

CHAPTER 4

Wallets and Gateways

Users have the ability to manage their accounts on blockchain networks like Bitcoin or Ethereum through the usage of wallets. With an account on Ethereum, one can participate on the Ethereum blockchain network, such as by performing transactions on it.

An address on the Ethereum blockchain is a public string of letters and integers that begins with 0x. Because an Ethereum address is represented by a string of numbers and letters, it is possible to check the balance of any Ethereum address on the blockchain. However, it is not possible to determine who controls any given address. Users are able to exert control over an unlimited number of addresses via wallets, which can be either software or hardware.

Users of Ethereum wallets can move their cash around within the wallet by using a private key. This key serves as the control mechanism for Ethereum wallets. Because of this, these private keys are supposed to be known only by the person who created the wallet, as anyone who knows them can access the funds in the wallet.

There is a wide variety of Ethereum wallets available, some of which may be stored on your computer or mobile device, and others of which can be stored offline by means of a piece of paper, titanium, or other hardware. You can choose the sort of Ethereum wallet that best suits your needs.

One needs to use a secured key (known as the private key) to create a wallet. A wallet can be thought of as a store that holds the private and public keys for a user. In cryptography, a private key can be used to sign a transaction and a public key can be used to verify the signer. The idea is that the user keeps the private key secured with herself and shares the public key on the network. This allows the user to sign the transactions she wants to execute, and the verifiers using her public key can verify if she is the signer of those transactions. Wallets simplify this process of key management by providing a single place and an abstraction for clients to interact with the chain. People who want to avoid managing wallets can make use of third-party exchanges like binance that do the wallet management on the user's behalf. Since these exchanges are mostly centralized, any compromise of their servers can lead to compromise of user wallets. So, if your keys are on the exchange via an exchange-managed wallet, it is at your own risk. As the common saying goes, "Your keys, your money." From a security standpoint, it's always a better choice to have the keys off the exchange.

4.1 Types of Wallets

Some people store their Ethereum holdings in wallets designed for users, while others use cryptocurrency exchanges or other services, such as online marketplaces or loan services, offered by wallets designed for users. Wallets like these are referred to as custodial wallets, and they are distinguished from other wallets by the fact that they store users' private keys on their behalf. The user does not have direct control over the funds stored in the wallet; rather, the service manages the wallet on behalf of the user. There is a cost associated with this arrangement.

The danger that another party will not fulfill their obligations is known as counterparty risk. This risk is increased when funds are stored with a third party through the use of custodial wallets. The service that stores the private keys runs the risk of being hacked or acting maliciously, for example.

It's possible that various users would benefit from using one particular wallet over another. There are a plethora of wallets available now for the user to choose from. Most of them use Ethereum or ERC 20 standard-based tokens to access the applications on the blockchain. This access is realized by executing code in the form of a smart contract on the blockchain network. ERC 20 is a set of standards defined for creating tokens on the chain. This allows people to create their own tokens, which then can be used for their own applications if so desired.

There are different kinds of wallets available, as follows:

1. Mobile Wallet – These are mobile applications like the bitcoin.com app, which provides an interface for using the keys for transactions on the blockchain.
2. Desktop Wallet – This is installed as a desktop application and can be used on a laptop or home PC. An example of such a wallet is electrum.
3. Web Extensions – These are browser-based extensions that allow the wallet to be accessed from within the browser. MetaMask is an example of such wallets.
4. Hardware Wallet – These are physical hardware devices that can be used to securely manage your wallet. This is the most secure means of keeping keys secure.

In this chapter, we will discuss the MetaMask wallet, which comes as an extension to browsers like Chrome and Brave.

Once we choose a wallet, we need funds to interact with the Ethereum network. These funds are in the form of ether, which is the cryptocurrency used on the Ethereum blockchain. Ether can be purchased via exchanges like Binance or Coinbase.

All transactions on the Ethereum blockchain are validated by nodes, which are called validator nodes. They charge a fee for validating the transactions, and this fee (called gas) has to be paid by the user initiating the specific transaction. There are means via which a user can check the estimated gas fee based on estimated resources needed to execute the specific smart contract.

Since the production Ethereum network (also known as mainnet) will charge the gas fee for every transaction, it's not feasible to do development on the mainnet. To facilitate the development of decentralized applications on the Ethereum blockchain, there are many development networks available, known as testnets, that allow one to develop applications using exactly the same interfaces as the Ethereum blockchain, but without incurring real gas costs.

