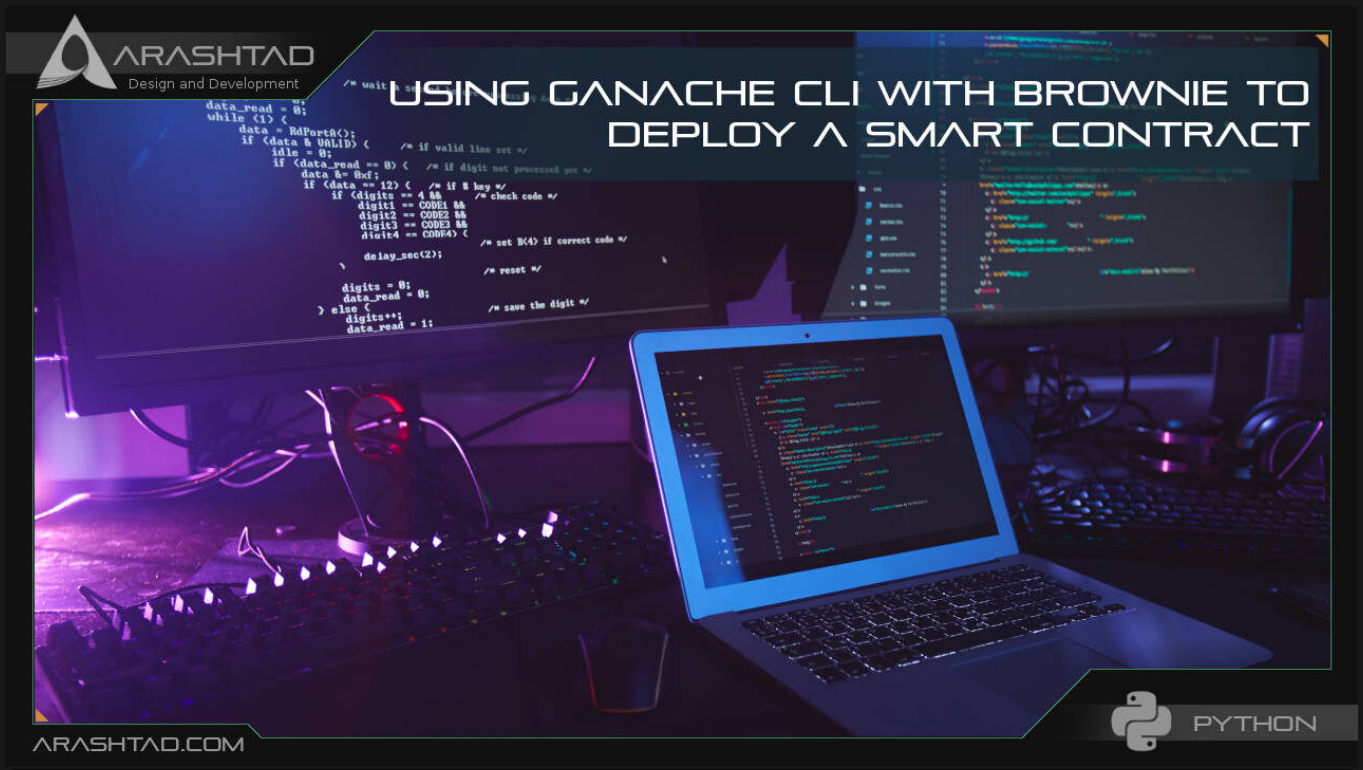
Using Ganache CLI with Brownie to Deploy A Smart Contract

No comments



In this article, after installing Brownie, we are going to install the Ganache CLI (Command Line Interface) and use it with Brownie. The Ganache CLI is the same as Ganache IDE with the difference that it automatically runs without the need to open the Ganache IDE. At the end end of this tutorial, we will be able to deploy our smart contracts on top of Ganache CLI simulated blockchain.

Installing Ganache:

To continue our journey with Brownie, we need to make sure Ganache CLI has been installed successfully. To make sure that it has, and also see its virtual accounts, in your console (terminal) write:

```
ganache
```

