https://github.com/smartcontractkit/truffle-starter-kit)

Chainlink Truffle Starter Kit¶

Implementation of the following 4 Chainlink features using theTruffle development environment: -Chainlink Price Feeds -Chainlink VRF -Chainlink Keepers -Request & Receive data

Getting Started¶

It's recommended that you've gone through theTruffle getting started documentation

before proceeding here.¶

Requirements¶

- git
- You'll know you did it right if you can rungit --version
- and you see a response likegit version x.x.x
- Nodejs
- You'll know you've installed nodejs right if you can run:* node --version
  - and get an ouput like:vx.x.x
- Yarn
- instead ofnpm
- You'll know you've installed yarn right if you can run:* yarn --version
  - And get an output like:x.x.x
  - You might need to install it with npm

If you're familiar withnpx andnpm instead ofyarn , you can usenpx for execution andnpm for installing dependencies.

Quickstart¶

1. Clone and install dependencies

After installing all the requirements, run the following:

git clone https://github.com/smartcontractkit/truffle-starter-kit/cd

truffle-starter-kit Then:yarn

or

npm i

1. You can now do stuff!

Runtruffle test

This will run your JavaScript, TypeScript, and Solidity tests underneath the./test folder. If thedevelopment network is not specified in yourtruffle-config ,truffle test will automatically bring up and tear down a local network onlocalhost:8545 .

Usage¶

If you runtruffle --help you'll get an output of all the tasks you can run.

Deploying Contracts¶

In yourtruffle-config.js you'll see section like:

networks: { This section of the file is where you define which networks you want to interact with. You can read more about that whole file in theTruffle documentation. . To deploy a contract, you would call:

truffle migrate --network This will deploy your contracts to the network you specify. If you do not specifynetwork , by defaulttruffle migrate will deploy to thedevelopment network if it is defined in yourtruffle-config .

If you'd like to interact with your deployed contracts, skip down toInteracting with Deployed Contracts .

Run a Local Network¶

One of the best ways to test and interact with smart contracts is with a local network. To start a local network with all your contracts in it, run the following:

ganache -d This will startganache , a local blockchain with private keys, pre-funded wallets, and an endpoint to potentially add to an EVM wallet.-d specifies a deterministic wallet. You can see all other ways you can configure Ganachehere .

In ourtruffle-config , we've defined the localganache network. To migrate, simply call:

truffle migrate --network ganache

Using a Testnet or Live Network (like Mainnet or Polygon)¶

To interact with a live or test network, you will need ETH and LINK token by following these steps:

1. Get some Sepolia Testnet ETH and LINK

Head over to theChainlink faucets and get some ETH and LINK. Please follow the chainlink documentation if unfamiliar.

1. Create VRF V2 subscription

Head over toVRF Subscription Page and create the new subscription. Save your subscription ID and place it in yourhelper-truffle-config.js undersubId .

Truffle Dashboard¶

This is our recommended way of deploying your contracts. WithTruffle dashboard , you can deploy contracts and sign transactions through MetaMask - so you never exposing your private key! To do so, run:

truffle dashboard

This should bring up the dashboard onlocalhost:24012 , where you can connect your wallet to the network of your choice. Note that if you want to deploy to a testnet or mainnet, you'll need ETH to pay the gas fees. Then, if you want to migrate your contracts, simply call:

truffle migrate --network dashboard

You should then see prompts to sign the transactions on the dashboard view. Check out this video on how to use it with Truffle and HardHathere .

Environment configuration¶

In some cases, you might want your transactions to be automatically signed. In which case, you will need to provide your wallet's private key

1. An RPC URL
2. A private key
3. ETH & LINK token (either testnet or real)

Let's look at an example of setting these up using the Sepolia testnet.

