

## **Blockchain Basics Workshop**

### **Overview**

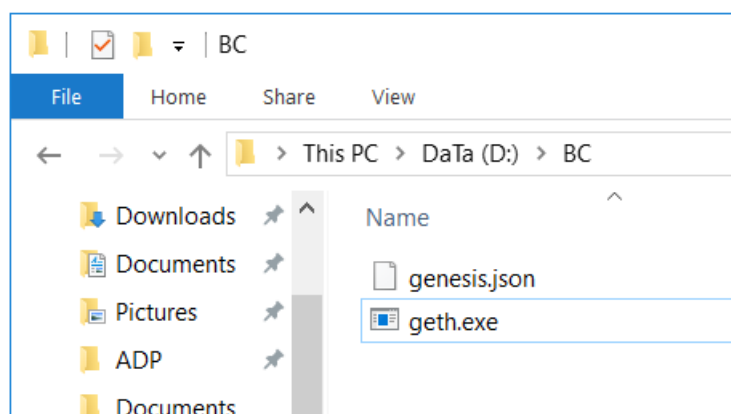
0. ขั้นตอนเตรียมการก่อนการติดตั้ง
1. การติดตั้งโหนด (2 โหนดบนเครื่องเดียวกัน)
2. การสำรวจรายละเอียดในโหนด
3. การเชื่อมต่อ Node1 และ Node2
4. แบบฝึกหัด: การสร้างและเชื่อมต่อ Node3
5. การสร้าง miner
6. การตรวจสอบเงินในบัญชีของโหนดต่างๆ
7. Transaction การโอนเงิน
8. การเชื่อมต่อโหนดผ่านเครือข่าย
9. การติดตั้งโหนดและใช้งานบล็อกเชนบนคลาวด์

### **Lab 0: การเตรียมการก่อนการติดตั้ง**

วัตถุประสงค์ เพื่อเตรียม environment ของการติดตั้งและการทำ workshop ในแล็บต่อไป

ขั้นตอนที่ 1 ให้สร้าง folder ชื่อ BC ในไดรฟ์ D: ดังนี้ D:\BC

ขั้นตอนที่ 2 ให้ copy ไฟล์ geth.exe และ genesis.json ที่ได้รับจากวิทยากรมาไว้ใน folder D:\BC

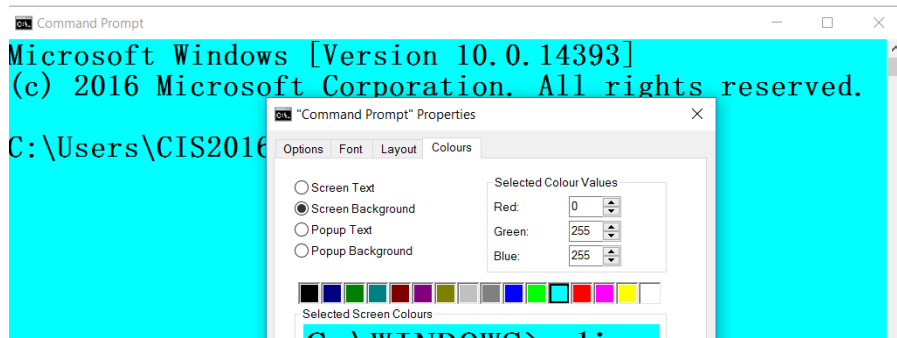


## Lab 1: การติดตั้งโหนด (Node Installation)

วัตถุประสงค์ ในแล็บนี้ เราจะติดตั้ง Node พร้อมสร้างบัญชีใหม่ จำนวน 2 โหนด บนเครื่องเดียวกัน ซึ่งทั้งสองโหนดจะยังไม่รู้จักกัน

### Lab 1.1 การสร้างโหนดที่ 1

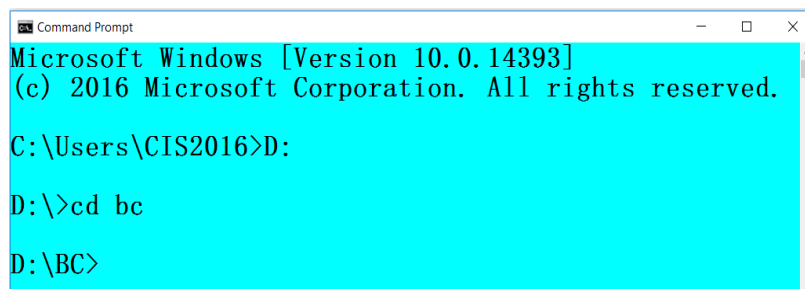
ขั้นตอนที่ 1 เปิด command prompt ขึ้นมา + เซตสี text + background