And it should give you the test accounts like this:

```
ganache v7.0.3 (@ganache/cli: 0.1.4, @ganache/core: 0.1.4)
```

```
Starting RPC server

Available Accounts
==================
(0) 0x960F41C52ffAef71fFef2fFBC3C0EA5Dc5748086 (1000 ETH)
(1) 0x39Cf64532C1126c6C9Fa0a02338389d1eb861A12 (1000 ETH)
(2) 0xA1E47c46425e24565666ED350e1edEf6532f348c (1000 ETH)
(3) 0xE441bb58b5369DB428a979fbE77Df2d3Da1F91d4 (1000 ETH)
(4) 0x4498662A9691cfef775d3D212f5938905B5de70E (1000 ETH)
(5) 0x4Da709036c65CB81bF74970945f21Ac2e42e9cA9 (1000 ETH)
(6) 0x32872e4dDB4c4876e7B1359013696344fd976C22 (1000 ETH)
(7) 0x6aadaB117c9f26726D5e23b9A0fdC475F3BEd181 (1000 ETH)
(8) 0xCf763def60f3247CE5bEF18BAfC080dA3573595b (1000 ETH)
(9) 0x3a6AFe48C0889820aa587E9Dd45D1B7d8E4e5039 (1000 ETH)

Private Keys
==================
(0) 0x1093c2edf08326559ffbaccfe2a5c5826d2bee9b42fc71c51ad8d989100a3a31
(1) 0x4ebeaf4a4bb07a875255d1f4627146dcc69c930cc380701e1f338d1975240135
(2) 0xa25ad60447f26ad265962660c88bb04ff671f3a55c0f59985530088dad82c9c8
(3) 0x706fce79c758f53fc9336507a35d66ec285c8f52147a6f7676a0e89925f21d0c
(4) 0xd92dc461456414fa41af48ca8806a50031bdc4d587b8e49b1a3f501790f4b853
(5) 0x0b467e91bb9fef30c51077d9a7ba4b9d0797fe46f7fc1cff202751d9bfcf0bb0
(6) 0xaa05bda4d14e2fd3cac04174feeca371567b8f990acc6fab001f0b3317680d50
(7) 0x3a263493e9763bb5b7f069cd51cbca836298831d59b7f57125c937f8ef76dd82
(8) 0x4d6edbb1e7d4283cdf7cc8ce802ef72bcb7b684a51ab7098834e48a638bcfae5
(9) 0x5cb8a425439d9db86c6a58438fb5a6fa425ab60c675d9fd90bdc14cf093bf33c

HD Wallet
==================
Mnemonic:      patrol juice song save pretty combine dish table sock robust g
Base HD Path:  m/44'/60'/0'/0/{account_index}

Default Gas Price
==================
2000000000

BlockGas Limit
==================
30000000

Call Gas Limit
==================
50000000

Chain Id
==================
1337

RPC Listening on 127.0.0.1:8545
```

If you don't get the above results or you get:

```
ganache is not recognized …
```

You might need to install it and that shows that Ganache CLI has not yet been installed. Notice that on Linux, instead of writing:

```
npm install –-global ganache
```

You should add sudo at the beginning of the command. Otherwise, it will throw an error saying that the node modules folder is not in usr/local/lib directory. So for Linux, we install the Ganache CLI like this:

```
sudo npm install –global ganache
```

Also, before installing the Ganache CLI, you need to make sure you have the node installed on your OS. You can check it by typing:

```
node -v
```

Result:

```
v16.14.2
```

Deploying the Contract Using Ganache CLI:

After that, we make sure that all the dependencies have been installed, we can continue with Brownie scripts on deploy.py:
First, run Brownie:

```
brownie
```

Then, fetch one of the Ganache accounts using the following code in the deploy.py file:

```
from brownie import accounts

def deploy_simple_storage():
 account = accounts[0]
print(account)

def main():
 deploy_simple_storage()
```

We will run the project by the following command in the terminal (In the directory of the project's folder):

```
brownie run scripts/deploy.py
```

Result:

```
Brownie v1.18.1 - Python development framework for
```

```
EthereumBrownieSimpleStorageProject is the active project. Attached to
local RPC client listening at '127.0.0.1:8545'...Running
'scripts/deploy.py::main'...
0x960F41C52ffAef71fFef2fFBC3C0EA5Dc5748086
```

Using Environment Variables for the Private Key:

If you can remember from the previous tutorial, we added the Metamask account to Brownie and the terminal asked for the private key. Now, if we do not want to enter the password every time and make the process run automatically without asking the user for a private key or password, we go after the environment variables again. But, this time is a little a bit different. Like before, we create a .env file and paste our private key:

```
export PRIVATE_KEY= “here goes your private key”
```

Then, we create a file called brownie-config.yaml and paste the following code:

```
dotenv: .env
wallets:
from_key: ${PRIVATE_KEY}
```

Then, we modify our deploy.py file as below:

```
from brownie import accounts, config

def deploy_simple_storage():
 account = accountsadd(config["wallets"]["from_key"])
print(account)

def main():
 deploy_simple_storage()
```

