A tool to easing the configuration and deploying process of Hyperledger Fabric

Domenico Cantone[1], Daniele Francesco Santamaria[1,*] and Vincenzo Spinello[1]

[1]University of Catania, Department of Mathematics and Computer Science, Viale Andrea Doria, 6, 95125, Catania, Italy

Abstract

Hyperledger Fabric is an open-source project for building permissioned blockchains that allow components such as consensus and membership services to be plug-and-play. Thanks to its modular and versatile design, a broad range of industry use-cases can be implemented. However, industries lack for tools to easing the instantiating and configuration process of Hyperledger Fabric, thus making difficult its dissemination. In this paper, we contribute to the Hyperledger Fabric ecosystem by delivering a user-friendly, free-to-use, and open-source tool that enables users to easily configure and deploy a Hyperledger Fabric instance, thus favouring its adoption and spreading.

Keywords

Hyperledger Fabric, Blockchain, GUI, Graphic User Interface

1. Introduction

Hyperledger [1] is a family of open-source projects started in 2015 by the *Hyperledger Foundation*, hosted by the *Linux Foundation* [2]. The main purpose of the project is to provide a set of developing tools for building blockchain-based distributed systems. Many sub-projects derive from Hyperledger, such as *Hyperledger Fabric*, *Hyperledger Besu*, and *Hyperledger Aries*. Among them, particularly relevant for the business and industry realms is the Hyperledger Fabric, a private blockchain conceived both for private and public organizations, which does not bind users to particular requisites thanks to its high extent of customization and configuration. Hyperledger Fabric offers a unique approach to consensus protocols that enables performance at scale while preserving privacy.

One of the limitations of the Hyperledger Fabric is the lack of high-level and user-friendly tools that allow neophyte users to deploy the network and build custom solutions, thus limiting the dissemination of the blockchain and its broad adoption. These premises motivate the effort of the work, currently under development, presented in this contribution.

We introduce the *Hyperledger Fabric GUI* (HFG, in short), a tool based on a graphic user interface that has been developed to tackle the above-mentioned problem by simplifying and hastening the instantiating and configuration process of Hyperledger Fabric, thus minimizing

BIR-WS 2023: BIR 2023 Workshops and Doctoral Consortium, 22nd International Conference on Perspectives in Business Informatics Research (BIR 2023), September 13-15, 2023, Ascoli Piceno, Italy

*Corresponding author.

domenico.cantone@unict.it (D. Cantone); daniele.santamaria@unict.it (D. F. Santamaria); 1000006449@studium.unict.it (V. Spinello)

0000-0002-1306-1166 (D. Cantone); 0000-0002-4273-6521 (D. F. Santamaria)

error risks: indeed, even a simple error, such as providing a wrong parameter or performing some step incorrectly, may lead to a wrong configuration of the network and, consequently, either a re-configuration or a re-instantiating of the network. Thanks to HFG, users are only requested to fill out a configuration form concerning the network, leaving the underlying steps to the configuration scripts provided by the tool. As a main consequence, users are warned of wrong parameters which may lead to the deployment of inconsistent instances of Hyperledger Fabric. In this case, the application automatically deletes the old instance and creates a new one. The presented work is part of a project that aims at delivering a tool for simplifying the deployment and configuration not only of Hyperledger Fabric but also of any other products concerning the Hyperledger ecosystem and of Hyperledger-based applications. The tool is available at [3], both for test and contributions.

The paper is organized as follows. Section 2 introduces some related works; Section 3 provides a general overview of Hyperledger Fabric, focusing on the features that are configured by HFG; Section 4 presents a general and short overview of the steps that should be carried out when instantiating a working instance of Hyperledger Fabric; Section 5 presents the Hyperledger Fabric GUI, illustrating how it simplifies the deployment and configuration of Hyperledger Fabric. Finally, in Section 6, we draw our conclusions and discuss some future research directions.

2. Related Works

There are several projects related to the deployment of Hyperledger Fabric instances.

