Experiment No.4

Aim: To demonstrate the basics of Ethereum smart contract development, deployment, and interaction using a simple "Hello World" example.

Once the Truffle is installed, you can start experimenting with it to develop, test, and deploy your smart contracts. Here's a simple experiment to get you started by creating a basic project:

I. Objective:

A. Learn how to write a basic smart contract in Solidity.
B. Understand the process of compiling and deploying a smart contract using Truffle.
C. Practice interacting with the smart contract through the Truffle console.

II. Steps:

1. **Create a New Directory for Your Project:** Create a new directory for your Truffle project and navigate into it.

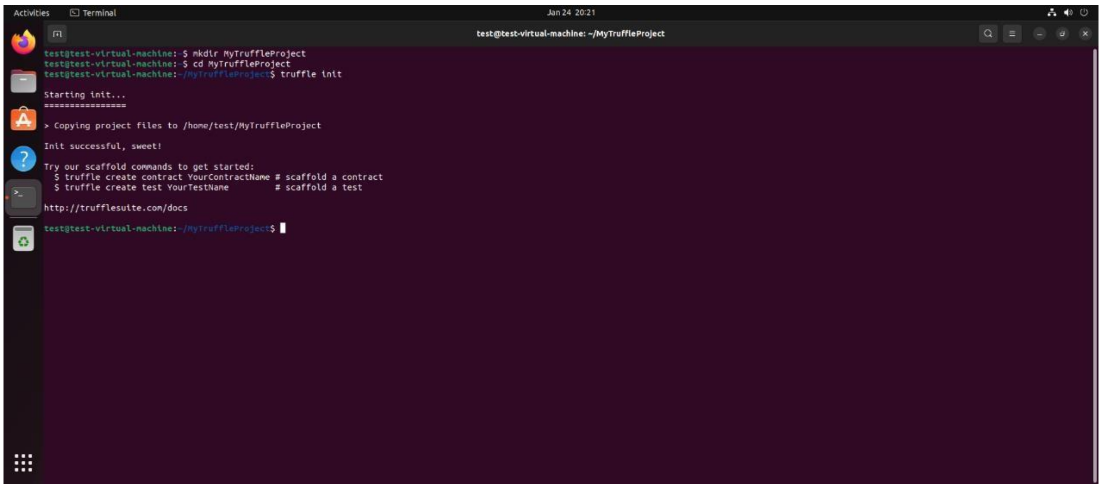
```
mkdir MyTruffleProject
```

```
cd MyTruffleProject
```

2. **Initialize a New Truffle Project:** Initialize a new Truffle project within the directory.

```
truffle init
```

This command sets up a new Truffle project with the necessary configuration files and directories. You'll see folders like contracts/, migrations/, and test/, along with a truffle-config.js configuration file.



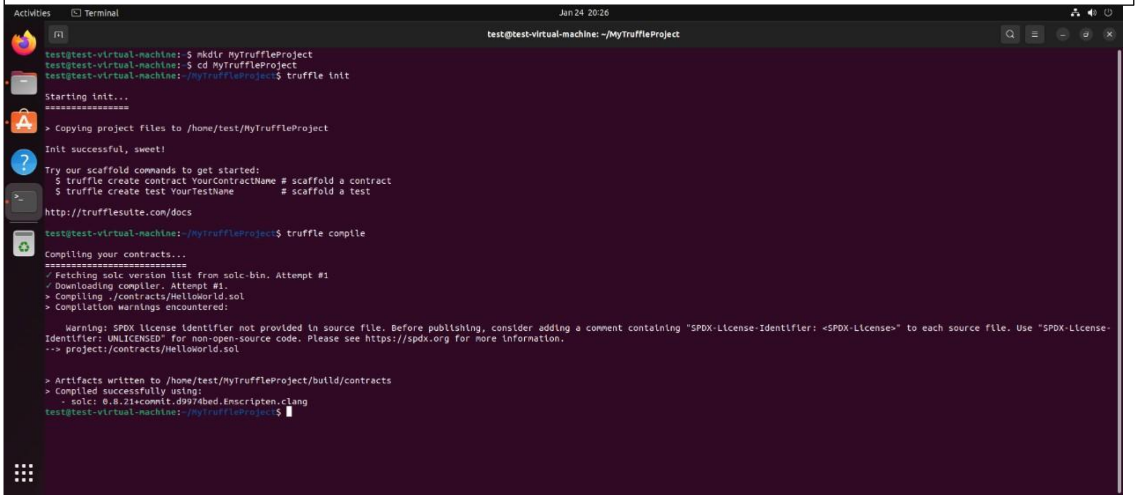
```
test@test-virtual-machine:~$ mkdir MyTruffleProject
test@test-virtual-machine:~$ cd MyTruffleProject
test@test-virtual-machine:~/MyTruffleProject$ truffle init

Starting init...
================

> Copying project files to /home/test/MyTruffleProject

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs

test@test-virtual-machine:~/MyTruffleProject$
```

