BLOCKCHAIN POC SETUP MANUAL

Authors: Arundhati Rao and Vivek Sah

We are going to build a simple bank which accepts deposits and lends loan to someone who needs it. We are going to use blockchain technology to achieve this. The bank that we are referring to is not an actual bank but just few lines of code, but that code will act as a bank. We will see how very soon.

We are going to use Blockapps which is a platform built on Ethereum . Ethereum is a blockchain architecture like bitcoin but is more than just a cryptocurrency(https://www.ethereum.org/).It can support many applications, which are also known as smart contracts . Smart contracts are basically few lines of code which reside in Ethereum ecosystem and can be executed by anyone on the blockchain.

In this tutorial, we are going to write a smart contract and deploy it to Blockapps blockchain.

SetUp

Our application has three layers: a frontend webpage, a nodejs server, and blockapps blockchain. The nodejs server will act as a middleware between the frontend and the blockchain. We will use the following tools:

1. Visual Studio

We will be using Visual studio to write, compile and deploy the code. The code will be written is a language called Solidity (it’s like JavaScript). So, let’s go ahead and install a solidity plug in for visual studio.

- Install Visual Studio https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx

2. Blockapps-bloc

We will also use bloc (a command line tool from Blockapps) to set up a project and start the node server.

- Install Nodejs and npm https://nodejs.org/en/download/
- test if node is installed
  - `node -v` (should print a version number)
- test if node is installed
  - `npm -v` (should print a version number)

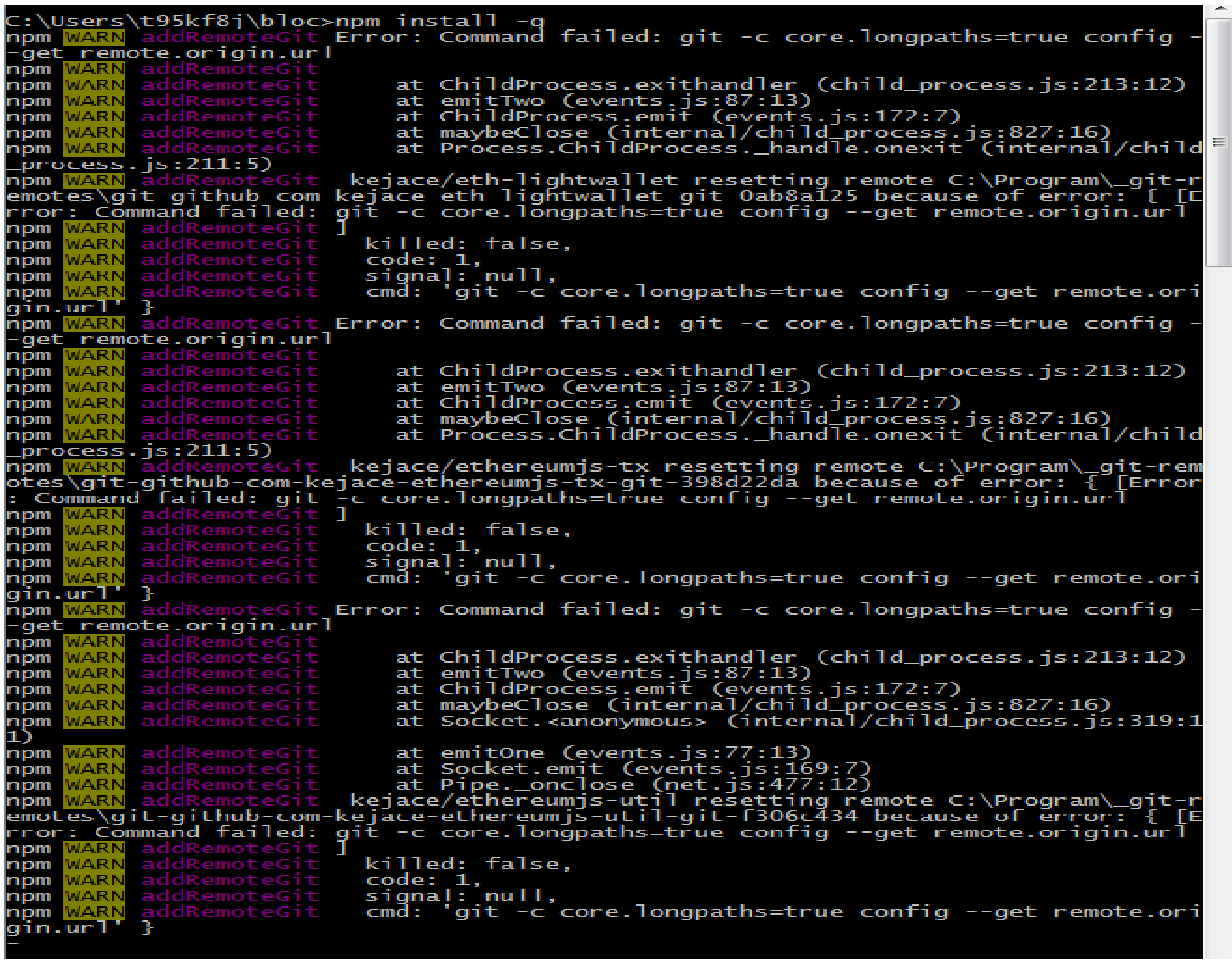
Installing Bloc:

1) Browse to the location where you want to install bloc.

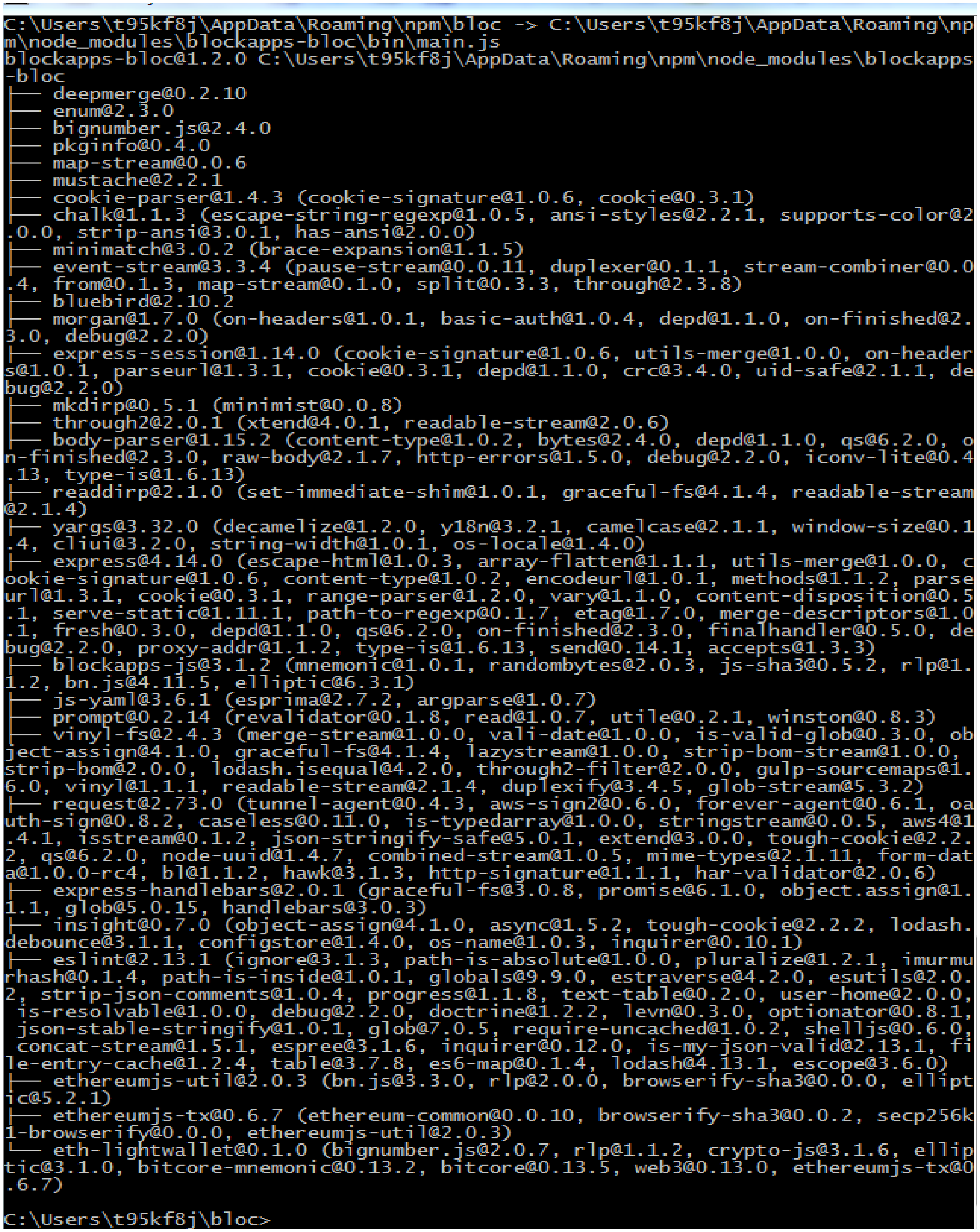
   Npm install –g blockapps-bloc

   Or you can also check out the github repo and build it by running:

   Git clone https://github.com/blockapps/bloc
   Cd bloc
   Npm install –g

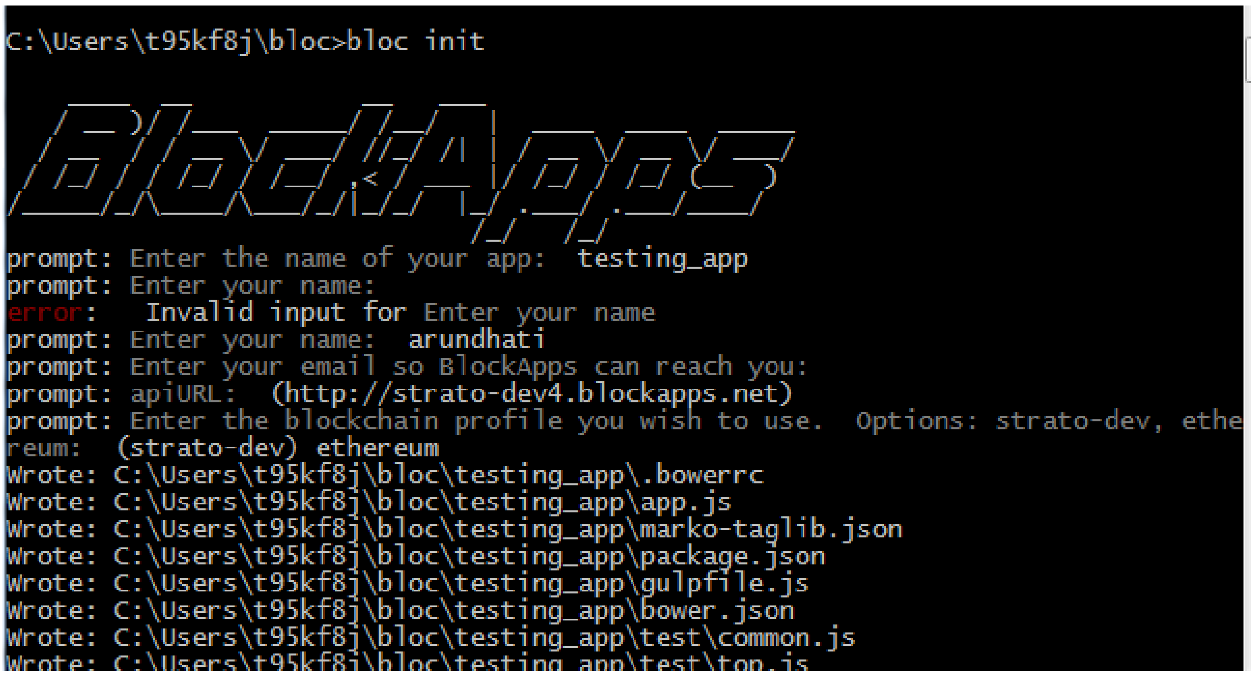


```
C:\Users\t95kf8j\bloc>npm install -g
npm WARN addRemoteGit Error: Command failed: git -c core.longpaths=true config -
-get remote.origin.url
npm WARN addRemoteGit
npm WARN addRemoteGit     at ChildProcess.exithandler (child_process.js:213:12)
npm WARN addRemoteGit     at emitTwo (events.js:87:13)
npm WARN addRemoteGit     at ChildProcess.emit (events.js:172:7)
npm WARN addRemoteGit     at maybeClose (internal/child_process.js:827:16)
npm WARN addRemoteGit     at Process.ChildProcess._handle.onexit (internal/child
_process.js:211:5)
npm WARN addRemoteGit  kejace/eth-lightwallet resetting remote C:\Program\_git-r
emotes\git-github-com-kejace-eth-lightwallet-git-0ab8a125 because of error: { [E
rror: Command failed: git -c core.longpaths=true config --get remote.origin.url
npm WARN addRemoteGit  ]
npm WARN addRemoteGit   killed: false,
npm WARN addRemoteGit   code: 1,
npm WARN addRemoteGit   signal: null,
npm WARN addRemoteGit   cmd: 'git -c core.longpaths=true config --get remote.ori
gin.url' }
npm WARN addRemoteGit Error: Command failed: git -c core.longpaths=true config -
-get remote.origin.url
npm WARN addRemoteGit
npm WARN addRemoteGit     at ChildProcess.exithandler (child_process.js:213:12)
npm WARN addRemoteGit     at emitTwo (events.js:87:13)
npm WARN addRemoteGit     at ChildProcess.emit (events.js:172:7)
npm WARN addRemoteGit     at maybeClose (internal/child_process.js:827:16)
npm WARN addRemoteGit     at Process.ChildProcess._handle.onexit (internal/child
_process.js:211:5)
npm WARN addRemoteGit  kejace/ethereumjs-tx resetting remote C:\Program\_git-rem
otes\git-github-com-kejace-ethereumjs-tx-git-398d22da because of error: { [Error
: Command failed: git -c core.longpaths=true config --get remote.origin.url
npm WARN addRemoteGit  ]
npm WARN addRemoteGit   killed: false,
npm WARN addRemoteGit   code: 1,
npm WARN addRemoteGit   signal: null,
npm WARN addRemoteGit   cmd: 'git -c core.longpaths=true config --get remote.ori
gin.url' }
npm WARN addRemoteGit Error: Command failed: git -c core.longpaths=true config -
-get remote.origin.url
npm WARN addRemoteGit
npm WARN addRemoteGit     at ChildProcess.exithandler (child_process.js:213:12)
npm WARN addRemoteGit     at emitTwo (events.js:87:13)
npm WARN addRemoteGit     at ChildProcess.emit (events.js:172:7)
npm WARN addRemoteGit     at maybeClose (internal/child_process.js:827:16)
npm WARN addRemoteGit     at Socket.<anonymous> (internal/child_process.js:319:1
1)
npm WARN addRemoteGit     at emitOne (events.js:77:13)
npm WARN addRemoteGit     at Socket.emit (events.js:169:7)
npm WARN addRemoteGit     at Pipe._onclose (net.js:477:12)
npm WARN addRemoteGit  kejace/ethereumjs-util resetting remote C:\Program\_git-r
emotes\git-github-com-kejace-ethereumjs-util-git-f306c434 because of error: { [E
rror: Command failed: git -c core.longpaths=true config --get remote.origin.url
npm WARN addRemoteGit  ]
npm WARN addRemoteGit   killed: false,
npm WARN addRemoteGit   code: 1,
npm WARN addRemoteGit   signal: null,
npm WARN addRemoteGit   cmd: 'git -c core.longpaths=true config --get remote.ori
gin.url' }
_
```



```
C:\Users\t95kf8j\AppData\Roaming\npm\bloc -> C:\Users\t95kf8j\AppData\Roaming\np
m\node_modules\blockapps-bloc\bin\main.js
blockapps-bloc@1.2.0 C:\Users\t95kf8j\AppData\Roaming\npm\node_modules\blockapps
-bloc
├── deepmerge@0.2.10
├── enum@2.3.0
├── bignumber.js@2.4.0
├── pkginfo@0.4.0
├── map-stream@0.0.6
├── mustache@2.2.1
├── cookie-parser@1.4.3 (cookie-signature@1.0.6, cookie@0.3.1)
├── chalk@1.1.3 (escape-string-regexp@1.0.5, ansi-styles@2.2.1, supports-color@2
.0.0, strip-ansi@3.0.1, has-ansi@2.0.0)
├── minimatch@3.0.2 (brace-expansion@1.1.5)
├── event-stream@3.3.4 (pause-stream@0.0.11, duplexer@0.1.1, stream-combiner@0.0
.4, from@0.1.3, map-stream@0.1.0, split@0.3.3, through@2.3.8)
├── bluebird@2.10.2
├── morgan@1.7.0 (on-headers@1.0.1, basic-auth@1.0.4, depd@1.1.0, on-finished@2.
3.0, debug@2.2.0)
├── express-session@1.14.0 (cookie-signature@1.0.6, utils-merge@1.0.0, on-header
s@1.0.1, parseurl@1.3.1, cookie@0.3.1, depd@1.1.0, crc@3.4.0, uid-safe@2.1.1, de
bug@2.2.0)
├── mkdirp@0.5.1 (minimist@0.0.8)
├── through2@2.0.1 (xtend@4.0.1, readable-stream@2.0.6)
├── body-parser@1.15.2 (content-type@1.0.2, bytes@2.4.0, depd@1.1.0, qs@6.2.0, o
n-finished@2.3.0, raw-body@2.1.7, http-errors@1.5.0, debug@2.2.0, iconv-lite@0.4
.13, type-is@1.6.13)
├── readdirp@2.1.0 (set-immediate-shim@1.0.1, graceful-fs@4.1.4, readable-stream
@2.1.4)
├── yargs@3.32.0 (decamelize@1.2.0, y18n@3.2.1, camelcase@2.1.1, window-size@0.1
.4, cliui@3.2.0, string-width@1.0.1, os-locale@1.4.0)
├── express@4.14.0 (escape-html@1.0.3, array-flatten@1.1.1, utils-merge@1.0.0, c
ookie-signature@1.0.6, content-type@1.0.2, encodeurl@1.0.1, methods@1.1.2, parse
url@1.3.1, cookie@0.3.1, range-parser@1.2.0, vary@1.1.0, content-disposition@0.5
.1, serve-static@1.11.1, path-to-regexp@0.1.7, etag@1.7.0, merge-descriptors@1.0
.1, fresh@0.3.0, depd@1.1.0, qs@6.2.0, on-finished@2.3.0, finalhandler@0.5.0, de
bug@2.2.0, proxy-addr@1.1.2, type-is@1.6.13, send@0.14.1, accepts@1.3.3)
├── blockapps-js@3.1.2 (mnemonic@1.0.1, randombytes@2.0.3, js-sha3@0.5.2, rlp@1.
1.2, bn.js@4.11.5, elliptic@6.3.1)
├── js-yaml@3.6.1 (esprima@2.7.2, argparse@1.0.7)
├── prompt@0.2.14 (revalidator@0.1.8, read@1.0.7, utile@0.2.1, winston@0.8.3)
├── vinyl-fs@2.4.3 (merge-stream@1.0.0, vali-date@1.0.0, is-valid-glob@0.3.0, ob
ject-assign@4.1.0, graceful-fs@4.1.4, lazystream@1.0.0, strip-bom-stream@1.0.0,
strip-bom@2.0.0, lodash.isequal@4.2.0, through2-filter@2.0.0, gulp-sourcemaps@1.
6.0, vinyl@1.1.1, readable-stream@2.1.4, duplexify@3.4.5, glob-stream@5.3.2)
├── request@2.73.0 (tunnel-agent@0.4.3, aws-sign2@0.6.0, forever-agent@0.6.1, oa
uth-sign@0.8.2, caseless@0.11.0, is-typedarray@1.0.0, stringstream@0.0.5, aws4@1
.4.1, isstream@0.1.2, json-stringify-safe@5.0.1, extend@3.0.0, tough-cookie@2.2.
2, qs@6.2.0, node-uuid@1.4.7, combined-stream@1.0.5, mime-types@2.1.11, form-dat
a@1.0.0-rc4, bl@1.1.2, hawk@3.1.3, http-signature@1.1.1, har-validator@2.0.6)
├── express-handlebars@2.0.1 (graceful-fs@3.0.8, promise@6.1.0, object.assign@1.
1.1, glob@5.0.15, handlebars@3.0.3)
├── insight@0.7.0 (object-assign@4.1.0, async@1.5.2, tough-cookie@2.2.2, lodash.
debounce@3.1.1, configstore@1.4.0, os-name@1.0.3, inquirer@0.10.1)
├── eslint@2.13.1 (ignore@3.1.3, path-is-absolute@1.0.0, pluralize@1.2.1, imurmu
rhash@0.1.4, path-is-inside@1.0.1, globals@9.9.0, estraverse@4.2.0, esutils@2.0.
2, strip-json-comments@1.0.4, progress@1.1.8, text-table@0.2.0, user-home@2.0.0,
 is-resolvable@1.0.0, debug@2.2.0, doctrine@1.2.2, levn@0.3.0, optionator@0.8.1,
 json-stable-stringify@1.0.1, glob@7.0.5, require-uncached@1.0.2, shelljs@0.6.0,
 concat-stream@1.5.1, espree@3.1.6, inquirer@0.12.0, is-my-json-valid@2.13.1, fi
le-entry-cache@1.2.4, table@3.7.8, es6-map@0.1.4, lodash@4.13.1, escope@3.6.0)
├── ethereumjs-util@2.0.3 (bn.js@3.3.0, rlp@2.0.0, browserify-sha3@0.0.0, ellipt
ic@5.2.1)
├── ethereumjs-tx@0.6.7 (ethereum-common@0.0.10, browserify-sha3@0.0.2, secp256k
1-browserify@0.0.0, ethereumjs-util@2.0.3)
└── eth-lightwallet@0.1.0 (bignumber.js@2.0.7, rlp@1.1.2, crypto-js@3.1.6, ellip
tic@3.1.0, bitcore-mnemonic@0.13.2, bitcore@0.13.5, web3@0.13.0, ethereumjs-tx@0
.6.7)

C:\Users\t95kf8j\bloc>
```

