Blockchain and Infrastructure as Code

Mateo Florez
mateofc@kth.se

Sanherib Elia
sanherib@kth.se

May 2022

Contents

1 Introduction

The demands placed on software products have risen rapidly in recent years. As a result, development cycles are becoming shorter and maximum availability and flexibility are constantly sought. Therefore, besides optimizing code development, one of the pillars of a functional, stable, and, above all, competitive overall architecture is the maintenance and constant improvement of the underlying hardware infrastructure. Here is where the concept of Infrastructure as Code (IaC) comes into play, specifically designed to increase the quality and efficiency of infrastructures.

Many of the processes within IaC are tedious and ineffective due to their centralized nature and lack of automatization. Here is where DevOps can take advantage of new technologies such as Smart Contracts and Blockchain. For example, if GitHub is used for the remote repository, its synchronization feature will be unavailable for developers if the servers are down. Having a decentralized architecture avoids such issues.

2 Background

2.1 Infrastructure as Code

Infrastructure as Code is an IT paradigm describing hardware in machine-readable code. This principle makes it possible to automate much of the architecture and management of the IT infrastructure in order to be able to react precisely to changing or completely new needs.

IaC's primary objectives and functions include, among others:

- Automate manual processes as much as possible.
- Blur the boundaries between applications and the environments in which they run.
- Establish a flexible workflow that facilitates the collaboration of all those involved in the development process across the enterprise.
- Present content changes and movements transparently and in detail at all times.
- Make hardware configuration as verifiable as software configuration [8].

2.2 Blockchain

Blockchain is a technology set that allows for a secure, decentralized, synchronized, and distributed record of digital transactions without third-party intermediation. In this sense, a complete definition is the one given by Don & Alex Tapscott in their book Blockchain Revolution: "an incorruptible digital ledger of economic transactions that can be programmed to record financial transactions and virtually everything of value."

Data is stored as Blocks, hence the name Blockchain. Each of the data blocks is protected and linked to each other, allowing the participation of specific users (each one associated with a block). Therefore, the transaction is not verified by a third party but rather by the network of nodes (computers connected to the network), which also authorizes any updates to the Blockchain by consensus [12].

For example, company A wants to send money to company B. When doing so with Blockchain, the transaction is represented as a block of data transmitted to each party that makes up the network. The network must then approve the transaction's validity. Upon approval, the money is transferred, and the block will be added to the chain, thus creating an immutable and transparent record.

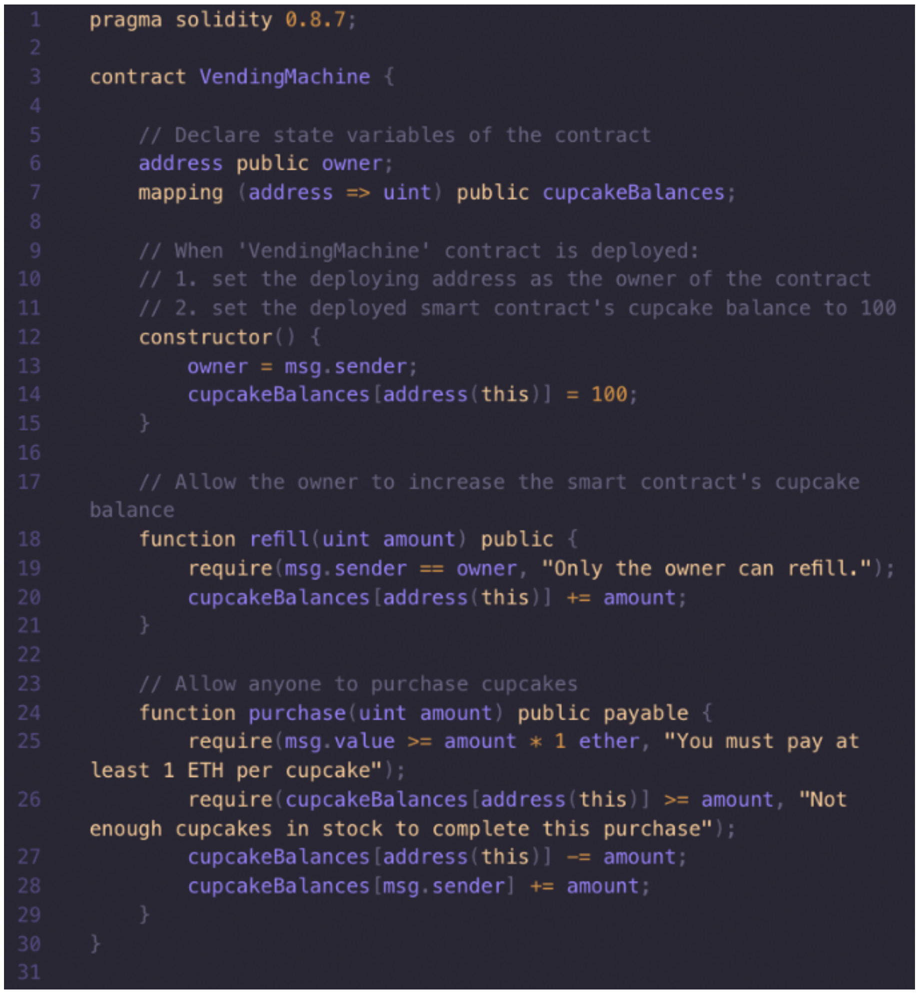
Thereby, Blockchain technology fulfills the function of recording, preserving, and protecting the information of any digital transaction, without the intervention of third parties. In other words, it operates as a shared and continuously updated database, which facilitates the exchange of assets and the management of smart contracts, among other options.