4.2 So, What Is a Testnet ?

When someone starts to develop a decentralized application (Dapp), they can deploy it to a test network, which from an interface perspective is the same as the main network. This test network is known as a testnet. This provides an opportunity for the developers, the community, and you to test it out before real assets are involved. Ether and tokens on a testnet are simple to acquire and have no value in the real world.

There are currently four major testnets in operation, and each functions in a manner that is analogous to the production blockchain (where your real ether and tokens reside). In most cases, projects will only be developed on a single testnet, despite the fact that individual developers may have a preference or favorite among them.

1. Ropsten – A testnet blockchain that is based on proof of work and most closely resembles Ethereum
2. Rinkeby – A blockchain based on proof of authority that was initiated by the Geth team
3. Kovan – Again, a blockchain based on proof of authority
4. Goerli – A proof of authority-based testnet

To connect our wallet to any of these testnets, we need to have these two things:

1. A wallet installed on the local machine. We will use MetaMask as the wallet here.
2. A gateway like Infura to enable our wallet connectivity to these testnets.

4.3 MetaMask

MetaMask belongs to a family of what we call HD (hierarchical deterministic) wallets. See <https://coinsutra.com/hd-wallets-deterministic-wallet/>.

4.3.1 Installation

In the next two steps, installation and configuration instructions for MetaMask are detailed for your convenience. After that, we will go through a few different configurations that you ought to become familiar with.

We can install MetaMask as an extension to Chrome/Brave. Figure 4-1 shows how the extension looks on Brave.

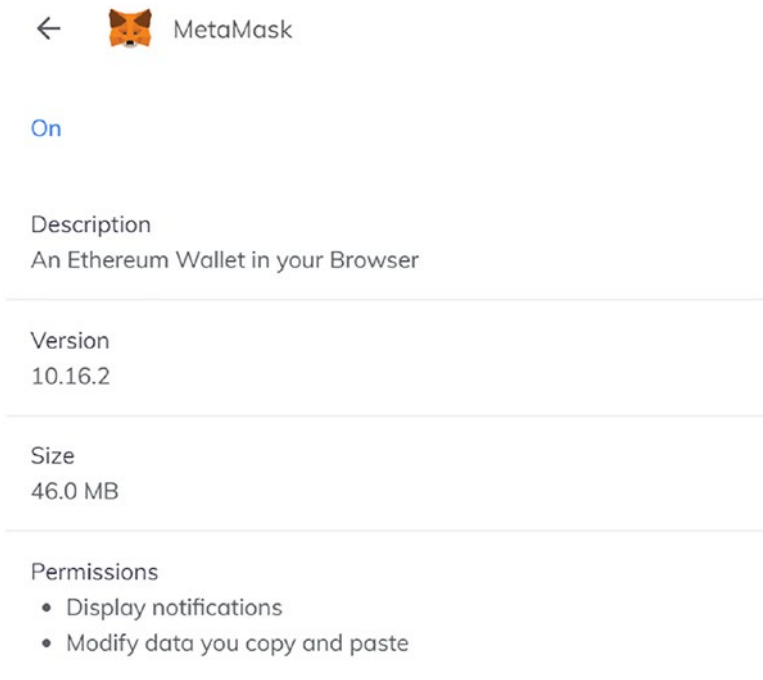


Figure 4-1. *MetaMask extension on Brave browser*

Make sure that only you can access your MetaMask account by coming up with a password and keeping it a secret from anyone else who uses the computer you share.

Immediately following the submission of your password, you will be presented with your 12-word seed phrase.

Even if they do not have the password that you chose for your account in the step before this one, anyone who knows these 12 words can log in to your account. You should never share your seed words with somebody in whom you do not have complete faith. In the event that you forget your password or something happens to your computer, you will need to re-enter these 12 words to regain access to your wallet.

Launching the MetaMask wallet extension, we see three stacked dots on the right-hand side, as shown in Figure 4-2.

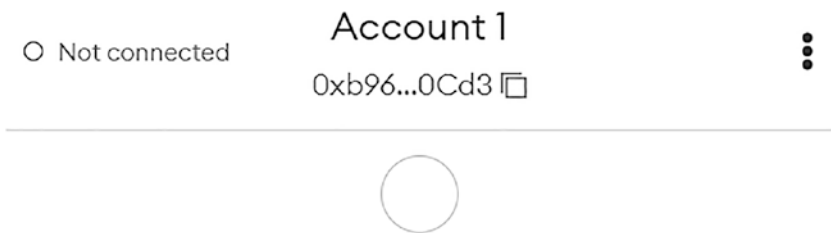


Figure 4-2. *MetaMask wallet extension*

By clicking on these three dots, we can get account details, as shown in Figure 4-3.