ขั้นตอนที่ 2 เปลี่ยนไดเรกทอรีไปยังที่เก็บโปรแกรม geth.exe และ ไฟล์ genesis.json

คำสั่ง D: เพื่อเปลี่ยนไปยัง drive D:\>

คำสั่ง cd BC เพื่อเข้าไปยัง folder D:\BC>

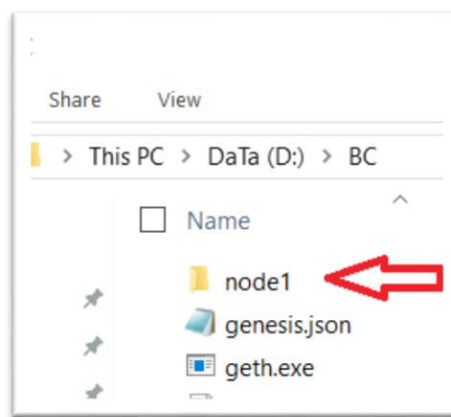


ขั้นตอนที่ 3 สร้างโหนดชื่อ node1 และโพลเดอร์ node1 เพื่อเก็บข้อมูลของโหนด โดยอาศัยรายละเอียดที่กำหนดใน genesis.json

```
geth --networkid 100 --identity node1 --verbosity 3 --nodiscover --  
nat none --datadir node1 init genesis.json
```

```
Command Prompt
D:\BC>geth --networkid 100 --identity node1 --verbosity 3 --nodiscover --
nat none --datadir node1 init genesis.json
I0615 15:28:10.856879 node/config.go:445] Failed to start Ledger hub, dis
abling: libusb: not found [code -5]
I0615 15:28:10.858870 cmd/utills/flags.go:613] WARNING: No etherbase set a
nd no accounts found as default
I0615 15:28:10.858870 ethdb/database.go:83] Allotted 128MB cache and 1024 file
handles to D:\BC\node1\geth\chaindata
I0615 15:28:10.948532 ethdb/database.go:176] closed db:D:\BC\node1\geth\c
haindata
I0615 15:28:10.948532 ethdb/database.go:83] Allotted 128MB cache and 1024
file handles to D:\BC\node1\geth\chaindata
I0615 15:28:10.974642 cmd/geth/chaincmd.go:132] successfully wrote genesi
s block and/or chain rule set: 6e92f8b23bcd9df34dc813cfaf1d84b71beac80530
506b5d63a2df10fe23a660
D:\BC>
```

เมื่อทำสำเร็จเราจะพบ folder node1 ใน folder BC (สามารถใช้คำสั่ง dir ใน command prompt)



ขั้นตอนที่ 4 สร้างบัญชี และ address ของบัญชี

#### 4.1 เข้าสู่ geth prompt

```
geth --networkid 100 --identity node1 --verbosity 3 --nodiscover --
nat none --datadir node1 --rpc --rpcapi "web3, eth, personal" --
rpccorsdomain "*" --ipcpath node1/geth.ipc console
```

ถ้าสำเร็จ จะปรากฏเครื่องหมาย prompt ">"

4.2 สร้างบัญชีใหม่ พร้อมกำหนดรหัส (หมายเหตุ ต้องจำรหัสให้ดี เพื่อใช้ในขั้นตอนต่อไป)

```
>personal.newAccount("password1")
"0xd01cb382caf5eb782644b5e63ad3cdb100e313bf"
```

คำสั่งนี้จะได้บัญชีใหม่ พร้อมกับ address ของบัญชีใหม่นี้ โดยเราจะใช้ address นี้เมื่อต้องการระบุถึงบัญชีใหม่นี้