3. **Create a Simple Smart Contract:** Create a new file within the contracts/ directory to hold your smart contract. You can use a simple contract like a HelloWorld contract. For example, create HelloWorld.sol.

```
// contracts/HelloWorld.sol

pragma solidity ^0.8.0;

contract HelloWorld {
        function sayHello() public pure returns (string memory) {

            return "Hello, World!";

        }

}
```

4. **Compile Your Smart Contract:** Compile your smart contract using Truffle.

```
truffle compile
```



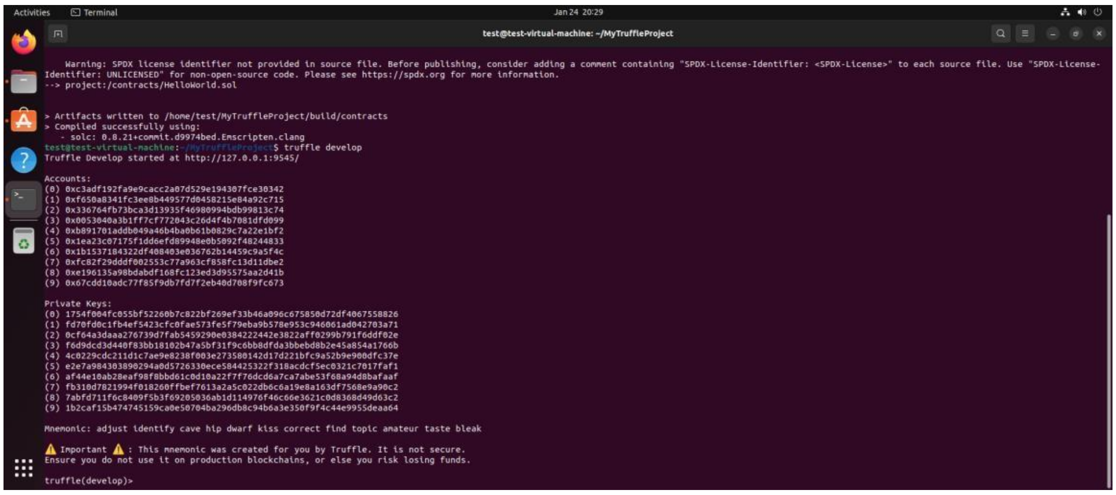
```
test@test-virtual-machine:~$ mkdir MyTruffleProject
test@test-virtual-machine:~$ cd MyTruffleProject
test@test-virtual-machine:~/MyTruffleProject$ truffle init

Starting init...
================

> Copying project files to /home/test/MyTruffleProject

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs

test@test-virtual-machine:~/MyTruffleProject$ truffle compile

Compiling your contracts...
===========================
✓ Fetching solc version list from solc-bin. Attempt #1
✓ Downloading compiler. Attempt #1.
> Compiling ./contracts/HelloWorld.sol
> Compilation warnings encountered:

    Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> project:/contracts/HelloWorld.sol


> Artifacts written to /home/test/MyTruffleProject/build/contracts
> Compiled successfully using:
   - solc: 0.8.21+commit.d9974bed.Emscripten.clang
test@test-virtual-machine:~/MyTruffleProject$
```

5. **Write a Migration Script:** Migrations are JavaScript files that help you deploy contracts to the Ethereum network. Create a new file in the migrations/ directory to deploy your HelloWorld contract, for example, 2_deploy_contracts.js.

```
const HelloWorld = artifacts.require("HelloWorld");

module.exports = function (deployer) {

    deployer.deploy(HelloWorld);

};
```

6. **Start a Local Ethereum Blockchain:** Truffle Develop provides a built-in blockchain for development purposes.

