Take a Dive into Truffle 5

Truffle 5 has had some exciting releases in the flexibilty they provide to your workflow. Truffle now offers the ability to version of Solidity in your contracts at compile time and allows for compilation from ABI files. Because of this, wrangling a project whose contracts use multiple version of Solidity is much easier going forward. We've also enhanced our testing capabilities with Solidity strack traces and better event decoding, making it much easier to get to the source of an error and get richer information from events. When our projects integrate with existing protocols, it's a pain to have to deal with those external contracts. Now Truffle makes this easy with the ability to fetch external contracts verified on Etherscan! Vyper support continues to improve with experimental debugging for Vyper contracts, as well as improved import handling for Vyper.

Did I mention support fordecentralized file storage ?

What's New in 5.2 and 5.3¶

IPFS, Filecoin, and Textile Bucket support viatruffle preserve

¶

Tests have ran, bugs are squashed, your smart contracts have been deployed successfully to mainnet; life is good. It's time to make your dapp accessible to the masses! So you... go to a centralized host and spin up the frontend? Something's definitely off here.

Entertruffle preserve : You can now use Truffle to upload your static assets toIPFS ,Filecoin , and/orTextile Buckets for a fully decentralized application across the entire stack!

Here are some companion pieces to help you get started: - TheFilecoin Truffle Box includes a decentralized art gallery to get you started with a fully working project! -The original Filecoin + Truffle & Ganache Blog Post -truffle preserve Documentation

Unbox a decentralized art gallery with the Filecoin Truffle Box.

Pragma compilation¶

As of version 5.2.0, Truffle now ships with a fun, experimental setting for compilation: the “pragma” setting. This feature analyzes your Solidity source files for their pragma expressions. It then uses these expressions to figure out which version of the Solidity compiler is required for each given source. This allows your project to compile when it contains multiple Solidity files, each requiring a different version of the Solidity compiler!

To use this feature, set the Solidity compiler version field in yourtruffle-config.js as follows:

module . exports

=

{

compilers :

{

solc :

{

version :

“ pragma ”

}

},

// … the rest of your config goes here }; Now you can runtruffle compile and your project will compile!

To illustrate this feature and the ones below, we have created aTruffle box for Truffle 5.2 .

In this box you can see how thetruffle-config.js is set up to use this new experimental feature. Notice thatcontracts/MetaCoin.sol requires^0.7.0 and howcontracts/Conversion.sol requires^0.6.0 . When runningtruffle compile on this example project you should get a printout of all the versions of the Solidity compiler used during compilation. Namely, you should see that both version0.6.12 and0.7.6 are listed.

When you use this feature, Truffle will analyze each of your Solidity sources one by one to find the appropriate version of the Solidity compiler for each source. Do note, however, that each source file and all of its imports must be compiled with one version of Solidity; this means there must be a version of the compiler that satisfies all of their pragma expressions.

Things you may have missed in Truffle 5.1 ¶

We've added a fair bit over the course of Truffle 5.1 as well, and there's a good chance you didn't catch all of it! We thought this would be a good time to review some of that as well!

Compilation of abi.json files¶

Another exciting addition to Truffle is the ability to compile from a contract's ABI. When you create a JSON source file in your "contracts" directory that contains a contract's ABI, Truffle will take it, use it to create an interface, and compile it. This might come in handy, for example, when you want to import and use a library that requires a different version of the Solidity compiler than the importing file. You can then simply compile the library and create a new JSON file containing the ABI of the imported library. Under the hood Truffle will compile the ABI into a Solidity interface and create an artifact for it. You will then be able to import the interface into your contract!

In the v5.2-example-box, consider thecontracts/MetaCoin.sol andcontracts/Conversion.sol . We would like to import Conversion into the MetaCoin contract to use it but cannot since they require different versions of the Solidity compiler. To work around this, we have createdcontracts/IConversion.abi.json which contains the ABI from the compiled Conversion. In this way we can now import the interface intocontracts/MetaCoin.sol (see line 4 in the MetaCoin contract) to use it!

Solidity stacktraces in Truffle Test¶

Try running your tests with the--stacktrace option (or-t for short) and get combined Solidity-Javascript stacktraces! This feature does have some limitations and is still somewhat experimental, but we expect you'll find it quite useful. In the future we may add support for Vyper stacktraces as well!

And speaking of Vyper...

Experimental Vyper debugging¶

You can now use Truffle Debugger with Vyper! Support for this is still in its early stages; you won't be able to inspect variables, I'm afraid. But you can step through a Vyper transaction, and there will likely be more in the future!

But perhaps the biggest improvement to the debugger is...

Debugging verified external contracts¶

Use the--fetch-external (or-x for short) option with Truffle Debugger and it will automatically download sources for any verified contracts and allow you to step through them! Contracts can be verified on either Etherscan or Sourcify. You no longer need to download these yourself and add them to your Truffle project! You can seethis earlier post for more information.

Improved import handling in Vyper¶

Here's another one for our Vyper users; you can now do imports from other projects (via NPM or EthPM) just like in Solidity!

For instance, supposeexamplepackage is an NPM package which is structured as a Truffle project, and you want to import a Vyper contract,VyperContract , from it. You can now do

from examplepackage.contracts import VyperContract or

import examplepackage.contracts.VyperContract as VyperContract to import it, much like you could do in Solidity!

Note that if your own project uses such "absolute" imports as a way to import its own files from the project root -- e.g., if you have a filecontracts/subdirectory/Contract1.vy which importscontracts/Contract2.vy via

import Contract2 as Contract2 then this will work within your own project, but won't work if other people try to import yourContract1 . We suggest that if you want other people to import contracts from your project, you use explicitly relative imports instead:

from .. import Contract2 But such imports are now possible. We hope you'll find all sorts of uses for this!

Enumeration values in Truffle Contract objects¶

When sending a transaction that takes anenum , it's inconvenient to have to look up the numeric constants that theenum uses. Well, now, each Truffle Contract constructor object contains enumeration constants, so you don't need to do that. Just do

contractInstance . exampleMethod (Contract . ExampleEnum . EnumValue); and don't worry about the particular numeric value ofEnumValue . You can also doContract.enums.ExampleEnum.EnumValue , if you want to be extra-certain.

Note that it's not currently possible to access enums declared outside of a contract this way, but we have plans in the future to make using those easier, too!

Even more

Other things you might have missed include: Improved event decoding intruffle test ; being able to use the debugger to step through Yul (assembly) sources generated by Solidity for its own internal subroutines (try theg command!); being able to inspect assembly variables in the debugger;truffle test --bail ;truffle create all ; and the debugger providing more information upon transaction reversion.

What's Next? Want to Help us Build it?¶

How will you use the new enhancements in Truffle? Let us know what you think about these features on Twitter, and if you have other needs we haven’t met yet.

WE'RE HIRING! Why stop at feature requests--cut out the middleman and work with us directly! There are many exciting problems to solve like layer 2 support, mapping decoding, and a revamped deployment system, to name a few. We'd love your help and perspective.Apply here on ConsenSys' website!