POLITECNICO DI TORINO

Politecnico di Torino

Master degree course in Computer Engineering

Master Degree Thesis

Building a Decetralized Architecture Merging Kubernetes and Blockchain Technology

Supervisor
prof. Danilo Bazzanella

Candidate
Tamietti Lorenzo
matricola: 290181

ACADEMIC YEAR 2022-2023

Summary

In today's ever-evolving digital landscape, the rise of decentralized architectures has revolutionized the way we interact with and perceive technology. Traditional centralized systems have faced inherent limitations in terms of trust, transparency, and resilience, prompting the exploration of alternative approaches that can address these challenges. This thesis embarks on an exploration of the fusion of two cutting-edge technologies: blockchain and Kubernetes, with the aim of creating a decentralized infrastructure that caters to the demands of Web3 applications.

The journey begins with an in-depth examination of blockchain technology, starting from its inception with the pioneering cryptocurrency, Bitcoin. Bitcoin introduced the concept of a decentralized ledger, providing a secure and transparent platform for peer-to-peer transactions without the need for intermediaries. Building upon the foundations laid by Bitcoin, the focus then shifts to the evolution of blockchain technology towards the emergence of Ethereum, often referred to as Blockchain 2.0. Ethereum introduced the concept of smart contracts, enabling the development of decentralized applications (dApps) that go beyond simple transactions, opening up new possibilities for decentralized systems in various industries. The exploration of Web3, the decentralized web, uncovers the transformative potential of blockchain technology, showcasing its ability to reshape our digital interactions and redefine trust and ownership in the online world.

While public blockchains have garnered significant attention, the thesis also delves into the realm of private blockchains and Distributed Ledger Technology (DLT). Private blockchains offer the advantages of blockchain technology within closed networks, catering to specific enterprise needs for data privacy and control. Open Ethereum is explored as a case study, providing insights into the development and functionalities of a public blockchain platform that is open to anyone to participate in.

Parallel to the investigation of blockchain, this thesis shines a spotlight on Kubernetes, a leading container orchestration framework widely adopted

for managing complex applications and microservices architectures. Kubernetes simplifies the deployment, scaling, and management of containerized applications, offering enhanced efficiency, scalability, and fault tolerance. By examining the components and inner workings of Kubernetes, a solid foundation is laid for understanding its role in managing distributed systems.

A key aspect of this thesis is contrasting the management of Kubernetes clusters with the orchestration of blockchain nodes. While Kubernetes excels in managing scalable and flexible containerized applications, the decentralized and consensus-driven nature of blockchain networks introduces unique challenges. By critically analyzing the differences and similarities between the management of these two technologies, insights are gained into the requirements and considerations necessary for merging them successfully.

The main objective of this thesis is to showcase how the fusion of blockchain and Kubernetes can lead to the creation of a decentralized infrastructure. By leveraging the benefits of both technologies, we aim to establish a dynamic network of nodes governed by blockchain consensus mechanisms, thereby eliminating central points of failure and enabling increased transparency and resilience. The culmination of this exploration will focus on illustrating how a distributed cloud environment can be realized, where node management is facilitated through blockchain governance. This groundbreaking approach has the potential to reshape traditional cloud infrastructures and pave the way for a new paradigm of decentralized computing.

Through this thesis, we strive to contribute to the understanding and advancement of decentralized architectures, uncovering their potential impact on the future of digital systems. By exploring the evolution of blockchain technology, the capabilities of Kubernetes, and their convergence, we aim to provide valuable insights and practical guidelines for architects and developers seeking to embrace the decentralized future of Web3 applications.

Acknowledgements

First of all, I would like to thank the Accenture blockchain team with whom I had the opportunity to do the internship which allowed me to broaden my knowledge of blockchain technologies and put into practice what I had previously seen as theoretical notions. Thanks to this experience I was able to acquire the knowledge and foundations on which the work of this thesis is based.In particular I would like to thank Francesco, who has been a guide for me during my internship and inspires me every day, passing on his passion and his interest in technology

Next, i would like to thank my supervisor Danilo Bazzanella, teacher of the blockchain course, who stimulated me to want to undertake this type of path and who provided me with the knowledge through which I was able to have a very detailed overview of blockchain technologies.

Finally, I would like to offer a special thanks to the people who have been close to me over the years. To my mother, Marina, who has always believed in my potential, even in the most difficult moments of my life. To my dad, Marcello who is a point of reference for me that I aspire to become, and who has helped me a lot in my studies. To my grandmother, Gabriella who has always shown her interest in me, always supporting me, and all the rest of my family. Finally I would like to thank my closest friends, in particular Roberto and Edoardo with whom I have shared my entire university career, between difficulties and successes, always supporting each other

Contents

List of Tables

List of Figures

Chapter 1

Introduction to Blockchain

1.1 Bitcoin overeview

In 2008 Satoshi Nakamoto published the white paper: "Bitcoin: A peer-to-peer electronic cash system" [1], where he introduced for the first time a distributed technology that allows users to make payments without having to go through a central institution. He therefore gave a first definition of blockchain that is a distribuited ledger or database where all transaction are written based on cryptographic proof instead of trust. A transaction represents the movement of a certain amount of money from one user to another and must be signed by owner private key. To achive security over each transaction we have to define three important cryptographic mechanisms:

- hash function: hash function is used to calculate digest, that is calculated over the content of transaction
- asymmetric cryptography: each account to be able to transfer money must have a key pair: a public key, used to another peer to send us money, and a private key, that is useful for authentication as a proof that we are the owners of money that we want to transfer
- digital signature: digest calculated with a secure hash function is encrypted with the sender of money's private key.

