

# BSN-DDC 网络部署 Solidity 合约快速上手指南

## 1 准备工作

solidity 相关知识文档：[Solidity 中文文档](#)、[Solidity 官方文档](#)

### 1.1 DDC 平台方

#### 1.1.1 获取 RPC 地址

登录 [DDC 门户](#)，如下图所示进入到【项目管理】菜单进行项目的创建，输入项目名称，选择对应框架，创建成功。列表查看项目 ID 进行替换获取对应链的 RPC 地址。**注意：如果要使用 remix + metaMask 部署合约不要开启项目 KEY，如用代码部署可开启项目 KEY，同时请求报文头 header 中增加 x-api-key:{项目 key 值}。另泰安链只能使用代码部署，可参考待补充。**

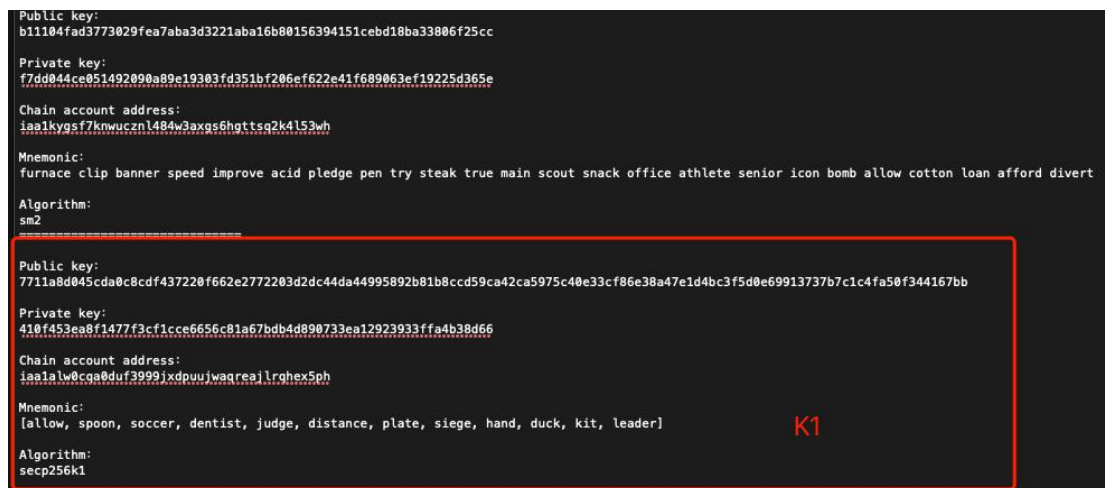
- 文昌链 RPC 地址：[https://opbningxia.bsngate.com:18602/api/\[项目 ID\]/evmrpc](https://opbningxia.bsngate.com:18602/api/[项目 ID]/evmrpc)  
链 ID：1223
- 武汉链 RPC 地址：[https://opbningxia.bsngate.com:18602/api/\[项目 ID\]/rpc](https://opbningxia.bsngate.com:18602/api/[项目 ID]/rpc) 链 ID：5555
- 泰安链 RPC 地址：[https://opbningxia.bsngate.com:18602/api/\[项目 ID\]/rpc](https://opbningxia.bsngate.com:18602/api/[项目 ID]/rpc) 链 ID：



### 1.1.2 创建链账户并进行能量值充值

登录 [DDC 门户](#)，进入到【链账户管理-链账户】菜单进行链账户的创建和能量值充值。

- 开放联盟链：选择要创建的框架类型
- 链账户名称：输入链账户名称
- 使用官方 DDC：选择是和否都可以正常部署合约，如果选择是可使用 [BSN 官方业务合约](#)
- 链账户创建方式：建议都使用上次链账户地址形式，点击右侧在线生成按钮，在线生成链账户，将生成的链账户地址（chain account address）填入就可以，注意：文昌链需要使用 K1 算法的链账户地址，如图所示



- 业务凭证：上传业务凭证，业务凭证在业务开通信息中下载，如图所示



链账户创建成功后，进行能量值充值即可，前期测试可以充值 1 块、1 毛都可以，部署合约大概几分钱。其他方式创建链账户请查看各链的快速上手指南的第六章创建链账户。

文昌链参考：[文昌链官方 DDC 快速上手指南](#)