Hyperledger Caliper, as described in [4], serves as a benchmarking tool for Hyperledger. It provides a set of predefined use cases that enable users to measure the performance of various configurations of a Hyperledger instance. This allows for the comparison of different setups to determine the most efficient one based on the organization's needs. Hyperledger Caliper can be used with Hyperledger Besu, Hyperledger Burrow, Ethereum, Hyperledger Fabric, FISCO BCOS, Hyperledger Iroha, and Hyperledger Sawtooth. Caliper measures various performance metrics related to resource utilization, including transaction latency, transactions per second (TPS), success and failure rates (i.e., successful and failed transactions in a test cycle), among others. It is important to note that Caliper does not offer tools for the rapid deployment of a Hyperledger instance.

Hyperledger Composer, as described in [5], is a suite of collaboration tools designed to simplify the development of Hyperledger applications. It streamlines the process of creating a functional Hyperledger Fabric instance from scratch and enables easy construction of blockchain business networks, smart contracts, and blockchain applications. It is worth noting that this tool has been deprecated since 2019, and there are no plans to reintroduce its functionalities.

Distributed Ledger Performance Scan (DLTG) is a tool designed to facilitate the instantiation of new Hyperledger instances. It currently supports various blockchain platforms, including Hyperledger Fabric, Ethereum with the Geth and Parity clients, Hyperledger Indy, Quorum with RAFT and IBFT consensus, and Hyperledger Sawtooth. DLTG distinguishes itself from HFG in that it creates an additional Linux virtual machine on the host, which contains all the necessary packages to initiate a Hyperledger instance. DLTG is primarily intended for creating prototype networks for testing purposes, while HFG is used for the instantiation and configuration of fully

operational Hyperledger Fabric instances. Additionally, DLTG can be utilized for benchmarking the performance of these prototype networks.

In [6], the authors introduce a Hyperledger Fabric Network instantiation tool called Fabnet. Fabnet retrieves the required binary files for starting certificate authorities from a remote server and then transfers them to the requesting host. Once all the necessary files have been received on the host machine, the user must execute an initialization script to instantiate Hyperledger Fabric. In contrast, HFG downloads all the essential files directly onto the user-specified host and executes them in the correct sequence. Notably, among these files, HFG utilizes configuration scripts that can be edited in real-time by the user through a user-friendly interface.

3. Preliminaries

Hyperledger Fabric stands out from other platforms due to its inherent modularity and flexibility. Hyperledger Fabric provides many customization options about every single aspect of the network; for instance, the choice of the consensus algorithm has been made feasible by the implementation of the *Pluggable Consensus Protocol* [7]. In this way, organizations can choose the consensus algorithm that better fits their own needs. Usually, organizations that adopt Hyperledger Fabric choose to create *consortia* in order to exchange information in a secure way. A *consortium* is an environment formed by organizations sharing common goals that chose to collaborate. The communication among organizations of the same consortium is performed via *channels*. In order to control the agents that can join a consortium and how the information is exchanged within consortia, Hyperledger Fabric introduces a consortium administrator. Moreover, Hyperledger Fabric provides organizations with many cryptographic mechanisms to secure the exchange of information through such channels. In particular, thanks to channels, Hyperledger Fabric is equipped with an execution environment allowing Turing-complete programs called *smart contracts* [8]. Smart contracts can be programmed in Hyperledger Fabric by exploiting the most common programming languages such as Java and Python.

Another primary advantage of permissioned blockchains such as Hyperledger Fabric is that native cryptocurrency is not required to utilize transactions; hence, all transactions are performed without paying any cost. The modularity of this network—thanks to consortia and channels—, the expressivity of smart contracts, and cost-free transactions make Hyperledger Fabric practical in many business domains. This is the case, for example, when *tokenizing* and tracking physical assets: organizations involved in such type of business activities can make up a consortium and monitor all asset movements, thus minimizing fraud risks. Hyperledger Fabric is also suitable for building environments where digital identities are of essential relevance, thanks to a joint collaboration among organizations that aims at defining *certificate authorities*.

A certificate authority is a trusted entity issuing digital certificates for the Hyperledger Fabric's participants. We can distinguish two main types of certificate authority (in short, CA):

- The *TLS CA* that releases TLS certificates. TLS [9] is a transport layer security protocol that exploits complex cryptographic security measures to guarantee secure communications through the network. TLS certificates are used in Hyperledger Fabric to secure communications among the entities of the Hyperledger Fabric. Although TLS CAs are not

mandatory to build a working instance of the network, they are strongly recommended since they permit to reduce the success rate of cyber-attacks.
- *Enrollment CAs* release certificates for different types of identities and roles on the network: such certificates allow entities to join Hyperledger Fabric by attesting their identities. As a permissioned blockchain, Hyperledger Fabric requires that entities are authorized beforehand to join the network: these certificates enable entities to see transactions or to perform some specific operations, such as executing smart contracts or adding new identities to the network.

