Lesson 13 Hardhat DeFi & Aave 19.16.15

What is DeFi ?

https://chain.link/education/defi

Traditional Agreements

Performance Data

0010111011

Contract Terms

Participants

1100100101

0110010010

Contract Operator Conflicts of Interest (Pricing Monopolies)

Cryptographic Agreements

Performance Data

〈0010111011〉

Contract Terms

Participants

〈1100100101〉

〈0110010010〉

CeFi Financial Products Will Not Support the Average User

VICE

Robinhood Stops Users From Trading GameStop Stocks, Other Reddit YOLO Picks

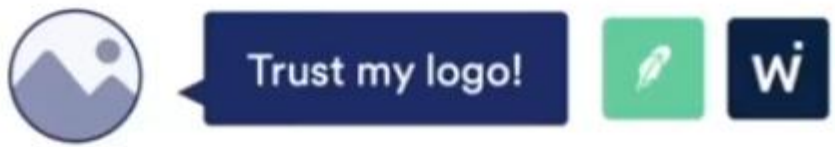
Chainlink


SAFE DEPOSIT VAULTS
AMERICAN UNION BANK
4 % PAID ON THRIFT ACCOUNTS
AMERICAN UNION BANK

Smart Contracts Will Solve Society's Critical Trust Issues

> **Counterparty Risk:** the likelihood or probability that one of those involved in a transaction might default on its contractual obligation.

Paper Guarantees (Brand Based)



- **Counterparty risk is high and opaque**
- Transparency is purposefully removed
- Interest yields are low and going lower

Cryptographic Guarantees (Math Based)

- **Counterparty risk is low and transparent**
- Transparency is unavoidably built-in
- Interest yields are consistently high

https://hackernoon.com/what-is-the-math-behind-elliptic-curve-cryptography-f61b25253da3

Chainlink

Access to Off-chain Resources Enables More Contracts

Off-chain Market Data

Off-chain Gaming Data

Off-chain Insurance Data

Various Proof of Reserves

New Data and Computations

Chainlink Network

DeFi Smart Contracts

Gaming Smart Contracts

Insurance Smart Contracts

ARBOL ETHERISC

Nexus Mutual

Wrapped + Stable Coins

WBTC Ren

and many more assets...