泰安链参考：[泰安链官方 DDC 快速上手指南](#)

中移链参考：[中移链官方 DDC 快速上手指南](#)

## 1.2 算力中心方

### 1.2.1 获取 RPC 地址

登录算力中心用户门户，进入到【网络接入-网络接入管理】，即可看到算力中心支持的框架的 rpc 地址（**注意文昌链选 evmrpc**），如图所示：

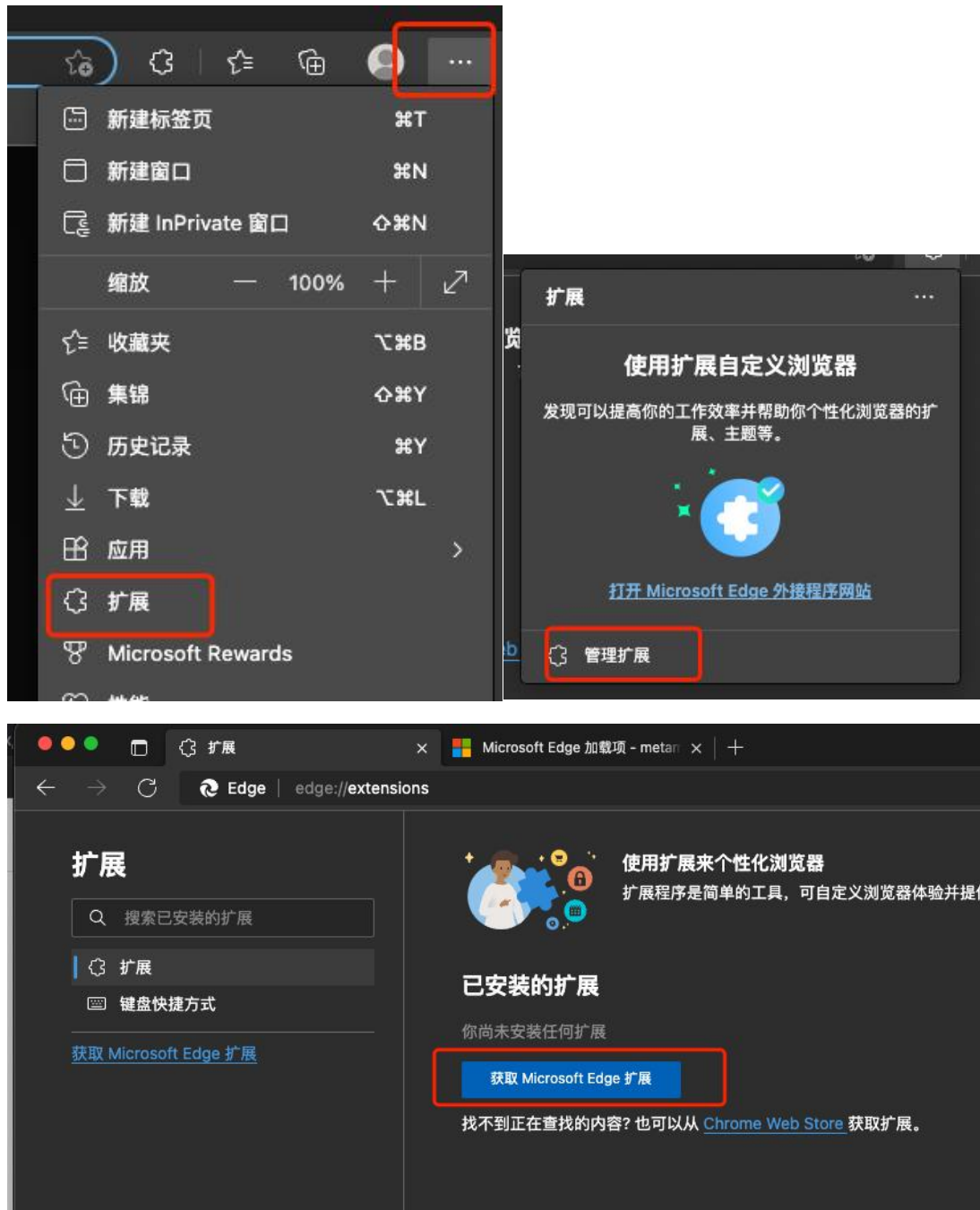
- 文昌链 ID: 1223
- 武汉链 ID: 5555
- 泰安链 ID: 1