With those mechanism we obtain some important security goals that a payment system must have. First of all we obtain the non-repudiation property, since if a transaction is signed with our private key it means that there

is proof that we have authorized it. Secondly, is impossible for a malicious user to steal our private key starting from the public key,this is true until we use a strong asymetric cripthography alghoritm to generate the pair.

Once we have achived this security properties, we need a way to check if a certain amount of money can be spent by a peer, so if he effectively has enough money. The solution is that each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A node can verify the signatures to verify the chain of ownership. At the end we need to protect system in the case of double spent: the simple solution is take the first arrived transaction.

At this point, transaction will be boradcasted to all nodes and they could be collected inside a basic block, ready to be validated and inserted inside the ledger, but we have still resolve the problem of: how can decide the time validity of a block and how can we agree about which set of transactions has arrived before?.

1.2 timestamping

At this point the system need a way to give a time attribute to a certain transaction or a certain block, to protect system in case of double spending money and to give a proof that data written inside the ledger was written in a certain time.

In cybesecruity we define secure timestamp, that is the process of creating a tamper-evident record of the time at which a particular event or data was created, modified, or accessed. It is used to prove the existence and integrity of data at a specific point in time. In secure timestamping, a trusted third party (TTP) generates a unique cryptographic hash of the data being timestamped and appends a digital signature to the hash. This creates a tamper-evident record, as any modification of the original data will result in a different hash value and invalidate the digital signature.

So, in this system we can give proof that at the time of each transaction, the majority of nodes agreed it was the first received using timestamp

mechanism in this way. A timestamp server calculates the hash of the entire block proposed and he puts over this data the timestamp information. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

1.3 Consensus protocols

Once defined how a decentralized payment system based on cryptographic proof of validity and timestamp can ideally be built, the final question opened that must be solved is: how can trust be created by the users who choose to be part of this system? What we need is a consensus protocol.

A consensus protocol is a set of rules and procedures that enable a distributed system to reach agreement on a single data value or a set of values across multiple nodes or participants in the network. It is a fundamental building block for distributed systems such as blockchains, where multiple nodes need to agree on the state of the system. The protocol ensures that all nodes agree on the order and validity of these entries or transactions, even in the presence of faulty or malicious nodes.

1.4 Proof of work

Proof of Work (PoW) is a consensus algorithm used in blockchain networks to validate transactions and create new blocks. It is a computationally intensive process that requires a significant amount of computing power to be expended to solve a complex mathematical problem, in order to prove that a certain amount of work has been done.

In PoW, a cryptographic hash function is used to generate a block header. This block header includes the hash of the previous block, a timestamp, and a nonce. A node must modify the nonce value in the block header until they find a hash that meets a specific difficulty requirement, which is determined by the network. This difficulty requirement is usually expressed as a certain number of leading zeros in the hash.

To find a valid hash, nodes use their computational power to perform numerous hash calculations until they find a hash that meets the required

difficulty. Once a miner finds a valid hash, they broadcast it to the network, along with the block header and the transactions included in the block. Other nodes in the network can verify the hash by re-performing the same hash function on the block header and checking that it meets the difficulty requirement.

1.5 building the chain

Node that wins challenge of proof of work is now able to insert the new block inside the distributed ledger. Therefore, the blockchain is a distributed database, formed by a chain of basic blocks, where each block is mathematically linked to the previous block. In this way we have added a further proof of integrity on the network, as it is not possible to modify the past state of the blockchain, as everything that is entered subsequently would be invalidated

Chapter 2

Addressing Scalability and privacy in public Blockchains

2.1 Web3 introduction

The internet has evolved over the years [2], and its different stages have brought about unique characteristics that have shaped the way we use the internet today. The first stage,Web 1.0: Read Only (1990-2004), began in the early 1990s and was characterized by static and basic websites that were mostly text-based. Initially, the Internet was thought of as a simple means of communication between users in different locations. In fact, the first generation web represented so-called "showcase" websites where static content was displayed without possible interaction with the end user. Web 1.0 was a one-way communication, where webmasters created and updated website content, and users were passive consumers who only read and viewed the content.

The second stage, Web 2.0: reading and writing (2004-present), began in the early 2000s and is considered the era of interactive and dynamic websites. Web 2.0 allowed users to interact, collaborate and share user-generated content through social media platforms and blogs, leading to increased communication, and a shift towards a more user-centered culture. The Web 2.0 paradigm was basically the client-server paradigm, where the client interacts

with an always-on machine that provides the service. The key to the client-server paradigm is the centralization of resources, this is because different peers who want to read content on the web establish a connection with the machine that holds the resources. The problem with this pattern is of course the fact that assets, such as data that are shared with by users, or any other digital assets are managed and processed by a central entity.

With Bitcoin and the emergence of the blockchain, there then arises the need to want to create an ecosystem that utlizes resources that are in the hands of everyone, where then any web content or digital assets are managed by the individuals who populate the web, and no longer by central entities. In general, the concept of decentralization posits the goal of transforming the ownership of resources into the hands of the users who populate the system. Web3[**?**] refers to the next generation of the internet or the "decentralized web." Unlike the current centralized internet, where information is controlled and stored on servers owned by companies, Web3 relies on blockchain technology to provide a decentralized platform that is more transparent, secure, and democratic. It aims to give back control of data and identity to users, allowing them to have ownership and control over their data.

These are the key concepts under Web3:

- blockchain technology at the core
- digital assets:by digital asset we mean any resource registered in the blockchain, in which there is no ambiguity about possession and origin, like token, or NFT
- smart contract, code that acts as a bridge for users' input, creation and management of resources, on the blockchain
- Decentralized applications (dApps): are software applications that are built on top of blockchain technology and use smart contract at the core of programming logic
- decentralized identity management:every user can log in inside application using his wallet, instead of using a centralized sign-on method wich implies that the application maneges credentials of users.

