**Demonstrate the installation process of Truffle suite and display  
the current version of Truffle installed in your computer.**

Truffle Setup[¶](https://ethereum-blockchain-developer.com/2022-06-nft-truffle-hardhat-foundry/03-truffle-setup/#truffle-setup)

installing Truffle is actually quite easy.

Open a Terminal (here with Ubuntu 20.04 in WSL2) and install Truffle globally.

npm install -g truffle

It will install truffle in your global npm directory. On Linux that's in ~/.npm-global/lib/node\_modules/truffle, and in a similar directory on Windows and Linux. As these are in the global path you can then simply type in "truffle" from anywhere on the system.

## First Truffle Project Initialization[¶](https://ethereum-blockchain-developer.com/2022-06-nft-truffle-hardhat-foundry/03-truffle-setup/#first-truffle-project-initialization)

Let's now create an empty folder and get directly into the VSCode Environment. As we're doing Truffle, Hardhat and Foundry let's create a subdirectory for Truffle in our NFT-Project:

nft-project

|\_ /truffle

|\_ /hardhat

|\_ /foundry

So, lets create the first folder for truffle:

mkdir -p nft-project/truffle

code nft-project/truffle

You should have an empty VSCode Project. Hit "Terminal" -> "New Terminal" (Here on my German Keyboard its Ctrl+Shift+ö)

## Truffle Default Folder Structure[¶](https://ethereum-blockchain-developer.com/2022-06-nft-truffle-hardhat-foundry/03-truffle-setup/#truffle-default-folder-structure)

In (almost) every truffle project, there's the same folder structure. As Truffle was the first Solidity Development Toolkit, other Toolkits adopted the same naming convention (or because its very self-explanatory somehow).

1. There's a contracts folder, which contains the Smart Contracts
2. There is a migrations folder, which contains scripts to deploy the contracts
3. There is a tests folder, which contains the unit tests
4. There is a config file named truffle-config.js which contains sensible defaults.

## Adding our NFT[¶](https://ethereum-blockchain-developer.com/2022-06-nft-truffle-hardhat-foundry/03-truffle-setup/#adding-our-nft)

As truffle builds itself upon nodejs and npm, we can also initialize the folder with npm init -y to add a package.json and add additional packages, such as the openzeppelin contracts:

npm init -y

echo "node\_modules" > .gitignore

npm install --save @openzeppelin/contracts

**display  
the current version of Truffle installed in your computer.**

In this tutorial we will be covering:

1. Setting up the development environment
2. Creating a Truffle project using a Truffle Box
3. Writing the smart contract
4. Compiling and migrating the smart contract
5. Testing the smart contract
6. Creating a user interface to interact with the smart contract
7. Interacting with the dapp in a browser

Background[¶](https://trufflesuite.github.io/tutorial/#background)

Pete Scandlon of Pete's Pet Shop is interested in using Ethereum as an efficient way to handle their pet adoptions. The store has space for 16 pets at a given time, and they already have a database of pets. As an initial proof of concept, **Pete wants to see a dapp which associates an Ethereum address with a pet to be adopted.**

The website structure and styling will be supplied. **Our job is to write the smart contract and front-end logic for its usage.**

Setting up the development environment[¶](https://trufflesuite.github.io/tutorial/#setting-up-the-development-environment)

There are a few technical requirements before we start. Please install the following:

1. Truffle initializes in the current directory, so first create a directory in your development folder of choice and then moving inside it.

mkdir pet-shop-tutorial

cd pet-shop-tutorial

1. We've created a special [Truffle Box](https://trufflesuite.github.io/boxes) just for this tutorial called pet-shop, which includes the basic project structure as well as code for the user interface. Use the truffle unbox command to unpack this Truffle Box.

truffle unbox pet-shop

### Your first function: Adopting a pet[¶](https://trufflesuite.github.io/tutorial/#your-first-function-adopting-a-pet)

Let's allow users to make adoption requests.

1. Add the following function to the smart contract after the variable declaration we set up above.

// Adopting a pet

function adopt(uint petId) public returns (uint) {

require(petId >= 0 && petId <= 15);

adopters[petId] = msg.sender;

return petId;

}