======================================================================================

**Truffle migrations - 理解部署脚本**（精通以太坊第九章）

Truffle提供了一个名为\_migration\_的部署系统。如果你曾在其他框架中工作过，你可能会看到类似的东西：Ruby on Rails，Python Django和许多其他语言和框架都有+migrate+命令。

在所有这些框架中，migration的目的是处理不同版本软件之间数据模式的变化。以太坊migration的目的略有不同。因为以太坊合约是不可变的，而且要消耗gas部署，所以Truffle提供了一个migration机制来跟踪哪些合约（以及哪些版本）已经部署。在一个拥有数十个合约和复杂依赖关系的复杂项目中，你不希望为了重新部署尚未更改的合约而支付gas。你也不想手动跟踪哪些合约的哪些版本已经部署了。Trufflemigration机制通过部署智能合约+Migrations.sol+完成所有这些工作，然后跟踪所有其他合约部署。

我们只有一份合约，Faucet.sol，这至少意味着migration系统是大材小用的。不幸的是，我们必须使用它。但是，通过学习如何将它用于一个合约，我们可以开始练习一些良好的开发工作习惯。随着事情变得更加复杂，这项努力将会得到回报。

Truffle的+migrations+目录是找到迁移脚本的地方。现在，只有一个脚本 1\_initial\_migration.js，它会部署 Migrations.sol 合约本身：

[[1\_initial\_migration]] .**1\_initial\_migration**.js - the migration script for Migrations.sol

include::code/truffle/Faucet/migrations/1\_initial\_migration.js

var Migrations = artifacts.require("./Migrations.sol");

module.exports = function(deployer) {

deployer.deploy(Migrations);

};

我们需要第二个migration脚本来部署 Faucet.sol。我们称之为“2\_deploy\_contracts.js”。它非常简单，就像“1\_initial\_migration.js”一样，只需稍作修改即可。实际上，你可以复制“1\_initial\_migration.js”的内容，并简单地将Migrations 的所有实例替换为 Faucet ：

[[2\_deploy\_contracts]] **.2\_deploy\_contracts**.js - the migration script for Faucet.sol

include::code/truffle/Faucet/migrations/2\_deploy\_contracts.js

var Faucet = artifacts.require("./Faucet.sol");

module.exports = function(deployer) {

deployer.deploy(Faucet);

};

脚本初始化变量 “Faucet”，将Faucet.sol Solidity源代码标识为定义Faucet的工件。然后，它调用部署功能来部署此合约。

注意：**artifacts.require**("./Faucet.sol")参数如果指定合约名，就只提取指名合约部署；如果指定的是sol文件名，就会部署此文件上全部合约。

我们都准备好了。我们使用 truffle migrate 来部署合约。我们必须使用”--network“参数指定在哪个网络上部署合约。我们只在配置文件中指定了一个网络，我们将其命名为”localnode“。确保你的本地以太坊客户端正在运行，然后输入：

Faucet $ Truffle migrate --network localnode

==========================================================================

mkdir yourContractName

cd yourContractName

**truffle init**

==========================================================================

Try our scaffold commands to get started:

$ truffle create contract YourContractName # scaffold a contract

$ truffle create test YourTestName # scaffold a test

===========================================================================

下载公用标准库，进入当前项目目录：cd myprojDir

> npm install zeppelin-solidity

新增的node\_modules目录就是库文件。每个项目单独设置管理标准库。

=================================================================

http Port: 8545 接受部署的开放端口；

1、**truffle init**生成目录与文件说明：

1. contracts # 智能合约目录
2. migrations # 发布脚本目录
3. test # 存放测试文件的目录
4. truffle.js # Truffle的配置文件

强制全部文件重编译：

2、**truffle compile** --compile-all

合约的迁移（测试部署？）：truffle migrate

指定合约使用的配置的网络设置(truffle-config.js配置live)：

3、部署合约，部署账户为合约拥有者（先解锁账户，而且必须是第一账户！）

**truffle migrate** --network **live** --reset

合约的部署执行（可以反复运行合约）：

t**ruffle deploy --reset**

注意合约执行包括save阶段，都需要挖矿支持，合约部署必须要解锁账户即geth端：

personal.unlockAccount(“……”)