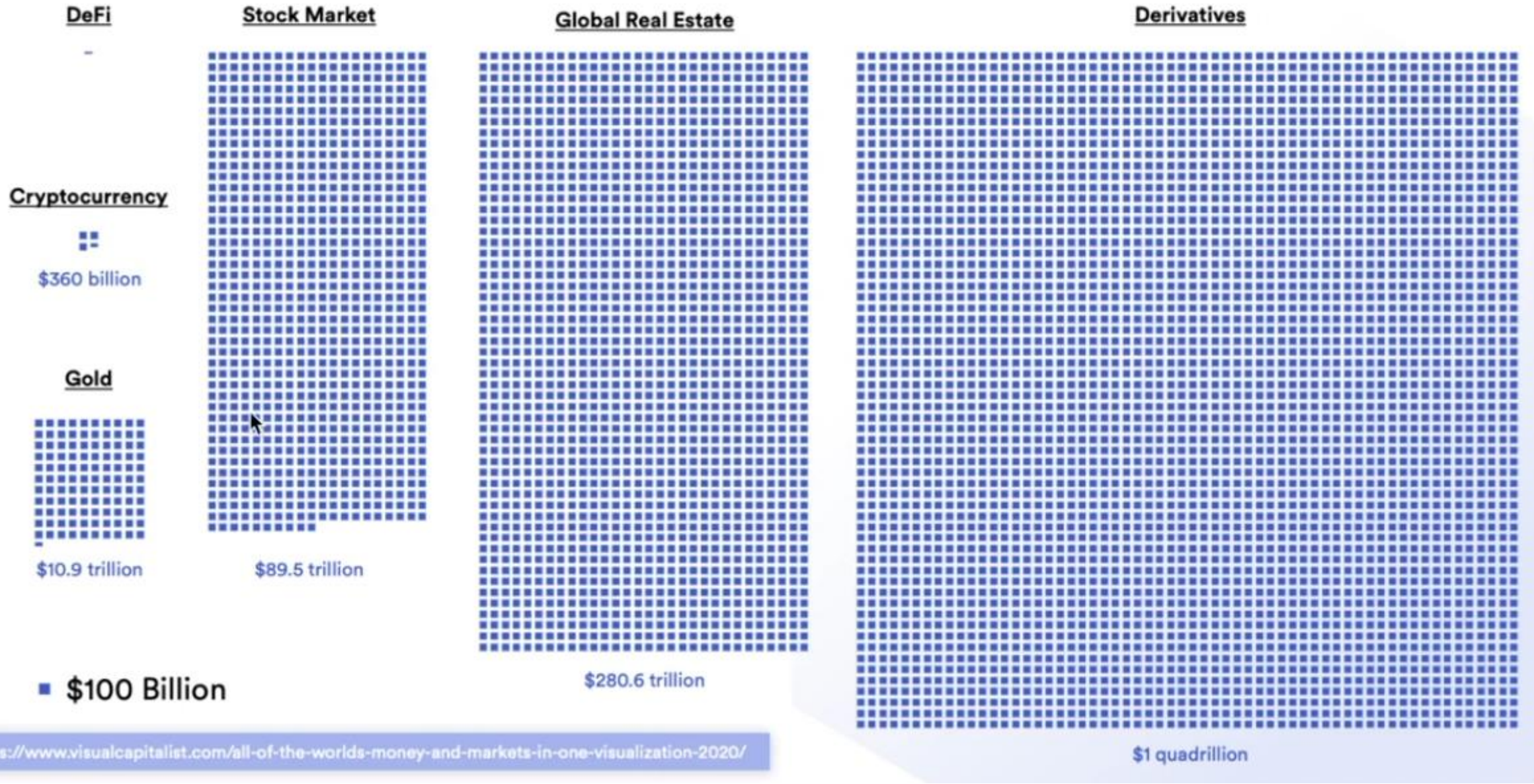
New Smart Contracts

Millions of new smart contracts

ethereum

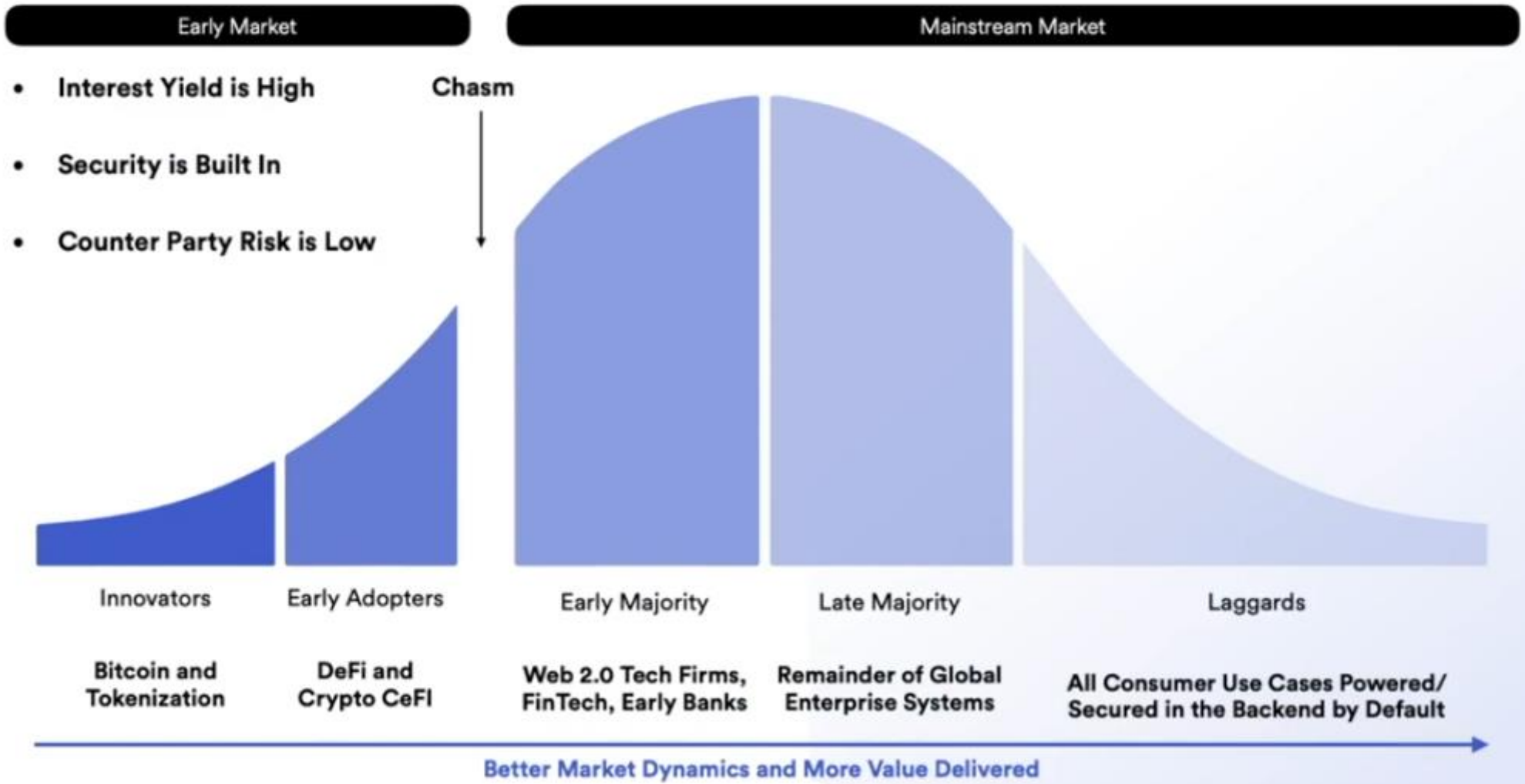
Chainlink

The Remaining Market For Smart Contracts is Trillions



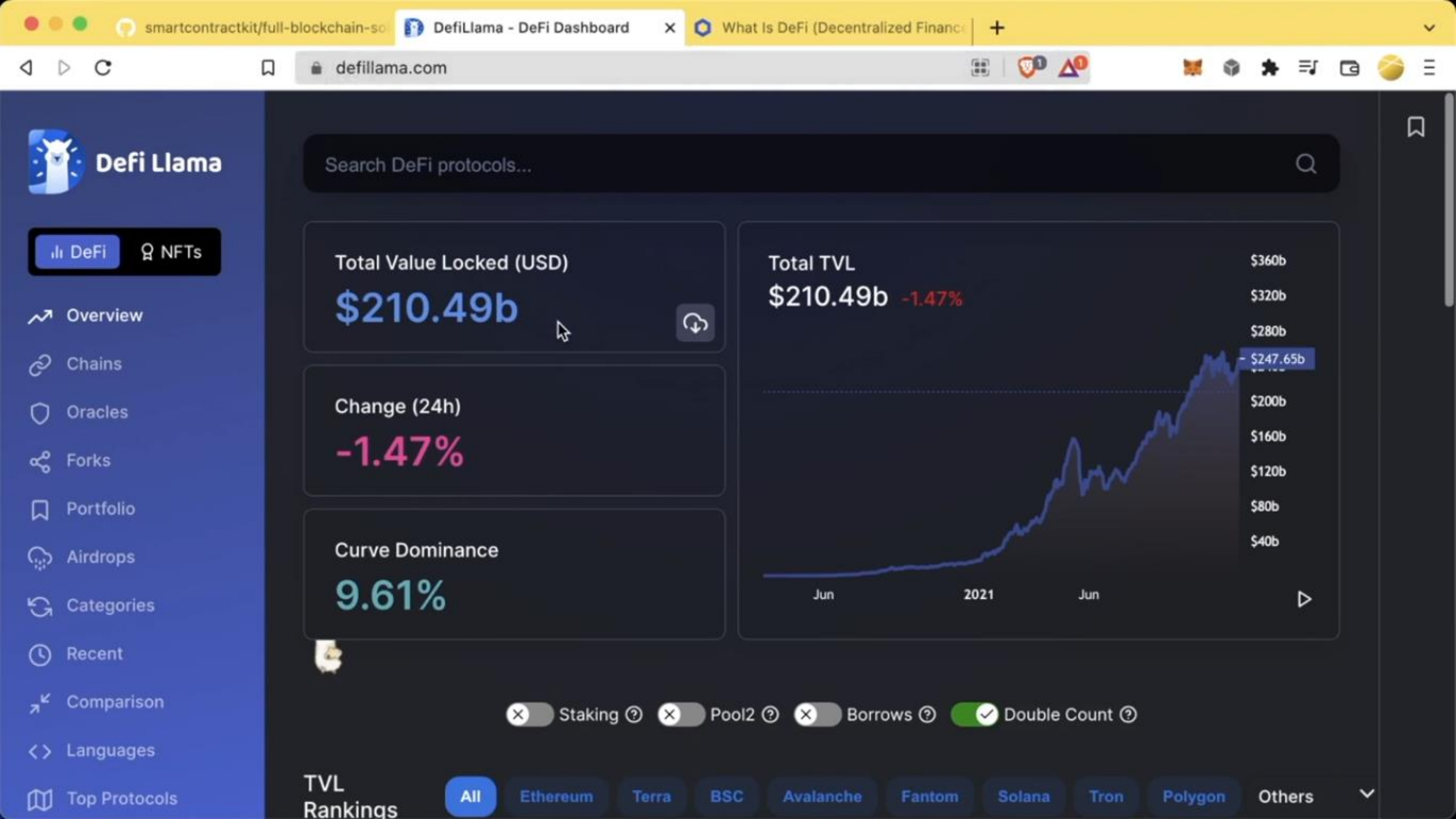
https://www.visualcapitalist.com/all-of-the-worlds-money-and-markets-in-one-visualization-2020/

Decentralized Finance Needs to Gain Larger Adoption



tps://en.wikipedia.org/wiki/Crossing_the_Chasm

Chainlink


smartcontractkit/full-blockchain-so
DefiLlama - DeFi Dashboard
What Is DeFi (Decentralized Financ
defillama.com
Defi Llama
Search DeFi protocols...
DeFi
NFTs
Overview
Chains
Oracles
Forks
Portfolio
Airdrops
Categories
Recent
Comparison
Languages
Top Protocols
Total Value Locked (USD)
$210.49b
Change (24h)
-1.47%
Curve Dominance
9.61%
Total TVL
$210.49b -1.47%
$360b
$320b
$280b
$247.65b
$200b
$160b
$120b
$80b
$40b
Jun
2021
Jun
Staking
Pool2
Borrows
Double Count
TVL Rankings
All
Ethereum
Terra
BSC
Avalanche
Fantom
Solana
Tron
Polygon
Others

What is Aave? 19.19.59

https://aave.com/

Programmatic Borrowing & Lending 19.25.47

```
We dont create smart contract. We will learn interact with protocol aave, makerdao etc
```

https://docs.aave.com/developers/v/2.0/

yarn add --dev hardhat

Yarn hardhat

Create an empty hardhat.config.js copy settings from github

Copy dependencies from lesson 9 to install

yarn add --dev @nomiclabs/hardhat-ethers@npm:hardhat-deploy-ethers ethers @nomiclabs/hardhat-etherscan @nomiclabs/hardhat-waffle chai ethereum-waffle hardhat hardhat-contract-sizer hardhat-deploy hardhat-gas-reporter prettier prettier-plugin-solidity solhint solidity-coverage dotenv

Copy prettier files

1. Deposit collateral : ETH / WETH
2. Borrow another asset : DAI-makerdao.com/en/-stable coin on makerdao blockchain- 1dolar -
3. Repay the DAI

CREAATE FOLDER AND FİLE

scripts/aaveBorrow.js

```
async function main() {
    // the protocol (aave) treats everything as an ERC20 token
// BUT ETHEREUM AND NATIVE BLOCKCHAIN TOKEN THAT YOURE USING, IS NOT AN ERC20
TOKEN
}
main()
    .then(() => process.exit(0))
    .catch((error) => {
        console.error(error)
        process.exit(1)
    })
```

