CCT College Dublin
ARC (Academic Research Collection)

ICT

Fall 2020

[CHAIN UP GROUP] Applied Technology Group Project

Maria Beluz Suarez
CCT College Dublin

Haein Kim
CCT College Dublin

Follow this and additional works at: https://arc.cct.ie/ict

Part of the Computer Sciences Commons

Recommended Citation

Suarez, Maria Beluz and Kim, Haein, "[CHAIN UP GROUP] Applied Technology Group Project" (2020). *ICT*. 14.
https://arc.cct.ie/ict/14

This Undergraduate Project is brought to you for free and open access by ARC (Academic Research Collection). It has been accepted for inclusion in ICT by an authorized administrator of ARC (Academic Research Collection). For more information, please contact debora@cct.ie.

Block chain Proof of concept

(Text hosting service application)

[**CHAIN UP GROUP**]

Applied Technology Group Project

Lecturers: Graham Glanville / Greg South

Group supervisor: Amilcar Ponte

| BSc. Information Technology - Year 3 || February 2020 |
Completed By: [MariaBeluz Suarez (2017367)/ Haein Kim (2017133)]

Chain_UP;

Table of Contents

1 Introduction

This document presents the draft of the introductory chapter as well as the research and planning aspects of developing a text hosting service application based on block chain technology named‘Chain UP. This report represents part I of the continuous assessment of the year-module named Applied Technology Group Project for the 3rd year of the Information Technology course at CCT College Dublin.

This chapter contains an introduction to Blockchain’s concepts and an overview of its benefits and the latest industry applications of this technology, specifically the financial services industry and the data security industry. In the next chapters, we will discuss how those concepts influenced our definition of the scope of the project and go into further detail on the proof of concept in Blockchain through developing a text hosting application.

A description of the provisional architecture diagram is also included. The schedule of tasks for the project can be found in the section named ‘Schedule’ in appendix B. Each team member contributed a report assessing their work and impact of duties on the deliverables up to this point. These can be found under the section named Appendix A – Individual Contribution Report.

1.1 Technology Background and use in industry

Blockchain is a permanent ledger that records transactions with anyone at any time, and it has often been introduced through Bitcoin to the world but back to Blockchain history, it was come out with cryptographically linking blocks in an append-only data structure as the main concept by Stuart Haber and W. Scott Stornetta in 1991. In January 2009, the first software-based on cryptocurrency was launched by Nakamoto. This is the bitcoin that we generally know when we look into the Blockchain network. From that time on, Smart contracts using cryptocurrency protocol have become widely available, resulting in many platforms such as Ethereum, Hyperledger Fabric and, IBM Blockchain(The History of Blockchain, 2020).

According to Imran Bashir(2017, p.14), this technology has been expected to influence many industries to further develop. Hence we could discover many use cases in industries that benefit from the Blockchain. Mainly internet of things, finance, governments, digital identify media and entertainment but not limited to them.

The financial services sector has some of the greatest examples of obsolete operational processes, time-consuming payment settlement requirements, limited transparency, and security vulnerabilities can be seen in. Blockchain offers functional solutions to these issues such as accountable and transparent governance systems, improved incentive alignment between relevant parties, technology infrastructure security and efficient business model solutions in Blockchain technology. Furthermore, Blockchain allows for the digitization of financial instruments, which brings greater liquidity, lower the costs of capital, reduces counterparty risk, and allows access to a broader investor and capital base.

In terms of Digital Identity, with ever-increasing data breaches and malicious hacking attacks occurring annually the management and protection of user and non-user identities and data are becoming more and more critical for the successful functioning of any organization. Not only are the practical business implications of a data breach incredibly onerous, but the damage to an organization's reputation in the eye of the public can also be extremely difficult to recover from. Digital identity theft negatively impacts millions of individuals annually. A Blockchain-based digital identity management system can provide a unified and tamper-proof infrastructure with positive benefits for enterprises, users, and IoT(Internet of Things) management systems.

1.2 Key concepts of Blockchain and process

We can see from the above examples traits of Blockchain are immutability, capacity, decentralization, security, and anonymity. Blockchain has potential issues with scalability, adaptability, regulation, relatively immature technology and privacy. These issues will be discussed in Chapter 4: Implementation of the system for more details.

1.2.1Blockchain nodes

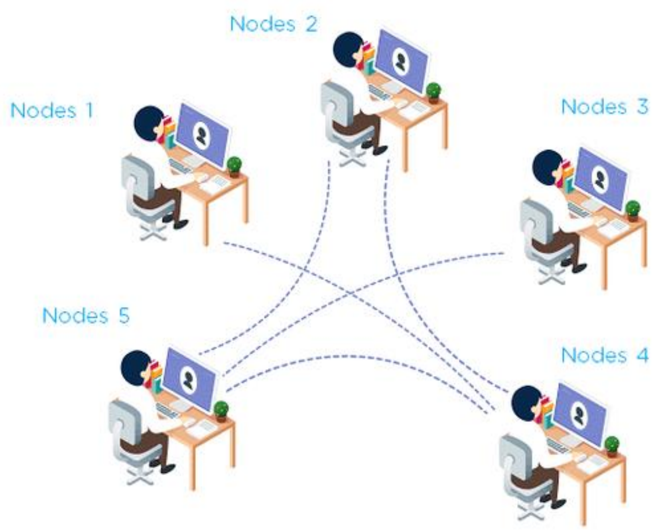


Figure 1Process flow (How are peers or nodes connected in Blockchain?, 2020)

Nodes can act in one of two roles, the first being 'miners' or alternatively 'signers'.A miners’ role is to create new blocks and mint cryptocurrency. A block signers’ role is to validate and digitally sign transactions. The decision of which node will append the next block to the Blockchain is a key decision in every Blockchain network.

1.2.2General Blockchainprocess

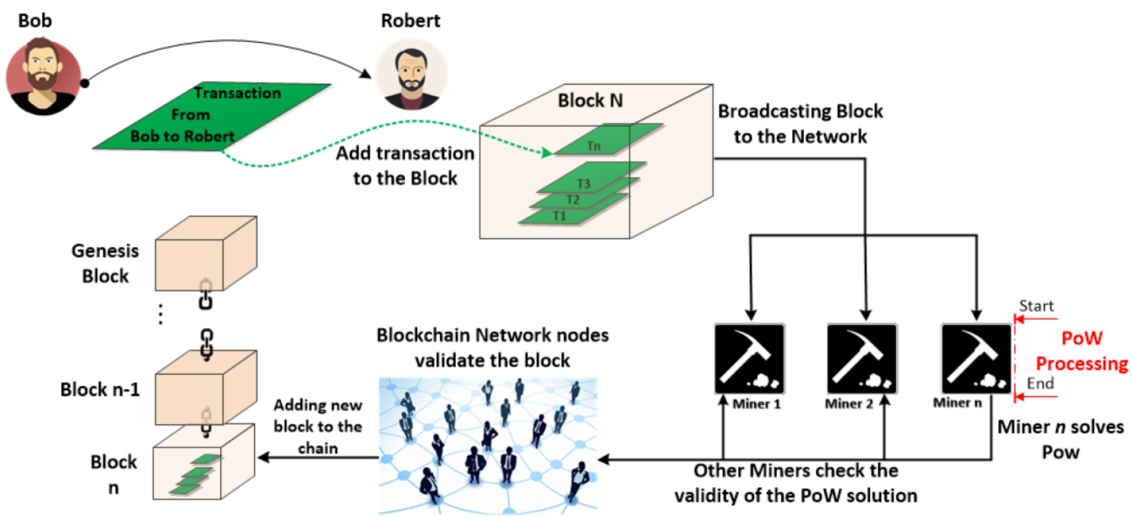


Figure 2Process flow(Panarello et al., 2020)

The following is a brief outline of how a Blockchain accumulates blocks and the relationship between transactions and blocks. There are five key steps.The view proposed by Imran Bashir (2017, p. 24)

1. A transaction is created and digitally signed by a node. In this case, a transaction typically consists of validation information such as source and destination addresses, relevant rules to the chain and some logic of transfer of value information. The transfer of data would be between two users who are both on the Blockchain network.

2. A flooding protocol (Gossip Protocol) propagates, or floods, the transaction to peer on the network and they then verify the transaction by using some predefined criteria.

3. After being validated the transaction effectively 'confirmed' and is included in a block and added to the network.

4. Being added to the network sees the block becoming part of the overall ledger and becomes a link in the chain, connecting to the last confirmed node and seeing

subsequent nodes being attached to it. The link of the node to the last confirmed node is called the 'hash pointer'. The transaction now gets its second confirmation and the block is confirmed for the first time as per the validation process mentioned above.

5. Every time a new block is created each previous transaction is reconfirmed, with six transactions usually needed to consider the transaction final.

1.2.3Consensus mechanism

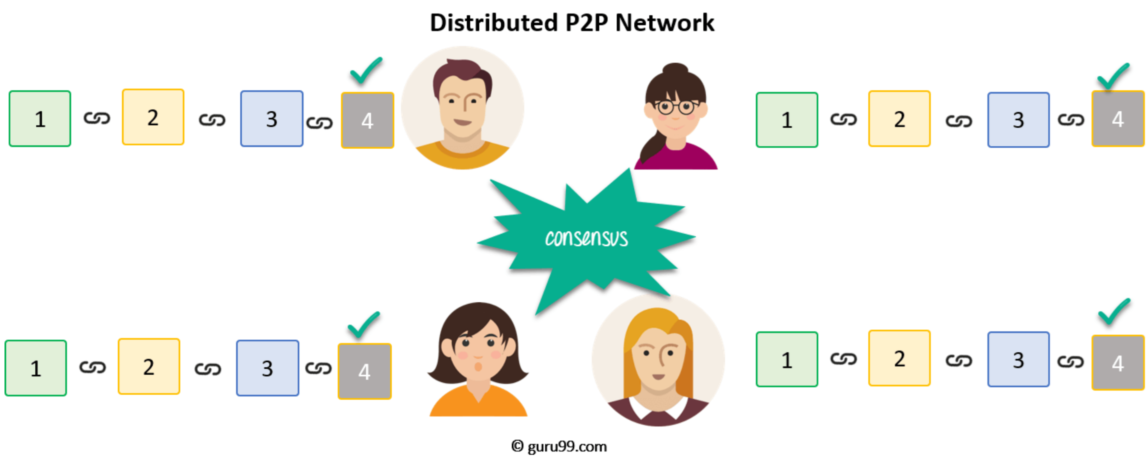


Figure 2Process flow (Blockchain Tutorial for Beginners: Learn BlockchainTechnology, 2020)

The consensus mechanism is a distributed computing concept used in Blockchain to provide a way for all peers in a Blockchain network to agree to a single version of the truth. This mechanism addresses two main types which are Traditional byzantine fault tolerance (BFT)–based known as the consortium or permissioned type and Leader election-based consensus mechanisms known as the fully decentralized or permissionless type. Apart from these consensus algorithms, the other noteworthy consensus algorithms are Proof of Work (PoW), Proof of Stake (PoS), and Proof of Activity (PoA).

1.3 Technologies to use – Why ETCD, why not Cassandra?

In the process of defining technologies to be applied to the present project, NoSQL Distributed Databases are an accurate and practical tool to work with. This approach offers Scalability, data consistency and high availability, these are some of the characteristics that provide functionality needed to represent the Blockchainprocess.

According to IBM Cloud Education (2019), there are 4 types of NoSQL databases:

1. Column-based, enable data access using a row key, column name, and Timestamp. The column does not have to be consistent across the records.
2. Key-Value based, it's a storage with a key (usually string of characters) and a corresponded value. Provide a way to store, retrieve and update data.
3. Document-based, A document is an Object and keys. It is used to store varying attributes along with large amounts of data.
4. Graph-based, is particularly useful to find connections between different pieces of data. (Use the Graph theory).

Particularly for the present project, the Key-Value model ideally becomes the best schema to be applied, mainly because, the elements of ETCD such as cluster, node, key-value data, RAFT consensus, etc., matched to the Blockchain elements(Network, node, transactions, and so on (we'll see how those concepts are related in the next chapters). In addition to that, there are few more functionalities such as APIs and Method access, Supported Programming languages and Replication methods that reinforce the use of it. We list more details in Table 1.1.

According to editorial information, by Solid IT (2020), the table below illustrates a summary of comparison about Cassandra and ETCD Databases.

