Modernizing Construction Planning Platform: DevOps Success Story

Big Data Infrastructure Revamp for a Construction Planning Leader

Our client provides planning software for construction firms tackling massive industrial projects. Their big data solution allows companies to virtually map out every aspect before building — from 3D design modeling to cost, timelines, and workforce planning.

The client aimed to scale up the product and enhance its reliability. IT Outposts helped make already powerful software even better.

Project Description

Our client was dealing with a sizable, intricate on-premises-style cloud setup hosted on Azure. To support each client, they duplicate the entire environment, creating separate single-tenant and multi-tenant versions for data isolation. As their architecture expanded with more microservices and virtual machines, management became more cumbersome.

IT Outposts modernized the client's setup by migrating to Kubernetes, automating CI/CD, implementing test automation, and more.

Provided Services

Kubernetes managed services

- Efficient scaling
- Improved reliability
- Cost savings
- Faster deployments

Cloud adoption services

- Cloud infrastructure management

DevSecOps services

- Automated security testing

DevOps services

- Infrastructure and architecture design
- CI/CD automation
- Integration

SRE services

- Log management and monitoring

Work agenda

Client
World-class construction big data company (confidential client)

Location
Undisclosed under NDA

Technical team
3 DevOps engineers

Project timeframe
March 2023 - ongoing

Budget
500,000

Project goals

- Migrate client environments from traditional virtual machines to a containerized setup on Kubernetes clusters
- Achieve cost optimization by transitioning to Linux from licensed Windows environments
- Implement automated CI/CD pipelines for streamlined application builds, testing, and deployments
- Establish comprehensive monitoring and observability across Kubernetes and Azure cloud infrastructure
- Introduce automated testing capabilities, including static code analysis and functional test automation

Challenges

01 Client environments based on virtual machines

A major part of this project was moving each client's individual environment from traditional virtual machines over to a modern, containerized setup using Kubernetes. But there was an important first step before Kubernetes could even enter the picture.

Since the client's applications were originally built to run on virtual machines, we had to first re-package them to run reliably inside containers. The process of "containerization" prepares applications to become more cloud-friendly and container-native.

02 Steep Windows licensing costs

Previously, our client's services ran on Windows. Migrating to Linux immediately provided cost savings since Linux is free to use, unlike Windows, which requires licensing fees. However, from a technical standpoint, the migration was akin to trying to run an Android app on an iPhone. It simply wouldn't work. The two operating systems are too fundamentally different under the hood.

03 Inefficient manual deployments

Before our cooperation, deploying any updates or new features was a fully manual process for the client. Developers packaged the latest code versions but then had to hand them off to operations teams to actually deploy. With so many code changes and environment transitions, having automated deployment pipelines was crucial.

04 Lack of unified observability across Kubernetes and Azure

As the client's apps moved to run on Kubernetes clusters in the Azure cloud, full visibility across the modern cloud setup was needed. They required insights into the health, performance, and behavior of workloads spread across multiple clusters. Monitoring also had to cover the Azure cloud infrastructure layer.

05 No existing automated testing

The client only tested features manually. But as developers created new code or made changes, it was easy to accidentally introduce security vulnerabilities. Static code analysis was needed to automatically scan through all the code and find these potential issues.

Additionally, the client's apps have many different features and user flows. Manually testing all the possible scenarios every time code changes would take a huge amount of time and effort.

Contacts

Ready to expand your infrastructure capabilities? If you are dealing with the complexities of on-premises systems, aiming to enhance scalability, or looking for significant cost reductions, our expertise is tailored to your needs. Discover how IT Outposts can modernize your big data infrastructure with Kubernetes solutions, as evidenced by our partnership with a leading construction planning firm.

Discuss your project

Solutions

1. Containerization and migration to Kubernetes

Our engineers first containerized the client's services. This involved repackaging their apps to run reliably inside containers. We've migrated part of their client workloads to the modern Kubernetes environment. The remaining client environments still need to go through containerization before they can also be migrated over.

2. Linux transitioning assistance

Our team provided assistance by directly contributing changes to the client's source code. We worked closely with their development team to adapt their services to run on Linux.

3. CI/CD implementation

We leveraged modern CI/CD tools and practices to establish automated software delivery pipelines.

4. Providing unified observability

To effectively operate and monitor the applications deployed across Kubernetes clusters on Azure, we implemented a unified monitoring and observability solution. This involved standard open-source tools:

- Prometheus for collecting metrics on resource usage, traffic, performance, etc., of the Kubernetes containers and pods
- Grafana, a dashboarding tool integrated with Prometheus to visualize and analyze Kubernetes metrics through customizable dashboards

In addition, our team configured monitoring for the underlying Azure cloud infrastructure.

5. Test automation

On our side, we implemented automated testing by setting up various testing systems, including code analysis automation and automated functional testing.

With code analysis automation, the client can catch code problems and vulnerabilities before buggy versions go into prod, where issues could cause bigger problems.

The automated functional testing allowed running full tests continuously to verify new changes don't break existing features.

Results

Migrating to a Kubernetes container environment made the entire solution much more scalable to handle future growth and highly available to minimize disruptive downtime.

By transitioning from Windows to Linux, the client eliminated hefty Windows licensing fees, lowering their operating costs

Now, with automated CI/CD pipelines in place, the client can push out around 250 automatic deployments every month. That's a game-changer for getting their latest work into end customers' hands quickly.

Monitoring tools like Prometheus and Grafana gave the ops team complete visibility into the Kubernetes clusters and Azure infrastructure. Real-time observability ensures issues get identified and resolved rapidly.

Software quality and reliability got a huge boost from test automation. Both the code itself and the app features go through automated checks, which helps catch bugs and before they cause problems for end users.