2.2 Limits of Bitcoin

Bitcoin was build with the main purpose of create a distibuted system for payments and for this reason is limited. For its nature was not create with purpose of build decentralized applications over this architecture, however, is possible to write a code over this, but Bitcoin's scripting language is also limited, which means that it cannot support complex code becouse is not Turing-complete programming language. While Bitcoin's scripting language is limited, it still enables basic programs, such as multisignature transactions, which can be used for more complex use cases.

One of the other significant limits is its proof-of-work consensus mechanism. PoW requires miners to solve a complex mathematical problem to validate transactions and add them to the blockchain. As more miners join the network, the computational power required to solve these problems increases, leading to a corresponding increase in energy consumption. This energy consumption has raised concerns about Bitcoin's environmental impact and long-term sustainability. Additionally, proof-of-work limits the number of transactions that can be processed in a given amount of time, leading to a scaling problem. This scaling issue has led to the development of alternative consensus mechanisms that we're going to explore in next chapters.

Another limit of Bitcoin is its high transaction fees. The high fees make it less attractive for small transactions. Bitcoin's transaction fees are driven by market demand and supply, and they can fluctuate wildly depending on the network's congestion level.

2.3 Ethereum over Bitcoin

Ethereum [3] was created with the premise of overcoming the limitations of bitcoin in terms of developing so-called inteligent programs. The goal of this chain is basically to be used to develop decentralized applications, dApps, providing developers with a complete programming language with which they can implement any logic.

Why is bitcoin not turing complete? In bitcoin we have a UTXO model, where each amount of money is locked by a locking SCRIPT. This script is

essentially a chain of operations that if executed correctly allow the money to be unlocked and allowed to be spent by a user. It is a procedural language, which does not allow loops, this is because an infinite loop would risk compromising the operation of the network. Ethereum, to solve this problem creates the concept of GAS: any transaction to be executed requires the payment of a certain amount of gas by the peer asking to execute it. Then the user indicates how much gas at most he intends to pay for the execution of a transaction or portion of code, and if the gas runs out, the code is no longer executed.

2.4 EVM and smart contract

In Ethereum, smart contracts are programmable agreements that enforce predefined rules and conditions, eliminating the need for intermediaries and enhancing trust and transparency in digital interactions.

At the heart of Ethereum's smart contract functionality lies the Ethereum Virtual Machine (EVM), a Turing-complete virtual machine that executes the bytecode of smart contracts. The EVM serves as the runtime environment for executing smart contracts on every participating node within the Ethereum network. It ensures consistency in contract execution across the decentralized network by enforcing consensus on contract outcomes.

The EVM operates through a stack-based architecture, where each operation involves manipulating items on a stack. These items can be data values, memory pointers, or program counters. The EVM bytecode consists of a series of instructions that perform operations such as arithmetic calculations, logical operations, and data storage.

When a transaction is sent to the Ethereum network to execute a smart contract, the EVM interprets the bytecode and executes the instructions step-by-step. The EVM maintains a deterministic execution model, meaning that given the same input and state, the contract execution will produce the same output. This determinism ensures that contract execution is predictable and replicable across all participating nodes in the network.

The EVM also provides a secure environment for executing smart contracts by enforcing several built-in constraints. Gas, a unit of computational effort, is used as a measure of resource consumption. Each operation in the bytecode consumes a certain amount of gas, and users must provide sufficient gas to cover the computational costs of contract execution. Gas incentivizes efficient

and optimized contract code, as excessive resource consumption can result in transactions being reverted due to insufficient gas.

Once a smart contract is written and compiled, after being deployed, the code will be publicly visible through a public address. This introduces a very important factor, transparency. A decentralized application that relies on smart contracts as a rationale provides trust and transparena, since each user can verify the code that operations do, eliminating any possible ambiguity. This is not possible in proprietary sofwtare, where we see a product whose code remains obscure as well as data manipulation.

2.5 Problems of public blockchain in enterprise use cases

Although ethereum succeeds in overcoming many of bitcoin's shortcomings, providing the ability to develop decentralized applications via smart contracts, and currently uses a proof-of-stake consensus protocol, increasing scalability and reducing the computational waste of block insertion, it continues to have some problems that make it difficult to develop large-scale applications, and be chosen. those problems are in common to all public blockchain and we need to consider them before we choose the more convenient architecture.

- Scalability: Public blockchains like Ethereum face significant challenges in terms of scalability. As every node must validate and record every transaction, the overall network throughput is limited. This can result in longer transaction confirmation times and higher transaction costs when the network is congested.

- Costs: Using public blockchains like Ethereum can entail significant costs. Transactions on the blockchain require the payment of fees in cryptocurrency, known as "gas," which can increase significantly during network congestion. Additionally, registering data on a public blockchain may require higher computational and storage resources compared to other centralized solutions.

- Data Privacy and Security: Public blockchains are inherently transparent, allowing all participants to view the complete transaction history and data stored on the blockchain. This can raise concerns regarding

the privacy and security of sensitive or corporate data, especially in enterprise contexts where information confidentiality is crucial.

- Governance and Flexibility: Decisions regarding rules and changes to public blockchains are made through a consensus process among participants, which can be complex and time-consuming. This decentralized governance can limit the flexibility of public blockchains to quickly adapt to the needs of an organization or a specific application.

Today, however, we have many alternatives to etherum such as blockchain to develop applications that rest on a more scalable and less expensive infrastructure. First and foremost are layer 2 solutions, such as Polygon, which allow transactions to be executed on a secondary blockchain to ument efficiency, allowing roll-back of data to layer 1 ethereum. Or algorand, a layer1 blockchain like ethereum that makes scalability its strength. The problems mentioned before, however, remain and are inherently related in public architectures . There are two ways to solve these problems.

