Blockchain Track Quickstart Document

During the competition you'll have the opportunity to Build and Deploy Smart Contracts locally to your own machine or to Microsoft Azure's Blockchain Service in the Cloud. This is **not required** in order to fulfill the criteria of the Rubric and Guiding Document for the competition, but working through these examples can help get you more familiar with Blockchain technology and organize your ideas for what you will present as your final submission.

Installing Prerequisite Software

We highly recommend that you download and install Visual Studio Code - Insiders as it has several extensions that will make it easier to interact with the Azure Blockchain Service. You can download the appropriate version for your Operatiing System using the following link:

https://code.visualstudio.com/insiders/

Once installed, you'll want to access the Extensions tab with Ctrl+Shift+X or Command+Shift+X and install the following Extensions from the Marketplace:

- Blockchain Development Kit for Ethereum
- Azure Account
- solidity

More information about the Blockchain Development Kit for Ethereum extension can be found here:

https://marketplace.visualstudio.com/items?itemName=AzBlockchain.azure-blockchain

Once installed, we'll need just a few more prerequisites before we're ready to begin developing:

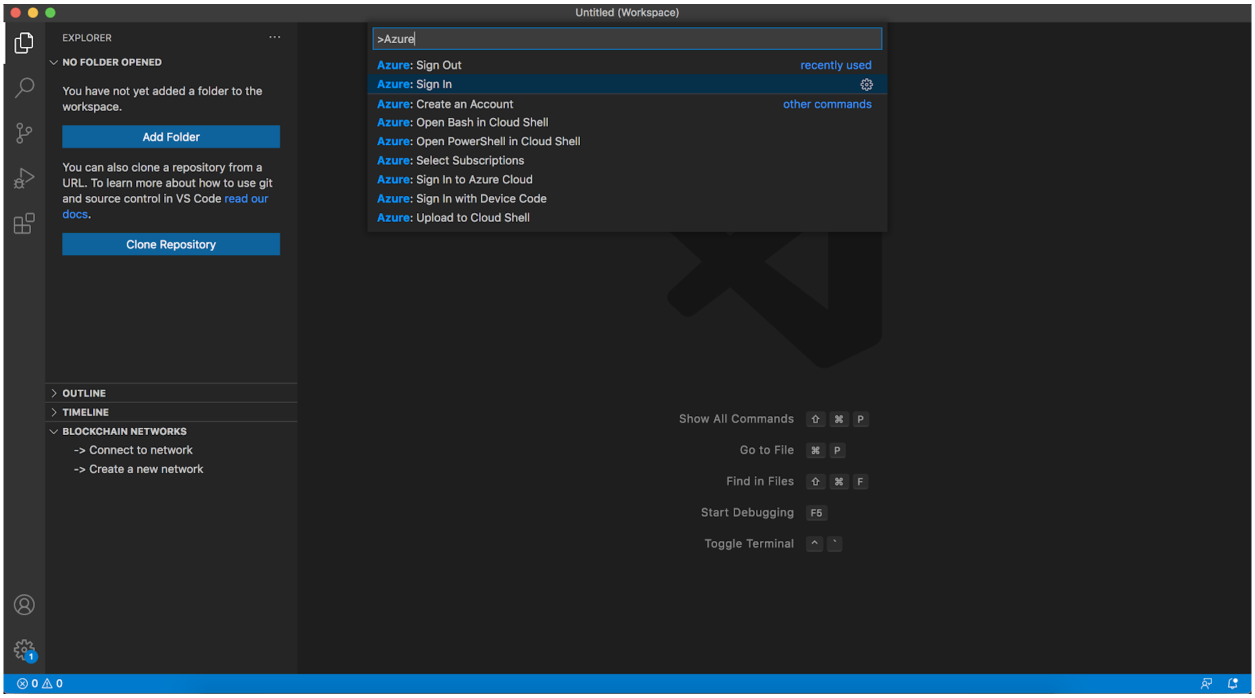
- Node.js 10.15.X or higher: https://nodejs.org/en/download/
- Git 2.10.X or higher: https://git-scm.com/downloads
- Truffle 5.0.0: https://www.trufflesuite.com/docs/truffle/getting-started/installation
- Ganache: https://www.trufflesuite.com/ganache

Once all of those prerequisites are installed then you're ready to begin!

Helloworld Smart Contract

A great place to start is with a simple Hello World smart contract. First we'll need to open Visual Studio Code and connect to your Azure Account assigned for this event. You can find your Azure Account email address and Password in your team's Discord channel.