WETH Wrapped ETH	19.30.49

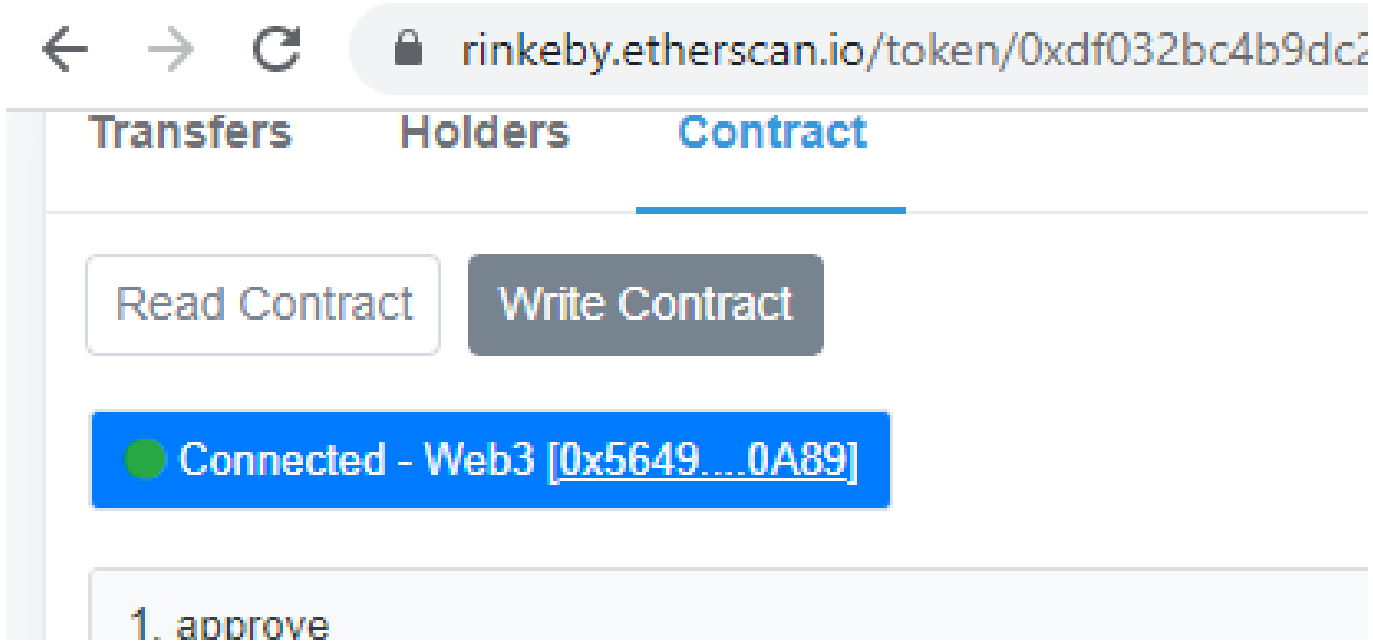
scripts/getWeth.js

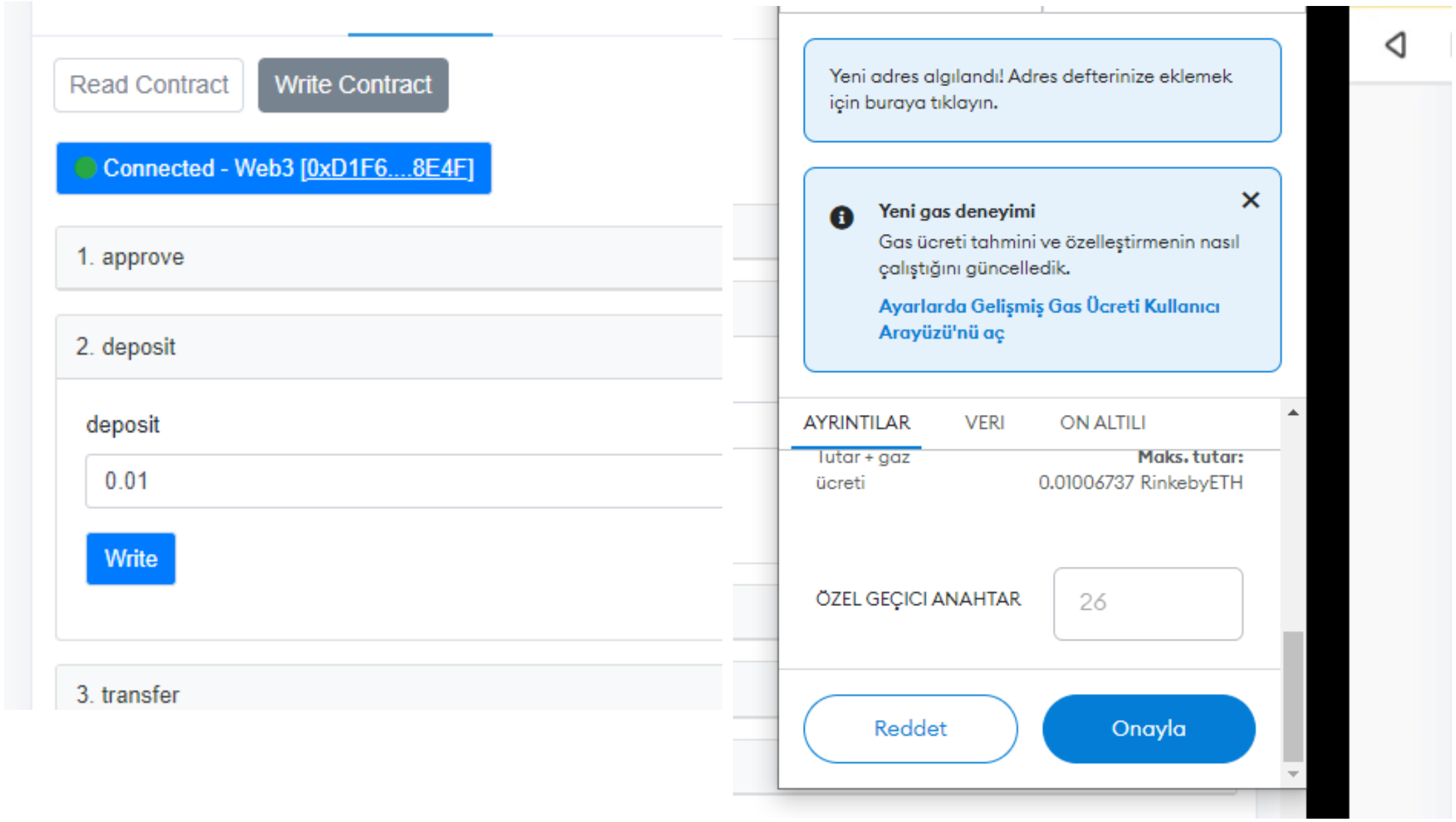
https://rinkeby.etherscan.io/token/0xdf032bc4b9dc2782bb09352007d4c57b75160b15

```
Will deposit our token for web token
```

To test with metamask

https://rinkeby.etherscan.io/token/0xdf032bc4b9dc2782bb09352007d4c57b75160b15

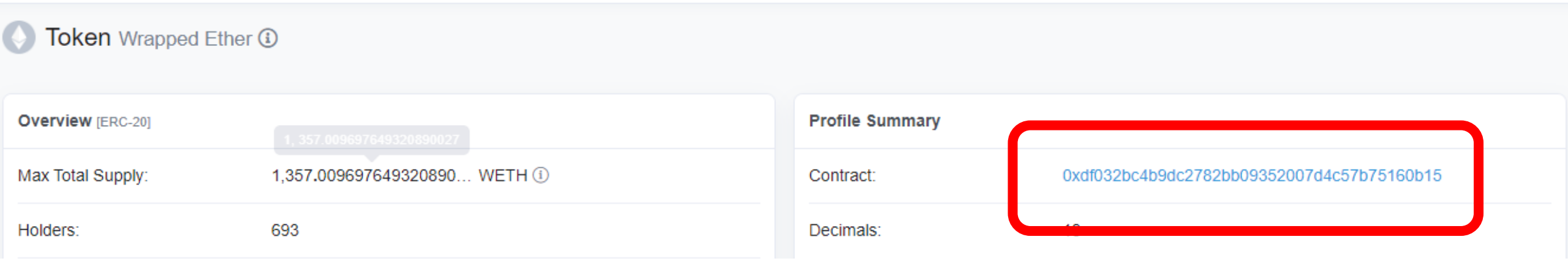


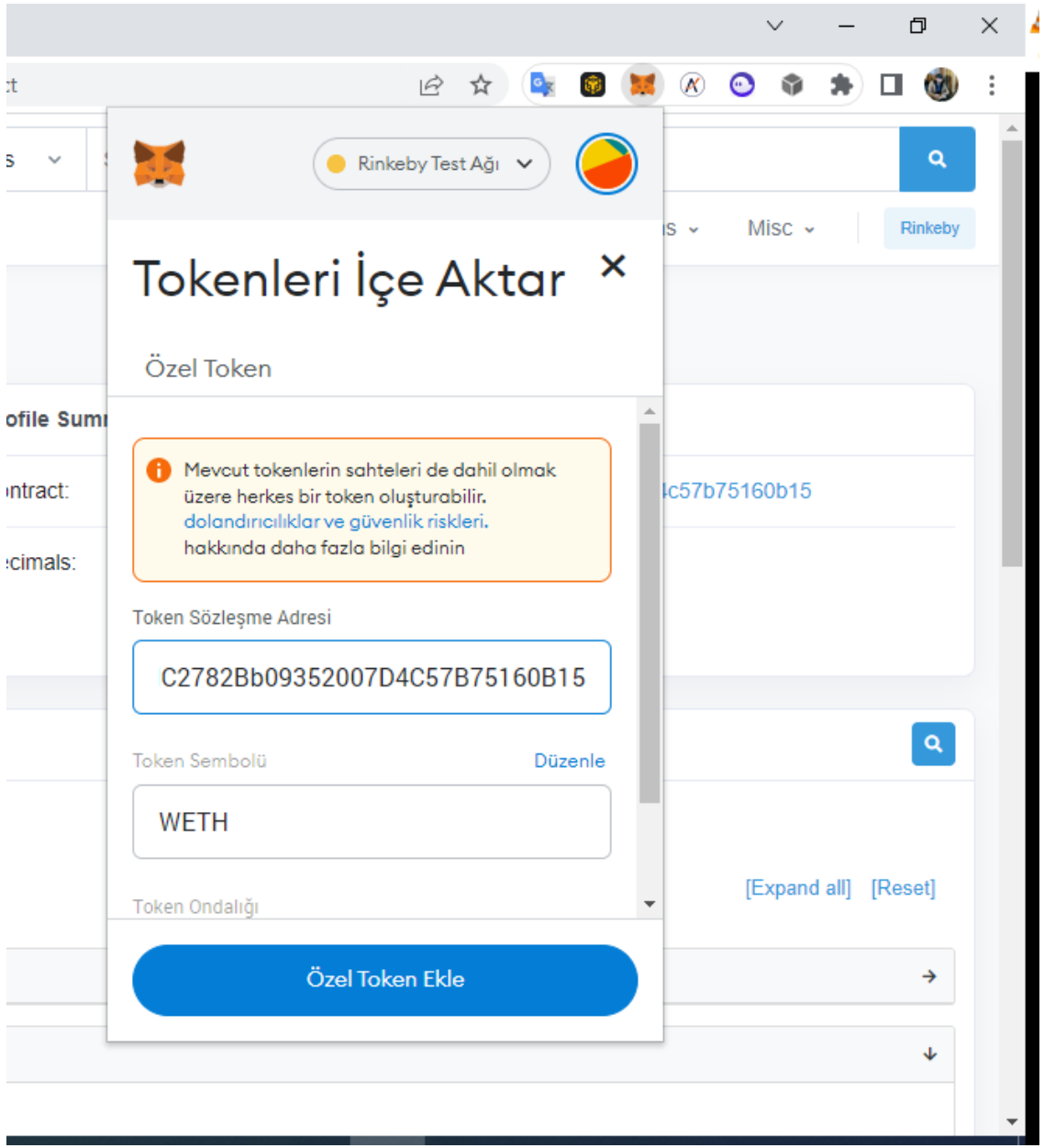

Read Contract
Write Contract
Connected - Web3 [0xD1F6....8E4F]
1. approve
2. deposit
deposit
0.01
Write
3. transfer
Yeni adres algılandı! Adres defterinize eklemek için buraya tıklayın.
Yeni gas deneyimi
Gas ücreti tahmini ve özelleştirmenin nasıl çalıştığını güncelledik.
Ayarlarda Gelişmiş Gas Ücreti Kullanıcı Arayüzü'nü aç
AYRINTILAR
VERI
ON ALTILI
Tutar + gaz ücreti
Maks. tutar:
0.01006737 RinkebyETH
ÖZEL GEÇICI ANAHTAR
26
Reddet
Onayla

