Upgradable-Proxy-Box¶

This box comes with everything you need to build upgradable-proxy contracts. We also have a simple react-app shipped along with it that you can use to see howyou can update a contract without losing any data .

Installation¶

First ensure that you are in an empty repository.

1. Create a new directory and unbox the box usingunbox
2. command using trufflemkdir
3. proxyBoxcd
4. proxyBox
5. truffle
6. unbox
7. pranav-singhal/upgradable-proxy-box
8. Install the box dependencies

yarn install 1. Run the development console 2. truffle 3. dev 4. OR bash truffle develop4. In the console now open run thecompileandmigrate5. ```js 6. compile // compile the contracts 7. migrate 8. // migrate the contracts 9. This will compile and migrate your contracts. It will also create the ABIs and put them inside the 10. app folder 11. Go into yourapp 12. folder andyarn install 13. the react app

cd

app yarn install 1. In the same folder,yarn start 2. the react app

yarn start

How it Works¶

The box has three parts

1. Contracts

There are 4 Primary Contracts 1.TokenStorage.sol - The contract that stores all the storage variables, their getter and setter functions. All storage variables must be defined inside this contract 2.TokenProxy.sol - This contract acts as the face of entire Dapp, all the calls from the client (web3) are made to this contract and this contract delagates request to the implementation contract. 3.Token_V0.sol - The first version of the implementation contract 4.Token_V1.sol - The second version of the implementation contract For Providing upgradeable nature to the Dapp, we write deploy 3 different contracts instead of just one. These contracts work hand in hand as shown in the flow chart below.

Flow Chart¶

The basic idea is to separate thestorage from the business logic . Hence the StorageTokenStorage.sol) (which includes, mappings and all the state variables) is deployed in a separate contract with all the getters and setters. The Implementation Contract(Token_V0.sol &Token_V1.sol) makes use of the storage contract for all of it's state variables.

The Proxy Contract(Token Proxy.sol) acts as the face of all the contracts. This contract delegates call to theimplementation contract which then uses Storage Contract .

Now, everytime we need to upgrade our implementation logic, we just need to deploya new Implementation Contract , and then call theupgradeTo function on Proxy Contract so that it starts delegating its call to the latest implementation contract. And hence, the storage is preserved and the calls from the browser are still made to the same Proxy Contract.

You can read more about how these contracts work here

1. Web3

theWeb3 folder inside the app directory gives you code for how to use the web3.js library to write Dapp code that can be used between different versions of your dapp. 1. React Appusing the contracts and the web3 code, we have built a sample dapp that provides a dynamic interface that changes with different versions of the Dapp.

Contributors¶

1. Arvind Kalra
2. Pranav Singhal

Also checkoutthis box that we've previously built to see how to use express with your Dapp