Name	**Cassandra**	**ETCD**
Description	Wide-column store	A distributed reliable key-value store
Primary database model	Wide column store	Key-value store
Current release	3.11.6, February 2020	3.4, August 2019
Implementation language	Java	Go
Server operating systems	BSD, Linux, OS X, Windows	FreeBSD, Linux, Windows
APIs and other access methods	Proprietary protocol Thrift	gRPC JSON over HTTP
Supported programming languages	C#, C++, Clojure, Erlang, Go, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby, Scala	.Net, C, C++, Clojure, Erlang, Go, Haskell, Java, JavaScript, (Node.js), Perl, PHP, Python, R, Ruby, Rust, Scala, Tcl
Replication methods	selectable replication factor	Using Raft consensusalgorithm to ensure data replication with strong consistency among multiple replicas.

Table 1.1 Comparison Cassandra vs.ETCD

By definition, "ETCD is a strongly consistent, distributed key-value store that provides a reliable way to store data that needs to be accessed by a distributed system or cluster of machines. It gracefully handles leader elections during network partitions and can tolerate machine failure, even in the leader node." (ETCD, 2020).

KodeKloud (2019), describe the ETCD process:

1. A data operation can Read/Write in an instance(node) of a network(cluster).
2. There must exist a Leader node and followers.
3. ETCD uses RAFT consensus to elect a Node Leader.
4. A Leader is the only one on a charge to write data. Use a Log Replication process to make sure every instance(node) on the network has exactly the same data.

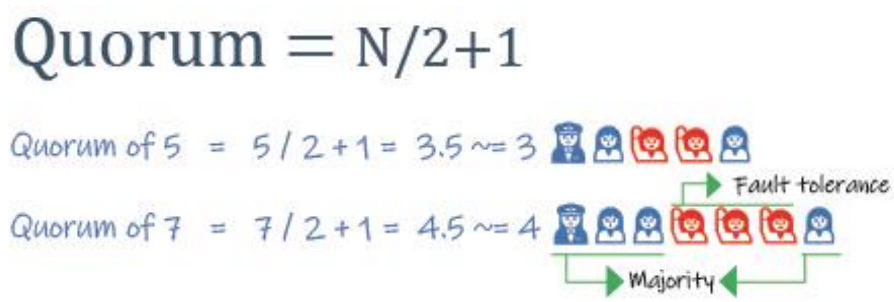


Figure 2Quorum example

5. An ETCD Cluster defines a quorum (minimum number of nodes that must be available for the cluster to function properly and make a successful write)

The entire ETCD process will be used to develop the main core functionalities we are looking for, thus adding other processes such as hashing and API requests will complete the Blockchain scheme. In the next section “System Design” we discuss and describe by using diagrams each process that is involved.

Our Idea

Stuart Haber and W. Scott Stornetta(1991, p2) outline that“First, one must find a way to time-stamp the data itself, without any reliance on the characteristics of the medium on which the data appears, so that it is impossible to change even one bit of the document without the change is apparent. Second, it should be impossible to stamp a document with a time and date different from the actual one.”This study indicates immutability which is one of Blockchain characteristics and immutability is verified through distributed ledgers and consensus algorithm. Back to the fundamentals of Blockchain, this study inspired us to demonstrate the Blockchain technology from scratch instead of using developed Blockchain platforms such as Hyperledger, Ethereum, and bitshare. In our case, we decided to demonstrate the Features of Blockchain technology providing a text hosting application. Our goal is to support the concept of Blockchain and seek a better understanding of tech. To do so, the structure of this application will be simple but functionally enough to verify

We are going to use these Blockchain characteristics to demonstrate each of the advantages and disadvantages of the Blockchain concept. From the Scratch, instead of using the Blockchain platform already developed, we will approach users in a more understandable way to demonstrate and verify the functionality of the Blockchain by showing them timestamp using the most basic Blockchain characteristics based on their work already mentioned before. We believe that this project can provide a direction for the next development by showing how Blockchain works.

1.1 Blockchainprocess in our application.

As a simple description, Blockchain is a distributed ledger, it can be compared with a dairy where we register every situation/transaction in chronological order. Therefore, this became a trusted and accurate way to know what happens since it started until the present, in other words, we will be able to know if any record has been altered by who and when.

This project aims to show how Blockchain works by showing, how each of its elements (mentioned in chapter 1.2), participates in the process. Having in mind the definition of Blockchain, we suggest using several technologies to demonstrate how Blockchain works.

Use Case: Hosting messages application (i.e. twitter)as illustrated in Figure 2.4.

It consists in a client (Webpage) that will send messages identified by UserId, in a JSON format, this Client will communicate with an API RESTful application, where to Read/Write messages, will send them by requests to an ETCD Cluster. This technology called ETCD will be held in the AWS Cloud.

A Blockchain process starts here; the ETCD cluster will simulate the distributed network in a Blockchain, having Virtual Machines (Linux) as Nodes where by using the Log replication guarantee all nodes contain the same data. Some Virtual Machines (acting as a miner) will run a Java program to hash the data received, and adding the new entries (Blocks) on the ETCD database. These entries will be registered with a Timestamp in a JSON file format and stored in the virtual machine.

The general ETCD process and The ETCD process in our project are illustrated for better understanding in Figure 2.1 and Figure 2.2.

1.2 Architecture diagram

In addition, to support the project idea, an Architecture Diagram of our application is seen in figure 2.3which is a High-Level graphical representation of the technologies and components separated by layers and describes how they work and interact with each other.

Therefore, in our particular case, we are going to describe a conceptual-Logical diagram which is based in 2 layers:

1. The API layer is a Web environment that contains a Website which will be developed in HTML and CSS and aims to allow any user to introduce data and make requests through a RESTFul API to the server.
2. On the ETCD cluster, we will set an ETCD (Distributed Ledger) mechanism's configuration. Its main functionality is making sure the data (Key-Value) will be stored in each of the nodes in the cluster. This feature will help to describe how a Blockchain works as a distributed ledger.

As an overview of the technologies we intend to use, we included the main tools and elements that will support the development of this project.

2 System Design

In this section, we use diagrams to describe the design process and decisions we made to carried out the Blockchain's proof of concept, by developing the Web Text Hosting Application in combination with ETCD and other frameworks we choose.

Our group pretend to simulate the Blockchain process, and how each step happens. It will be demonstrating by using ETCD. Generally speaking, the picture below illustrates the basic structure of this technology. It works with the concept of **cluster** which has an ID to be identified, a cluster contains and manage **nodes**, each one of them store a database. Cluster use a feature called **Quorum** to have the fault tolerance characteristic, and finally, use a **RAFT** consensus algorithm to select a node leader and replicate data across the cluster.

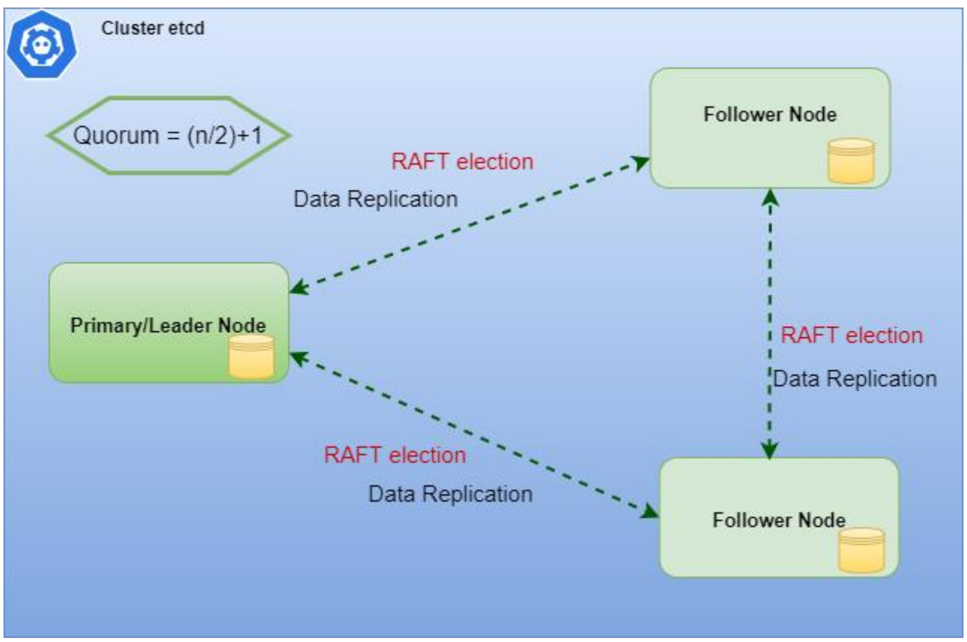


Figure 2.1 ETCD General Process

The process already explained, is the most difficult part of a Blockchain to be simulated, due to the complex networking's process to communicate between nodes in a network, and the ability to replicate data.

How, are we using this technology in the present project? Next figure shows how we use each element to simulate the Blockchain environment. The **network** is represented by the cluster, each **miner/user** represented by a node, **blocks** are replicated to each node. For technical and practical reasons, these tools are held in the AWS cloud.

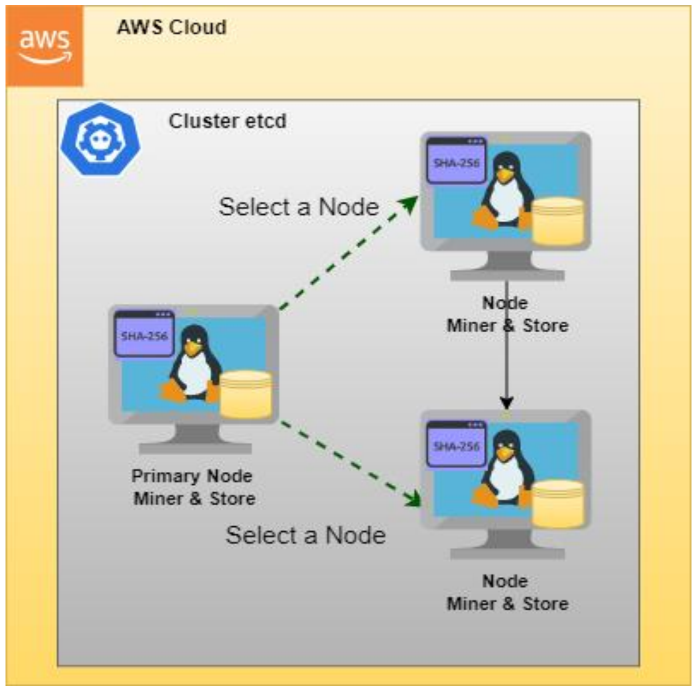


Figure 2.2ETCD applied to the project

Next, a high-level graphical representation of the project is described by the diagram below. As it was mentioned in chapter 1, this diagram has two layers: a) API-layer, is the Web hosting application, where a web page allows users to introduce data (messages) to be sent by a RESTful API to ETCD database. b) ETCD layer, where the data will be managed and stored.

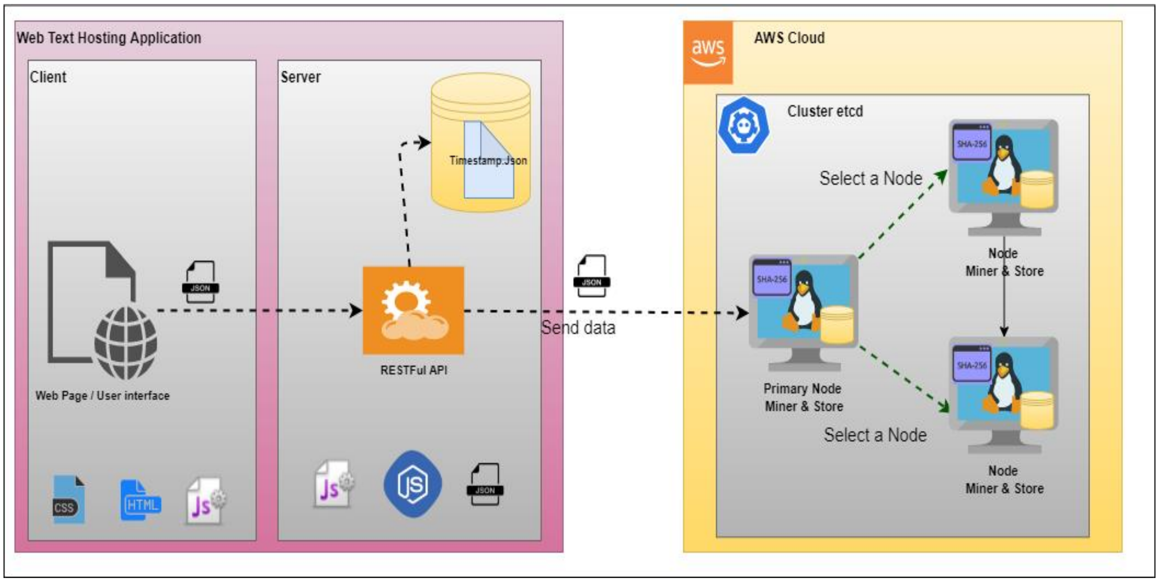


Figure 2.3 Architecture Diagram of Webhosting Application

In addition, to support the project idea, an Architecture Diagram of our application is seen in figure 2.3which is a High-Level graphical representation of the technologies and components separated by layers and describes how they work and interact with each other.