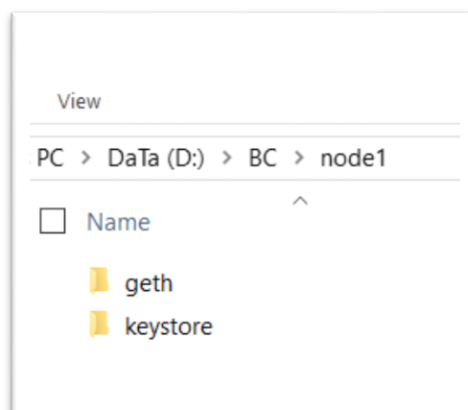
#### 4.3 เก็บ address ไว้ใน temp file

```
lab.txt - Notepad
File Edit Format View Help

Node: Node1
Password: password1
Account1:
"0x6b8f11e3bad36d18ad9fdb12291958c21d09550c"
```

ขั้นตอนที่ 5 การตรวจสอบ และสำรวจ Folder node1

5.1 ภายใน folder node1 เราจะพบ 2 folder ที่ถูกสร้างขึ้นมาคือ folder geth และ keystore



5.2 folder geth จะใช้สำหรับเก็บข้อมูลที่ ethereum ต้องใช้

5.3 folder keystore จะใช้เก็บ key ของ node1

### Lab 1.2 การสร้างโหนดที่ 2

ในการสร้างโหนดที่ 2 เราก็จะใช้คำสั่งเดียวกัน แต่มีการเปลี่ยนค่าบางตัว ดังต่อไปนี้  
ขั้นตอนที่ 1 เปิด command prompt ขึ้นมาอีกหน้าจอหนึ่ง (ควรกำหนดสีพื้นหลังให้แตกต่างกันเพื่อให้ง่ายต่อการแยกแยะโหนด 2 กับโหนด 1)

ขั้นตอนที่ 2 เปลี่ยนไดเรกทอรีไปยังที่เก็บไฟล์ genesis.json

ขั้นตอนที่ 3 สร้างโหนดและโพลเดอร์เพื่อเก็บข้อมูลของโหนด โดยอาศัยรายละเอียดที่กำหนดใน genesis.json

```
geth --networkid 100 --identity node2 --verbosity 3 --nodiscover --  
nat none --datadir node2 init genesis.json
```

ขั้นตอนที่ 4 สร้างบัญชีและ Address ของ node 2

4.1 เข้าสู่ geth prompt

```
geth --networkid 100 --identity node2 --verbosity 3 --nodiscover --  
nat none --datadir node2 --rpc --rpcapi "web3, eth, personal" --  
rpccorsdomain "*" --rpcport 2222 --port 2 --ipcpath node2/geth.ipc  
console
```

4.2 สร้างบัญชีใหม่ พร้อมกำหนดรหัส (หมายเหตุ ต้องจำรหัสให้ดี เพื่อใช้ในขั้นตอนต่อไป)

```
>personal.newAccount("password2")
```

```
"0xaa97b233dc3b903c064219a22ab1b762316b90f5"
```

4.3 เก็บ address ไว้ใน temp file

## Lab 2: การสำรวจรายละเอียดในโหนด

ขั้นตอนที่ 1 กลับมาที่โหนดที่ 1 ใน command prompt

ขั้นตอนที่ 2 ตรวจสอบจำนวน peer ที่โหนด 1 รู้จัก ด้วยคำสั่ง **net**

```
> net
{
  listening: true,
  peerCount: 0,
  version: "100",
  getListening: function(callback),
  getPeerCount: function(callback),
  getVersion: function(callback)
}
```

ขั้นตอนที่ 3 คำสั่ง **admin**

```
> admin
{
  datadir: "D:\\BC\\node1",
  nodeInfo: {
    enode:
"enode://c999aca89ef7a1af1c972ef7050a8cb846b478e5d8853b4d83c7b58b03d5ff6f49c470549ccc92e0495b9bc77d319401733a36575fbce4ec1de47080256fc130@[::]:30303?discport=0",
    id:
    "c999aca89ef7a1af1c972ef7050a8cb846b478e5d8853b4d83c7b58b03d5ff6f49c470549ccc92e0495b9bc77d319401733a36575fbce4ec1de47080256fc130",
    ip: "::",
    listenAddr: "[::]:30303",
    name: "Geth/v1.4.5-stable/windows/go1.6.2/node1",
    ports: {
      discovery: 0,
      listener: 30303
    },
  },
}
```

```
protocols: {
  eth: {
    difficulty: 16384,
    genesis:
"0x6e92f8b23bcd9df34dc813cfaf1d84b71beac80530506b5d63a2df10fe23a660
",
    head:
"0x6e92f8b23bcd9df34dc813cfaf1d84b71beac80530506b5d63a2df10fe23a660
",
    network: 100
  }
},
peers: [],
addPeer: function(),
exportChain: function(),
getContractInfo: function(),
getDatadir: function(callback),
getNodeInfo: function(callback),
getPeers: function(callback),
httpGet: function(),
importChain: function(),
register: function(),
registerUrl: function(),
saveInfo: function(),
setGlobalRegistrar: function(),
setHashReg: function(),
setSolc: function(),
setUrlHint: function(),
sleep: function(),
sleepBlocks: function(),
startNatSpec: function(),
startRPC: function(),
```

```
startWS: function(),
stopNatSpec: function(),
stopRPC: function(),
stopWS: function()
}
```