A correct set-up of CAs is one of the necessary condition to guarantee both a secure deployment of the Hyperledger Fabric instance, hence of the ledger, and a network satisfying the organization's needs. This process can result particularly risky due both to the high number of steps that have to be executed through a command-line interpreter and to the files that must be manually edited. For this reason, affordable tools that minimize the risk of errors are of critical importance.

4. Manual instantiating of Hyperledger Fabric

After recalling how Hyperledger Fabric is manually instantiated, in this section, we present a demonstration of how the Hyperledger Fabric GUI significantly simplifies this procedure.

To instantiate Hyperledger Fabric, as a first step it is required to install and download the Hyperledger Fabric binary files and prepare a hierarchy of folders, as depicted in Figure 1.

Then, the certificate authorities must be deployed by typing in the shell the following two commands, the first one for registering the CA administrator in the database and the second one to enroll it:

1. **./fabric-ca-client register -d –id.name [USERNAME]**
   –id.secret [SECRET] -u [CA_URL] –mspdir [CA_ADMIN]
   –id.type [ID_ROLE] –tls.certfiles [TLS_CERT]

2. **./fabric-ca-client enroll -d -u**
   https://[ENROLL_ID]:[ENROLL_SECRET] @[CA_URL]:[PORT]
   –mspdir [MSP_FOLDER] –csr.hosts [CSR_HOSTNAME] –tls.certfiles [TLS_CERT]

The first command requires the username and the password of the administrator of the CA, the URL of the server, the directory where to store the CA, the role of the administrator, and the TLS certificate for enabling secure communications. The second command requires the username and the password of the identity to be added, the address of the CA server, and the related port.

Moreover, the registration and enrollment processes require the configuration of the trusted entity (here omitted for space reasons). Such configuration is carried out by editing the configuration files associated with each certificate authority. Since there are two types of certificate authority, and there can be more than one certificate authority, the user is called to manually edit a certain number of files.

For security reasons, it is recommended that users first create the TLS certificate authority. Then, once the administrator of the TLS CA is enrolled, the TLS CA can be started by executing

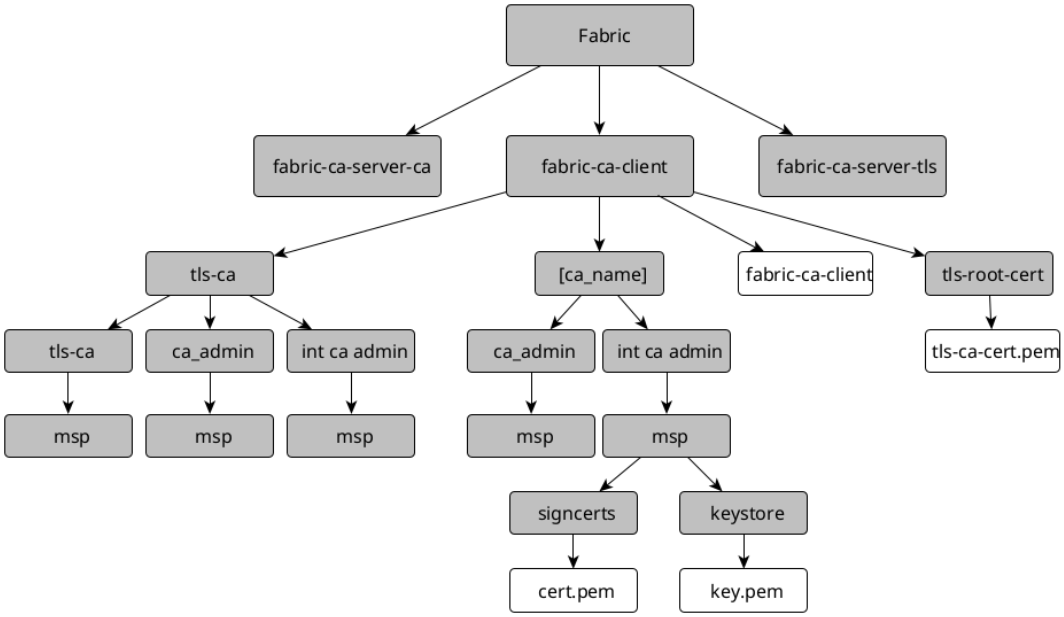

Figure 1: Folder hierarchy required to start up a Hyperledger Fabric instance

the related shell binary file. At this point, the TLS CA is listening on the port specified by the user and is ready to release the TLS certificates, and from now on the user can create all the desired root enrollment CA.