Therefore, in our particular case, we are going to describe a conceptual-Logical diagram which is based in 2 layers:

3. The API layer is a Web environment that contains a Website which will be developed in HTML and CSS and aims to allow any user to introduce data and make requests through a RESTFul API to the server.
4. On the ETCD cluster, we will set an ETCD (Distributed Ledger) mechanism's configuration. Its main functionality is making sure the data will be stored in each of the nodes in the cluster. This feature will help to describe how a Blockchain works as a distributed ledger.

As an overview of the technologies we intend to use, we included the main tools and elements that will support the development of this project.

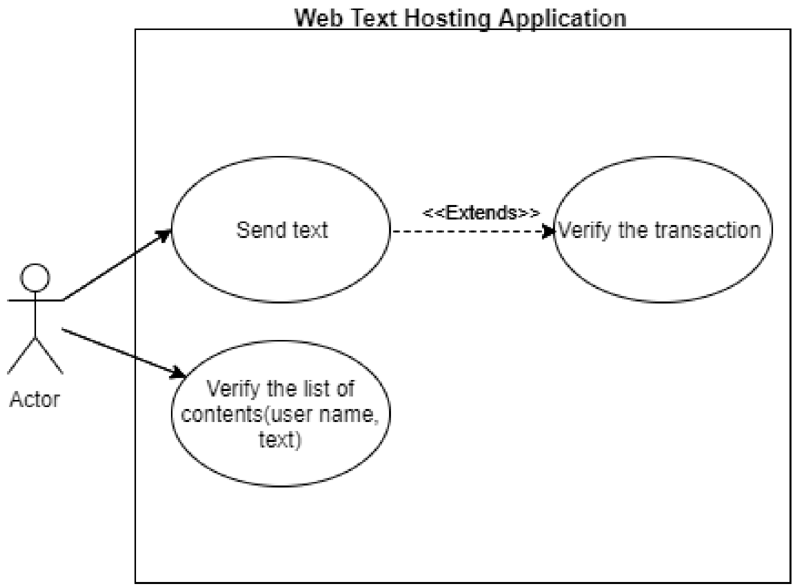


Figure 2.4(a)Use case of web text hosting application

NAME: SENDING TEXT.

Precondition: A user of the web text hosting application chose "Send text".
1. The user enters user ID and type the text that the user wishes to send.
2. The user clicks 'Submit'
3. The system displays the state of transaction from ETCD.
Post-condition: The user has sent the text.

NAME: VERIFY THE LIST OF TEXT.

Precondition: A user of the web text hosting application chose "List the content (user name, text)".
1. The user is able to see the contents that are made.
Post-condition: The user confirmed the messages

Figure 2.4(b)Use case Text of web text hosting application

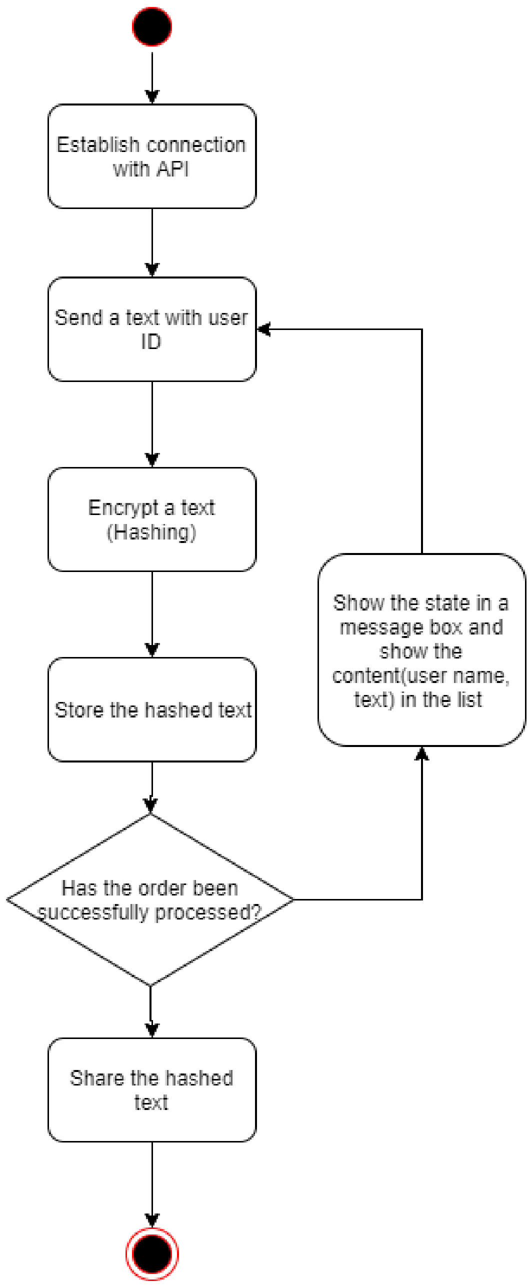


Figure 2.4(c)Activity diagram of web text hosting application

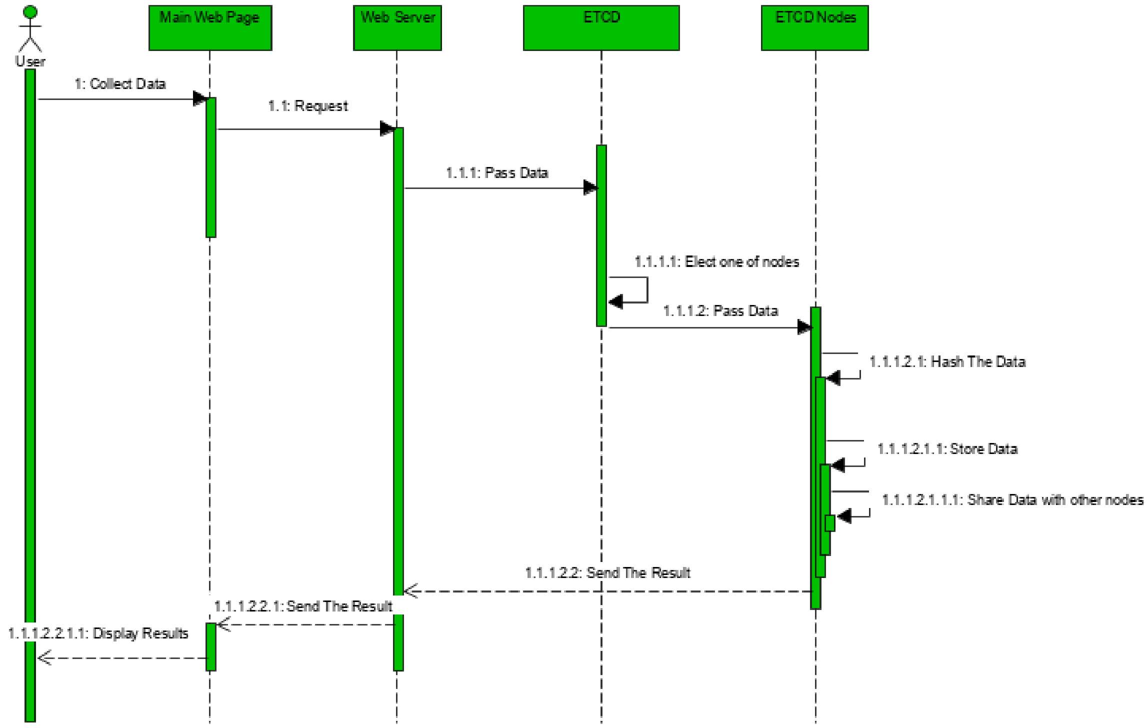


Figure 2.4(d)Sequence diagram of web text hosting application

The illustration below, is a defined and developed consensus algorithm used by ETCD, it is called “RAFT”. Is implemented to select a node leader to be on charge of replicate data across the ETCD cluster and making sure it will be available in every node. This feature in terms of Blockchain process, refers to, every user/node must have a copy of the whole Blockchain, specifically when a new user is added to the network.