View your transaction

Add token to metamask

COPY CONTRACT ADRES




Rinkeby Test Ağı
Misc
Rinkeby
Tokenleri İçe Aktar
Özel Token
Mevcut tokenlerin sahteleri de dahil olmak üzere herkes bir token oluşturabilir.
dolandırıcılıklar ve güvenlik riskleri.
hakkında daha fazla bilgi edinin
Token Sözleşme Adresi
C2782Bb09352007D4C57B75160B15
Token Sembolü
Düzenle
WETH
Token Ondalığı
Özel Token Ekle
[Expand all]
[Reset]

İn getweth.js we dont use main function, we will import it to aaveBorrow.js

In getweth we will call Weth Ethers deposit function.
For this its needs abi and contract address and we use interfaces to get abi and contract address

```
const { getNamedAccounts } = require("hardhat")

async function getWeth() {
    const { deployer } = await getNamedAccounts()
    // call the "deposit" function on the weth contract
    // abi, contract address -- we will do this with interfaces
}
module.exports = { getWeth }
```

CREATE
contracts/interfaces/IWeth.sol

```
pragma solidity ^0.4.19;

interface IWeth {
    function allowance(address owner, address spender) external view returns (uint256
remaining);

    function approve(address spender, uint256 value) external returns (bool success);
    function balanceOf(address owner) external view returns (uint256 balance);
    function decimals() external view returns (uint8 decimalPlaces);
    function name() external view returns (string memory tokenName);
    function symbol() external view returns (string memory tokenSymbol);
    function totalSupply() external view returns (uint256 totalTokensIssued);
    function transfer(address to, uint256 value) external returns (bool success);
    function transferFrom(        address from,        address to,        uint256 value
    ) external returns (bool success);
    function deposit() external payable;
    function withdraw(uint256 wad) external;
}
```

Add compiler solidity version for interface

hardhat.config.js

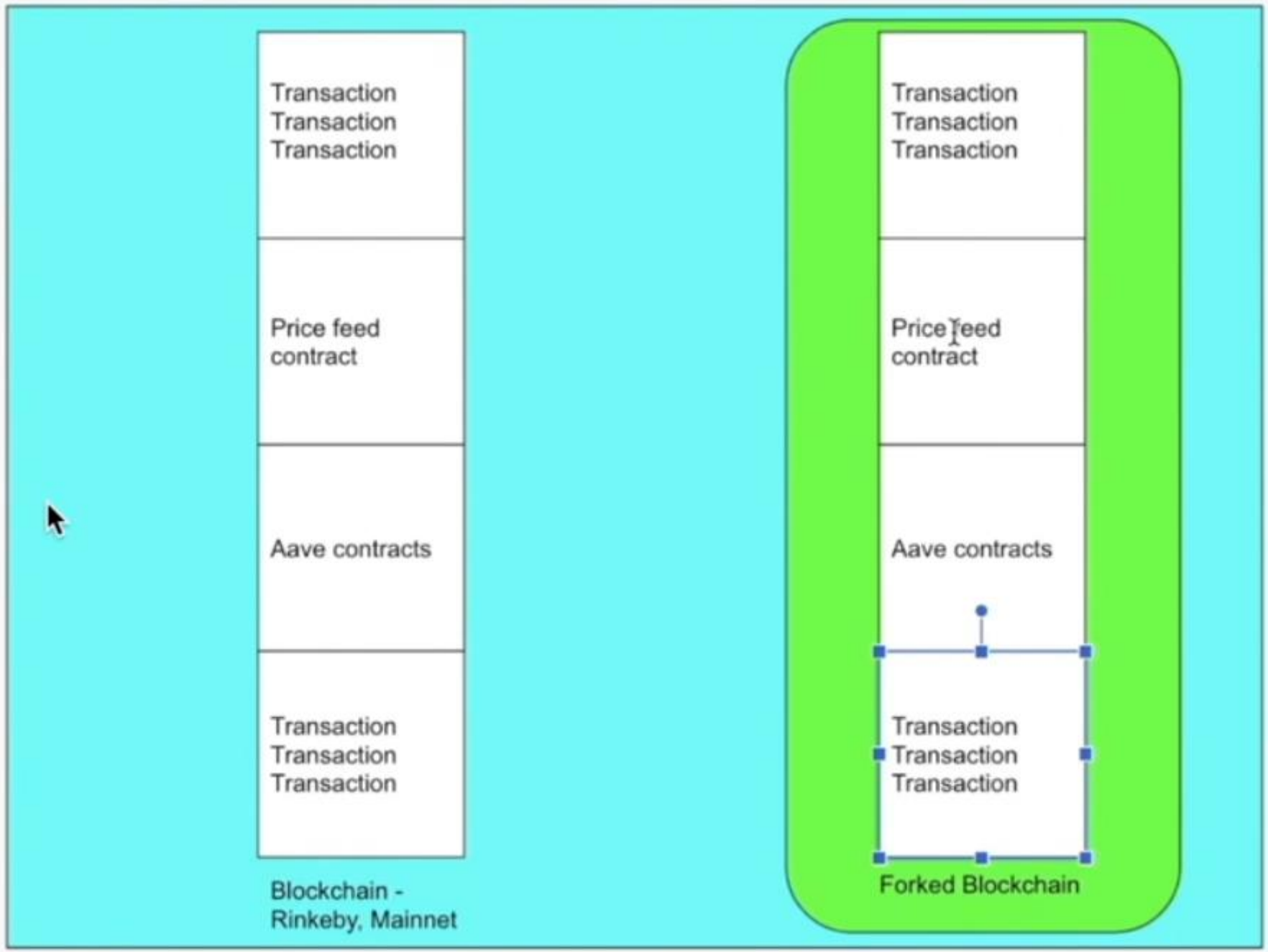
```
module.exports = {
    solidity: {
        compilers: [{ version: "0.8.8" }, { version: "0.4.19" }],
    },
```

```
Yarn hardhat compile  // run this command to get abi
```

```
WETH mainnet address https://etherscan.io/token/0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2

Token address 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2
```

```
const { getNamedAccounts, ethers } = require("hardhat")
const AMOUNT = ethers.utils.parseEther("0.02")

async function getWeth() {
    const { deployer } = await getNamedAccounts()
    // call the "deposit" function on the weth contract
    // abi, contract address  -- we will do this with interfaces
    // 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2
    const iWeth = await ethers.getContractAt(
        "IWeth",
        "0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2",
        deployer
    )
    const tx = await iWeth.deposit({ value: AMOUNT })
    await tx.wait(1)
    const wethBalance = await iWeth.balanceOf(deployer)
    console.log(`Got ${wethBalance.toString()} WETH`)
}
module.exports = { getWeth }
```

Forking Mainnet 19.38.11

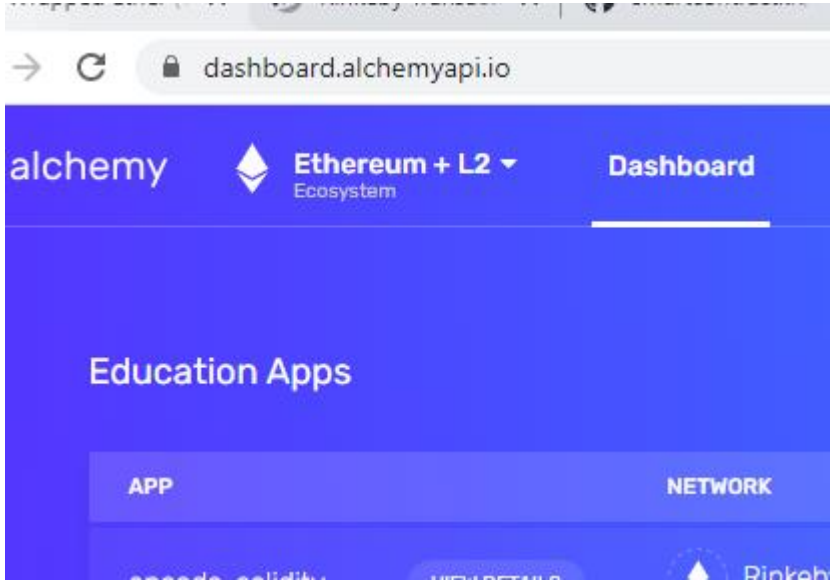

Transaction
Transaction
Transaction
Price feed
contract
Aave contracts
Transaction
Transaction
Transaction
Blockchain -
Rinkeby, Mainnet
Transaction
Transaction
Transaction
Price feed
contract
Aave contracts
Transaction
Transaction
Transaction
Forked Blockchain

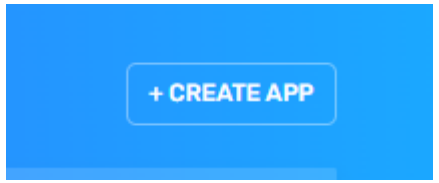
We used for WETH real network adress. Because we will fork mainnet. That means we wiil copy mainnet to local pc.
That not means all mainnet data ☺). Just used.
So we can use this forked mainnet for tests, local environment usage etc.

