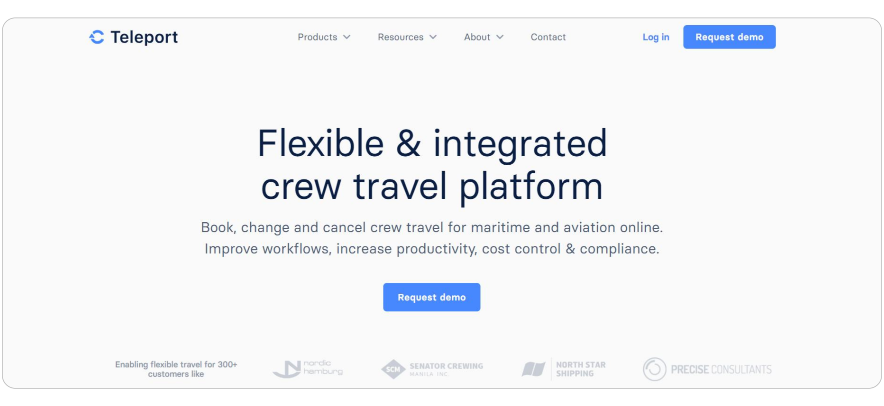
Teleport

Improved Scalability, No Downtime: C-Teleport's Docker Swarm to Kubernetes Transition

C-Teleport's vision is to simplify marine travel bookings for businesses. Founded in 2017, the company built an automated self-service platform so businesses could easily arrange or change bookings. The flexible self-service model proved popular; however, existing infrastructure struggled to scale. Scaling became even more difficult due to the multi-cloud environment and poor visibility.

Collaboration with IT Outposts empowered C-Teleport to stay fixed on what they do best: delivering top-tier booking experiences.



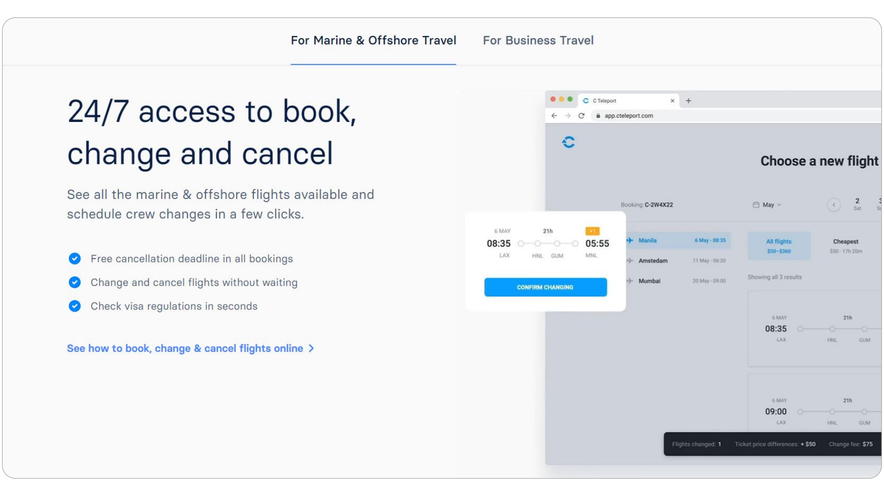
Project Description

C-Teleport's expanding clientele relies on over 70 critical microservices powering booking functionalities that must stay online 24/7. Over the years, the underlying Docker Swarm started posing scalability issues, while C-Teleport had ambitious plans to grow 2.5 times. Our main task was upgrading the platform to an optimized Kubernetes architecture.

We also needed to implement critical site reliability engineering (SRE) practices, as outages directly impact revenue streams.

Adding further complexity, C-Teleport's infrastructure spanned both AWS and lower-cost Hetzner cloud environments. The other task was to address this multi-cloud model's security, networking, and configuration challenges.

Taking each challenge one step at a time, we achieved stable results.



