# Exercises: Install RSK Node and Deploy Smart Contract

The goal of this exercise is to install and setup **RskJ**. RskJ is a **Java implementation of the RSK node**. Following the instructions below, you will setup a local node to work connected to RSK **Regtest**.

For this exercise, we use **Ubuntu 16.04.3** and install RskJ using **PPAs for Ubuntu**.

## Install Truffle and Solc

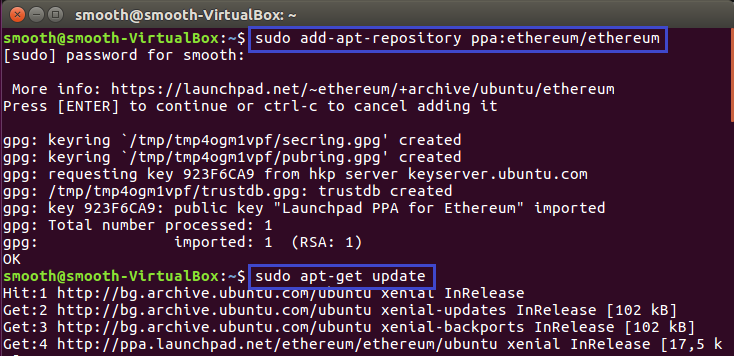
1. We will use truffle for this exercise. First, install **Truffle**. We assume that you have already installed **nodejs** 8+ and **curl**.

|  |
| --- |
| sudo npm install -g truffle |



1. Install Solidity Compiler. We will use PPAs for Ubuntu, for the latest stable version.

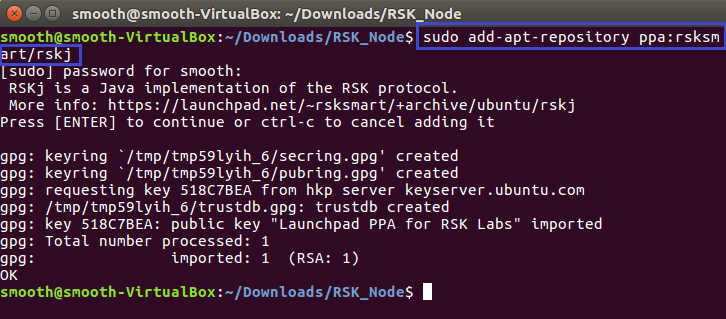
|  |
| --- |
| sudo add-apt-repository ppa:ethereum/ethereum  sudo apt-get update  sudo apt-get install solc |



## Install the RSK Node

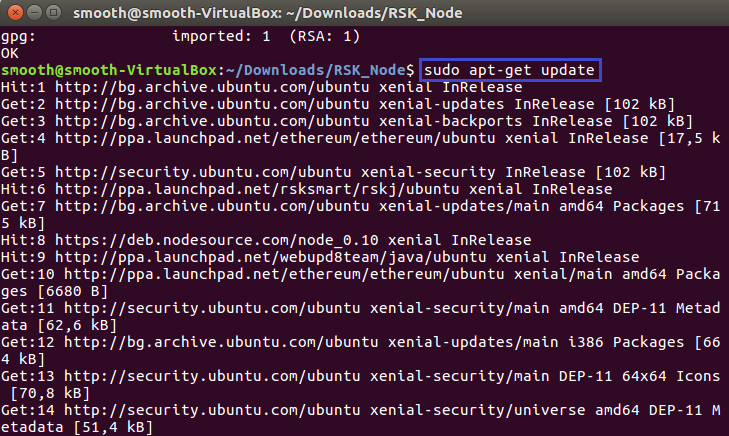
1. Now, install **RskJ** using RSK's **PPAs** for Ubuntu. Type this command:

|  |
| --- |
| sudo add-apt-repository ppa:rsksmart/rskj |



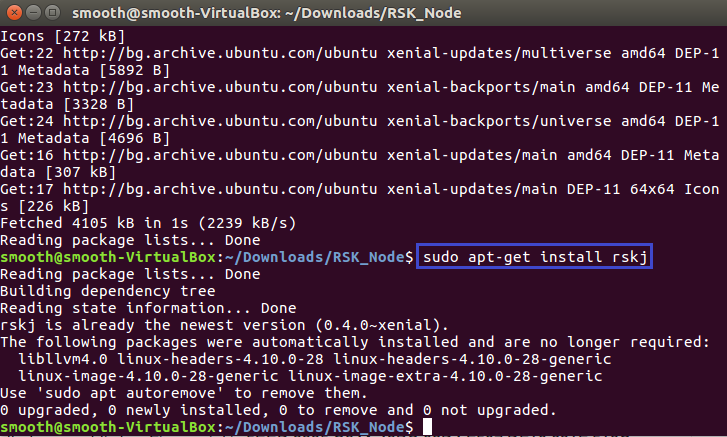
1. Next make update. Type:

|  |
| --- |
| sudo apt-get update |

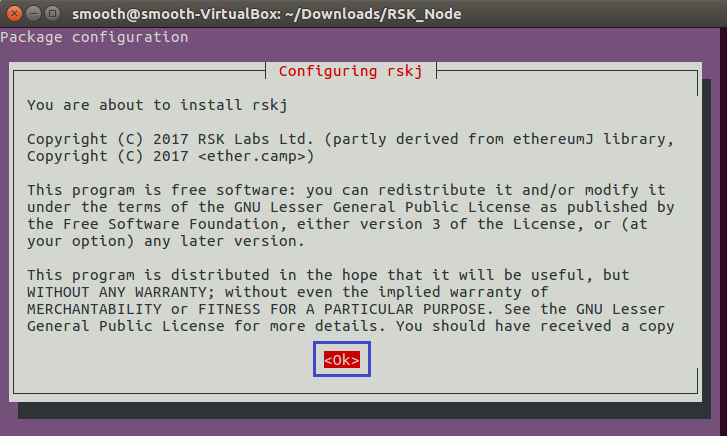


1. Now is time to **install RskJ**. Type:

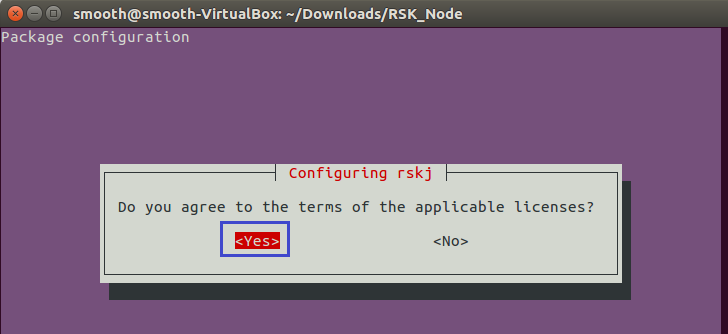
|  |
| --- |
| sudo apt-get install rskj |



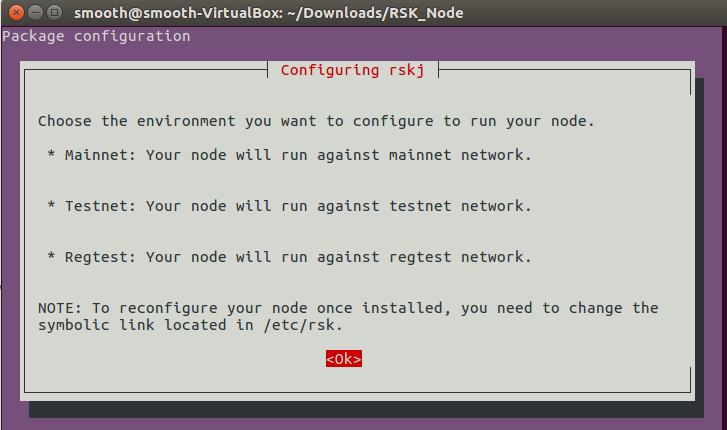
1. Confirm your wish to install rskj.Click **[OK].**