2) Generate a new blockchain app
    Bloc init

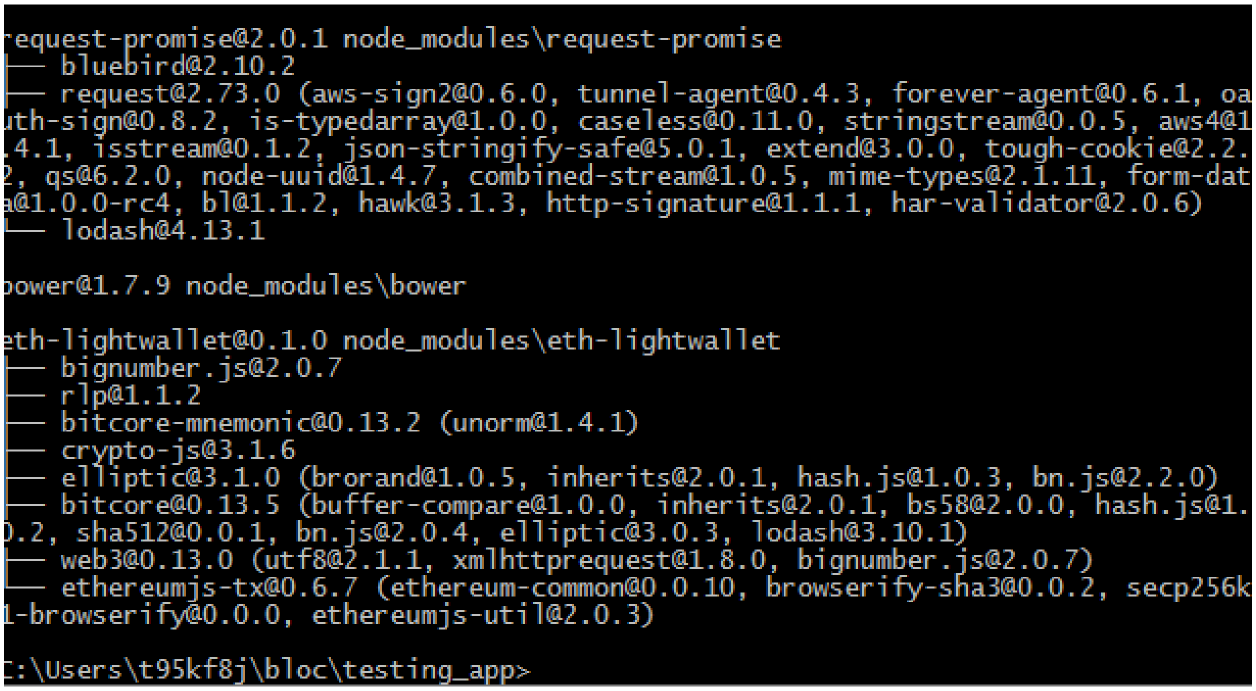


```
C:\Users\t95kf8j\bloc>bloc init

prompt: Enter the name of your app:  testing_app
prompt: Enter your name:
error:   Invalid input for Enter your name
prompt: Enter your name:  arundhati
prompt: Enter your email so BlockApps can reach you:
prompt: apiURL:  (http://strato-dev4.blockapps.net)
prompt: Enter the blockchain profile you wish to use.  Options: strato-dev, ethe
reum:  (strato-dev) ethereum
Wrote: C:\Users\t95kf8j\bloc\testing_app\.bowerrc
Wrote: C:\Users\t95kf8j\bloc\testing_app\app.js
Wrote: C:\Users\t95kf8j\bloc\testing_app\marko-taglib.json
Wrote: C:\Users\t95kf8j\bloc\testing_app\package.json
Wrote: C:\Users\t95kf8j\bloc\testing_app\gulpfile.js
Wrote: C:\Users\t95kf8j\bloc\testing_app\bower.json
Wrote: C:\Users\t95kf8j\bloc\testing_app\test\common.js
Wrote: C:\Users\t95kf8j\bloc\testing_app\test\top.js
```

Enter details as needed.
Name of app
Name
For blockchain app: ethereum

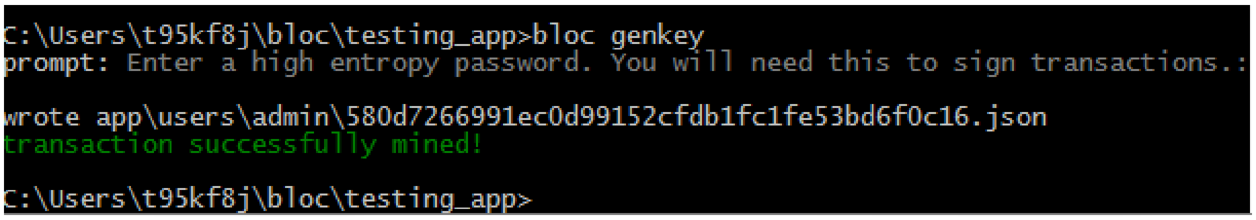
3) After that is completed:
    cd testing_app && npm install

You should results like these at the end.



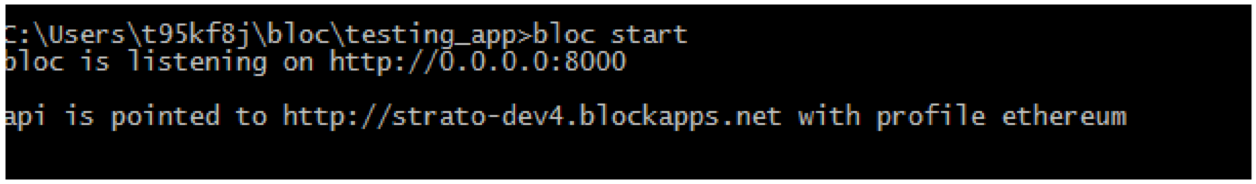
```
equest-promise@2.0.1 node_modules\request-promise
├── bluebird@2.10.2
├── request@2.73.0 (aws-sign2@0.6.0, tunnel-agent@0.4.3, forever-agent@0.6.1, oa
uth-sign@0.8.2, is-typedarray@1.0.0, caseless@0.11.0, stringstream@0.0.5, aws4@1
.4.1, isstream@0.1.2, json-stringify-safe@5.0.1, extend@3.0.0, tough-cookie@2.2.
2, qs@6.2.0, node-uuid@1.4.7, combined-stream@1.0.5, mime-types@2.1.11, form-dat
a@1.0.0-rc4, bl@1.1.2, hawk@3.1.3, http-signature@1.1.1, har-validator@2.0.6)
└── lodash@4.13.1

ower@1.7.9 node_modules\bower

th-lightwallet@0.1.0 node_modules\eth-lightwallet
├── bignumber.js@2.0.7
├── rlp@1.1.2
├── bitcore-mnemonic@0.13.2 (unorm@1.4.1)
├── crypto-js@3.1.6
├── elliptic@3.1.0 (brorand@1.0.5, inherits@2.0.1, hash.js@1.0.3, bn.js@2.2.0)
├── bitcore@0.13.5 (buffer-compare@1.0.0, inherits@2.0.1, bs58@2.0.0, hash.js@1.
.2, sha512@0.0.1, bn.js@2.0.4, elliptic@3.0.3, lodash@3.10.1)
├── web3@0.13.0 (utf8@2.1.1, xmlhttprequest@1.8.0, bignumber.js@2.0.7)
└── ethereumjs-tx@0.6.7 (ethereum-common@0.0.10, browserify-sha3@0.0.2, secp256k
1-browserify@0.0.0, ethereumjs-util@2.0.3)

:\Users\t95kf8j\bloc\testing_app>
```

