info@apriorit.com
www.apriorit.com

Case study

Building AWS-based Blockchain Infrastructure for International Banking

A US-based blockchain development company hired us to implement secure and scalable cloud infrastructure for their smart contracts. They were looking for experts in blockchain development and cloud infrastructure that could deliver a high-quality product under strict deadlines.

We designed, tested, and implemented the AWS architecture and taught the client's DevOps team how to support it. Our team managed to do all of this just in time for the client to present the project to their customers — banking and government organizations.

The client

Our client is a US-based blockchain development company that provides their customers with smart contracts, custom blockchain networks, and tools to help them build a decentralized economy.

Being blockchain experts themselves, they developed several smart contracts for their new project. However, to create cloud infrastructure for these smart contracts, they required specific expertise. That's why the client was looking for an experienced contractor.

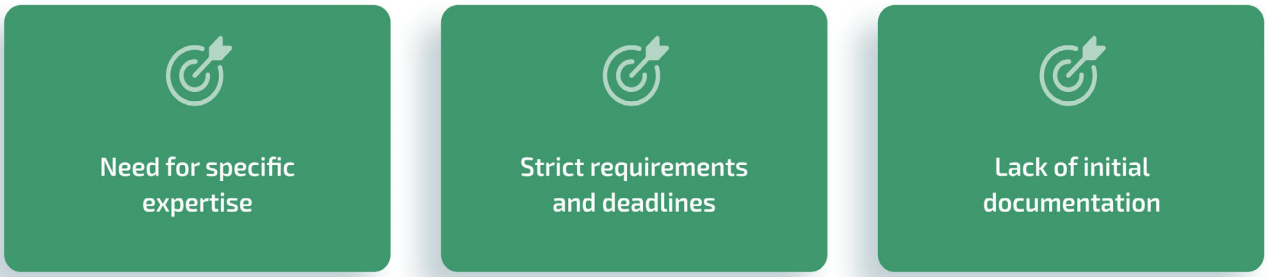
The challenge

The client was looking for a blockchain development team that could set up AWS infrastructure for further deployment of their smart contracts. The cloud infrastructure had to be configured with Terraform, Ansible, and the ELK stack because these technologies were used by the client's previous contractor. Our team also had a limited time to implement the infrastructure.

The client had specific requirements for infrastructure scalability, flexibility, and security because they designed the project for international banks and government organizations. They needed the solution up and running by a certain date to be able to present it to their customers, so we had to finish the project on schedule.