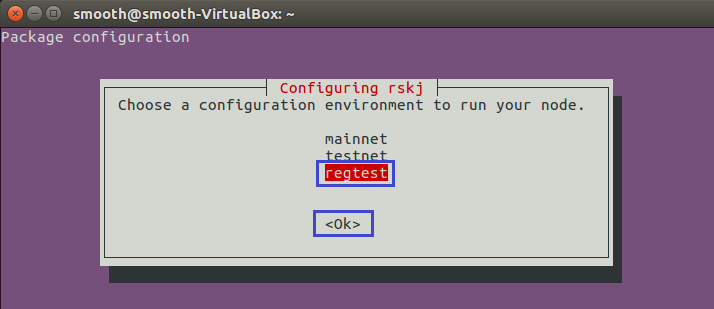
1. Accept the terms of applicable license. Click **[Yes]**.



1. Now the message informs you that RSK Node can be installed in three environments: Mainnet (default), Testnet and Regtest.Click **[OK]**.

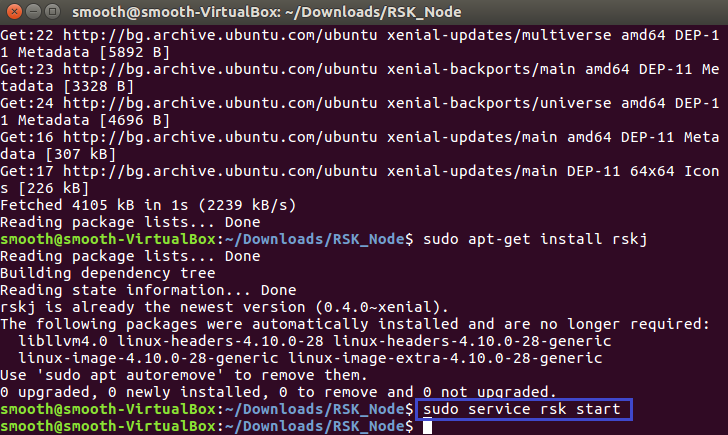


1. In the first window and in the next choose **Regtest** andclick “**OK**”.



1. To **start** the node type**:**

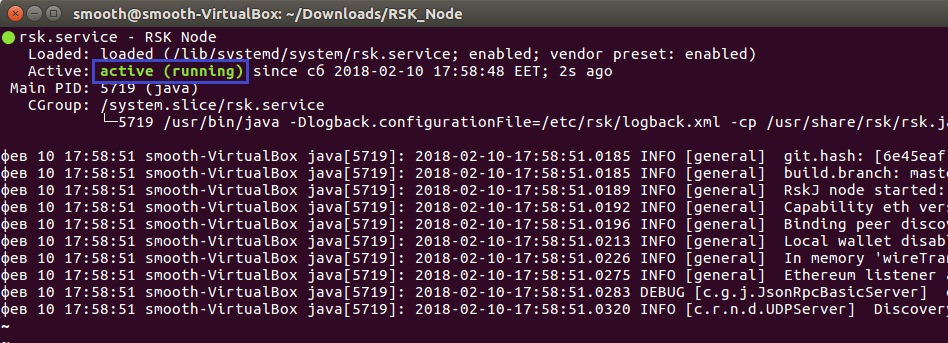
|  |
| --- |
| sudo service rsk start |



In response, you will receive blank line from console.