The simpler one is to move toward private blockchains, so you can give the company an administrable product, on which you can manage security and visibility over the data, implementing your own access policies. Obviously, private blockchains have no scalability problems since the consensus protocols are either dummy or much less onerous. Second, with a private blockchain we do not have the problem of having to pay for transactions since the validator nodes are administered by the company and therefore there is no need to remunerate them for the validation work. In the next chapters we will introduce private blokchains as a first solution to solve the problems mentioned above, then we will look at some innovative solutions for public blockchains

2.6 Permissioned blockchain and DLT overview

Although the term "blockchain" is often associated with cryptocurrencies like Bitcoin and Ethereum, it encompasses a wider range of applications and use cases. most businesses don't care too much about cryptocurrencies, what's far more significant for enterprises is the technology behind them. Blockchains can be categorized into two main types: permissionless (public) and permissioned (private) blockchains. This introduction aims to provide a

comprehensive introduction to permissioned, private blockchains, highlighting their key differences from permissionless blockchains and the purposes they serve in various industries.

What are Permissioned, Private Blockchains? Permissioned, private blockchains are a type of distributed ledger technology (DLT) that restricts access and participation in the network to only authorized users. These blockchains are typically operated by a single organization or a consortium of organizations that have the authority to grant and revoke access to the network. Unlike public blockchains, where anyone can join and participate without restrictions, private blockchains require an invitation and validation by the network administrators. These are the Key Differences from Permissionless Blockchains

- Access Control: The most notable difference between permissioned and permissionless blockchains is access control. In permissioned blockchains, only authorized participants can read, write, and validate transactions. In contrast, permissionless blockchains allow any user to join the network and participate in the consensus process, without the need for prior authorization.

- Consensus Mechanism: Permissioned blockchains often use different consensus mechanisms than their permissionless counterparts. While permissionless blockchains rely on resource-intensive mechanisms like Proof of Work (PoW) or Proof of Stake (PoS), permissioned blockchains may use more efficient and scalable mechanisms such as Practical Byzantine Fault Tolerance (PBFT), Raft, or Tendermint.

- Security and Privacy: Permissioned blockchains provide enhanced security and privacy due to the restricted access and controlled environment. Transactions and data stored in a private blockchain are only visible to authorized participants, ensuring confidentiality and protection against unauthorized access.

- Scalability and Performance: Permissioned blockchains can handle a higher transaction throughput and offer faster confirmation times compared to permissionless blockchains. This is mainly because of the controlled environment, reduced number of nodes, and the use of more efficient consensus mechanisms.

- Governance: In a permissioned blockchain, the network's governance is usually centralized to a certain extent, with network administrators or a consortium of organizations having control over decision-making, rule enforcement, and protocol updates. On the other hand, permissionless blockchains have a decentralized governance structure, with decisions typically reached through community consensus.

- DLT : decentralized ledger technology

In the context of enterprise and public blockchains, rather than talking about blockchain, it is more appropriate to talk about DLT.

A DLT (decentralized ledger technology) is the decentralized architecture that relies on a common database, precisely a ledger that is at a higher level and is not supervised and accessed by a single entity.

In a DLT, the ledger is made accessible only to those who have the permissions to do so.

A Decentralized database is so the hearth of this architecture. A blockchain ledger is often described as decentralized because it is replicated across many network participants, each of whom collaborate in its maintenance.

2.7 blockchain trilemma and layer2

The Blockchain Trilemma, also known as the DLT (Distributed Ledger Technology) Trilemma, is a term coined by Ethereum co-founder Vitalik Buterin. It describes the challenge of achieving all three primary attributes of a blockchain - scalability, security, and decentralization - simultaneously. The Trilemma suggests that at most, only two of these characteristics can be fully achieved at any one time.

this problem indicates that it is not possible, for example, to have high scalability and security without losing decentralization. For example, the small size of the blocks is designed this way so that validation does not remain in the hands of only those with so much computing power, risking centralization of control of data input. Increasing the size of the blocks by inserting more transactions allows yes to gain in scalability but losing in decentralization

A Layer 2 blockchain solution, such as Polygon, is designed to address two of the biggest challenges faced by Layer 1 blockchains like Ethereum - scalability and high transaction costs.

The main principle behind Layer 2 solutions is to take most of the workload off the main chain (Layer 1) by handling transactions and smart contracts off-chain, and then batch these results into a single transaction that is recorded on the main chain. This method significantly increases transaction throughput and speed, while also reducing associated costs.

Polygon (formerly known as Matic Network) is a well-known Layer 2 scaling solution for Ethereum. It stands out from other Layer 2 solutions by providing a framework for building and connecting Ethereum-compatible blockchain networks. This "internet of blockchains" is able to communicate with one another due to their shared Ethereum compatibility.

2.8 Polygon and zkEVM

Polygon zkEVM [4], henceforth zkEVM, is a virtual machine designed and developed to emulate the Ethereum Virtual Machine (EVM) by recreating all existing EVM opcodes for transparent deployment of existing Ethereum smart contracts. Zero-knowledge Rollups (ZK-Rollups)run on top of the Ethereum Mainnet and exponentially improve the scalability and transactions per second (TPS) of Ethereum. In order to prove that the off-chain computations are correct, Polygon zkEVM employs verifiable zero-knowledge proofs as validity proofs. Although the Layer 2 zero-knowledge proofs are based on complex polynomial computations to provide validation and finality to off-chain transactions, the validity proofs are quick and easy to verify. We will discuss ZKP in more detail later, for the time being it is basic for us to know that it is a cryptography-based technology that allows us to provide proofs of correctness. As a state machine, zkEVM carries out state changes, which come from executions of Ethereum's Layer 2 transactions that users send to the network, and subsequently produces validity proofs attesting to the correctness of the state change computations carried out off-chain.

