Tech Stack Lab

Dr. Kurz

Ronan Kearns

Jeff Turner

CHAPMAN UNIVERSITY

Review Assignment 2

https://github.com/kearns-cu/Blockchain_Course_Repo/tree/main/Assignments/Solutions

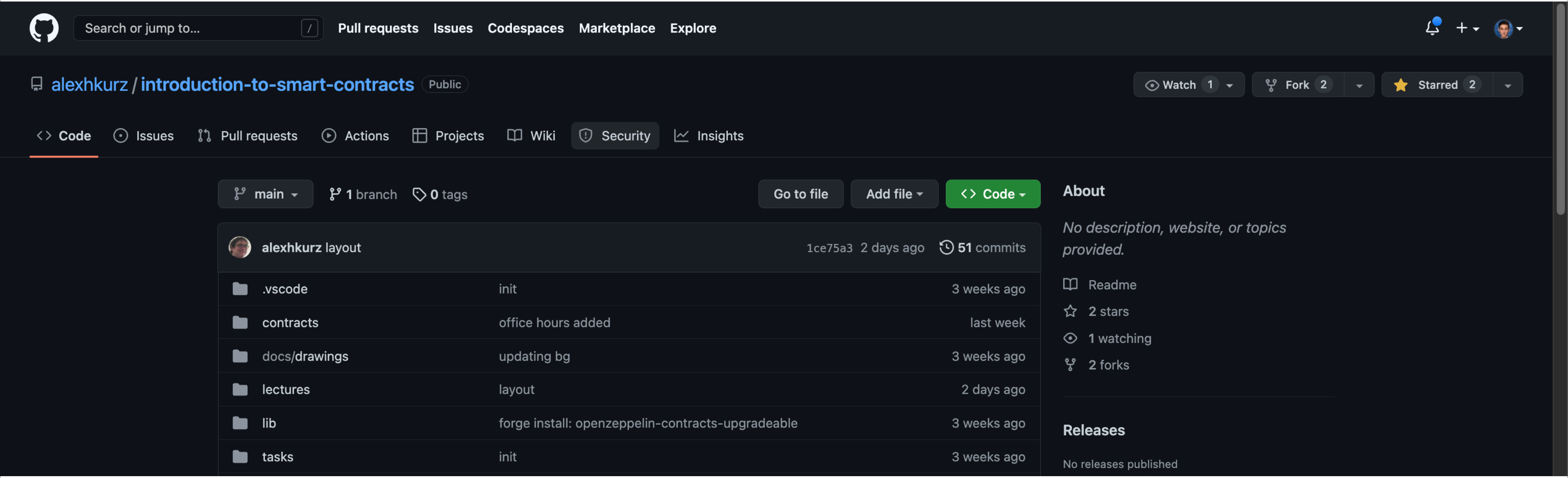
CHAPMAN UNIVERSITY

Requirements

- Github Account
- VS Code
- Solidity Extension – Download in Class
- Node
    - Homebrew for macOS and chocolatey for Windows or install via browser

CHAPMAN UNIVERSITY

Forking Repo – In Class



Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

alexhkurz / introduction-to-smart-contracts Public

Watch 1 Fork 2 Starred 2

Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags Go to file Add file Code

alexhkurz layout 1ce75a3 2 days ago 51 commits

.vscode	init	3 weeks ago
contracts	office hours added	last week
docs/drawings	updating bg	3 weeks ago
lectures	layout	2 days ago
lib	forge install: openzeppelin-contracts-upgradeable	3 weeks ago
tasks	init	3 weeks ago

