TRUFFLECON 2019

WHAT CAN WE DO WITH TRUFFLE PLUGINS?

By Rosco Kalis

ABOUT ME

- Graduate Student at University of Amsterdam
- High level contract language for Bitcoin Cash (CashScript)
- truffle-assertions
- truffle-plugin-verify

CONTENTS

- What are Truffle plugins?
- How do Truffle plugins work?
- What can we do with Truffle plugins?
- Walk through truffle-plugin-verify

WHAT ARE PLUGINS?

1. Install the plugin from NPM.

```
npm install --save-dev truffle-plugin-hello
```

2. Add a `plugins` section to your Truffle config.

```
module.exports = {
  /* ... rest of truffle-config */

  plugins: [
    "truffle-plugin-hello"
  ]
}
```

3. Run the command

```
$ truffle run hello
Hello, World!
```

WHAT PLUGINS EXIST?

- truffle-plugin-verify

```
truffle run verify SmartContractName [--network networkName]
```

- truffle-security

```
truffle run verify [options] [solidity-file[:contract-name] [solidity-file[:contract-name] ...]]
```

HOW DO PLUGINS WORK?

1. Implement the command as a Node module with a function as its default export.

   Example: `hello.js`

```
module.exports = async (config) => {
  if (config.help) {
    console.log(`Usage: truffle run hello [name]`);
    done(null, [], []);
    return;
  }

  let name = config._.length > 1 ? config._[1] : 'World!';
  console.log(`Hello, ${name}`);
}
```

2. Define a `truffle-plugin.json` file to specify the command. Example: `truffle-plugin.json`

```
{
  "commands": {
    "hello": "hello.js"
  }
}
```

THE CONFIG OBJECT

```
Config {
  // Information about the CLI
  truffle_directory: string, // Truffle executable location
  working_directory: string, // CWD
  _: string[], // Raw positional command line arguments

  // Any additional command line flags / arguments
  myString: string // --myString hello
  myint: number // --myInt 5

  // Information included in the truffle-config.js file
  networks: { rinkeby: object, ropsten: object },
  compilers: { solc: object, vyper: {} },
  plugins: string[],
  api_keys: { etherscan: string },
  /* ... any other fields defined in truffle-config.js */

  // Network config
  network: string, // --network rinkeby|ropsten|etc
  network_id: number // Current network id
  network_config: object // Current network config (from truffle-config.js)

  // Directory information
  build_directory: string, // ./build
  contracts_directory: string, // ./contracts
  contracts_build_directory: string, // ./build/contracts
  migrations_directory: string, // ./migrations
  migrations_file_extension_regexp: RegExp,
  test_directory: string, // ./test
  test_file_extension_regexp: RegExp,
}
```

THE CONFIG OBJECT

```
// Information about the CLI
truffle_directory: string, // Truffle executable location
working_directory: string, // CWD
_: string[], // Raw positional command line arguments

// Any additional command line flags / arguments
myString: string // --myString hello
myint: number // --myInt 5
```

THE CONFIG OBJECT

```
// Information included in the truffle-config.js file
networks: { rinkeby: object, ropsten: object },
compilers: { solc: object, vyper: {} },
plugins: string[],
api_keys: { etherscan: string },
/* ... any other fields defined in truffle-config.js */

// Network config
network: string, // --network rinkeby|ropsten|etc
network_id: number // Current network id
network_config: object // Current network config (from truffle-config.js)
```

SUPPLYING PARAMETERS

```
// Any additional command line flags / arguments
myString: string // --myString hello
myint: number // --myInt 5

// Information included in the truffle-config.js file
networks: { rinkeby: object, ropsten: object },
compilers: { solc: object, vyper: {} },
plugins: string[],
api_keys: { etherscan: string },
/* ... any other fields defined in truffle-config.js */
```

THE CONFIG OBJECT

```
// Directory information
contracts_directory: string, // ./contracts
build_directory: string, // ./build
contracts_build_directory: string, // ./build/contracts
migrations_directory: string, // ./migrations
migrations_file_extension_regexp: RegExp,
test_directory: string, // ./test
test_file_extension_regexp: RegExp,
```

CONTRACT INFORMATION

```
contracts_directory: string, // ./contracts
contracts_build_directory: string, // ./build/contracts
```