The instantiating process for the root enrollment CA is similar to the one described above, but requires different configuration files. Moreover, the administrator of the enrollment CA requires the TLS CA. Therefore, there are four requests to be formed in this case: two requests for the release of the TLS certificate and two more request for the enrollment of the administrator identity.

In order to release the TLS certificate, a registration and enrollment request must be sent to the TLS CA server, by providing the administrator credentials. Once the administrator's TLS certificate is obtained, the root enrollment CA can be initialized. Subsequently, the administrator can be enrolled by specifying the TLS CA certificate obtained so far. Henceforth, the user can deploy all the desired intermediate enrollment CAs, according to the configuration provided in the previous steps. To deploy an intermediate CA, it is required an additional registration and enrollment request for each CA. In this case, a total of six requests are required: two requests for obtaining the TLS certificate, two requests for enrolling the intermediate CA administrator to the root CA, and two requests for enrolling it to the intermediate CA. Moreover, the setup of the configuration files must be coherent with each other, otherwise unexpected behaviors may occur. Finally, in addition to the configuration files to edit and the registration and enrollment requests to perform, some environment variables must be set as well, to store the paths of the binary files in the machine. Note that when an incorrect request is performed (for instance, if the user provides an incorrect domain for the TLS certificates), all the subsequent steps need to

be executed again, resulting in a considerable waste of time. Moreover, all the files generated during the deployment phase are to be arranged in a specific order.

5. Hyperledger Fabric GUI

To simplify and speed up the deployment of Hyperledger Fabric, as well as to minimize user errors that can arise from a manual deployment, we developed a tool called Hyperledger Fabric GUI, which automatizes the creation of three certificate authorities, two of *Enrollment* type and one of *TLS* type, with a minimum user effort. The tool is released as a web application that connects the user with the Hyperledger Fabric instance through the Secure Shell (SSH) protocol (see Figure 2).

The application, written in PHP, requires only that users provide the Hyperledger Fabric instance connection information to initialize it. Secure communication is established between the application and the instance thanks to the SSH protocol. The SSH protocol, which nowadays replaces Telnet, has been conceived to provide the user with the means for accessing a remote server through port 22, guaranteeing confidentiality and security. The SSH credentials are securely submitted via the HTTPS protocol to the web application in order to establish a connection with the remote machine and are neither stored, nor used anymore. Then, the application executes on the remote servers three different scripts that are customized by users through the web interface, one for each certificate authority:

- the first script creates the TLS certificate authority, which is required to start up a Hyperledger Fabric instance;
- the second script creates the root enrollment certificate authority; finally,
- the third script creates the intermediate certificate authority.

Initially, the three scripts contain *placeholders*, namely fragments of text (non-syntactically belonging to the script language) that is suitably replaced by script code once the user is invoked to decide how such substitution must be performed. Specifically, users provide a set of parameters through the web interface that replace the script placeholders once they are checked for correctness. Then, the scripts are performed in sequence so that the related certificate authorities can be installed. For example, Figure 3 shows a script with placeholders (on the top) and the resulting script (on the bottom) once the placeholders have been replaced.

The resulting scripts are then executed on the remote server, thus completing the instantiating of Hyperledger Fabric with no additional effort from users.

The graphic component of the web application consists of two distinct sections organized on two subsequent pages. The first section (see Figure 4) is devoted to the connection information that allows the application to connect with the remote server, and where the Hyperledger Fabric instance should be installed. This information includes the IP address, the port number (by default, the SSH port is identified by the number 22), the username and password used to access the host.

