Deploy Proxy Contract using a script

These walkthrough slides are related to the previous walkthough "ProxyContract" but can be done separately.

We are here going to deploy the proxy contract using a script rather than the UI interface.

Deploy using script

Let’s first load all the dependencies from github.

Run in the terminal:

```
remix.loadurl('
https://github.com/ethereum/remix-workshops/runningScript/proxyContract_AwardToken/loadAwardProxyDependencies.js
')
```

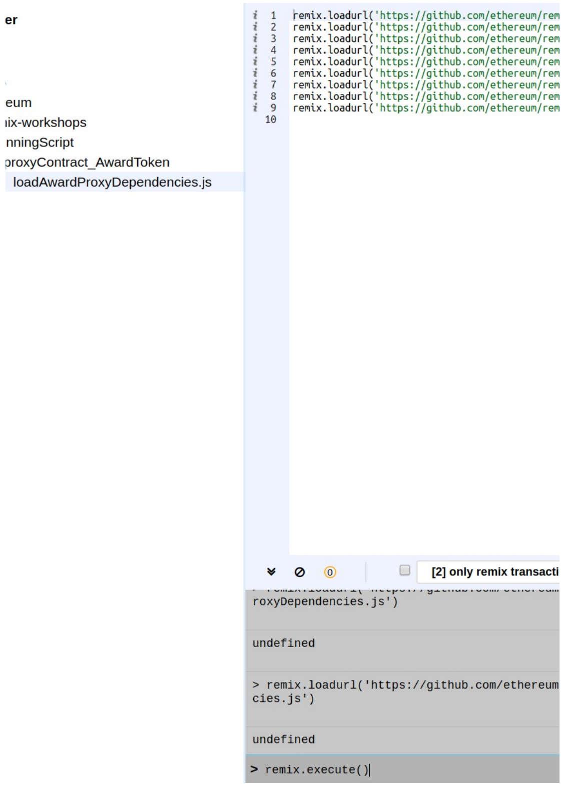
This script will be used to retrieve all the solidity contracts

Deploy using script

Then, select it in the explorer and run

`remix.execute()`

fold / unfold the github explorer and the contracts should be visible.



er

eum
ix-workshops
nningScript
proxyContract_AwardToken
loadAwardProxyDependencies.js

```
remix.loadurl('https://github.com/ethereum/rem
remix.loadurl('https://github.com/ethereum/rem
remix.loadurl('https://github.com/ethereum/rem
remix.loadurl('https://github.com/ethereum/rem
remix.loadurl('https://github.com/ethereum/rem
remix.loadurl('https://github.com/ethereum/rem
remix.loadurl('https://github.com/ethereum/rem
remix.loadurl('https://github.com/ethereum/rem
remix.loadurl('https://github.com/ethereum/rem
```

```
roxyDependencies.js')

undefined

> remix.loadurl('https://github.com/ethereum
cies.js')

undefined

> remix.execute()
```

Deploy

Then we load the deployment scripts:

run in the terminal:

```
remix.loadurl('
https://github.com/ethereum/remix-workshops/runningScript/proxyContract_AwardToken/global.js
')
```

```
remix.execute('github/ethereum/remix-workshops/runningScript/proxyContract_AwardToken/global.js')
```

The above is 1) loading the script “global.js” from github and 2) executing it in a sandbox JavaScript runtime.

That’s a generic JavaScript script which setup all you need for logging the deployment result in a file and deploy a contract.