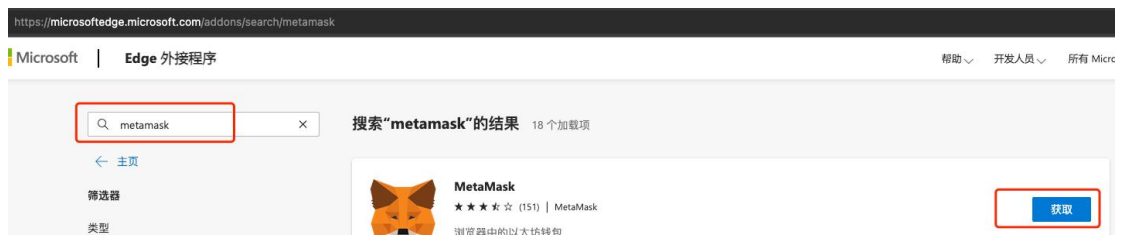


## 2.1 使用 metaMask 连接 BSN-DDC 网络

### 2.1.1 安装 metaMask

Chrome 谷歌浏览器、Firefox 火狐浏览器、Microsoft Edge 新版浏览器。都可以安装 MetaMask 插件，以 Microsoft Edge 为例，点击右侧【...】选择【扩展】-【管理扩展】-【获取 Microsoft Edge 扩展】，搜索 metaMask，获取扩展，添加成功，如图所示





## 2.1.2 导入已创建链账户

点击 metaMask 插件，【开始使用】–【我同意】–【导入钱包】，打开之前在线生成的链账户信息文件（PublicPrivateKeys.txt）里面有一个助记词（Mnemonic），将助记词里的每个单词粘贴进去，点击导入，如果长时间没反应刷新看下，应该是添加成功了，刷新出现一个输密码的界面输入刚导助记词填写。如下图所示



## 欢迎使用 MetaMask

将您连接到以太坊和去中心化网络。

我们很高兴见到您。

开始使用

 METAMASK



## 帮助我们改进 MetaMask

MetaMask 希望收集使用数据，以更好地了解我们的用户如何与扩展程序交互。这些数据将被用于持续改进我们产品和以太坊生态系统的可用性和用户体验。

MetaMask..

- ✓ 始终允许您通过“设置”选择退出
- ✓ 发送匿名化点击和页面浏览事件

- ✗ 永不 收集密钥、地址、交易记录、余额、哈希或任何个人信息
- ✗ 永不 收集您的完整 IP 地址
- ✗ 永不 为利益而出售您的数据，永远不会！

不，谢谢

我同意

这些数据是汇总的，因此，根据《通用数据保护条例》（欧盟）2016/679，这些数据是匿名的。有关我们隐私惯例的更多信息，请参见我们的 [隐私政策](#)。



METAMASK

## MetaMask 的新用户？



不，我已经有一个账户助记词  
使用账户助记词导入您的现有钱包

导入钱包



是的，让我们开始吧！  
这将创建新的钱包和账户助记词

创建钱包

```
PublicPrivateKeys (92).txt

Public key:
0x5083183bdfa8f4c0185b8e4d12098a3c84e61c940d86a19b0bd3d3b858f163f3916acf4cfa4ae91fdfe73d766847a0bf217
5811665153472ca5733a6f422ea63

Private key:
0xbe1b08c6e0e2e6a7e1158a96fe9912cee3710c12080a90c09119650fb6039740

Chain account address:
0xb20d75b501aa5cd5b2036d7eab77abf1736dd6b8

Mnemonic:
[between, rent, useless, damage, approve, same, acoustic, magic, task, disagree, spring, coil]

Algorithm:
secp256k1
```

## 使用账户助记词导入钱包

只有这个钱包上的第一个账户将自动加载。完成此流程后，若要添加额外的账户，点击下拉菜单，然后选择“创建账户”。

助记词

我有一个包含12个单词的助记词

您可以将整个助记词粘贴到任何字段中

1. ....

