

**Managing and Deploying Applications on a Quorum blockchain network**  
**20.10.2020**

Dr. Enis Karaarslan, MSKU  
Umutcan Korkmaz, MSKU  
Melih Birim, Tubu.io  
Şafak Öksüzer, Tubu.io

**Table of Contents**

<b>Introduction</b>	<b>2</b>
<b>Node Setup</b>	<b>3</b>
Node Install	3
Node Test	4
<b>Create a Decentralized App</b>	<b>6</b>
Connect to an Available Network	6
Create a new application	6
Deploy Contract	7
<b>Interact with Smart Contract</b>	<b>10</b>
Share Application	11
Create Api Key	11
Deploying New Version	12
Accessing Contract Details	13

Version Notes:

- V 1.0 - 20.10.2020

Latest version of this document can be obtained at the project web site:

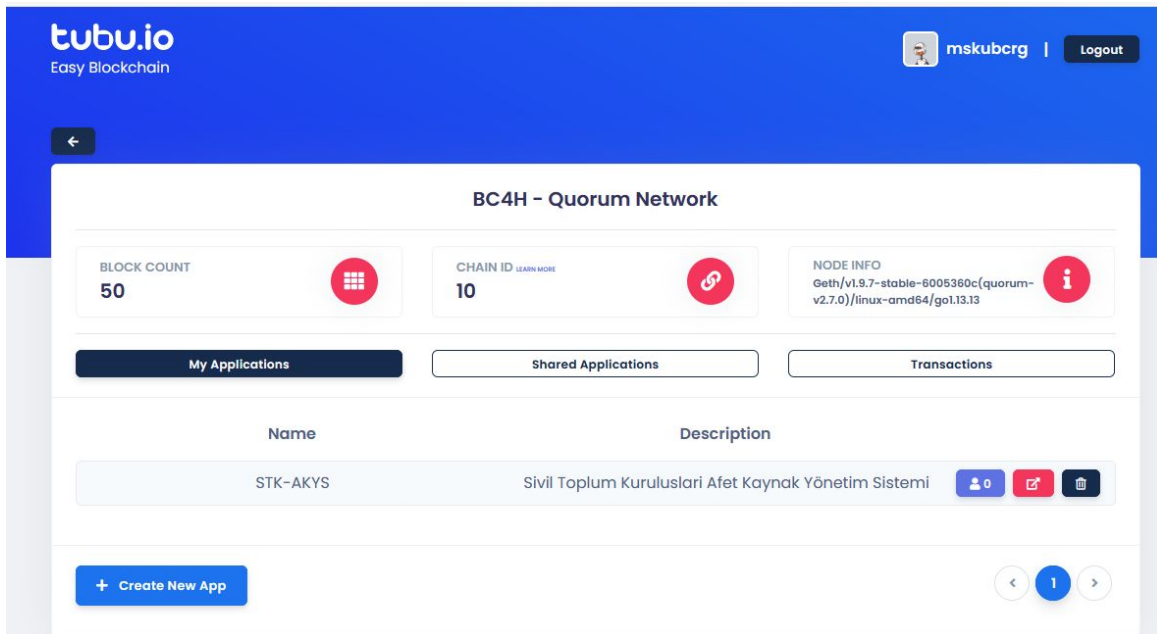
[http://wiki.netseclab.mu.edu.tr/index.php?title=Decentralized\\_Solutions\\_for\\_Humanity\\_\(DS4H\)\\_Blockchain\\_Network\\_Project](http://wiki.netseclab.mu.edu.tr/index.php?title=Decentralized_Solutions_for_Humanity_(DS4H)_Blockchain_Network_Project)

# 1. Introduction

This report will include the details to deploy a DS4H similar blockchain network:

- Node setup
- Creating a decentralized application
- Interact with Smart Contract

We deployed our Decentralized Resource Management System (NGO-RMSD) ( STK-AKYS Sivil Toplum Kuruluslari Afet Kaynak Yönetim Sistemi) on the DS4H network by using the tubu.io platform.



This platform is preferred for its user friendly interface and to manage the projects easily. The details of installing a blockchain network and using this platform is given in this report.