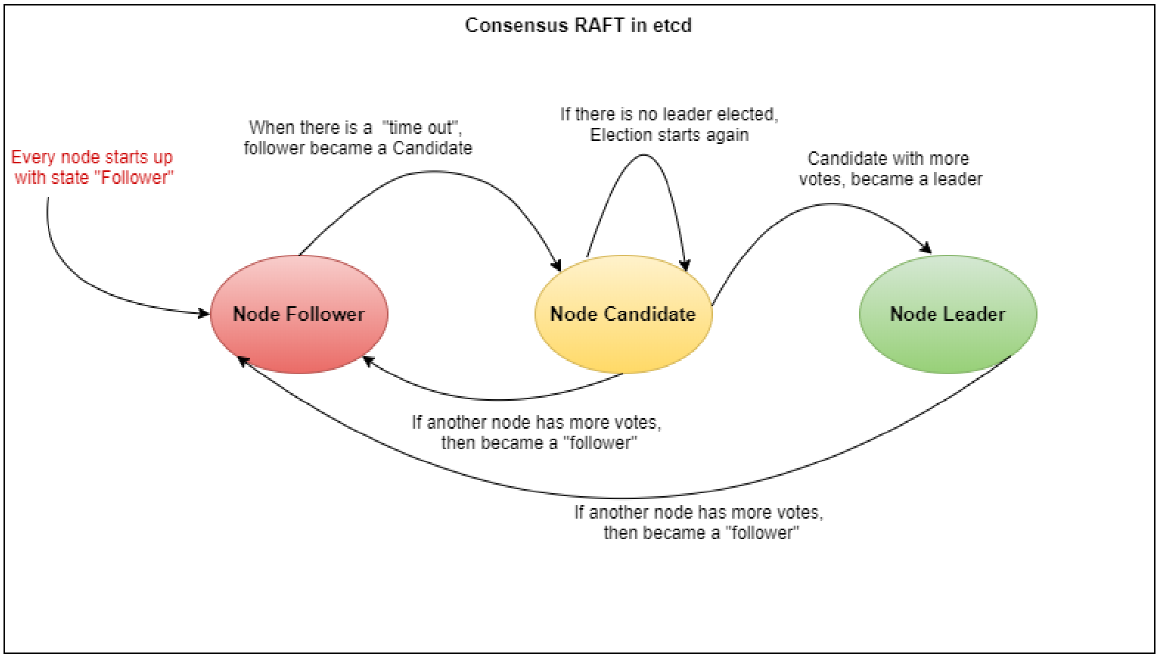


Figure 2.5 RAFT Consensuses in ETCD

Figure 2.6(a)Design user interface

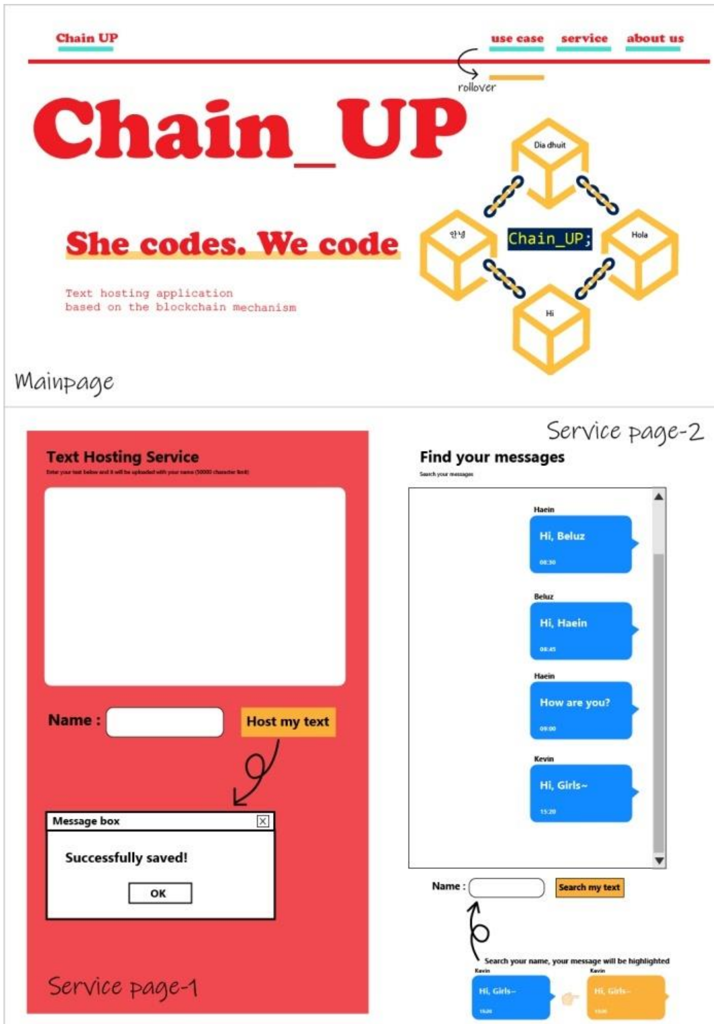


Figure 2.6(b)Design user interface

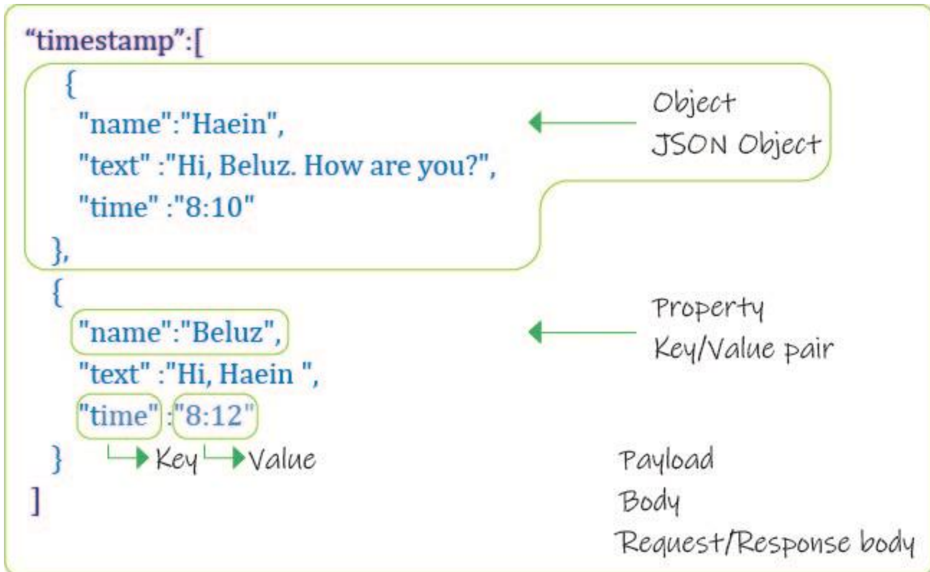


Figure 2.7Json File structure

The next figure shows the node's connection process to replicate the data. When the node leader in the cluster receives a request to store data, it will replicate it across the cluster and notify when the data was stored successfully in each one of the nodes alive. This make sure data will be available in every node of the cluster.

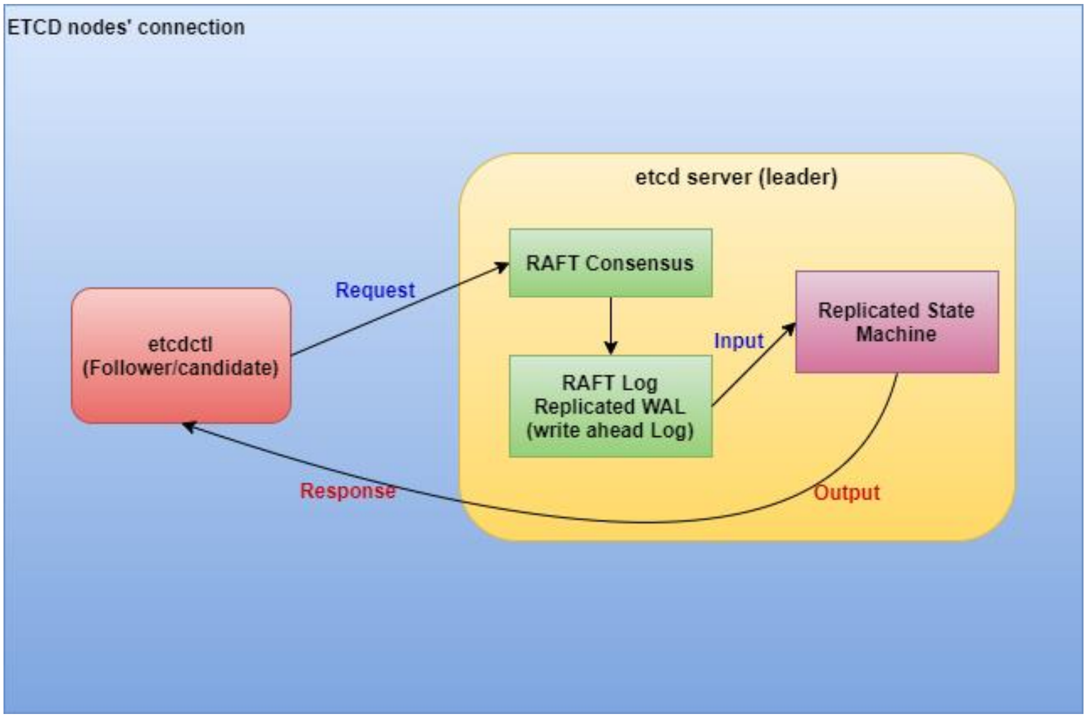


Figure 2.8 ETCD Node's Connections and parameters

3 System implementation

This chapter is the support of the theoretical demonstration of the concept. We aim to create an environment, to develop, configure, and combine the technologies we have mentioned before, and get the opportunity to technically describe the Blockchain process for a better understanding.

3.1 Etcd implementation

This technology can be installed in several Operating Systems, and also there is some pre-built binary configurations released to be installed in each of them. The approach we defined is, to install ETCD in 3 virtual machines with Linux OS, and configure them as a cluster, these VM must be located in a Cloud service.

- ETCD System Requirements:

According to ETCD (2020), the System requirements to install ETCD are: “The etcd performance benchmarks run etcd on 8 vCPU, 16GB RAM, 50GB SSD GCE instances, but any relatively modern machine with low latency storage and a few gigabytes of memory should suffice for most use cases.”

- Creation of Virtual Machines in AWS Cloud

In order to create the ETCD cluster, we had to set up 3 Linux virtual machines. Our team choose AWS Linux instances. Each machine has a static IP address, Public IP address and DNS hostname, and as the picture below shows they are located in AWS Cloud:

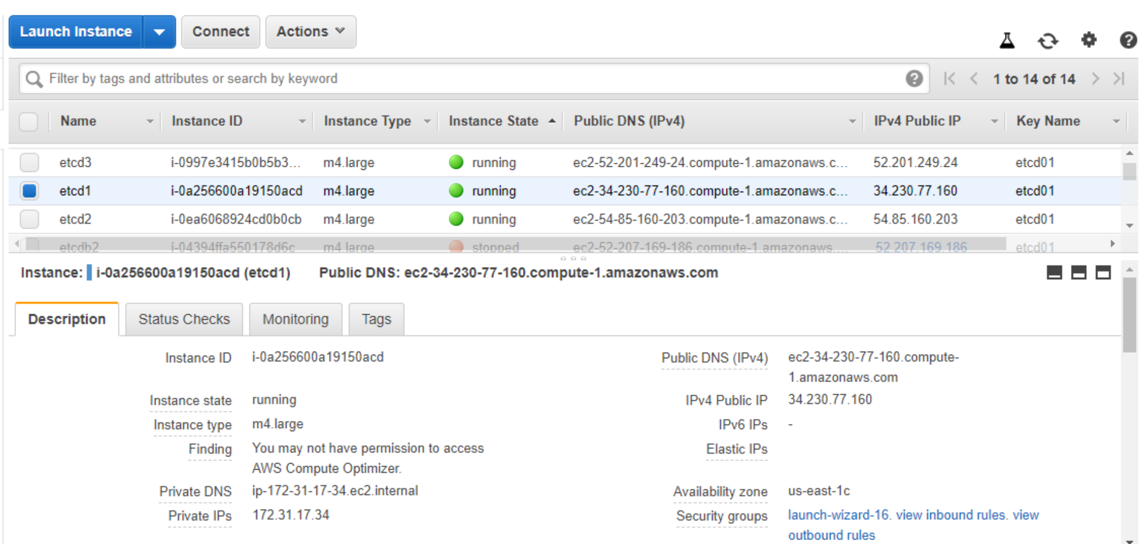


Figure 3 AWS Cloud VM

- ETCD installation and Cluster Configuration

The steps to configure this technology are shown in the picture below. This code must be run in each node of the cluster:

```
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// UPDATE LINUX OS
sudo yum update

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// DOWNLOAD ETCD BINARIES. YOU CAN SPECIFY THE VERSION
sudo yum -y install wget
export RELEASE="3.3.13"
wget https://github.com/etcd-io/etcd/releases/download/v${RELEASE}/etcd-v${RELEASE}-linux-amd64.tar.gz

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// EXTRACT THE DOWNLOADED ETCD FOLDER AND MOVE THE ETCD AND ETCDCTL BINARIES TO A NEW PATH
tar xvf etcd-v${RELEASE}-linux-amd64.tar.gz
cd etcd-v${RELEASE}-linux-amd64
sudo mv etcd etcdctl /usr/local/bin

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// VERIFY VERSION
etcd --version

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// CREATE A DATA DIRECTORY FOR ETCD
sudo mkdir -p /var/lib/etcd/
sudo mkdir /etc/etcd

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// DECLARE SOME ENVIRONMENT VARIABLES
TOKEN=token-03
CLUSTER_STATE=new
NAME_1=etcd1
HOST_1=172.31.17.5
CLUSTER=${NAME_1}=http://${HOST_1}:2380,${NAME_2}=http://${HOST_2}:2380,${NAME_3}=http://${HOST_3}:2380

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// CREATE A SERVICE CONFIGURATION FILE FOR ETCD SERVICE
cat <<EOF | sudo tee /etc/systemd/system/etcd.service
[Unit]
Description=etcd key-value store
Documentation=https://github.com/etcd-io/etcd

[Service]
User=etcd
ExecStart=/usr/local/bin/etcd \
  --data-dir /var/lib/etcd --name ${THIS_NAME} \
  --initial-advertise-peer-urls http://${THIS_IP}:2380 --listen-peer-urls http://${THIS_IP}:2380 \
  --advertise-client-urls http://${THIS_IP}:2379 --listen-client-urls http://${THIS_IP}:2379 \
  --initial-cluster ${CLUSTER} \
  --initial-cluster-state ${CLUSTER_STATE} --initial-cluster-token ${TOKEN}

Restart=on-failure
Restart=always
RestartSec=10s
LimitNOFILE=40000

[Install]
WantedBy=multi-user.target
EOF

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// USING ETCD API V3 WITH ENVIROMENT VARIABLES TO POINT THE ENDPOINTS IN THE CLUSTER
export ETCDCTL_API=3
HOST_1=172.31.17.5
HOST_2=172.31.31.235
HOST_3=172.31.25.110
ENDPOINTS=$HOST_1:2379,$HOST_2:2379,$HOST_3:2379

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// BY USING SYSTEMD, WE NEED TO CREATE AN ETCD SYSTEM USER
sudo groupadd --system etcd
sudo useradd -s /sbin/nologin --system -g etcd etcd
sudo chown -R etcd:etcd /var/lib/etcd/


//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// START DAEMON AND ETCD SERVICE
sudo systemctl daemon-reload
sudo systemctl start etcd.service
sudo systemctl status etcd

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///// CHECK THE NODE MEMBERS OF THE CLUSTER
etcdctl --endpoints=$ENDPOINTS -w table member list
```

Figure 4 ETCD configuration

- Testing Cluster Connectivity

After the nodes are configured in the cluster, it is important to check connectivity between them. The following pictures show the connectivity between nodes and how can all of them provide the same data.

