# **Fieres Blockchain**

Fieres is an open-source platform for launching decentralised applications and enterprise blockchain deployments in one interoperable, highly scalable ecosystem. Fieres Blockchain is the first decentralised smart contracts platform built for the scale of global finance, with near-instant transaction finality. Ethereum developers can quickly build on the Fieres blockchain as Solidity works out of the box.

Fieres is a fast, scalable, and secure layer-1 platform built on the POA consensus protocol.

# **How can I add Fieres to Metamask?**

To connect with Metamask to Fieres network, you should go to Metamask network settings and paste following parameters.

**Testnet Configuration Parameters:**

**Network Name:** Fieres Blockchain - Testnet

**New RPC URL:** http://ec2-23-23-118-55.compute-1.amazonaws.com:8545

**ChainID:** 1337

**Symbol:** FieresE

**Block Explorer URL:** http://ec2-23-23-118-55.compute-1.amazonaws.com:4000/

# **What is Blockchain?**

A blockchain is a shared, publicly accessible database that is shared among numerous computers in a network.

Data and information are kept in units called "blocks," which are grouped together in sequential order. The transaction data must be included to a block in order for FieresE to be sent to a recipient.

The term "chain" describes how each block contains a cryptographic reference to its parent. Blocks are therefore connected together. The data in a block cannot be modified without also changing all blocks that come after it, which would necessitate network-wide consensus.

Every computer in the network must agree to accept both the chain as a whole and each new block. "Nodes" are the titles given to these computers. Nodes ensure that the same data is accessible to everyone using the blockchain. A consensus mechanism is required for blockchains in order to implement this distributed agreement.

In blockchain platforms, consensus mechanisms can be divided into *permissionless* (Bitcoin, Etherium) and *permissioned* (Apla, Etherium Private).

In a permissioned blockchain, all nodes are pre-authenticated. This advantage allows to use consensus types that provide high transaction rate in addition to other benefits. One of these consensus types is *Proof-of-Authority* (PoA) consensus.

*Proof-of-Authority* (PoA) is a new consensus algorithms family that provides high performance and fault tolerance. In PoA, rights to generate new blocks are awarded to nodes that have proven their authority to do so. To gain this authority and a right to generate new blocks, a node must pass a preliminary authentication.

The network's nodes check and verify new blocks once they are broadcast to them, updating everyone's knowledge of the blockchain's current state.

# **What is Fieres?**

The native cryptocurrency of the Fieres Blockchain is called **Fiero**. In order to provide a market for computation, **Fiero** was created. Such a market gives participants a financial incentive to validate and execute transaction requests and gives the network computational resources.

Any participant that broadcasts a transaction request is also required to provide a bounty to the network in the form of some **Fiero**. Whoever completes the process of verifying the transaction, carrying it out, committing it to the blockchain, and broadcasting it to the network will ultimately receive this payout.

# **What are Smart Contracts?**

In reality, participants don't create fresh code each time they need to ask the EVM to perform a computation. Instead, programmers for applications upload programmes (reusable bits of code) into the EVM state, and users submit requests to run these bits of code with various parameters. The programmes that are uploaded to and run by the network are referred to as smart contracts.

At its most basic level, a smart contract can be compared to a type of vending machine: a script that, when invoked with specific parameters, does certain actions or computations if specific criteria are met. For instance, if the caller transfers **Fiero** to a particular receiver, a straightforward vendor smart contract might generate and assign ownership of a digital asset.

With the blockchain serving as its data layer, any developer may build a smart contract and publish it to the network for a fee. The smart contract can then be called by any user, again in exchange for a charge given to the network to run its code.

Developers can therefore create and implement any levels of complexity in user-facing apps and services, such as marketplaces, financial instruments, games, etc., using smart contracts.

# **What are Dapps?**

A frontend user interface and a smart contract are both components of a decentralised application (dapp), which is a programme created on a decentralised network. Your dapp can even incorporate a smart contract that someone else has created because smart contracts on the Fieres Blockchain are transparent and accessible, much like open APIs.

PREREQUISITES

Learn about the fundamentals of blockchain technology and the decentralised nature of the Fieres Blockchain network before diving into the world of dapps.

DEFINITION OF A DAPP

A dApp is a decentralized application that runs on a distributed ledger, such as a blockchain. Unlike traditional apps, which rely on a central server, dApps are powered by a peer-to-peer network. This decentralized structure offers several advantages, including improved security and transparency.

* **Decentralised** - dApps run on Fieres Blockchain, an open, public, decentralised platform that is not controlled by any one entity or group.
* **Deterministic** - dApps serve the same purpose no matter what environment they are executed in.
* **Turing complete** - dApps can carry out any activity if they have the necessary resources.
* **Isolated -** dApps are run in a virtual environment called the Ethereum Virtual Machine so that if a smart contract has a problem, it won't interfere with the blockchain network's regular operation.

In order to introduce dapps, smart contracts—the backend of a dapp, for lack of a better term—must be introduced. Visit our section on smart contracts for a thorough description.