Key DevOps Metrics

Metric	
< 25	Frequency of Deployments per month
10 min	Average Lead Time for Changes
1%	Change Failure Rate
1-2 min	Time to Restore Service

Provided Services

DevOps Services

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

Kubernetes managed services

- Efficient scaling
- Improved reliability
- Faster deployments

SRE services

- Log management and monitoring
- Incident management
- Release management
- Performance optimization

Operations managed services

- Technical support
- Infrastructure maintenance
- Capacity planning
- Disaster recovery as a service

Cloud adoption

- Cloud cost optimization
- Cloud infrastructure management

Work agenda

Client

Flexible & integrated crew travel platform

cteleport.com

Location

Netherlands

Technical team

Tech lead

3-5 DevOps engineers (depending on the workload)

Project timeframe

July 2022 - ongoing

Budget

150,000

Project goals

- Incrementally migrate mission-critical microservices to Kubernetes without disruption
- Interconnect old Docker Swarm and new Kubernetes infrastructure
- Establish SRE practices focused on actionable metrics tied to business sentiment
- Build in monitoring redundancy to assure visibility in case of outages
- Connect and secure hybrid cloud environments
- Optimize project costs
- Preserve developer productivity during migration

Challenges

01 Performing migration without downtime

Swarm's shortcomings around lots of manual work, outdated scripts, and an ageing management system made it insufficient to meet the client's growing needs. However, C-Teleport had no in-house expertise in Kubernetes, so orchestrating this shift would pose extreme availability risks.

C-Teleport has an intricate web of microservices, providing a smooth booking experience. Any downtime or migration-related asynchronous delays would directly cause ecosystem breakdowns, resulting in revenue losses from abandoned bookings.

A key challenge was migrating services from Docker Swarm to Kubernetes without causing any downtime or other issues for the client.

02 Enabling secure hybrid cluster connectivity

While some apps have been migrated to Kubernetes, the legacy Swarm cluster still hosts several business-critical services. An overnight migration would result in intolerable downtime and data loss. Interdependencies now exist between Swarm and Kubernetes clusters: applications on Kubernetes need secure access to backend data repositories remaining on the Swarm cluster.

Enabling reliable data flows between different orchestrators causes multifaceted integration obstacles.

03 Lack of observability in the complex microservices landscape

With multiple microservices powering booking workflows, our client lacked full visibility into the health of their services. After migrating to Kubernetes, strong monitoring and alerting were critical. Otherwise, C-Teleport would face performance problems, causing revenue impacts.

04 Maintaining visibility in case of cluster-wide outages

A major benefit of Kubernetes is its capacity to self-heal routine issues. However, some failures can still affect the entire cluster. Such large-scale failure scenarios could impact the core monitoring tools like Prometheus and Grafana deployed inside the cluster.

This would be catastrophic from an observability perspective — with core monitoring failing, there's no way to get alerts that the main Kubernetes cluster is down. Lack of visibility would critically slow down incident response and remediation efforts.

05 Managing a complex multi-cloud environment

With infrastructure and services split across AWS and Hetzner, managing this multi-cloud landscape posed significant challenges. Lack of connectivity between AWS and Hetzner can make it hard to pinpoint the root cause of outages due to visibility gaps. Running several environments also increases security risks, with more entry points vulnerable to attacks and data theft.

Additionally, C-Teleport's production infrastructure resides in AWS across multiple accounts serving different business functions (marketing systems, databases, etc.). Each account requires controlled interconnectivity with the other accounts.

06 Optimizing cloud costs without impacting platform resilience

C-Teleport is highly cost-conscious. As a booking platform, their costs scale in proportion to traffic and revenue. So, there's constant pressure to maximize value from every dollar spent on resources.

However, excessive cost optimization could compromise stability and resilience, which can be even more expensive due to the revenue impact of downtimes.

07 Disrupted workflows for C-Teleport's developers

The gradual migration from Docker Swarm to new Kubernetes stacks could complicate workflows for developers at C-Teleport. Deployment destinations diversified across Swarm and Kubernetes might disrupt developers' routines, increasing the need to switch between contexts when debugging code or releasing code changes.

Contacts

Ready to transform your platform's scalability and reliability? Discover how IT Outposts can architect your seamless transition to a cutting-edge Kubernetes infrastructure, just as we did for C-Teleport. Whether you're facing scaling challenges, aiming for zero downtime, or navigating the complexities of multi-cloud environments, our expertise is your solution.

Discuss your project

CTO at C Teleport
Petr Kirillov

“They're a great expert that we can trust!

Read full review

Solutions

1. The gradual and staged migration process

Instead of a risky “lift-and-shift” migration, microservices were moved incrementally to minimize errors. As the first step, we migrated non-critical applications without intricate dependencies.

As the next step, foundational systems like Redis, MongoDB, and RabbitMQ databases were gradually transitioned upon passing dependency analysis.

We continue migrating the other sensitive services, ensuring no disruption occurs.

