and installed and the new CI replaced the legacy CI for all Kubernetes-based projects. Our client's teams gained the knowledge required to use the system effectively. The updated cluster contains a cutting-edge architectural stack, it's upgradable, reliable, requires less effort to maintain, and is transparent in any single trait, plus the vast majority of its configuration is controlled via code under Git. The number of worker nodes in the cluster was cut in half. EC2 instances were replaced with their modern generation and reserved upfront to reduce costs. Each worker node now has better resource utilization.

The migration was completed under load in real-time with minimal requests interruption. The newer recipes for Chef (orchestration framework) are no longer crucial to the operation of the cluster and can be replaced with any other framework when necessary. Additionally, the secret storage has been unbound and can also be replaced when appropriate. The solution is extremely flexible and designed to meet future expectations.

## Lessons Learned

No matter how thoroughly research was conducted, new findings are guaranteed to be discovered at the implementation stage. This means that efforts put into planning are important by definition, but proper testing is required. Small changes should be delivered as soon as possible when they are ready, but big changes must stand for a while on stage to be proven. Migrations cannot be controlled completely unless all of the available resources are applied to achieve the goal. Since migrations are not something that business need to focus on, there will always be external factors that block some of the stages and increase the time. Continuous control and efforts allow what was started to be completed to clear a path for new aspirations and ideas.



# ABOUT US

SoftServe is a digital authority that advises and provides at the cutting-edge of technology. We reveal, transform, accelerate, and optimize the way enterprises and software companies do business. With expertise across healthcare, retail, energy, financial services, and more, we implement end-to-end solutions to deliver the innovation, quality, and speed that our clients' users expect.

SoftServe delivers open innovation, from generating compelling new ideas, to developing and implementing transformational products and services.

Our work and client experience is built on a foundation of empathetic, human-focused experience design that ensures continuity from concept to release.

We empower enterprises and software companies to (re)identify differentiation, accelerate solution development, and vigorously compete in today's digital economy. No matter where you are in your journey.

Visit our **website**, **blog**, **Facebook**, **Twitter**, and **LinkedIn** pages.


**NORTH AMERICAN HQ**

201 W 5th Street, Suite 1550
Austin, TX 75703
+1 866 687 358

1 University Avenue Suite 11-112
Toronto, ON M5J 2P1
+1 647 948 7638

**EUROPEAN HQ**

14 New Street
London EC2M 4HE
+44 (0) 800 302 9436

info@softserveinc.com
www.softserveinc.com


softserve