About

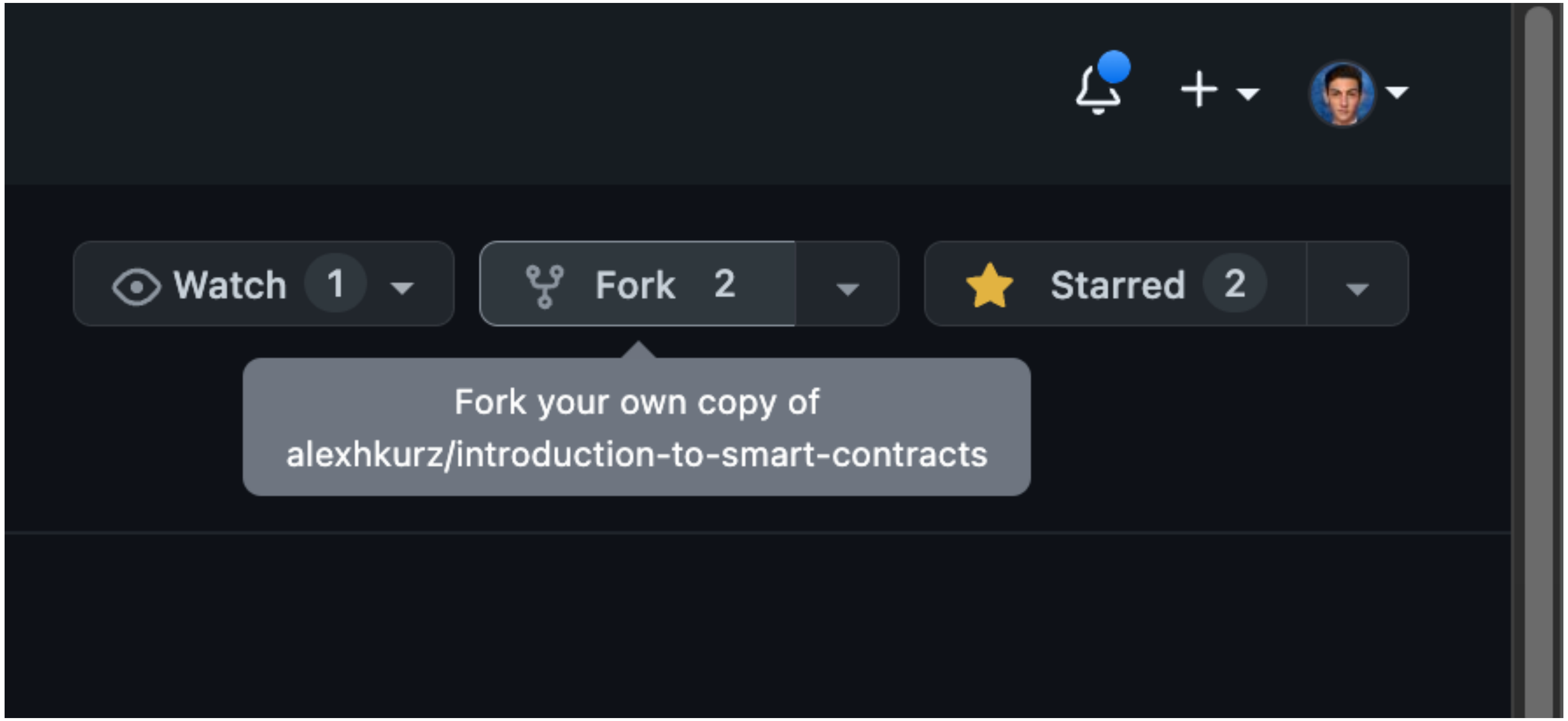
No description, website, or topics provided.

Readme
2 stars
1 watching
2 forks

Releases

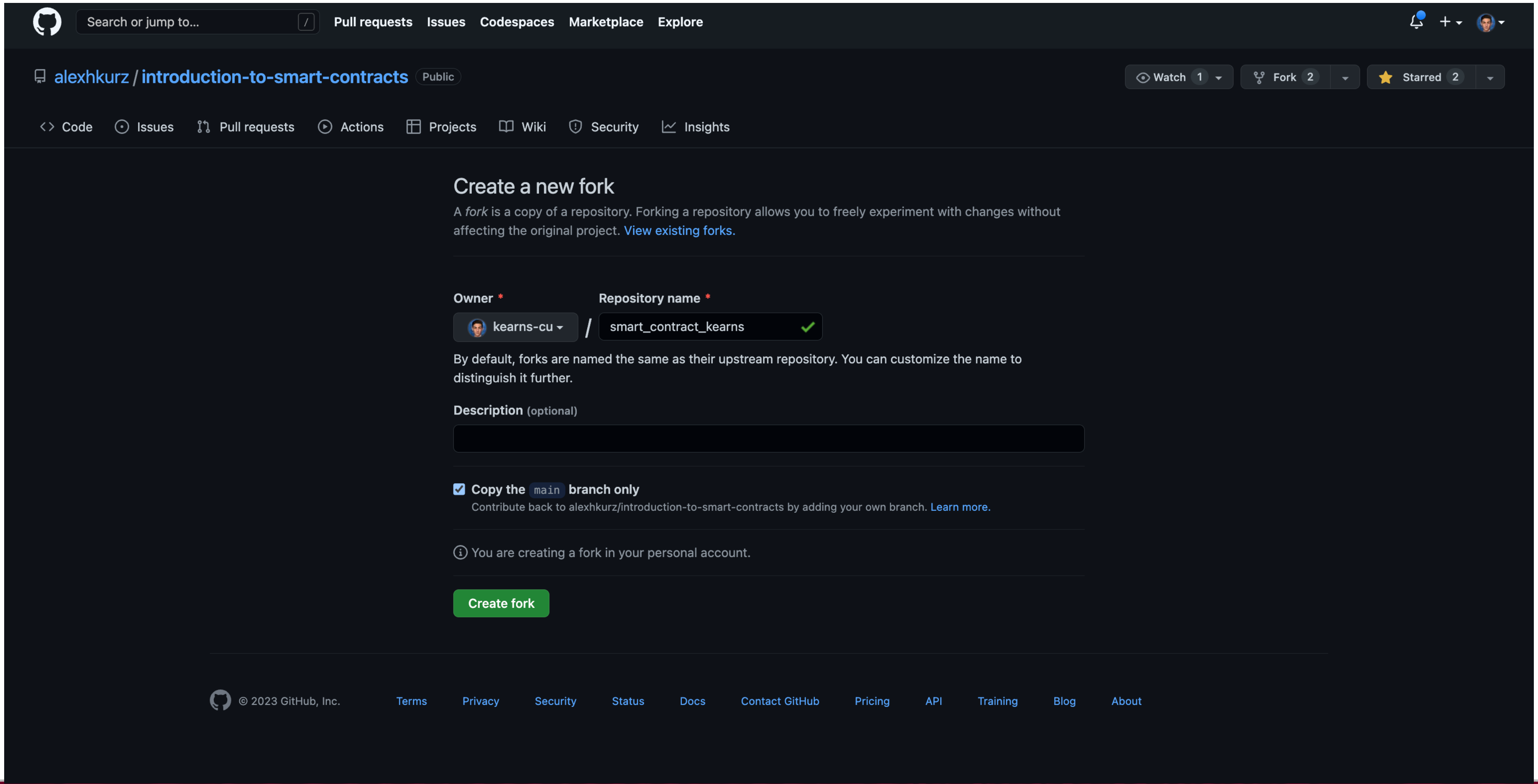
No releases published

Forking Repo – In Class



Fork your own copy of alexhkurz/introduction-to-smart-contracts

Forking Repo – In Class



Search or jump to...