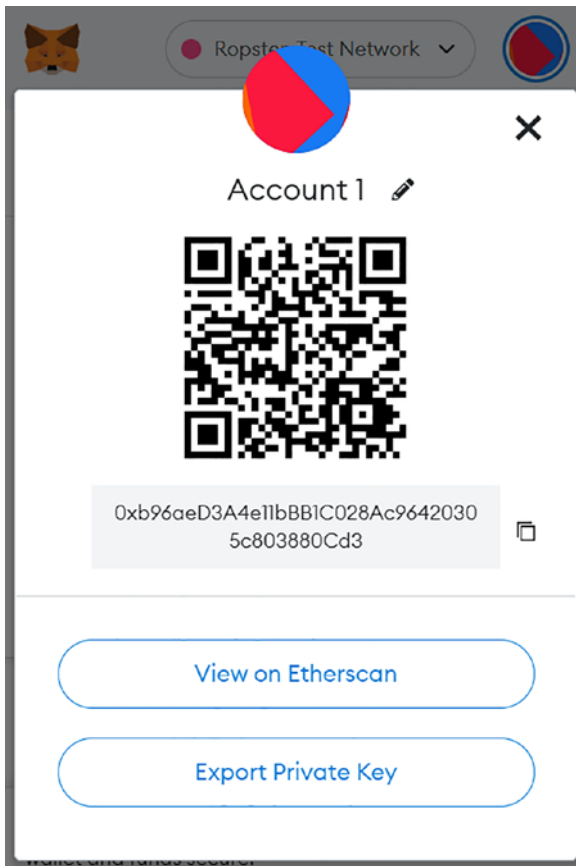


Figure 4-3. Account details of the MetaMask wallet

We can view the account on Etherscan by clicking on the button View on Etherscan, as shown in Figure 4-4.



Figure 4-4. *Etherscan view of the account*

Click on the dropdown menu to show the networks available, as shown in Figure 4-5.

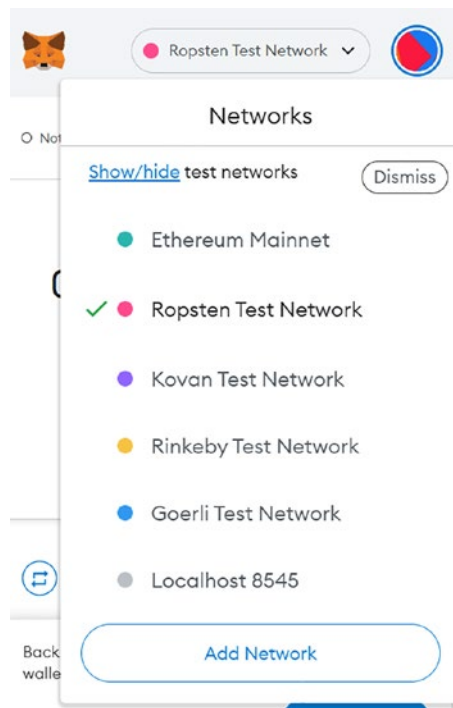


Figure 4-5. *The networks available to MetaMask wallet*

Before we add funds to the wallet from a testnet, we need to understand how the connectivity to these testnets works from MetaMask. One option is to run a testnet node ourselves and then connect our wallet to it. Another option is to go via a hosted testnet.

There are a few hosting providers for testnets. Here, we discuss one of them known as Infura.

Infura is an infrastructure-as-a-service (IaaS) and Web3 backend provider that offers a variety of services and tools to blockchain developers. This includes the application programming interface (API) suite for the Infura platform. The Infura Web3 service revolves around the flagship Infura Ethereum API as its central component. However, communication with both the InterPlanetary File System (IPFS) and Filecoin is currently

being worked on. Having said that, certain alternatives to Infura currently offer wider cross-chain connectivity than Infura itself does. Many blockchain developers are currently seeking Infura alternatives, despite the fact that Ethereum is currently the most popular programmable blockchain for the launch of decentralized applications (DApps). This occurs as Binance Smart Chain (BSC) and Polygon Network are becoming increasingly well known (previously Matic Network).

A high-level architecture of the Infura gateway is shown in Figure 4-6.