1. You can check the **node status** with command:

|  |
| --- |
| sudo service rsk status |



1. If you want to **stop** the node the command is:

|  |
| --- |
| sudo service rsk stop |

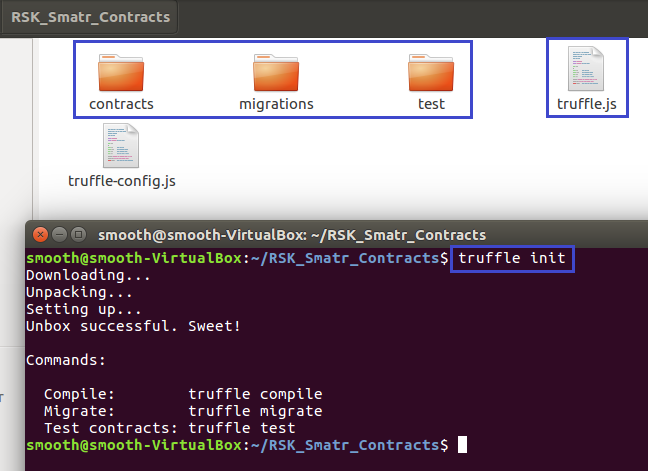
1. If you want to **restart** the node the command is:

|  |
| --- |
| sudo service rsk restart |

## Setup Truffle and Test the Node

1. Configure **truffle**. **Create** directory RSK\_Smart\_Contracts. Open **console** and write **command**:

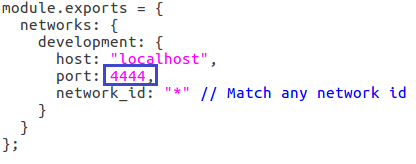
|  |
| --- |
| truffle init |



Truffle init create three directories.

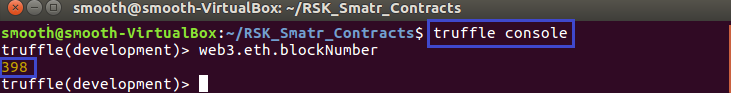
* **contracts -** there we will store our smart contracts like our contract **StoreSomeData.sol**.
* **migrations** – contains .**js** files which setup migrations and starts with number like **1\_initial\_migration.js**
* **tests** – containing tests

1. When initialize, in his directory, truffle creates a very important file **truffle.js** who contains setup parameters. Let’s **open this file** and **configure truffle**. Make the file looks like the picture. **Default port RSK Node using** is **4444.** Afterwards we will change this file and add the keys but this is enough for now.



1. Open truffle console and test the Node. Type:

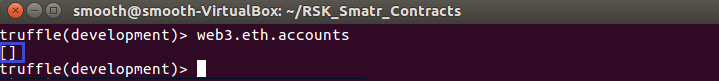
|  |
| --- |
| truffle console  web3.eth.blockNumber |