A sample contract in the environment is as follows:

**tubu.io**  
Easy Blockchain

mskuberg | Logout

←

**BC4H - Quorum Network - STK-AKYS - c349748166ae4e33 - v1.0**

**Requests**

TODAY 0 WEEK 0 MONTH 0 QUARTER 0 YEAR 0

**Versions**

Description	Tag	
First Draft	v1.0	Current

**Methods**

Find Method

**Methods**

Find Method

[SEND] approveNeed

[SEND] approveSupport

[SEND] cancelNeed

[SEND] cancelSupport

**Inputs**

Name: \_needID Type: string

**Accounts** [+ Add Account](#)

Name	Address	Status	
Main	0xFF0Af5432478cA21c05e1A0A68e044Bd5F99D2f8	✓	Details
NGO 1	0x3cB67087908fB2681D255e50a009142f88a7fd30	✓	Details

To learn how to call smart contract functions please visit the [docs](#).

## 2. Node Setup

Nodes will be installed on sites and will work autonomously. However, still there should be a point of contact; to reboot the device or issue basic commands for the network connectivity when needed.

Node which will run the blockchain network are ordinary PC Hardware whose hardware specs are (minimum):

- 4 core CPU
- Min 8 gb RAM,
- Sufficient disk (Min 512 GB)

The operating system (OS):

- ubuntu 16.04+
- preferably no GUI

Network:

Static IP address (the network IP addresses should be static and known, so that further security measures can be taken

The following TCP ports (at firewall) should be open (inbound/outbound) for the network communication

- Consensus Port : 50405
- P2P Port : 30305
- WS Port: 8545
- RPC Port: 8645

Software

- Docker
- Quorum

## Node Install

As machine name and user in setup;

- Machine name; DS4H-00X (where X will be the id given to you)
- It is recommended to make "DS4H-00x" for the username "username" that you use with root right.

Make all updates

To make the hostname definition later:

- hostnamectl
- sudo hostnamectl set-hostname DS4H-00X (where X will be your given id)

Disable sleep

- `sudo systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target`

### Static IP Address identification

It will be necessary to make a static definition in the "01-network-manager-all.yaml" file for ubuntu on devices purchased from the project.

Guide:

<https://linuxize.com/post/how-to-configure-static-ip-address-on-ubuntu-18-04/#:~:text=Configuring%20Static%20IP%20address%20on%20Ubuntu%20Desktop,-Setting%20up%20%a%26text=Click%20%20ten%20cog%20icon,ten%20the%20%20E2%2080%209CAppl%20E2%2080%209D%20button>

- Docker setup
  - `sudo apt update`
  - `sudo apt install apt-transport-https ca-certificates curl software-properties-common`
  - `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
  - "`sudo nano /etc/apt/sources.list.d/additional-repositories.list`" and add
 

```
deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable
```
  - `sudo apt update`
  - `apt-cache policy docker-ce`
  - `sudo apt install docker-ce`
  - `sudo systemctl status docker`
  - `sudo usermod -aG docker ${USER}`
  - `su - ${USER}`
  - `id -nG`
  - `sudo usermod -aG docker USER-NAME`
- docker control
  - `docker ps`
- Make and Install Quorum setup scripts
  - Quorum version 2.7.0 is used.
  - Geth is configured with the specified ports and IP addresses of the specified nodes

## Node Test

- Ensure geth is working  
`Ps -efl | grep geth`
- `./attach.sh console`  
To see our four p2p nodes:

> `raft.cluster`

```
raft.cluster
[{
  hostname: "194.27.153.191",
  nodeActive: true,
  nodeId:
  "ae5066f7519128d467b07913cce08f21bb0da33f231b949262c31bfc0cddb52b6a17fe27a69d37ff8
  ed85b7b5f95bfee8f3842aa1c2dff445e247747a2a6ea9",
```

```
    p2pPort: 30305,
    raftId: 1,
    raftPort: 50405,
    role: "verifier"
}, {
    hostname: "193.140.28.224",
    nodeActive: true,
    nodeId:
"108a88e19db7c27f0d774cd1fd674eafbb76ce1692125d8971f6c996601f5eec7fe0e68c51d07645
8264f0d6d60d9f8f4cfb0abc3e9aa12f1025d09c659224ff",
    p2pPort: 30305,
    raftId: 3,
    raftPort: 50405,
    role: "verifier"
}, {
    hostname: "34.65.212.13",
    nodeActive: true,
    nodeId:
"41574083e9a74fe82110ca942179a51c090a085e89d3a02c7b872fdb43e70eef9e2d34b3de79fd0
42549ef2deec64a4cb219c0b3404f73afa8bbf9172da0c241",
    p2pPort: 30305,
    raftId: 4,
    raftPort: 50405,
    role: "minter"
}, {
    hostname: "79.123.237.60",
    nodeActive: true,
    nodeId:
"2de4bdc1d5632460edbbb7ef3dd47049bcc5c3abf4a6be3e88e393144dfa520e7fe0165bafd78149
e15872f3fdf6aa26eb16d91291e748827da74dfbf0fa6753",
    p2pPort: 30305,
    raftId: 2,
    raftPort: 50405,
    role: "verifier"
}]
```