而且每次合约执行完成，账户都会再次被锁定；即每次执行合约都要解锁账户！

参数migrate成功执行一次后，就不会再次执行合约，需要用deploy参数执行交易。

（Question:编译不报错也不执行合约，信息： Network up to date

***"If you are doing that the contracts are deployed in your private ganache network, and I guess if you try to deploy the same contract again it without changes it wont let you because you will deploy contract that is the same as the one you deployed before"***

And for correct redeploy like

大意就是：用migrate把合约成功部署上去了，再次migrate就不会重复部署了）

即使同一内容合约每部署(deploy)一次，就相当于交给miner生成一个区块，不可更改，挂在了链上，每次部署的hash地址都是独一无二的。

甚至可以部署到外测试网（？）：

truffle migrate --reset --network ropsten

===========================================================================

多个合约部署：

重新修改一下 2\_deploy\_contracts.js 让它可以完成 InfoManager 的部署：

var Storage = artifacts.require("./Storage.sol");

var InfoManager = artifacts.require("./InfoManager.sol");

module.exports = function(deployer) {

// 部署 Storage

deployer.deploy(Storage)

// 等待、直到合约部署完成

.then(() => Storage.deployed())

// 传递 Storage 合约地址，部署 InfoManager 合约

.then(() => deployer.deploy(InfoManager, Storage.address));

}

部署的语法是:

...

deployer.deploy(`ContractName`, [`constructor params`]) // 返回一个 promise

...

因为 deploy(...) 返回一个 promise， 我们可以按自己喜欢的方式处理它，不要注意的是在部署文件里，还不支持 async 。

====================================================================

进入truffle命令行：

**$ truffle console**

truffle(development)>

执行如下命令(注意命令结束加";")：

> var contract;

undefined

> Greeter.deployed().then(function(instance){<!-- -->contract= instance;});

undefined

> contract.helloWorld()

详细信息列表：

> web3.eth.accounts

、执行js文件：

> exec gas\_estimates.js

------------------------------------------------------------------------------------------------------------

、试出来gas limit最大是11,000,00到13，500，000！超过不会通过检查！

------------------------------------------------------------------------------------------------------------------------

**、交易无法返回结果给web3.js**. 注意这一点！refundTicket函数会返回一个布尔值，但是这在测试中无法检查。因为这个方法是一个交易函数（会改变合约内数据或是发送以太币的调用），而web3.js得到的交易运行结果是一个交易哈希（如果打印出来是一个长长的十六进制/怪怪的字符串）。既然如此为什么还要让refundTicket返回一个值？因为在Solidity合约内可以读到这个返回值，例如当另一个合约调用refundTicket()的时候。也就是说**Solidity合约可以读取交易运行的返回值，而web3.js不行**。另一方面，**在web3.js中你可以用事件机制**（Event, 下文会解释）来监控交易运行，而合约不行。合约也无法通过call()来检查交易是否修改了合约内变量的值。

https://blog.51cto.com/zhangxueliang/3119979

==============================================================================

contract MetaCoin {//合约样例

function sendCoin(address receiver, uint amount) public returns(bool sufficient) {……

控制台带参数调用合约内函数：

truffle(develop)> **let instance = await MetaCoin.deployed()**
truffle(develop)> instance

、简单的代币转账成功：

truffle(development)> instance**.transferCZT(accounts[0],accounts[1],100,{from: accounts[0]})**

、简单挖币也成功：

instance.**customerMint**({from:accounts[0],gas:90000})

truffle(develop)> let accounts = await web3.eth.getAccounts()
truffle(develop)> instance.sendCoin(accounts[1], 10, {from: accounts[0]})

我用的是：

instance.mint(accounts[2],50000,{from:accounts[0]})

理论上这样带参数调用合约函数应该没问题，这是官方文档推荐的，实际却报错：

Uncaught Error: Returned error: execution reverted

at **evalmachine**.<anonymous>

at sigintHandlersWrap (node:vm:268:12)

at Script.runInContext (node:vm:137:14)

at runScript (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/core/lib/console.js:364:1)

at Console.interpret (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/core/lib/console.js:379:1)

、另外一种调用合约函数的方法：

MetaCoin**.deployed().then(function(instance)** **{return instance.sendCoin(a1, 100);}**).then(function(value) {console.log(value);});