Pull requests Issues Codespaces Marketplace Explore

alexhkurz / introduction-to-smart-contracts Public

Watch 1 Fork 2 Starred 2

Code Issues Pull requests Actions Projects Wiki Security Insights

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. View existing forks.

Owner * **Repository name** *

kearns-cu / smart_contract_kearns

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

☑ **Copy the `main` branch only**
Contribute back to alexhkurz/introduction-to-smart-contracts by adding your own branch. Learn more.

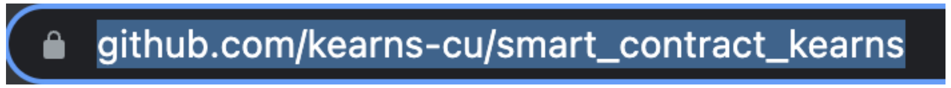
You are creating a fork in your personal account.

Create fork

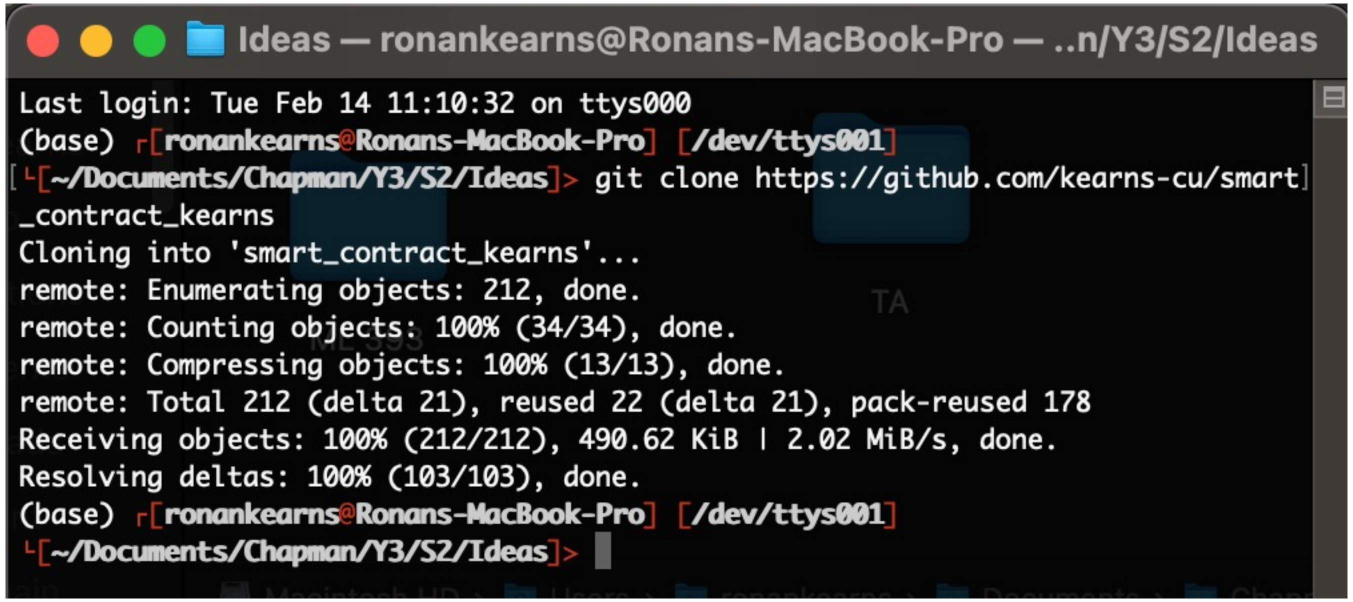
© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