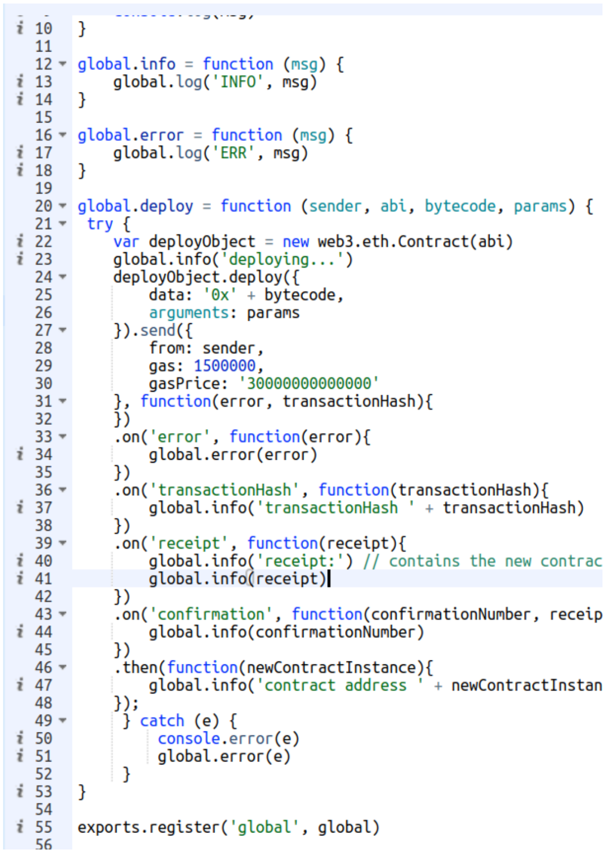
Deploy

See the last statement:

exports.register(‘global’, global)

this allows to use the “global” variable

within other execution context.



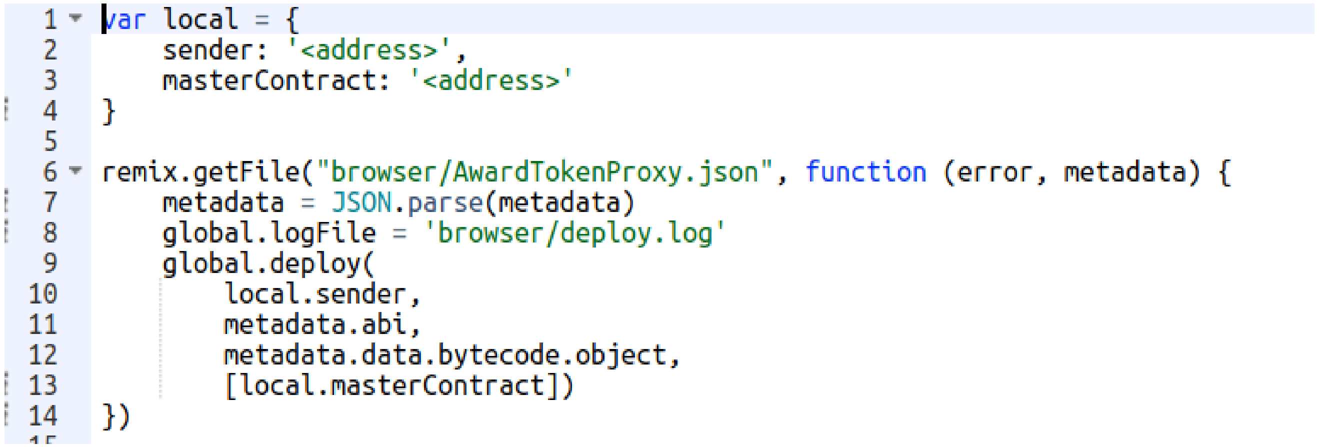
```
    console.log(msg)
}

global.info = function (msg) {
    global.log('INFO', msg)
}

global.error = function (msg) {
    global.log('ERR', msg)
}

global.deploy = function (sender, abi, bytecode, params) {
 try {
    var deployObject = new web3.eth.Contract(abi)
    global.info('deploying...')
    deployObject.deploy({
        data: '0x' + bytecode,
        arguments: params
    }).send({
        from: sender,
        gas: 1500000,
        gasPrice: '30000000000000'
    }, function(error, transactionHash){
    })
    .on('error', function(error){
        global.error(error)
    })
    .on('transactionHash', function(transactionHash){
        global.info('transactionHash ' + transactionHash)
    })
    .on('receipt', function(receipt){
        global.info('receipt:') // contains the new contrac
        global.info(receipt)
    })
    .on('confirmation', function(confirmationNumber, receip
        global.info(confirmationNumber)
    })
    .then(function(newContractInstance){
        global.info('contract address ' + newContractInstan
    });
     } catch (e) {
        console.error(e)
        global.error(e)
     }
}

exports.register('global', global)
```

Deploy

Then we load the script that will actually deploy:

remix.loadurl('https://github.com/ethereum/remix-workshops/runningScript/proxyContract_AwardToken/deploy.js')

See that “sender” and “masterContract” still need to be defined and that we need the abi and bytecode of AwardTokenProxy.