ขั้นตอนที่ 3 สํารวจบัญชี

```
> eth.accounts
["0x40fa9967135000124efdd62a90ebf08e1170793b"]
> eth.accounts[0]
"0x40fa9967135000124efdd62a90ebf08e1170793b"
> eth.coinbase
"0x40fa9967135000124efdd62a90ebf08e1170793b"
>
```

ขั้นตอนที่ 4 สร้างบัญชีที่ 2 ใน node1

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
I0615 16:00:57.516572 cmd/geth/main.go:286] New wallet appeared:
keystore://D:\BC\node1\keystore\UTC--2017-06-15T09-00-56.748530000Z--
03db47d1881d3f1e4c8f03bb333151560c3c5485, Locked
"0x03db47d1881d3f1e4c8f03bb333151560c3c5485"
> eth.accounts
["0x40fa9967135000124efdd62a90ebf08e1170793b",
"0x03db47d1881d3f1e4c8f03bb333151560c3c5485"]
> eth.accounts[1]
"0x03db47d1881d3f1e4c8f03bb333151560c3c5485"
> eth.coinbase
"0x40fa9967135000124efdd62a90ebf08e1170793b"
>
```

ในตอนนี้ node1 จะมี 2 บัญชี node2 มีเพียง 1 บัญชี เราสามารถตรวจสอบรายละเอียดของโหนดที่ 2 ได้เช่นกันโดยใช้คำสั่งใน lab นี้



## Lab 3: การเชื่อมต่อ Node1 และ Node2

คำอธิบาย เราจะให้โหนดที่ 2 เชื่อมต่อเข้ากับโหนดที่ 1 โดยโหนดที่ 2 ต้องรู้ข้อมูล enode ของโหนดที่ 1 ก่อนจึงจะสามารถเชื่อมต่อได้

ขั้นตอนที่ 1 เลือก enode ของโหนดที่ 1 (เราสามารถดึงข้อมูล enode จากคำสั่ง admin) ออกมาแก้ไขใน editor ตัวใดก็ได้ เช่น notepad เป็นต้น ทั้งนี้ต้องแก้ไข [::] ให้เป็น 127.0.0.1

```
"enode://c999aca89ef7a1af1c972ef7050a8cb846b478e5d8853b4d83c7b58b03d5ff6f49c470549ccc92e0495b9bc77d319401733a36575fbce4ec1de47080256fc130@127.0.0.1:30303?discport=0"
```

ขั้นตอนที่ 2 ใน command prompt ของโหนด 2 ให้ใช้คำสั่ง addPeer() ในการเชื่อมต่อ โดยกำหนดพารามิเตอร์ enode ตามที่ได้แก้ไขในขั้นตอนที่ 1

```
>admin.addPeer("enode://c999aca89ef7a1af1c972ef7050a8cb846b478e5d8853b4d83c7b58b03d5ff6f49c470549ccc92e0495b9bc77d319401733a36575fbce4ec1de47080256fc130@127.0.0.1:30303?discport=0")
true
```

หมายเหตุ ค่า enode ต้องเอาจากเครื่องตนเอง ห้าม copy จากเอกสาร

ในตอนนี้ ถือได้ว่าการเชื่อมต่อ โหนด 2 กับโหนด 1 เสร็จสิ้นแล้ว โดยเราสามารถตรวจสอบผลการเชื่อมต่อได้จากรายละเอียดของ peercount จากคำสั่ง net

