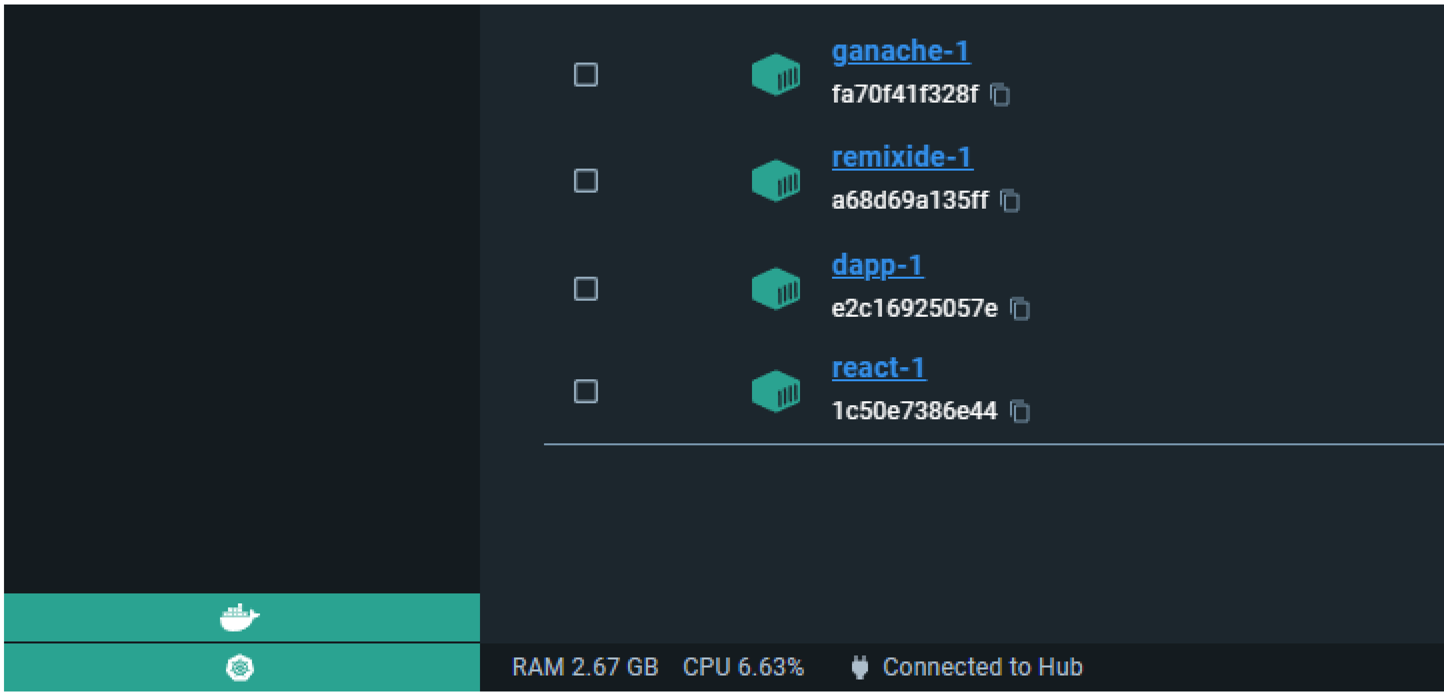
/*
#-- BCDV 1019 Summer 2023 – Lab 3
#-- P. Mahama, 101458594
#-- Email: Patrick.Mahama@georgebrown.ca
#-- Date: May 8, 2023
*/