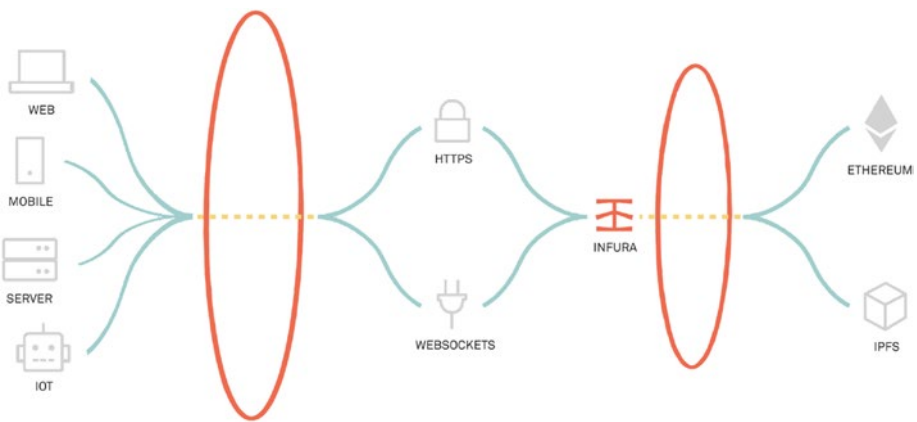


Figure 4-6. *Infura architecture*

On the left side of Figure 4-6, we see the wallet that connects to the Infura infrastructure via either https or websockets. Infura in turn provides connectivity to the blockchain networks, like Ethereum mainnet or testnets. Infura acts as a gateway for the wallets to the blockchain world.

Users who make use of the Infura Ethereum API are able to devote more of their time and resources to activities such as doing market research and product development. In addition, users are provided with a straightforward and user-friendly dashboard that allows them to obtain a greater understanding of how apps are performing. Utilizing the dashboard makes it simple to conduct application analysis and configuration. In addition, developers are able to track usage times,

the effectiveness of various request types, and a great deal more. These insights allow developers to improve their programs by gaining a deeper understanding of the people who use those applications. In addition, the Infura Ethereum API is interoperable with both testnets and mainnets, and it uses client-compatible JSON-RPC that is transferred through HTTPS and WSS. Users are also given the opportunity to obtain access to the Ethereum Archive node data that is made accessible as an add-on.

Infura is the default node provider that MetaMask utilizes, although users have the opportunity to switch to another node provider or even host their own node.

Let us add some funds to our MetaMask wallet.

We will experiment a bit with the Ropsten testnet for this example.

Navigate to <https://faucet.egorfine.com/>.

We see a screen similar to the one shown in Figure 4-7.

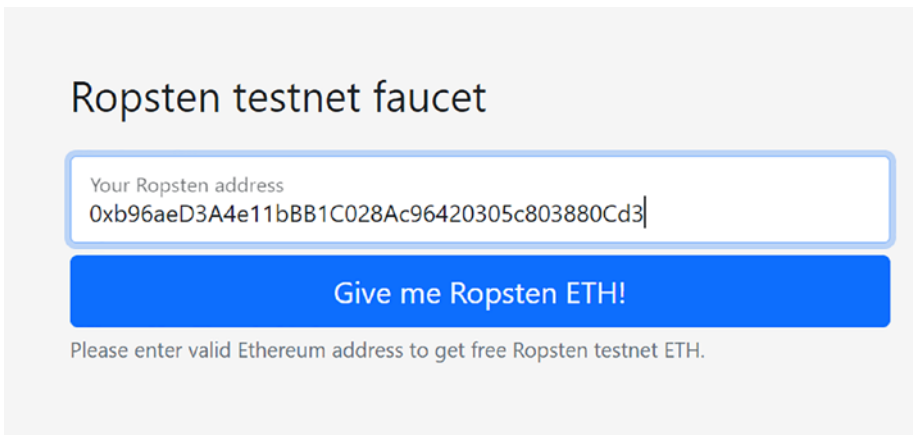


Figure 4-7. *Ropsten testnet Faucet*

In the address field, we need to put the public key that we get from the MetaMask wallet. Upon clicking the Give me Ropsten ETH! Button, we see the result shown in Figure 4-8.

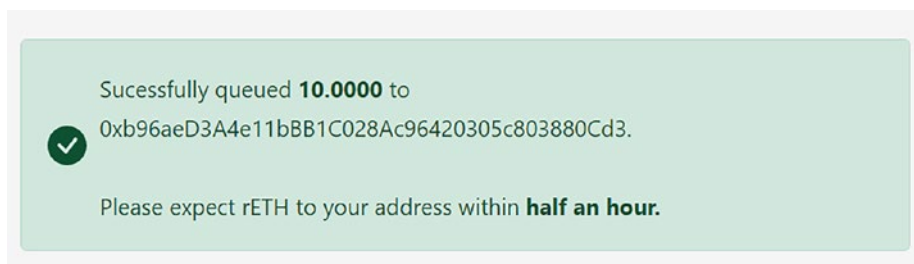


Figure 4-8. Confirmation of ETH added to the wallet

We can check our MetaMask wallet to see if funds got added, as shown in Figure 4-9.