The second section is focused on setting up the placeholders of the three scripts (see Figure 5). This step is done through four main subsections. The first step requires the input of the host

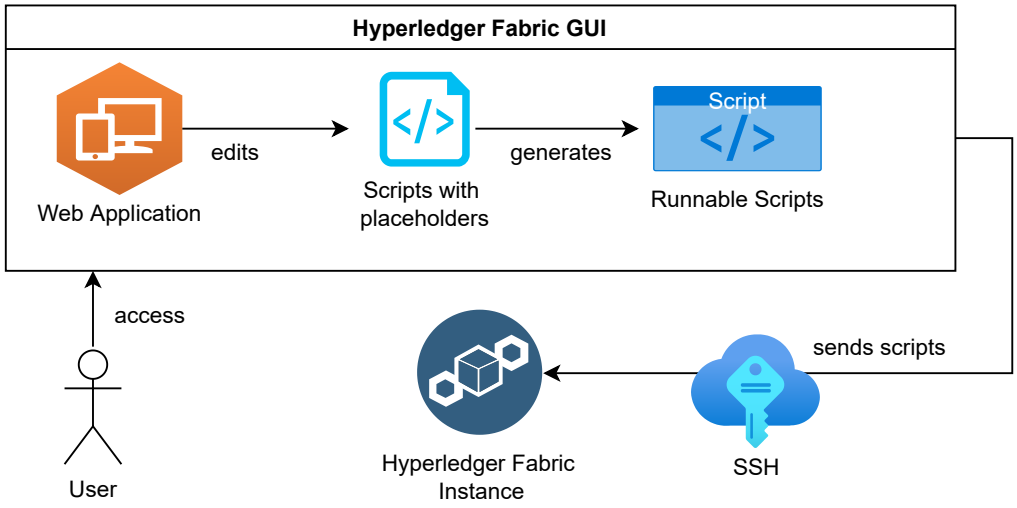

Figure 2: General view of the Hyperledger Fabric GUI

name of the target host, while the remaining three are devoted to the certificate authorities as follows.

TLS certificate authority section. This section is devoted to the creation of the TLS certificate, namely the certificate that enables secure communication channels through the network. The user is called to provide (a) the username and the password of the TLS CA administrator, (b) the port number the TLS CA server will listen on, and (c) the name and an alias name for the TLS CA server. The port number must be unique and must not be used by any other process.

Enrollment certificate authority section. This section requires the information of the enrollment certificate authority that releases the certificates responsible for the assignment of identities. The user is called to input (a) the username and password of the enrollment CA administrator, (b) the port number the enrollment CA will listen on, (c) the name of the enrollment CA, and (d) the so-called "Validity domain of the certificates issued by CA". The latter field contains all the host names for which the certificates released by the authority are valid. The user can also use the "`*`" symbol to indicate a set of domains. For instance, `156.146.180.*` denotes all the hosts whose IP address starts with `156.146.180`. The last field is devoted to the CA's tree height, namely, how long the chain of trust can be. By inputting the value 0. the enrollment CA cannot have children, hence the only authority able to release certificates is the root. By default, such a value is set to 1.

