

Hardhat

Récupérer des éther de test

- <https://sepoliafaucet.com/> créer un compte sur alchemy qui permettra d'avoir un rpc sur le réseau de test de spolia. (0,5eth)
- <https://sepolia-faucet.pk910.de/> miner des ether sur le réseau spolia
- <https://faucet.sepolia.dev/>

Hardhat un framework?

- Hardhat est un des deux framework les plus connus. (Avec truffle)
- Il offre une blockchain de test
- La création d'un environnement complet pour démarrer
- Offre une interface de testing
- Permet d'automatiser certaines taches.

Démarrer un projet hardhat

- ❖ Vous devez avoir nodejs (v16 mini) d'installé ainsi que npm
- ❖ Installer hardhat : `npm install --save-dev hardhat`
- ❖ Npx hardhat démarrera un menu
- ❖ Choisir Create a JavaScript project
- ❖

La structure des dossiers et fichiers de config

- Contracts : contiendra vos contrats
- Scripts contiendra vos scripts déploiement compris
- Test vos tests unitaires
- hardhat.config.js contiendra les informations de compilation & de déploiement (rpc des blockchains ou déployer)

Configurer hardhat.config.js

```
require("@nomicfoundation/hardhat-toolbox");
require("dotenv/config")
require("@nomiclabs/hardhat-etherscan")

/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  networks: {
    // for mainnet
    'optimism': {
      url: "https://mainnet.optimism.io",
      accounts: [process.env.pvtKey]
    },
    // for testnet
    'optimisticGoerli': {
      url: "https://goerli.optimism.io",
      accounts: [process.env.pvtKey]
    },
    // for the local dev environment
    'optimism-local': {
      url: "http://localhost:8545",
      accounts: [process.env.pvtKey]
    },
    etherscan: {
      apiKey: [process.env.apiKey],
    },
    solidity: {
      compilers: [
        {version: "0.5.16"}, 
        {version: "0.8.4"}, 
        {version: "0.6.6"}, 
        settings: {
          optimizer: {
            enabled: true,
            runs: 1000
          }
        }
      ]
    }
  };
}
```

Créer un fichier de déploiement.

```
// We require the Hardhat Runtime Environment explicitly here. This is optional.  
// but useful for running the script in a standalone fashion through `node <script>`.  
//  
// You can also run a script with `npx hardhat run <script>` . If you do that, Hardhat  
// will compile your contracts, add the Hardhat Runtime Environment's members to the  
// global scope, and execute the script.  
const hre = require("hardhat");  
  
async function main() {  
  
    const PancakeRouter = await hre.ethers.getContractFactory("PancakeRouter");  
    const pancakeRouter = await PancakeRouter.deploy("0xb7247eFb2c2f88B8829343799Cbaa735278B5e0f", "0x4200000000000000000000000000000000000000000000000000000000000006");  
    await pancakeRouter.deployed();  
    console.log(`  
        deployed to ${pancakeRouter.address}`  
    );  
}  
  
// We recommend this pattern to be able to use async/await everywhere  
// and properly handle errors.  
main().catch((error) => {  
    console.error(error);  
    process.exitCode = 1;  
});
```

Compiler avec hardhat

- Npx hardhat compile :)
-

Deployer avec hardhat

```
npx hardhat run --network localhost scripts/deploy.js (pour la blockchain de test hardhat)  
npx hardhat run --network <your-network> scripts/deploy.js (network tel que défini dans le fichier  
hardhat.config.js)
```

Deployer hello.sol

- Copier le contenu de hello.sol dans le dossier contacts et l'appeler hello.sol
- Voir au tableau ou sur le GitHub pour fichier deployHello.js
- Deployer avec la commande : npx hardhat run --network sepolia scripts/deployHello.js
- Récupérer l'adresse et voir sur etherscan de sepolia ce qu'il en ressort
- Pour que le contrat soit lisible ainsi que les fonctions il doit être vérifié :
-

Verifier son contrat sur etherscan ou autres.

- ❖ Il vous faudra une clé API d'etherscan (il faudra un compte)

- ❖ Définir dans le fichier hardhat.config.js la valeur de la clé api :

```
etherscan: {  
    apiKey: 'votreCléApiEtherscan',  
},
```

- ❖ Executer la commande : npx hardhat verify --network sepolia <address>

❖

Deploiement et verif contrat multiples arguments

- Utiliser le contrat ERC20.sol le mettre dans le dossier contrats
- Dans le dossier script copier coller le deploy.js en deployErc20.js (voir les paramètres à rentrer dans le fichier)
- Importer les contrats standards open zeppelin : npm i @openzeppelin/contracts
- npx hardhat verify --network sepolia --contract contracts/ SimpleToken.sol:SimpleToken adresseDuContrat "token" "tkn" "20000000000000000000000000"