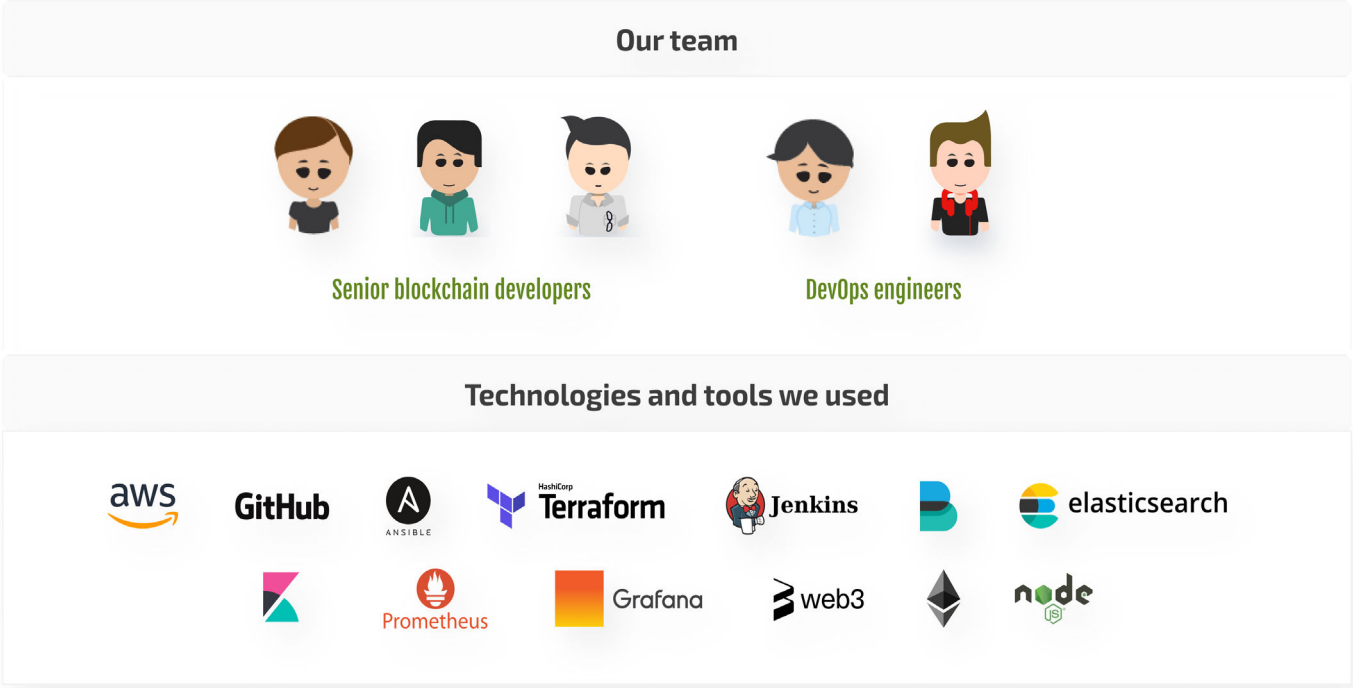
Another challenge of this project was the lack of documentation. The client developed smart contracts in cooperation with a third-party contractor, but we couldn't consult with this contractor on their code.

Key project challenges



Our approach

After analyzing the code and the project's high-level architecture, we put together a team of senior blockchain developers and a DevOps engineer. Our team assessed the project's requirements and current state, then offered to use the most suitable technology stack for developing the infrastructure:



To make sure we delivered exactly the solution the client needed before the deadline, we initiated regular meetings with the client's team via video calls and messengers. This way, we could coordinate our efforts and use our time efficiently.

The result

Our team designed and implemented AWS infrastructure that satisfied the client's requirements and met the project deadlines. As a result, the client was able to deploy custom smart contracts and conduct a project demo for their customers.

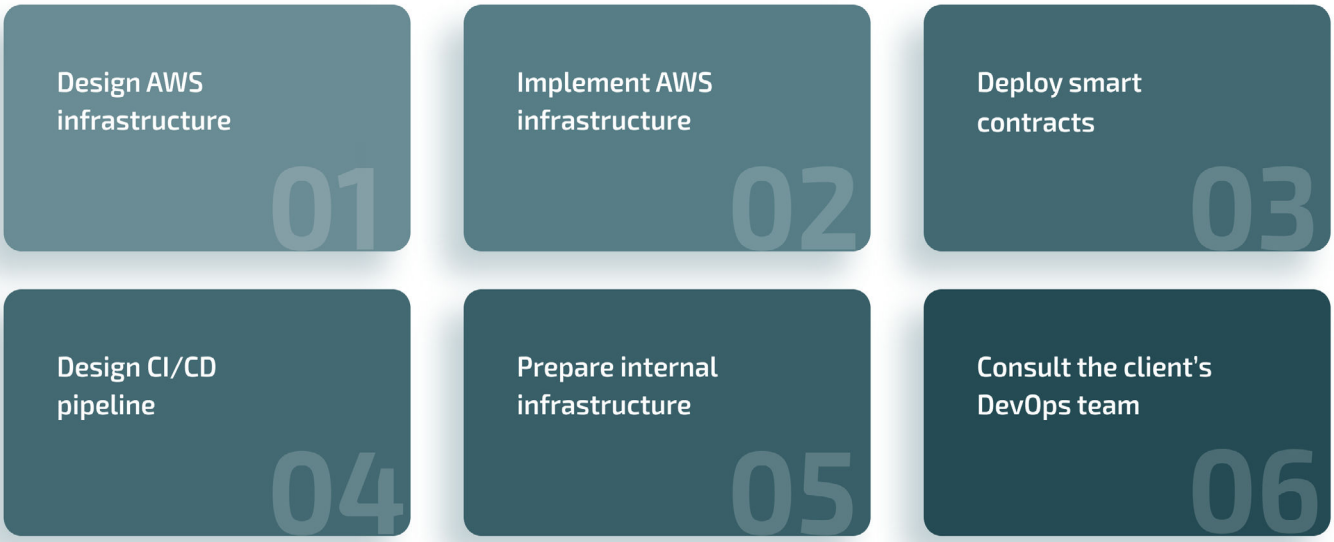
In particular, our team:

- Designed and implemented AWS-based infrastructure
- Added pipelines for solution support and future improvements
- Restored and improved project documentation
- Consulted the client's DevOps team

How we did it

To deliver an efficient product and meet the client's deadlines, we carefully planned our work on the project and outlined the following six stages:

Key project stages



1. Design AWS infrastructure

At the beginning of the project, the client provided us with the documentation they had for their smart contracts and a high-level overview of the infrastructure they needed. We analyzed this information and proposed an AWS-based infrastructure design that would be maintainable, scalable, and fault-tolerant enough for the client's needs.

The client developed smart contracts together with a third party that didn't provide full documentation for their code. Our team couldn't consult with this third party to figure out the missing parts, so we requested the code for smart contracts and analyzed it.

Analyzing code takes more time than studying documentation, but it allowed us to:

- Determine exactly how the code works
- Design infrastructure that perfectly fits the smart contracts
- Save time on future rework and infrastructure improvements

Once the architecture design was finished and approved, we started its implementation.

2. Implement AWS infrastructure

When working on project infrastructure, we used the following AWS tools:

- S3
- EC2
- DynamoDB
- Virtual Private Cloud
- Elastic Container Service
- and others

The infrastructure included three environments: development, testing, and production.

The client also asked us to get their DevOps engineers acquainted with the infrastructure. To do that, we organized several calls and demonstrated how the infrastructure works and how the client could interact with it.

3. Deploy smart contracts

Next, our team started preparing for smart contract deployment, configuring the environments and scripts for testing and deployment. To help the client test the contracts in a secure and controllable manner, we added Docker containers to the infrastructure.

When deploying smart contracts in the testing environment, we paid special attention to their security. Once the client had finished their QA activities, our team deployed and initialized the smart contracts in the main AWS environment.

4. Design the CI/CD pipeline

Adding a continuous implementation and continuous deployment (CI/CD) pipeline to the infrastructure helped us simplify and automate maintenance activities. For example, this pipeline automatically uploaded all changes and new commits to AWS and eliminated the need to manually deploy new code.

5. Prepare internal infrastructure

At the final stages of the project, we needed to finish the AWS infrastructure configurations and set up all internal tools. The client had tried to configure the internal infrastructure before they started working with us. But the mechanisms they used were outdated and did not fit the new infrastructure, so our team started from scratch.

During this stage, our team managed to:

- Configure automated log collection and analysis
- Add new dashboards
- Configure ways for solution components to communicate with each other

These activities helped the client monitor the solution's performance and discover issues and improvement opportunities.

6. Consult the client's DevOps team

Once the infrastructure was working according to the client's requirements, we started transferring management of it to the client's DevOps team. However, they lacked experience supporting such infrastructure and working with some of the tools we used.

To help the client's DevOps engineers, our team prepared detailed documentation and descriptions of infrastructure mechanisms and answered all their questions.

The impact

The client got efficient, scalable, and secure AWS infrastructure for the project within strict deadlines. They also received some improvements, as a part of the project developed earlier by a third party was enhanced by the Apriorit team.

Apart from that, the client's DevOps engineers leveraged our expertise and received extensive consultations on how to support the system and implement further improvements.

As a result, the client was able to conduct a project demo for several international banks and government organizations. They also have the knowledge and tools to manage the solution according to their needs.