2.9 ZK rollups

zkRollups are a Layer 2 scaling solution for blockchains like Ethereum, aimed at increasing transaction speed and reducing costs. The "zk" in zkRollups stands for "zero-knowledge," which is a type of proof used in cryptography.

Here's a simplified explanation of how they work:

- Batching Transactions: zkRollups bundle or "roll up" many transactions

into a single one. It's like taking many individual letters (transactions) and putting them in a large envelope (the rollup) to be sent all at once. This batch is then committed to the main blockchain.

- Data Availability and Computation Off-chain: In zkRollups, all transaction data is stored on-chain, but the computation and state transition happen off-chain. This means the details of the transactions and their state transitions (e.g., account balance changes) are processed outside of the main Ethereum network, greatly reducing the computational load on the main chain.

- Zero-Knowledge Proofs: For each batch of transactions, a zero-knowledge proof, also known as a zk-SNARK ("Zero-Knowledge Succinct Non-Interactive Argument of Knowledge"), is generated and verified on-chain. This proof attests to the validity of all transactions in the batch without revealing their details, hence the term "zero knowledge". Essentially, it proves that no rules of the network were violated, such as creating extra tokens out of thin air or spending coins you don't own.

- Security: zkRollups inherit the security of the underlying Layer 1 blockchain (like Ethereum). Even if the zkRollup operator goes offline or behaves maliciously, the funds are safe as all the data necessary to resume operations is available on-chain, and anyone can reconstruct the state and continue processing transactions.

2.10 Zero knowledge proof(ZKP)

In the previous section, we presented a scalability solution that would allow one to move beyond the performance and cost constraints that may limit a company in its choice of obtare to build a decentralized application. Through layer 2 mechanisms we then have the ability to have security and decentralization, but still have high performance and low cost. The fundamental problem remains inherent in a public blockchain architecture, namely privacy and access mechanisms. The data written on the blockchain are public and visible to all, so this is a major limitation in case we want to create a mechanisim of access control to resources, thus writing decentralized and public logics without, however, making the secrets and identities of the users enabled to a given ecosystem disnibile to all. Zero knowledge proof mechanisms represent a breakthrough in this regard, allowing one to write knowledge proof logics

sneza having to write them publicly.

Zero-Knowledge Proofs (ZKPs) are a revolutionary concept in cryptography that allows one entity (the prover) to prove to another entity (the verifier) that a certain statement is true, without revealing any additional information beyond the fact that the statement is actually true.

At a very high level, ZKPs are based on advanced mathematical and logical principles. The fundamental idea is to create a "proof" that can be verified, without it revealing any details about the knowledge or data that were used to construct the proof. This is done through complex algebraic manipulations that create a series of equations and inequalities, whose solutions can be verified without knowing the original values used to create the equations.

2.11 ZKP and blockchain

The application of Zero-Knowledge Proofs (ZKP) to the blockchain has the potential to resolve a fundamental dilemma in its design: the tension between transparency and privacy. A public blockchain, by definition, maintains a transparent and immutable record of all transactions, accessible to all participants in the network. This makes transactions easily auditable and verifiable; however, it can compromise user privacy and the sensitive information contained in transactions.

ZKPs can be integrated into the blockchain to ensure that transactions are valid without revealing specific details about the transactions. For example, a smart contract on the blockchain could use ZKPs to verify that a user has enough tokens to make a transaction, without revealing the exact balance in their account. This allows the network to verify and record the transaction, while preserving the user's privacy.

The real potential of ZKPs in blockchain emerges in the context of decentralized applications that require a certain level of privacy. For example, an access rights management system could be built where users can prove their right to access a particular file or data, without revealing their identity or access details. This type of system, previously only possible on private blockchains, could now be built on public blockchains, offering the security guarantees and censorship resistance of public blockchains, without compromising user privacy.

In summary, the integration of ZKP into blockchain enables the creation of secure and private decentralized systems, overcoming the inherent limitations

of public blockchains and paving the way for a wide range of new applications.

Chapter 3

Building blockchain architecture with kubernetes

3.1 introduction

In this chapter we will see how to create an EVM- compatible blockchain, using Open ethereum. The blockchain will have a PoA(proof-of-authority) configuration, meaning we will have that validation will take place by selecting one of the validator nodes included in the configuration list. We will then see the creation of the network specification, and then go on to pull up the nodes. The goal is to pull up a network, entirely through kubernetes, thus being able to scale directly on the creation of the nodes and write configurations that will be placed within the nodes as default volumes. To do this we will use Open ethereum, which is open source software, directly downloadable from docker Hub. Once we have downloaded the base image, we can create containers inside of which we will place the standard node configurations, that is, the accounts, the genesis file, and the parity.toml in which we place the various node specifications. For example, K8s allows us to write the configurations and pull up n nodes within which we directly insert the same genesis file and configurations in a simplified and automated manner.

3.2 Kubernetes (K8s) and blockchain: Convergences in a decentralized context

Kubernetes (K8s) and blockchain networks, while intended for different uses, share many key structural characteristics. Both are composed of a collection of machines, or nodes, that work together to create a distributed environment. In this chapter, we will explore how these two technologies complement and complement each other.

A Kubernetes cluster is a set of nodes that work together to provide a flexible and scalable environment for running containerized applications. This decentralized architecture model closely resembles the structure of a blockchain network, where each node participates in the overall functionality of the network by contributing its own copy of the distributed ledger.

In the context of Kubernetes, each node runs a portion of the application, and the cluster as a whole ensures that the application is always available and responsive, despite individual node failures. This resilience is also reflected in blockchain networks, where the loss of a single node does not compromise the overall functionality of the network.