==============================================================

truffle控制台帮助命令：

truffle(development)>

AbortController AbortSignal AggregateError Array ArrayBuffer

Atomics BasicToken BigInt BigInt64Array BigUint64Array

Boolean Buffer BurnableToken CappedToken CztToken

DOMException DataView Date ERC20 ERC20Basic

Error EvalError Event EventTarget FinalizationRegistry

Float32Array Float64Array Function Infinity Int16Array

Int32Array Int8Array Intl JSON Map

Math MessageChannel MessageEvent MessagePort Migrations

MintableToken NaN Number Object Ownable

Promise Proxy RangeError ReferenceError Reflect

RegExp SafeMath Set SharedArrayBuffer StandardToken

String Symbol SyntaxError TextDecoder TextEncoder

TypeError URIError URL URLSearchParams Uint16Array

Uint32Array Uint8Array Uint8ClampedArray WeakMap WeakRef

WeakSet WebAssembly \_ \_error accounts

assert async\_hooks atob btoa buffer

child\_process clearImmediate clearInterval clearTimeout cluster

console constants crypto decodeURI decodeURIComponent

dgram diagnostics\_channel dns domain encodeURI

encodeURIComponent escape eval events fs

global globalThis http http2 https

inspector interfaceAdapter isFinite isNaN module

net os parseFloat parseInt path

perf\_hooks performance process punycode querystring

queueMicrotask readline regeneratorRuntime repl require

setImmediate setInterval setTimeout stream string\_decoder

structuredClone sys timers tls trace\_events

tty undefined unescape url util

v8 vm web3 worker\_threads zlib

\_\_proto\_\_ constructor hasOwnProperty isPrototypeOf propertyIsEnumerable

toLocaleString toString valueOf

======================================================================================

编译Etheroll出错：

pragma solidity >=0.4.18;

ParserError: Expected type name

--> project:/contracts/Etheroll.sol:59:27:

|

59 | function setProofType(byte \_proofType);

| ^^^^

这个truffle版本不支持**byte**数据类型是非常意外的：

solc, the solidity compiler commandline interface

Version: **0.8.11**+commit.d7f03943.Linux.g++

----------------------------------------------------------------------------

ParserError: Source file requires different compiler version (current compiler is 0.8.10+commit.fc410830.Emscripten.clang) - note that nightly builds are considered to be strictly less than the released version

--> zeppelin-solidity/contracts/**math/SafeMath.sol:**1:1:

|

1 | pragma solidity ^0.4.24;

| ^^^^^^^^^^^^^^^^^^^^^^^^

==============================================================

TypeError: Contract "IERC721" ***should be marked as abstract.***

--> /Users/delete/transfer-proxy/contracts/TransferProxy.sol:7:1:

|

7 | contract IERC721 is IERC165 {

| ^ (Relevant source part starts here and spans across multiple lines).

Note: Missing implementation:

--> /Users/delete/transfer-proxy/contracts/TransferProxy.sol:16:5:

|

16 | function approve(address to, uint256 tokenId) public;

| ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Note: Missing implementation:

--> /Users/delete/transfer-proxy/contracts/TransferProxy.sol:12:5:

|

12 | function balanceOf(address owner) public view returns (uint256 balance);

| ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Answer 1

Abstract contracts were introduced in Solidity 0.6. As in many other programming languages, an abstract class allows to define an interface with partial implementation.

So you can mark your contract as abstract

**abstract contract** IERC721 **is** IERC165 {

And also mark your functions without implementations as external virtual (instead of public).

function balanceOf(address owner) external virtual view returns (uint256 balance);

// ... etc

Because your IERC721 does not implement any function, it can be also marked as interface (instead of abstract)

interface IERC721 is IERC165 {

This will also give you a warning on the redundant virtual modifier, so you can remove it as well (but the visibility modifier needs to stay external)

function balanceOf(address owner) external view returns (uint256 balance);

Share

Follow

answered May 9 at 19:54

----------------------------------------------------------------------------------------------------------------------------

Couple of issues

1. Constructor public () - Warning: Visibility for constructor is
   ignored. If you want the contract to be non-deployable, making it
   "abstract" is sufficient.
2. function getOwner() - TypeError: Overriding function is missing
   "**override**" specifier. --> MoneyPaigerToken.sol:365:3: | 365 |
   function getOwner

Not sure what is missing here.

