

# Tutorial 5

## Deploying (connecting) Truffle dapps to live networks

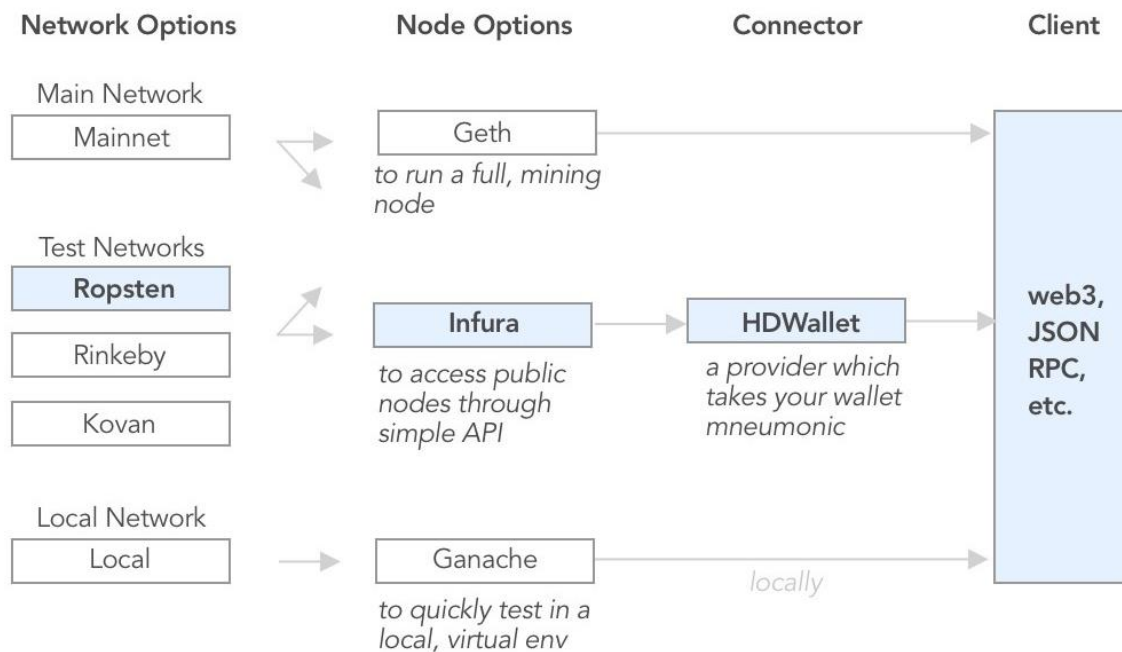
(extension to Tutorial 2)

Prepared by: Reza Parizi

In this tutorial, you will learn how to connect your Truffle project to a public/live blockchain network, i.e., Ethereum mainnet or any public testnets such as Ropsten or Rinkeby. Up until this point, you've used a personal in-memory blockchain (Ganache) to deploy and test your Truffle dapps which is convenient for testing and interacting with your smart contracts during development from your local machines.

We will use the same voting dapp (created in Tutorial 2) to deploy it to Ropsten network--run Tutorial 2 again to refresh your memory.

Below is a general framework depicting all options for common *dapp* setups and what we'll tackle today: **the path in blue** (note, the last path is what we exercised in Tutorial 2).



## Step 1: Setup HDWallet

The [Truffle HDWallet provider](#) is a convenient and easy way to configure network connection to ethereum through infura.io (or any other compatible provider). HDWallet provider simply adds some features required by Truffle that are not available with infura like event filtering and transaction signing.

**Install HDWalletProvider** (the whole guide is [here](#))

```
npm install truffle-hdwallet-provider
```

## Step 2: Configure/update network file

In your project's `truffle-config.js`, add the following snippet inside the file:

```
var HDWalletProvider = require("truffle-hdwallet-provider");
var mnemonic = "orange apple banana ... ";

// Add a Ropsten network definition
module.exports = {
  networks: {
    ropsten: {
      provider: function() {return new HDWalletProvider(mnemonic,
"https://ropsten.infura.io/<Your_Access_Token>")
      },
      network_id: 3
    }
  }
};
```

Note, while the example has only a single network defined in 'module.experts', you can define multiple networks as needed.

For mnemonic, you can use any of mnemonic associated with your wallets (for instance, Metamask account). If you don't have a mnemonic, you can generate one using an [online mnemonic generator](#)

## Step 3: Deploy your contracts and run the dapp

- Migrate your contracts using this command:

```
truffle migrate --network ropsten
```

```
or truffle deploy --network ropsten
```

Note, make sure you will be in the same directory where your Truffle project is located before running this command.

If all goes well, you should see a console log as follows:

```
Running migration: 1_initial_migration.js
Deploying Migrations...
... 0xd79bc3c5a7d338a7f85db9f86febbec9494f49bda8f9f4c90b649db7
Migrations: 0x0c6c4fc8831755595eda4b5724a61ff989e2f8b9
Saving successful migration to network...
... 0xc37320561d0004dc149ea42d839375c3fc53752bae5776e4e7543ad16c1b06f0
Saving artifacts...
```

```
Running migration: 2_deploy_contracts.js
  Deploying Election...
... 0x7efbb3e4f028aa8834d0078293e0db7ff8aff88e72f33960fc806a618a6ce4d3
  MyContract: 0xda05d7bfa5b6af7feab7bd156e812b4e564ef2b1
Saving successful migration to network...
... 0x6257dd237eb8b120c8038b066e257baee03b9c447c3ba43f843d1856de1fe132
Saving artifacts...
```

If you want to verify that your contract(s) was deployed successfully, you can check this on the [Ropsten Etherscan](#). In the search field, type in the transaction hash.

Note, you just need to deploy your contracts once. If you had deployed them before and try to deploy again you will see this message “ Network up to date”. This means you can go ahead and run your dapp.

[optional] Now, let's read some data from the ropsten blockchain to verify our connection to the network.

```
truffle console --network ropsten
```

Let's get some information about the latest block. Type this code into your console:

```
web3.eth.getBlock('latest').then(console.log);
```

- It is time to run your whole dapp:

```
npm run dev
```

Now that your dapp is running, anyone with an account on Metamask that has some (Ropsten) ether in it can use it.