Once the configuration is done, we can test the cluster creation by using the etcd client "etcdctl --endpoints=$ENDPOINTS -w table member list", as the picture below illustrate:

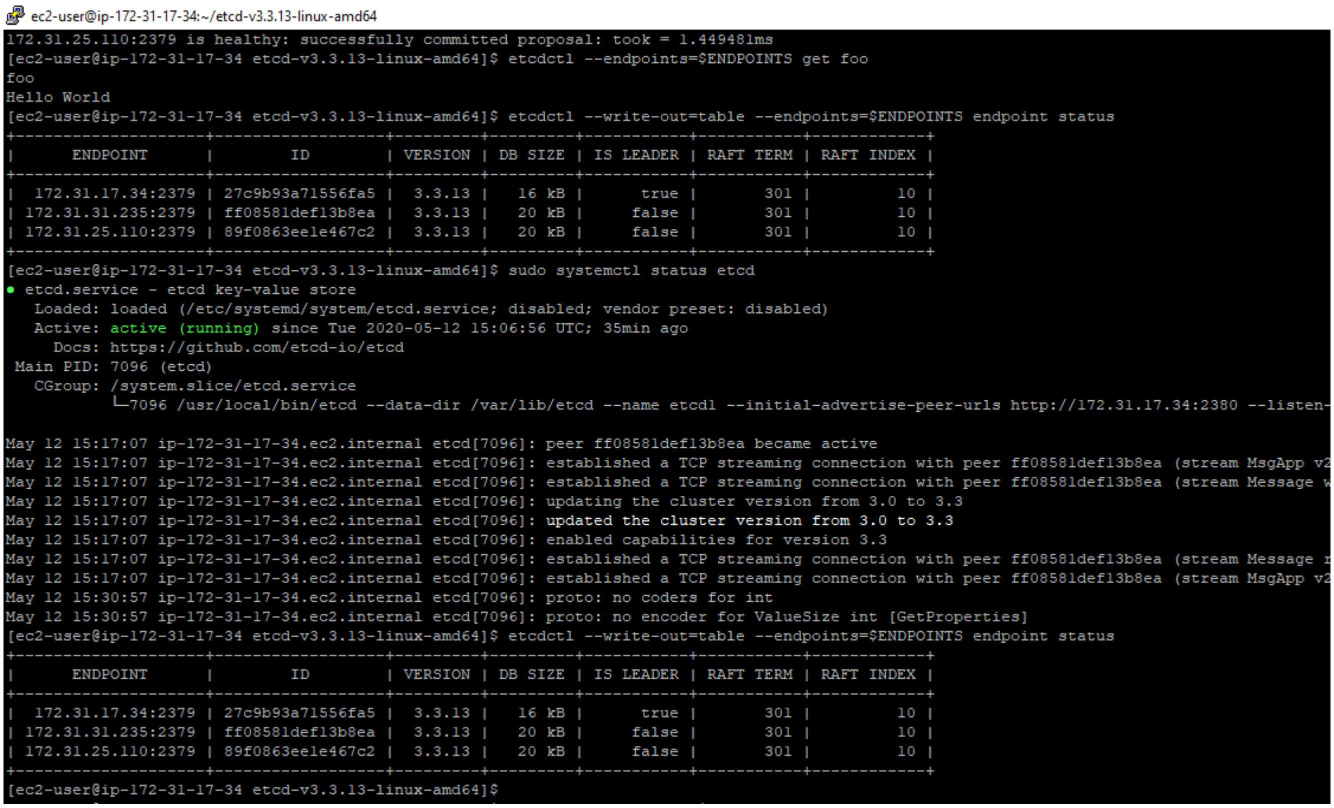
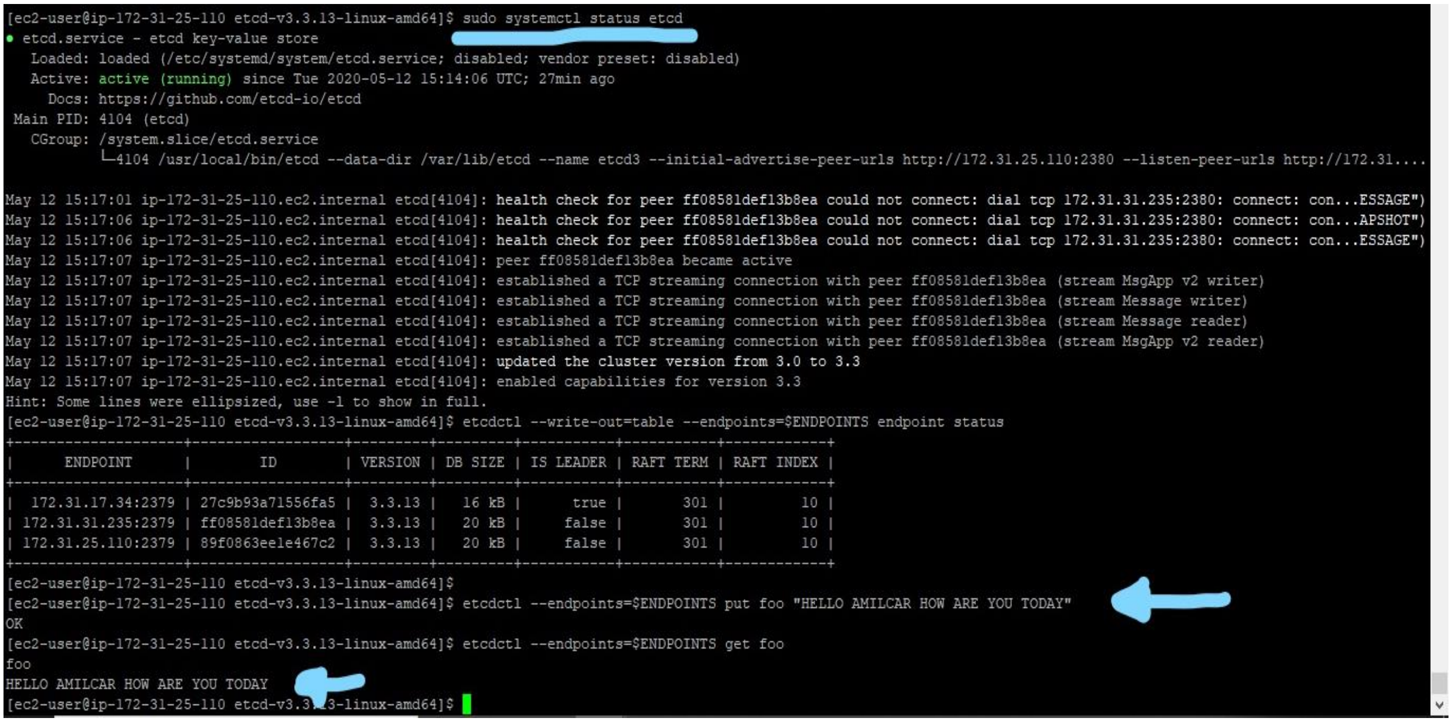


Figure 5 ETCD member list

The next illustration, corresponds to Virtual Machine called etcd3. In this test we set a message "HELLO AMILCAR HOW ARE YOU TODAY", which will be stored in the machine by using the command "PUT".



```
[ec2-user@ip-172-31-25-110 etcd-v3.3.13-linux-amd64]$ sudo systemctl status etcd
● etcd.service - etcd key-value store
   Loaded: loaded (/etc/systemd/system/etcd.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2020-05-12 15:14:06 UTC; 27min ago
     Docs: https://github.com/etcd-io/etcd
 Main PID: 4104 (etcd)
   CGroup: /system.slice/etcd.service
           └─4104 /usr/local/bin/etcd --data-dir /var/lib/etcd --name etcd3 --initial-advertise-peer-urls http://172.31.25.110:2380 --listen-peer-urls http://172.31....

May 12 15:17:01 ip-172-31-25-110.ec2.internal etcd[4104]: health check for peer ff08581def13b8ea could not connect: dial tcp 172.31.31.235:2380: connect: con...ESSAGE")
May 12 15:17:06 ip-172-31-25-110.ec2.internal etcd[4104]: health check for peer ff08581def13b8ea could not connect: dial tcp 172.31.31.235:2380: connect: con...APSHOT")
May 12 15:17:06 ip-172-31-25-110.ec2.internal etcd[4104]: health check for peer ff08581def13b8ea could not connect: dial tcp 172.31.31.235:2380: connect: con...ESSAGE")
May 12 15:17:07 ip-172-31-25-110.ec2.internal etcd[4104]: peer ff08581def13b8ea became active
May 12 15:17:07 ip-172-31-25-110.ec2.internal etcd[4104]: established a TCP streaming connection with peer ff08581def13b8ea (stream MsgApp v2 writer)
May 12 15:17:07 ip-172-31-25-110.ec2.internal etcd[4104]: established a TCP streaming connection with peer ff08581def13b8ea (stream Message writer)
May 12 15:17:07 ip-172-31-25-110.ec2.internal etcd[4104]: established a TCP streaming connection with peer ff08581def13b8ea (stream Message reader)
May 12 15:17:07 ip-172-31-25-110.ec2.internal etcd[4104]: established a TCP streaming connection with peer ff08581def13b8ea (stream MsgApp v2 reader)
May 12 15:17:07 ip-172-31-25-110.ec2.internal etcd[4104]: updated the cluster version from 3.0 to 3.3
May 12 15:17:07 ip-172-31-25-110.ec2.internal etcd[4104]: enabled capabilities for version 3.3
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-25-110 etcd-v3.3.13-linux-amd64]$ etcdctl --write-out=table --endpoints=$ENDPOINTS endpoint status
+-------------------+------------------+---------+---------+-----------+-----------+------------+
|     ENDPOINT      |        ID        | VERSION | DB SIZE | IS LEADER | RAFT TERM | RAFT INDEX |
+-------------------+------------------+---------+---------+-----------+-----------+------------+
|  172.31.17.34:2379 | 27c9b93a71556fa5 |  3.3.13 |   16 kB |      true |       301 |         10 |
| 172.31.31.235:2379 | ff08581def13b8ea |  3.3.13 |   20 kB |     false |       301 |         10 |
| 172.31.25.110:2379 | 89f0863ee1e467c2 |  3.3.13 |   20 kB |     false |       301 |         10 |
+-------------------+------------------+---------+---------+-----------+-----------+------------+
[ec2-user@ip-172-31-25-110 etcd-v3.3.13-linux-amd64]$
[ec2-user@ip-172-31-25-110 etcd-v3.3.13-linux-amd64]$ etcdctl --endpoints=$ENDPOINTS put foo "HELLO AMILCAR HOW ARE YOU TODAY"
OK
[ec2-user@ip-172-31-25-110 etcd-v3.3.13-linux-amd64]$ etcdctl --endpoints=$ENDPOINTS get foo
foo
HELLO AMILCAR HOW ARE YOU TODAY
[ec2-user@ip-172-31-25-110 etcd-v3.3.13-linux-amd64]$
```

Figure 6Virtual Machine etcd-3Sending message command

After setting a message and store it in the etcd database, we test to retrieve that message from the other machines. In the picture below shows how the virtual machine called etcd2 is retrieving the message.