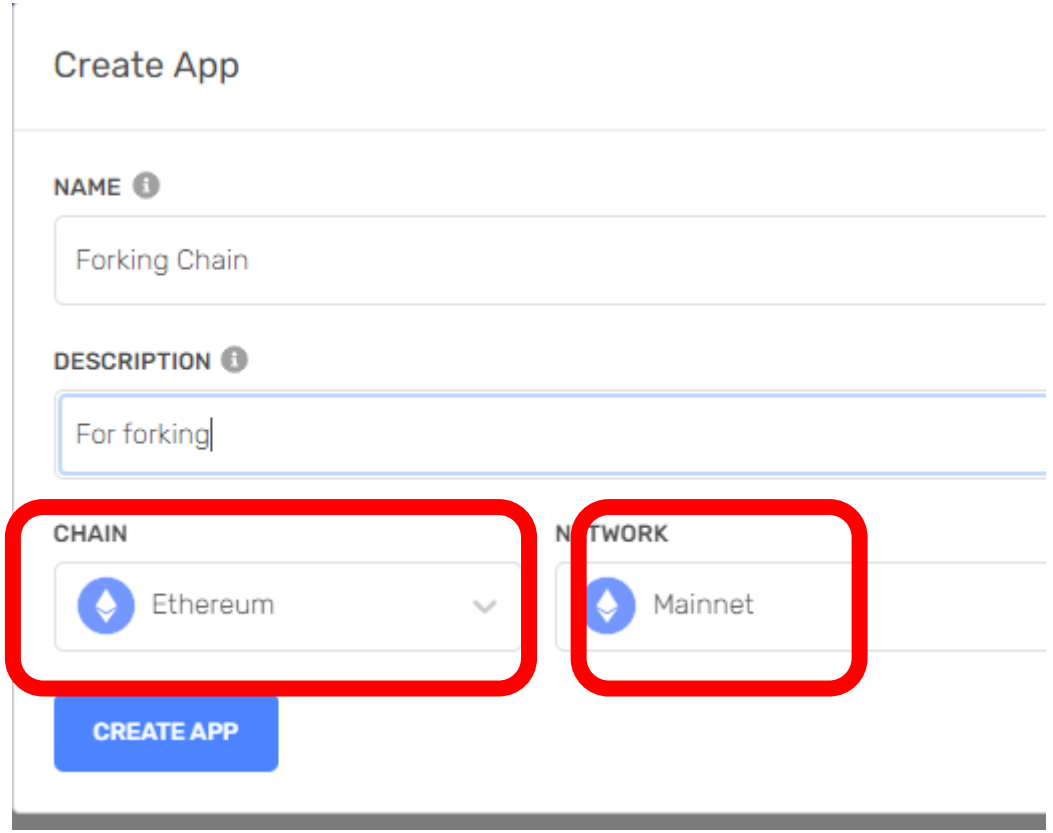
We can use fork mainnet for many thing
// TradeOffs
Pros : Quick, easy, resemble whats on mainnet
Cons: We need an API, some contracts are complex to work with. Mocks can be better to use

Add config for mainnet forking

```
        hardhat: {
            chainId: 31337,
            forking: {
                url: MAINNET_RPC_URL,
            },
        },
```


dashboard.alchemyapi.io
alchemy
Ethereum + L2
Ecosystem
Dashboard
Education Apps
APP
NETWORK


+ CREATE APP


Create App
NAME
Forking Chain
DESCRIPTION
For forking
CHAIN
Ethereum
NETWORK
Mainnet
CREATE APP


```
https://eth-mainnet.g.alchemy.com/v2/I6ESUowHbkhcAfdjSHAsUxhhFbtMShHj
```

```
Add to .env file
And config file
```

scripts/aaveBorrow.js

```
const { getWeth } =
require("../scripts/getWeth")
async function main() {
    // the protocol (aave) treats everything as
an ERC20 token
    await getWeth()
}
```

RUN SCRIPT (default network is hardhat)

yarn hardhat run scripts/aaveBorrow.js

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$ yarn hardhat run scripts/aaveBorrow.js
yarn run v1.22.18
warning package.json: No license field
$ /home/eemcs/freecodecamp/hardhat-defi-fcc/node_modules/.bin/hardhat run scripts/aaveBorrow.js
Got 20000000000000000 WETH
Done in 8.12s.
```

0.020000000000000000 WETH

So with this, we have a way to interact with main net locally

Now we can simulate and see transactions.

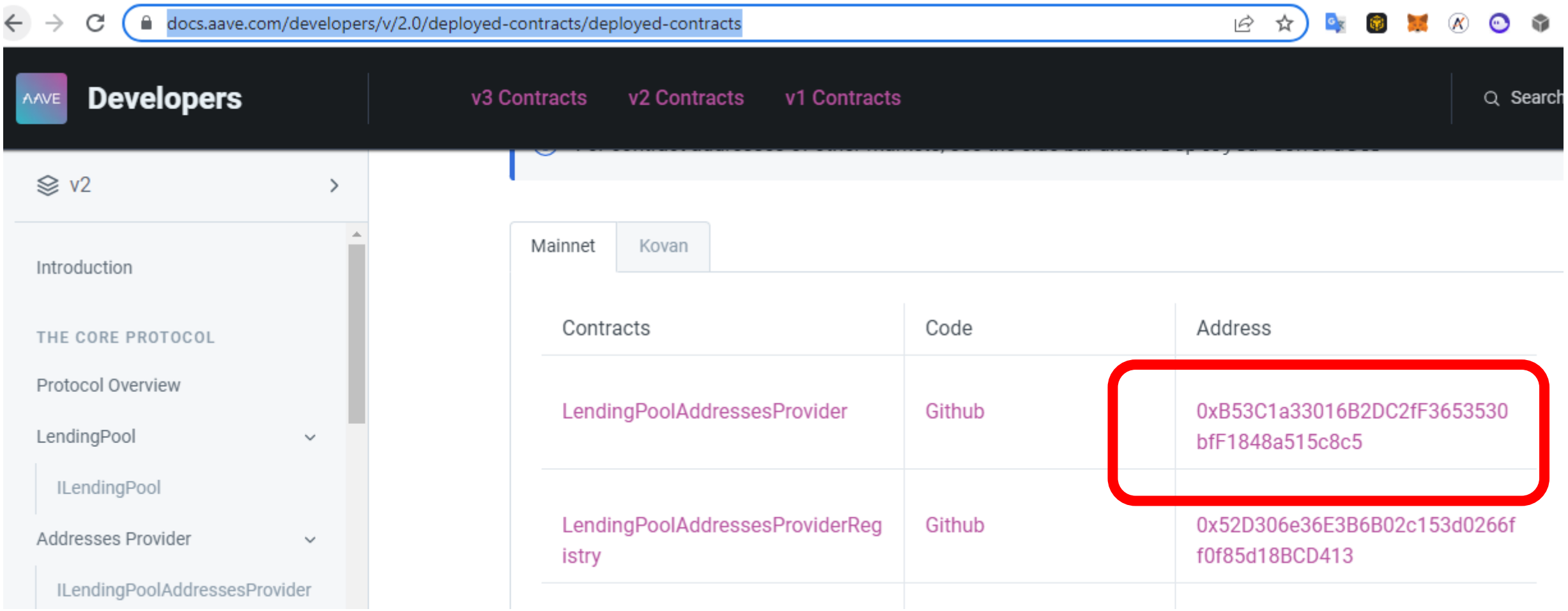
Depositing into Aave 19.45.20

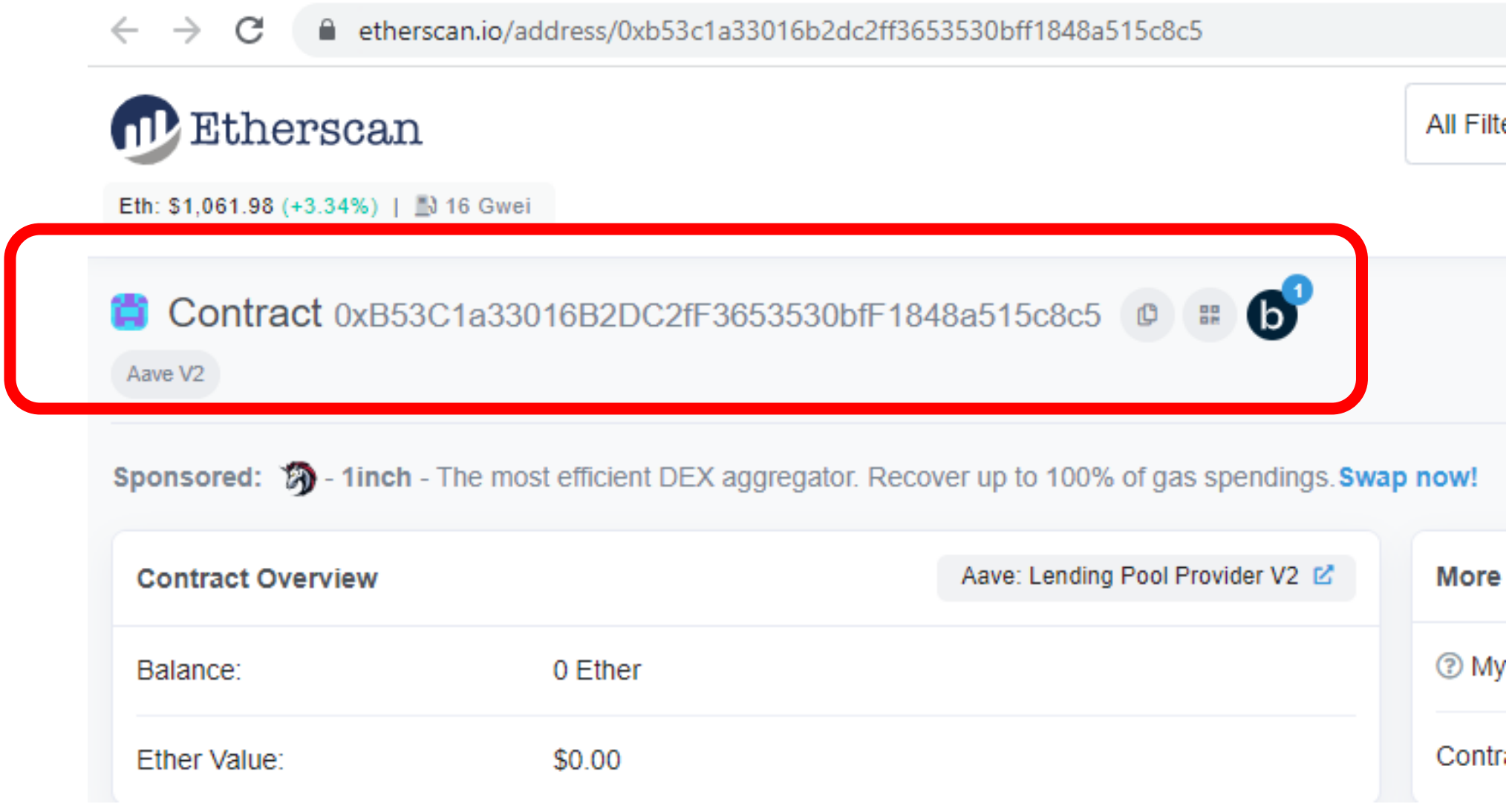
scripts/aaveBorrow.js