CHAPMAN UNIVERSITY

Cloning Repo to System – In Class



github.com/kearns-cu/smart_contract_kearns

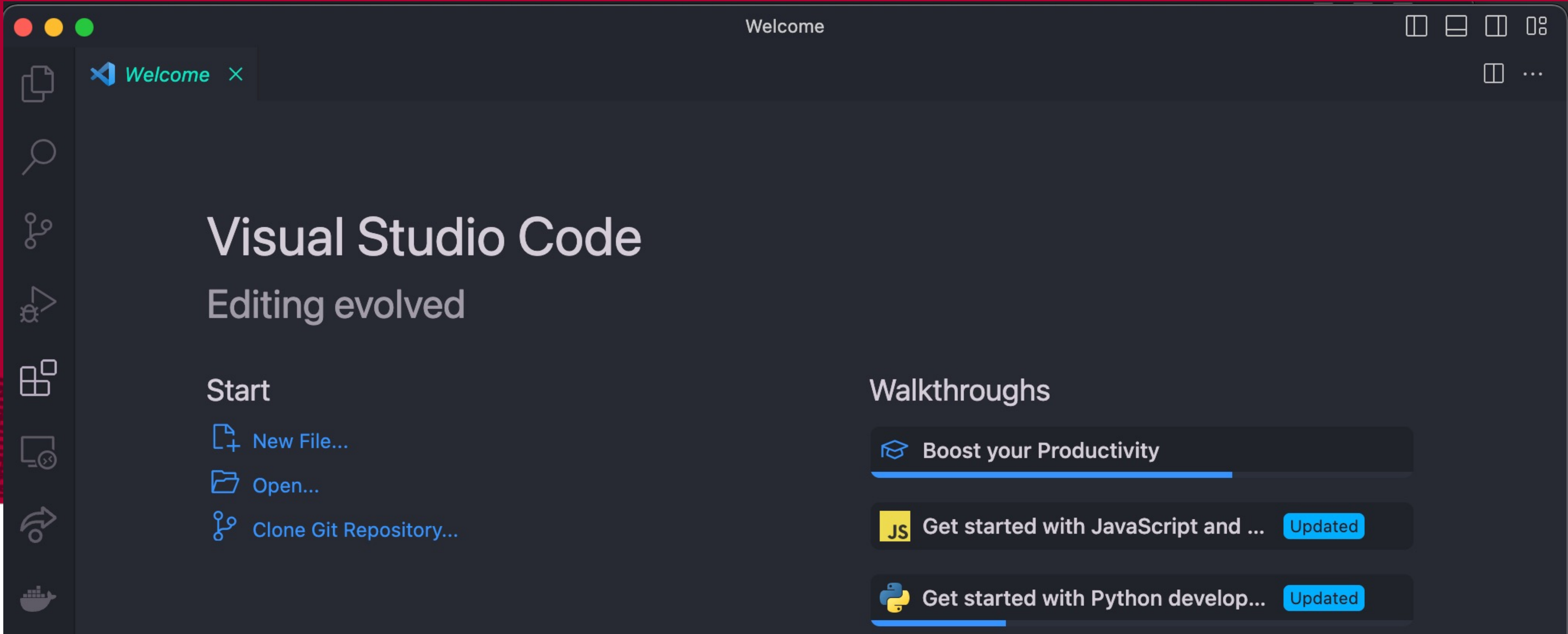


```
Last login: Tue Feb 14 11:10:32 on ttys000
(base) ┌[ronankearns@Ronans-MacBook-Pro] [/dev/ttys001]
└[~/Documents/Chapman/Y3/S2/Ideas]> git clone https://github.com/kearns-cu/smart
_contract_kearns
Cloning into 'smart_contract_kearns'...
remote: Enumerating objects: 212, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 212 (delta 21), reused 22 (delta 21), pack-reused 178
Receiving objects: 100% (212/212), 490.62 KiB | 2.02 MiB/s, done.
Resolving deltas: 100% (103/103), done.
(base) ┌[ronankearns@Ronans-MacBook-Pro] [/dev/ttys001]
└[~/Documents/Chapman/Y3/S2/Ideas]>
```

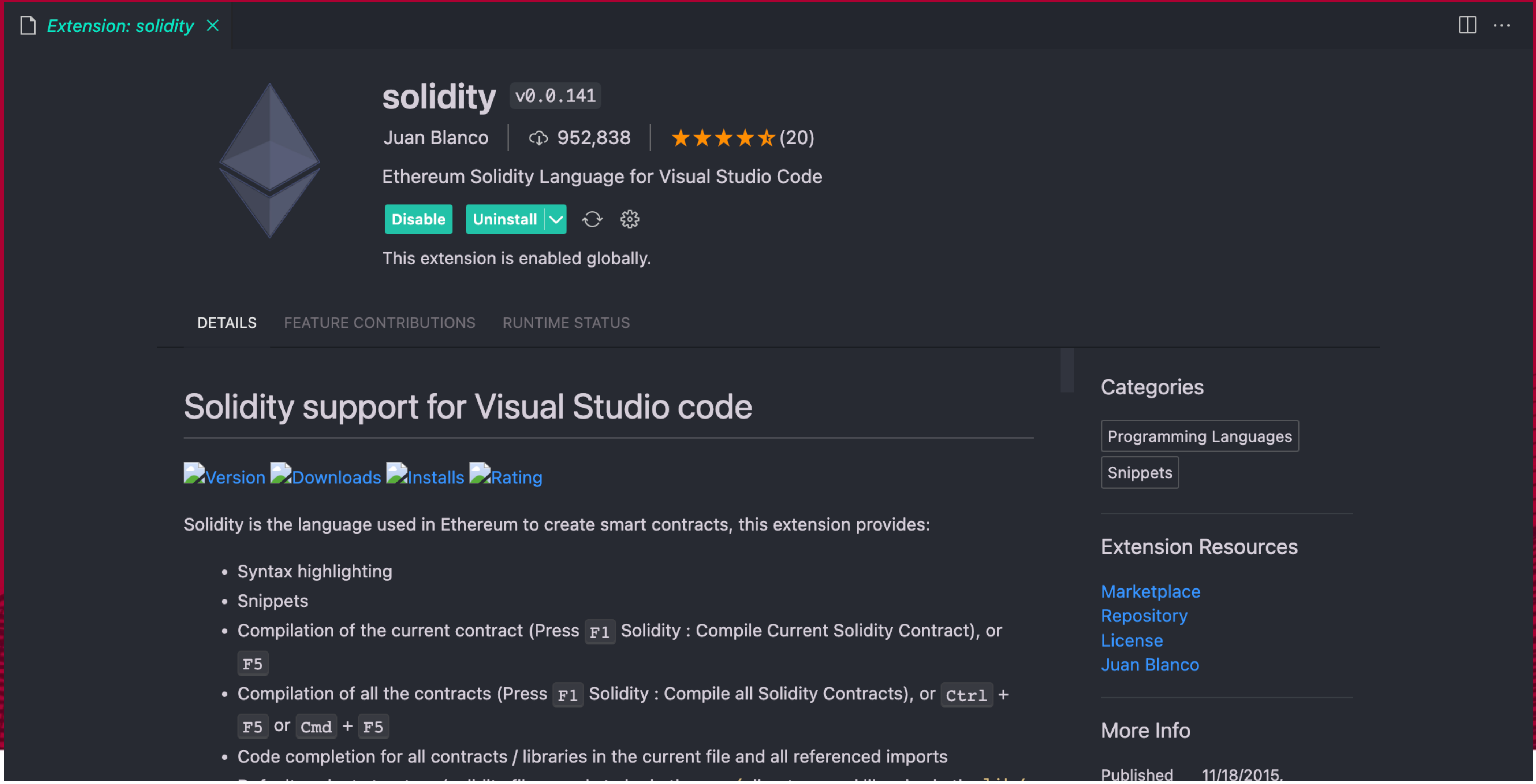
- git clone *your personal repository url*
- Remove unused files in contracts and test directories if need

