Ganache 7 Ethereum Simulator - Building on Web3 is now easier and faster than ever before

By Kingsley Arinze, Mei Chan and David Murdoch

Ganache is a pioneer in the Ethereum development space, aiding DApp developers and enthusiasts to build, test, and explore blockchain since 2016. We are excited to announce the wide release of the latest version of Ganache today, with 30 times faster forking performance and an Infura integration that allows you to replay historical transactions with free access to archive data.

For the uninitiated, Ganache is a local development blockchain used to develop decentralized applications on the Ethereum blockchain. It simulates the Ethereum network, and you can see how your DApp will perform before you release it to production.

Typically, to start developing distributed applications on the Ethereum network, you will need to set up an Ethereum client such as Geth or OpenEthereum to provide you access to anEthereum Virtual Machine (EVM) .

Although that is a great way to develop distributed applications on Ethereum, it is not the most efficient and user-friendly approach, as you need to configure these clients manually and keep them running. Maintaining a self-hosted node can be expensive and time-consuming, you don't want to spend valuable time troubleshooting a failed node during development.

With Ganache, all you need to do is start the application and you have a preconfigured Ethereum client with 10 pre-funded and unlocked accounts ready for use. This allows you to quickly test your DApp throughout your development cycle.

Ganache 7: 30x faster forking and operations¶

Many Web3 developers building on Ethereum already use Ganache to fork and test smart contracts in their local environments for confidence before deployment. Armed with the knowledge gained from previous iterations, we rewrote Ganache from the ground up with the mission to make the blockchain simulator tool even more flexible, faster, and more stable.

We heard your feedback that Ganache v6 was slow to install, slow to run, and leaked memory. It also didn't use the modern language features and paradigms developers use today, like TypeScript types and the Promise abstraction. Ganache v6's architecture made testing, fixing bugs, and adding features difficult and time-consuming.

How is Ganache v7 better?

- With advanced caching capabilities on the latest version, Ganache's forking feature is up to 30 times faster than v6.
- Zero-config Mainnet forking! Ganache v7 has anative integration with Infura
- which enables free access to historical data
- You can run Ganache indefinitely without running into crashes due to memory issues
- Common operations are about 3 times faster than v6.

Installing and Setting up Ganache 7¶

Ganache can be used in many ways, including as a command-line tool, programmatically via Node.js, or in the browser. See the instructions below for installing and using Ganache 7 for all these use cases.

A. Using Ganache 7 On the command line¶

To run Ganache 7 on the command line, you must have Node.js >= v12.0.0 and NPM >= 6.4.1 installed on your computer. Seehere to download the latest version for your operating system, or use nvm to manage your Node installation foUnix orWindows .

With Node.js and NPM installed, you can install Ganache 7 globally by runningnpm install ganache --global . Once installed, you can runganache from anywhere in your command line.

Your terminal should look like this:

By default, Ganache provides 10 test accounts, each with 1000 (fake) Ether along with the corresponding private keys and the mnemonic phrase used to generate them. You can use this phrase to import the accounts into wallets likeMetaMask during development.

We can override all defaults by specifying options when starting Ganache 7; for example, we can have more control over account creation by providing a mapping of private keys to account balances like so:

ganache --wallet.accounts "0xfd485338e322f5930f7cf475f385341ec88bfc4f8a0a16f30b2fb417d1bb5427,

10000000000000000000" "0x05bba0b9f7a251080aa23feee4eab3f75a1abee905c0271008c93e5d2e2e7541, 10000000000000000000" We can also specify a mnemonic phrase to derive all initial accounts from, a miner gas price, and the block gas limit. For example:

ganache --miner.defaultGasPrice 200 --miner.blockGasLimit 90071 --miner.callGasLimit 898989 --wallet.mnemonic "alarm cause brave super lab glide awake hunt rose win sugar idea" Run the commandganache --help for a list of all the available options you can pass to Ganache.

We can also install ganache into a local Node.js project usingnpm install ganache without the--global flag, then add it to your project's package.jsonscripts , e.g.,"scripts": {"ganache": "ganache --mnemonic <12 words mnemonic>"} .

B. Using Ganache 7 programmatically¶

Ganache 7 can be used programmatically in your Node.js project as:

- A standalone EIP-1193 provider,
- A JSON-RPC server and EIP-1193 provider,
- A web3.js provider and
- An ethers.js provider.

See code samples for these different use cases below.