- Contract source code
  - Flattening, static analysis, ...
- Contract artifacts
  - Bytecode, abi, ast, ...
- Deployed contracts
  - Transactions, events, ...
  - + abi & network config → use contract

TRUFFLE-PLUGIN-VERIFY

```
var HDWalletProvider = require("truffle-hdwallet-provider");
require('dotenv').config();

module.exports = {
  networks: {
    rinkeby: {
      provider: function() {
        return new HDWalletProvider(`${process.env.MNEMONIC}`, `https://rinkeby.infura.io/v3/${process.env.INFURA_ID}`)
      },
      network_id: 4
    }
  },
  plugins: [
    'truffle-plugin-verify'
  ],
  api_keys: {
    etherscan: process.env.ETHERSCAN_API_KEY
  }
};
```

TRUFFLE-PLUGIN-VERIFY

- Extract config information

```
// Truffle handles network stuff, just need network_id
const networkId = config.network_id
const apiUrl = API_URLS[networkId]
enforce(apiUrl, `Etherscan has no support for network ${config.network} with id ${networkId}`)

const apiKey = config.api_keys && config.api_keys.etherscan
enforce(apiKey, 'No Etherscan API key specified')

enforce(config._.length > 1, 'No contract name(s) specified')

const workingDir = config.working_directory
const contractsBuildDir = config.contracts_build_directory
const optimizerSettings = config.compilers.solc.settings.optimizer
const verifyPreamble = config.verify && config.verify.preamble

const contractNames = config._.slice(1)
```

TRUFFLE-PLUGIN-VERIFY

- Extract data from artifact & source
- Retrieve data from Etherscan

```
const artifact = require(`${options.contractsBuildDir}/${contractName}.json`)
const mergedSource = await merge(artifact.sourcePath)

const contractAddress = artifact.networks[`${options.networkId}`].address
const res = await axios.get(
  `${options.apiUrl}?module=account&action=txlist&address=${contractAddress}&page=1&sort=asc&offset=1`
)
const encodedConstructorArgs = res.data.result[0].input.substring(artifact.bytecode.length)
```

TRUFFLE-PLUGIN-VERIFY

- Build & send Etherscan API request

```
const postQueries = {
  apikey: options.apiKey,
  module: 'contract',
  action: 'verifysourcecode',
  contractaddress: artifact.networks[`${options.networkId}`].address,
  sourceCode: mergedSource,
  contractname: artifact.contractName,
  compilerversion: `v${artifact.compiler.version.replace('.Emscripten.clang', '')}`,
  optimizationUsed: options.optimizationUsed,
  runs: options.runs,
  constructorArguements: encodedConstructorArgs
}

const libraries = artifact.networks[`${options.networkId}`].links || {}
Object.entries(libraries).forEach(([key, value], i) => {
  postQueries[`libraryname${i + 1}`] = key
  postQueries[`libraryaddress${i + 1}`] = value
})

const guid = axios.post(options.apiUrl, querystring.stringify(postQueries))
```

TRUFFLE-PLUGIN-VERIFY

- Retrieve verification result
  - Infinite loop

```
while (true) {
  await delay(1000)
  const verificationResult = await axios.get(
    `${options.apiUrl}?module=contract&action=checkverifystatus&apikey=${options.apiKey}&guid=${guid}`
  )
  if (verificationResult.data.result !== VerificationStatus.PENDING) {
    return verificationResult.data.result
  }
}
```

TRUFFLE-PLUGIN-VERIFY

```
$ truffle run verify Casino --network rinkeby

Verifying Casino
Pass - Verified: https://rinkeby.etherscan.io/address/0xaB31b2d77505533cAda0d6D0930507a04d1F47eB#contracts

Successfully verified 1 contract(s).
```

TRUFFLE PLUGIN WISHLIST

- Renaming commands
- Integrated plugin management
  - truffle plugin add / remove
- Hooks into other Truffle processes

WORD OF CAUTION

Be mindful of the access that plugins have

FURTHER READING

- https://www.trufflesuite.com/docs/truffle/getting-started/writing-external-scripts
- https://kalis.me/creating-truffle-plugins/
- https://github.com/rkalis/awesome-truffle-plugins
- https://kalis.me/verify-truffle-smart-contracts-etherscan/
- https://github.com/rkalis/truffle-university-workshop