Visual Studio Code (VSCode)

- https://code.visualstudio.com/download
- Use this link to download the app



Visual Studio Code (VSCode)



Extension: solidity

solidity v0.0.141

Juan Blanco | 952,838 | ★★★★☆ (20)

Ethereum Solidity Language for Visual Studio Code

This extension is enabled globally.

Solidity support for Visual Studio code

Version Downloads Installs Rating

Solidity is the language used in Ethereum to create smart contracts, this extension provides:

- Syntax highlighting
- Snippets
- Compilation of the current contract (Press `F1` Solidity : Compile Current Solidity Contract), or `F5`
- Compilation of all the contracts (Press `F1` Solidity : Compile all Solidity Contracts), or `Ctrl` + `F5` or `Cmd` + `F5`
- Code completion for all contracts / libraries in the current file and all referenced imports

Categories

Programming Languages

Snippets

Extension Resources

Marketplace

Repository

License

Juan Blanco

More Info

Published 11/18/2015,

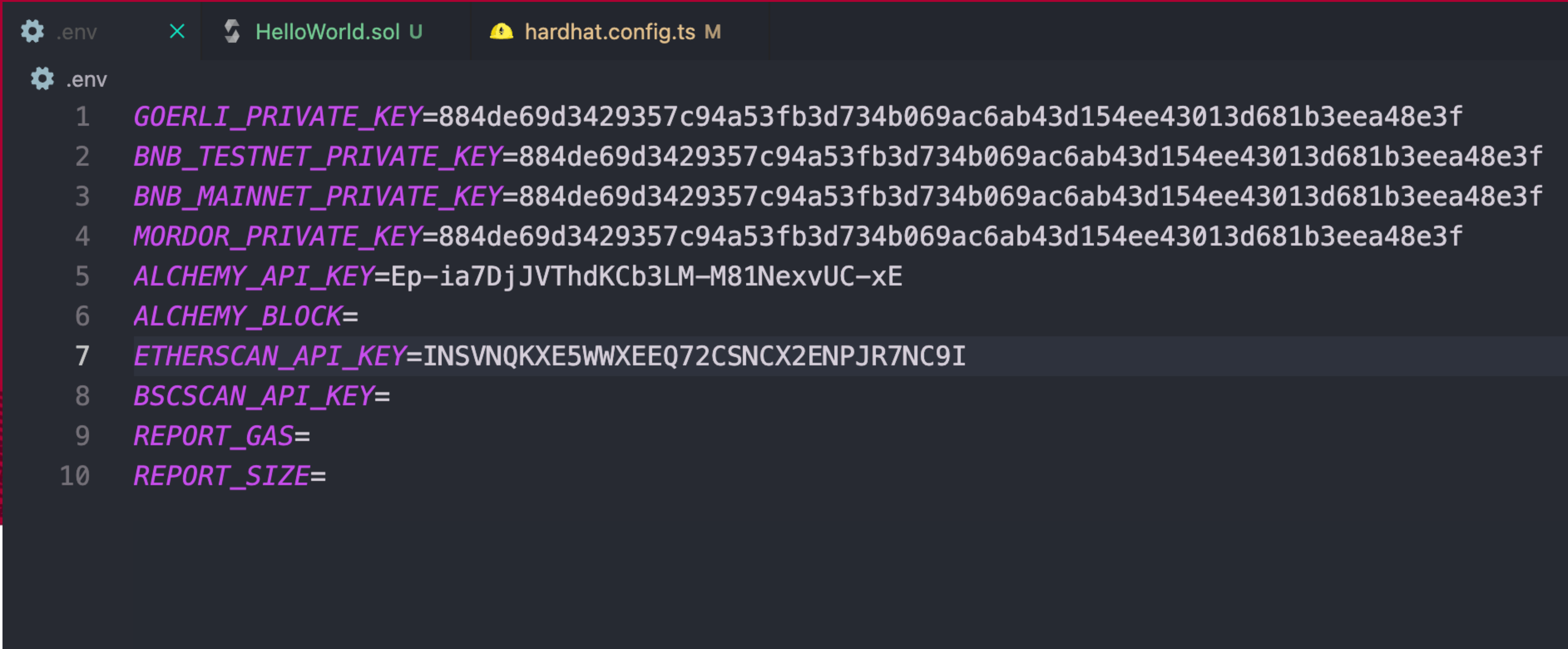
CHAPMAN UNIVERSITY

Follow Github README

- Install Foundry - > https://book.getfoundry.sh/getting-started/installation.html
- npm i
- forge install