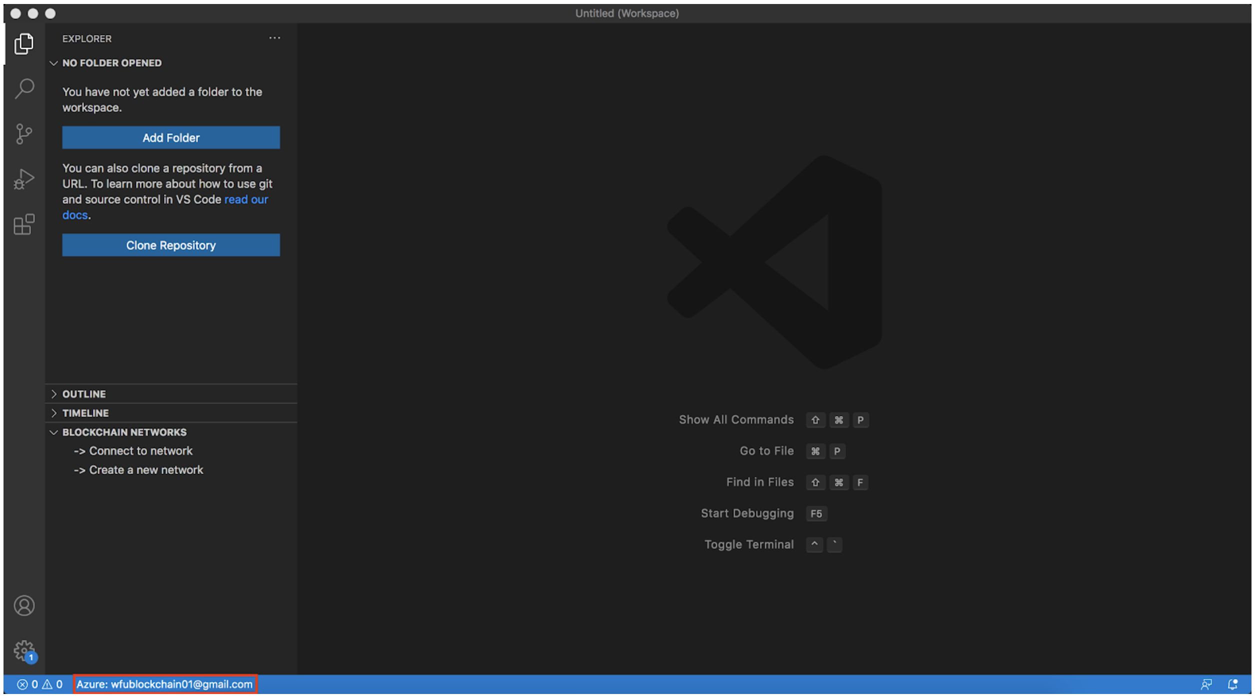
Start by opening the Command Panel in Visual Studio Code by selecting View -> Command Palette in the menu bar. Once the Command Palette is up search for **Azure: SIgn In**



You have not yet added a folder to the workspace.

You can also clone a repository from a URL. To learn more about how to use git and source control in VS Code read our docs.

Hit Enter and a window will open in your browser where you can log in. Use the Azure Email Address and Azure Password to login. You can verify you’re logged in if you see your Azure Email Address in the bottom left corner of Visual Studio Code.