2. ....

3. ....

4. ....

5. ....

6. ....

7. ....

8. ....

9. ....

10. ....

11. ....

12. ....

新密码（至少 8 个字符）

.....

确认密码

.....

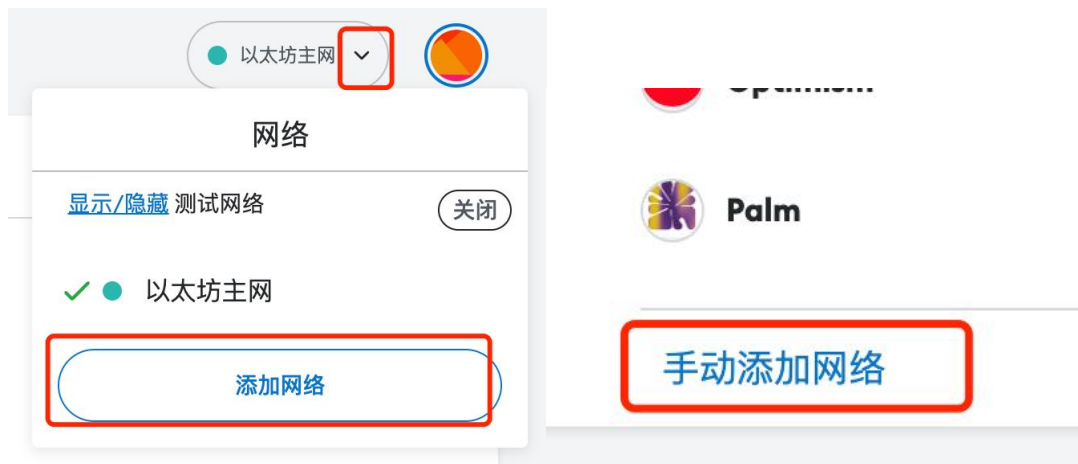
☒ 我已阅读并同意 [使用条款](#)

导入

### 2.1.2 添加网络

添加 BSN-DDC 网络至 metaMask。如下图所示：





RPC 地址在 [1.准备工作](#) 中查看，链 ID 也是，货币符号武汉链：OAS，文昌链：UGAS，添加成功如下图，如果是文昌链链账户里面有能量值这显示为 0 是正常的，是因为精度问题这显示不出来。

#### 网络 > 添加网络 > 手动添加网络

**ⓘ** 恶意网络提供商可能会谎报区块链的状态并记录您的网络活动。只添加您信任的自定义网络。

网络名称

武汉链

新的 RPC URL

https://opbningxia.bsngate.com:18602/api/b26bb1306f

链 ID ⓘ

5555

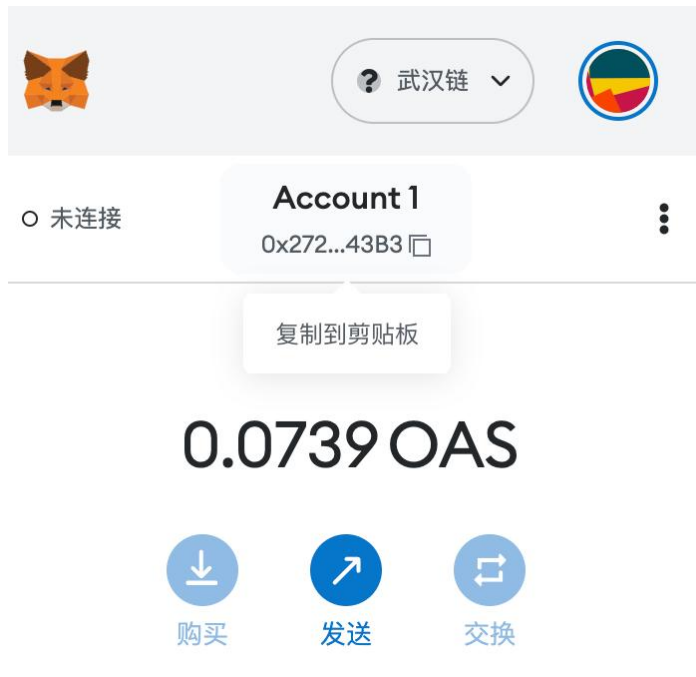
货币符号

OAS

区块浏览器 URL (可选)

取消

保存



1.