Importantly, though, while blockchain networks focus on decentralized data storage and consensus mechanisms, Kubernetes provides an infrastructure for managing, deploying, and scaling containerized applications. This brings significant advantages, such as high availability, fault tolerance and scalability, characteristics also highly appreciated in blockchain networks.

In the context of a Kubernetes cluster, each node can have a different role. For example, master nodes are responsible for maintaining the desired state of the cluster, while worker nodes run applications. This parallels the differentiation of nodes in blockchain networks, with full nodes retaining the entire history of the blockchain and lightweight nodes or "SPVs" retaining only the necessary parts.

Finally, the synergy between Kubernetes and blockchain can be leveraged to create more secure and decentralized infrastructures. A Kubernetes cluster can manage and orchestrate a network of blockchain nodes, while a blockchain network could be used within a Kubernetes cluster for distributed configuration and service discovery. By integrating these two technologies, it is possible to combine the advantages of both, creating highly resilient, decentralized and distributed systems.

3.3 Orchestrating Blockchain Nodes with Kubernetes

Kubernetes (K8s), thanks to its flexibility and scalability, can represent the ideal environment to host and manage a blockchain network. In this chapter, we will dive into how this can happen and the potential benefits.

Blockchain nodes, similar to other applications, can be containerized and managed within a Kubernetes cluster. In this scenario, each container would represent a blockchain node and include the necessary blockchain software and dependencies. Blockchain distributed ledger data can be held in persistent volumes provided by Kubernetes, ensuring that data is not lost when containers are moved or replaced.

Kubernetes takes care of managing the blockchain nodes. Thanks to its declarative configuration, it is possible to specify the desired state of the blockchain network, for example the number of nodes, and Kubernetes takes care of maintaining this state. In the event that a node fails, Kubernetes can automatically replace it with a new one, ensuring high availability and minimizing downtime.

Another significant benefit is scalability. As the needs of the blockchain network grow, Kubernetes can automatically scale the number of blockchain nodes as needed. This allows the network to efficiently handle variable workloads, ensuring optimal use of resources.

Networking within Kubernetes can be configured to support peer-to-peer communication, which is central to a blockchain network. The service discovery capabilities of Kubernetes enable each node to find and communicate with its peers.

Additionally, Kubernetes namespaces can be used to isolate different blockchain networks if needed. This allows you to run several separate blockchain networks on the same Kubernetes cluster, each with its own dedicated resources and isolated from the others.

As far as security is concerned, Kubernetes offers several features that can be used to secure the blockchain network. These include secret management for storing sensitive data such as cryptographic keys, role-based access control (RBAC) to control access to Kubernetes resources, and network policies to control access across the network and between blockchain nodes.

3.4 Creation of genesis block of PoA blockchain

A genesis file is a critical configuration file for starting a blockchain. It contains the initial parameters that define the rules, consensus parameters and starting accounts for the blockchain. The genesis file is used to create the first block, known as the "genesis block," which establishes the initial state of the chain.

The genesis file is used to:

- Define configuration parameters: Specify the key parameters of the blockchain, such as the consensus algorithm, block size, gas limit and other specific parameters.

- Configure initial accounts: Defines accounts with their initial balances for the genesis block. These accounts can represent validators, participants or specific services on the blockchain.

- Establish consensus rules: Specifies the consensus rules and algorithms that determine how blocks are created and validated on the blockchain.

- Initialize blockchain state: Provides an initial state for the blockchain, including account balances, default functions and other specific details.

In our case we are going to configure a chain that uses proof-of-authority consensus protocol, which is a mechanism that sees a list of accounts as predefined validators. The list of validators is included directly in the genesis file in which we enter the specification.

```
genesis.json: |-
  {
    "name": "gChain",
    "engine": {
      "authorityRound": {
        "params": {
          "stepDuration": "5",
          "validators": {
            "list": [
              "0x24B2aD989ad4E2903b1471d1B49Fa3F8a2f24980",
              "0xc8316e34430E116550c7AdcE812b8CD95e00BC9C",
              "0xD28aE2fF6a6E4074e75B127031B6Ad4fd68A63ba"
            ]
          }
        }
      }
    },
          You, now • Uncommitted changes
    "genesis": {
      "seal": {
        "authorityRound": {
          "step": "0x0",
          "signature": "0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
        }
      },
      "difficulty": "0x20000",
      "gasLimit": "0x5F5E100"
    },
```

Figure 3.1. Genesis file configuration

3.5 creation of a node configuration

Once the genesis json is created, we are ready to write the generic node configuration. We assume that we have created via the open ethereum api three basic accounts, which we will use as validators for proof of authority. So the goal is to pull up three nodes, one per validator, where each node will have transaction signer address equal to the validator account we created earlier.

Within this configuration file we notice a few important things. First, we indicate in the chain variable where the path within the container is where we will find the genesis block specifications. In the network part we find some configurations, such as the port that the node will expose to be accessible by other nodes(json RPC port). We then find the specification of the chain ID that must be common to all nodes. Finally we find some specifications such as the engine signer, which would be the account associated with our validator.

```
parity.toml: |-
  [parity]
  chain = "/config/genesis.json"
  base_path = "/data"
  keys_path = "/keys"
  identity = "validator-1"

  [network]
  id = 1000
  interface = "all"
  reserved_only = true


  [rpc]
  port = 8545

  [account]
  password = ["/config/node.pwd"]

  [mining]
  engine_signer = "0x24B2aD989ad4E2903b1471d1B49Fa3F8a2f24980"
  gas_cap = "0x5F5E100"
  tx_gas_limit = "0x5F5E100"
  gas_floor_target = "0x5F5E100"
```

Figure 3.2. parity node configuration

3.6 Building the k8s infrastructure

Once we have seen how to create the genesis file and the node confuration file, we can write three different node configuration files for the three validators and pull up the nodes. We create the following k8s components:

- Config Maps:The config map is a basic component of k8s in which we can write static data that can be imported within a container, in this case our nodes. So we write 1 config map with the validator accounts and the genesis. json inside. Next 3 configMaps, one per node in which we will have written the node specifications seen earlier as parity.toml
- Deployments: We will have a deployment in which we define the container we are going to create. The base image will be that of Open ethereum downloaded from Docker Hub, and as volumes we will insert the data present in the config Map, i.e. genesis file and the specific parity.toml for each node.
- Services:We're going to create a service that maps external requests to our nodes, then we'll map the requests to the JSON rpc ports of our validator nodes created with the deployments
- Persistent volume claims: We declare volumes for our nodes, i.e. for the persistence part, indicating a storage space in which we will insert the configuration data

Once all the listed components have been created we are ready to apply our configuration and create our private infrastructure within the k8s cluster.

3.7 Advantages of this architecture

In this chapter we have therefore created a decentralized architecture with the blockchain at the base. The architecture we created solves many of the problems mentioned in the previous chapter, including scalability and privacy. This architecture would allow a company that adopts it to have control over data access methodologies, being able to choose which type of policies it wants to apply. The solution, although it is a blockchain solution that can be considered private, but it can be said that it is a hybrid, as it allows some people to manage the governance and operation, but opens up the possibility of making access to nodes even from external users. This solution could be a fair compromise of the reasons why companies are hesitant about the idea of wanting to build their business on a public blockchain, without however having to build a totally private architecture that would take some time to find. Once we have this type of architecture in hand, it can be used to create any web application 3, leaving the company with the ability to manage access to resources as it sees fit. The advantages are many:

- Control and Privacy: A semi-private blockchain allows you to maintain control over who can participate in the network and what transactions they can view. This is especially useful for organizations that want to benefit from the decentralization and security of the blockchain, but also need privacy and control over their data.

- Speed and Efficiency: Unlike public blockchains like Ethereum, which must process all transactions publicly and openly, a semi-private blockchain can be much more efficient. Because you have control over who can participate in the network, transactions can be processed faster and with fewer resources.

- Customization: With Open Ethereum, you have the freedom to customize your blockchain rules to fit your specific needs. For example, you could implement a specific access policy, custom rules for transactions, or even create your own token.

- Restricted Access: While a public blockchain allows anyone to participate, a semi-private one can restrict access to only authorized participants. This can be useful for preventing malicious attacks, limiting the ability to manipulate the ledger, or keeping the network more stable.

- Selective Transparency: While the semi-private blockchain can provide privacy and security, it can also provide selective transparency. For example, you may decide to make certain information available to the public or to specific users, while retaining control over who can transact or access more sensitive data.

- Interoperability: The semi-private blockchain can interact with other blockchain networks, public or private, thus enabling a wide range of applications and collaborations between different blockchain ecosystems

3.8 Creation of dynamic cluster with blockchain

We have previously seen how to create a semi-private blockchain, and manage its operation through k8s. Now let's put ourselves under another perspective: what if instead we used the blockchain to control the access of new participants to a cluster of k8s nodes in a decentralized and distributed way? The idea is to create a cluster of nodes, which provide their computational power and are part of this set capable of generating a certain computational power, on which computational capabilities can be distributed based on availability and needs. Now let's see the pessages that can be done to implement this thing. First, k8s provides an object called kubeadm, which represents an access token generated by the cluster of nodes. When a certain node wants to join, it presents this access token to the cluster which accepts its participation. We would like to try to automate this through the benefits of the blockchain, thus making this centralized architecture more distributed by implementing smart contract access control.

phase1: setup cluster and blockchain

Phase one can be said to be what we obtained in the previous step, i.e. an architecture with a blockchain, but with single administration, i.e. only one executive node belongs to the cluster we have created, we therefore want to study how we can add nodes dynamically without being in a centralized and single-control mechanism such as the kubeadm, let's try to automate it with the newly created blokchain

phase 2:The new user who wants to join the cluster first generates a valid account, i.e. a private/public key pair with which he can authenticate and sign transactions. Let's suppose that to do this the cluster publicly exposes a small single sign on service, to which the user can access and register, providing his public account.

phase 3:In the cluster, access and sending of access tokens is managed by a smart contract, which keeps track of the various public addresses of users, saving for each user the access token encrypted with his private key.

phase 4: Once the access request is received, the cluster then sends the kubeadm access token encrypted with the user key to the smart contract. Using his public key, the user can then access one of the blockchain nodes exposed by the cluster and request his encrypted kubeadmin.

phase 5:The user generates the cluster access request by presenting the newly obtained token and becomes part of the cluster

In this context we hypothesized to start with the experiment with two PCs, one personal on which to create the initial architecture and one on which to experiment with the entrance, the corporate one. We have foreseen in the architecture the possibility of having distributed storage, since the blockchain due to its properties is better than managing only the access and writing policies of the data but without saving them.