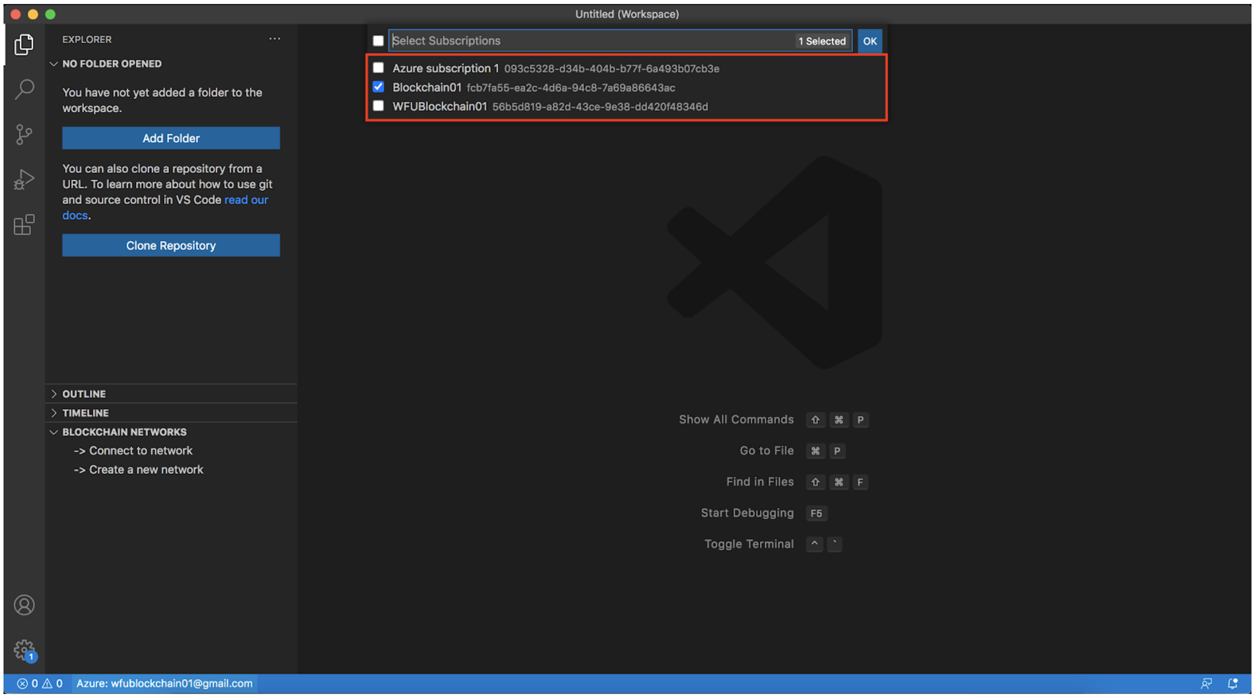


You have not yet added a folder to the workspace.

You can also clone a repository from a URL. To learn more about how to use git and source control in VS Code read our docs.

Azure: wfublockchain01@gmail.com

Next, left click on your Azure Account email address (highlighted in the previous screenshot) and make sure only the checkbox is checked next to the Subscription listed in your team's Discord channel. Once you've selected the correct checkbox, click **OK** in the drop down.



Untitled (Workspace)

□ Select Subscriptions 1 Selected OK

□ Azure subscription 1 093c5328-d34b-404b-b77f-6a493b07cb3e
☑ Blockchain01 fcb7fa55-ea2c-4d6a-94c8-7a69a86643ac
□ WFUBlockchain01 56b5d819-a82d-43ce-9e38-dd420f48346d