## 2.2 使用 Remix 部署调试合约

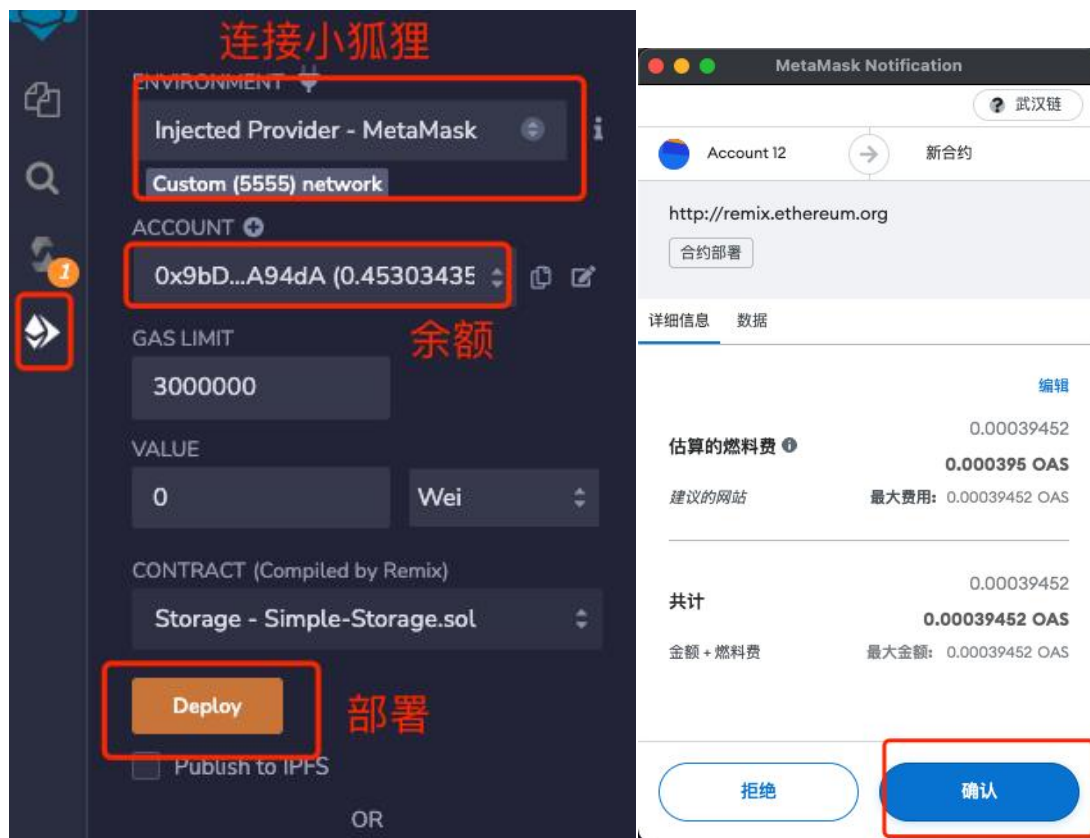
以第三类的简单存储合约为例，合约地址  
<https://github.com/BSN-DDC/Beginner-Smart-Contracts/tree/main/Simple-Storage>

