FabNet: an Automatic Hyperledger Fabric Network Wizard

Alessandro Marcelletti[1] and Barbara Re[1]

University of Camerino, Via Madonna delle Carceri 7, Italy
{alessand.marcelletti,barbara.re}@unicam.it

Abstract. Hyperledger Fabric is continually evolving technology, thanks to its active community providing new features and functionalities. However, it still lacks a user-friendly interface that makes its adoption immediate and straightforward. In particular, one of the main barriers for non-expert users is the comprehension and creation of a network. In this paper, we present FabNet, a user-friendly wizard reducing the effort for the configuration and deployment of the Fabric blockchain network.

Key words: Blockchain, Hyperledger Fabric, Network, Configuration

1 Introduction

The rapidly growing of interest in blockchain technologies made IT specialists (es. developers and system administrators) approach this new topic. However, also due to its novelty, they often find this technology hard to learn and master [1]. The scenario is slightly different considering different blockchain platforms. Indeed, in cases of consolidated permissionless blockchains such as Bitcoin [2] and Ethereum [3], the community has been working for many years. This has led to the creation of documentation and also of tools for automatic generation of developing environments. Differently, the permissioned Hyperledger Fabric blockchain is continuously changing with new versions that sometimes modify the core of the technology significantly. Also, for this reason, high-level interfaces still miss, creating an initial obstacle for new people approaching Fabric. Although Fabric has a deep documentation with theory and tutorials, sometimes it is hard to understand concepts without a significant effort from the user. In particular, one of the most challenging technological approaches is the one concerning Fabric network. Indeed the complexity and the amount of files required to create a customised network is very complicated to understand in the first moment. Hyperledger itself provides a configured test network with preconfigured commands; however, it is not easy to customise, and it aims to be only an initial example for developers.

Fabric Composer[1] instead, was born with the idea of providing a high-level environment abstracting from all the low-level details to focus, for example, only on smart contracts implementation. However, the way used in Composer to write chaincodes and to interact with the network through the Software Development Kit (SDK) changed with the new Fabric versions, making the application deprecated since 2019. With our work, we focus only on the network file system, without involving chaincode or APIs, that could make obsolete the entire system with the first change.

Our objective is to address technological difficulties for developers and system administrators. The firsts are end-users that want to create a network for various purposes, without entering in low-level operations, like the creation and the execution of scripts and files. The seconds are developers that instead need to learn how to build a network, in this case, we provide an easy way to accelerate this process, saving a lot of time in the initial approach that, otherwise, can result very complicated.

Already existing payable solutions, such as Blockchain-as-a-Service (BaaS) provided by companies like Oracle[2], Microsoft[3] or Amazon[4] not always provide a Fabric solution, in addition, the network is deployed directly in their cloud. What we want, is to abstract the configuration and the deployment phase giving also the possibility for developers to enter directly inside the system, modifying files or moving the infrastructure on another space, without remaining close to the bought service.

For these reasons, we present FabNet, a free tool useful to configure all the network components just using a high-level interface. This makes the creation of a customised network easy also for those developers and system administrators that does not have experiences in this topic. An easy network configuration is useful especially for starting a project without losing time in undesired configurations. Indeed, sometimes they are not even needed when the main objective is to work on other components like smart contracts and applications. Moreover, having under a unique interface the required parameters can help new developers to understand better the theory and the correlations between the configurations and the generated files. What we provide is a tool capable of deploying both a test and a production network, that automatically creates all the configuration files and the scripts needed to deploy the final network.

The rest of the paper is organised as follows. Section 2 gives an overview of blockchain technology and in particular of Hyperledger Fabric. Section 3 describes the project structure while section 4 shows a short demo of FabNet tool. Finally, Section 5 concludes recapping the motivation.

