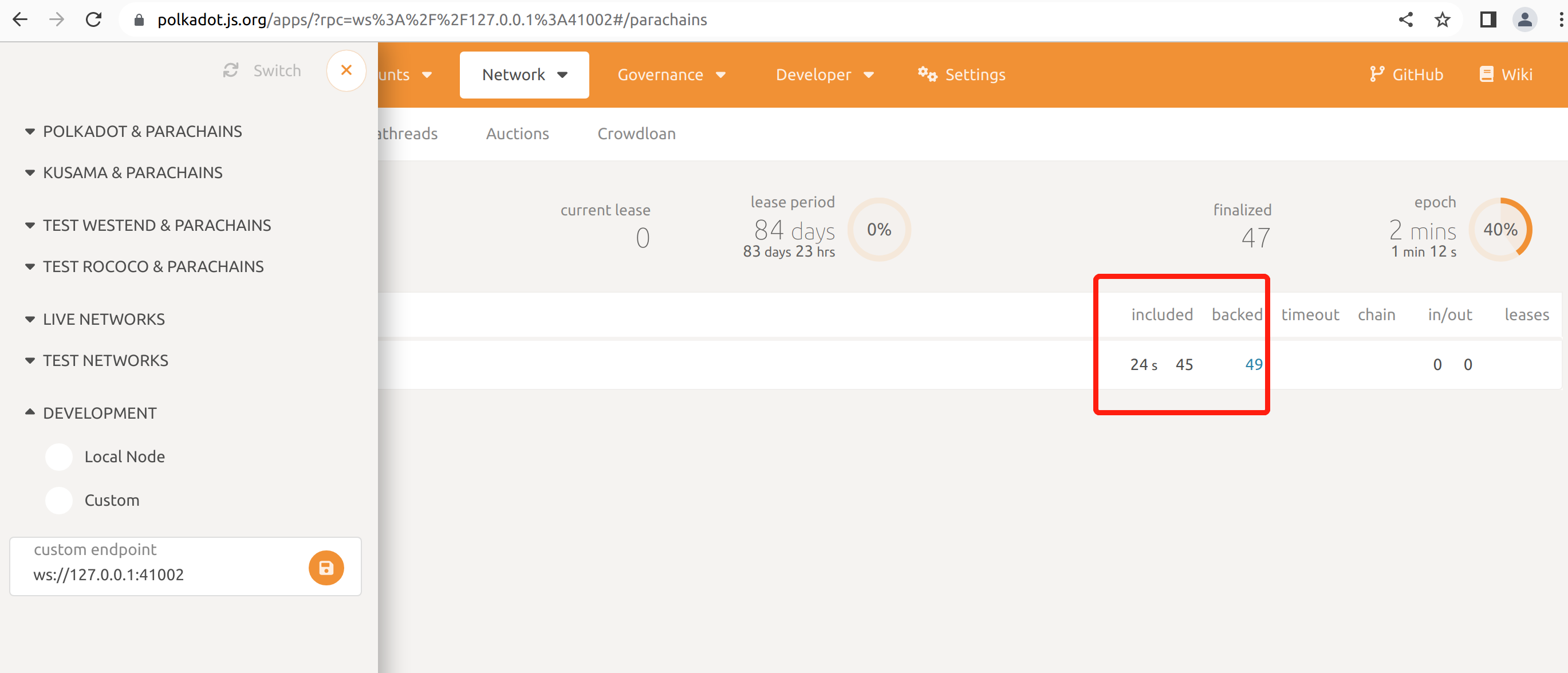
**测试代码文件**

src/test包含了用于测试的代码文件，其中evm.js是测试代码，contract.sol是测试用的合约，MyToken.json是合约编译出来的文件。

**测试准备**

先检查collator平行链是否正常出块：



**测试配置**

在evm.js上部定义了一些配置，包括：

* PROVIDER：riochain的ws地址，其中端口必须是collator（非relay）的docker映射端口。比如collator-node-01的端口是43001
* SENDER：riochain测试账户的uri，默认提供了从"//...//01"到"//...//40" 等账号（05是sudo账号）。其中的"..."是安装时提供的助记词，即：export MNEMONIC="..."
* test\_eth\_address：测试用的eth账户
* test\_eth\_privatekey：和test\_eth\_address对应的private key

**账号对应关系**

一个Riochain账号（即substrate SS58地址）对应了一个EVM的H160地址，一个H160地址对应了一个substrate H256地址，具体参见evm.js的转换函数。地址转换原理：

<https://iconosphereio.medium.com/substrate-evm-address-39bfa92f875c>

EVM pallet的H160转换实现参见：

<https://github.com/paritytech/frontier/blob/master/frame/evm/src/lib.rs>

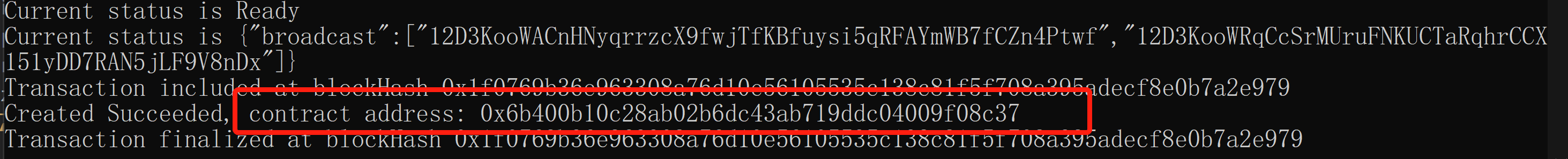
的EnsureAddressTruncated。

**测试流程**

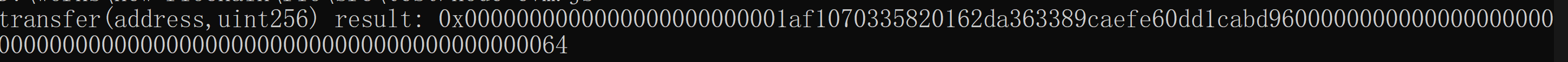
1. 给SENDER对应的H256地址转账，即transfer(PROVIDER, SENDER, SENDER)。注：转账amount不能小于existential deposit。
2. 给test\_eth\_address对应的H256地址转账，即

transfer(PROVIDER, SENDER, test\_eth\_address)

1. 测试evm.call方法，即call(PROVIDER, SENDER, test\_eth\_address, test\_call\_input)，其中test\_call\_input是预先计算的calldata.
2. 测试evm.withdraw方法，即withdraw(PROVIDER, SENDER)
3. 测试evm.create方法，即创建合约create(PROVIDER, SENDER, Json['bytecode'])，其中Json是合约的json文件。创建成功后打印部署的地址：



1. 根据以上合约地址测试合约方法（RPC调用），包括multiply、transfer、getbalance，合约调用者是test\_eth\_address。成功后结果：



其中result是合约里面emit的事件参数，格式是32bit\_Hex\_str的参数列表。

注：为方便测试，solidity代码里的owner也是test\_eth\_address

1. 其他RPC接口类似

**合约调用的错误**

※ gasLimit/gasPrice：大部分错误是这个配置不正确（过大或过小）

※ 如果出现InvalidTransaction::Custom()，错误定义在：

<https://paritytech.github.io/frontier/rustdocs/src/pallet_ethereum/lib.rs.html>

里面的TransactionValidationError

※ 查询合约调用结果getTransactionReceipt在代码中延迟了30秒，因为InBlock的时间不确定，可能会出现超时报错。

**测试结果**

以上步骤的测试均通过，能成功部署合约，并且调用合约的multiply、transfer、getbalance方法能返回正确结果。