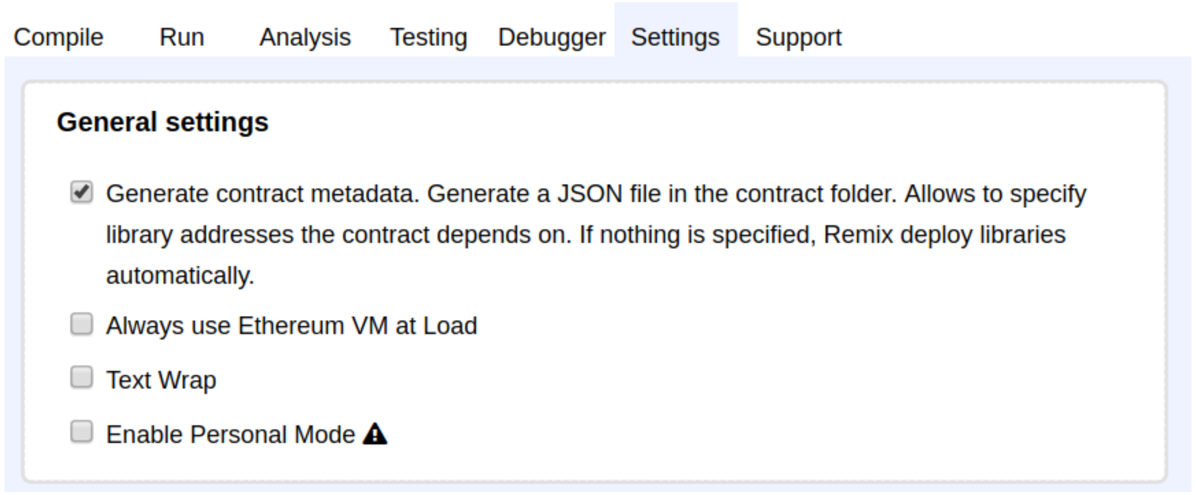


```
var local = {
    sender: '<address>',
    masterContract: '<address>'
}

remix.getFile("browser/AwardTokenProxy.json", function (error, metadata) {
    metadata = JSON.parse(metadata)
    global.logFile = 'browser/deploy.log'
    global.deploy(
        local.sender,
        metadata.abi,
        metadata.data.bytecode.object,
        [local.masterContract])
})
```

Deploy

For Deploying, we need the abi and bytecode accessible from the script. The contract metadata file is already on github so we don't need to generate it.

In case generating this file is needed, the "Generate contract metadata" should be checked in the settings.



General settings

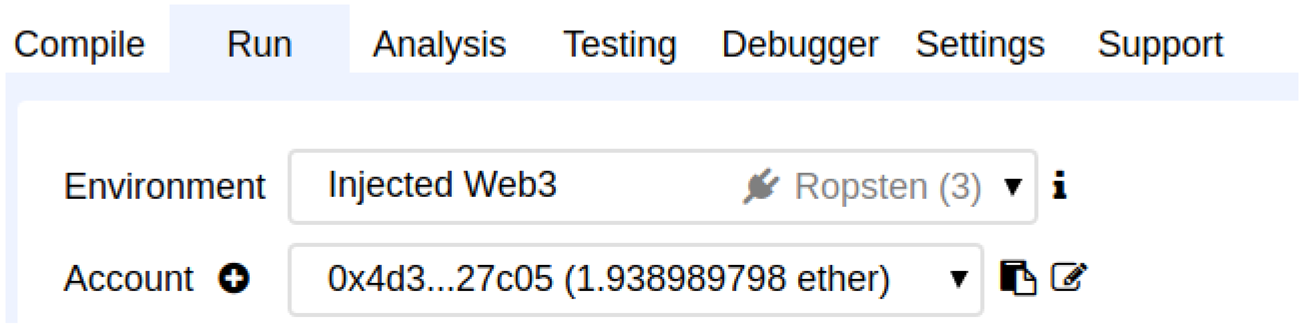
- [✓] Generate contract metadata. Generate a JSON file in the contract folder. Allows to specify library addresses the contract depends on. If nothing is specified, Remix deploy libraries automatically.
- [ ] Always use Ethereum VM at Load
- [ ] Text Wrap
- [ ] Enable Personal Mode ⚠

Deploy

We need the AwardToken master contract deployed:

Select a test network on Metamask, switch to it in Remix.

Compile and deploy the AwardToken (this is our master contract).