To begin, install Ganache 7 as npm package in your Node.js project

npm install ganache As a standalone EIP-1193 provider

const

ganache

=

require ("ganache"); const

options

=

{}; const

provider

=

ganache . provider (options); const

accounts

=

await

provider . request ({

method :

"eth_accounts" ,

params :

[]

}); ... As a JSON-RPC web server and an EIP-1193 provider

const

ganache

=

require ("ganache"); const

options

=

{}; const

server

=

ganache . server (options); const

PORT

=

8545 ; server . listen (PORT ,

err

=>

{

if

(err)

throw

err ;

console . log (ganache listening on port { PORT } ...);

const

provider

=

server . provider ;

const

accounts

=

await

provider . request ({

method :

"eth_accounts" ,

params : []

}); }); ... As a Web3.js provider

const

Web3

=

require ("web3"); const

ganache

=

require ("ganache"); const

web3

=

new

Web3 (ganache . provider ()); ... As an Ethers.js provider

const

ganache

=

require ("ganache"); const

provider

=

new

ethers . providers . Web3Provider (ganache . provider ());

C. Using Ganache 7 in the browser.¶

You can use Ganache 7 in the browser by adding the following script tag to your HTML code:

< script

src

"https://cdn.jsdelivr.net/npm/ganache@7.0.0/dist/web/ganache.min.js"

< /script> By doing that, Ganache 7 is automatically available in your browser for use:

const

options

=

{}; const

provider

=

Ganache . provider (options); Again, seehere for a list of available options you can pass in.

New Ganache 7 Features¶

1. Zero-config mainnet forking¶

Ganache supports mainnet forking out of the box without any configuration, allowing you to simulate having the same state as the Ethereum mainnet, but on your local machine.

This opens up many possibilities; for example, you can interact with a real-world protocol/contract locally during development, step through, and debug main net transactions locally (usingtruffle debug), allowing visibility to details about your contract.

Here's a 2016 demo of the Ganache team using the Mainnet forking feature to exploit the DAO hack .

To use this feature, start Ganache 7 using theganache --fork command. Ganache usesInfura as its provider under the hood by default, but you can also specify a provider by passing a URL; for example, you can use your own Infura URL by running this command:

ganache --fork.url wss://mainnet.infura.io/ws/v3/ Ganache fetches five blocks back from the latest by default to avoid missing blocks due to reordering.

This is a configurable behavior; you can either specify the exact block you want to fork from using--fork.blockNumberor by setting--fork.preLatestConfirmations 0 .

2. Fork any Ethereum test network without waiting for sync time¶

In addition to being able to fork the Ethereum main network with zero configuration, Ganache also allows you to fork from any Ethereum test network, including Ropsten, Kovan, Rinkeby, and Görli.

You can do that by running the fork command with the network option:ganache -- fork.network Cool, yeah? Here's what forking from the Rinkeby testnet looks like:

3. Support for massive transaction traces (over 10GB+)¶

Ganache supports massive transaction tracing using thedebug_traceTransaction RPC method.

To use this feature, start Ganache by forking off of Mainnet (or any test network) at a block number greater than that of the transaction you're trying to trace; for example, if you want to trace a transaction inside block 13,886,877, you will need to start Ganache at block number 13,886,878:

ganache --fork.network mainnet --fork.blockNumberThen send a curl request using the debug_traceTransaction method specifying the transaction hash like so:

curl -H 'Content-Type: application/json'

--data '{"jsonrpc":"2.0", "id": 1, "method": "debug_traceTransaction", "params": [**"**] }'

http://localhost:8545 This would return the aggregated summary of this transaction for post-processing.

Try it with a large transaction like0x8bb8dc5c7c830bac85fa48acad2505e9300a91c3ff239c9517d0cae33b595090 (Warp Finance hack):

First, start ganache (with lots of extra memory):

NODE_OPTIONS

--max-old-space-size= 16384

ganache --fork --fork.blockNumber 14037983

then executedebug_traceTransaction and send the output to trace.json:

curl -H 'Content-Type: application/json'

--data '{"jsonrpc":"2.0", "id": 1, "method": "debug_traceTransaction", "params": [
"0x8bb8dc5c7c830bac85fa48acad2505e9300a91c3ff239c9517d0cae33b595090"] }'

http://localhost:8545 -o trace.json

If you really want to test the limits of what's possible (and have a lot of time - about 1-2 hours) try out the 10GB+ Cream Finance hack transaction,0x0fe2542079644e107cbf13690eb9c2c65963ccb79089ff96bfaf8dced2331c92 !

4. Snapshot and Revert state¶

In blockchain terms, a snapshot refers to the act of capturing and recording the state of a blockchain at a particular block number. A snapshot is the complete view of the blockchain in that specific block and includes all existing addresses and their associated data, including transactions, fees, balance, metadata, etc.

In Ganache, you can take a snapshot of your development blockchain (or even a fork of Mainnet) using theevm_snapshot RPC method; it takes no parameters and returns an ID of the snapshot that it creates.

Ganache also supports reverting state to a previously taken snapshot using theevm_revert RPC method, although once a revert is invoked and succeeds, you can no longer use the snapshot with the corresponding ID as you can only revert a snapshot once.