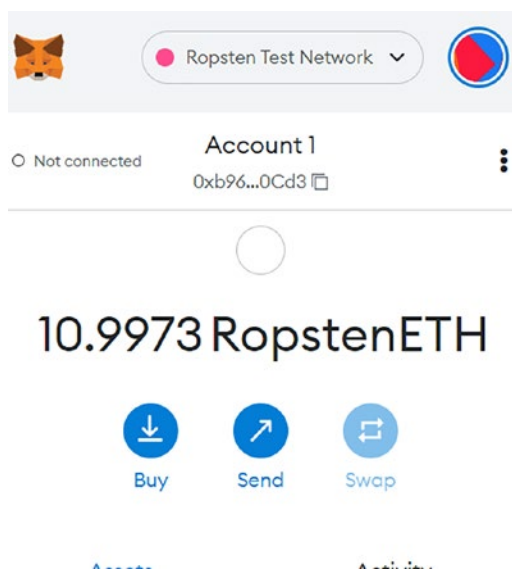


Figure 4-9. The MetaMask wallet view. It shows the Ropsten ETH credited to the wallet

We can see 10.9973 ETH in my wallet, out of which 0.9973 were already there from my previous addition.

We will look into Etherscan once to check the transaction. We see a screen like that shown in Figure 4-10.

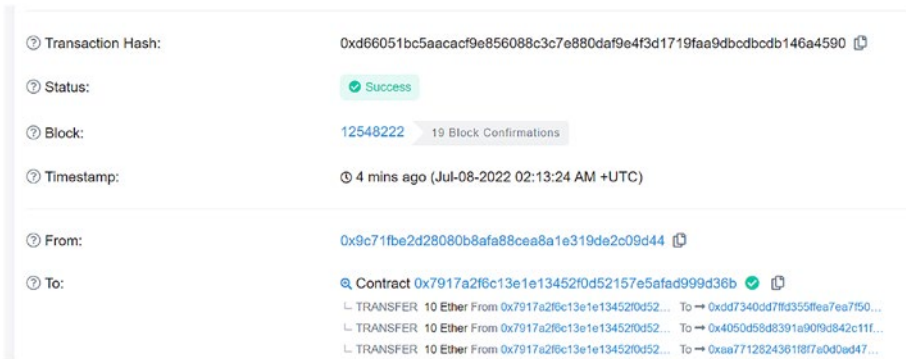


Figure 4-10. Etherscan transaction view

We can see that 10 ether got added to my account on the Ropsten network.

We will need these funds when we create a smart contract and invoke functions on it. We already talked about gas fees in the previous chapter.

First, we will create a simple web page that uses the web3.js library to connect to the testnet via Infura. Let's now get a small introduction to web3.js.

4.4 Web3.js

The Ethereum APIs can be consumed over an HTTP-based protocol. So, one can create HTTP clients, which allows us to interface with the Ethereum networks (either mainnet or testnet). To ease the process of development, the Ethereum Foundation created a library in JavaScript that allows us to access the Ethereum blockchain programmatically. They called this library web3.js, and, as the name suggests, it's for web3-based applications, and the library itself is written in the JavaScript programming language.

This library comprises different modules, which are described in the next subsections.

4.4.1 web3-eth

A user of `web3.js` is able to connect with the Ethereum blockchain thanks to the `web3-eth` module, which offers functions that make this possible. To be more specific, these functions are able to communicate with smart contracts, accounts that are controlled by other parties, nodes, blocks that have been mined, and transactions.

Using the `web3-eth` library functions, one can sign the transactions, can check balances in your Ethereum wallet, as well as send the signed transaction over the internet to the Ethereum blockchain.

4.4.2 web3-shh

You will be able to interact with the Whisper protocol if you make use of the `web3-shh` module. Whisper is a messaging protocol that was developed to facilitate the easy broadcasting of messages and the low-level, asynchronous transmission of data. The following are two instances that illustrate the point:

1. The network receives a Whisper message when `web3.shh.post` is called.
2. Apart from just sending messages, one can subscribe to messages using the `web3.shh.subscribe` method. This allows the user to receive messages from the network.