EXPLORER

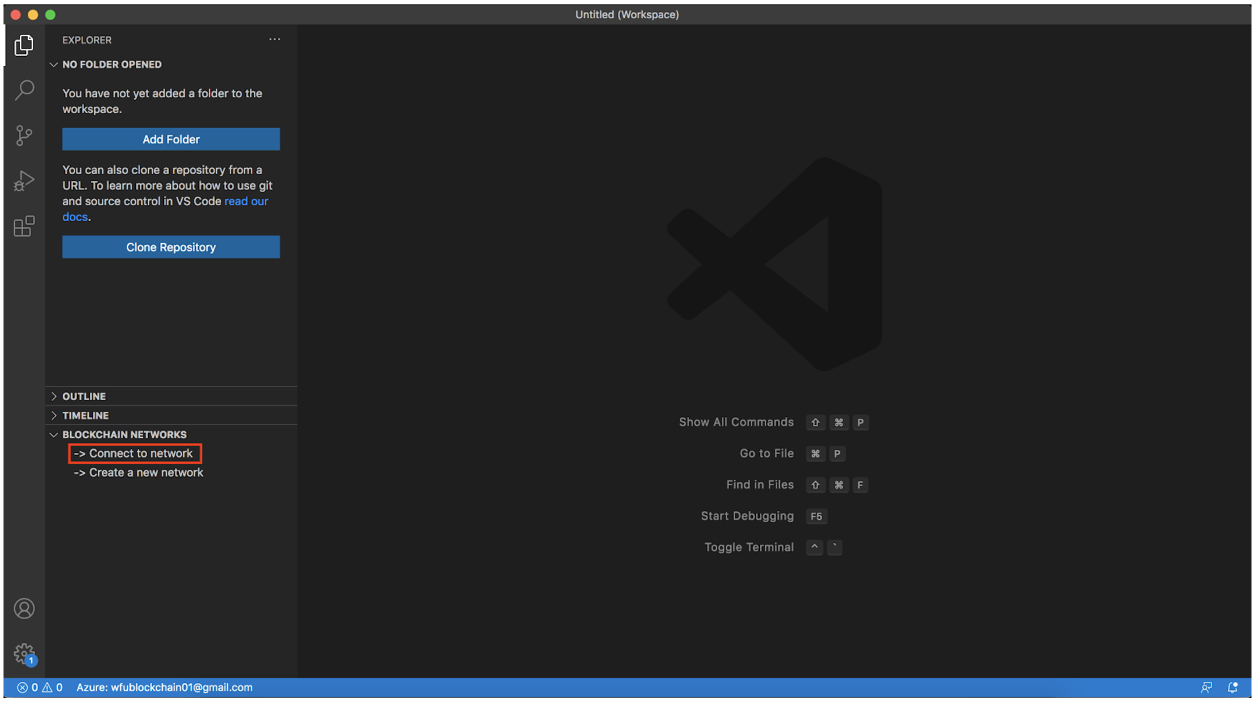
NO FOLDER OPENED

You have not yet added a folder to the workspace.

You can also clone a repository from a URL. To learn more about how to use git and source control in VS Code read our docs.

Azure: wfublockchain01@gmail.com

Once you've selected the correct subscription, then you can connect to your Blockchain Member in Microsoft Azure. Under Blockchain Networks, select **Connect to Network**



Untitled (Workspace)

EXPLORER

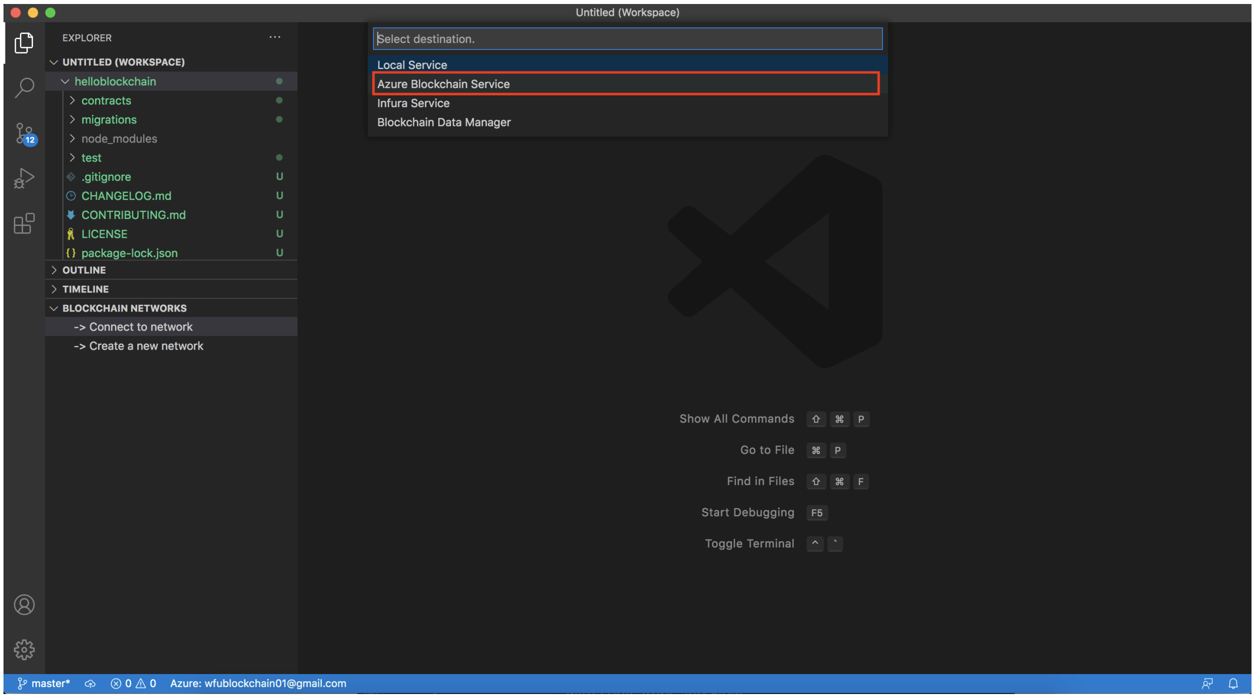
NO FOLDER OPENED

You have not yet added a folder to the workspace.