**Last block** number is **398**, so **our node works**.

1. Now to deploy smart contracts we need **account**. Let’s see what is situation.

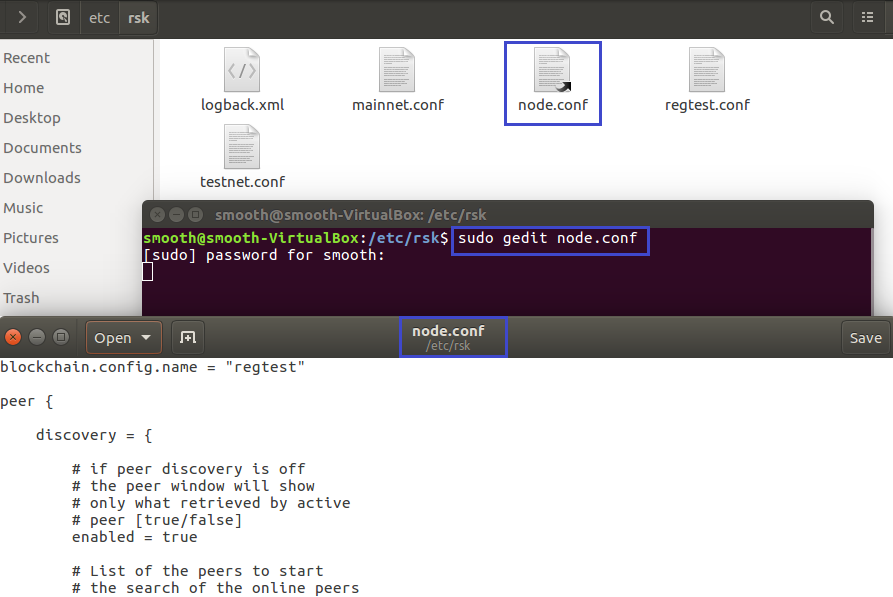
|  |
| --- |
| web3.eth.accounts |



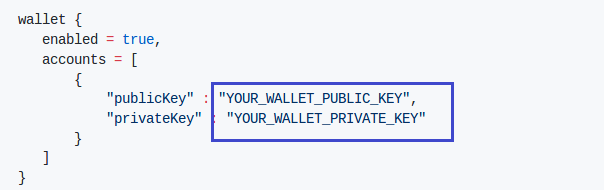
Well, it **returns an empty array**, so we **have no accounts** yet.

1. Account creating is little more complicated. **Open** node configuration file in **/etc/rsk/node.conf.** The file is write protected so open console and type command:

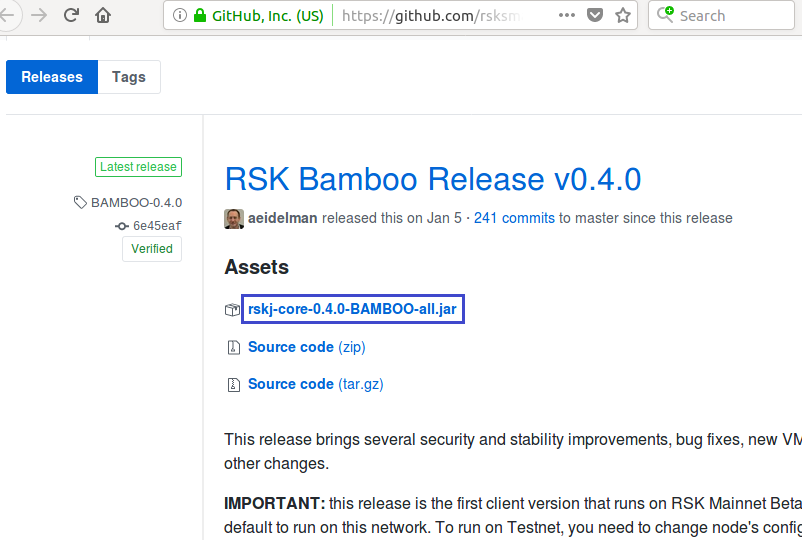
|  |
| --- |
| sudo gedit node.conf |



1. Find the **wallet** section:

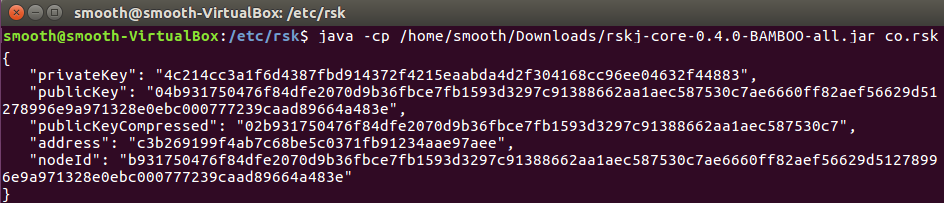


1. So obviously, we need the **private** and **public** **key**. For this purpose, we must generate them.

The keys can be **generated** using jar. Go to [https://github.com/rsksmart/rskj/releases and download the .jar](https://github.com/rsksmart/rskj/releases%20and%20download%20the%20.jar) file. 