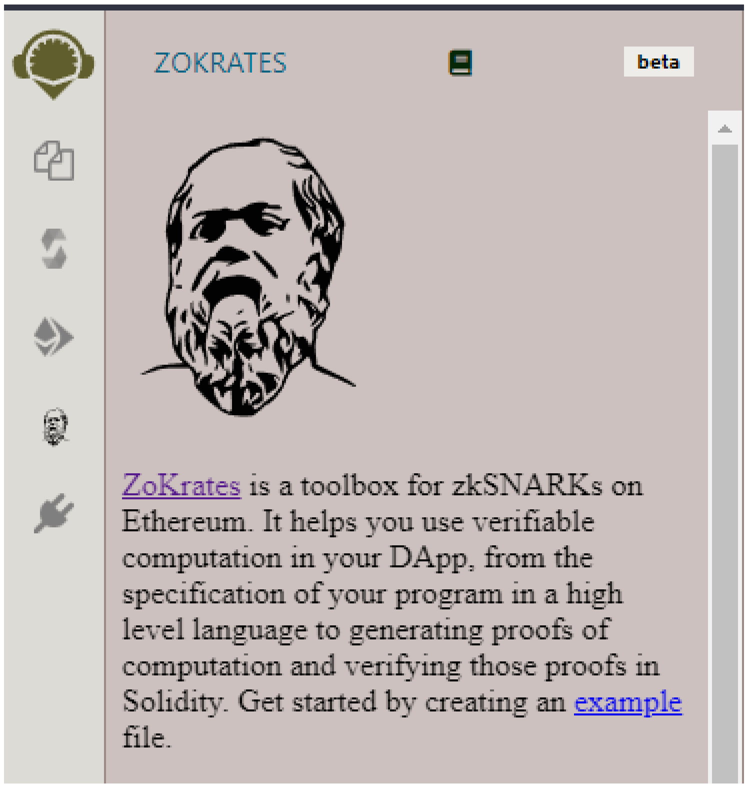
Lab 6 – code in

Due May 19, 2023 6:00 PM
Instructions
ZKP Smart Contract using Zokrates:
Follow the instructions from today's lecture and complete the following:
• Create a Smart Contract to verify a simple ZKP equation
○ a * a = b
• Use Zokrates to create the proof and verify it using smart contract
• The output of the transaction should return:
• Upload the code to github and submit the link along with the snapshot to Assignments in Microsoft Teams

Step 0: Fire up dev environment


```
ganache-1
fa70f41f328f
remixide-1
a68d69a135ff
dapp-1
e2c16925057e
react-1
1c50e7386e44
RAM 2.67 GB   CPU 6.63%   Connected to Hub
```

Access Remix and Add Zocrates plugin



ZoKrates is a toolbox for zkSNARKs on Ethereum. It helps you use verifiable computation in your DApp, from the specification of your program in a high level language to generating proofs of computation and verifying those proofs in Solidity. Get started by creating an example file.

ZoKrates is a toolbox for zkSNARKs on Ethereum. It helps you use verifiable computation in your DApp, from the specification of your program in a high level language to generating proofs of computation and verifying those proofs in Solidity. Get started by creating an example file.

Select a proving scheme Groth16
Univeral Setup

Step 1: Compile: click Compile

Compiled successfully!
(1 constraint)

Step 2: Compute

Computes a witness for the compiled program. A witness is a valid assignment of the variables, which include the results of the computation.

