

**Code flash loan/ flash loan arbitrage**

\_Words mentioned

Github

Solidity

Brownie

Hardhat

Remix

个 this symbol means reference to above sentence entry

∇ this symbol is referencing the below line of sentence entry.

\_\_\_\_\_\_\_\_\_\_\_\_

In GITHUB run a GITCLONE

Grab AAVE FLASHLOAN MIX at

[https://GitHub.com/brownie-mix/aave-flashloan-mix](https://github.com/brownie-mix/aave-flashloan-mix)

Work with V2 of AAVE FLASHLOANS within the contracts tab of repository on the left side of GitHub

(This contract interacts with the AAVE Protocol {protocol for borrowing/lending/partnered w/polygon which offers lower gas fees})

Grab imports ∇

FLASHLOAN reciever base ∇

{Makes the contract a FLASHLOAN contract}

∇(Interface) ILending pool address provider

∇(Interface) ILending pool

个{Pools in aave that are lenders giving the FLASHLOAN}

In the constructor of GitHub repository give the address of the ledning pool address provider.

个 when working w/ brownie ∇

              (    brownie-config.yaml    )

              The addresses for:

               MAINNET FORK: aave\_lending\_pool\_v2

               KOVAN: aave\_lending\_pool\_v2

               MAINNET: aave\_lending\_pool\_v2

AKA THE ADDRESSES OF LENDERS个

When making a FLASHLOAN (some code may be obstructed)

We have to write the functionality for making the loan

Getting the funds

& Repaying the funds in a specific function ∇

We have the FLASHLOAN function

Which takes the address of some asset (ERC20 token)

uint amount = 1 asset;

个 the amount of asset you're borrowing.

( Can also add more tokens for the same transaction )

By adding the erc20 address to the list of assets

& Adding the amounts to the list of amounts.

At line 66 in GitHub repository

LENDING\_POOL. flashloan, (aka lending pool address given)

Reciever address, (aka getting assets back into this contract)

Assets, (erc20 token)

Amounts, (amount of erc20 tokens being borrowed)

On behalf, the lp address

The lending pool will an execute operation function.

We have to approve the token transfer back to the lending pool so we can repay when it calls it.

Can add logic to do the rest of the functionality.

(Don't worry about initiator perams)

Once logic is done.

You have to approve the lending pool to get the tokens back because the lending contract is going to call transfer from and try to get the funds back. If it can't it will null out your entire functionality (everything you're doing)

-------------

Steps:

Go here [https://GitHub.com/brownie-mix/aave-flashloan-mix](https://github.com/brownie-mix/aave-flashloan-mix)

Install BROWNIE

Install Ganache-CLi

Sign up for INFURA & generate API key

• In .env within GitHub

[https://GitHub.com/brownie-mix/aave-flashloan-mix](https://github.com/brownie-mix/aave-flashloan-mix)

• Export: WEB3\_INFURA\_PROJECT\_ID=YourProjectID

\* Don't worry about etherscan

• Export: your private key (PRIVATE\_KEY= 0x< my private key address here)

• Get your private key from METAMASK by exporting private key

( must add 0x at the start of your address when exporting from METAMASK.

Go to:

<https://faucet.kovan.network>

Get $KETH

Send it to your METAMASK Kovan Test Network address

Once all keys are in .env

Run [~/code/aave-flashloan-mix - (master)] $ source .env

个

Then run script ( [get\_\_weth.py](http://get__weth.py) ) which is a script to get wrapped eth. AKA $WETH

\* this grabs your key from your brownie config which you have set up with your private key

• Add the network Kovan

• Deposit 1 $keth to pay the premium/fee of the FLASHLOAN to get $weth

$ brownie run scripts/ [get\_weth.py](http://get_weth.py) ---network Kovan

This creates a transaction on the Kovan testnet

\*You can use the local net or MAINNET FORK

$ brownie run scripts/ [deployment\_v2.py](http://deployment_v2.py) ---network Kovan

个 this will grab our private key and deploy our flashloan contract using the AAVE lending pool address found in the config (since using KOVAN it will be next to aave\_lending\_pool\_v2: 0x88757f.....)

\*If you want to add flashloan with polygon you just have to add the polygon network.

• Last step--

$ brownie run scripts/[flashloan\_v2.py](http://flashloan_v2.py) ---network Kovan

\*This function above is Borrowing weth,to do something with it(i.e dex arbitrage), repay the weth right back.