### 2.2.1 部署合约

1. 打开 [remix](#), 上传合约 Simple-Storage.sol



文昌链需要在 metaMask 调整精度，详细看下面 4. 注意事项。



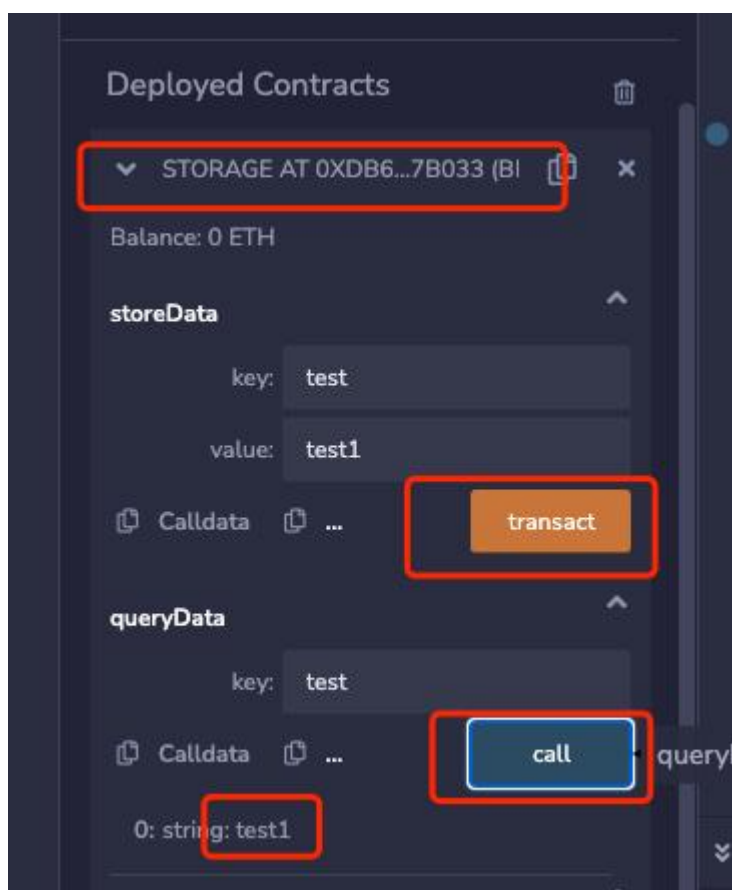
#### 4. 注意事项

如果是文昌链账户需要更改精度，否则不能部署成功，如下图所示，可以在 remix 的 account 中看到链账户能量值余额，然后部署合约的时候，在 metaMask 里点击【编辑】 - 将最大基本费用和优先费用改成 0.0000000001，然后点击【保存】，再点击第二张图的【确认】进行合约部署。



### 2.2.2 合约调用

点击刚部署好的合约地址前面的 【V】，可以看到合约里的所有方法，在这就可以进行调试了。首先调用 storeData 方法，传入 key: test, value: test1, 点击 transact, metaMask 钱包里确认，等待交易完成，完成后，调用 queryData 传入 key: test, 可以看到返回的事 刚传入的值 test1。