Populate ENV File

- Export private keys from your Metamask
- Etherscan API is used for verifying contracts via command line



```
GOERLI_PRIVATE_KEY=884de69d3429357c94a53fb3d734b069ac6ab43d154ee43013d681b3eea48e3f
BNB_TESTNET_PRIVATE_KEY=884de69d3429357c94a53fb3d734b069ac6ab43d154ee43013d681b3eea48e3f
BNB_MAINNET_PRIVATE_KEY=884de69d3429357c94a53fb3d734b069ac6ab43d154ee43013d681b3eea48e3f
MORDOR_PRIVATE_KEY=884de69d3429357c94a53fb3d734b069ac6ab43d154ee43013d681b3eea48e3f
ALCHEMY_API_KEY=Ep-ia7DjJVThdKCb3LM-M81NexvUC-xE
ALCHEMY_BLOCK=
ETHERSCAN_API_KEY=INSVNQKXE5WWXEEQ72CSNCX2ENPJR7NC9I
BSCSCAN_API_KEY=
REPORT_GAS=
REPORT_SIZE=
```

Don’t Forget to Use Faucet for test ETH

https://goerlifaucet.com/

- Sign Up for Alchemy account and use account to sign into Goerli faucet
  - https://www.alchemy.com/

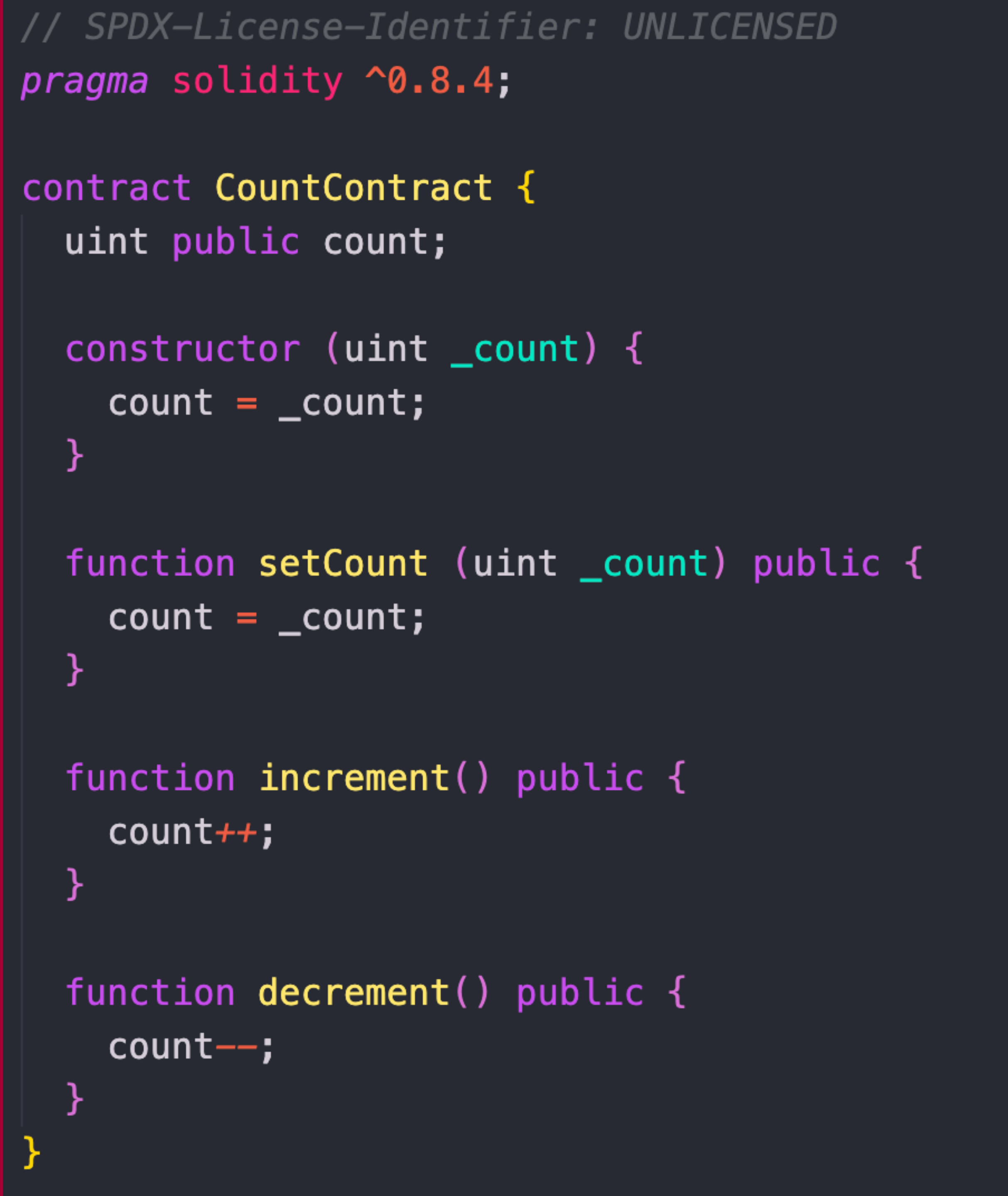
Contract Structure

Solidity Documentation

- https://docs.soliditylang.org/en/v0.8.18/structure-of-a-contract.html

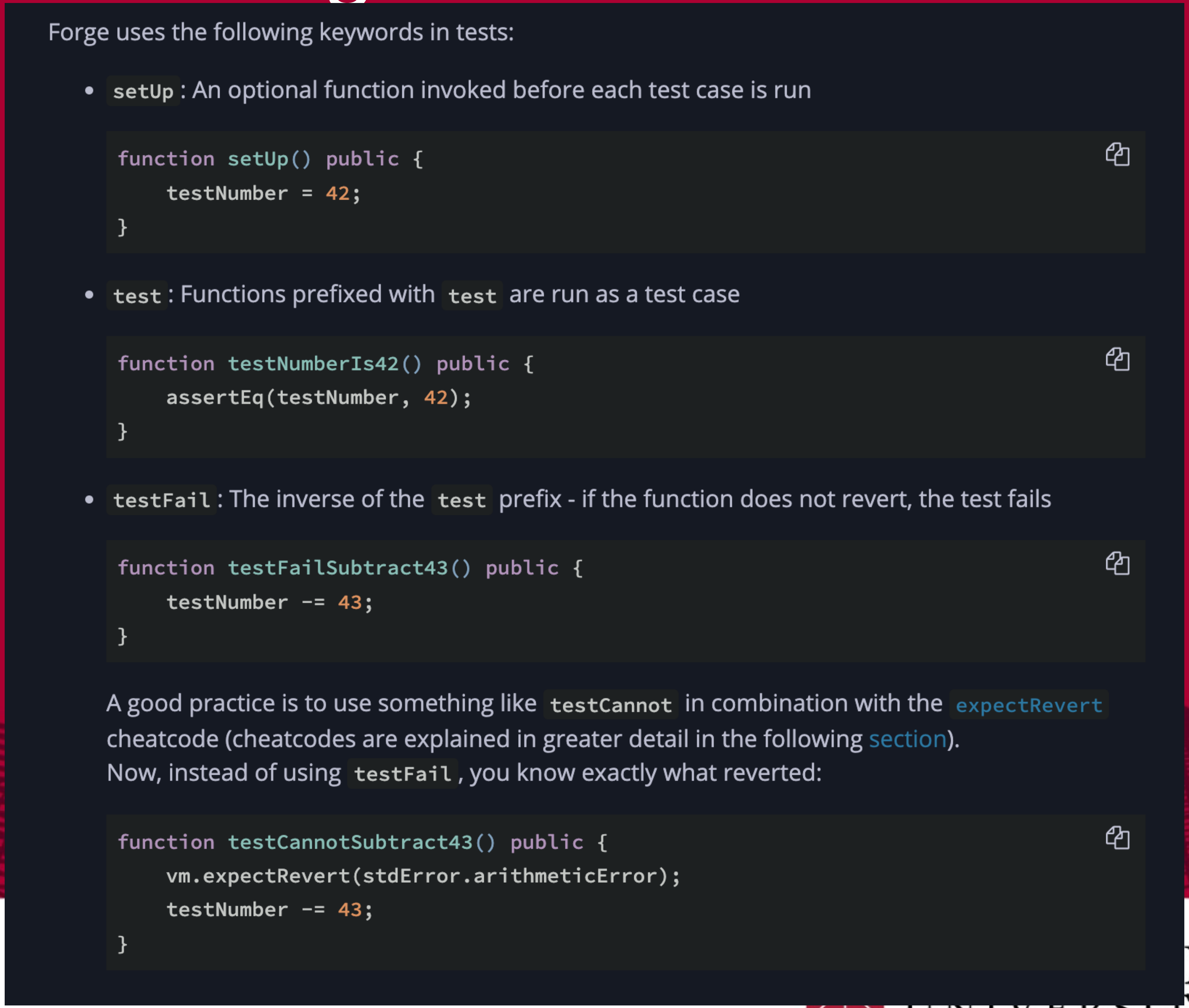
Cheatsheet

- https://docs.soliditylang.org/en/v0.8.18/cheatsheet.html



```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.4;

contract CountContract {
  uint public count;

  constructor (uint _count) {
    count = _count;
  }

  function setCount (uint _count) public {
    count = _count;
  }

  function increment() public {
    count++;
  }

  function decrement() public {
    count--;
  }
}
```

Writing Unit Tests



Forge uses the following keywords in tests:

- `setUp`: An optional function invoked before each test case is run

```
function setUp() public {
    testNumber = 42;
}
```

