**Introduction to smart contracts testing**

**Analyzing the Counter contracts**

Continuing from the previous lesson, the forge init populated our project with the Counter files.

**Counter.sol**

It's a simple smart contract that stores a number. You have a function to setNumber where you specify a newNumber which is a uint256, and store it, and you have a function to increment the number.

**Note:** number++ is equivalent to number = number + 1.

**Counter.s.sol**

Just a placeholder, it doesn't do anything

**Counter.t.sol**

This is the interesting part. We haven't talked that much about carrying tests using Foundry. This is an essential step for any project. The test folder will become our new home throughout this course.

Please run the following command in your terminal:

forge test

After the contracts are compiled you will see an output related to tests:

* How many tests were found;
* In which file;
* Did they pass or not?;
* Summary;

**How does forge test work?**

forge test has a lot of options that allow you to configure what is tested, how the results are displayed, where is the test conducted and many more!

Run forge test --help to explore the options. I suggest reading [this page](https://book.getfoundry.sh/forge/tests) and navigating deeper into the Foundry Book to discover how tests work.

But in short, in our specific case:

1. Forge identified all the files in the test folder, went into the only file available and ran the setUp function.
2. After the setup is performed it goes from top to bottom in search of public/external functions that start with test.
3. All of them will be called and the conclusion of their execution will be displayed. By that we mean it will run all the assert statements it can find and if all evaluate to true then the test will pass. If one of the assert statements evaluates to false the test will fail.