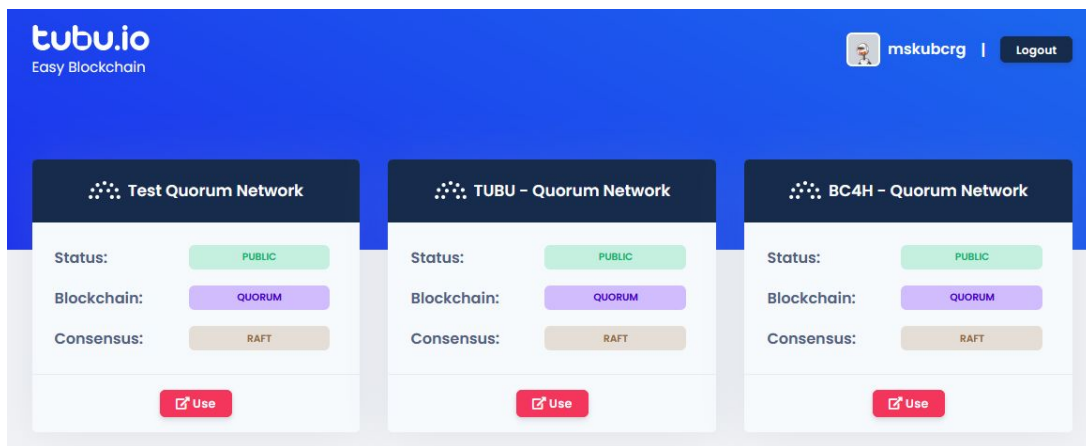
### 3. Create a Decentralized App

Creating the decentralized App (Dapp) is a really simple flow. All you have to do is to follow the steps below:


- Connect to an Available Network
- Create a new application
- Deploy Contract

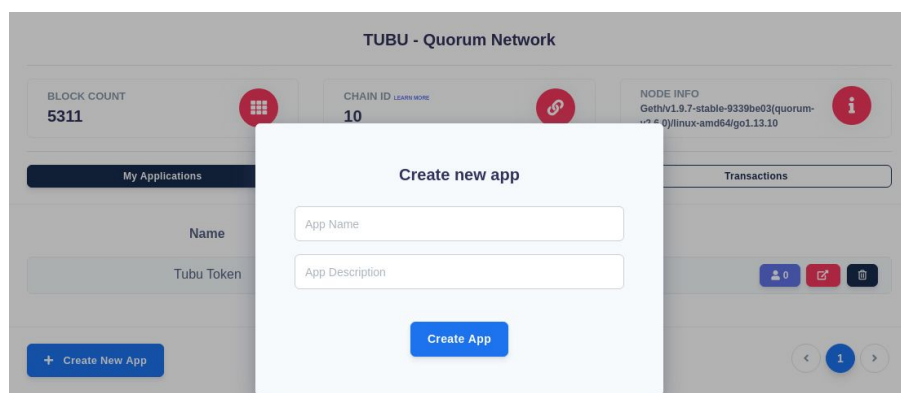
#### Connect to an Available Network

Tubu.io platform serves different networks with different blockchain versions and ledgers. We created and used the “BC4H” quorum network. Appropriate network or the test network can be selected and the “use” button of that network can be clicked to continue with the process.