- `test`: Functions prefixed with `test` are run as a test case

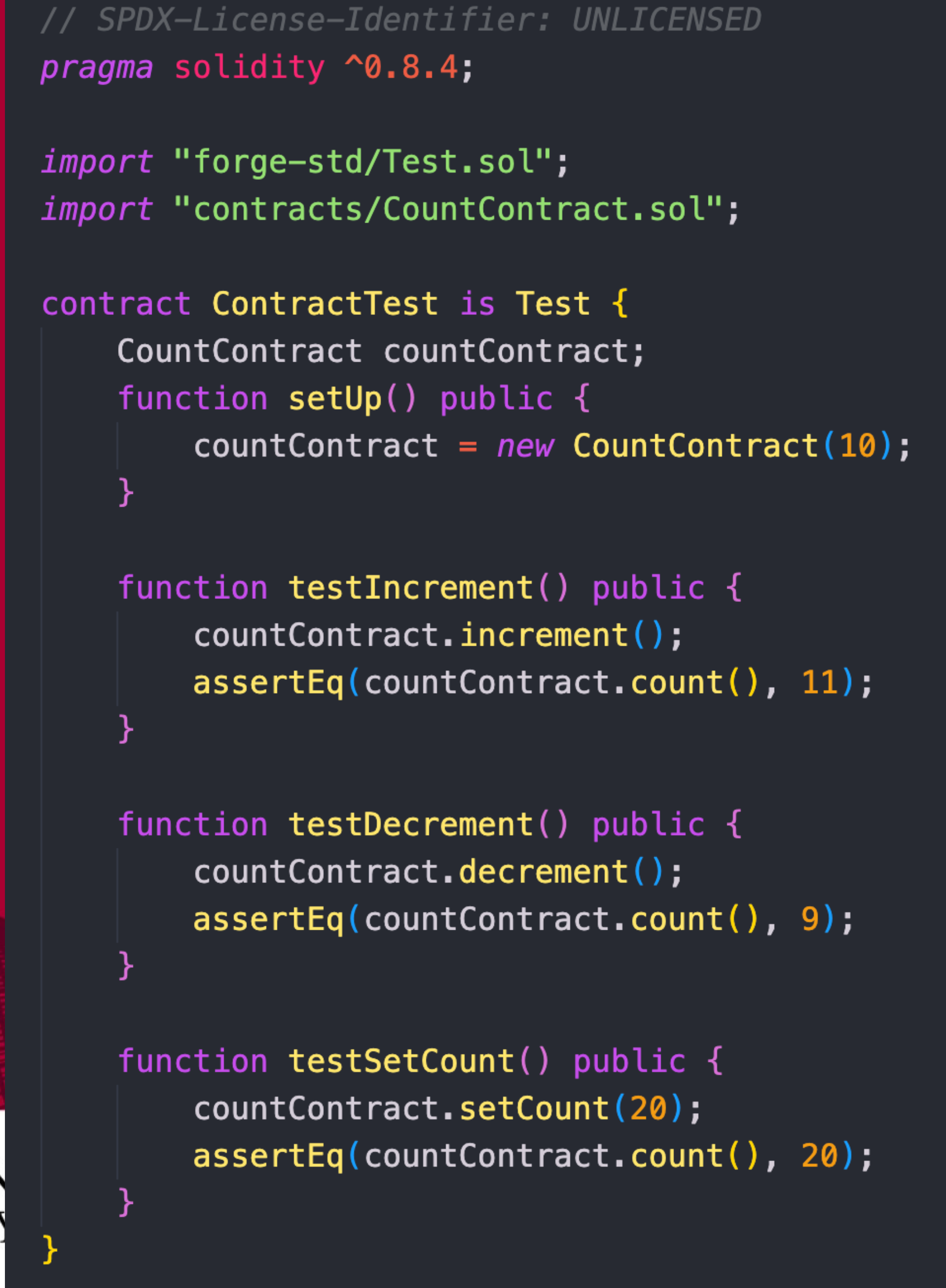
```
function testNumberIs42() public {
    assertEq(testNumber, 42);
}
```

- `testFail`: The inverse of the `test` prefix - if the function does not revert, the test fails

```
function testFailSubtract43() public {
    testNumber -= 43;
}
```

A good practice is to use something like `testCannot` in combination with the `expectRevert` cheatcode (cheatcodes are explained in greater detail in the following section). Now, instead of using `testFail`, you know exactly what reverted:

```
function testCannotSubtract43() public {
    vm.expectRevert(stdError.arithmeticError);
    testNumber -= 43;
}
```



```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.4;

import "forge-std/Test.sol";
import "contracts/CountContract.sol";

contract ContractTest is Test {
    CountContract countContract;
    function setUp() public {
        countContract = new CountContract(10);
    }

    function testIncrement() public {
        countContract.increment();
        assertEq(countContract.count(), 11);
    }

    function testDecrement() public {
        countContract.decrement();
        assertEq(countContract.count(), 9);
    }

    function testSetCount() public {
        countContract.setCount(20);
        assertEq(countContract.count(), 20);
    }
}
```

Assignment 3

- Write a getCount() function for the CountContract.sol
- Alter the CountContract.t.sol to add a test for your getCount function you added
- Compile your contract using npm run compile
- Test your contract with npm run test
- Once you have finished development and testing your solution, you will deploy it to the Goerli chain
  - npx hardhat --network goerli deploy --contract CountContract
    - Make sure contract name is the same name defined inside of your contract
  - Ensure your .env file is populated
  - Verify smart contract with the command:
    - npx hardhat --network goerli verify <deployed_contract_address>

CHAPMAN UNIVERSITY

CHAPMAN
UNIVERSITY