a

1

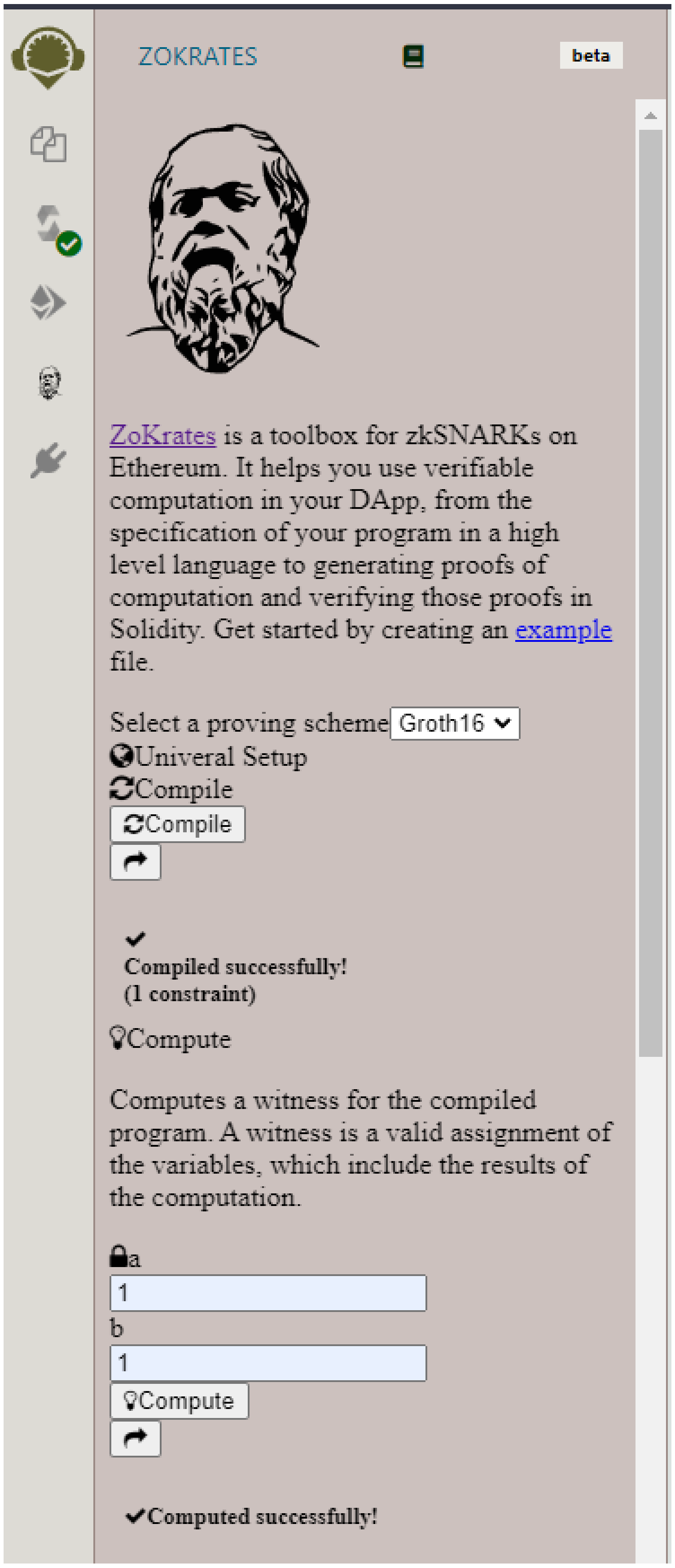
b

1

Click Compute

Computed successfully!

```
[]
```



ZOKRATES

beta

ZoKrates is a toolbox for zkSNARKs on Ethereum. It helps you use verifiable computation in your DApp, from the specification of your program in a high level language to generating proofs of computation and verifying those proofs in Solidity. Get started by creating an example file.

Select a proving scheme Groth16

Univeral Setup

Compile

Compile

✓
Compiled successfully!
(1 constraint)

Compute

Computes a witness for the compiled program. A witness is a valid assignment of the variables, which include the results of the computation.

a
1
b
1

Compute

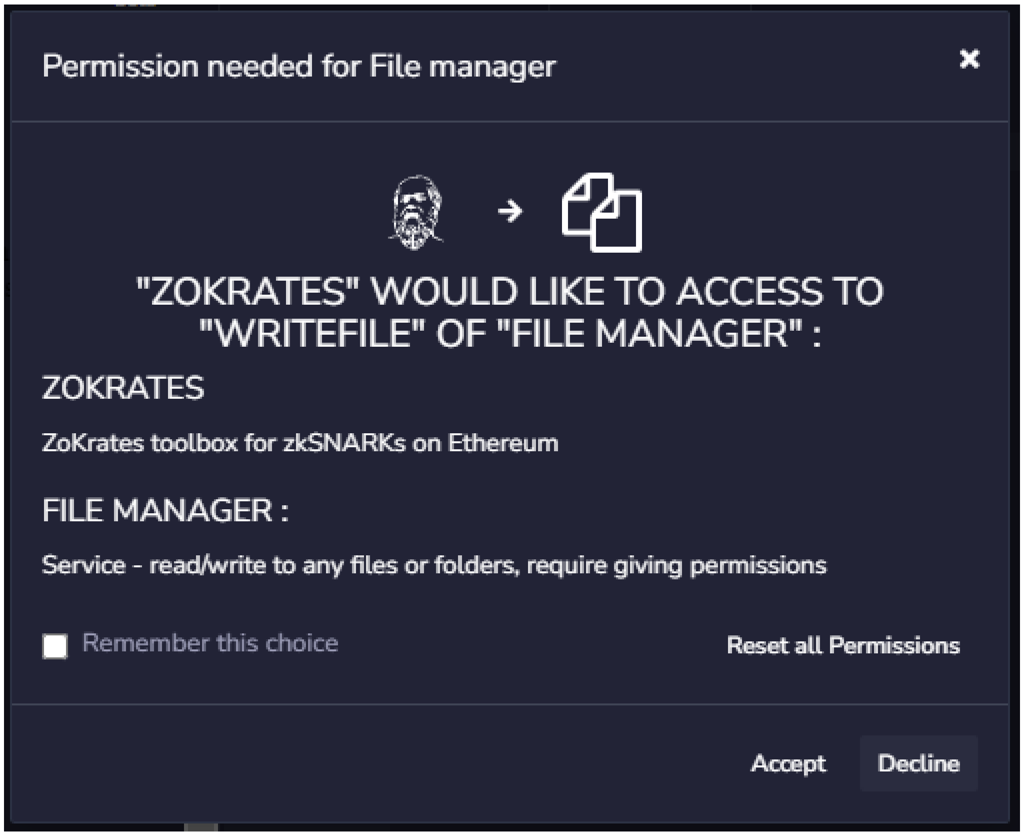
✓Computed successfully!