## 3 使用代码部署调用合约

### 3.1 泰安链

我们提供 Java 语言的示例代码供开发者参考。简单存储合约地址

<https://github.com/BSN-DDC/Beginner-Smart-Contracts/tree/main/Simple-Storage>

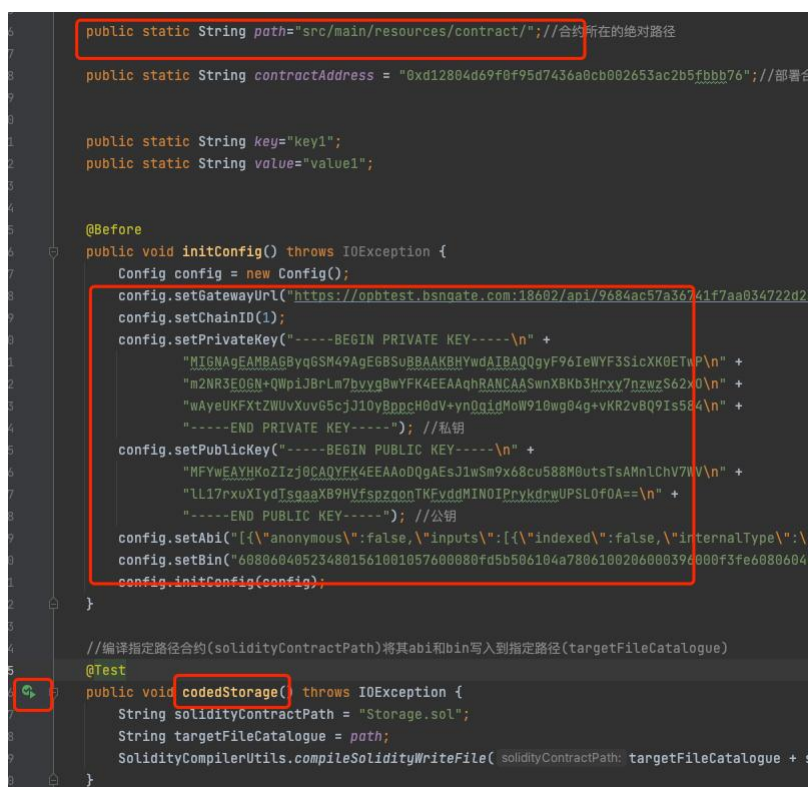
1. 更改 StroageTest 中的以下参数：

- privateKey: 在 [1. 准备工作](#)，创建链账户中查看
- publicKey: 在 [1. 准备工作](#)，创建链账户中查看
- path: 合约.sol 文件所在的绝对路径，**注意如果使用简单存储合约将合约名称改为**

## Storage.sol

- gatewayUrl: 即 RPC 地址在 [1. 准备工作](#), 获取 rpc 地址中查看
- chainID: 1

填写以上参数后运行 codedStorage 方法, 生成合约 abi 和 bin, 填写进去



```
public static String path="src/main/resources/contract/";//合约所在的绝对路径

public static String contractAddress = "0xd12804d69f0f95d7436a0cb002653ac2b5fbbb76";//部署名

public static String key="key1";
public static String value="value1";

@Before
public void initConfig() throws IOException {
    Config config = new Config();
    config.setGatewayUrl("https://opbtest.bsnqgate.com:18602/api/9684ac57a36741f7aa034722d2");
    config.setChainID(1);
    config.setPrivateKey("-----BEGIN PRIVATE KEY-----\n" +
        "MIGNAgEAMBA6B8yq6SM49AgE6BSu8BAAKBHYwdAIBAQQgyF96IeWYF3S1cXK0ETwP\n" +
        "m2NR3E0GN+QWpiJBrLm7pvygBwYFK4EEAAqhRANCAASwnXBKb3Hrxy7nzwzS62x0\n" +
        "wAyeUkFXtZWUvXuvG5cjJ10yBppcH0dV+yn0qidMoW910wg84g+vKR2vBQ9Is564\n" +
        "-----END PRIVATE KEY-----"); //私钥
    config.setPublicKey("-----BEGIN PUBLIC KEY-----\n" +
        "MFYwEAYHKoZIzj0CAQYFK4EEAAoDQgAEsJ1wSm9x68cu588M0utsTsAMnLChV7WV\n" +
        "lL17rxuXIydTsgaaX89HYfsgpZqenTKFvdMIN0IPrykdrwUPL0f0A==\n" +
        "-----END PUBLIC KEY-----"); //公钥
    config.setAbi("{\"anonymous\":false,\"inputs\":[{\"indexed\":false,\"internalType\":\"\n" +
        "config.setBin("608060405234801561001057600080fd5b506104a780610020600039e800f3fe60806040\n" +
        "config.initConfig(config);
    }

//编译指定路径合约(solidityContractPath)将其abi和bin写入到指定路径(targetFileCatalogue)
@Test
public void codedStorage() throws IOException {
    String solidityContractPath = "Storage.sol";
    String targetFileCatalogue = path;
    SolidityCompilerUtils.compileSolidityAndWriteFile(solidityContractPath, targetFileCatalogue + s
}
```

2. 运行 deploy 方法, 进行合约部署, 部署成功, 返回合约地址
3. 将获得的合约地址更新代码里的 contractAddress 字段, 然后运行 storeData 方法, 传入 key, value, 并打印出合约事件
4. 然后运行 queryData 方法, 传入 key, 返回 value