2.3 Blockchain use cases

The financial sector and the world of cryptocurrencies are not the only ones that can benefit from Blockchain technology. Many sectors are seeing the potential of this technology, which, together with other technologies, can be of great use. In fact, 81 of the 100 top companies worldwide are already using Blockchain technology [9]. Blockchain has, therefore, many use cases, ranging from Digital Identities, Cloud Storage, and Smart Contracts. The latter is the one we are interested in in this paper.

2.4 Smart Contracts

A Smart Contract is an agreement between two parties in the form of computer code programmed to execute automatically. The Smart Contract is written in code and has the power to execute and enforce itself autonomously and automatically based on a series of pre-determined parameters. Smart contracts are executed on a Blockchain. The terms are stored in a distributed database and cannot be modified, reinforcing security, transparency, and trust among signatories, avoiding misunderstandings, falsifications, or alterations, and dispensing with intermediaries.



```solidity
pragma solidity 0.8.7;

contract VendingMachine {

    // Declare state variables of the contract
    address public owner;
    mapping (address => uint) public cupcakeBalances;

    // When 'VendingMachine' contract is deployed:
    // 1. set the deploying address as the owner of the contract
    // 2. set the deployed smart contract's cupcake balance to 100
    constructor() {
        owner = msg.sender;
        cupcakeBalances[address(this)] = 100;
    }

    // Allow the owner to increase the smart contract's cupcake balance
    function refill(uint amount) public {
        require(msg.sender == owner, "Only the owner can refill.");
        cupcakeBalances[address(this)] += amount;
    }

    // Allow anyone to purchase cupcakes
    function purchase(uint amount) public payable {
        require(msg.value >= amount * 1 ether, "You must pay at least 1 ETH per cupcake");
        require(cupcakeBalances[address(this)] >= amount, "Not enough cupcakes in stock to complete this purchase");
        cupcakeBalances[address(this)] -= amount;
        cupcakeBalances[msg.sender] += amount;
    }
}

```

Figure 1: Vending machine Smart Contract example.

3 Blockchain and Infrastructure as Code

3.1 Smart Contracts and Service Level Agreements

A service level agreement (SLA) is a contract that lists the different products or services offered by a supplier to a customer (internal or external) and sets out the requirements that the supplier must meet. The service provider commits to meet a series of requirements when providing the service with the SLA. These requirements are quantified within the service level agreement itself, usually through metrics that indicate a range in which the service must be established, a maximum that cannot be exceeded, or a minimum that cannot be lowered [4].

For example, a hosting service offered with an availability of 99.95% implies that it can only go down for 43 seconds per day to meet this condition.

