Aave Flash Loan Truffle Box¶

This Truffle box comes with everything you need to startdeveloping on flash loans

Installation and Setup¶

1. Install Truffle globally, if not already installed.npm install -g truffle@latest
2. Note: there is an issue with some older Truffle versions, e.g. v.5.1.25.This truffle box is confirmed working with the latest version (Truffle v5.1.32)
3. Download the box.truffle unbox aave/flashloan-box
4. Rename theenv
5. file to.env
6. and edit the following values in the file:* Sign up forInfura
7.
    - (or a similar provider) and replaceYOUR_INFURA_KEY
8.
    - with an API key for your project (this is called Project ID in the Infra dashboard).
9.
    - ReplaceYOUR_ACCOUNT_KEY_FOR_DEPLOYMENT
10.
    - with the private key of the ethereum account you will be using to deploy the contracts. This account will become theowner
11.
    - of the contract.
12. Ensure your ethereum account has some ETH to deploy the contract.
13. In your terminal, navigate to your repo directory and install the dependencies (if not already done):npm install
14. In the same terminal, replaceNAME_OF_YOUR_NETWORK
15. with eitherkovan
16. ,ropsten
17. , ormainnet
18. (depending on where you want to deploy the contract):truffle console --network NAME_OF_YOUR_NETWORK
19. You are now connected to the network you chose. In the same terminal window:migrate --reset
20. After a few minutes, your contract will be deployed on your chosen network.* If you have not added any profitable logic toFlashloan.sol
21.
    - line 23, then you will need to fund your contract with the desired asset.
22.
    - See ourdocumentation
23.
    - for token address and faucets.
24. Call your contract's flashloan function within the truffle console, replacingRESERVE_ADDRESS
25. with thereserve address
26. found in our documentation:let f = await Flashloan.deployed()
27. await f.flashloan(RESERVE_ADDRESS)
28.
    - if the above operation takes an unreasonably long time or timesout, tryCTRL+C
29.
    - to exit the Truffle console, repeat step 5, then try this step agin. You may need to wait a few blocks before your node can 'see' the deployed contract.
30. If you've successfully followed the above steps, then congratulations, you've just made a flash loan.* For reference, here is anexample transaction
31.
    - that followed the above steps onRopsten
32.
    - usingDai
33.
    - .
34.
    - For reference, here is anexample transaction
35.
    - that followed the above steps onRopsten
36.
    - usingETH
37.
    - .

Setup for cross protocol flash lending¶

If you are working across protocols, such as using the flash loaned amount on another #DeFi protocol, sometimes it is easier to fork mainnet and use each protocol's production contracts and production ERC20 tokens.

1. Follow the steps 0 --> step 4 from above.
2. (Install and) RunGanache
3. , preferably theCLI version
4. Intruffle-config.js
5. , ensure the details for thedevelopment
6. network match up with your running Ganache instance.
7. To minimise set up steps with Aave's lending pools, use Ganache's fork feature. This will 'fork' mainnet into your Ganache instance. Open terminal, replaceYOUR_INFURA_KEY
8. (this is called Project ID in the Infra dashboard) in the following and run:ganache-cli --fork https://mainnet.infura.io/v3/YOUR_INFURA_KEY -i 1
9. In a new terminal window in your repo directory, run:truffle console
10. Migrate your Flashloan contract to your instance of Ganache with:migrate --reset
11. After a few minutes, your contract will be deployed.* If you have not added any profitable logic toFlashloan.sol
12.
    - line 23, then you will need to fund your contract with the desired asset.
13.
    - See ourdocumentation
14.
    - for token address and faucets.
15. Your contract is now deployed on your local Ganache, which is mirroring mainnet. Call your contract's flashloan function within the truffle console, replacingRESERVE_ADDRESS
16. with thereserve address
17. found in our documentation:
18. let f = await Flashloan.deployed()
19. await f.flashloan(RESERVE_ADDRESS)
20. Be patient as your ganache instance works its magic.
21. If your implementation is correct, then the transaction will succeed. If it fails/reverts, a reason will be given.

Known issues¶

No access to archive state errors¶

If you are using Ganache to fork a network, then you may have issues with the blockchain archive state every 30 minutes. This is due to your node provider (i.e. Infura) only allowing free users access to 30 minutes of archive state. To solve this, upgrade to a paid plan, or simply restart your ganache instance and redploy your contracts.

Unable to debug executeOperation() with mainnet ganache fork¶

The Truffle debugger does not work too well with proxy / complex calls. You may find that the Truffle debugger returns an error such as:

TypeError: Cannot read property 'version' of undefined at ... - In this case you can try calling yourexecuteOperation() function directly, instead of having Aave'sLendingPool contract invoke the function. This will allow you to debug the function directly, however you will need to supply the relevant parameters (e.g._amount ,_fee ,_reserve , etc). - Alternatively, see the 'Troubleshooting' link.

Troubleshooting¶

See ourTroubleshooting Errors documentation.

Resources¶

- Ourflash loan documentation
- OurDeveloper Discord channel