[1] `https://hyperledger.github.io/composer/latest/`
[2] `https://www.oracle.com/it/application-development/cloud-services/blockchain-platform/`
[3] `https://azure.microsoft.com/it-it/services/blockchain-service/`
[4] `https://aws.amazon.com/it/blockchain/`

2 Background on Hyperledger Blockchain

Blockchain technology was born in the first years with the concept of cooperation between dislocated entities, ensuring a high level of trust between the parties, and provides an immutable way of storing data. With the first permissionless blockchains (Bitcoin, Ethereum) it is possible to make exchanges of assets between its network members. These interactions take place by means of transactions, stored in the form of blocks in a chain along with a timestamp. Each block contains a limited number of transactions and, among other information, the hash of the current block and the hash of the previous block. Data are maintained within a distributed network of mutually untrusted peers, that will be in sync with data changes request, thanks to the consensus mechanism. In the last years, blockchains have adapted to other contexts, with the support of features provided by the so-called smart contracts. These can be considered special programs equipped on each node of the blockchain, which define a transaction protocol that applies the terms of an agreement, between one or more parties involved, enabling auditability between parties [4]. Finally, the execution of smart contracts produces the transactions whose effects are stored in the blockchain. In particular, these kind of systems use an **order-execute architecture** and requires all peers to execute every transaction and all transactions to be deterministic.

Hyperledger Fabric [5] differs from the previous described blockchains, due to its permissioned nature. It is a modular system more versatile for enterprise applications, providing features such as consensus management, private channels and contracts, full-featured programming languages in smart contracts and access control policies. It introduces the execute-order-validate architecture, that allows distributed execution of untrusted code in an untrusted environment. Indeed, Fabric executes transactions before reaching final agreement on their order, then all peers validate transactions in the same order with a deterministic validation. Execute-order-validate paradigm, represents the main innovation in Fabric architecture, making Fabric a scalable system for permissioned blockchains supporting flexible trust assumption.

This kind of system allows to provide flexibility, scalability and privacy in contexts that require these features. The absence of a mining mechanism enables a fast validation and confirmation of transactions. This is achieved thanks to the consensus management system, that allows configuring an arbitrary consensus algorithm, taking into account the requirements of the system intended to implement. Indeed, the arbitrary consensus and the permissioned nature of Fabric guarantee a faster protocol respecting to the Ethereum Proof of Work.

The privacy aspect concerns the confidentiality of transaction and data. This is achieved in Fabric by using its channel architecture and membership service to restrict the distribution of confidential information exclusively to authorised nodes. This is achieved defining channels inside the network in which only a set of chosen nodes can participate. Nodes and channels are also regulated through policies. Consequently, channel's ledger is only accessible to its members and the channel's organisations must approve each peer's membership to the channel. Authentication and identity management are guaranteed through a flexible infrastructure based on the PKI cryptographic scheme and Certificate Authority, which facilitates the joining of a new organisation in the private network. Also, Transport Layer Security cryptographic protocol is used to provide communications security over the network nodes.

The Fabric network is made by different components[5], reported in Fig. 1 and described below.