Regards Sam

2 Answers

**Constructor public () - Warning: Visibility for constructor is
ignored. If you want the contract to be non-deployable, making it
"abstract" is sufficient.**

The warning message says it all. You can safely remove the public visibility modifier because it's ignored anyway.

If you marked the BEP20Token contract abstract, you would need to have a child contract inheriting from it, could not deploy the BEP20Token itself, but would have to deploy the child contract. Which is not what you want in this case.

constructor() { // remove the `public` modifier

**function getOwner() - TypeError: Overriding function is missing
"override" specifier. --> MoneyPaigerToken.sol:365:3: | 365 |
function getOwner**

Using interfaces in Solidity is not the same as in most other
languages. Using expressions from other languages, there's no keyword
for implements (an interface). There's just extends (a parent class) - in Solidity called simply is (a child of a class).

Interfaces are mostly used when you're interacting with other contracts (deployed on a different address). Example:

function getOtherTokenDecimals(address tokenAddress) external returns (uint8) {

IBEP20 tokenInstance = IBEP20(tokenAddress);

return tokenInstance.decimals();

}

As I wrote earlier, there's currently no expression for a contract
implementing an interface. Just for extending a class. So your BEP20Token is IBEP20 extends the (interface) class and because the function getOwner() is already defined in the parent, you need to explicitly state that you want to override by using the override modifier.

function getOwner() external override view returns (address) { // add the `override` modifier

However, my approach instead of explicitly overriding the functions would be to move the events definitions to your BEP20Token, remove the interface definition and its inheritance because it's not used anywhere else in the code.

contract BEP20Token is Context, Ownable { // remove the `IBEP20` inheritance

// move the event definitions here

event Transfer(address indexed from, address indexed to, uint256 value);

event Approval(address indexed owner, address indexed spender, uint256 value);

You can also remove the Context from the list of BEP20Token parents, because Ownable is also inheriting from Context.

contract BEP20Token is Ownable { // remove the `Context` parent

Share Follow edited May 11 at 13:28 answered May 11 at 11:32
---------------------------------------------------------------------------------------------------------------------------

TypeError: Trying to override non-virtual function. Did you forget to add "virtual"?

function unpause() public **virtual** {

function \_beforeTokenTransfer(address from, address to, uint256 amount) internal **virtual** override(ERC20PresetMinterPauserUpgradeSafe) {

function num() public pure override returns(uint){

**继承**

1. 用is来实现合约继承
2. 继承的子合约会继承父合约所有非private的东西，包括modifier

Contract A{…}

Contract B is A {…}

1. 如果要override一个函数，在**被覆盖的函数声明时必须加virtual**关键字，**在覆盖的函数后加override**
2. 如果希望能被在此override，也要加virtual关键字
3. 函数默认调用最衍生的类定义的函数
4. 如果要调用父类的函数，使用super.f()
5. 函数override的时候必须保持同样的可见性，或从external变成public
6. 另一个参数只能变得更加严格，即non-payable（也就是无参数）可以被override为view，view可以变成pure。payable是特殊的，必须保持为payable

contract Base{

function foo() virtual external view{}

}

contract Middle is Base{}

contract Inherited is Middle{

function foo() virtual override public pure{}

}

external -> public view -> pure 符合override后的函数修饰

**Abstract Contracts**

Abstract contracts是未实现的Contract。可以用来做父类

abstract contract A{

function num() public pure virtual returns(uint);

}

contract B is A{

function num() public pure override returns(uint){

return 0;

}

}

--------------------------------------------------------------------------------------------------------------

.**sub()** in this context is almost certainly a function of the SafeMath library.

In Solidity 0.8+ you don't need to use SafeMath anymore, because the integer underflow/overflow check is performed on a lower level.

So **you can safely replace**

uint256 newViperId = vipers.push(newViper).sub(1);

with

vipers.push(newViper);

uint256 newViperId = vipers.length - 1;

Share Follow

edited Mar 13 at 20:09 answered Mar 13 at 20:02

========================================================================

、下面这个问题的根本原因是geth没有开启http连接，所以合约无法部署上去，加上重开geth可：

leon@ubuntu:~/ethmchain/consortium\_blockchain/trufTest01$ truffle compile

Compiling your contracts...

===========================

> Compiling ./contracts/dnc.sol

> Compilation warnings encountered:

Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.