4.4.3 web3-bzz

It's in the user's interest to have clarity as to what kind of data is stored on the blockchain. Generally, we store only transactional data on the chain. Other data, like documents, images, videos, and so on, are not stored on the blockchain. We can use decentralized storage services like Swarm, ipfs,

and so forth for those needs. We can store references to such content on the blockchain though. And this is where the module `web3-bzz` helps. It provides us a library-based abstraction to communicate with Swarm.

We can upload and download images, documents, videos, and audio clips to the Swarm network using the `web3.bzz.upload` and `web3.bzz.download` methods.

4.4.4 web3-net

You will be able to interact with the network attributes of an Ethereum node if you make use of the `web3-net` module. You will be able to get information about the node by using the `web3-net` module. This module allows us to extract metadata about the Ethereum node itself. As an example, the network ID can be obtained by calling `web3.net.getID`, and using `web3.net.peerCount` will return the number of peer nodes connected to a specific node.

4.4.5 web3-utils

The `web3-utils` module allows us to make use of some of the utility functions defined inside this library module. Included in `web3-utils` is a collection of utility functions that can search databases, convert numbers, and check to see whether a value satisfies a given criterion. The following are three instances that illustrate the point:

1. `web3.utils.toWei` is a converter that goes from wei to ether.
2. `web3.utils.hex` converts a hexadecimal value to a string with the `ToNumberString` function.

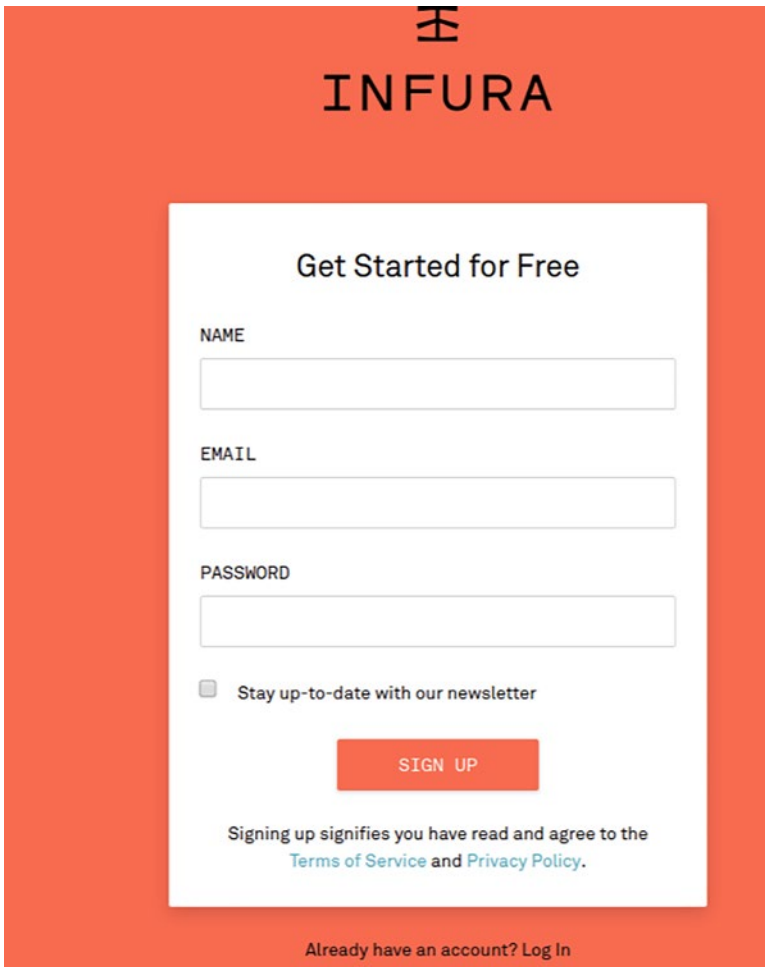
3. The `web3.utils.isAddress` function determines whether or not the given string represents a valid Ethereum address.

4.5 Infura Setup

Since we use Infura as the gateway, we need to configure the Infura endpoint, which can then be used from our applications.

In order for us to access the Infura network, the first thing that has to be done is to sign up for an Infura account and obtain an API key. We can go to <https://infura.io> to access Infura.

Please visit the Infura website to create a new account for yourself. When you open the account creation page, you will see the screen shown in Figure 4-11.



The image shows a registration form for Infura. At the top, there is a logo consisting of a stylized 'I' and 'F' symbol above the word 'INFURA'. Below this, the heading 'Get Started for Free' is centered. The form contains three input fields: 'NAME', 'EMAIL', and 'PASSWORD'. Below the password field is a checkbox labeled 'Stay up-to-date with our newsletter'. A red 'SIGN UP' button is positioned below the checkbox. At the bottom of the form, there is a line of text: 'Signing up signifies you have read and agree to the [Terms of Service](#) and [Privacy Policy](#).' Below the form, there is a link: 'Already have an account? Log In'.