```
async function main() {
    // the protocol (aave) treats everything as an ERC20 token
    await getWeth()
    const {deployer} = await getNamedAccounts()
    // we need abi and aave contract adress
// to get address first we get IlendigPoolAddressesProvider
}
```

Get address (provider) from aave docs in next slayt

https://docs.aave.com/developers/v/2.0/deployed-contracts/deployed-contracts




etherscan.io/address/0xb53c1a33016b2dc2ff3653530bff1848a515c8c5
Etherscan
All Filte
Eth: $1,061.98 (+3.34%) | 16 Gwei
Contract 0xB53C1a33016B2DC2fF3653530bfF1848a515c8c5
Aave V2
Sponsored: - 1inch - The most efficient DEX aggregator. Recover up to 100% of gas spendings. Swap now!
Contract Overview
Aave: Lending Pool Provider V2
More
Balance:
0 Ether
My
Ether Value:
$0.00
Contr

0xB53C1a33016B2DC2fF3653530bfF1848a515c8c5

```
async function getLendingPool() {
    // abi, address
}
```

Using pool we will get the addres to use for deposit

Here, wee need to ILendingPoolAddressesProvider (this is to get addres for deposit adres)

First implement interface from docs

https://docs.aave.com/developers/v/2.0/the-core-protocol/addresses-provider/ilendingpooladdressesprovider

create contracts/interfaces/ILendingPoolAddressesProvider.sol

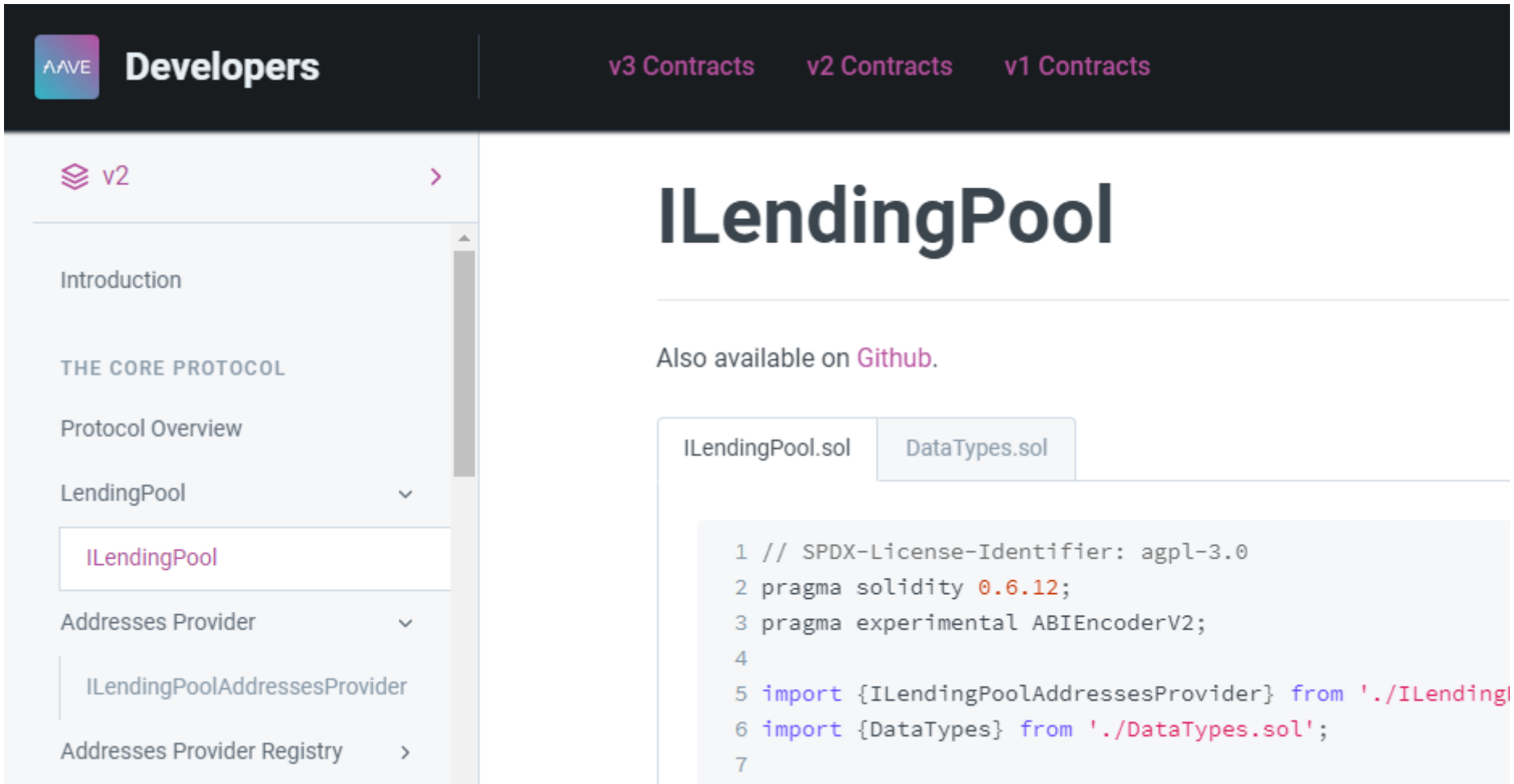
```
// SPDX-License-Identifier: agpl-3.0
pragma solidity 0.6.12;