## 3.2 武汉链、文昌链

我们提供两种语言的示例代码供开发者参考。Java Demo: Go Demo, 简单存储合约地址

<https://github.com/BSN-DDC/Beginner-Smart-Contracts/tree/main/Simple-Storage>

ge

### 3.2.1 Java

1. 更改 StroageTest 中的以下参数：

- privateKey: 在 [1. 准备工作](#)，创建链账户中查看
- path: 合约.sol 文件所在的绝对路径，**注意如果使用简单存储合约将合约名称改为 Storage.sol**
- gatewayUrl: [https://opbningxia.bsngate.com:18602/api/\[项目 ID\]/evmrpc](https://opbningxia.bsngate.com:18602/api/[项目 ID]/evmrpc)，项目 ID 在在 [1. 准备工作](#)，获取 RPC 地址查看
- chainID: 武汉链：5555、文昌链：1223

填写以上参数后运行 codedStorage 方法，生成合约 abi 和 bin，填写进去

### 3.2.2 Go

[demo 地址](#)，本 demo 已经生成了 storage.go 文件。用户自己生成可以使用 solc 编译合约，获得 abi、bin，使用 abigen，生成 go 文件，然后就可以调用.go 文件进行合约部署和调用，这步网上有教程大家可以参考网上的教程。

1. 修改 storage\_test.go 中的参数

- NodeUrl: [https://opbningxia.bsngate.com:18602/api/\[项目 ID\]/evmrpc](https://opbningxia.bsngate.com:18602/api/[项目 ID]/evmrpc)，项目 ID 在在 [1. 准备工作](#)，获取 RPC 地址查看
- NodeWsUrl : [wss://opbningxia.bsngate.com:18602/api/\[项目 ID\]/ws](wss://opbningxia.bsngate.com:18602/api/[项目 ID]/ws)，项目 ID 在在 [1. 准备工作](#)，获取 RPC 地址查看
- PrivateKey: 在 [1. 准备工作](#)，创建链账户中查看

2. 部署合约可参考以下方法：





```

func TestDeployStorage(t *testing.T) {
    log.InitLog()
    cli, err := ethclient.Dial(NodeUrl)
    if err != nil {
        log.Logger.Error(err)
    }
    auth, err := eth.GenAuth(cli, PrivateKey)
    if err != nil {
        log.Logger.Error(err)
    }
    ContAddress, tx, _, err := storage.DeployStorage(auth, cli)
    if err != nil {
        log.Logger.Error(err)
    }
    fmt.Println("ContractAddress:", ContAddress)//部署后的合约地址
    fmt.Println("TxHash:", tx.Hash())
}

```

3. 运行 TestStorageData 进行 key, value 的传入, 运行 TestQueryData 获取 key 对应的 value 值, 运行 TestDeployStorage 方法获取 StoreData 事件的内容。
4. 合约事件可以通过 demo 中的 websocket 监听方式获取, 也可以通过交易 hash 获取, 参考以下代码。

```

func TestEventOption(t *testing.T) {
    log.InitLog()
    cli, err := ethclient.Dial(NodeUrl)
    if err != nil {
        log.Logger.Error(err)
    }
    receipt, err := cli.TransactionReceipt(context.Background(),
common.HexToHash("0x36aa2a692fbfc69d91462afcd570fbf26c15cd2575ac14751662fedabdbd11"))
    abi,err:=abi.JSON(strings.NewReader(storage.StorageMetaData.ABI))

    if err != nil {
        log.Logger.Error(err)
    }
    contractAddress := common.HexToAddress(Address)
    for _, vlog := range receipt.Logs {

```

```
switch vlog.Address {
case contractAddress:
    log.Logger.Error(err)
    switch vlog.Topics[0] {
    //StoreData 事件的签名
    case abi.Events["StoreData"].ID:
        data, err := abi.Events["StoreData"].Inputs.UnpackValues(vlog.Data)
        if err != nil {
            log.Logger.Error(err)
        }
        //打印监听到的参数
        fmt.Println("Key 值: ",data[0])
        fmt.Println("Value 值: ",data[1])
        fmt.Println("交易 hash",vlog.TxHash.String())
    }
}
}
```