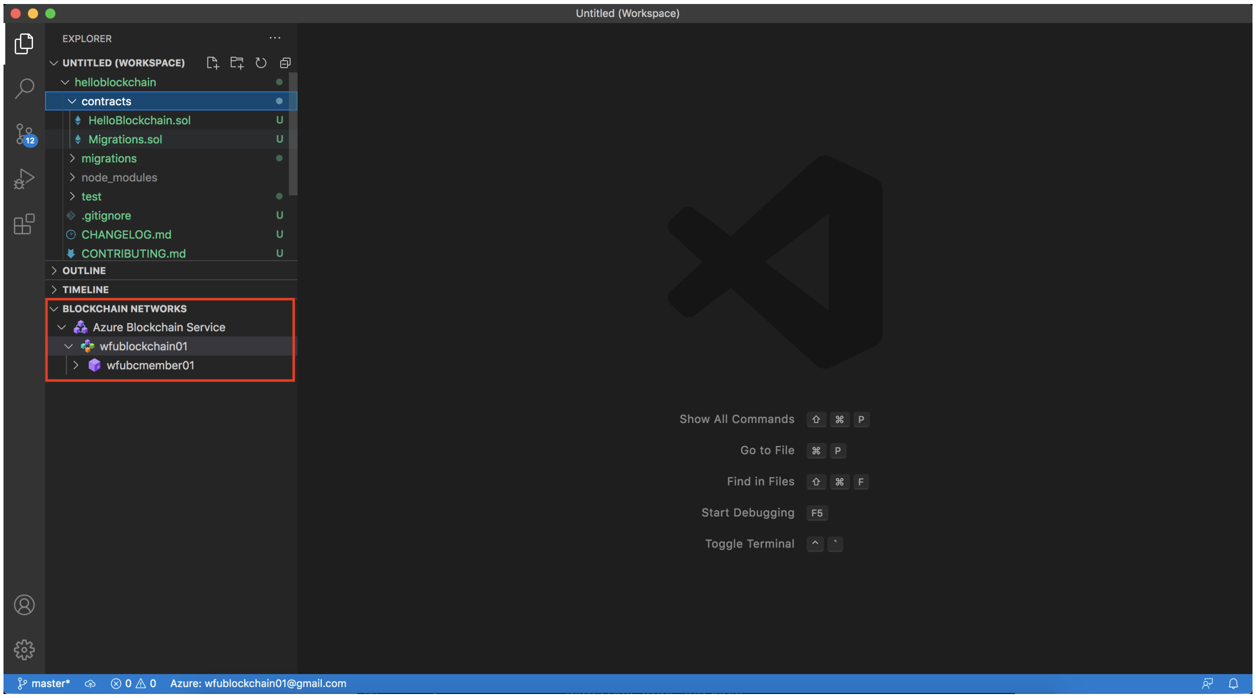
You can also clone a repository from a URL. To learn more about how to use git and source control in VS Code read our docs.

Azure: wfublockchain01@gmail.com

Select **Azure Blockchain Service** from the drop down menu, and proceed to select the only option for the remaining drop down for Resource Group and Consortium



Once you are finished you should see the blockchain member in the bottom left panel.



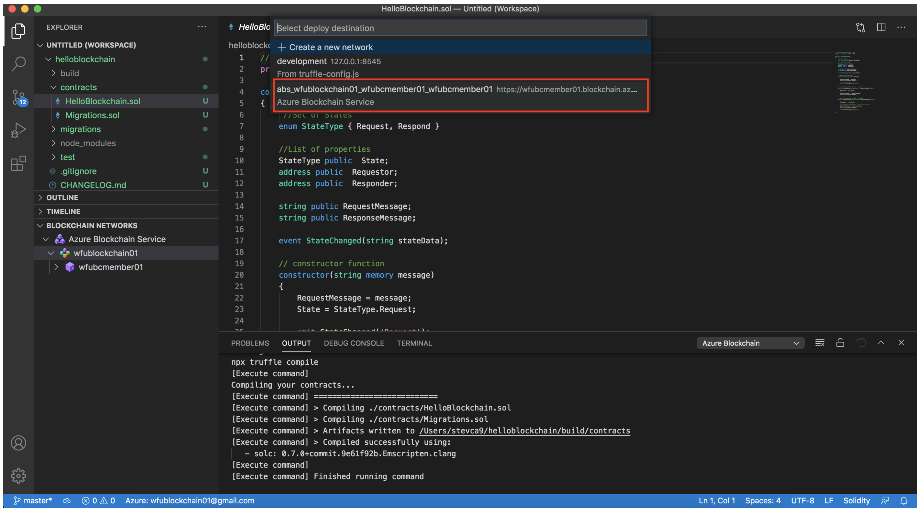
Once connected, I recommend you follow this tutorial to build and deploy your first first HelloBlockchain.sol smart contract:

https://docs.microsoft.com/en-us/azure/blockchain/service/send-transaction

This tutorial will show you:

- How to create a new Solidity Project in Visual Studio Code
- Build your HelloBlockchain.sol contract
- And Deploy your contract to your blockchain member we connected to previously

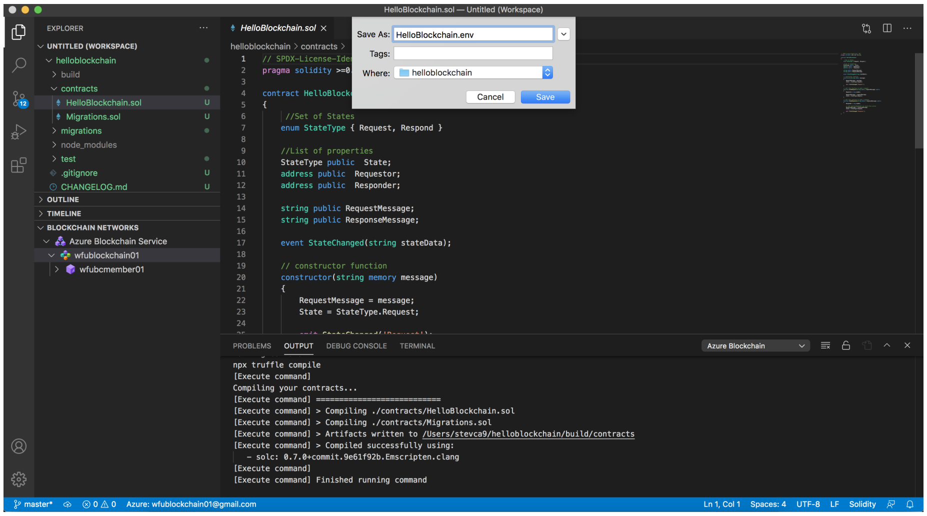
When you're ready to deploy your smart contract, make sure you select the blockchain member we connected to previously.



```
    enum StateType { Request, Respond }

    //List of properties
    StateType public  State;
    address public  Requestor;
    address public  Responder;

    string public RequestMessage;
    string public ResponseMessage;

    event StateChanged(string stateData);

    // constructor function
    constructor(string memory message)
    {
        RequestMessage = message;
        State = StateType.Request;
```

```
npx truffle compile
[Execute command]
Compiling your contracts...
[Execute command] ===========================
[Execute command] > Compiling ./contracts/HelloBlockchain.sol
[Execute command] > Compiling ./contracts/Migrations.sol
[Execute command] > Artifacts written to /Users/stevca9/helloblockchain/build/contracts
[Execute command] > Compiled successfully using:
   - solc: 0.7.0+commit.9e61f92b.Emscripten.clang
[Execute command]
[Execute command] Finished running command
```

You will also want to generate a new mnemonic and place it in your project folder.

