**Interact with a smart contract using the CLI**

**Interacting With Contract Addresses via Command Line & Foundry's Cast Tool**

This lesson builds on top of previous lessons where we deployed SimpleStorage via forge script. We have Anvil running and the smart contract is deployed.

Copy the contract address.

**Sending information to the blockchain**

Foundry has a built-in tool known as Cast. Cast comes loaded with numerous commands to interact with. Learn more about them by typing cast --help. One such useful command is send which is designed to sign and publish a transaction. To view help about send, type cast send --help.

To use send we need a signature and some arguments.

Please call the following in your terminal:

**Note**: Down below use the address you copy-pasted from your terminal, there's a chance it will be different than the one mine was deployed.

cast send 0x5FbDB2315678afecb367f032d93F642f64180aa3 "store(uint256)" 1337 --rpc-url $RPC\_URL --private-key $PRIVATE\_KEY

**What did we just do?**

Let's break it down:

* cast send is the command we used to sign and publish our transaction;
* 0x5FbDB2315678afecb367f032d93F642f64180aa3 or any other address is the target of our cast send, the contract we are interacting with;
* "store(uint256)" is the [signature of the function](https://ethereum.stackexchange.com/questions/135205/what-is-a-function-signature-and-function-selector-in-solidity-and-evm-language) we are calling.
* 1337 is the number we pass to the store function. As we can see in the function signature, we are expected to provide an uint256 input. You can obviously provide any number you want, as long as it fits uint256.
* you already know what --rpc-url $RPC\_URL --private-key $PRIVATE\_KEY are. The place where we send and the private key we use to sign.

**Reading information from the blockchain**

cast conveniently provides a way to read information stored on the blockchain. Type cast call --help in your terminal to find out more. It works similarly to send, where you have to provide a signature and some arguments. The difference is you are only peering into the storage, not modifying it.

Call the following command in your terminal:

cast call 0x5FbDB2315678afecb367f032d93F642f64180aa3 "retrieve()"

We receive back the following:

0x0000000000000000000000000000000000000000000000000000000000000539

This represents a hex value. In the previous lessons, we learned how to convert this to a normal number.

Type the following command in your terminal:

cast --to-base 0x0000000000000000000000000000000000000000000000000000000000000539 dec

And surprise, surprise, 1337 came back.

I recommend you play around and send multiple transactions with different numbers and then read them from the blockchain.

Awesome! We've learned something very valuable. You are going to use this more times than you can count.

**Up next:** Deploying a smart contract on Sepolia