2. Bridging Swarm and Kubernetes clusters with Consul service discovery

We set up a VPC tunnel between the old and new clusters to enable a secure connection between Swarm and Kubernetes. Both share a common service discovery tool, Consul, making them appear as a single ecosystem. Any service can query Consul to locate where its dependent service resides, regardless of which cluster hosts it.

3. Implementing foundational SRE practices

We're adopting site reliability engineering (SRE) practices to better understand our complex tech systems. The main idea behind SRE is to tie metrics to real-world business outcomes. The plan is to implement service-level objectives (SLOs) and service-level agreements (SLAs) for each infrastructure service and its component. This will give us a clearer insight into how well our technology delivers on key metrics impacting customers and revenue.

Meanwhile, we implemented standard tools for metric monitoring, including Prometheus, Grafana, and Alert Manager. We categorized dynamic services into tiers based on potential end-user or revenue impact. Each tier has tailored SLOs defining action plans in case of emergencies.

We prioritize metrics like latency and error rate directly impacting the end-user experience. This allows us to gain visibility into how infrastructure changes ultimately affect customers.

In addition, we set up multiple alerting channels for issue detection. The multi-channel approach helps connect monitoring to the right skill sets and deliver timely resolutions. App admins assess code-level causes, while infrastructure alerts call on ops experts suited to tackle those system-wide issues.

4. Setting up a redundant monitoring cluster

We deployed dedicated Prometheus and Grafana instances onto a separate management cluster to ensure monitoring continuity even if the main Kubernetes cluster goes down.

If a large failure disables the main cluster, the redundancy cluster with a mirrored monitoring stack will still gather health metrics. It will promptly trigger alerts that the main Kubernetes cluster is unreachable.

5. Hybrid cloud management through unified connectivity, firewalls, and access controls

To connect the AWS and Hetzner Cloud environments securely, we established an IPsec tunnel. It enables the services to communicate internally using private IP addresses.

Further, we configured VPC peering to enable communication between services across the Amazon accounts.

6. Optimizing costs through the hybrid cloud, metrics retention policy, and SRE practices

One way to provide project cost-efficiency is through the use of multiple clouds. Hetzner offers a cheaper infrastructure for workloads where AWS resilience isn't mission-critical.

For example, GitLab has been self-hosted on dedicated Hetzner hardware instead of using GitLab's SaaS offering. Though self-hosted options require their own maintenance, for GitLab's scale, this is still cheaper than continual SaaS licensing fees.

Furthermore, cloud-based monitoring systems typically consume large budgets — monitoring generates terabytes of data, which translates to cloud storage bills. To control these costs, we implemented a metrics retention policy. Metrics are automatically deleted after 30 days. Keeping metrics for this period is sufficient for troubleshooting and spotting trends (60 days max for some special cases).

Finally, setting up SLOs and SLAs will prove beneficial for revenue protection. Diagnosing root causes is challenging in complex tech stacks. SLOs and SLAs will let us quickly identify the exact weak link that triggered the overall failure.

With SRE practices established, we can also translate abstract system failures into actual dollars lost based on downtime minutes. We'll have clear data showing, "If service X is unavailable for 3.5% of the month, this will translate to $Y in lost revenue." Consequently, we'll be able to prevent the most extensive issues before major problems actually impact revenue.

7. Maintaining developer productivity across transitional infrastructure

A priority for our CI/CD architecture was preserving developer familiarity and workflows, even as deployment targets shift with progressive Kubernetes adoption. So, while the underlying infrastructure phases out Docker Swarm for Kubernetes, the developer experience remains minimally disrupted.

Engineers can still logically build, test, and check code through customary pipelines. Committing to the standard Git repository automatically triggers the appropriate deployment workflow to either Swarm or Kubernetes environments without developer intervention.

Results

- Upgraded Kubernetes infrastructure now enables C-Teleport to confidently pursue 2.5x growth.
- Observability tooling with graduated SLOs helps avoid downtimes and customer impact risks from unstable services.
- Zero downtime migration sustained platform functionality.
- Successful modernization reinforced C-Teleport's track record of resilience and availability.

Even with systems growing more robust, the team stays dedicated to keeping the product convenient for customers. IT Outposts stays dedicated to being a responsive and strategic partner, striving not just to maintain but also to surpass expectations around the delivery of a seamless, world-class booking experience.

DevOps tech stack

CI/CD

Gitlab, Flux CD

Monitoring

Prometheus, Grafana, GCP Alerting, ELK

Infrastructure component provisioning

AWS, Docker, Terraform, Kubernetes

Services & databases

Postgresql, MongoDB, RabbitMQ, Redis