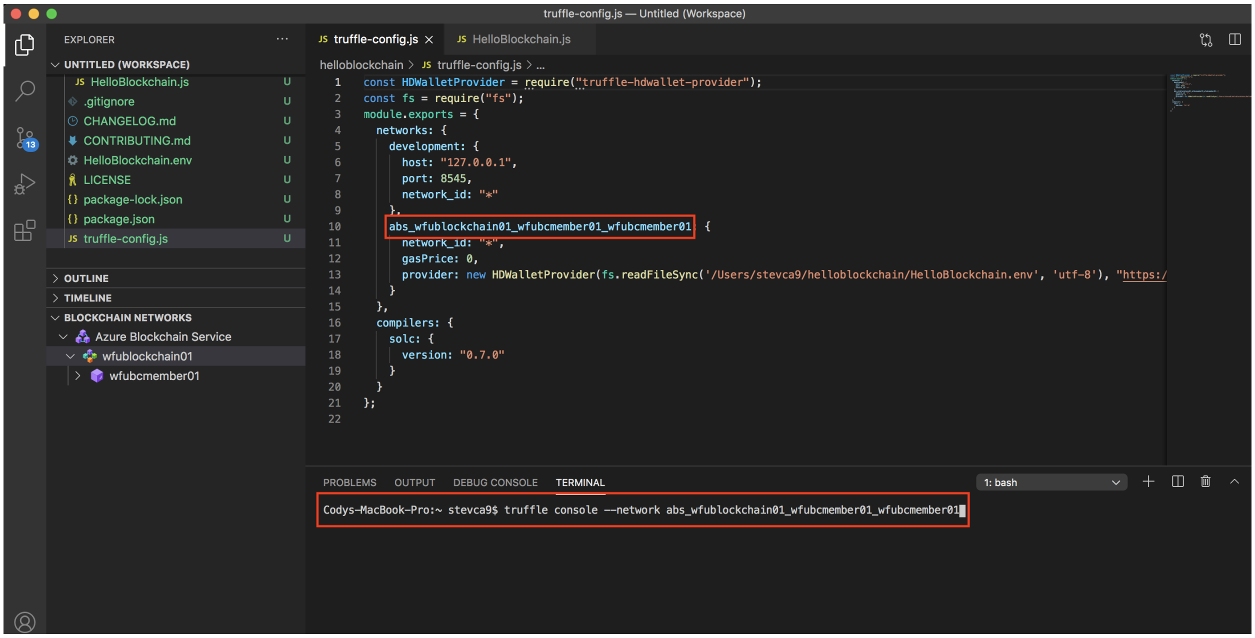


Save As: HelloBlockchain.env
Tags:
Where: helloblockchain

```
// SPDX-License-Ide
pragma solidity >=0

contract HelloBlock
{
     //Set of States
    enum StateType { Request, Respond }

    //List of properties
    StateType public  State;
    address public  Requestor;
    address public  Responder;

    string public RequestMessage;
    string public ResponseMessage;

    event StateChanged(string stateData);

    // constructor function
    constructor(string memory message)
    {
        RequestMessage = message;
        State = StateType.Request;
```

```
npx truffle compile
[Execute command]
Compiling your contracts...
[Execute command] ===========================
[Execute command] > Compiling ./contracts/HelloBlockchain.sol
[Execute command] > Compiling ./contracts/Migrations.sol
[Execute command] > Artifacts written to /Users/stevca9/helloblockchain/build/contracts
[Execute command] > Compiled successfully using:
   - solc: 0.7.0+commit.9e61f92b.Emscripten.clang
[Execute command]
[Execute command] Finished running command
```

You can end the tutorial above once you get to this step. If you want to call your contract's functions, you can then proceed with the following steps. First, find your blockchain network in the file *truffle-config.js*

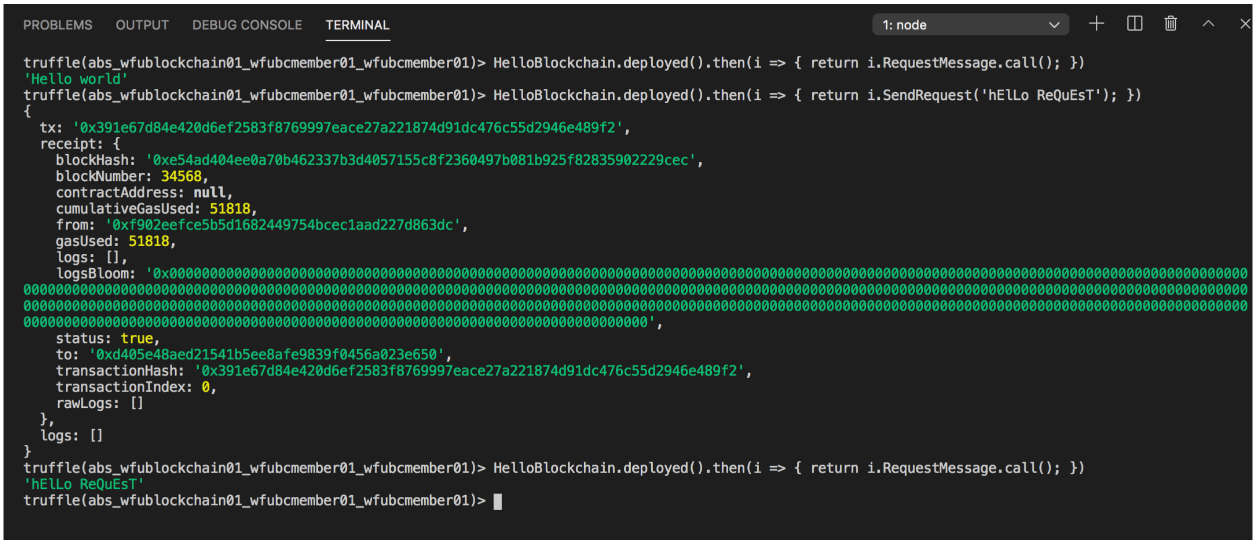


```
const HDWalletProvider = require("truffle-hdwallet-provider");
const fs = require("fs");
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 8545,
      network_id: "*"
    },
    abs_wfublockchain01_wfubcmember01_wfubcmember01: {
      network_id: "*",
      gasPrice: 0,
      provider: new HDWalletProvider(fs.readFileSync('/Users/stevca9/helloblockchain/HelloBlockchain.env', 'utf-8'), "https:/
    }
  },
  compilers: {
    solc: {
      version: "0.7.0"
    }
  }
};
```

```
Codys-MacBook-Pro:~ stevca9$ truffle console --network abs_wfublockchain01_wfubcmember01_wfubcmember01
```