Step 3: Setup

Creates a proving key and a verification key. These keys are derived from a source of randomness, commonly referred to as “toxic waste”.

Click Run Setup >> click Accept, Click Remember this choice

Setup completed!



Permission needed for File manager

"ZOKRATES" WOULD LIKE TO ACCESS TO "WRITEFILE" OF "FILE MANAGER" :

ZOKRATES

ZoKrates toolbox for zkSNARKs on Ethereum

FILE MANAGER :

Service - read/write to any files or folders, require giving permissions

Remember this choice

Reset all Permissions

Accept Decline

Step 4: Generate Proof

Generates a proof for a computation of the compiled program using proving key and computed witness.

Click Generate button

Verifier inputs:

```
[["0x26ff8fba5e3766cdfe306f8029a080246b845876e701ca829fe3d1fa4e4dd52a","0x
2f425f1ae3602d26c45ae614bdd074ab5fbab004c20be801d2abb8ff259fd8e4"],[["0x1
fc740c3ca1fa7b4fcb67129796c817e603d5e8595ff2f234e1bb831fa84c8c5","0x0e5e6
5d5ae03eb9ba2f09a1d8807c2e8512ada73e700c2465579aeb0f6f745c4"],["0x132f52
4d7c1dad12bd460f9fcef9afe25eebd1045861def6b97f68c98135907c","0x0fab520e29
559308fb3f84727a987ff2ad3283c8105a84a66ca96440b2d5baa8"]],["0x151a824380
92119de54bbfed92dc6934b35cd53ae9de74dfb22ab37665dae8df","0x02a35bcbd85
79d4cc40244467bb20855748a3f786db1a8ecbe5a82c5a0ea6795"]],["0x0000000000
00000000000000000000000000000000000000000000000000001"]] [[
```

Proof generated!