/**
 * @title LendingPoolAddressesProvider contract
 * @dev Main registry of addresses part of or connected to the protocol, including permissioned roles
 * - Acting also as factory of proxies and admin of those, so with right to change its implementations
 * - Owned by the Aave Governance
 * @author Aave
 **/
interface ILendingPoolAddressesProvider {
  event MarketIdSet(string newMarketId);
  event LendingPoolUpdated(address indexed newAddress);
  event ConfigurationAdminUpdated(address indexed newAddress);
  event EmergencyAdminUpdated(address indexed newAddress);
  event LendingPoolConfiguratorUpdated(address indexed newAddress);
  event LendingPoolCollateralManagerUpdated(address indexed newAddress);
  event PriceOracleUpdated(address indexed newAddress);
  event LendingRateOracleUpdated(address indexed newAddress);
  event ProxyCreated(bytes32 id, address indexed newAddress);
  event AddressSet(bytes32 id, address indexed newAddress, bool hasProxy);

  function getMarketId() external view returns (string memory);

  function setMarketId(string calldata marketId) external;

  function setAddress(bytes32 id, address newAddress) external;

  function setAddressAsProxy(bytes32 id, address impl) external;

  function getAddress(bytes32 id) external view returns (address);

  function getLendingPool() external view returns (address);

  function setLendingPoolImpl(address pool) external;

  function getLendingPoolConfigurator() external view returns (address);

  function setLendingPoolConfiguratorImpl(address configurator) external;

  function getLendingPoolCollateralManager() external view returns (address);

  function setLendingPoolCollateralManager(address manager) external;

  function getPoolAdmin() external view returns (address);

  function setPoolAdmin(address admin) external;

  function getEmergencyAdmin() external view returns (address);

  function setEmergencyAdmin(address admin) external;

  function getPriceOracle() external view returns (address);

  function setPriceOracle(address priceOracle) external;

  function getLendingRateOracle() external view returns (address);

  function setLendingRateOracle(address lendingRateOracle) external;
}
```

Add solidity version to config
Yarn hardhat compile // to get abi file

```
async function getLendingPool(account) {
    // abi, address
    const lendingPoolAddressProvider = await ethers.getContractAt(
        "ILendingPoolAddressesProvider",
        "0xB53C1a33016B2DC2fF3653530bfF1848a515c8c5",
        account
    )
    const lendingPoolAddress = await lendingPoolAddressProvider.getLendingPool()
    const lendigPool = await ethers.getContractAt("")
}
```

To get lendinpool contract we need to do same things about interface.

https://docs.aave.com/developers/v/2.0/the-core-protocol/lendingpool/ilendingpool



In Ilendgin Pool there is some imports we actualy dont have in our contracts area,
So we will add aave/protocol-v2 from npm

Npmjs.com/package/@aave/protocol-v2

WE COULD HAVE OPTIONALLY ALSO USED THE INTERFACES FROM THIS PACKAGE

```
yarn add --dev @aave/protocol-v2
```

Update the imports in ILendingPool.sol

```
import {ILendingPoolAddressesProvider} from "@aave/protocol-
v2/contracts/interfaces/ILendingPoolAddressesProvider.sol";
import {DataTypes} from "@aave/protocol-
v2/contracts/protocol/libraries/types/DataTypes.sol";
```

RUN yarn hardhat compile (it must be run without error)

```
async function getLendingPool(account) {
    // abi, address
    const lendingPoolAddressProvider = await ethers.getContractAt(
        "ILendingPoolAddressesProvider",
        "0xB53C1a33016B2DC2fF3653530bfF1848a515c8c5",
        account
    )
    const lendingPoolAddress = await lendingPoolAddressProvider.getLendingPool()
    const lendigPool = await ethers.getContractAt("ILendingPool", lendingPoolAddress, account)
    return lendigPool
}
```

İn main function we can get lending pool address

```
    // Lending Pool Address Provider : 0xB53C1a33016B2DC2fF3653530bfF1848a515c8c5
    const lendingPool = await getLendingPool(deployer)
    console.log(`Lending Pool address : ${lendingPool.address}`)
}
```

Run script

İt will get error, becacuse we have same contract twice

yarn hardhat run scripts/aaveBorrow.js

```
Please replace ILendingPoolAddressesProvider for one of these options wherever you are trying to read its artifact:

@aave/protocol-v2/contracts/interfaces/ILendingPoolAddressesProvider.sol:ILendingPoolAddressesProvider
contracts/interfaces/ILendingPoolAddressesProvider.sol:ILendingPoolAddressesProvider
```

Delete in interfaces which we make it

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$ yarn hardhat run scripts/aaveBorrow.js
yarn run v1.22.18
warning package.json: No license field
$ /home/eemcs/freecodecamp/hardhat-defi-fcc/node_modules/.bin/hardhat run scripts/aaveBorrow.js
Got 20000000000000000 WETH
Lending Pool address : 0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9
Done in 7.57s.
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$
```

This is the actual ethereum address. You can check etherscan.io

Before deposit we have to approve the aave account

Add approve function to scripts/aaveBorrow.js

To approve wee need ERC20 token address and abi. We will user interface for this. Add contracts/interfaces/IERC20.sol

```
pragma solidity ^0.6.6;

interface IERC20 {
    function allowance(address owner, address spender) external view returns (uint256 remaining);
    function approve(address spender, uint256 value) external returns (bool success);
    function balanceOf(address owner) external view returns (uint256 balance);
    function decimals() external view returns (uint8 decimalPlaces);
    function decreaseApproval(address spender, uint256 addedValue) external returns (bool success);
    function increaseApproval(address spender, uint256 subtractedValue) external;
    function name() external view returns (string memory tokenName);
    function symbol() external view returns (string memory tokenSymbol);
    function totalSupply() external view returns (uint256 totalTokensIssued);
    function transfer(address to, uint256 value) external returns (bool success);
    function transferFrom(        address from,        address to,        uint256 value
    ) external returns (bool success);
}
```

Before deposit we have to approve the aave account

Add approve function to scripts/aaveBorrow.js

```
// spenderAddress--> this is going to be the contract that
// weare going to give the approval to, to spend our token
// amount -->amount to spend, how much we want to approve it
async function approveERC20(erc20Address, spenderAddress, amountToSpend, account) {
    const erc20Token = await ethers.getContractAt("IERC20", erc20Address, account)
    const tx = await erc20Token.approve(spenderAddress, amountToSpend)
    await tx.wait(1)
    console.log("APPROVED")
}
```

```
// spenderAddress--> this is going to be the
contract that
// weare going to give the approval to, to
spend our token
// amount -->amount to spend, how much we want
to approve it
```

Run approve function in main function

```
    const wethTokenAddress = "0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2"
    // approve
    await approveERC20(wethTokenAddress, lendingPool.address, AMOUNT, deployer)
    console.log("Depositing..............========")
    // deposit
```

And deposit to aave
To deposit we will use function from lending pool. You can see parameter from the https://docs.aave.com/developers/v/2.0/the-core-protocol/lendingpool methods

```
    await approveERC20(wethTokenAddress, lendingPool.address, AMOUNT, deployer)
    console.log("Depositing..............========")
    // deposit
    await lendingPool.deposit(wethTokenAddress, AMOUNT, deployer, 0)
    console.log("DEPOSİTED==============")
}
```

RUN THE SCRIPT

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$ yarn hardhat run scripts/aaveBorrow.js
yarn run v1.22.18
warning package.json: No license field
$ /home/eemcs/freecodecamp/hardhat-defi-fcc/node_modules/.bin/hardhat run scripts/aaveBorrow.js
contracts/interfaces/IERC20.sol: Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comm
ent containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-sou
rce code. Please see https://spdx.org for more information.

Compiled 1 Solidity file successfully
Got 20000000000000000 WETH
Lending Pool address : 0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9
APPROVED
Depositing.............========
DEPOSITED==============
Done in 21.05s.
```

Borrowing from Aave 19.57.40

```
 // BORROW TIME - borrow other assets with
deposited weth
    // We want to know how much we can borrow.
    // how much we have in collateral
    // how much we have borrowed
```

https://docs.aave.com/risk/asset-risk/risk-parameters

If we have one eth and collateral that doesnt mean we can borrow one eth of assets. Each one of these tokens (https://docs.aave.com/risk/asset-risk/risk-parameters) have some different values like loan to value.

For ex. İf you have one eth you can borrow 0.75 for DAI token. This is reduce risk of the colleteral and reduce risk of people

Name	Symbol	Collateral	Loan To Value	Liquidation Threshold
Stablecoins				
Ampleforth	AMPL	no		
Binance USD	BUSD	no	-	-
DAI	DAI	yes	77%	80%
Fei	FEI	yes	65%	75%
Frax	FRAX	no		

Liquidation: https://docs.aave.com/developers/v/2.0/guides/liquidations
When you put down colleteral and you borrow, if the amount that you have borrowed past this liquidation thershold is passed that 80 %, or depending on different assets, its different people can do whats called liquidate you.This is when they pay back some of your loan that you took out. And they also get to buy some of your collateral at a cheaper prive. This keeps the Aave platform solvetn, and it makes it so that theres never more borrows than there are collateral in order to borrow assets, we still need that collateral down.

Basically if you borrowed more Money than you have put up, other users can take the Money that you are put up in return for them paying for your loans. So we abviously dont want this to happen. And the audit protocol programmatically does not want to have not enough Money to do this. So they incentivize users to liquidate, in case of these failures is the protocols come with this thing called a health factor, which if this health factor is below one, you go ahead and you get liquidated.

The actual function to liquidate somebody is called liquidationCall(). Sou you can actually build a bot and you can liquidate users who go insolvent and you can make a fee, you can make a reward for actually doin this.

These protocols need to stay solvent, they need to have enough Money to lend out. And they programmatically enforced this, which is why its so great.

https://docs.aave.com/developers/v/2.0/the-core-protocol/lendingpool

İf our health factor ever falls below one, we get liquidated. So we never want this health factor to fall below one when were borrowing assets.

To get user data / aaveBorrow.js

```
async function getBorrowUserData(lendigPool, account) {
    const { totalCollateralETH, totalDebtETH, avaliableBorrowETH } =
        await lendigPool.getUserAccountData(account)
    console.log(`You have ${totalCollateralETH} worth of ETH deposited.`)
    console.log(`You have ${totalDebtETH} worth of ETH borrowed.`)
    console.log(`You can borrow ${avaliableBorrowETH} worth of ETH.`)
    return {avaliableBorrowETH,totalDebtETH}
}
```

Add user data to main function.