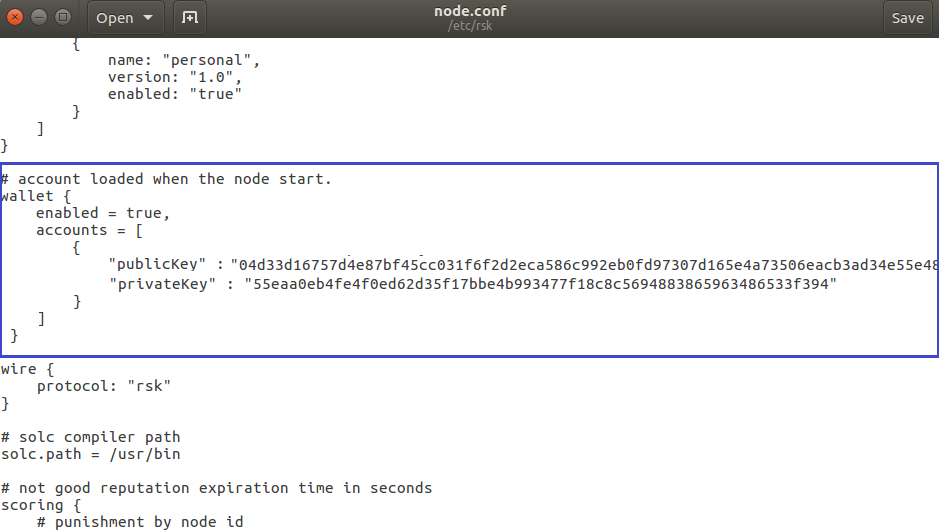
1. **Generate keys**. Type this command:

|  |
| --- |
| java -cp <PATH-TO-THE-RSKJ-FATJAR> co.rsk.GenNodeKeyId |



1. Here are the already generated **keys** and an example of the final code in **node.conf**. Save the file.

|  |
| --- |
| {  "privateKey": "55eaa0eb4fe4f0ed62d35f17bbe4b993477f18c8c5694883865963486533f394",  "publicKey": "04d33d16757d4e87bf45cc031f6f2d2eca586c992eb0fd97307d165e4a73506eacb3ad34e55e48cb513b1ebf5900c43fb55ee8983d71f6c39f3edfce4f8e43d62b",  "publicKeyCompressed": "03d33d16757d4e87bf45cc031f6f2d2eca586c992eb0fd97307d165e4a73506eac",  "address": "7c898904afa613974daa2e01ff5079598e1721d3",  "nodeId": "d33d16757d4e87bf45cc031f6f2d2eca586c992eb0fd97307d165e4a73506eacb3ad34e55e48cb513b1ebf5900c43fb55ee8983d71f6c39f3edfce4f8e43d62b"  } |

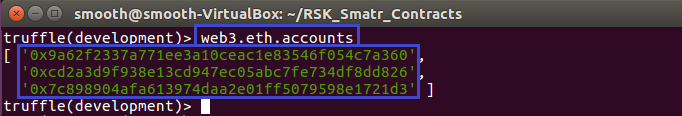


1. **Restart** the node:

|  |
| --- |
| sudo service rsk restart |

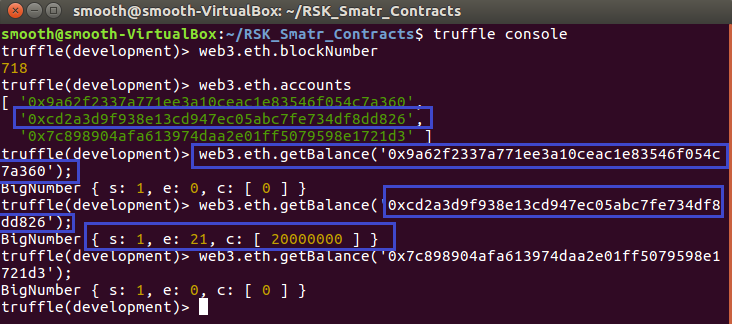
1. Open **truffle console** and **check the accounts**:

|  |
| --- |
| web3.eth.accounts |



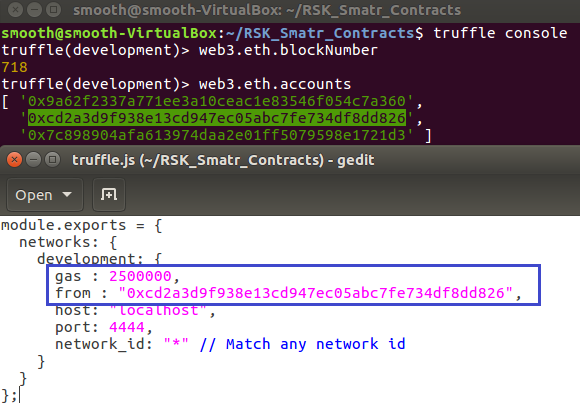
1. Check the **funds in accounts**. Type in truffle console:

|  |
| --- |
| web3.eth.getBalance(`<ACOUNT NUMBER>`); |



The second account has funds and we will use it to deploy smart contract.

1. To deploy the contracts, we need to **specify the account to be used by Truffle**. **Address** with **some funds** must be added in the truffle configuration file. For gas, you just need to set the value to something meaningful, like 2500000. Open the **truffle.js** and change it:

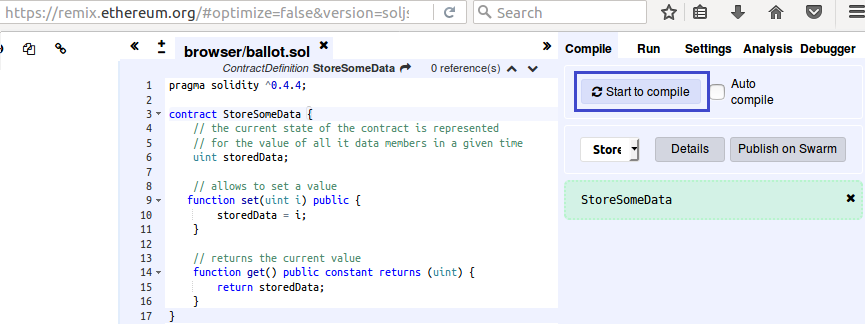


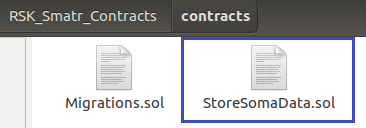
**Exit** from truffle console.

## Create and Deploy Smart Contract

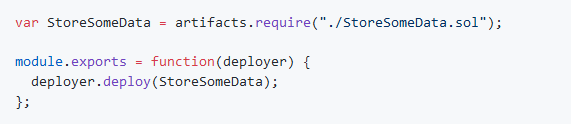
1. Now is time to create our smart contract. Open **remix** Solidity IDE and write a simple contract. Compile the contract. When the contract is compiled, copy the contract and paste it in file **StoreSomeData.sol** in the **contracts** folder (you can do it directly from **Truffle console** typing **truffle create contract StoreSomeData**)

|  |
| --- |
| truffle create contract StoreSomeData |



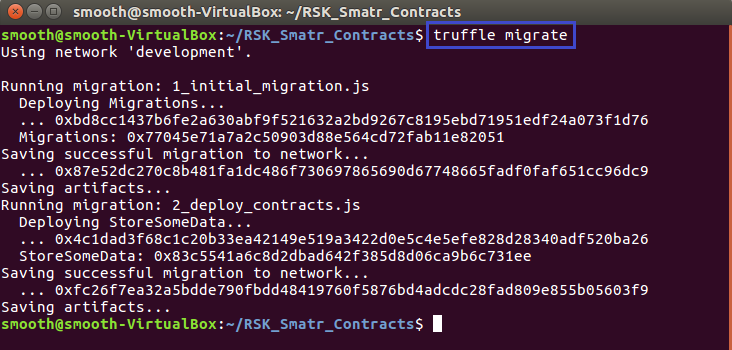


1. Now is time to setup migrations. Go to migrations folder and create file **2\_deploy\_contracts.js.** Open the file and type:



1. We are ready to compile the contract. Type:

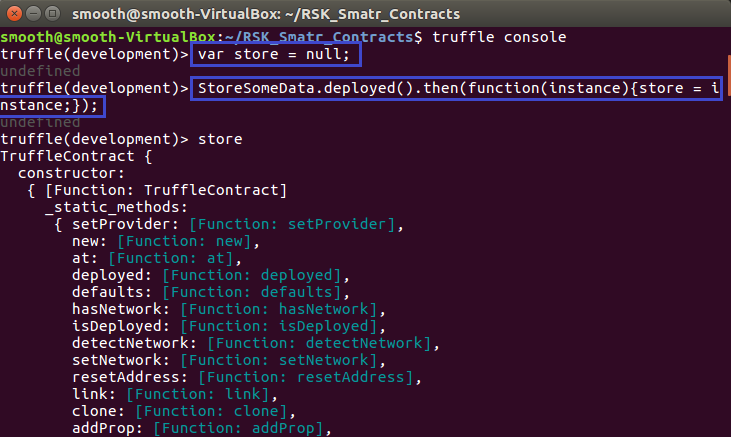
|  |
| --- |
| truffle compile  truffle migrate |



It will **take some time** because **the block must be mined**. Finally, we did this! Congratulations, our contract is in RSK blockchain!

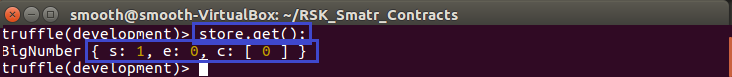
1. Now **interact with the contract**. First step to interact with our contract is **getting a reference**, in the Truffle console write:

|  |
| --- |
| var store = null;  StoreSomeData.deployed().then(function(instance){store = instance;}); |



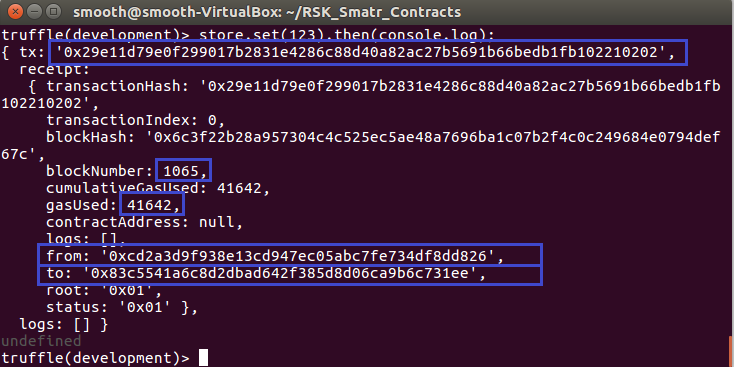
1. Next let's get the value of contract's variable.

|  |
| --- |
| store.get(); |



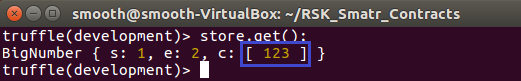
1. Well the value is “0”, let's change it: type command **“store.set(123).then(console.log);”** in truffle console. We call the set operation of our contract, and this is an asynchronous operation, so we get a promise and we pass **console.log** as a parameter, and the result is dumped in the console when the promise is solved. When setting a variable, we send a transaction. Then, this **transaction** has become partofa **new block** intheblockchain. After some time, the block with our transaction is mined and we can see the **transaction information**.

|  |
| --- |
| store.set(123).then(console.log); |



1. Let’s check the state of store again:

|  |
| --- |
| store.get(); |



The new value is **123**. Congratulations! Now you know how to deploy smart contracts in RSK blockchain, and interact with them.

# What to Submit?

Create a **zip file** (e.g. your-name-rsk-smart-contracts-exercise.zip) holding the screenshots with your experiments. Make screenshots of console with running node, migrations and contract’s transactions.

Submit your **zip** file as **homework** at the course Web site.