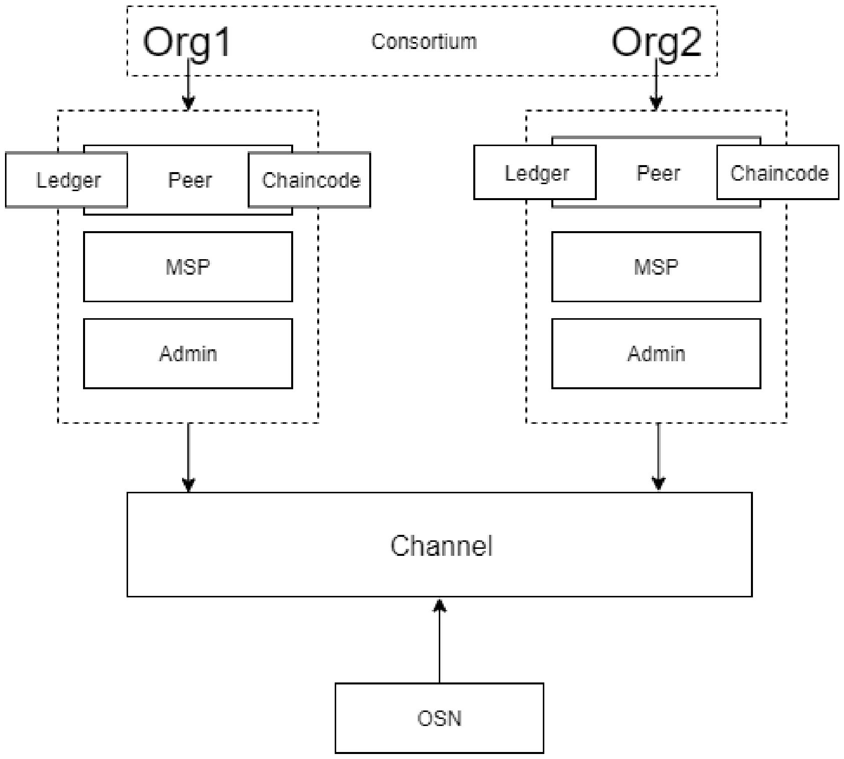


Fig. 1. Example of 2 organisations network components

Peers are the nodes grouped into **organisations**, defined as trust domain in which a peer trusts all peers only within its organisation. Peers execute and/or validate transactions, and maintain the ledger. The group of defined organisations participating in a channel is called **consortium**. The ordering service, composed by a set of **Ordering Service Nodes** (OSNs), establishes consensus and atomically broadcast state updates. The ordering is stateless and decoupled from the peers, it does not take part in the transaction execution and validation process. The **Membership Service Provider** (MSP), maintains the identities

[5] `https://hyperledger-fabric.readthedocs.io/en/release-2.2/key_concepts.html`

of all the nodes (clients, peers, OSNs) inside an organisation. It comprises mechanisms for authenticate transactions, verify the integrity of transactions, sign and validate endorsements, key management and registration of nodes. Smart contracts with system **chaincodes** define the transaction logic and the blockchain settings. They are defined in the channel and stored in each organisation peer. The **ledger** maintains the transactions history, there is one ledger for channel which copies are stored in the organisations peers. In addition, a snapshot of the most recent state is stored in a key-value world state. Finally, the **administrators** have permissions for different operations, like creating and assigning peers, creating network configurations files and modifying policies through system files.

3 FabNet settings

In this section we present the FabNet tool focusing on its architecture and interface. The tool is composed by a SpringBoot back-end in Kotlin and an Angular front-end, available to everyone at http://pros.unicam.it/FabNet/. The main task of the interface is to take inputs from the user and to pass them to the server. These inputs are described in Tab. 3, that contains all the parameters that the user has to fill in order to generate a configuration. From these parameters, the server automatically creates the files needed to create a network. In particular, these files are 51, counting files, scripts and sub-folders; they are related to:

1. Certifications and keys;
2. Peers and orderers containers;
3. Channel profile, defined by different sections:
   a) orderer;
   b) consortium, defining the one creating the network;
   c) consortiums, defining all the consortiums in the network;
   d) application;
   e) capabilities and policies.

These files are filled up with the information passed through different parameters in the interface, listed in Tab. 3.

Once the configuration has been defined, the client will send it as a JSON file to the server, which will generate the related files, returning a zip folder containing them. The sequence flow in Fig. 2 recaps the steps regarding network creation.

It is important to notice that with our approach the user does not have to create manually all the described infrastructure. However, once downloaded, the user is free to modify all the desired parts without any restriction.

3.1 Netowrk deployment

Once the system folder is generated, to deploy the network is necessary to launch the autorun file. This file contains and automatically executes all the commands

Network name		
From 2 to N organisations, for each:	First name Domain Certificate Authority N members, for each:	First name Type IP address and port Status in the certificate
From 1 to N consortiums, for each:	First name Organisations inside	
From 1 N channels, for each:	First name Consortium administrator Organisations inside	

Table 1. Interface inputs fillable by the user.

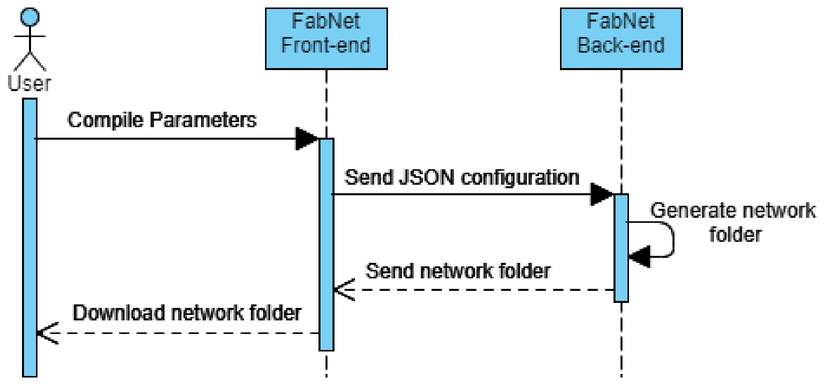


Fig. 2. Sequence diagram of the network configuration and generation

required to deploy the final network, lifting the user from other operations. In particular, the automatised operations for a total of 14 commands can be synthesised in:

1. Download required binaries
2. Start CAs and creates artefacts for each organisation;
3. Create genesis block and channels txs;
4. Start Docker containers for each org;
5. Link the started containers so that they can communicate;
6. Create the channel;
7. Join peers to the channel.