4) *Bloc GenKey*



```
:\Users\t95kf8j\bloc\testing_app>bloc genkey
prompt: Enter a high entropy password. You will need this to sign transactions.:

wrote app\users\admin\580d7266991ec0d99152cfdb1fc1fe53bd6f0c16.json
transaction successfully mined!

:\Users\t95kf8j\bloc\testing_app>
```

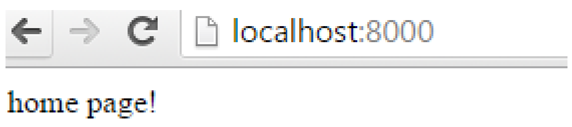
5) *Bloc Start*



```
:\Users\t95kf8j\bloc\testing_app>bloc start
loc is listening on http://0.0.0.0:8000

api is pointed to http://strato-dev4.blockapps.net with profile ethereum
```

This means that the bloc has started listening on your localhost.

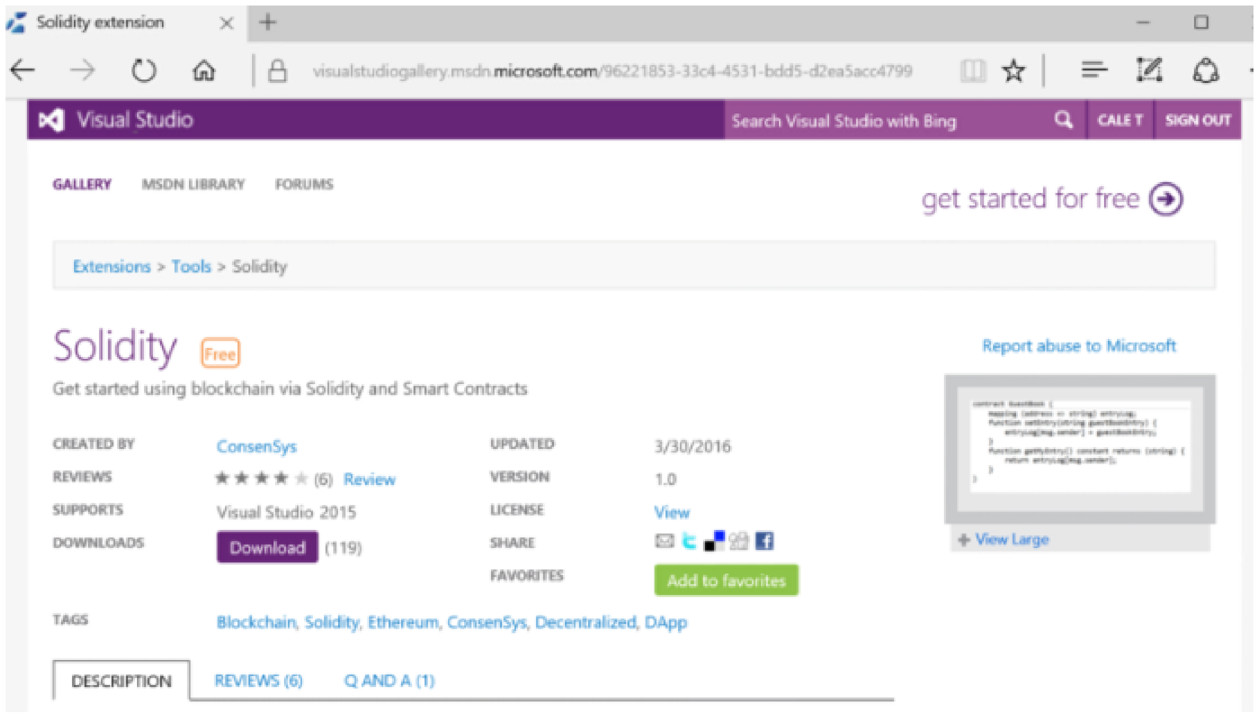
If you see the contents of localhost:8000 in your browser, you should see something like this.


localhost:8000

home page!

Installing Solidity:
We need to install Solidity plugin for Visual Studio to be able to write smart contracts.

Download the Solidity plugin for Visual Studio from here.
https://visualstudiogallery.msdn.microsoft.com/96221853-33c4-4531-bdd5-d2ea5acc4799/,
(You will need Visual Studio 2015 or above)


Solidity extension
visualstudiogallery.msdn.microsoft.com/96221853-33c4-4531-bdd5-d2ea5acc4799
Visual Studio
Search Visual Studio with Bing
SIGN OUT
GALLERY
MSDN LIBRARY
FORUMS
get started for free
Extensions > Tools > Solidity
Report abuse to Microsoft
View Large
Download
Add to favorites
DESCRIPTION
REVIEWS (6)
Q AND A (1)

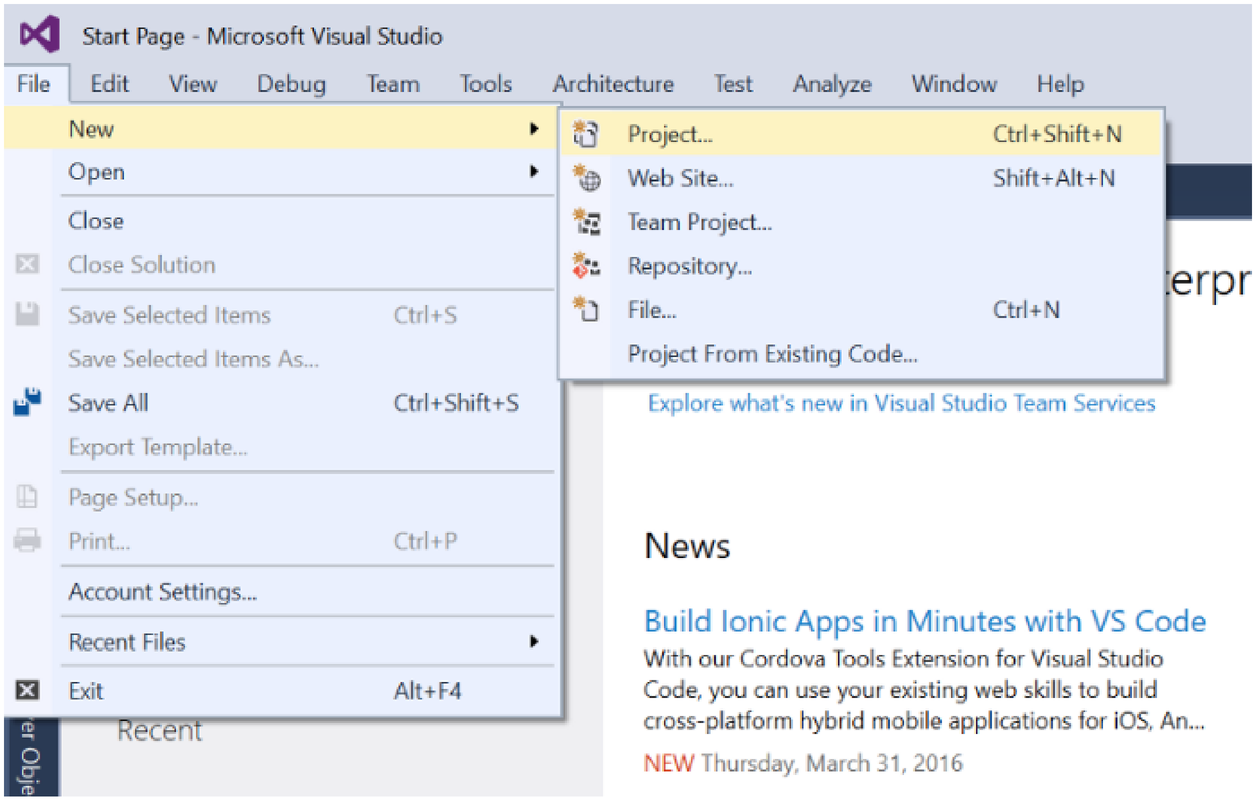
Solidity Free

Get started using blockchain via Solidity and Smart Contracts

CREATED BY	ConsenSys	UPDATED	3/30/2016
REVIEWS	★★★★☆ (6) Review	VERSION	1.0
SUPPORTS	Visual Studio 2015	LICENSE	View
DOWNLOADS	Download (119)	SHARE	
		FAVORITES	Add to favorites
TAGS	Blockchain, Solidity, Ethereum, ConsenSys, Decentralized, DApp		

Creating an App with Smart Contract to Run on Ethereum:

1. From the **File** menu, click **New Project**

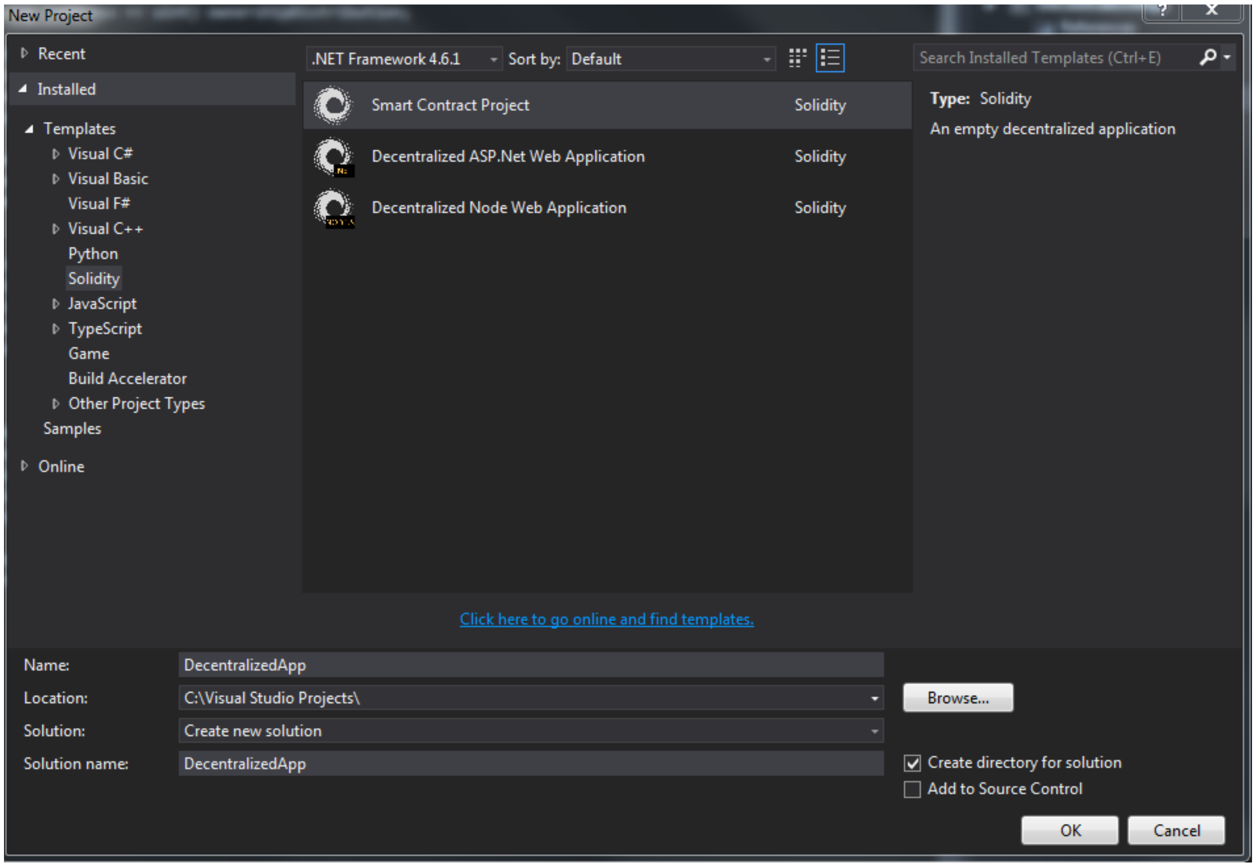


Build Ionic Apps in Minutes with VS Code

With our Cordova Tools Extension for Visual Studio Code, you can use your existing web skills to build cross-platform hybrid mobile applications for iOS, An...

NEW Thursday, March 31, 2016

2. In the **New Project** dialog box, click on **Solidity** under Installed Templates, and then select **Smart Contract Project**.



New Project

- Recent
- Installed
  - Templates
    - Visual C#
    - Visual Basic
    - Visual F#
    - Visual C++
    - Python
    - Solidity
    - JavaScript
    - TypeScript
    - Game
    - Build Accelerator
    - Other Project Types
  - Samples
- Online

.NET Framework 4.6.1 Sort by: Default

Search Installed Templates (Ctrl+E)

Smart Contract Project	Solidity
Decentralized ASP.Net Web Application	Solidity
Decentralized Node Web Application	Solidity

Type: Solidity

An empty decentralized application

Click here to go online and find templates.