A smart contract is a piece of code that operates exactly as intended and is stored on the Fieres Blockchain. Once implemented on the network, smart contracts cannot be changed. Dapps can be decentralised because the logic spelt forth in the contract, not a person or organisation, governs them. This also implies you need to draft your contracts very carefully and test them thoroughly.

BENEFITS OF DAPP DEVELOPMENT

**Zero downtime** – The network as a whole will always be able to serve users trying to engage with the smart contract once it has been placed on the blockchain. Therefore, malicious actors are unable to conduct denial-of-service attacks against individual dapps.

**Privacy** – A real-world identity is not required in order to use or deploy a dapp.

**Resistance to censorship** – Users cannot be prevented from submitting transactions, establishing dapps, or reading data from the blockchain by a single entity on the network.

**Complete data integrity** – Cryptographic primitives provide for the immutability and irrefutability of data recorded on the blockchain. Malicious actors are unable to alter transactions or other data that has already been made available to the public.

**Trustless computation/verifiable behaviour** – Without the need to rely on a centralised authority, smart contracts can be analysed and are guaranteed to execute in predictable ways. This is not the case with traditional models; for instance, while using online banking services, we must have faith that financial companies won't mishandle our personal information, falsify records, or experience a hack.

### **DRAWBACKS OF DAPP DEVELOPMENT**

**Maintenance** – dApps can be more difficult to maintain since it is more difficult to change the code and data that is published on the blockchain. Once a dapp (or the underlying data stored by a dapp) is launched, it might be challenging for developers to update it, even if defects or security threats are found in an older version.

**Performance overhead** – Scaling is quite challenging, and there is a significant performance overhead. Every node processes and saves every transaction in order to reach the degree of security, integrity, transparency, and dependability that Fieres Blockchain strives for. Additionally, the proof-of-work process takes time. Back of the envelope math estimates the overhead at approximately 1,000,000x that of the current standard compute.

**Network congestion** – The entire network experiences a lag when one dapp consumes excessive amounts of processing resources. If transactions are put in quicker than the network can currently handle (15–20 transactions per second), the number of unconfirmed transactions will quickly increase.

**User experience** – Because the typical end user might find it too challenging to set up the tool stack required to connect with the blockchain in a properly safe manner, it might be more difficult to design user-friendly experiences.

**Centralisation** – On top of the Fieres Blockchain's basic layer, user- and developer-friendly solutions may ultimately resemble centralised systems. Such services might, for instance, conduct critical business logic on a centralised server before writing to the blockchain, offer a frontend using a centralised server, or store keys or other sensitive information there. Many, if not all, of blockchain's advantages over the conventional approach, are eliminated by centralised control.

# **Consensus**

Consensus is the process of getting a bunch of computers to reach a consensus. By following a series of instructions known as a consensus protocol, computers can come to a consensus. POA, a trustworthy consensus process that is scalable, reliable, and decentralised, is used by Fieres. It offers a quick throughput and minimal **latency. It uses little energy and doesn't need specialised computer hardware. It functions effectively in difficult conditions and is resistant to "51% attacks."**

Consensus systems (networks of computers) can cooperate and maintain security thanks to consensus methods, often known as consensus protocols or consensus algorithms.

Consensus algorithms and protocols are frequently used interchangeably. Algorithms and protocols, however, differ from one another. A protocol is a set of guidelines outlined in a standard that controls how a system and all of its component elements work in unity. Algorithms are like detailed instructions on how to perform calculations or solve problems.

The establishment of consensus among database nodes, application servers, and other enterprise infrastructure has been accomplished for many years using these approaches. New consensus techniques have been developed in past years to enable crypto-economic systems, such Fieres Blockchain, to concur on the network's state.

# **Proof Of Authority**

# 

In blockchain platforms, consensus mechanisms can be divided into *permissionless* (Bitcoin, Etherium) and *permissioned* (Apla, Etherium Private).

In a permissioned blockchain, all nodes are pre-authenticated. This advantage allows to use consensus types that provide high transaction rate in addition to other benefits. One of these consensus types is *Proof-of-Authority* (PoA) consensus.

*Proof-of-Authority* (PoA) is a new consensus algorithms family that provides high performance and fault tolerance. In PoA, rights to generate new blocks are awarded to nodes that have proven their authority to do so. To gain this authority and a right to generate new blocks, a node must pass a preliminary authentication.

# **Run a Fieres Blockchain full node**

This section provides documents on how to build and maintain a Fieres blockchain full node and then validate the Fieres network using a Fieres node.

The quickest way to learn about Fieres is to run a node and interact with the network. In this tutorial, we will:

* Install and run an Fieres node
* Connect to Fieres

Requirements[​](https://docs.avax.network/nodes/build/run-avalanche-node-manually#requirements)

Computer Hardware and OS

Nodes can operate on common hardware because Fieres is such a lightweight protocol. Keep in mind that hardware requirements may alter as network demand grows.