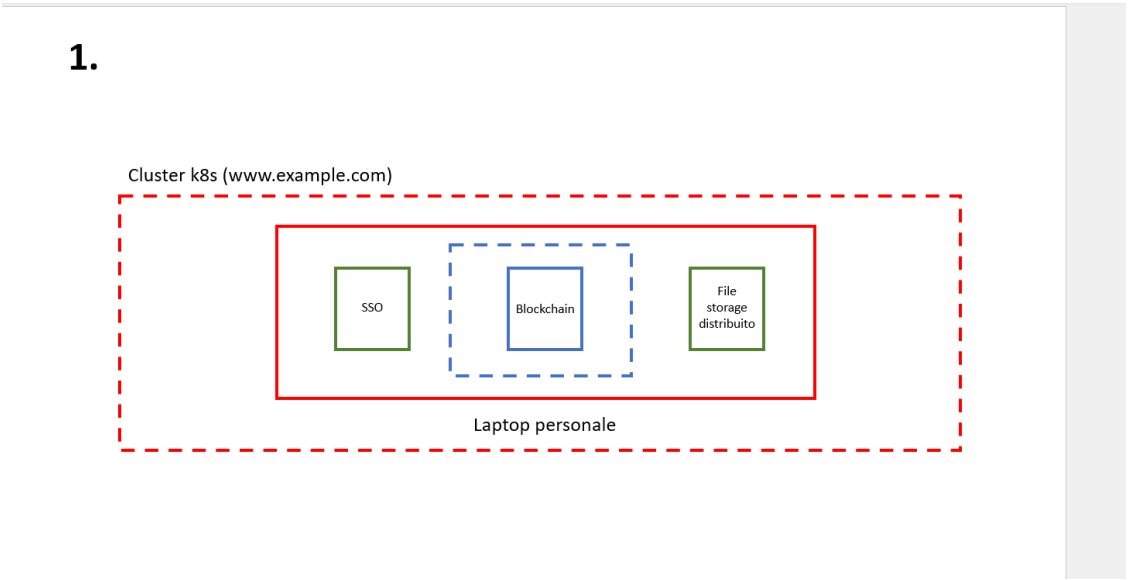

Figure 3.3. beginning phase

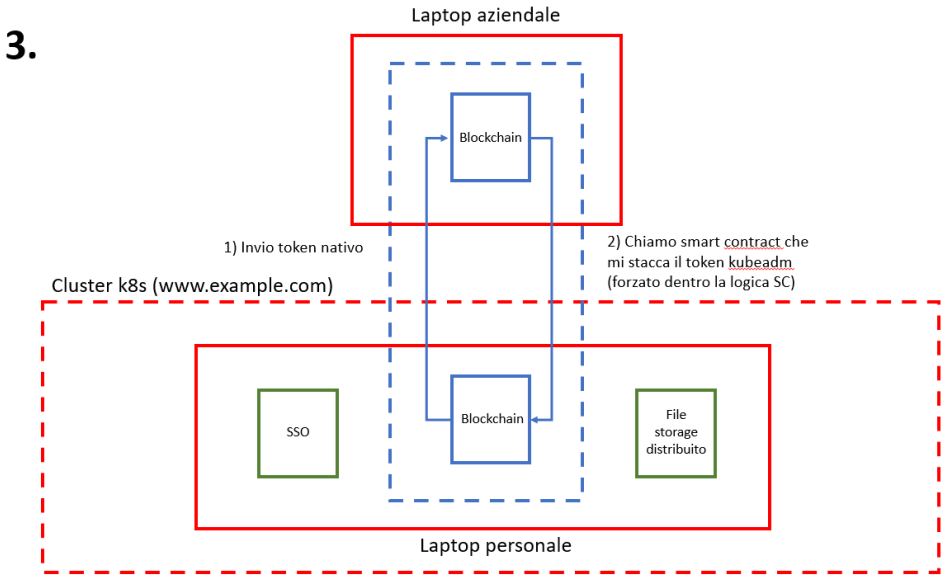

Figure 3.4. intermidiete phase

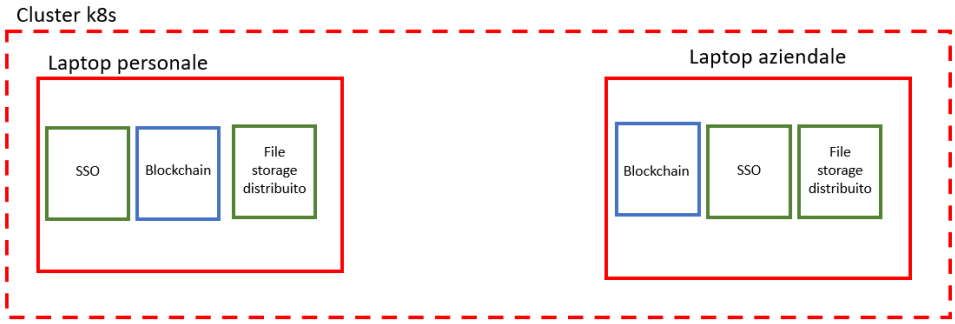

Figure 3.5. final phase

3.9 Conclusions

The technologies distributed on the internet are of different types and solve different problems. k8s manages distributed workload efficiently, blockchain guarantees security, interoperability, immutability. The union of their strengths makes it possible to build a higher-level distributed system in which the security part (k8s control plane) is managed via blockchain while the computationally intensive workload part is distributed via k8s. The unification of technologies makes it possible to create an open source aws like cloud system, fully distributed and mediated by a blockchain. If we think that a k8s cluster is made up of several nodes that offer a certain computational power, we could think of creating a sort of decentralized administration cloud, where each node supplies computational power and uses that of the cluster according to its need. In addition to the combination of computational power, this type of architecture presents an entry point for building any web application, web3 compliant, in which we would ideally have the management between Kubernetes of the various microservices that compose it and of the blockchain whose role is to apply access policies to assets and data by managing access from different realities. Finally, the solution we have presented represents a hybrid between public and private blockchains, allowing you to have the strengths of a public blockchain and a private blockchain

Bibliography

[1] Nakamoto, Satoshi. "Bitcoin whitepaper." URL: https://bitcoin.org/bitcoin. pdf-(: 17.07. 2019) (2008).
[2] "Introduzione a Web3" URL: https://ethereum.org/it/web3/
[3] "Ethereum whitepaper"" URL: https://ethereum.org/it/whitepaper/
[4] Polygon documentation "https://wiki.polygon.technology/docs/home/polygon-basics"