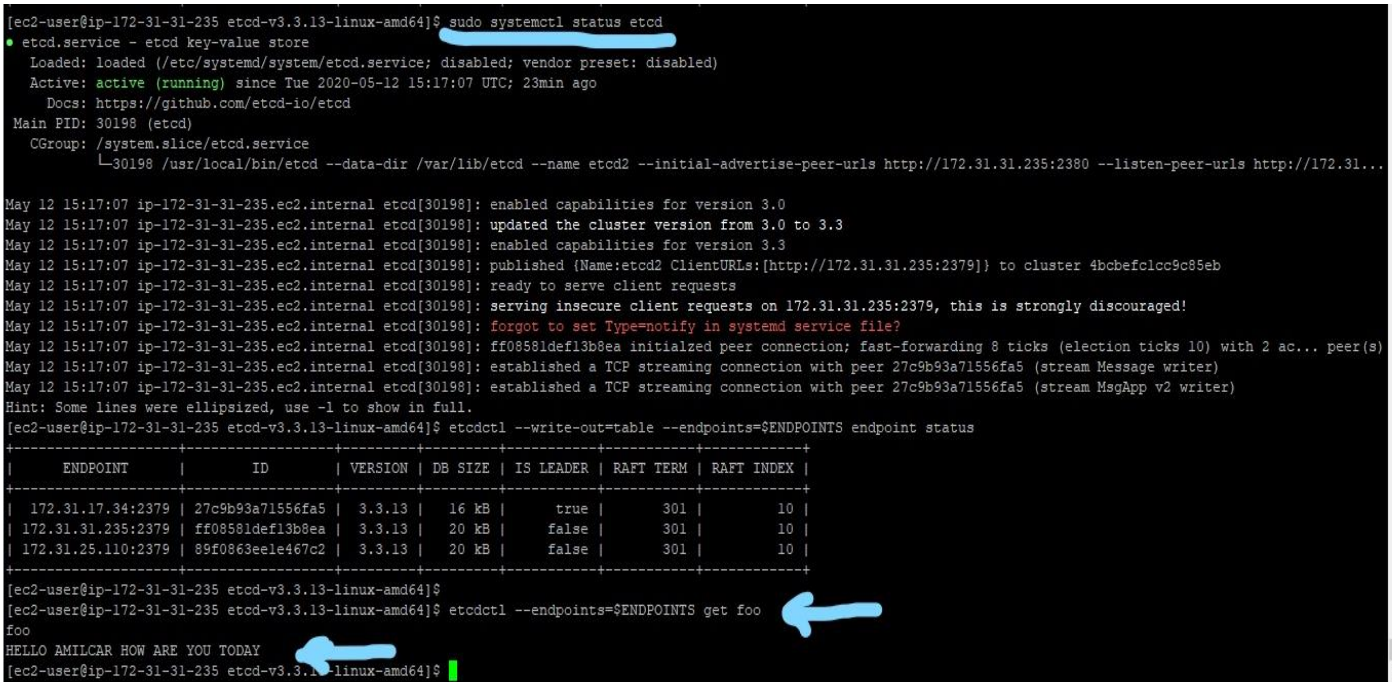


Figure 7 Virtual Machine etcd-2 retrieve message command

Finally, the next figure, shows how virtual machine called etcd1, retrieves the same message.

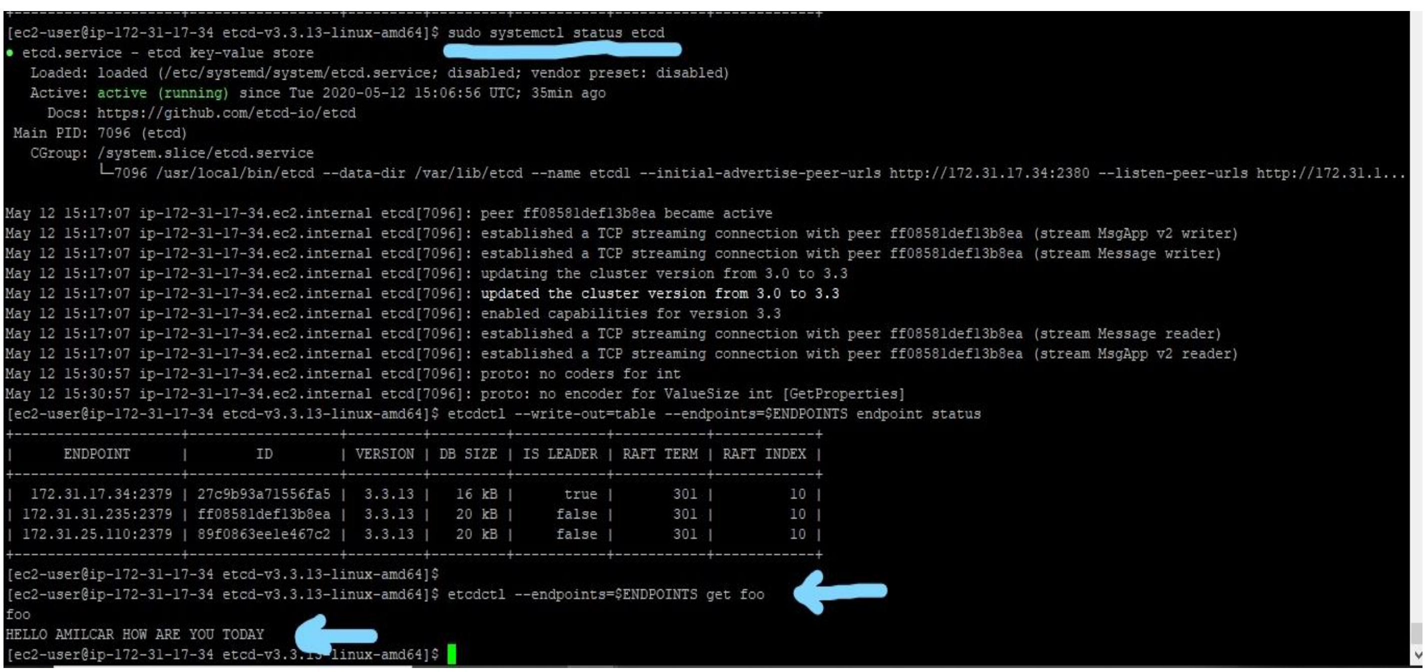


Figure 8Virtual Machine etcd-1 retrieve message command

In order to interact with ETCD from our application, is necessary to configure a gateway. ETCD v3, for its messaging protocol, use by default gRPC google Remote Procedure Call, which is an open source RPC created by Google. For communicating with the cluster, ETCD provides gRPC Gateway, and must be configured using this code "etcd gateway start --endpoints=$ENDPOINTS". As the picture below shows:

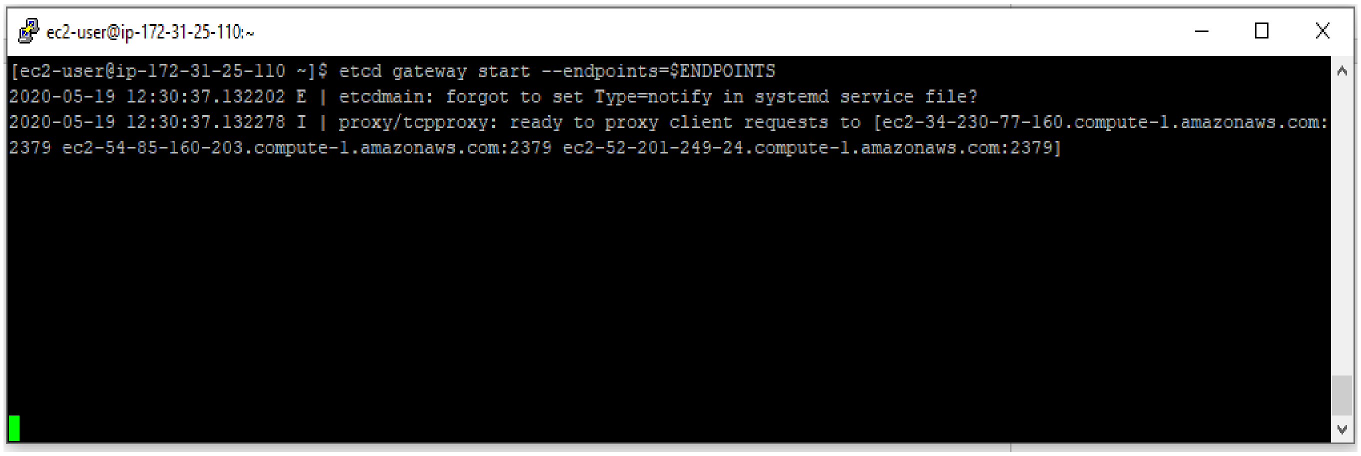


Figure 9 ETCD Gateway configuration

- Hashing Program

There is a program developed in Java to do Hashing. It is a developed demo, and according to Kass (2017), This program uses the algorithm SHA-256. The picture below is an example of the result of run that Java program:

```
Trying to Mine block 1...
Block Mined!!! : 000005d22653f64aed3cd519c87e51e8f1b4dc6fe432398c67c6ce25fad64864
Trying to Mine block 2...
Block Mined!!! : 0000008c464dba7c07970b7653bbdf4c28ed407b61ec712575477a3aff830a03
Trying to Mine block 3...
Block Mined!!! : 00000cae8c496cdd544b37019490938393d5121ed63253b3ef2cc832a4fdf4af

Blockchain is Valid: true

The block chain:
[
  {
    "hash": "000005d22653f64aed3cd519c87e51e8f1b4dc6fe432398c67c6ce25fad64864",
    "previousHash": "0",
    "data": "Hi im the first block",
    "timeStamp": 1589920719764,
    "nonce": 480900
  },
  {
    "hash": "0000008c464dba7c07970b7653bbdf4c28ed407b61ec712575477a3aff830a03",
    "previousHash": "000005d22653f64aed3cd519c87e51e8f1b4dc6fe432398c67c6ce25fad64864",
    "data": "Yo im the second block",
    "timeStamp": 1589920723157,
    "nonce": 564513
  },
  {
    "hash": "00000cae8c496cdd544b37019490938393d5121ed63253b3ef2cc832a4fdf4af",
    "previousHash": "0000008c464dba7c07970b7653bbdf4c28ed407b61ec712575477a3aff830a03",
    "data": "Hey im the third block",
    "timeStamp": 1589920725013,
    "nonce": 1685490
  }
]
```

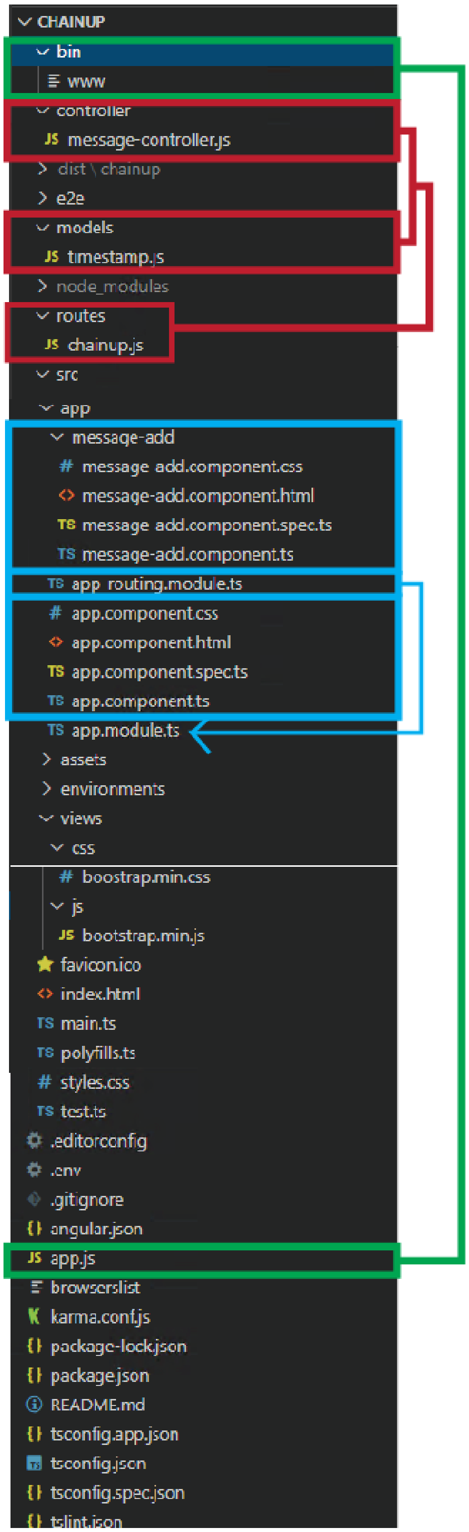
Figure 10 Java program to hash data

3.2 Web Deployment

To help users better see the process of the Blockchain, Web deployment is a perfect choice. It is aimed to have the least core functionality to demonstrate the flow of Blockchain works.

3.2.1 AngularJS(Framework)

AngularJS is a Google-based JavaScript framework that was announced in 2009 and a JavaScript client-side MVC/MVVM framework that is the essence of modern single-page web application development. AngularJS is used to reduce the amount of code to be written in JavaScript. It clearly separates development areas such as HTML, CSS, and logic in our project.