Then in the Terminal run the command **truffle console --network *your_network***

Once connected to the truffle console, you can then are start running command to interact the functions defined in HelloBlockchain.sol



```
truffle(abs_wfublockchain01_wfubcmember01_wfubcmember01)> HelloBlockchain.deployed().then(i => { return i.RequestMessage.call(); })
'Hello world'
truffle(abs_wfublockchain01_wfubcmember01_wfubcmember01)> HelloBlockchain.deployed().then(i => { return i.SendRequest('hElLo ReQuEsT'); })
{
  tx: '0x391e67d84e420d6ef2583f8769997eace27a221874d91dc476c55d2946e489f2',
  receipt: {
    blockHash: '0xe54ad404ee0a70b462337b3d4057155c8f2360497b081b925f82835902229cec',
    blockNumber: 34568,
    contractAddress: null,
    cumulativeGasUsed: 51818,
    from: '0xf902eefce5b5d1682449754bcec1aad227d863dc',
    gasUsed: 51818,
    logs: [],
    logsBloom: '0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000',
    status: true,
    to: '0xd405e48aed21541b5ee8afe9839f0456a023e650',
    transactionHash: '0x391e67d84e420d6ef2583f8769997eace27a221874d91dc476c55d2946e489f2',
    transactionIndex: 0,
    rawLogs: []
  },
  logs: []
}
truffle(abs_wfublockchain01_wfubcmember01_wfubcmember01)> HelloBlockchain.deployed().then(i => { return i.RequestMessage.call(); })
'hElLo ReQuEsT'
truffle(abs_wfublockchain01_wfubcmember01_wfubcmember01)>
```

In the above example we ran the commands:

```
HelloBlockchain.deployed().then(i => { return i.RequestMessage.call(); })
```

Which returned the current string stored in our blockchain ledger. And

```
HelloBlockchain.deployed().then(i => { return i.SendRequest('hElLo ReQuEsT'); })
```

Which set a new string of our choosing to our blockchain ledger, replacing the previous one.

Review the code in HelloBlockchain.sol to see if you can understand how these functions work.

Other Helpful Tutorials

Some other helpful tutorials to consider are:

- Ethereum Pet Shop: https://www.trufflesuite.com/tutorials/pet-shop
    - In this tutorial you do not need to use Visual Studio Code or the Blockchain Memnber configured in Azure. You will use Ganache to create a development blockchain network on your local machine and then use a service called MetaMask to facilitate the transactiions to that network. This tutorial is also helpful as it can help you see some different ideas of how Blockchain technology can be used for other applications than Hello World.
- Connect MetaMask to your Azure Blockchain Member: https://docs.microsoft.com/en-us/azure/blockchain/service/connect-metamask
    - In this tutorial you will connect MetaMask to your Azure Blockchain member and then deploy a smart contract using Remix, a browser based IDE, to your Blockchain member.
    - https://remix.ethereum.org/

If you need any assistance connecting to any of the platforms discussed in this document, or questions about the above tutorials you can contact @CodyS or @windozer in the WakeHacks Discord server or open a Blockchain ticket.