--> project:/contracts/dnc.sol

> Artifacts written to /home/leon/ethmchain/consortium\_blockchain/trufTest01/build/contracts

> Compiled successfully using:

- solc: 0.8.10+commit.fc410830.Emscripten.clang

leon@ubuntu:~/ethmchain/consortium\_blockchain/trufTest01$ truffle migrate

Compiling your contracts...

===========================

> Compiling ./contracts/Migrations.sol

> Artifacts written to /home/leon/ethmchain/consortium\_blockchain/trufTest01/build/contracts

> Compiled successfully using:

- solc: 0.8.10+commit.fc410830.Emscripten.clang

> **Something went wrong while attempting to connect to the network at** http://127.0.0.1:7545. Check your network configuration.

Could not connect to your Ethereum client with the following parameters:

- host > 127.0.0.1

- port > 7545

- network\_id > 5777

Please check that your Ethereum client:

- is running

- is accepting RPC connections (i.e., "--rpc" or "--http" option is used in geth)

- is accessible over the network

- is properly configured in your Truffle configuration file (truffle-config.js)

Truffle v5.4.25 (core: 5.4.25)

Node v17.3.0

leon@ubuntu:~/ethmchain/consortium\_blockchain/trufTest01$

**CruzMolina commented on 13 Jun 2020**

|  |
| --- |
| Ah, you need to nest the development field in your config within a networks object. module.exports = { // See <http://truffleframework.com/docs/advanced/configuration> // to customize your Truffle configuration! networks: { development: { host: '127.0.0.1', port: 23889, network\_id: '\*', // eslint-disable-line camelcase from: '0x7926223070547d2d15b2ef5e7383e541c338ffe9', gasPrice: '0x64', } }}; |

--------------------------------------------------------------------------------------------------------------------------------

geth开放的端口确实是8545：

development: {

host: "127.0.0.1", // Localhost (default: none)

port: **8545**, // Standard Ethereum port (default: none)

network\_id: "\*" // Any network (default: none)

}///,

leon@ubuntu:~/ethmchain/consortium\_blockchain/trufTest01$ truffle migrate --network development --reset

Compiling your contracts...

===========================

> Everything is up to date, there is nothing to compile.

Starting migrations...

======================

> Network name: 'development'

> Network id: 84348

> Block gas limit: 44606115 (0x2a8a2a3)

1\_initial\_migration.js

======================

/usr/local/lib/node\_modules/truffle/build/459.bundled.js:27159

throw new Error("Could not find **artifacts** for " + import\_path + " from any sources");

^

Error: Could not find artifacts for dnc from any sources

at Resolver.require (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/resolver/dist/lib/resolver.js:61:1)

at Object.require (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/index.js:172:1)

at ResolverIntercept.require (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/ResolverIntercept.js:22:1)

at /home/leon/ethmchain/consortium\_blockchain/trufTest01/migrations/1\_initial\_migration.js:1:30

at Script.runInContext (node:vm:139:12)

at Script.runInNewContext (node:vm:144:17)

at Object.file (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/require/require.js:94:1)

at Migration.\_load (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/Migration.js:44:1)

at processTicksAndRejections (node:internal/process/task\_queues:96:5)

at Migration.run (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/Migration.js:217:1)

at Object.runMigrations (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/index.js:150:1)

at Object.runFrom (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/index.js:110:1)

at Object.runAll (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/index.js:114:1)

at Object.run (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/index.js:79:1)

at runMigrations (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/core/lib/commands/migrate/run.js:76:1)

at Object.module.exports [as run] (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/core/lib/commands/migrate/run.js:44:1)

Truffle v5.4.25 (core: 5.4.25)

Node v17.3.0

leon@ubuntu:~/ethmchain/consortium\_blockchain/trufTest01$ ^C

、这个合约编译成功但是部署报错原因，不太直接：

Make sure you have the Migrations.sol and that the name of the contract is the same as the name of the file, that solved the issue for me.

就是缺少**Migrations.sol** 文件，我觉得没用手动把它删除了！

Migrations文件是使用truffle部署智能合约时必要的文件，其内容一般不会变。

===========================================================================

2021-12-31

第三个实例部署第二个deploy出错：

leon@ubuntu:~/ethmchain/consortium\_blockchain/trufTest03$ truffle migrate

Compiling your contracts...

===========================

> Everything is up to date, there is nothing to compile.