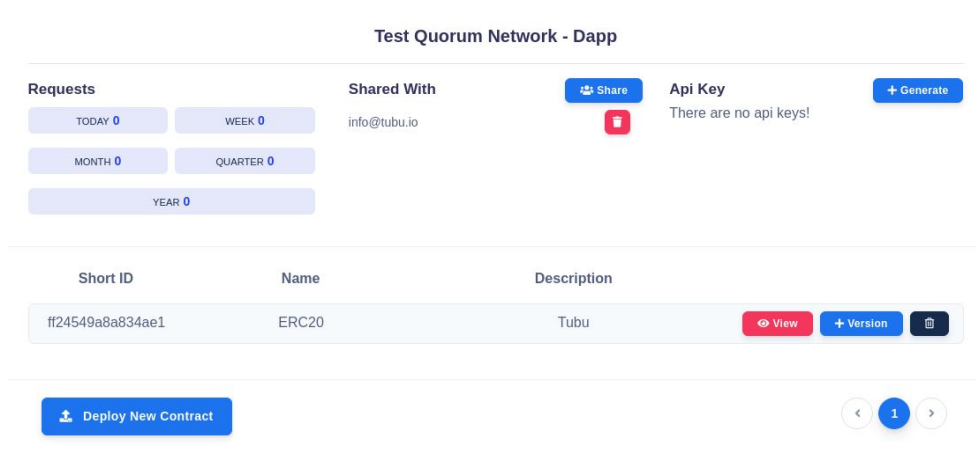


#### Create a new application

In this step, you are expected to create a new application or select one from the listed existing applications. Click the detail button  to carry on with an existing application

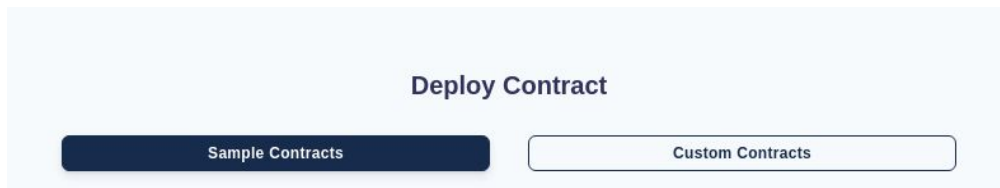


After you create and select your desired application, you will face the application details screen. In this stage, you can deploy a brand new smart contract by clicking deploy new contract with “Deploy new contract” button or select an existing one by clicking the “view” button.

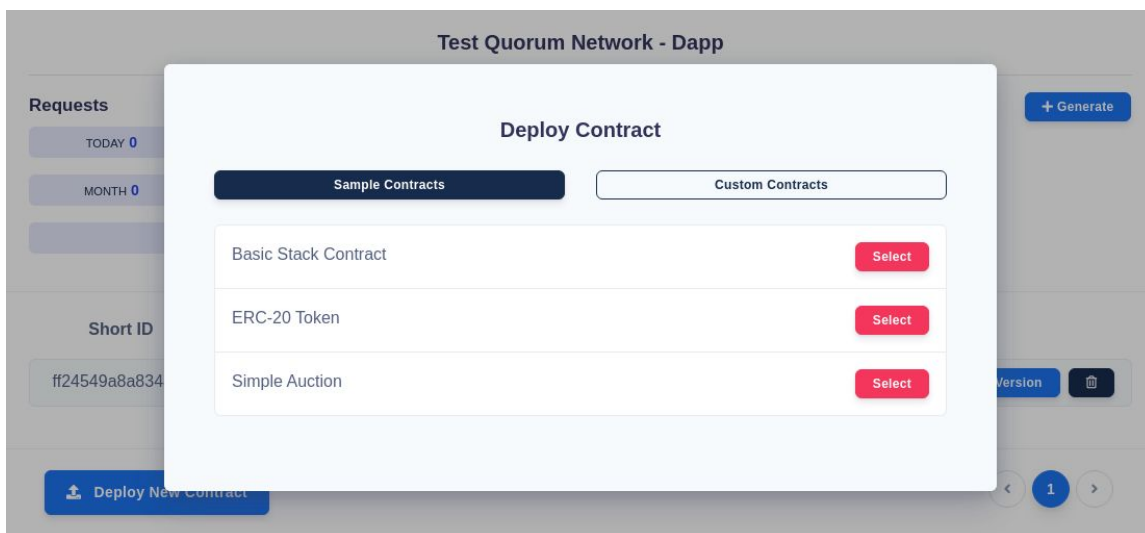


## Deploy Contract

When it comes to deploying contracts, there are two options. Deploying a contract from the “sample contracts” tab, or deploying a custom new contract from the “custom contracts” tab.

