An homage to Truffle and its local Ethereum simulator, Ganache!

By Kingsley Arinze

Did you know the Truffle local Ethereum simulator was inspired by the original Ethereum testnet? This story and more as we look back on the history of Truffle.

Introduction¶

Testnets are a fundamental part of the blockchain ecosystem, as they allow dapp and smart contract developers to fully test their applications in an environment that closely resembles the blockchain mainnet, but without the risk of losing real funds.

Local blockchain simulators like Ganache take this a step further by making it possible to run an implementation of the Ethereum Virtual Machine (EVM) on your local machines, leading to faster testing and iteration speed during development.

From the early days of Ethereum, Truffle has established itself as a pioneer in identifying and building the right tooling to further the blockchain ecosystem.

With Layer 2 networks like Arbitrum, Polygon, and Optimism on the rise, there's a need for a more robust testing environment for dapp developers building on both layer 1 and layer 2 networks. New concerns arise, such as bridging between a layer 1 and a layer 2 or bridging between two layer 2s.

This blog tells the story of the early days of the Truffle local Ethereum simulator, Ganache, how it started, who and what contributed to its growth, and how it has evolved into becoming the center of the Ethereum development experience. We'll also touch on what the future holds for the simulator and Truffle as we seek to support and build a multi-chain future.

Eth-testRPC - The first Ethereum local simulator¶

The first Ethereum local simulator was created by Truffle's founder, Tim Coulter. It was derived from original Python scripts that were written by the CTO and Founder of Gnosis, Stefan D George, during the early days of ConsenSys - days that Tim refers to as "the days of throwing cupcakes at the wall to see which could stick." Eth-testRPC stuck!

Eth-testRPC was first introduced to the world by Tim at DevCon1 in January 2016. At the conference, Tim presented Dappstore, a before its time marketplace for dapps, and revealed how the Eth-testRPC had helped make it possible.

But this was only some days before Tim was about to give up on his Dappstore due to issues with the local simulator not responding to requests. It was around this time that Tim discovered an alternative, which although responded to RPC requests, required a lot of rewrites and fixes for it to work as intended. This local simulator was Etherism.

Etherism and the EthereumJS community - The entrance of Ethereum into the Javascript ecosystem¶

Etherism was created in September 2015 by Luris Matias and became Tim's new obsession. Tim needed additional features and RPC methods to support his Dappstore, but since his custom-built simulator wasn't working as intended, he immediately started contributing to the EtherSim repository .

About the same time, EthereumJS, a new community of Javascript developers and Ethereum enthusiasts, began to form.

EthereumJS quickly became the entry point and home for new Javascript developers entering the Ethereum ecosystem. Tim saw this and took advantage of it by porting over his version of EtherSim, which, at this point, had diverged significantly from the master branch, to the EthereumJS community under the name EthereumJS-testRPC.

2016 through 2017 - Mainnet forking¶

After joining the EthereumJS community, Truffle's local simulator, now EthereumJS-testRPC, would see the most growth and innovation. The Truffle team, together with other EthereumJS community members contributed to making the testRPC as close to the Mainnet as possible by adding other developer-focused features.

The most notable innovation that happened around that time was the mainnet forking feature , which allowed developers to simulate having the exact same state as the Ethereum mainnet, but on their local machines. Tim would go ahead to create a demo that used this feature to fork mainnet and replay the 2016 DAO hack .

Lately, Hardhat has received praise for creating the mainnet forking feature, but the first version of forking actually came out of the Truffle local simulator.

Ganache - The blockchain simulator is born

Over time, the EthereumJS-testRPC would differentiate itself within the EthereumJS community, and as Tim would put it, “like one of those things that was not like the others”. EthereumJS represented the Virtual Machine layer and the testRPC, an application on top of it that was primarily maintained by the Truffle team.

In the third year of conceiving and building the initial simulator, EthereumJS-testRPC was renamed to Ganache and became central to the web3 development experience for Ethereum dapp developers.

Eventually, the Truffle team released a standaloneGUI application that allowed users to see what was going on under the hood. Ganache’s monthly usage exploded with popularity, especially since this feature made blockchain development more accessible for beginners.

Truffle, which includes Ganache, has since grown to become one of the dominant tools in the blockchain ecosystem, amassing over 10 million downloads combined.

As the multi-chain world had just started to take shape, the Truffle team, again acting as pioneers, began experimenting with simulating other chains, including Tezos, Corda, and Filecoin. This is a testament to the vision of the team, who, at the time, could see that we were headed towards a multi-chain future.

Tezos, Corda, and Filecoin represent Truffle’s early experiments to see whether or not Ganache could become a generic framework for blockchain simulators.

Important

The Corda flavor is deprecated and has been removed from Ganache UI in v2.7.0. See therelease notes for details.

What does the future hold for Truffle and its local blockchain simulator?¶

The future of web3 is complex, characterized by multiple layer 1s and their corresponding layer 2s. It is clear, then, that the successful tools are the ones that can support and simplify all the complex nuances and problems that come with how these chains will interact.

At Truffle, we’ve identified these problems and have started making significant strides in solving them. Some of the features we’re most excited to launch/have launched are:

1. A protocol flavored Ganache
2. : We’ve been speaking to dapp developers over the past year and one recurring request we’ve kept getting is the need for a protocol flavored Ganache - one that would have the rules of these layer 2 protocols baked in. We’ve listened and are hard at work! We’re getting ready to launch the infrastructure for Ganache Plugins.
3. Ganache plugins would allow dapp developers building on layer 2s to continue using the Ganache experience they know and love. Imagine a world where a user would be able to have an Optimism Ganache, Arbitrum Ganache, and a Starknet Ganache!
4. Declarative deployment
5. : Imagine a scenario where you want to deploy contract A to a layer 2 blockchain, and then in the same breath, use the contract address of the deployed contract A to deploy contract B to another layer 2 or a layer 1 blockchain. To achieve this, you’d need to write and maintain potentially complex scripts to manage the process. Given changing requirements, manually maintaining these scripts is a recipe for bugs and a developer’s nightmare.
6. Our solution for this multi-chain deployment problem:
7. Truffle Declarative deployment
8. . The goal is to tell Truffle what your completed project should look like and then Truffle would take it up from there, deploying your contracts in the order you’ve described and to the chains or layers you’ve specified, all while maintaining state, handling contract dependencies as well as deploying to multiple RPC endpoints.
9. Truffle boxes
10. : These are boilerplate code, similar to the Ruby on Rails scaffold, where running a simple command on your CLI will scaffold a new project, including all the directories you need as well as a detailed README with deployment instructions and things to be mindful of. As of now, Truffle has theArbitrium box
11. , theOptimism box
12. , and thePolygon box
13. all live and available for dapp developers to use. We’re currently working on more advanced use cases for our boxes like the bridge box and starknet box.
14. Truffle dashboard
15. : This is a tool that removes the need to copy/paste your private keys into regular files during contract deployment. With Truffle dashboard running, you can connect through the UI and deploy contracts viaMetaMask
16. or Wallet connect. This was released in January, and we’re working on implementing our multi-network approach to the Truffle Dashboard as well.
17. Truffle for VS Code extension
18. : This is a tool that allows you to perform all the actions you would perform with Truffle, but all within VS Code. It allows you to compile, debug, and deploy your contracts. This was released a few weeks ago, and we’re working on making VS Code extension incorporate all these new layer 2 support tools.

If you haven’t already, you should definitely check out Truffle’s local Ethereum blockchain simulatorGanache , today, as well

as the Truffle suite of tools that are available for dapp developers. You can get started by first visiting the Truffle website and documentation .