Figure 4-11. *Infura account creation page*

The next step is to go to the dashboard page and click the Create New Project button. Give your project a name and click the Create button, as shown in Figure 4-12.

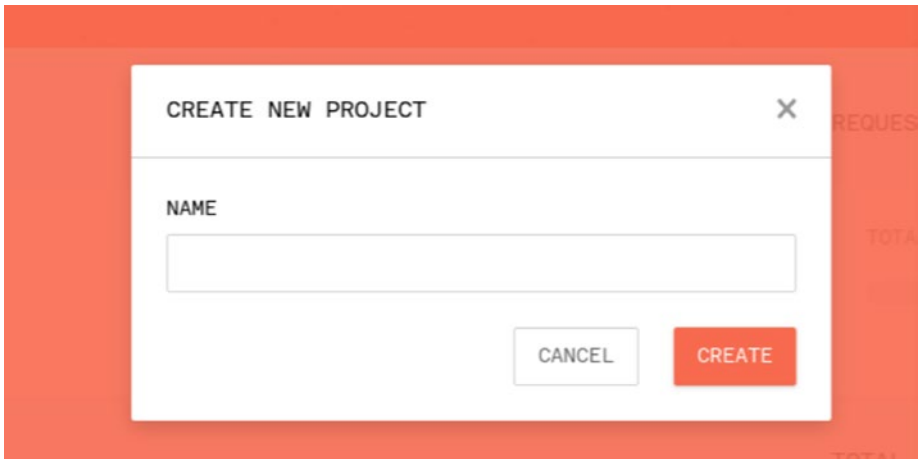


Figure 4-12. *Create New Project screen*

I created a project named `trial`, as shown in Figure 4-13.



Figure 4-13. *Creating a project named trial*

We next get the project ID and project secret, as shown in Figure 4-14.

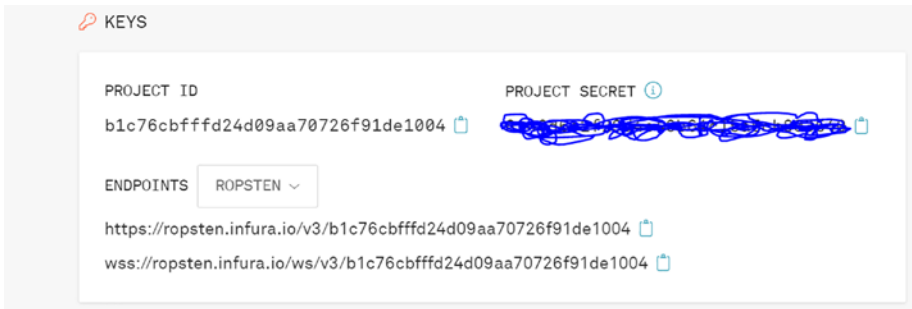


Figure 4-14. Keys for the Infura project we created

Since I am using the Ropsten network, I choose the endpoint as Ropsten.

We get two endpoints:

1. HTTPS based
2. Secure web socket

4.5.1 Interfacing with Ropsten Network via Infura Gateway

Please copy the HTTPS endpoint URL. Remember this URL constitutes your project ID. We will use this URL in our HTML page, which we will create next.

Create a directory `web3`

Cd to `web3` directory

Inside the `web3` directory, create an HTML page. Copy the following content to this HTML page:

```
<html>
```

```
  <header>
```

```

<title>Sample Infura connectivity check</title>

<script
//importing the web3.js library
  src="https://cdn.jsdelivr.net/gh/ethereum/web3.js@1.0.0-
beta.36/dist/web3.min.js" integrity="sha256-nWBTbvvhJgjslRyuAK
JHK+XcZPlCnmIAAMixz6EefVk=" crossorigin="anonymous"></script>

  <script src="https://code.jquery.com/jquery-3.4.1.min.js"
integrity="sha256-CSXorXvZcTkaiX6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
crossorigin="anonymous"></script>

  <script>
    if (typeof web3 !== 'undefined') {
      web3 = new Web3(web3.currentProvider);
    } else {
      // Set the provider you want from Web3.providers
      //see the ropsten url is the one we copied from infura
      web3 = new Web3(new Web3.providers.HttpProvider("https://
ropsten.infura.io/v3/b1c76cbfffd24d09aa70726f91de1004"));
    }
  </script>
</header>
<body>

  <div>
//get the last block on the ropstentestnet
  <h2>Latest Block</h2><span id="lastblock"></span>

  </div>

```

```

<script>
  web3.eth.getBlockNumber(function (err, res) { if (err)
console.log(err)
  $( "#lastblock" ).text(res)
  })
</script>
</body>
</html>

```

We get the following output on loading this page in the browser; we see the response as shown in Figure 4-15.