Intermediate enrollment certificate authority section. This section is analogous to the previous one and concerns the intermediate enrollment certificate authorities. The height of the CA tree is omitted in this section because it can be inferred from the previous one.

```
#Go into home directory
cd $HOME

#Hyperledger Fabric Home variable
HFH=Hyperledger_Fabric_Network/Fabric

#create the required folders and download the binary
mkdir Hyperledger_Fabric_Network
mkdir Hyperledger_Fabric_Network/Fabric
curl --output-dir ./Hyperledger_Fabric_Network/ -sSLO https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh && chmod +x Hyperledger_Fabri
./Hyperledger_Fabric_Network/install-fabric.sh docker samples binary
mkdir $HFH/fabric-ca-client
mkdir $HFH/fabric-ca-client/org1-ca && mkdir $HFH/fabric-ca-client/tls-ca && mkdir $HFH/fabric-ca-client/tls-root-cert
mkdir $HFH/fabric-ca-server-tls

#copy the binaries into the related folders
cp Hyperledger_Fabric_Network/fabric-samples/bin/fabric-ca-client $HFH/fabric-ca-client
cp Hyperledger_Fabric_Network/fabric-samples/bin/fabric-ca-server $HFH/fabric-ca-server-tls

#initialize the network
./$HFH/fabric-ca-server-tls/fabric-ca-server init -b [TLS_INIT_ADMIN_USER]:[TLS_INIT_ADMIN_PASSWD]

#edit the configuration file of the tls ca server
sed -i 's/port: 7054/port: [TLS_CA_SERVER_PORT]/' ./$HFH/fabric-ca-server-tls/fabric-ca-server-config.yaml #changes listening port of the tls ca server
sed -z -i 's/# Enable TLS (default: false)\n  enabled: false/# Enable TLS (default: false)\n  enabled: true/' ./$HFH/fabric-ca-server-tls/fabric-ca-server-config.yaml
sed -i -z 's/# Name of this CA\n  name:/# Name of this CA\n  name: [TLS_CA_NAME]/' ./$HFH/fabric-ca-server-tls/fabric-ca-server-config.yaml
sed -i -z 's/    hosts:/    hosts:\n      - [TLS_CA_SERVER_HOSTS]/' ./$HFH/fabric-ca-server-tls/fabric-ca-server-config.yaml
sed -i -z 's/      ca:\n         usage:\n           - cert sign\n           - crl sign\n         expiry: 43800h\n         caconstraint:\n           isca: true\n

#remove the old certificate and the msp folder

rm -r ./$HFH/fabric-ca-server-tls/msp
rm ./$HFH/fabric-ca-server-tls/ca-cert.pem

#start the TLS CA Server
```

```
#Go into home directory
cd $HOME

#Hyperledger Fabric Home variable
HFH=Hyperledger_Fabric_Network/Fabric

#create the required folders and download the binary
mkdir Hyperledger_Fabric_Network
mkdir Hyperledger_Fabric_Network/Fabric
cd Hyperledger_Fabric_Network
curl -sSLO https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh && chmod +x install-fabric.sh
./install-fabric.sh samples binary
cd $HOME
mkdir $HFH/fabric-ca-client
mkdir $HFH/fabric-ca-client/CA && mkdir $HFH/fabric-ca-client/tls-ca && mkdir $HFH/fabric-ca-client/tls-root-cert
mkdir $HFH/fabric-ca-server-tls

#copy the binaries into the related folders
cp Hyperledger_Fabric_Network/fabric-samples/bin/fabric-ca-client $HFH/fabric-ca-client
cp Hyperledger_Fabric_Network/fabric-samples/bin/fabric-ca-server $HFH/fabric-ca-server-tls

#initialize the network
cd $HFH/fabric-ca-server-tls
./fabric-ca-server init -b tls_admin:tls_pass

#edit the configuration file of the tls ca server
sed -i 's/port: 7054/port: 8501/' fabric-ca-server-config.yaml #changes listening port of the tls ca server
sed -z -i 's/# Enable TLS (default: false)\n  enabled: false/# Enable TLS (default: false)\n  enabled: true/' fabric-ca-server-config.yaml #activates the TLS protocol
sed -i -z 's/# Name of this CA\n  name:/# Name of this CA\n  name: TLS_CA/' fabric-ca-server-config.yaml
sed -i -z 's/    hosts:/    hosts:\n      - alternative/' fabric-ca-server-config.yaml  #This line is still to be fixed
sed -i -z 's/      ca:\n         usage:\n           - cert sign\n           - crl sign\n         expiry: 43800h\n         caconstraint:\n           isca: true\n

#remove the old certificate and the msp folder

rm -r msp
rm ca-cert.pem

cd $HOME
```

Figure 3: An example of script with placeholders (on the top) and the corresponding script after the placeholder substitution (on the bottom)

It is worth observing that the port numbers provided for the CA must differ from each other. Additionally, the user does not have to arrange all the files necessary for the deployment process, or to declare the environmental variables, or to modify any configuration file: all these tasks are automatically performed by HFG. Moreover, in order to create a new instance of the Hyperledger Fabric, the old instance can be removed and the tool executed again with the new parameters.