And if we run it by the following command in the terminal:

```
brownie run scripts/deploy.py
```

Result:

```
Brownie v1.18.1 - Python development framework for
EthereumBrownieSimpleStorageProject is the active project. Attached to
local RPC client listening at '127.0.0.1:8545'...Running
'scripts/deploy.py::main'...
```

```
0x25E681EE76469E4cF846567b772e94e082907117
```

Now to actually deploy the simple storage contract, we import the SimpleStorage in the deploy.py file and also add the contract.deploy({}) line to the first function:

```
def deploy_simple_storage():


from brownie import accounts, config, SimpleStorage

def deploy_simple_storage():
 account = accountsadd(config["wallets"]["from_key"])
 SimpleStoragedeploy({"from": account})

def main():
deploy_simple_storage()
```

And

```
brownie run scripts/deploy.py
```

Result:

```
Brownie v1.18.1 - Python development framework for
EthereumBrownieSimpleStorageProject is the active project.Launching
'ganache-cli --chain.vmErrorsOnRPCResponse true --server.port 8545 --
miner.blockGasLimit 12000000 --wallet.totalAccounts 10 --hardfork
istanbul --wallet.mnemonic brownie'...Running
'scripts/deploy.py::main'... Transaction sent:
0x1d70d1a30b0266da050d696c14b749bb5e34425159817e5f6e66b1e615221685 Gas
price: 0.0 gwei Gas limit: 12000000 Nonce: 0 SimpleStorage.constructor
confirmed Block: 1 Gas used: 334180 (2.78%) SimpleStorage deployed at:
0x647Dc9EE9731d35b9031A9ca6c664D6b75d14385Terminating local RPC
client…
```

Deploying the Contract with the Ganache CLI Accounts

You can also test this with the Ganache accounts:

```
from brownie import accounts, config, SimpleStorage

def deploy_simple_storage():
 account = accounts[0]
 SimpleStoragedeploy({"from": account})

def main():
deploy_simple_storage()
```

Result:

```
Brownie v1.18.1 - Python development framework for
EthereumBrownieSimpleStorageProject is the active project.Launching
'ganache-cli --chain.vmErrorsOnRPCResponse true --server.port 8545 --
miner.blockGasLimit 12000000 --wallet.totalAccounts 10 --hardfork
istanbul --wallet.mnemonic brownie'...Running
'scripts/deploy.py::main'... Transaction sent:
0xbbd21a1abc42f0d21f4651b71cddadddecf6ba99af3a31a140434950c7e36876 Gas
price: 0.0 gwei Gas limit: 12000000 Nonce: 0 SimpleStorage.constructor
confirmed Block: 1 Gas used: 334180 (2.78%) SimpleStorage deployed at:
0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87Terminating local RPC
client...
```

Great! Now we have managed to deploy our contract using Brownie. The Brownie adventure however is goes on in our other articles.

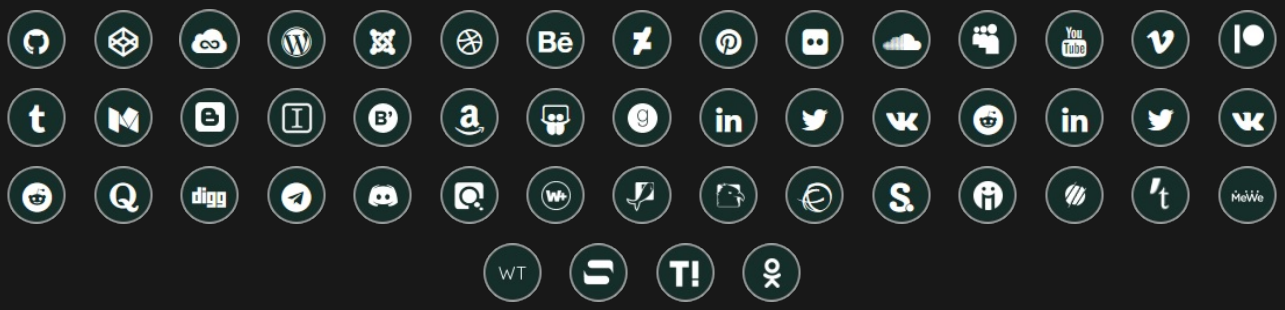
Summing Up

In this article, we have managed to install the Ganache CLI, to be able to automatically use it whenever we want to test our deployment in a simulated blockchain. We have also considered a way to include our private key somewhere safe (in the .env file) out of the deploy.py script file. In the end, we deployed our contract using the Ganache CLI.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.



Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP | NEWSLETTER | RSS FEED