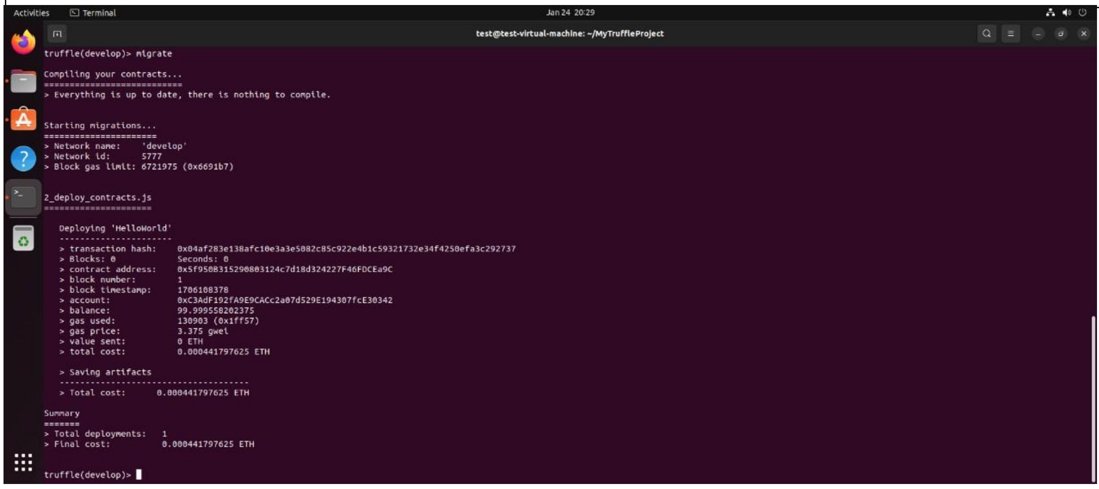
```
truffle develop
```



```
   Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-
Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> project:/contracts/HelloWorld.sol


> Artifacts written to /home/test/MyTruffleProject/build/contracts
> Compiled successfully using:
   - solc: 0.8.21+commit.d9974bed.Emscripten.clang
test@test-virtual-machine:~/MyTruffleProject$ truffle develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0xc3adf192fa9e9cacc2a07d529e194307fce30342
(1) 0xf650a8341fc3ee8b449577d0458215e84a92c715
(2) 0x336764fb73bca3d13935f46980994bdb99813c74
(3) 0x0053040a3b1ff7cf772043c26d4f4b7081dfd099
(4) 0xb891701addb049a46b4ba0b61b0829c7a22e1bf2
(5) 0x1ea23c07175f1dd6efd89948e0b5092f48244833
(6) 0x1b1537184322df408403e036762b14459c9a5f4c
(7) 0xfc82f29dddf002553c77a963cf858fc13d11dbe2
(8) 0xe196135a98bdabdf168fc123ed3d95575aa2d41b
(9) 0x67cdd10adc77f85f9db7fd7f2eb40d708f9fc673

Private Keys:
(0) 1754f004fc055bf52260b7c822bf269ef33b46a096c675850d72df4067558826
(1) fd70fd0c1fb4ef5423cfc0fae573fe5f79eba9b578e953c946061ad042703a71
(2) 0cf64a3daaa276739d7fab5459290e0384222442e3822aff0299b791f6ddf02e
(3) f6d9dcd3d440f83bb18102b47a5bf31f9c6bb8dfda3bbebd8b2e45a854a1766b
(4) 4c0229cdc211d1c7ae9e8238f003e273580142d17d221bfc9a52b9e900dfc37e
(5) e2e7a984303890294a0d5726330ece584425322f318acdcf5ec0321c7017faf1
(6) af44e10ab28eaf98f8bbd61c0d10a22f7f76dcd6a7ca7abe53f68a94d8bafaaf
(7) fb310d7821994f018260ffbef7613a2a5c022db6c6a19e8a163df7568e9a90c2
(8) 7abfd711f6c8409f5b3f69205036ab1d114976f46c66e3621c0d8368d49d63c2
(9) 1b2caf15b474745159ca0e50704ba296db8c94b6a3e350f9f4c44e9955deaa64

Mnemonic: adjust identify cave hip dwarf kiss correct find topic amateur taste bleak

⚠ Important ⚠ : This mnemonic was created for you by Truffle. It is not secure.
Ensure you do not use it on production blockchains, or else you risk losing funds.

truffle(develop)>
```

7. **Deploy Your Contract:** Inside the Truffle Develop console, deploy your contract to the local blockchain.

Migrate



```
truffle(develop)> migrate

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.


Starting migrations...
======================
> Network name:    'develop'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


2_deploy_contracts.js
=====================

   Deploying 'HelloWorld'
   ----------------------
   > transaction hash:    0x04af283e138afc10e3a3e5082c85c922e4b1c59321732e34f4250efa3c292737
   > Blocks: 0            Seconds: 0
   > contract address:    0x5f950B315290803124c7d18d324227F46FDCEa9C
   > block number:        1
   > block timestamp:     1706108378
   > account:             0xC3AdF192fA9E9CACc2a07d529E194307fcE30342
   > balance:             99.999558202375
   > gas used:            130903 (0x1ff57)
   > gas price:           3.375 gwei
   > value sent:          0 ETH
   > total cost:          0.000441797625 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:      0.000441797625 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.000441797625 ETH


truffle(develop)> ▌
```

III. **Conclusion:** This basic experiment introduces you to the process of developing with Truffle, including writing, compiling, deploying, and interacting with a smart contract. From here, you can explore more complex contracts, write tests in JavaScript or Solidity, and deploy to public testnets or the main Ethereum network.