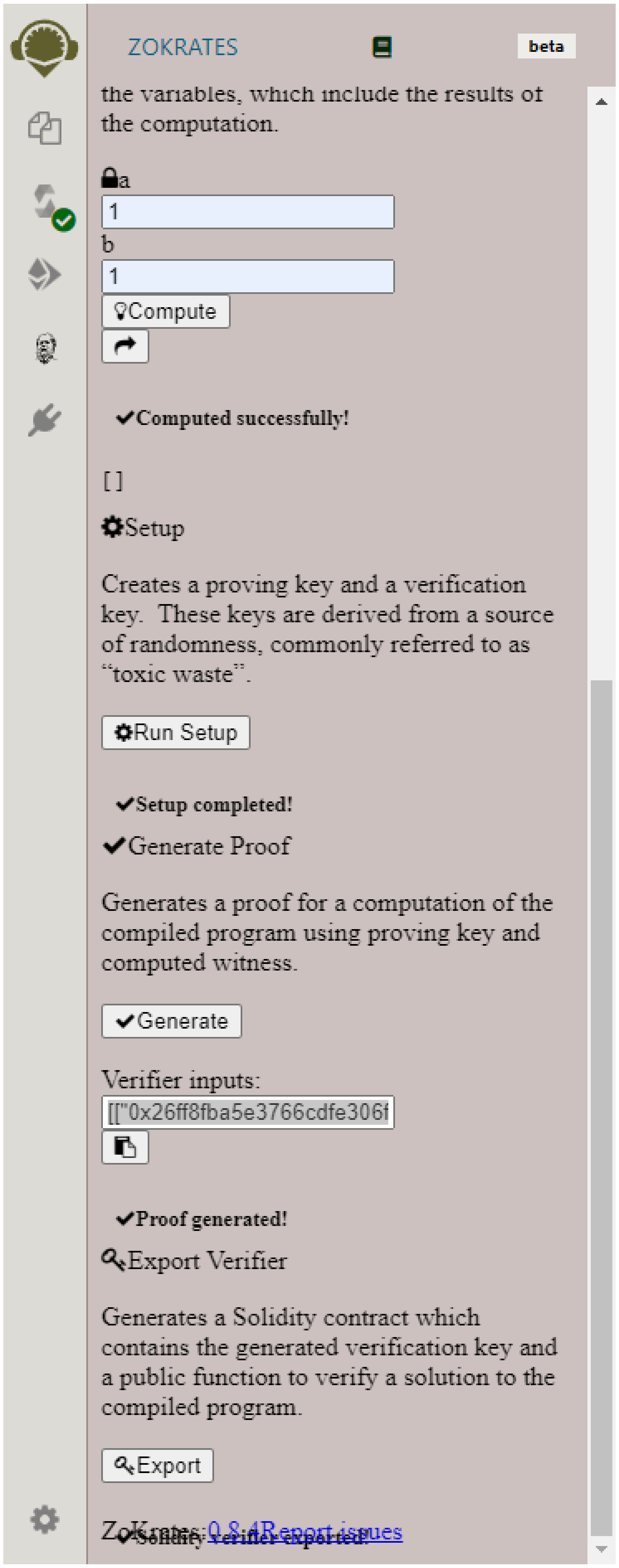
Output: proof.json file

Step 5: Export Verifier

Generates a Solidity contract which contains the generated verification key and a public function to verify a solution to the compiled program.

Click export button

Output: verifier.sol file


ZOKRATES

beta

the variables, which include the results of the computation.

a

1

b

1

Compute

✓Computed successfully!

[ ]

Setup

Creates a proving key and a verification key. These keys are derived from a source of randomness, commonly referred to as "toxic waste".

Run Setup

✓Setup completed!

✓Generate Proof

Generates a proof for a computation of the compiled program using proving key and computed witness.

✓Generate

Verifier inputs:

[["0x26ff8fba5e3766cdfe306f

✓Proof generated!

Export Verifier

Generates a Solidity contract which contains the generated verification key and a public function to verify a solution to the compiled program.

Export

ZoKrates: 0.8.4 Report issues

✓Solidity verifier exported!

Step 6: Compile verifier.sol

Environment: Web3 Provider (Ganache http://127.0.0.1:8545)

