**Deploying a Contract from a Contract**

**Introduction:**

This lesson covers the process of programmatically deploying a `SimpleStorage` contract and saving it to a storage or state variable. By the end of this lesson, you will have a comprehensive understanding of how one contract can seamlessly deploy and manage another one.

**Creating a new variable:**

Following the format type-visibility-name, we can declare a new state variable of type `SimpleStorage`.

// SPDX-License-Identifier: MIT

Pragma solidity ^0.8.18;

Contract StorageFactory {

SimpleStorage public simpleStorage;

Function createSimplestorageContract() public {

simpleStorage = new SimpleStorage();

}

}

**Important:** `SimpleStorage` on the left and `simpleStorage` on the right are treated as entirely distinct entities due to their differing capitalization. `Simple Storage` refers to the contract type while `simpleStorage` refers to the variable name.

When the new keyword is used, the compiler recognizes the intention to deploy a new contract instance. After compiling, we can proceed to deploy it.

In Remix, you’ll then notice two buttons: an orange `createSimpleStorageContract` and a blue one, `SimpleStorage`, generated by the `public` keyword. If we call both, first `createSimpleStorageContract` and then `SimpleStorage`, the address that appears below confirms that our `SimpleStorage` contract has been deployed.

**Conclusion:**

We have just deployed a contract that can programmatically create another contract, showing the principle of composability. In this way, contracts can know and interact with each other seamlessly.