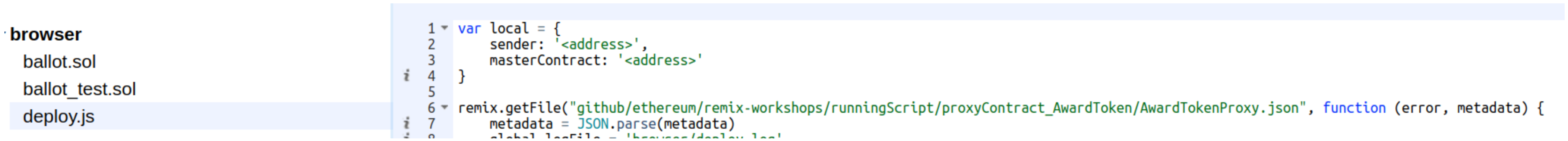
Environment: Injected Web3 Ropsten (3)

Account: 0x4d3...27c05 (1.938989798 ether)

Deploy

As we need to modify deploy.js

create a new file in browser explorer and copy the content of deploy.js to it.



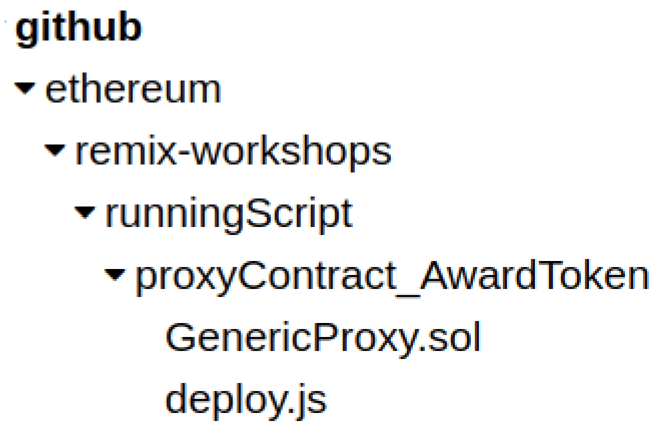
```
var local = {
    sender: '<address>',
    masterContract: '<address>'
}

remix.getFile("github/ethereum/remix-workshops/runningScript/proxyContract_AwardToken/AwardTokenProxy.json", function (error, metadata) {
    metadata = JSON.parse(metadata)
    global logFile = 'browser/deploy.log'
```

Deploy

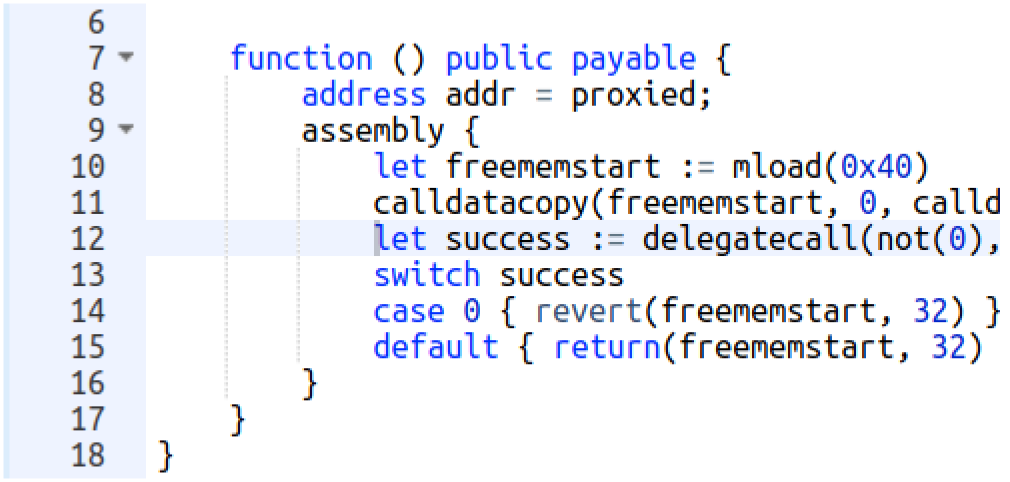
Update the `local` JavaScript variable with the transaction sender (that your metamask account) and the address of the master contract.

Select deploy.js and run remix.execute().

you can also check the deployment log in browser/deploy.log



- **github**
  - ethereum
    - remix-workshops
      - runningScript
        - proxyContract_AwardToken
          - GenericProxy.sol
          - deploy.js



```

    function () public payable {
        address addr = proxied;
        assembly {
            let freememstart := mload(0x40)
            calldatacopy(freememstart, 0, calld
            let success := delegatecall(not(0),
            switch success
            case 0 { revert(freememstart, 32) }
            default { return(freememstart, 32)
        }
    }
}
```

Deploy

Check in deploy.log for the transaction hash, and the contract address.

Please be sure to check the following post:

https://blog.gnosis.pm/solidity-delegateproxy-contracts-e09957d0f201