Sepolia Ethereum Testnet Setup¶

First, we will need to set environment variables. We can do so by setting them in our.env file (create it if it's not there). You can also read more aboutenvironment variables from the linked twilio blog. You'll find a sample of what this file will look like in.env.example

IMPORTANT: MAKE SURE YOU'D DONT EXPOSE THE KEYS YOU PUT IN THIS.env FILE. By that, I mean don't push them to a public repo, and please try to keep them keys you use in development not associated with any real funds. 1. Set yourSEPOLIA_RPC_URL 2. environment variable.

You can get one for free fromInfura ,Alchemy , orMoralis . This is your connection to the blockchain.

1. Set yourPRIVATE_KEY
2. environment variable.

This is your private key from your wallet, ieMetaMask . This is needed for deploying contracts to public networks.

WARNING

When developing, it's best practice to use a Metamask that isn't associated with any real money. A good way to do this is to use Truffle Dashboard. Alternatively, make a new browser profile (on Chrome, Brave, Firefox, etc) and install Metamask on that brower, and never send this wallet money.

Don't commit and push any changes to .env files that may contain sensitive information, such as a private key! If this information reaches a public GitHub repository, someone can use it to check if you have any Mainnet funds in that wallet address, and steal them!

.env example:

SEPOLIA_RPC_URL='https://sepolia.infura.io/v3/asdfasdfasdfasdf' PRIVATE_KEY='abcdef' bash exampleexport SEPOLIA_RPC_URL='https://sepolia.infura.io/v3/asdfasdfasdfasdf' export PRIVATE_KEY='abcdef'

For other networks like mainnet and polygon, you can use different environment variables for your RPC URL and your private key. See thetruffle-config.js to learn more.

Uncomment thesepolia network in yourtruffle-config . You should now be all setup! You can run any command and just pass the--network sepolia now!

To deploy all contracts:

truffle migrate --network sepolia

Test¶

Tests are located in thetest directory, and are split between unit tests and staging/testnet tests. Unit tests should only be run on local environments, and staging tests should only run on live environments.

To run unit tests:

truffle test

Interacting with Deployed Contracts¶

After deploying your contracts, the deployment output will give you the contract addresses as they are deployed. You can then use these contract addresses in conjunction with Truffle scripts to perform operations on each contract. Alternatively, if you want to interact with contracts on the fly, you can read about how to usetruffle console ortruffle develop here .

Chainlink Price Feeds¶

The Price Feeds consumer contract has one script, to read the latest price of a specified price feed contract.

You can deploy just the price feed consumer with:

truffle migrate --f 3

--to 3

--network After deployment, run the following:

truffle exec

scripts/readPriceConsumer.js --network

Request & Receive Data¶

The API Consumer contract has one script, to request data from the API and wait for a response.

You can deploy just the API Consumer with:

truffle migrate --f 4 --to 4 --networkAfter deployment, run the following:

yarn truffle exec

scripts/requestAndReadAPI.js --network

VRF Get a random number¶

The VRF Consumer contract has one script, to request a random number and wait for a response.

You can deploy just the VRF Consumer with:

truffle deploy --f 5 --to 5 --networkAfter deployment, you'll need to add your contract address to your subscription. Head over tovrf.chain.link and add your consumer.

Then, run the following:

truffle exec

scripts/requestAndReadRandomNumber.js --network

Keepers¶

The Keepers Consumer contract has one script, to check the upkeep. After deployment, run the following:

truffle exec

scripts/checkUpkeep.js --network To see everything in action, you'll want to set up a consumer atkeepers.chain.link .

Verify on Etherscan¶

You'll need anETHERSCAN_API_KEY environment variable. You can get one from theEtherscan API site. . If you have it set, your deploy script will try to verify them by default, but if you want to verify any manually, you can run:

truffle run verify --network example:

truffle run verify PriceConsumerV3 --network sepolia

Contributing¶

Contributions are always welcome! Open a PR or an issue!

Thank You

Resources¶

- Chainlink Documentation
- Truffle Documentation