```
    // BORROW TIME - borrow other assets with deposited weth
    // We want to know how much we can borrow.
    // how much we have in collateral
    // how much we have borrowed
    let { avaliableBorrowETH, totalDebtETH } = await getBorrowUserData(lendingPool,
deployer)
}
```

With running the script we will see how much we can barrow. It work on our forked blockchain
It will be slow because it does have to make an api calls whenever we want to interact with these chains.

```
Hh run scripts/aaveBorrow.js
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$ hh run scripts/aaveBorrow.js
Got 4000000000000000 WETH
Lending Pool address : 0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9
APPROVED
Depositing..............========
DEPOSİTED==============
You have 4000000000000000 worth of ETH deposited.
You have 0 worth of ETH borrowed.
You can borrow 3300000000000000 worth of ETH.
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$
```

Now no borrowed eth
Borrow amount always less then deposit

Now we can borrow. What the conversion rate of DAI is? We must learn. For this we use Price Oracle

https://docs.aave.com/developers/v/2.0/the-core-protocol/price-oracle

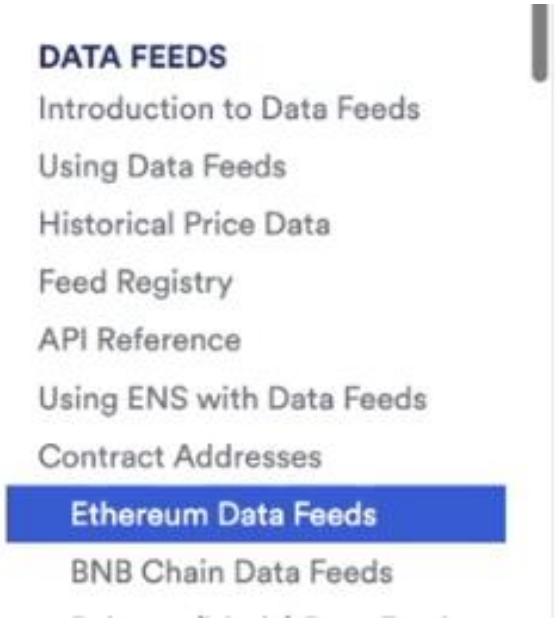
And we use aggregator.

Copy aggregator contract to Project. We can import from chainlin NPM

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

interface AggregatorV3Interface {
    function decimals() external view returns (uint8);

    function description() external view returns (string memory);

    function version() external view returns (uint256);

    // getRoundData and latestRoundData should both raise "No data present"
    // if they do not have data to report, instead of returning unset values
    // which could be misinterpreted as actual reported values.
    function getRoundData(uint80 _roundId)
        external
        view
        returns (
            uint80 roundId,
            int256 answer,
            uint256 startedAt,
            uint256 updatedAt,
            uint80 answeredInRound
        );

    function latestRoundData()
        external
        view
        returns (
            uint80 roundId,
            int256 answer,
            uint256 startedAt,
            uint256 updatedAt,
            uint80 answeredInRound
        );
}
```

```
async function getDAIPrice() {
    const daiEthPriceFeed = await ethers.getContractAt(
        "AggregatorV3Interface",
        "0x773616E4d11A78F511299002da57A0a94577F1f4" // DAI/ETH price addres
    )
    const price = (await daiEthPriceFeed.latestRoundData())[1]
    console.log(`The DAI/ETH price is ${price.toString()}`)
    return price
}
```

Dai/eth address → https://docs.chain.link/docs/etherum-addresses/



>Ethereum ADA PoR	Cardano		
CelsiusX Dogecoin->Ethereum DOGE PoR		-	0xe6D28A56E6bD1C123c8210f9A9c95bb6e107A1ef
DAI / ETH	DAI	Crypto	0x773616E4d11A78F511299002da57A0a94577F1f4
DAI / USD	DAI	Crypto	0xAed0c38402a5d19df6E4c03F4E2DceD6e29c1ee9
DASH / USD	Dash	Crypto	0xFb0cADFEa136E9E343cfb55B863a6Df8348ab912
DATA / ETH	Data Economy Index (DATA)	Crypto	0xD48B96131F3de05B7C3500891C8c4c1E2dbc6E3d

```
const price = (await daiEthPriceFeed.latestRoundData())[1]
```

```
The index 1 is from contracts/interfaces/AggregatorV3Interface.sol contract
getRoundData() function's second return value "answer"
```

```
function getRoundData(uint80 _roundId)
        external
        view
        returns (
            uint80 roundId,
            int256 answer,
            uint256 startedAt,
            uint256 updatedAt,
            uint80 answeredInRound
        );
```

Add function to main function and run script

```
    let { availableBorrowsETH, totalDebtETH } = await getBorrowUserData(lendingPool,
deployer)
    const daiPrice = await getDAIPrice()
}
```

hh run scripts/aaveBorrow.js

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$ hh run scripts/aaveBorrow.js
Got 2000000000000000 WETH
Lending Pool address : 0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9
APPROVED
Depositing.............========
DEPOSİTED==============
You have 2000000000000000 worth of ETH deposited.
You have 0 worth of ETH borrowed.
You can borrow 1650000000000000 worth of ETH.
The DAI/ETH price is 9419127713885678
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$
```

Now, figüre out the amount that we can borrow DAI

```
const daiPrice = await getDAIPrice()
    const amountDaiToBorrow = availableBorrowsETH.toString() * 0.95 * (1 /
daiPrice.toNumber())
    console.log(`You can borrow ${amountDaiToBorrow} DAI.`)
```

```
console.log(`You can borrow ${amountDaiToBorrow} DAI.`)
    const amountDaiToBorrowWei = ethers.utils.parseEther(amountDaiToBorrow.toString())
```

Now we can write actual borrow function.

```
async function borrowDAI(daiAddress, lendigPool, amountDaiToBorrowWei, account) {
    const borrowTx = await lendigPool.borrow(daiAddress, amountDaiToBorrowWei, 1, 0,
account)
    await borrowTx.wait(1)
    console.log(`You are borrowed !`)
}
```

Add to main function

```
    const daiTokenAddress = "0x6B175474E89094C44Da98b954EedeAC495271d0F"
    await borrowDAI(daiTokenAddress, lendingPool, amountDaiToBorrowWei,deployer)
    await getBorrowUserData(lendingPool,deployer)
```

DAI token address from mainnet
https://etherscan.io/token/0x6b175474e89094c44da98b954eedeac495271d0f

hh run scripts/aaveBorrow.js

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$ hh run scripts/aaveBorrow.js
Got 20000000000000000 WETH
Lending Pool address : 0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9
APPROVED
Depositing.............========
DEPOSITED==============
You have 20000000000000000 worth of ETH deposited.
You have 0 worth of ETH borrowed.
You can borrow 1050000000000000000 worth of ETH.
The DAI/ETH price is 941912713885678
You can borrow 16.641669412589017 DAI.
You are borrowed !
You have 20000000060267315 worth of ETH deposited.
You have 15674999999999997 worth of ETH borrowed.
You can borrow 8250000049720538 worth of ETH.
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$
```

Repaying with Aave 20.12.01

```
async function repay(amount, daiAddress, lendigPool, account) {
    await approveERC20(daiAddress, lendigPool.address, amount, account)
    const repayTx = await lendigPool.repay(daiAddress, amount, 1, account)
    await repayTx.wait(1)
    console.log(`Repaid`)
}
```

Main function

```
    await getBorrowUserData(lendingPool, deployer)

    await repay(amountDaiToBorrowWei, daiTokenAddress, lendingPool, deployer)
    console.log("After repaid====================")
    getBorrowUserData(lendingPool, deployer)
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$ hh run scripts/aaveBorrow.js
Got 20000000000000000 WETH
Lending Pool address : 0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9
APPROVED
Depositing.............========
DEPOSITED==============
You have 20000000000000000 worth of ETH deposited.
You have 0 worth of ETH borrowed.
You can borrow 16500000000000000 worth of ETH.
The DAI/ETH price is 941912713885678
You can borrow 16.641669412589017 DAI.
You are borrowed !
After borrowed=========================
You have 20000000056351270 worth of ETH deposited.
You have 15674999999999997 worth of ETH borrowed.
You can borrow 825000046489801 worth of ETH.
APPROVED
Repaid
After repaid===================
You have 20000000108677450 worth of ETH deposited.
You have 714613155 worth of ETH borrowed.
You can borrow 16499999375045741 worth of ETH.
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-defi-fcc$
```

After paying we are still going to have a DAI balance, and a little bit of Weth borrowed. Because as we borrow dye, we actually accured interest. We still owe dai back.

Visualizing the Transactions (And aTokens) 20.14.58

Listen here again!!!!!!!!!!!!!!!!!!!!!!!!!!!!

endddddddddddddddddddddddddddddd