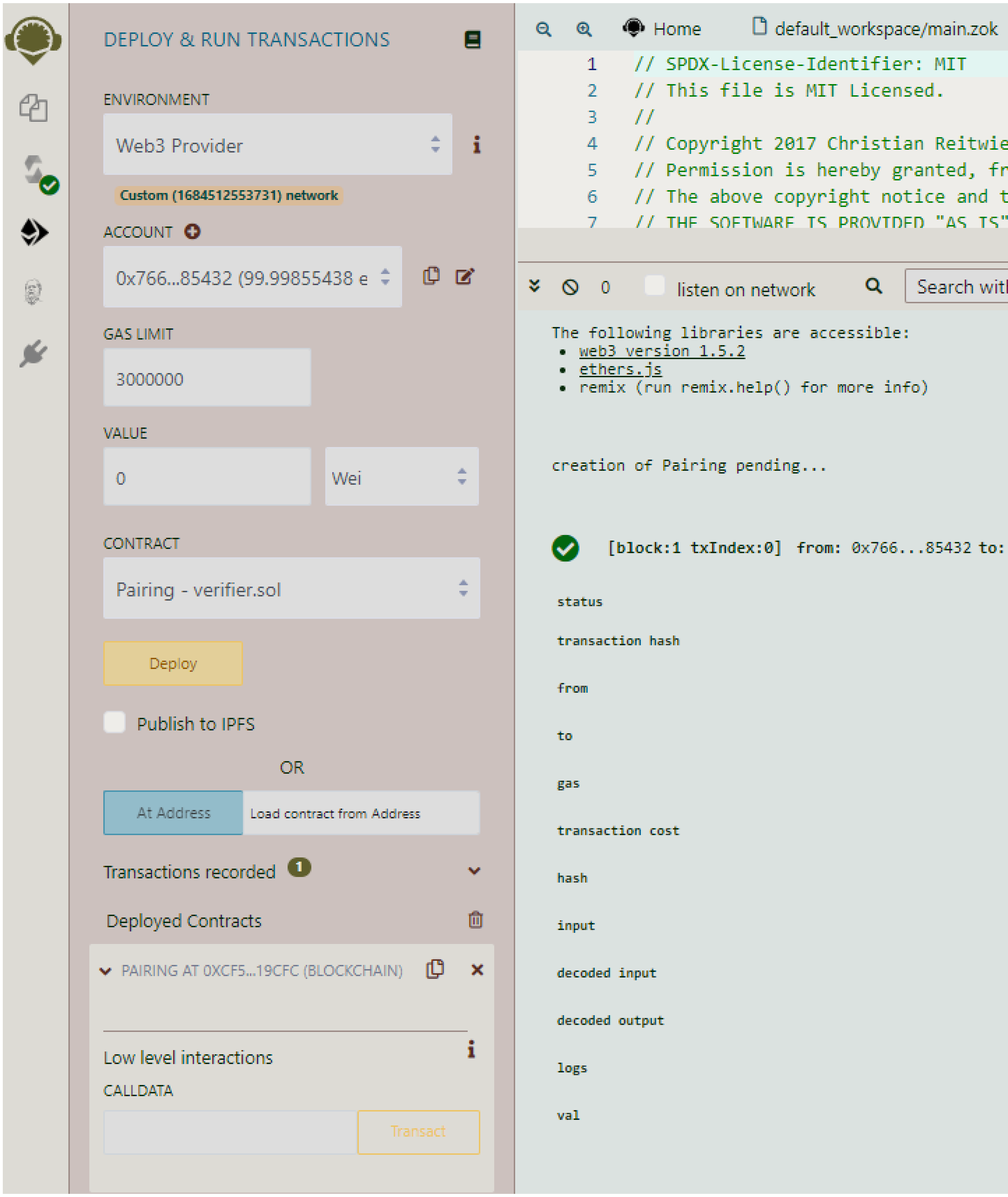
Account: 0x766f6caB2ff1A0c6Db3e5102237c3B2225485432

Gas Limit: 3000000

Value: 0 Wei

Contract: Pairing – contracts/verifier.sol

Step 7a: Deploy Contract: pairing ??



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT
Web3 Provider
Custom (1684512553731) network

ACCOUNT
0x766...85432 (99.99855438 e

GAS LIMIT
3000000

VALUE
0 Wei

CONTRACT
Pairing - verifier.sol

Deploy

Publish to IPFS

OR

At Address | Load contract from Address

Transactions recorded 1

Deployed Contracts

PAIRING AT 0XCF5...19CFC (BLOCKCHAIN)

Low level interactions

CALLDATA

Transact

Home | default_workspace/main.zok

```
// SPDX-License-Identifier: MIT
// This file is MIT Licensed.
//
// Copyright 2017 Christian Reitwie
// Permission is hereby granted, fr
// The above copyright notice and t
// THE SOFTWARE IS PROVIDED "AS IS"
```