This image shows the structure of our web API, first of all, when the API runs, it starts from www.js as a door of web API. www.js consists of networking setting such as port and it integrates with app.js(in the green box). index.html is the front base and it calls each component of pages. the web page is divided into pieces as components by the functionalities in the app directory. Each piece of the web page gets managed separately and assembled in index page by app.module.ts in the end (blue box in the image). Directory controller, models, routes are the ingredient of the API that allows to constitutes the pieces of web pages (message-add, app.component). In addition, .env holds the credentials such as Mongo DB key.

3.2.2 Mongo DB

Mongo DB is a document-oriented database and is a NoSQL open source database such as ETCD. We have built a json database called timestamp to show the results of the process in which the message is sent to the user via etcd, which is not included in the characteristics of the Blockchain. The timestamp is currently in the model file.(See Red Box)

3.2.3 Heroku

To connect web server and ETCD, we had to have a static IP, to do so, we could host web API using Heroku. Since Heroku is deployed only in node.js format, the angular based web API requires additional processes in package.json.The code in the package.json is as follow:

(Our WEB API URL: https://chainup.herokuapp.com/)

Figure 3.1 Web API directory structure

```
{
  "name": "chainup",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "node ./bin/www",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular-devkit/build-angular": "~0.901.5",
    "@angular/cli": "^9.1.6",
    "@angular/compiler-cli": "^9.1.7",
    "@angular/animations": "~9.1.6",
    "@angular/common": "^9.1.7",
    "@angular/compiler": "^9.1.7",
    "@angular/core": "~9.1.6",
    "@angular/forms": "~9.1.6",
    "@angular/platform-browser": "~9.1.6",
    "@angular/platform-browser-dynamic": "~9.1.6",
    "@angular/router": "~9.1.6",
    "bluebird": "^3.7.2",
    "body-parser": "^1.19.0",
```

```
    "dotenv": "^8.2.0",
    "env": "0.0.2",
    "express": "^4.17.1",
    "heroku": "^7.41.1",
    "mongodb": "^3.5.7",
    "mongoose": "^5.9.14",
    "morgan": "^1.10.0",
    "rxjs": "~6.5.4",
    "serve-favicon": "^2.5.0",
    "tslib": "^1.10.0",
    "zone.js": "~0.10.2",
    "typescript": "~3.8.3"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.901.5",
    "@angular/cli": "^9.1.6",
    "@angular/compiler-cli": "^9.1.7",
    "@types/jasmine": "~3.5.0",
    "@types/jasminewd2": "~2.0.3",
    "@types/node": "^12.11.1",
    "codelyzer": "^5.1.2",
    "jasmine-core": "~3.5.0",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "~5.0.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage-istanbul-reporter": "~2.1.0",
    "karma-jasmine": "~3.0.1",
    "karma-jasmine-html-reporter": "^1.4.2",
    "protractor": "~5.4.3",
    "ts-node": "~8.3.0",
    "tslint": "~6.1.0",
    "typescript": "~3.8.3"
  }, "engines":
    {
      "node": "12.16.3",
      "npm" :"6.14.4"
    }
}
```

Figure 3.2 package.Json

3.3 Integration between ETCD and web API

As we saw in the gRPC configuration earlier, we can communicate with each node of web API and ETCD through gRPC. gRPC enables client and server applications to communicate transparently, simplifying the deployment of connected systems (gRPC, 2020)

The scenarios we are trying to do here are as follows:

Users send messages through web API and web server forwards to ETCD through gRPC. At this time, ETCD receives the message, elects the leader through the RAFT consensus algorithm, and delivers and stores the hashed message to each node. After this process, the ETCD returns the value of success to Boolean. So that it tells users that the transactions successfully made.

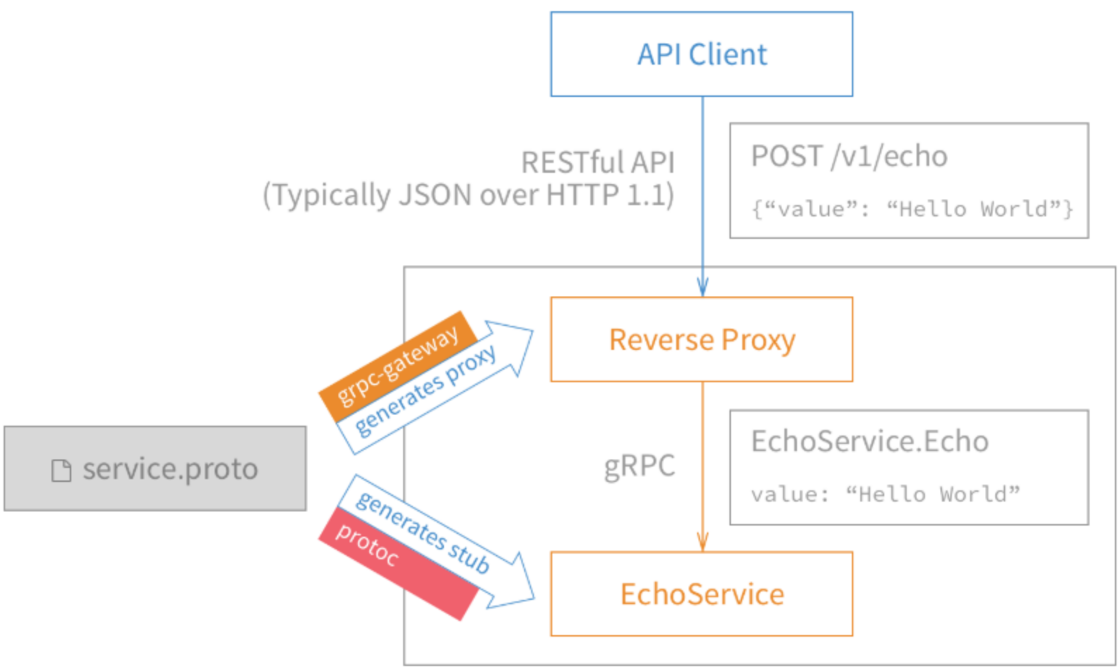


Figure 3.3 the process of gRPC(gRPC, 2020)

- Toperform gRPC, the code in web API is listed below.

```
if r.ProtoMajor == 2 && strings.Contains(r.Header.Get("Content-Type"), "application/grpc") {
    grpcServer.ServeHTTP(w, r)
  } else {
    otherHandler.ServeHTTP(w, r)
  }
```

- To create a message and send the POST request

```
  # target Create the message
curl -L http://chainup.herokuapp.com/v3/message \
  -X POST \
  -
d '{"compare":[{"target":"CreateMsg","key":"Zm9v","createRevision":"2"}],"success":[{"requestPut":{"name":"beluz","text":"hello"}}]}'
# {"header":{"cluster_id":"5461722588757394923","member_id":"18377035165073651946","revision":"34","raft_term":"4329"},"succeeded":true,"responses":[{"response_put":{"header":{"revision":"3"}}}]}
```

- Verified result in ETCD as follows

```
etcdctl--endpoints = SEBDOIUBTS get Haein -
 w=json{ "header": { "cluster_id": 5461722588757394923, "member_id":
 18377035165073651946, "revision": 34, "raft_term": 4329 }, "kvs": [
{ "key": "SGFlaW4=", "create_revision": 14, "mod_revision": 21, "version": 3, "value": "SGkgQmVsdXosIGhvdyBhcmUgeW91Pw==" }], "count": 1
 }
```

4 Results

After having all ETCD implemented, gRPC gateway configured and the Web Application deployed, we can test the integration of both technologies.

The technical way to test, is by the Web App sending a request to ETCD. The structure of that is:

curl -L http://ec2-34-230-77-160.compute-1.amazonaws.com:2379/v3beta/kv/range -X POST -d '{"key":"key1"}'

- ✓ We use "curl" a command-line tool, that is used for transferring data by using several protocols such as HTTP.
- ✓ An endpoint address must be specified, with a port opened.
- ✓ "v3beta" is must be specified after the endpoint address, is there where gRPC services resides.
- ✓ ETCD use gRPC Gateway to interact with other applications. It has 3 services such as; a) "KV" for Key-Value data management (use "range" or "put" to read or write key-values). b) "Watch" is a service that watch keys, when they are updates and follow their behaviour, c) "Auth" a service to set up authentication in order to access data.
- ✓ "-X POST -d" to indicate the method we want to use and data details.
- ✓ And lastly, the data itself in Json format.

5 Conclusion

As stated in the introduction, our initial goal of this project was to demonstrate some of the Blockchain mechanism characteristics from scratch. Even understanding what a Blockchain is can be difficult, so we want to give users a better understanding through the delivery of WebHost applications by providing our web host application. Blockchain, as it is today, has been mentioned as a popular topic for the past decade. As the Stuart Haber and W, Scott Stoneta's report on timestamps came out in 1991, it's an old but still new field with infinite possibilities. Recently, there have been so many attempts and projects using the Blockchain, but the high level of initiation has made it difficult for many people to even use the Blockchain tool. However, Blockchain will no longer be a difficult technology if it is developed as a simple tool that is more accessible and commonly used around it.Although it has been widely used in the virtual currency sector since a few years ago, it is now beginning to take effect in other areas, especially finance. After testing the tools we have and implementing ETCD, we can conclude that a Blockchain is a good tool for use and implementation. It will not lag behind other technologies in that security risks are lower than other applications and that everyone can participate in transactions and monitor everything like the owner.

The project also helped us once again understand web development by using the restful API development theoretical content covered in the class during the lecture.Also, the use of ETCD was a new attempt for us as a topic we had never explored before. Similarly, it was a new experience that could further enhance the understanding of typescript and Linux.Although we recognize that there are many improvements that need to be implemented and that the project is far from complete due to the time-consuming problem, we believe that there are many improvements to be implemented.

First, I would like to point out the significance of creating a project using technology that has never studied the mechanism of the Blockchain from the beginning. Furthermore, I hope that future research using Blockchain will serve as a stepping stone and opportunity for future users. We believe this project has helped us develop our own problem-solving skills. I think this is not the end, but the opportunity to study a project using another Blockchain.

Schedule

This chapter intends to describe the planning of each step towards the completion of the text hosting application.

We structured the project into 5 separate phases:

1. Planning. This phase will see us deciding the purpose of the project and how the app will behave i.e. defining the consensus algorithms and parameters needed, the use cases and diagrams. This will be done by researching and ultimately deciding on what platform and cloud provider we will use (ETCD, Cassandra).

2. Design. In this phase we will outline our vision of what the final output will look like and how it will work. To achieve this we will design the following; the user case, the preceding diagram(transaction flow architecture), the Blockchainapplication(Distributed system), the user interface and the access control capability for the users of the network

3. Development. Here we will bring together all of the aspects identified in the planning and design phases. Starting by installing the necessary prerequisites (VS code, Docker, golang, node.js etc) we will also model the network, generate the network on ETCDand deploy the nodes for hashing and storing data. This phase also sees us engage with website development i.e. user interaction and also interacting HTML and Javascript with Node.js (Web3 over RPC).

4. Testing. For the main component of this phase we will install the system developed in the previous stage, deploy the network and then evaluate the whole process and the possibility of adding nodes to the network. Data in clusters will also be tested and an end to end test of the app will be carried out Following of the tests mentioned above we will carry out a review to evaluate the actual results compared to our initial goals.

5. Documentation. In this phase we will formally document the project and the measures that were taken to achieve our final outcome. We will also devise the final presentation and update the reports over all look i.e. table of contents, diagrams.

References

Haber, S. and Stornetta, W., 1991. How to time-stamp a digital document. Journal of Cryptology, 3(2), pp.99-111.

ETCD, (2020), *ETCD.* Available at:https://ETCD.io/ [Accessed by 20 of February 2020]

IBM Cloud Education, (2019), *NoSQL Databases*. Available at: https://www.ibm.com/cloud/learn/nosql-databases [Accessed by 11 March 2020]

Solid IT, (2020), *System Properties Comparison Cassandra vs. ETCD vs. TerarkDB*. Available at: https://db-engines.com/en/system/Cassandra%3BTerarkDB%3BETCD[Accessed by 28 March 2020]

KodeKloud, (2029), *Kubernetes: ETCD for Beginners,* Available at: https://www.youtube.com/watch?v=L9xkXzpEY6Q&t=737s [Accessed by 30 March 2020]

Bashir, I., 2017. Mastering Blockchain. Birmingham: Packt Publishing Ltd.