SLAs are not a great solution. They are often hard to negotiate and can be costly to dispute in many cases. Furthermore, it is up to the customer to find an issue and then report it. The same service agreements and conditions can be stored in a Blockchain with the help of Smart Contracts. A Smart Contract would guarantee that if one or many conditions on the agreements are not met, it would be detected automatically and potentially reporting it to the vendor, thus, taking away the unnecessary manually handling of the issue. Furthermore, given that the Smart Contract is stored in a Blockchain, it inherits many of its properties such as Transparency, Security, and easy accessibility, making the Smart Contract available for anyone on the network to verify and audit.

3.2 Controlling code flow with Smart Contracts

One could record information about an applications state on a pipeline on the Blockchain instead of depending on tools like continuous integration servers and automated test suites.

It would increase software quality and reliability by making software distribution more transparent. Anyone with access to the Blockchain could see the status of application delivery and information about software issues in real-time, giving them more insight into the continuous delivery process.

Smart contracts might also be used to control the flow of code thru the pipeline. They could also make sure that the software meets user expectations as we saw can be done with automated SLAs.

Information can be independently and automatically checked by smart contracts on the Blockchain, rather than depending on tools inside the pipelines to catch issues such as continuous integration errors. If an issue arises, the smart contract can require it to be resolved before it spreads further down the pipeline and becomes a more significant issue [3].

3.3 Software Licensing with Smart Contracts

Software licenses are a contract between the author of a computer program and the users. They set out the terms, conditions, and clauses that must be met in order to use that software, and each user who downloads, installs, copies, or uses it must accept these conditions.

DevOps managers can use Smart Contracts to purchase software licenses. A Smart Contract can manage the interchange of the software key between the vendor and the manager. The contract will then ensure that both parties are satisfied, meaning that the manager gets the software key if they pay, and the contract will release the money to the vendor if they provide the key. The Smart Contract is automatically canceled based on conditions such as a predetermined time or after the contract is fulfilled. Since the Smart Contract is stored on a Blockchain, the code becomes immutable, and the contract cannot be interfered with by any of the parties.

3.4 Version Control

Version control, also known as revision control or source control, is the practice of managing and tracking changes to a source code repository. Version control tools, such as Git, help software teams manage changes to source code. They are quite useful for DevOps teams since they help reduce development time and deployment. [7]

Version control works by keeping track of modifications made to source code. By utilizing offshoots of the main source code called branches, developers can work in tandem and then merge together the work that was done in parallel. Should any errors or mistakes be made, developers can look back at earlier versions and revert any faulty errors made. Version control is ubiquitous in today's software development process, in particular for DevOps teams. So the question is not if a team should or should not use version control, the question is which version control tool. [11]

As version control keeps track of the history of source code, one can see the similarity between it and the digital ledger that is Blockchain technology. Thus, an implementation of Blockchain on, for example, Git seems reasonable. Gitchain is one such technology. Gitchain is a Layer 2 chain-agnostic (meaning it allows multiple chains) application state synchronization and syndication protocol based on Git. Layer 2 architecture means that not everything is written to the Blockchain. The reason for this is that writing to the Blockchain is time-consuming and expensive, and it is thus more efficient to write big amounts of data somewhere else. Gitchain consists mostly of two architectures, Distributed Storage and Distributed Ledger. If you have a large file, you cannot put it on the Blockchain. Instead, you can use Git's packfile. Packfile in Git protocol is similar to a zip file, in that it packs several files along with changes, metadata and descriptions into one file. You then put the packfile in the Distributed Storage, which could be a shared drive on the cloud, a peer-to-peer distributed file system like IPFS or some sort of Blockchain-driven storage system. Then, you would have to tell where this packfile is in order for someone to recreate the repository locally. That would be done by putting a record on the Distributed Ledger saying "whoever finds this link on the ledger can find the packfile, and recreate the repository". When someone sees the record, they can look up and download the packfile, unpack it and sync and recreate the repository. [6]