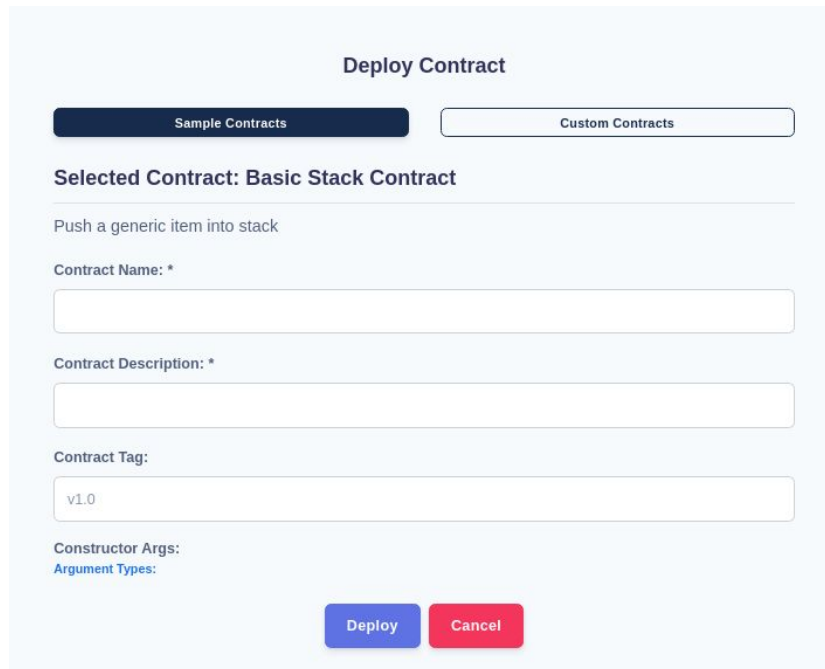


Here you can see the sample contracts provided for you to deploy.





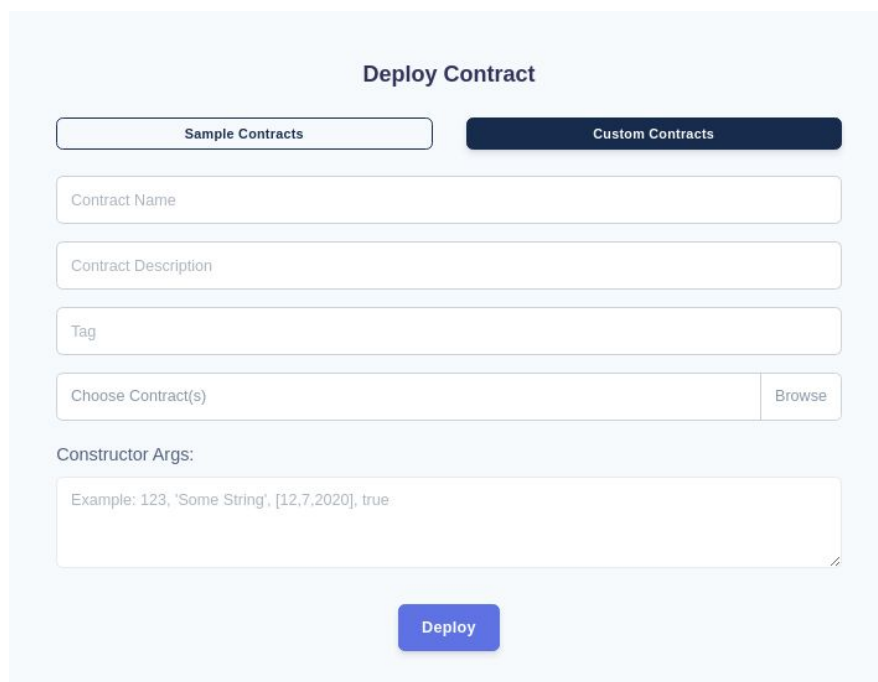
You can select the contract you desire to continue deploying the process by clicking on the “select” button. After you select the contract, a modal will pop-up for you to provide the details to the contract. Regarding details are the name, description, tag (optional) and the constructor parameters (if they exist in the contract) of the contract.