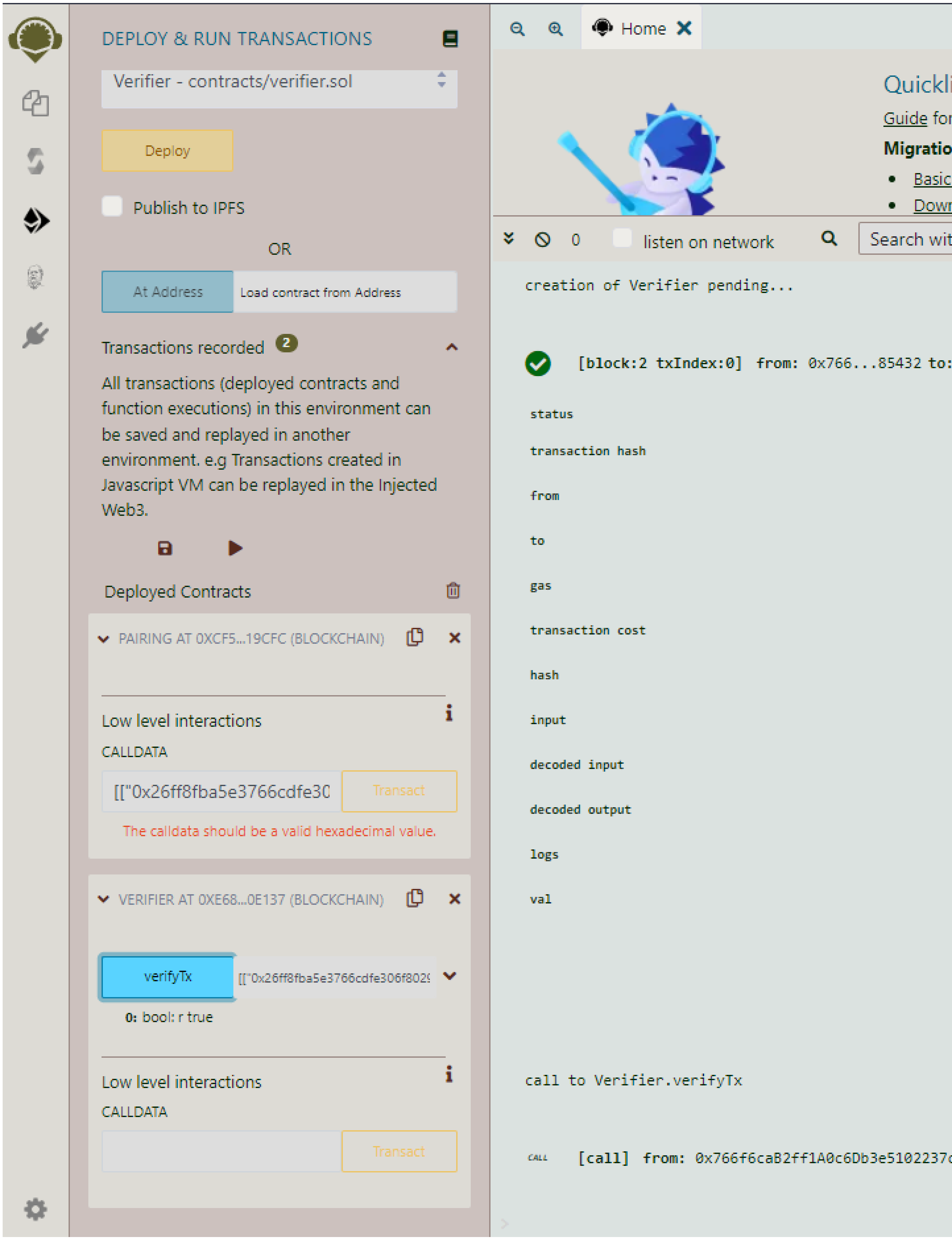
0 listen on network Search wit

```
The following libraries are accessible:
 • web3 version 1.5.2
 • ethers.js
 • remix (run remix.help() for more info)


creation of Pairing pending...


[block:1 txIndex:0] from: 0x766...85432 to:

status
transaction hash
from
to
gas
transaction cost
hash
input
decoded input
decoded output
logs
val
```

Step 7b: Deploy Contract - verifer

Step 8: call to verifier

Output: true


DEPLOY & RUN TRANSACTIONS

Verifier - contracts/verifier.sol

Deploy

Publish to IPFS

OR

At Address | Load contract from Address

Transactions recorded 2

All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.

Deployed Contracts

PAIRING AT 0XCF5...19CFC (BLOCKCHAIN)

Low level interactions

CALLDATA

[["0x26ff8fba5e3766cdfe30 | Transact

The calldata should be a valid hexadecimal value.

VERIFIER AT 0XE68...0E137 (BLOCKCHAIN)

verifyTx | [["0x26ff8fba5e3766cdfe306f802

0: bool: r true

Low level interactions

CALLDATA

Transact

Home

Quickli
Guide for
Migratio
- Basic
- Dowr

0 listen on network Search wit

```
creation of Verifier pending...

[block:2 txIndex:0] from: 0x766...85432 to:

status
transaction hash
from
to
gas
transaction cost
hash
input
decoded input
decoded output
logs
val

call to Verifier.verifyTx

CALL [call] from: 0x766f6caB2ff1A0c6Db3e5102237
```