You should consider creating a new snapshot after every evm_revert if you need to revert to the same point multiple times; here's a demo of taking a snapshot and reverting state in Ganache:

const

provider

=

ganache . provider (); const

[from ,

to]

=

await

provider . send ("eth_accounts"); const

startingBalance

=

BigInt (await

provider . send ("eth_getBalance" ,

[from]

)); // take a snapshot const

snapshotId

=

await

provider . send ("evm_snapshot"); // send value to another account (over-simplified example) await

provider . send ("eth_subscribe" ,

["newHeads"]

); await

provider . send ("eth_sendTransaction" ,

[{ from ,

to ,

value :

"0xffff" }]

); await

provider . once ("message");

// Note: await provider.once is non-standard // ensure balance has updated const

newBalance

=

await

provider . send ("eth_getBalance" ,

[from]

); assert (BigInt (newBalance)

<

startingBalance); // revert the snapshot const

isReverted

=

await

provider . send ("evm_revert" ,

[snapshotId]

); assert (isReverted); // ensure balance has reverted const

endingBalance

=

await

provider . send ("eth_getBalance" ,

[from]

); const

isBalanceReverted

=

assert . strictEqual (BigInt (endingBalance),

startingBalance); console . log ({ isBalanceReverted :

isBalanceReverted });

5. Mine blocks instantly, at interval, or on demand¶

Ganache allows you to configure how you want blocks to be mined during development; by default, blocks will be mined immediately upon receiving a transaction, but you can choose between the following options:

- At interval
- : You can specify the time in seconds Ganache should wait before mining the next transaction by passing the--miner.blockTime
- option. A block time of 0 means Ganache should mine new transactions instantly.
- On demand
- : Ganache also exposes anevm_mine
- RPC method which forces a single block to be mined whether mining is running or stopped. This will mine an empty block if no transaction exists in the mempool. See code sample below:

console . log ("start" ,

await

provider . send ("eth_blockNumber")); await

provider . send ("evm_mine" ,

[{ blocks :

5 }]

);

// mines 5 blocks console . log ("end" ,

await

provider . send ("eth_blockNumber"));

6. Fast forward time¶

Ganache exposes two RPC methods for manipulating time on your development blockchain.

A typical use case would be a smart contract that requires that a specific time passes before users can take certain actions; if the set time isn't something you want to wait for, you can use theevm_increaseTime to increase the blockchain current timestamp by the specified amount of time in seconds (passed in as hexadecimal).

curl -H 'Content-Type: application/json' --data' {"jsonrpc": "2.0", "id": 1, "method": "evm_increaseTime", "params": ["0x15180"] }' http://localhost:8545 This returns the total time adjusted in milliseconds. Furthermore, you can use theevm_setTime to set it to a specific timestamp; it accepts a JavaScript timestamp (Unix epoch) with millisecond precision and returns the number of seconds between the given timestamp and the current time.

curl -H 'Content-Type: application/json' --data '{"jsonrpc":"2.0", "id": 1, "method": "evm_increaseTime", "params": ["0x15180"] }' http://localhost:8545 You should use this method cautiously as it allows you to move backwards in time, which may cause new blocks to appear to be mined before older blocks, thereby invalidating the blockchain state.

7. Impersonate any account¶

Ganache allows you to impersonate another account during development. This is more interesting when used with the fork feature; you can fork Mainnet and impersonate any account.

Here's an example of how you can fork Mainnet, impersonate an account, and send some tokens to another account.

- Start Ganache by forking Mainnet and impersonating (unlocking) a random account:ganache --fork --wallet.unlockedAccounts
- Get the balance of the unlocked account:curl -H 'Content-Type: application/json' --data '{"jsonrpc":"2.0", "id": 1, "method": "eth_getBalance", "params": [""] }' http://localhost:8545
- . This returns the balance in WEI encoded as a hex number.
- Send some ether from the unlocked account to another account:curl -H 'Content-Type: application/json' --data '{"jsonrpc":"2.0", "id": 1, "method": "eth_sendTransaction", "params": [{"from": "UNLOCKED ACCOUNT", "to": "RECEIVING ACCOUNT", "value": "AMOUNT TO SEND"}] }' http://localhost:8545

Now verify that the unlocked account's balance has been reduced by the number of Ether sent out.

Sending Ether to another account isn't the only use case as you can impersonate an account to call certain owner-only functions in a smart contract as we did in the convex benchmark contract we saw earlier.

8. Pending transactions¶

In Ethereum, every transaction has a nonce. The nonce is the number of transactions sent from a given address to date, and anytime you send a transaction, the nonce increases by 1.

For a transaction to be mined, it has to have a nonce greater than that of the previous transaction by 1. In earlier versions of Ganache, if you send a transaction with a nonce greater than the last nonce by more than 1, Ganache would error, and the transaction would be rejected.

With Ganache 7, if the nonce of the previous transaction is 1, and you send a transaction with the nonce set to 3 for whatever reason, this transaction with nonce 3 will sit in the transaction pool until a transaction with nonce 2 is sent, at which time both transactions will be mined and added to the blockchain state.

Start using Ganache 7 today by simply runningnpm install ganache --global && ganache --help , or view the v7 Release Notes .

This release has been years in the making and we're really proud of the work we've done. We hope you love it as much as we do.

Thank you to everyone who has been a part of making this release happen — contributors, mentors, reviewers, issue reporters, and community participators have all been instrumental in making Ganache v7. We are immensely thankful to you all.

The Truffle Team