The screenshot shows a modal titled "Deploy Contract". At the top, there are two tabs: "Sample Contracts" (which is active and highlighted in dark blue) and "Custom Contracts". Below the tabs, the text "Selected Contract: Basic Stack Contract" is displayed. Underneath, there is a sub-header "Push a generic item into stack". The form contains several fields: "Contract Name: \*" with an empty text input; "Contract Description: \*" with an empty text input; "Contract Tag:" with a text input containing "v1.0"; and "Constructor Args:" with a sub-label "Argument Types:". At the bottom of the modal, there are two buttons: "Deploy" (blue) and "Cancel" (red).

Clicking on the “deploy” button will finish the deployment process. Then you can see your freshly deployed contract in the application details screen.

If you want to deploy your own contract, the “custom contracts” tab can be used.



The screenshot shows the same "Deploy Contract" modal, but with the "Custom Contracts" tab selected and highlighted in dark blue. The "Sample Contracts" tab is now inactive. The form fields are: "Contract Name" (empty), "Contract Description" (empty), "Tag" (empty), and "Choose Contract(s)" (empty) with a "Browse" button to its right. Below these is the "Constructor Args:" section with a sub-label "Argument Types:" and a text area containing the example: "Example: 123, 'Some String', [12,7,2020], true". At the bottom, there is a single "Deploy" button (blue).

After you fill the name, description, tag of the contract, you are expected to upload the contract (.sol file). Afterwards, you have to provide the constructor parameters of your contract (if they exist in the contract).

Clicking on the “deploy” button will finish the deployment process. Then you can see your freshly deployed contract in the application details screen.

## 4. Interact with Smart Contract

After you deploy your contract, you will want to interact with your contract such as “Invoke” or “Call” the methods in it. Application details page provides every necessary information to interact with your contract.

Test Quorum Network - Dapp

Requests

TODAY 0 WEEK 0

MONTH 0 QUARTER 0

YEAR 0

Shared With info@tubu.io

Share

Api Key There are no api keys!

+ Generate

Short ID	Name	Description	
ff24549a8a834ae1	ERC20	Tubu	View + Version

Deploy New Contract

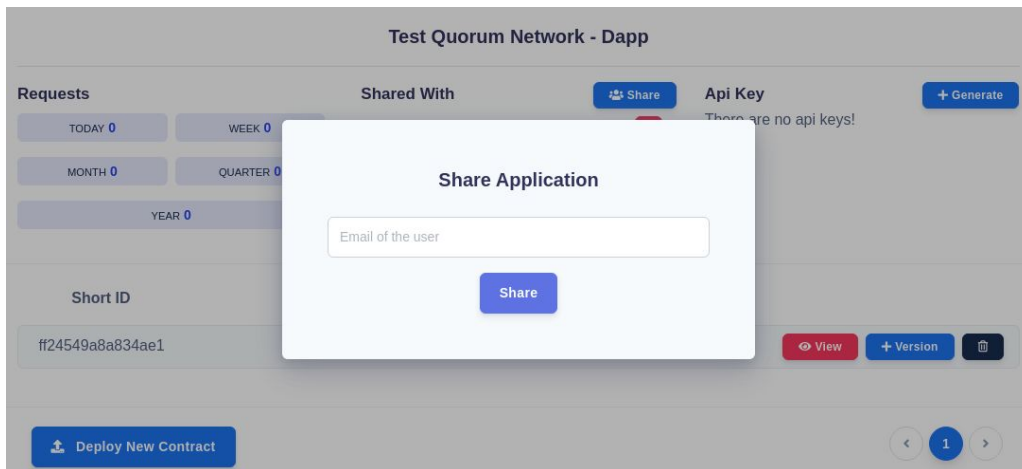
< 1 >

The following processes will be given in the subsections:

- Share Application
- Create Api Key
- Deploying New Version
- Accessing Contract Details
- Using SDKs