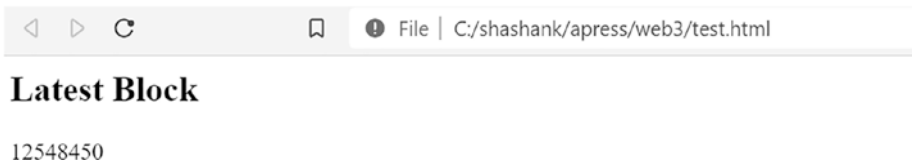


Figure 4-15. Getting the latest block via web3 library

This means that our test succeeded in connecting to the Ropsten testnet via the Infura gateway.

We modify the following HTML file to extract more information, like the ETH balance as well as the node information:

```

<html>
  <header>
    <title>Sample Infura connectivity check</title>
  <script

```

```

src="https://cdn.jsdelivr.net/gh/ethereum/web3.js@1.0.0-
beta.36/dist/web3.min.js" integrity="sha256-nWBTbvvhJgjslRyuAK
JHK+XcZPlCnmIAAMixz6EefVk=" crossorigin="anonymous"></script>

<script src="https://code.jquery.com/jquery-3.4.1.min.js"
integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
crossorigin="anonymous"></script>

<script>
if (typeof web3 !== 'undefined') {
web3 = new Web3(web3.currentProvider);
} else {
// Set the provider you want from Web3.providers
web3 = new Web3(new Web3.providers.HttpProvider("https://
ropsten.infura.io/v3/b1c76cbfffd24d09aa70726f91de1004"));
}
</script>
</header>
<body>
<div>
<h2>Latest Block</h2><span id="lastblock"></span>
<h2>My balance</h2><span id="balance"></span>
<h2>node information</h2><span id="nodeInfo"></span>
</div>
<script>

```

CHAPTER 4 WALLETS AND GATEWAYS

```
web3.eth.getBlockNumber(function (err, res) { if (err)
console.log(err)
$( "#lastblock" ).text(res)
})
web3.eth.getBalance("0xb96aeD3A4e11bBB1C028Ac96420305c803880
Cd3", function (err, res) { if (err) console.log(err)
$( "#balance" ).text(res)
})

web3.eth.getNodeInfo(function (err, res) { if (err) console.
log(err)

$( "#nodeInfo" ).text(res)
})
</script>
</body>
</html>
```

Loading this in a browser gives the output shown in [Figure 4-16](#).

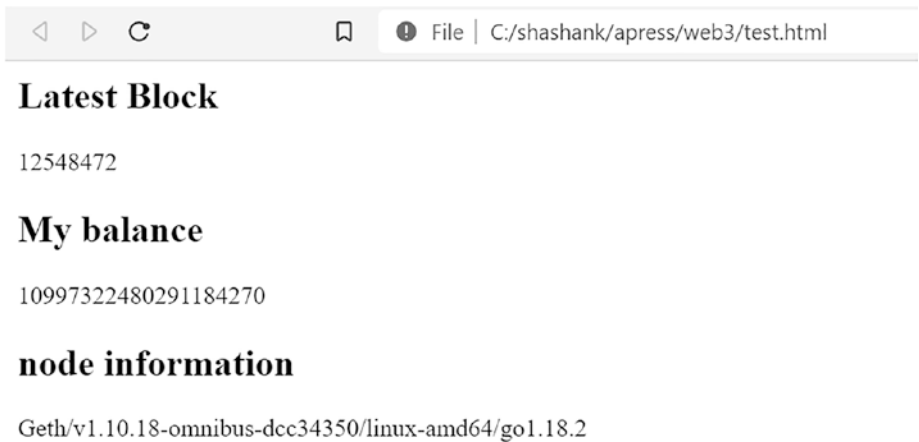


Figure 4-16. *Showing block information and balance information*

The same functionality we showcased via the browser can be achieved using nodejs as the JavaScript-based application server.

4.6 Summary

In this chapter, we looked at different kinds of wallets and how they work. We detailed how MetaMask wallet works. We also looked at how gateways work and their purpose in the web3 world.

In the next chapter, we will look into how we can use the Remix IDE (a browser-based environment) to compile and deploy smart contracts.