5.1. Work in progress

Hyperledger GUI has been recently used for the configuration of *EtnaLedger* [10], a work-in-progress project aiming at building an affordable and scalable Hyperledger Fabric network for

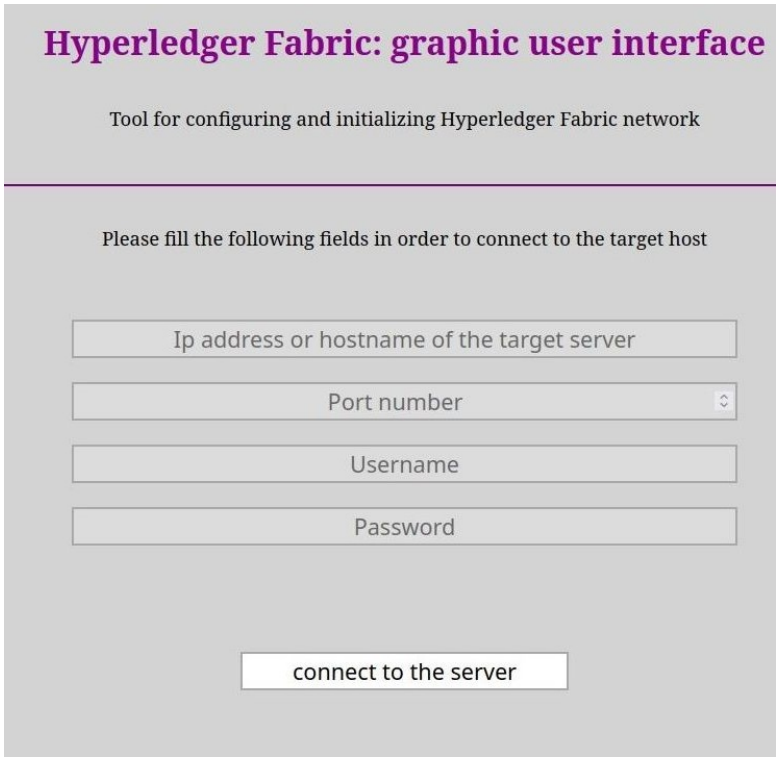

Figure 4: The first section of the Hyperledger Fabric GUI

the Italian public administration and enterprises. The network is organized into three main levels. The first level is managed by the University of Catania, which is responsible for adding the principal authorities, constituted by the partner organizations. The second level is made up of the principal authorities that are allowed to enroll any participants in the network. The last level is formed by the contributors enrolled by the EtnaLedger partners: contributors such as small, medium, and large enterprises are allowed to publish smart contracts, use the blockchain to run their own business, or even act as validating nodes, depending on their means.

6. Conclusions and future directions

In this contribution, we presented a tool for easing and speeding up the process of instantiating *Hyperledger Fabric* in a secure way. The tool utilizes a GUI to reduce the potential for errors for users in editing and configuring the Hyperledger Fabric instance, and resetting the configuration in case of unrecoverable errors. The tool, called Hyperledger Fabric GUI, is framed in a more rich context that includes many feature enhancements since it aims at being a means for fully managing multiple instances of Hyperledger Fabric. Among future extensions of HFG, we shall consider the introduction of an SQL database to store the settings of all the Hyperledger Fabric components such as peers, MSPs, and channels. We also intend to include a set of tools for monitoring the state of the network together with the capability to restore previous