Notably, the single file is used to deploy a local network, for remote peers, some additional steps are required. Indeed, in a distributed context, each organisation should extract, from the generated file system, the folder regarding its org. After this, it is also necessary to run the CA script generating the artefacts. However, we are currently working on a FabNet extension to provide the GUI also for the creation of a distributed network.

4 FabNet in practice

In this section we show a demo of FabNet using a basic testing configuration, describing the flow and the tool functionalities linked with some screens. All the steps described in the following subsection can be replicated using the links at http://pros.unicam.it/FabNet.

4.1 Network scenario

The tool is composed by 4 main panels, the first one, reported in Fig. 3, allows to specify the network name and the organisations involved (name, domain); it is also possible to add other organisations and to import/export JSON configuration. We introduce a brief scenario involving two organisations, org1 and org2, that want to collaborate in their network. The configuration used in this example is available at http://pros.unicam.it/test-network.

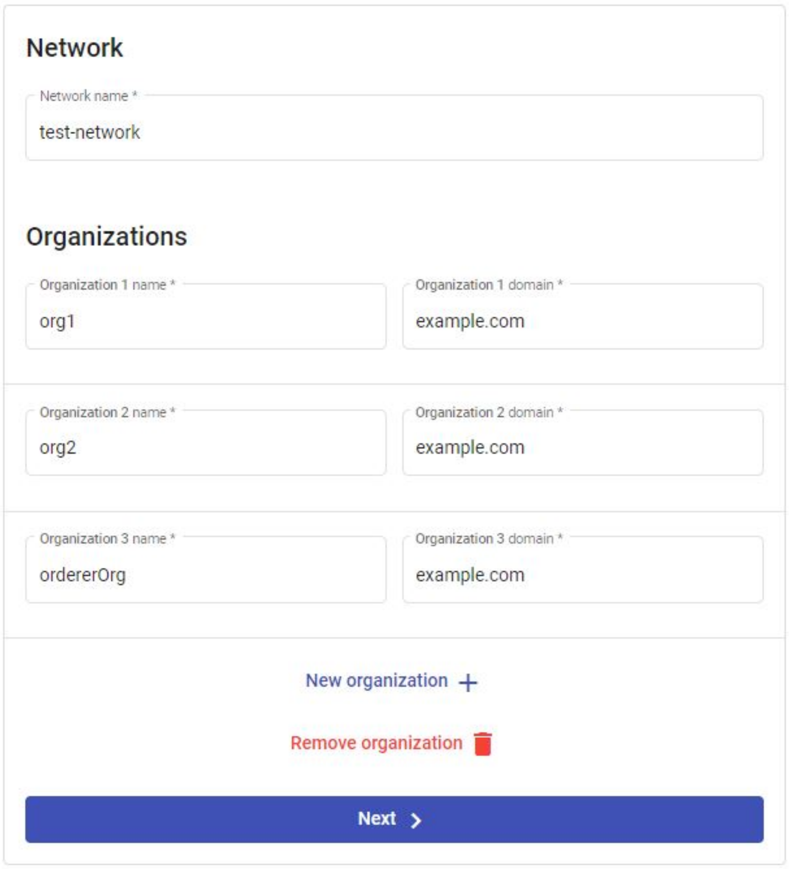
In the tool there is also a panel related to organisations, where for each of them the user has to insert certification authority information and can handle members. In particular, it is possible to add or remove peers, clients, admins and orderers. Depending on the member type, different information is required, like host IP and ports. As soon organisations and network are defined, the consortium must be created. It is possible to create 1 or more consortiums, defining the participating organisations. Finally, it is possible to specify the channel in the network, requiring a consortium and, optionally, external organisations.

If the network configuration is made with local peers, to deploy the entire network is necessary to run only the *start.sh* file (download at http://pros.unicam.it/start/). Instead, if the organisation's members are dislocated at remote addresses, each user has to take his related folder and run the inside scripts. The main focus of our future extension aims at removing also these steps, automatically generating one folder and one start script for each organisation. Fig. 4 shows the network participants once deployed, in particular, Fabric network uses Docker containers to run organisation members. In Fig. 5 instead, the status, addresses and names of members are displayed.