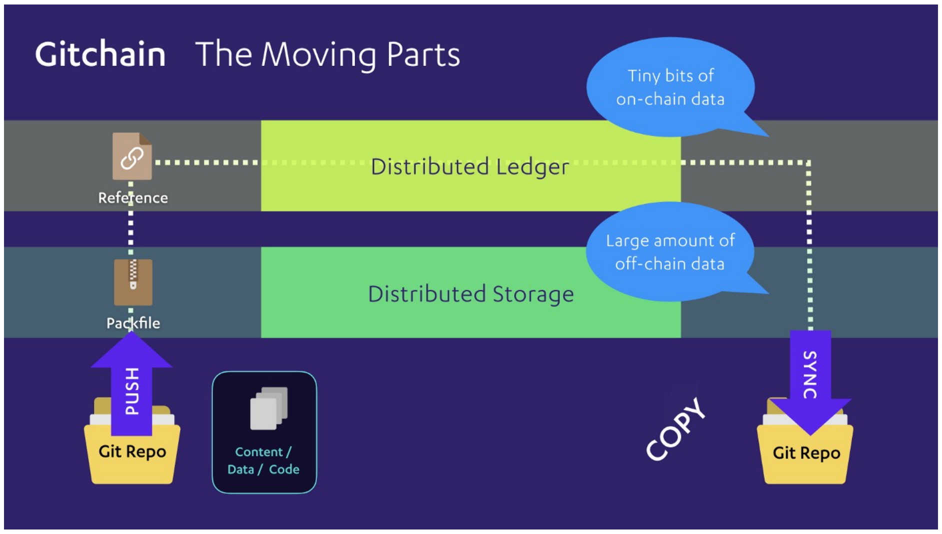


Figure 2: Gitchain infrastructure and workflow. [6]

These mechanisms of using established protocols like Git means that it is proven to work and so there is no need to reinvent something. However, using Blockchain technology with it allows developers to avoid a centralized infrastructure like GitHub, where remotes are stored.

Another attempt of a decentralized Git implementation is Mango. It is similar to Gitchain in that all Git objects (data and metadata) are stored on an IPFS, but there is an Ethereum smart contract that provides means for access control and stores pointers to the latest repository versions. Mango defines a Repository Interface which allows anyone to write a contract with custom access control or other features. [2]

Other efforts to decentralize Git are GitTorrent, in which Git objects are retrieved by the BitTorrent protocol, and Radicle [1]. Radicle, a decentralized code collaboration network, uses public key cryptography instead of user accounts to identify projects and collaborators [10].

3.4.1 Document version control

A group of researchers have proposed a Blockchain based solution and framework for document version control to facilitate collaboration and track changes without involvement of a centralized authority or third party. It is based on Ethereum smart contracts for the version control functions among developers and uses IPFS for storage of the documents, quite similar to Gitchain. The researchers demonstrate that their smart contract code is free of commonly known security vulnerabilities. [5]

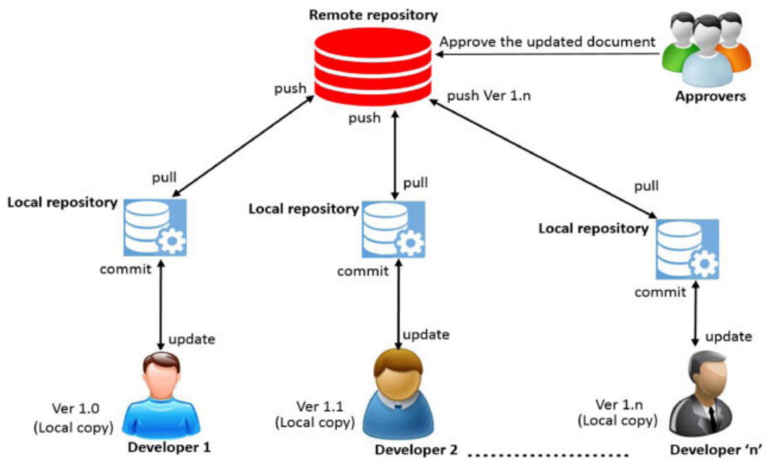
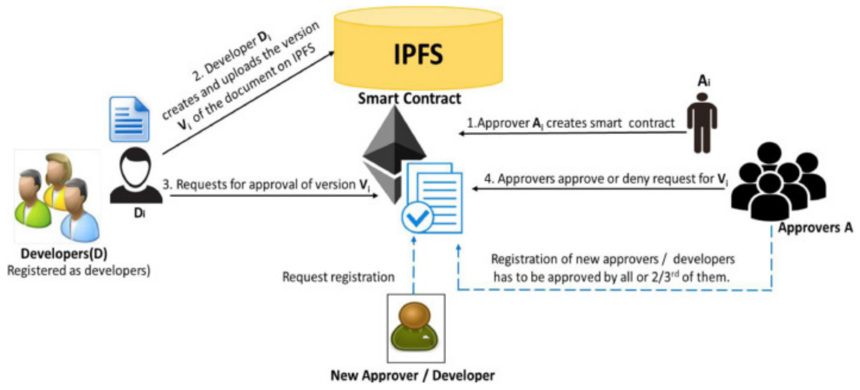