Name: DecentralizedApp

Location: C:\Visual Studio Projects\

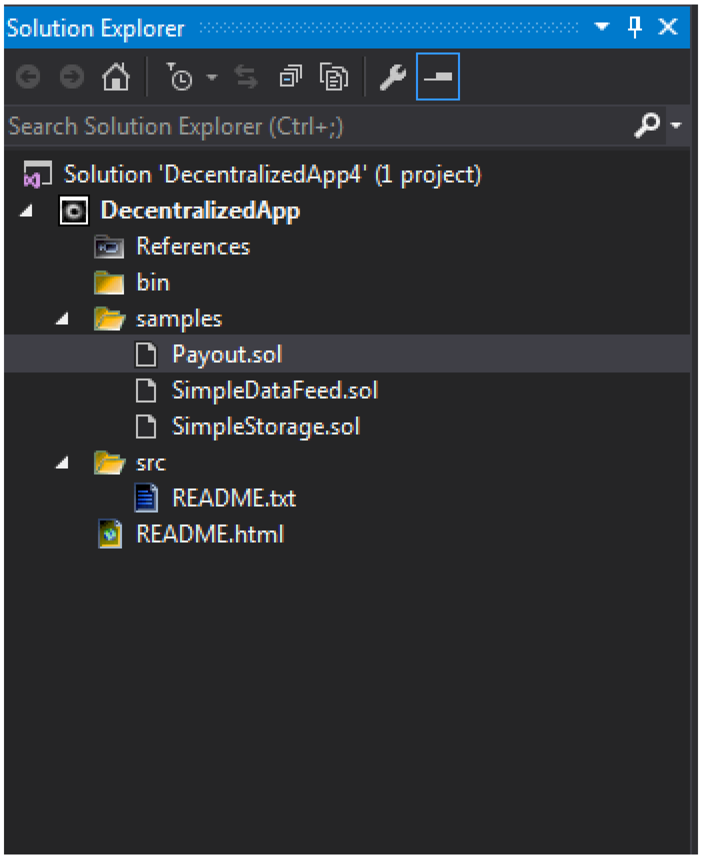
Solution: Create new solution

Solution name: DecentralizedApp

[✓] Create directory for solution

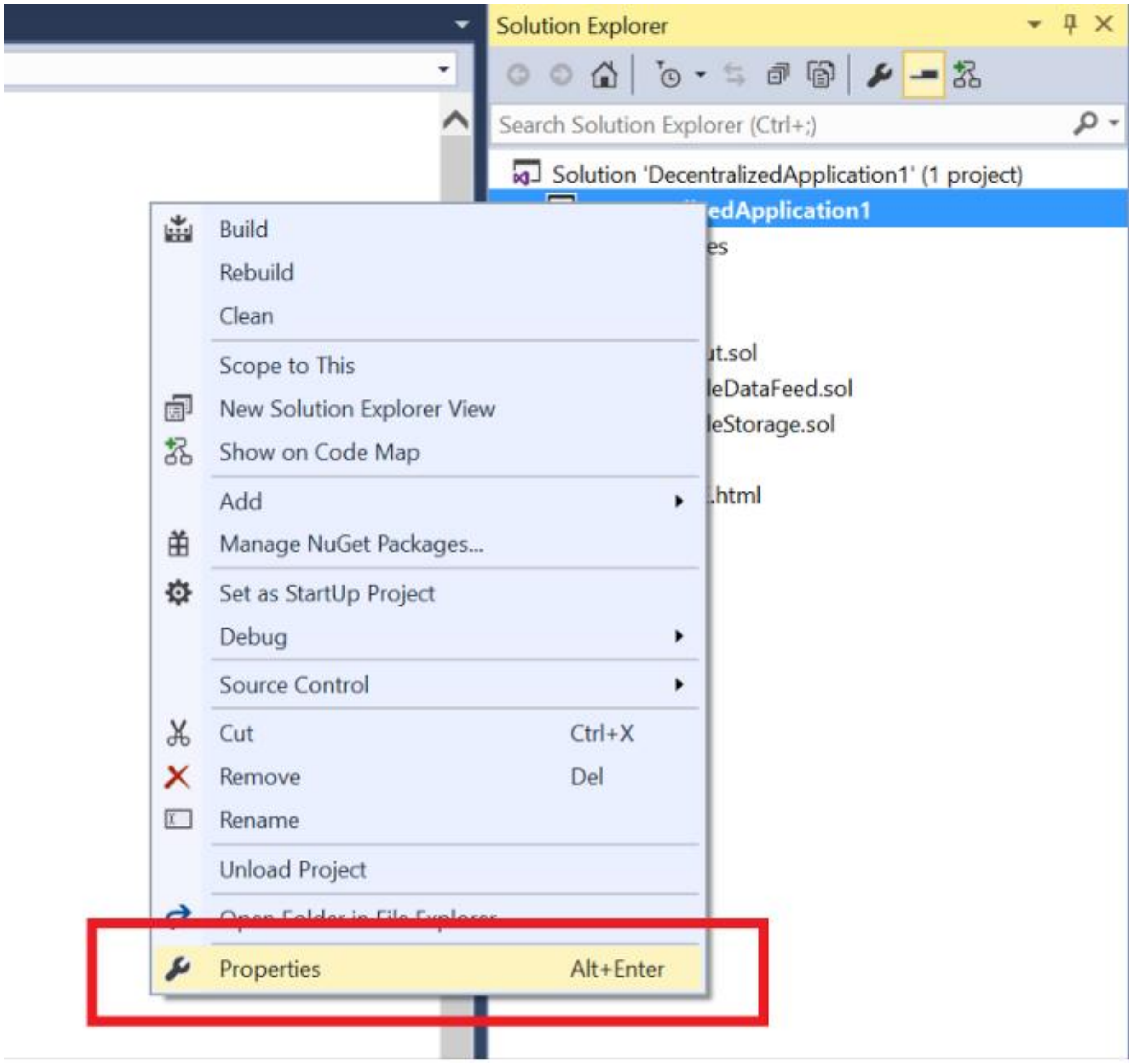
[ ] Add to Source Control

3. The sample project is created with 3 sample smart contracts.

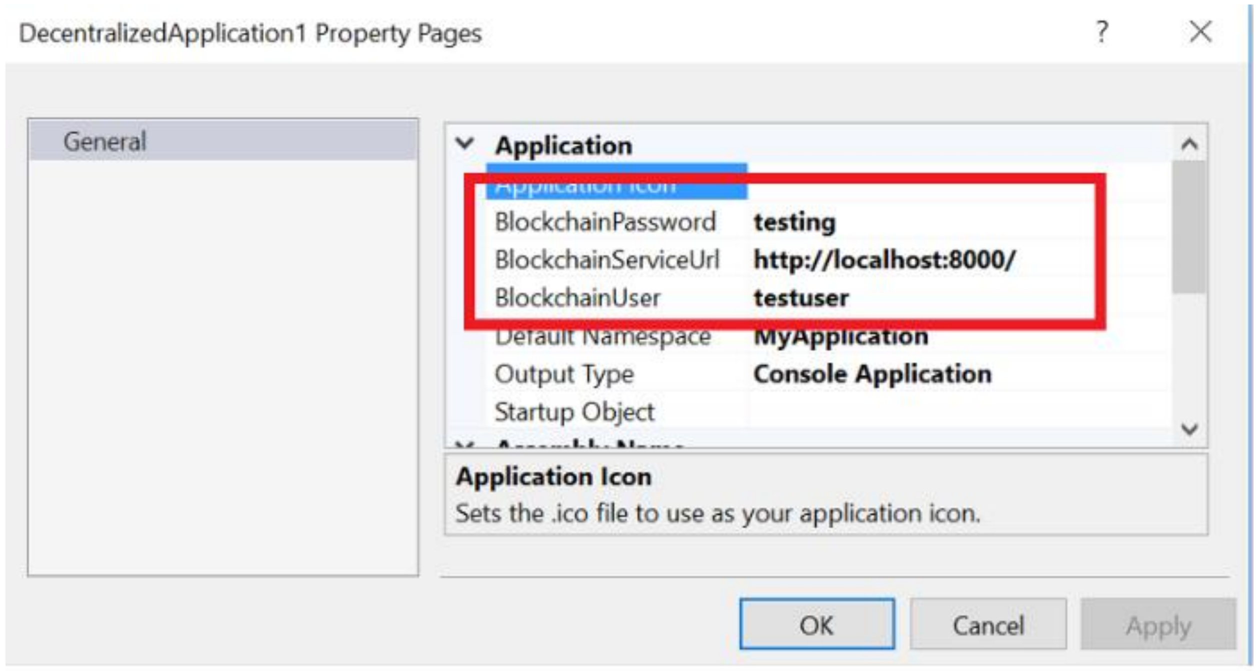


As per this document, we already have a keyserver running on our **localhost:8000**. Now we will try to run a sample smart contract on it.

4. Right Click on **Project** in the Solution Explorer.



5. This dialog will show the configuration values that have been set by default.

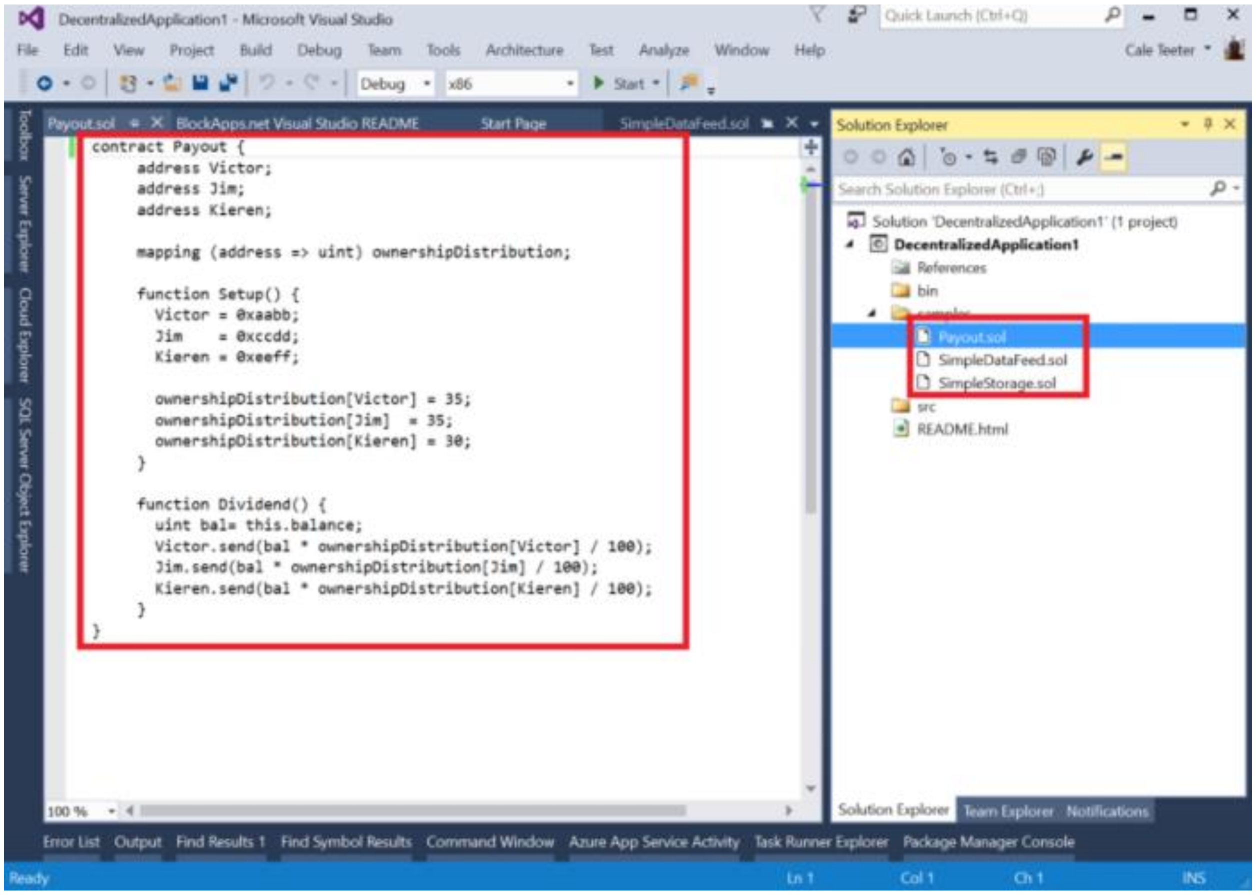


Application Icon	
BlockchainPassword	testing
BlockchainServiceUrl	http://localhost:8000/
BlockchainUser	testuser
Default Namespace	MyApplication
Output Type	Console Application
Startup Object	

Application Icon
Sets the .ico file to use as your application icon.

Deploying the Smart Contract:

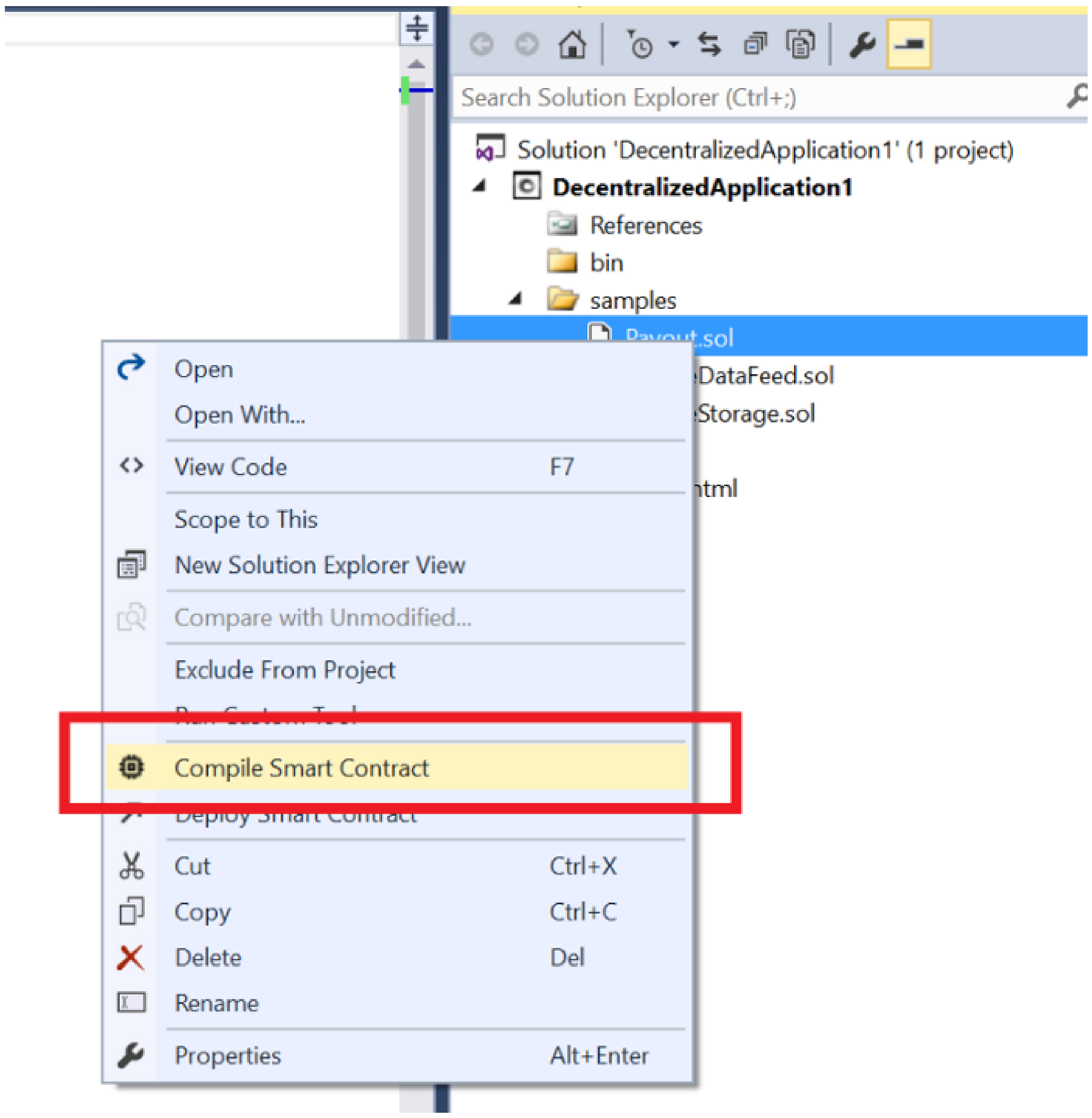
1. By default, we have a particular folder structure for the sample project in Visual Studio. We will deploy a Smart Contract, Payments, to our keyserver.


DecentralizedApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Payout.sol
BlockApps.net Visual Studio README
Start Page
SimpleDataFeed.sol
Solution Explorer
Solution 'DecentralizedApplication1' (1 project)
DecentralizedApplication1
References
bin
samples
Payout.sol
SimpleDataFeed.sol
SimpleStorage.sol
src
README.html

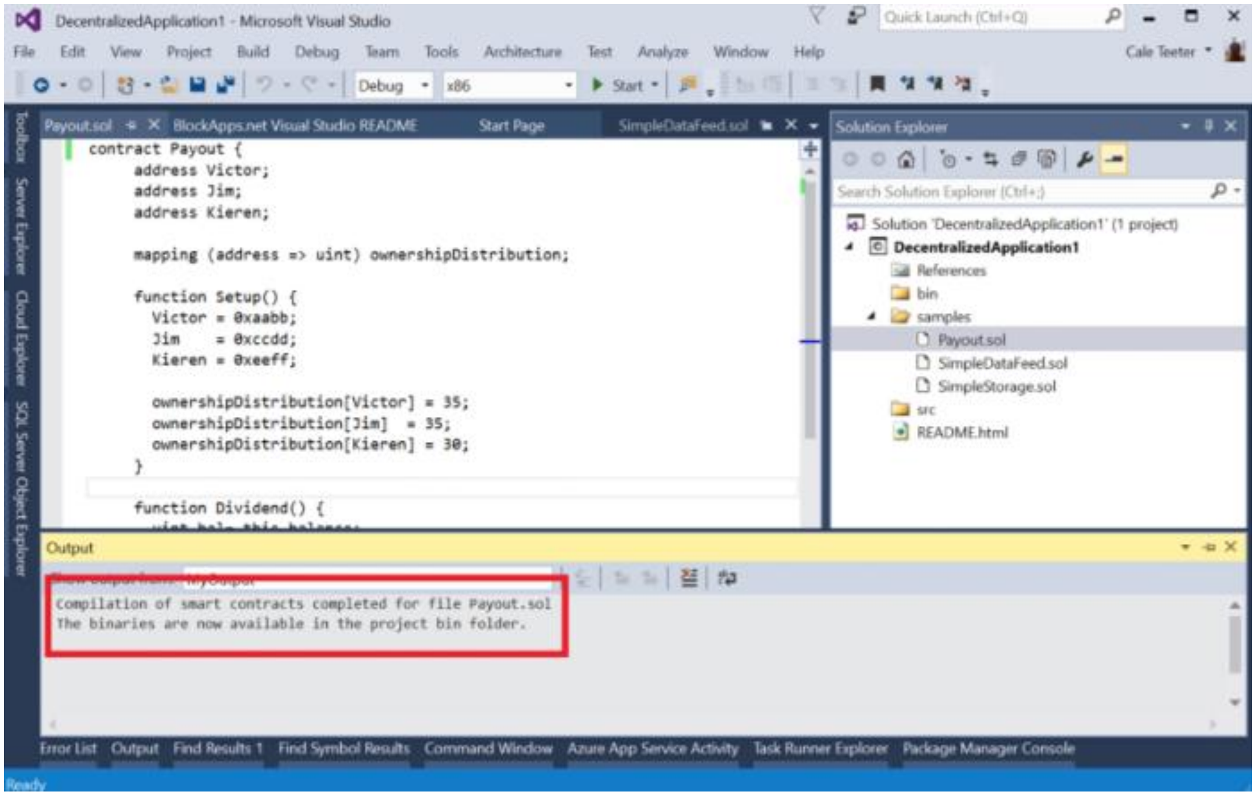
```
contract Payout {
      address Victor;
      address Jim;
      address Kieren;

      mapping (address => uint) ownershipDistribution;

      function Setup() {
        Victor = 0xaabb;
        Jim    = 0xccdd;
        Kieren = 0xeeff;

        ownershipDistribution[Victor] = 35;
        ownershipDistribution[Jim]  = 35;
        ownershipDistribution[Kieren] = 30;
      }

      function Dividend() {
        uint bal= this.balance;
        Victor.send(bal * ownershipDistribution[Victor] / 100);
        Jim.send(bal * ownershipDistribution[Jim] / 100);
        Kieren.send(bal * ownershipDistribution[Kieren] / 100);
      }
}
```

2. Compile the smart contract.

Right click on Payments.sol from the Solution Explorer. Click on 'Compile Smart Contract'.



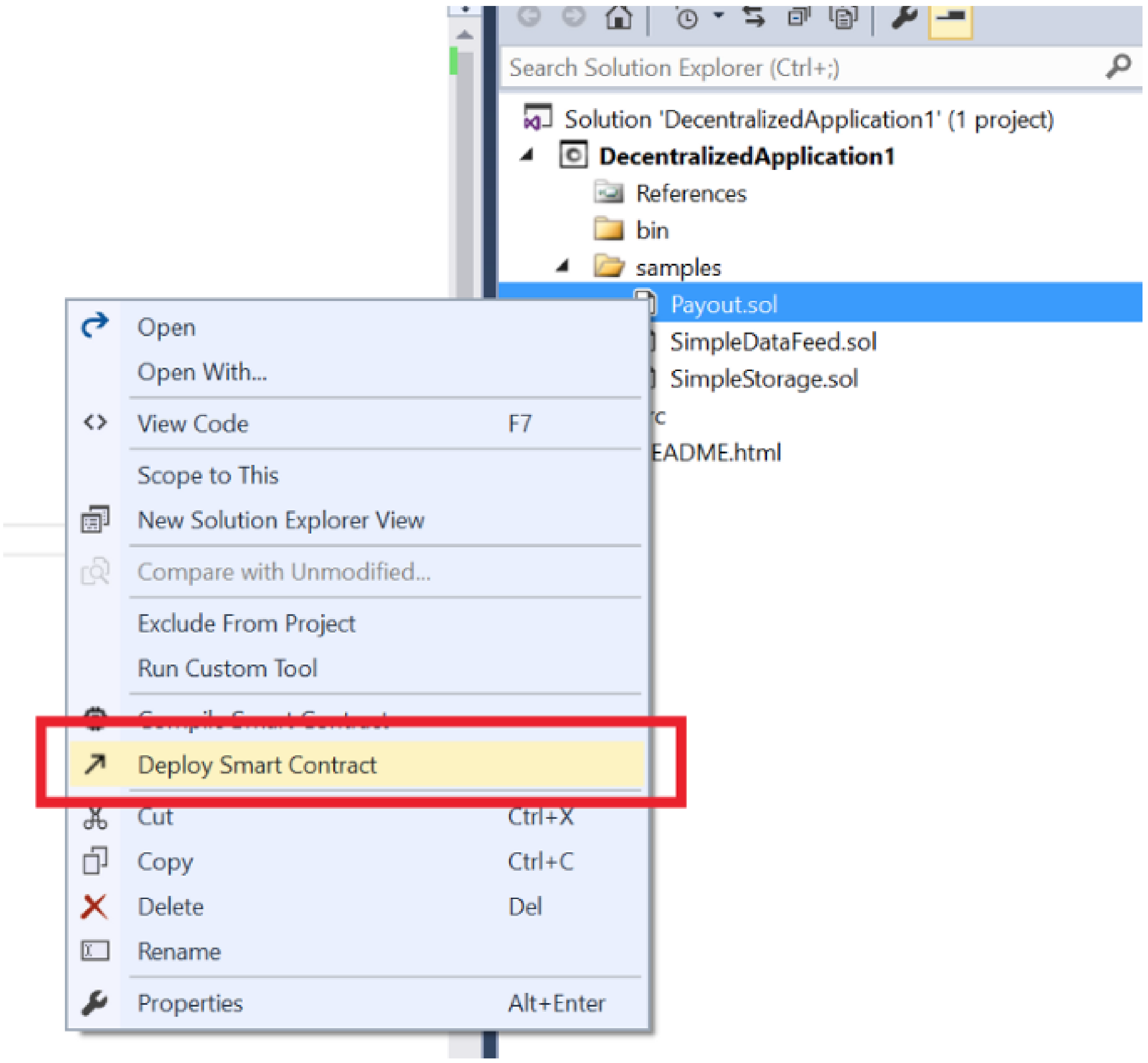
3. The output after compilation can be found in the output window.


DecentralizedApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Payout.sol
BlockApps.net Visual Studio README
Start Page
SimpleDataFeed.sol
Solution Explorer
Solution 'DecentralizedApplication1' (1 project)
DecentralizedApplication1
References
bin
samples
Payout.sol
SimpleDataFeed.sol
SimpleStorage.sol
src
README.html
Output
Error List Output Find Results 1 Find Symbol Results Command Window Azure App Service Activity Task Runner Explorer Package Manager Console
Ready

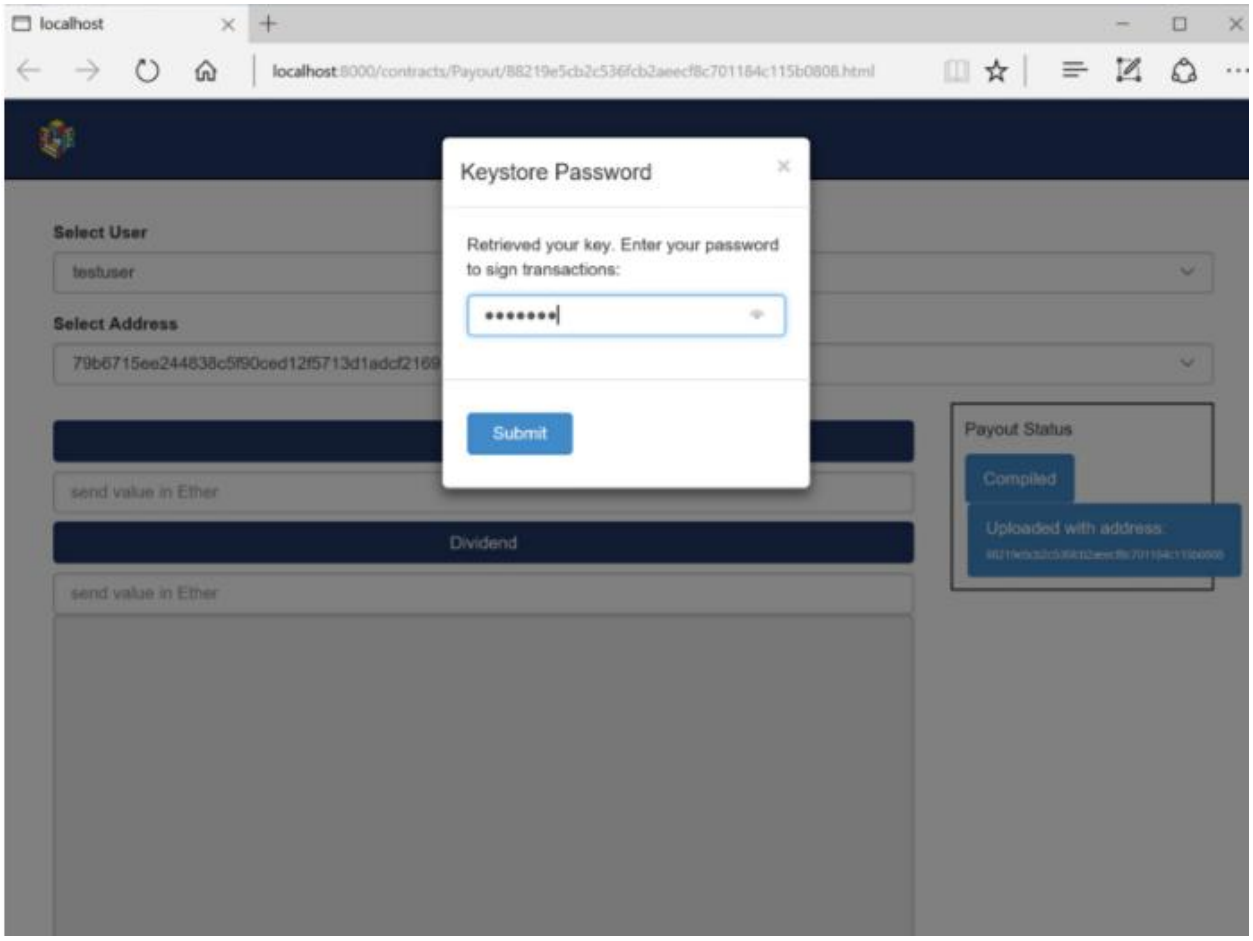
```
contract Payout {
    address Victor;
    address Jim;
    address Kieren;

    mapping (address => uint) ownershipDistribution;

    function Setup() {
      Victor = 0xaabb;
      Jim    = 0xccdd;
      Kieren = 0xeeff;

      ownershipDistribution[Victor] = 35;
      ownershipDistribution[Jim]  = 35;
      ownershipDistribution[Kieren] = 30;
    }

    function Dividend() {
```

```
Compilation of smart contracts completed for file Payout.sol
The binaries are now available in the project bin folder.
```

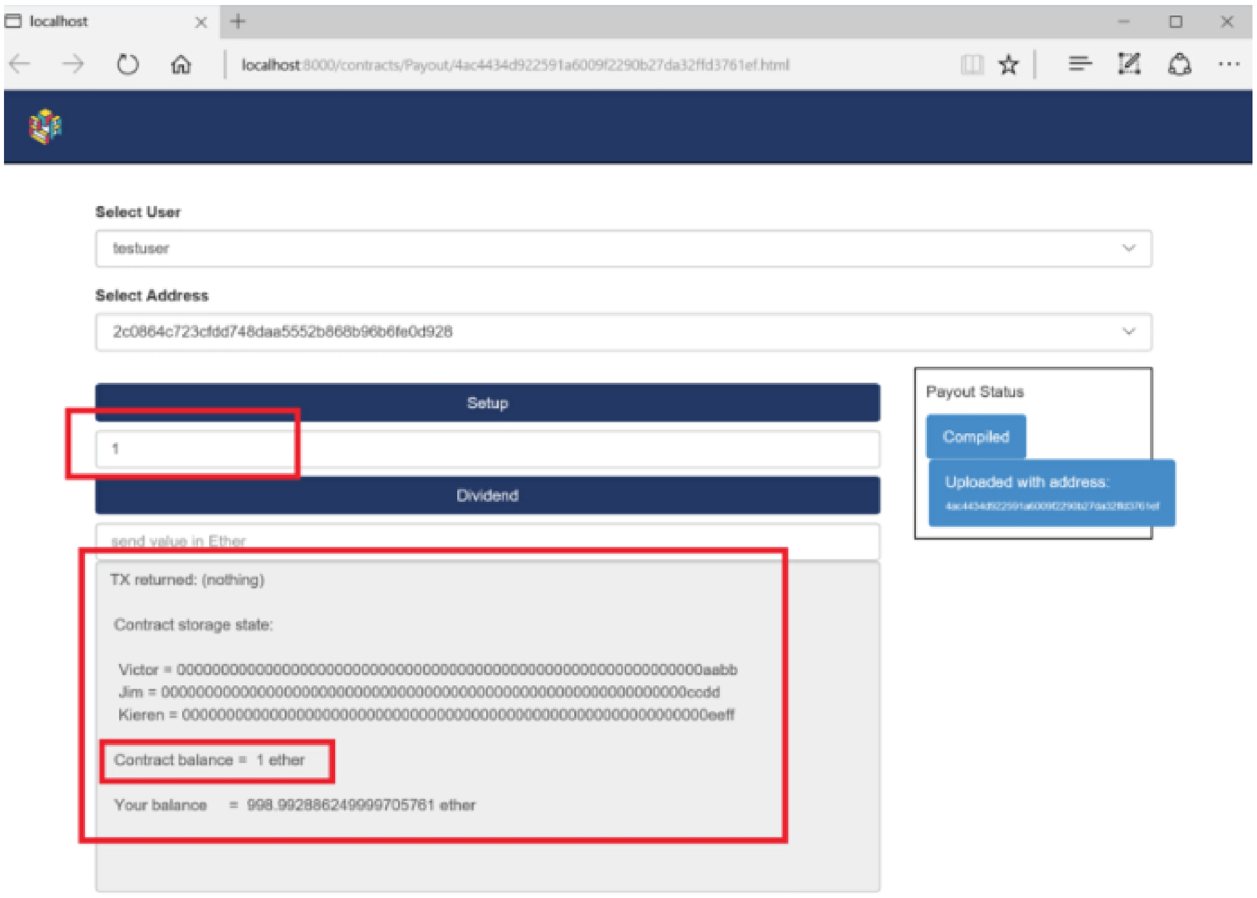
4. **Deploying the Smart Contract:**
   Right click on Payment.sol and click on 'Deploy Smart Contract'.



5. This smart contract will be deployed and a new browser window will open Enter the password for the bloc server (The same password we had generated using ‘bloc genkey’ command. By default, this password is ‘testing’ if bloc genkey was not implemented).


localhost
localhost:8000/contracts/Payout/88219e5cb2c536fcb2aeecf8c701184c115b0808.html
Select User
testuser
Select Address
79b6715ee244838c5f90ced12f5713d1adcf2169
send value in Ether
Dividend
send value in Ether
Payout Status
Compiled
Uploaded with address:
Keystore Password
Retrieved your key. Enter your password to sign transactions:
Submit

6. You can then exercise the contract from the browser.


localhost
localhost:8000/contracts/Payout/4ac4434d922591a6009f2290b27da32ffd3761ef.html

Select User

testuser

Select Address

2c0864c723cfdd748daa5552b868b96b6fe0d928

Setup

1

Dividend

send value in Ether

```
TX returned: (nothing)

Contract storage state:

Victor = 000000000000000000000000000000000000000000000000000000000000aabb
Jim = 000000000000000000000000000000000000000000000000000000000000ccdd
Kieren = 000000000000000000000000000000000000000000000000000000000000eeff

Contract balance =  1 ether

Your balance    =  998.992886249999705761 ether
```

Payout Status

Compiled

Uploaded with address:
4ac4434d922591a6009f2290b27da32ffd3761ef