Ecommerce Guider. 2020. The History OfBlockchain. [online] Available at: <https://ecommerceguider.com/history-of-Blockchain/> [Accessed 16 April 2020].

Panarello, A., Tapas, N., Merlino, G., Longo, F. and Puliafito, A., 2020. Blockchain And Iot Integration: A Systematic Survey. [online] Available at: <https://www.mdpi.com/1424-8220/18/8/2575/htm> [Accessed 21 April 2020].

Quora.com. 2020. *How Are Peers Or Nodes Connected In Blockchain?*. [online] Available at: <https://www.quora.com/How-are-peers-or-nodes-connected-in-Blockchain> [Accessed 21 April 2020].

Guru99.com. 2020. *Blockchain Tutorial For Beginners: Learn Blockchain Technology*. [online] Available at: <https://www.guru99.com/Blockchain-tutorial.html> [Accessed 21 April 2020].

gRPC. 2020. Grpc. [online] Available at: <https://grpc.io/> [Accessed 19 May 2020].

ETCD, (2020), *ETCD.* Available at:https://etcd.io/docs/v3.3.13/dl_build/

[Accessed by 20 of February 2020]

Kass, (2017), Creating Your First Blockchain with Java. Part 1. Available at: https://medium.com/programmers-blockchain/create-simple-blockchain-java-tutorial-from-scratch-6eeed3cb03fa [Accessed by 10 of March 2020]

Appendix A – Individual Contribution Report

HAEIN KIM'S CONTRIBUTION REPORT

At a point in this project we found it very difficult to work together within the team because of differences in working styles and opinions with other team members. As such, we lost a considerable amount of time trying to find a more effective way to co-operate and communicate better with each other which ultimatly did not work out. This caused us to fall behind schedule on the project and left me under a lot of time pressure during the rest of the project. I had a tremendous increase in the workload on this project to balance with my other responsibilities. I prioritized the most important areas by using my organizational skills to focus on the most important areas and to endure that I had adequate time to study and research potential problems or areas that would take more time as development progressed.

I find it disappointing that I spent so much time researching what software and ideas I would use that I was not able to progress with, but without this research, I don't think I would now have as solid understanding of the blockchain environment as I do now.

I organized the documentation first, and after that, I mainly took charge of web management, and other members took charge of ETCD. I updated the information on basecamp after consulting with other team members. In conclusion, I did not implement this project as planned, but I feel that I developed: my ability to solve problems as they arise, my ability to adapt and think on my feet to overcome difficulties and blockages. I also developed my troubleshooting abilities. We each had our own task, but we tried to solve the problem together by informing each other when there was a problem. Especially at the end, I had a lot of problems in implementing gRPC and hosting the web in HEROKU, but thanks to the encouragement and help from each other, I was able to try to the end.

I'm really glad that the nearly year-long project that seemed to never end is over. We've had so many challenges and troubles in the last few months of doing the project. It's a pity that I didn't get a good outcome from thefirst group, but I learnt so much from this situation also which will be a benefit to me in my future career. It was also a good opportunity for me to understand each other and listen to each other's opinions and think about what the team project is. Of course, we needed a lot of effort to do many tasks given to us in other modules, but I am glad that we learned a lot from this project. This is not the end, but from this point on, I would like to do this project again if I have a chance in the future.

MARIABELUZSUAREZ'S CONTRIBUTION REPORT

At the beginning of the year, we attend an event about Blockchain organized by IBM, where we saw an application developed based on Blockchain methodology, we contact with the presenters to get some ideas about our project.

This semester it was more challenging than I expected, due the fact that our team split, and that represented to start with a new project. Having that in consideration, the rest of the team, decided to pick a new project and start the whole process of planning again. That took time and take out the work we have done for the previous project.

I made a proposal about developing a Blockchain for Journalism, where basically the project will mainly focus on store news details, such as authorship, copyright, timestamp which will allow journalists to put their work safe, and auditable (where a media company wants to keep track and save their news). The project will be focus on a small group of 10 journalist (Bloggers, Students, etc.) who can save their work on the Blockchain solution developed in Hyperledger Fabric. We pick up one, which is described in the present project.

Once we start to planning the new project, we attend some meetings with a CCT lecturer to help us with useful information to shape the project goals.

Another decision we took was, as a team do a research about new technologies we wanted to use, such as ETCD and Cassandra and have meetings every Wednesday to discuss ideas and defined the planning and design. In our meetings we decided to use AWS Cloud for our project. For that I opened an AWS student account to be able to use the Cloud. I also apply for the Github Developer Pack, which provides a free access to many tools such as AWS credits, which we used in our account to be able to work with the Cloud.

After having the planning and design done, we decided to split the work between both of us. I took the part of ETCD implementation. I have to do a deeply research about it, specifically the configuration and installation of the technology. I configured several virtual machines in the Cloud as a testing trying to get ETCD configured. I set up the ETCD cluster in 3 machines, the ones we have up and ready.

I also worked in a Java program which we pretend to use for hashing data. I found a blockchain demo in a java program. It was explained in a website, I follow the instructions and set the program up and running. I also worked in set the program in the virtual machines located in the Cloud.

However, we were working closer even in this time with a lockdown. We manage to have online meetings. My teammate supports me all the time and I have learn a lot about working in a team, it is not easy as it looks but is not impossible.

Appendix B – Gantt Chart

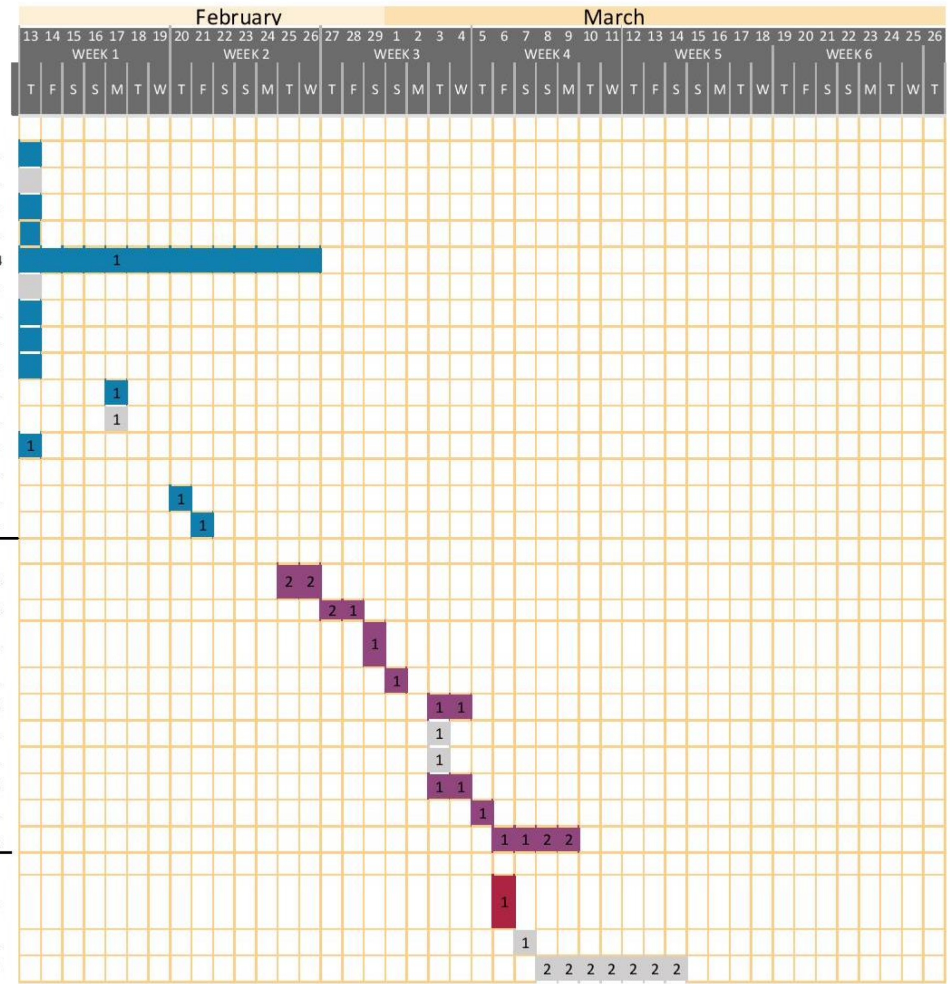
Text hosting application based on Blockcl

Gantt Chart

Group Chain_UP

Project Start Date:	13/2/2020
Project End Date:	17/05/2020

Task	Start	Days
PLANNING PHASE		
Decide what the application is going to be	13/2/2020	1
Define purpose of the project	13/2/2020	1
Decide how the app is going to behave (actions)	13/2/2020	1
Verify privacy regulations for the autentication bit (GDPR / private data	13/2/2020	1
Clearly define the core functionalities of the app [proposal form]	13/2/2020	14
Define the cloud provider to store the data (IBM Cloud)	13/2/2020	1
Define use cases and diagrams	13/2/2020	1
Define blockchain's functionality [with scope] and type (permissioned	13/2/2020	1
Outline how to apply the concept of tokens	13/2/2020	1
Define consensus algorithm and parameters needed	17/2/2020	1
Decide platforms and softwares that support our project	17/2/2020	1
Define the ETCD structure - what will be in the cloud	13/2/2020	1
CognitiveClass course, VSCode tutorial on BC app & labs for info	15/12/2019	1
Define security features - how to?	20/2/2020	1
Define the programing modules on the back-end	21/2/2020	1
DESIGN PHASE		
Design application on blockchain(Distributed system) (Identity, transactions, ledger, network, storage, and events)	25/2/2020	2
Design the use case, preceding diagram(transaction flow architecture)	27/2/2020	2
Design access control capability for the users of the network (Consensus [RAFT], User-registration [backend]	29/2/2020	1
Development of sequence diagrams (submitters)	1/3/2020	1
Design user interface (HTML, CSS & JS)	3/3/2020	2
Design Json File structure	3/3/2020	1
Design the ETCD nodes' connections and storage parameters	3/3/2020	1
Design a README.md file (program manual - user focused)	3/3/2020	2
Deliver the System Design Document	5/3/2020	1
Review the design	6/3/2020	4
DEVELOPMENT PHASE		
Installing prerrequisites VS code, Docker, Docker Compose, Golang, node.js, Java, Git, Virtual Machines and Operating Systems Linux	6/3/2020	1
Installing ETCD on Virtual machine	7/3/2020	1
Develop distributed system on ETCD	8/3/2020	7



Develop the node for store, hashing	15/3/2020	7																															2	2	2	2	2	2	2					
IPFS integration	22/3/2020	7																																						2	2	2	2	2
Develop consensus service[also required for testing purposes]	29/3/2020	7																																										
Modeling the Network	5/4/2020	7																																										
Generating the network on ETCD	12/4/2020	7																																										
Website development (user-interaction)	19/4/2020	5																																										
Interacting HTML and Javascript with ETCD	24/4/2020	7																																										
TESTING PHASE																																													
Install the existing system	1/5/2020	2																																										
Deploy the network	3/5/2020	2																																										
Write tests (intro and end-to-end test)	3/5/2020	3																																										
Evaluate the process of adding nodes to the network (efficiency gains)	6/5/2020	2																																										
Test the data in clusters (efficiency gains)	8/5/2020	2																																										
App will be submmited to end-to-end test (and maybe to human-	10/5/2020	3																																										
Submit the app to end-to-end test > write results on report	13/5/2020	2																																										
Evaluate results versus goals	15/5/2020	1																																										
DOCUMENTATION PHASE																																													
Report writing - chapter 1 Intro	3/3/2020	10																			1	1	1	1	1	2	2	2	2	2														
Review The first chapter of report with supervisior	5/3/2020	1																					1																					
Update the Table of Contents - finally	3/5/2020	2																																										
Put together a scheme for the diagrams (such as appendix, Chapter)	5/5/2020	3																																										
Select the paragraphs written for the report chapters (Review)	5/5/2020	7																																										
Final presentation - ask Supervisor's help	7/5/2020	3																																										
Wrap up the final report itself - final adjustments	10/5/2020	7																																										

					April																														May																										
27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
WEEK 7						WEEK 8							WEEK 9							WEEK 10							WEEK 11							WEEK 12							WEEK 13							WEEK 14							WEEK 15						
F	S	S	M	T	W	T	F	S	S	M	T	W	T	F		S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W