Figure 3: Traditional document version control systems. [5]



IPFS
Smart Contract
2. Developer D_i creates and uploads the version V_i of the document on IPFS
1.Approver A_i creates smart contract
A_i
3. Requests for approval of version V_i
4. Approvers approve or deny request for V_i
D_i
Developers(D)
Registered as developers)
Approvers A
Request registration
Registration of new approvers / developers has to be approved by all or 2/3rd of them.
New Approver / Developer

Figure 4: Ethereum smart contract-based system for controlled document sharing.[5]

4 Reflection & Conclusion

Blockchain and Smart Contracts are more than cryptocurrencies and NFTs, and we are optimistic about the underlying technology and its potential. We also understand that Blockchain is an upcoming technology and is therefore not so well established, thus making it harder to implement. The use cases for Smart contracts are many, and the area of DevOps is one of them. There are a lot of possible complications arising from current the centralized implementation of tools, such as Git, which Blockchain technology can help avoid.

However we realize that many of these use cases are either theoretical, not very established or used. Smart contracts for code flow and software licenses are only conceptual at the moment which means that they may not pan out as planned. Blockchain usage for Git exists, but is not very popular currently. The repositories have not been updated in years, which either suggests abandoned projects or moving of source code platforms (since they are trying to avoid using GitHub this might be likely case). Technology moves fast however, and we believe that at least some of these ideas might become widespread should developers find them useful.

References

[1] Chris Ball. *A decentralization of GitHub using BitTorrent and Bitcoin.* 2015. URL: https://github.com/cjb/GitTorrent.

[2] Alex Beregszaszi. *Mango: Git, completely decentralised.* 2016. URL: https://github.com/axic/mango/blob/master/MangoRepoInterface.sol.

[3] EY. In: *Blockchain in DevOps Implementing transparent continuous delivery* (2017). URL: https://assets.ey.com/content/dam/ey-sites/ey-com/en_ca/news/2017/10/ey-blockchain-in-devops.pdf.

[4] Gillian Cordall. *Service level agreements.* July 2019. URL: https://www.keystonelaw.com/keynotes/service-level-agreements.

[5] N. Nizamuddin et al. "Decentralized document version control using ethereum blockchain and IPFS". In: *Computers & Electrical Engineering* 76 (2019), pp. 183–197. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2019.03.014. URL: https://www.sciencedirect.com/science/article/pii/S0045790618333093.

[6] Chris Tse. *Introducing Gitchain.* 2019. URL: https://medium.com/cardstack/introducing-gitchain-add61790226e.

[7] Stephen Watts. *Source and Version Control in DevOps.* 2019. URL: https://www.bmc.com/blogs/devops-source-version-control/.

[8] RedHat. *What is infrastructure as code (IAC)?* Dec. 2020. URL: https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac.

[9] Lucas Schweiger. *81 of the top 100 public companies are using Blockchain technology.* Sept. 2021. URL: https://www.blockdata.tech/blog/general/81-of-the-top-100-public-companies-are-using-blockchain-technology.

[10] Radicle. *What is Radicle?* 2022. URL: https://docs.radicle.xyz/.

[11] Atlassian. *What is version control?* URL: https://www.atlassian.com/git/tutorials/what-is-version-control.

[12] IBM. *What is blockchain technology? - IBM Blockchain.* URL: https://www.ibm.com/topics/what-is-blockchain.