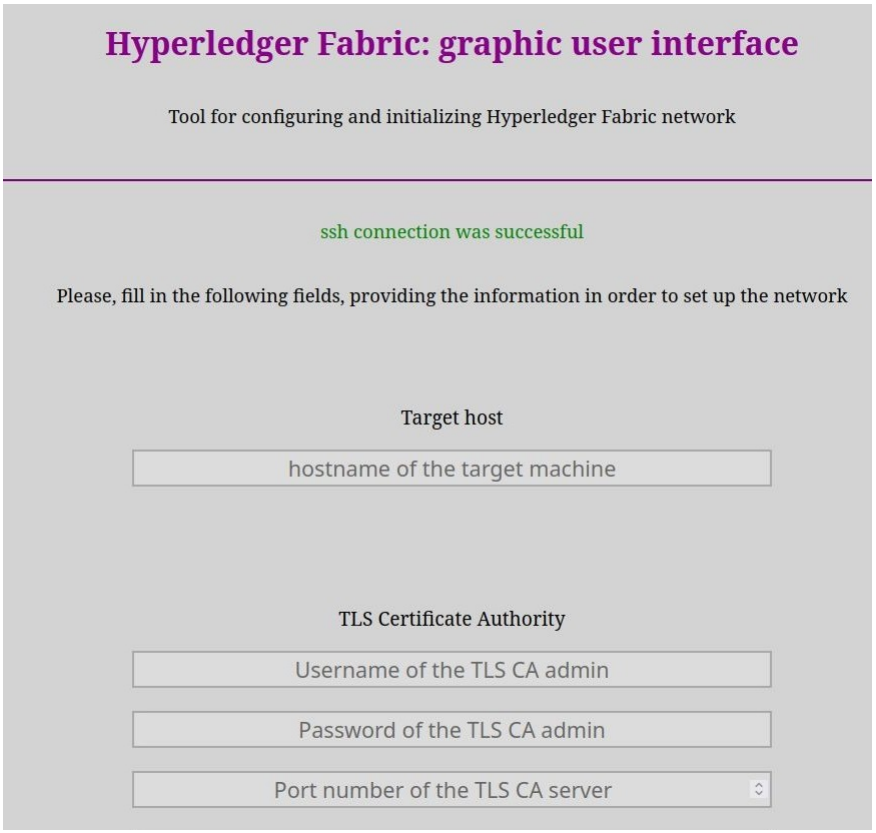

Figure 5: The second section of the Hyperledger Fabric GUI

configurations. Moreover, we shall take into account the integration with the *Interplanetary File System* (IPFS) protocol [11] in order to enable Hyperledger Fabric smart contracts with capabilities for managing files. Finally, an integration of Hyperledger Fabrics with ontological model as in [12, 13] is on-going.

References

[1] The Hyperledger Foundation, The Hyperledger Project, https://www.hyperledger.org/, 2015.
[2] The Linux Foundation, The Linux Foundation, https://www.linuxfoundation.org/, 2000.
[3] V. Spinello, D. F. Santamaria, The Hyperledger Fabric GUI, https://github.com/vinc-00/Hyperledger-Fabric.git, 2022.
[4] The Hyperledger Foundation, Hyperledger Caliper, https://www.hyperledger.org/projects/caliper, 2015.
[5] The Hyperledger Composer Project, Hyperledger Composer, https://hyperledger.github.io/composer/latest/index.html, 2017.
[6] F. Corradini, A. Marcelletti, A. Morichetta, A. Polini, B. Re, F. Tiezzi, Engineering trustable

choreography-based systems using blockchain, in: C. Hung, T. Cerný, D. Shin, A. Bechini (Eds.), SAC '20: The 35th ACM/SIGAPP Symposium on Applied Computing, online event, [Brno, Czech Republic], March 30 - April 3, 2020, ACM, 2020, pp. 1470–1479. doi:10.1145/3341105.3373988.

[7] The Hyperledger Foundation, Hyperledger Architecture, Volume I, https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf, 2017.

[8] The Hyperledger Foundation, Hyperledger Architecture, Volume II, https://www.hyperledger.org/wp-content/uploads/2018/04/Hyperledger_Arch_WG_Paper_2_SmartContracts.pdf, 2018.

[9] T. Dierks, E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246 (2008) 1–104.

[10] University of Catania, EtnaLedger: HyperLedger Fabric for Public Administration and Enterprises, https://www.dmi.unict.it/santamaria/projects/etnaledger/call.pdf, 2023.

[11] Protocol Labs, The Interplanetary File Systems (IPFS), 2013. https://ipfs.io.

[12] D. Cantone, C. F. Longo, M. Nicolosi-Asmundo, D. F. Santamaria, C. Santoro, Ontological Smart Contracts in OASIS: Ontology for Agents, Systems, and Integration of Services, in: In Camacho et al. (eds.), Intelligent Distributed Computing XIV, Studies in Computational Intelligence 1026, Chapter 22, 2022, pp. 237–247. doi:10.1007/978-3-030-96627-0_22.

[13] G. Bella, D. Cantone, C. Longo, M. Nicolosi-Asmundo, D. F. Santamaria, Blockchains through ontologies: the case study of the Ethereum ERC721 standard in OASIS, in: In D. Camacho et al. (eds.), Intelligent Distributed Computing XIV, Studies in Computational Intelligence 1026, Chapter 23, 2022, pp. 249–259. doi:10.1007/978-3-030-96627-0_23.