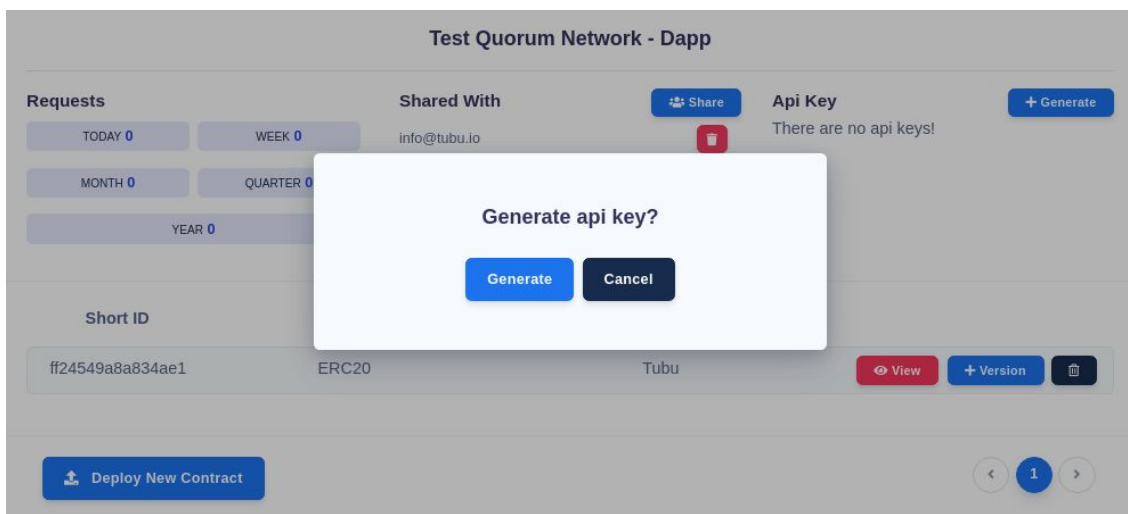
## Share Application

If you want your contracts to be managed by others, you can share them. If you click on the “share” button, a screen will appear for you to provide the email address of the person you want to cooperate with on your application.



## Create Api Key

You will have to have an Api Key for the application in order to interact with the contracts in it from the SDK side. To create an Api Key, click on the “generate” button. Afterwards, click “Generate” to confirm, “Cancel” to revert.



## Deploying New Version

You can deploy a new version to a pre-deployed contract. After deploying, you will access the latest deployed contract by default. If you want to access a specific version from the SDK, you will have to provide the tag you entered while deploying the version. If you haven't provided a tag in the first deployment of the contract, the default tag is 'v1.0'. You can see your different versions in the Contract Details page.

In order to deploy a new version, you have to click on the “version” button. That will open up a modal just like the one back in the deploying contract process. You have to provide the necessary information (this time the tag is obligatory) in the model, select your contract and deploy it.

### Deploy New Version

  
  
  
   
**Constructor Args:**

# Accessing Contract Details

In order to see the methods, versions, accounts of the contract, you can click on the “view” button. Clicking on that button will direct you to the contract details page.

**Test Quorum Network - Dapp - ff24549a8a834ae1 - v1.0**

---

**Requests**

TODAY 0   WEEK 0   MONTH 0   QUARTER 0   YEAR 0

---

**Versions**

Description	Tag	
Tubu	v1.0	<a href="#">Go To Version</a>

---

**Methods**

- [Approval\(\)](#)
- [Paused\(\)](#)
- [RoleAdminChanged\(\)](#)
- [RoleGranted\(\)](#)
- [RoleRevoked\(\)](#)

**Inputs**

Name: owner	Type: address
Name: spender	Type: address
Name: value	Type: uint256

---

**Accounts** [+ Add Account](#)

Name	Address	Status	
Tubu	0x9a9677B1d4863e6b30B887212F023022b5Ce7BBF	✓	<a href="#">Details</a>

To learn how to call smart contract functions please visit the [docs](#).

Here you can see the total request number, the versions, the methods, method inputs and method outputs and the accounts of the contract. You can click on the method name to see the inputs and outputs of the method. If you want to add an account to the contract, just click on the “add account” button and type the name of the account to the field in the selected model.

The screenshot shows the same contract details page as above, but with a modal dialog box open in the center. The modal is titled "Add Account" and contains a text input field labeled "Account Name" and a blue "Add Account" button. The background content is dimmed, showing the "Methods" section with "Approval()" selected, the "Inputs" table, and the "Accounts" table with one entry for "Tubu".