* CPU: Equivalent of 6 AWS vCPU
* RAM: 8 GiB
* Storage: 250 GB

#### **Networking**[**​**](https://docs.avax.network/nodes/build/run-avalanche-node-manually#networking)

To run successfully, Fieres needs to accept connections from the Internet on the network port 1337 for test net. You must decide the networking environment your node will operate in before you start the installation.

**Running on a Cloud Provider**[**​**](https://docs.avax.network/nodes/build/run-avalanche-node-manually#running-on-a-cloud-provider)

Your node will have a static IP if it is operating on a cloud provider's computer instance. Find out what that static IP is, or if you haven't already, configure it.

**Running on a Home Connection**[**​**](https://docs.avax.network/nodes/build/run-avalanche-node-manually#running-on-a-home-connection)

You have a dynamic IP, which means that it will change from time to time if you are running a node on a computer connected to a residential internet connection. In order to access the computer, the node is installed on from the Internet, you must configure inbound port forwarding 1337 for the test net.

# **Starting a Full Node**

For running a full node of Fieres Blockchain, execute the following shell script with root privileges. It will install and run Besu Client on your linux instance.

Testnet Shell Script

#!/bin/sh

sudo ufw allow 30303

sudo apt-get update

sudo apt install default-jdk

java - version

wget https://hyperledger.jfrog.io/artifactory/besu-binaries/besu/22.7.4/besu-22.7.4.zip

sudo apt-get install unzip

unzip besu-22.7.4.zip

cd besu-22.7.4/bin

export PATH=$PWD:$PATH

cd

mkdir EVMProtocol

cd EVMProtocol

echo "Copy the testnet genesis file details on nano text editor and Press control+x and then press y and hit the enter to save genesis data"

nano genesis.json

echo "Run below command in $HOME/EVMProtocol directory"

besu --data-path=data --genesis-file=../genesis.json --bootnodes= enode://b0e1ea5ee7f81c941023f314510b58761fda1b213f7de081dcb049e9aa4f7846acbad9d41cdb09e1e99bf7cb160dfbc85edec9615ca3e1f5c0e4e884b290ccad@ec2-23-23-118-55.compute-1.amazonaws.com:30303 --p2p-port=30304 --rpc-http-enabled --rpc-http-api=ETH,NET,QBFT --host-allowlist="\*" --rpc-http-cors-origins="all" --rpc-http-port=8545 --rpc-http-host=0.0.0.0

By default RPC port is 8545. If the port is open, anyone can make RPC

If you want to customize node discover Besu official [docs](https://besu.hyperledger.org/en/stable/public-networks/reference/cli/options/).

# **Genesis File**

You need to utilise following genesis file for setup Fieres Blockchain Node.

Testnet Genesis file (.json format)

{

"config" : {

"chainId" : 1337,

"berlinBlock" : 0,

"qbft" : {

"blockperiodseconds" : 2,

"epochlength" : 30000,

"requesttimeoutseconds" : 4

}

},

"nonce" : "0x0",

"timestamp" : "0x58ee40ba",

"gasLimit" : "0x47b760",

"difficulty" : "0x1",

"mixHash" : "0x63746963616c2062797a616e74696e65206661756c7420746f6c6572616e6365",

"coinbase" : "0x0000000000000000000000000000000000000000",

"alloc" : {

"fe3b557e8fb62b89f4916b721be55ceb828dbd73" : {

"privateKey" : "8f2a55949038a9610f50fb23b5883af3b4ecb3c3bb792cbcefbd1542c692be63",

"comment" : "private key and this comment are ignored. In a real chain, the private key should NOT be stored",

"balance" : "0xad78ebc5ac6200000"

},

"627306090abaB3A6e1400e9345bC60c78a8BEf57" : {

"privateKey" : "c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3",

"comment" : "private key and this comment are ignored. In a real chain, the private key should NOT be stored",

"balance" : "90000000000000000000000"

},

"f17f52151EbEF6C7334FAD080c5704D77216b732" : {

"privateKey" : "ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f",

"comment" : "private key and this comment are ignored. In a real chain, the private key should NOT be stored",

"balance" : "90000000000000000000000"

}

},

"extraData" : "0xf88fa00000000000000000000000000000000000000000000000000000000000000000f8699488db7c97610fd56d2da71443bb3c881f807bdb8b9467d8c651089b1a8f1c414148276e1a3cba0146b094444f77c6a304cad319adccc7bae141c7d2a28a4c943154935963b434d36d5c8076237d8f1fd374ac0794e6ad12a9e6f27f85c8dc7cbecbb57e8a2c9d0065c080c0"

}

# **Why Besu?**

We support Client Diversity. Having many independently developed and maintained clients is vital for the health of a decentralized network. It's the reason, why we choose [Besu](https://besu.hyperledger.org/en/stable/) Java client.

Having many clients creates more room to explore ideas, algorithms, and architectures. It can also provide the inherent advantages of different languages, e.g., the binary size, speed, compatibility, etc., and act as a portal to a programming language community.

“The foundation of a client in a particular language opens and invites experimentation and innovation in that language. The base tooling around the client often snowballs into a robust ecosystem of tools and contributors in that language.”