Starting migrations...

======================

> Network name: 'development'

> Network id: 84348

> Block gas limit: 10475443 (0x9fd7b3)

1\_initial\_migration.js

======================

Deploying 'Migrations' //这个合同只是标准辅助

----------------------

> transaction hash: 0xdd6452e10c524a3b019866b1a5ab13536b1b52e39dc6e3c0e96ba66f9bab51ba

> Blocks: 0 Seconds: 16

> contract address: 0x881bBF99B62BFAc6a66a3d9A9FffeBDdCdD6a1E6

> block number: 4667

> block timestamp: 1640940963

> account: 0xCED27893eeA69b3fF1E57b5Ce40F19662e9296F5

> balance: 7068

> gas used: 290582 (0x46f16)

> gas price: 1 gwei

> value sent: 0 ETH

> total cost: 0.000290582 ETH

> Saving migration to chain.

> Saving artifacts

-------------------------------------

> Total cost: 0.000290582 ETH

2\_deploy\_contracts.js

=====================

/usr/local/lib/node\_modules/truffle/build/459.bundled.js:27159

throw new Error("Could not find artifacts for " + import\_path + " from any sources");

^

Error: Could not find artifacts for **dnc** from any sources

at Resolver.require (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/resolver/dist/lib/resolver.js:61:1)

at Object.require (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/index.js:172:1)

at ResolverIntercept.require (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/ResolverIntercept.js:22:1)

at /home/leon/ethmchain/consortium\_blockchain/trufTest03/migrations/2\_deploy\_contracts.js:1:23

at Script.runInContext (node:vm:139:12)

at Script.runInNewContext (node:vm:144:17)

at Object.file (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/require/require.js:94:1)

at Migration.\_load (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/Migration.js:44:1)

at processTicksAndRejections (node:internal/process/task\_queues:96:5)

at Migration.run (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/Migration.js:217:1)

at Object.runMigrations (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/index.js:150:1)

at Object.runFrom (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/index.js:110:1)

at Object.run (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/migrate/index.js:87:1)

at runMigrations (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/core/lib/commands/migrate/run.js:76:1)

at Object.module.exports [as run] (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/core/lib/commands/migrate/run.js:44:1)

at Command.run (/usr/local/lib/node\_modules/truffle/build/webpack:/packages/core/lib/command.js:189:1)

Truffle v5.4.25 (core: 5.4.25)

Node v17.3.0

leon@ubuntu:~/ethmchain/consortium\_blockchain/trufTest03$

、这个错误，其实是dnc.sol里面，没有任何dnc创造物，改2\_deploy\_contract.js：

var DncToken = artifacts.require("./DncToken.sol");

module.exports = function (deployer) {

deployer.deploy(DncToken);

即可成功部署（出了DncToken之外的其他编译后json文件是否部署就不知道了）！

Truffle console调试命令：  
1、调整Token：

测试的函数：

**let instance = await CztToken.deployed()**

instance.playerMint(1000,{from:accounts[1],gas:500000})

instance.betForGame(1000,{from:accounts[1],gas:500000})

instance.getPlayerCount({from:accounts[1]})

instance.getBetCount({from:accounts[1]})

instance.checkMinter({from:accounts[1]})

instance.getApporintCZT(accounts[1],0,{from:accounts[1],gas:90000})

instance.startGameToContract({from:accounts[1],gas:5000000})

instance.startGameToOtherOne({from:accounts[0],gas:5000000})

instance.balanceOf(accounts[1])

instance.getPlayerBet(accounts[2])

instance.TestNonce({from:accounts[1],gas:5000000})

let rst01=await instance.TestNoParam({from:accounts[1],gas:5000000})

查看结果返回：

**> let rst01 =await** instance.startGameToContract({from:accounts[2],gas:5000000})

> rst01.logs.at()

> rst01.logs.at().args //查看返回数组详情

> rst04.receipt.rawLogs //查看receipt 的rawLogs详情

1. 调试NFT：

**let instNFT = await** Czt01NFT**.deployed()**

**let instMakt=await** CztMarket**.deployed()**

(注意命令行不能有多的空格:<)

instNFT.mintOneToken("http://192.168.11.120:8080/ipfs/QmNz9YTrNftUfYMPxymRh55A5EtcapRuGcBLTMMcMpwqmb")