4.2 FabNet applications

The previous network scenario showed a simple scenario with two collaborating organisations but, more in general, FabNet can be used in more contexts. Firstly, it can be used by different type of users with different purposes, like developers that want to approach this new technology or for users that want only to create already usable artefacts.

Also in inter-organisational environments, like the ones analysed in [6], FabNet can be a core element. It helps to have a fast generation of networks with different configurations, very useful for performance and scalability analysis. Indeed, the easy configuration allows to increment the peers and organisations number having more and more networks to test.


Network

Network name *
test-network

Organizations

Organization 1 name *	Organization 1 domain *
org1	example.com
Organization 2 name *	Organization 2 domain *
org2	example.com
Organization 3 name *	Organization 3 domain *
ordererOrg	example.com

Fig. 3. Network and organisations declaration page

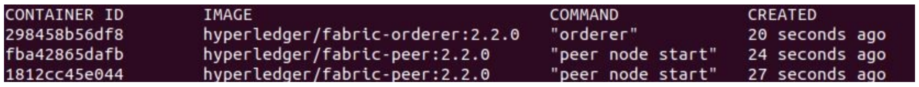

```
CONTAINER ID    IMAGE                              COMMAND              CREATED
298458b56df8    hyperledger/fabric-orderer:2.2.0   "orderer"            20 seconds ago
fba42865dafb    hyperledger/fabric-peer:2.2.0      "peer node start"    24 seconds ago
1812cc45e044    hyperledger/fabric-peer:2.2.0      "peer node start"    27 seconds ago
```

Fig. 4. Docker containers of the deployed network

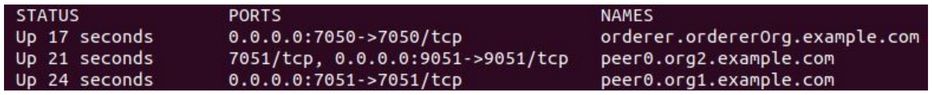

```
STATUS            PORTS                               NAMES
Up 17 seconds     0.0.0.0:7050->7050/tcp              orderer.ordererOrg.example.com
Up 21 seconds     7051/tcp, 0.0.0.0:9051->9051/tcp    peer0.org2.example.com
Up 24 seconds     0.0.0.0:7051->7051/tcp              peer0.org1.example.com
```

Fig. 5. Status of the running peers in the network

Think about companies that decide to work together, creating also a blockchain infrastructure, in this case FabNet suits also the need of prototyping networks. Indeed, in the first phases, designers have to collaborate, modelling the organisations communication and thinking about structuring the network, handling peers and all the other components. In that moment, a tool allowing the creation of more networks for testing different configurations could be very useful.

In this context, developers can work together in a sandbox environment, changing configurations if needed without effort only by creating and running new networks.

5 Conclusions

In this work, we presented FabNet, a tool that aims to help IT specialists to automatise the configuration and the deployment of an Hyperledger Fabric network, thanks to the user-friendly interface that allow configuring files without manually entering into each of them. Our contribution mainly targets novices in the blockchain technology, in particular when they have to deal with the creation of a Fabric network. Indeed, FabNet doesn't require any effort for configurations, delegating the creation and the filling of files to the interface, and automating the generation of artefacts. We also contribute, providing a tool capable of creating and deploying advanced production-ready networks without involving the user with additional tasks. FabNet supports a large scale of usages, starting from benchmarking applications until development ones.

References

1. Liu, X.: A small java application for learning blockchain. In: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), IEEE (2018) 1271–1275
2. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot (2019)
3. Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014) (2014) 1–32
4. Corradini, F., Marcantoni, F., Morichetta, A., Polini, A., Re, B., Sampaolo, M.: Enabling auditing of smart contracts through process mining. In: From Software Engineering to Formal Methods and Tools, and Back. Springer (2019) 467–480
5. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the thirteenth EuroSys conference. (2018) 1–15
6. Corradini, F., Marcelletti, A., Morichetta, A., Polini, A., Re, B., Tiezzi, F.: Engineering trustable choreography-based systems using blockchain. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. (2020) 1470–1479