```
> net
{
  listening: true,
  peerCount: 1,
  version: "100",
  getListening: function(callback),
  getPeerCount: function(callback),
  getVersion: function(callback)
}
```

## Lab 4: การสร้างและเชื่อมต่อ Node3

คำอธิบาย แล็บนี้เป็นการทบทวนทั้ง 3 แล็บที่ผ่านมา โดยกำหนดให้สร้าง Node3 ขึ้นมาใหม่อีก 1 โหนด จากนั้นจะเชื่อมต่อ Node3 เข้ากับ Node1

ขั้นตอนที่ 1 เปิด command prompt ขึ้นมาอีกหน้าจอหนึ่ง (ควรกำหนดสีพื้นหลังให้แตกต่างกันเพื่อให้ง่ายต่อการแยกแยะ)

ขั้นตอนที่ 2 เปลี่ยนไดเรกทอรีไปยังที่เก็บไฟล์ genesis.json

ขั้นตอนที่ 3 สร้างโหนดและโพลเดอร์เพื่อเก็บข้อมูลของโหนด โดยอาศัยรายละเอียดที่กำหนดใน genesis.json

```
geth --networkid 100 --identity node3 --verbosity 3 --nodiscover --  
nat none --datadir node3 init genesis.json
```

ขั้นตอนที่ 4 สร้างบัญชี

4.1 เข้าสู่ geth prompt

```
geth --networkid 100 --identity node3 --verbosity 3 --nodiscover --  
nat none --datadir node3 --rpc --rpcapi "web3, eth, personal" --  
rpccorsdomain "*" --rpcport 3333 --port 3 --ipcpath node3/geth.ipc  
console
```

4.2 สร้างบัญชีใหม่ พร้อมกำหนดรหัส (หมายเหตุ ต้องจำรหัสให้ดี เพื่อใช้ในขั้นตอนต่อไป)

```
>personal.newAccount("password")  
"0x304eff60393b8577baf7ca6621143654ab2a4cde"
```

ขั้นตอนที่ 5 เชื่อมต่อโหนด3 เข้ากับโหนด1 โดยใช้ข้อมูล enode ของ Node1 เดิมที่ทำไว้แล้วในแล็บที่ผ่านมา

```
admin.addPeer("enode://c999aca89ef7a1af1c972ef7050a8cb846b47  
8e5d8853b4d83c7b58b03d5ff6f49c470549ccc92e0495b9bc77d319  
401733a36575fbce4ec1de47080256fc130@127.0.0.1:30303?discport=0  
")
```

หมายเหตุ ใช้ enode ของเครื่องตนเอง

ขั้นตอนที่ 6 ตรวจสอบผลการเชื่อมต่อที่โหนด1 โดยให้กลับไปที่ command prompt ของ Node1 และใช้คำสั่ง net

```
> net  
{  
  listening: true,
```

```
peerCount: 2,
```

```
version: "100",
```

```
getListening: function(callback),
```

```
getPeerCount: function(callback),
```

```
getVersion: function(callback)
```

```
}
```

## **Lab 5: การสร้าง Miner**

คำอธิบาย จากทั้ง 4 แล็บที่ผ่านมาตอนนี้เรามี โหนดทั้งสิ้น 3 โหนด ในแล็บนี้เราจะทดลองการสร้าง miner โดยกำหนดให้โหนด Node2 เป็น miner ด้วยคำสั่ง miner.start()

ขั้นตอนที่ 1 เปิด command prompt ของ Node2

ขั้นตอนที่ 2 สั่งให้ทำงานเป็น miner ในขั้นตอนนี้จะมีการ generate DAG ซึ่งจะใช้เวลาสักหน่อย

```
> miner.start(2)
```

(2) คือ จำนวนคอร์ของ CPU ที่ใช้ในการประมวลผล ถ้าไม่ระบุจำนวน จะใช้ทั้งหมดที่มี

หมายเหตุ คำสั่งนี้จะมีผลให้เกิดการสร้าง DAG ใน ethash dir เช่น

C:\Users\CIS2016\AppData\Ethash ซึ่งถ้าเคยมีการสร้างไว้ก่อนหน้านี้แล้ว จะต้องไปลบของเดิมทิ้งก่อน

## **Lab 6: การตรวจสอบเงินในบัญชี**

### **บัญชีของ Node1**

ขั้นตอนที่ 1 เปิด command prompt ของ Node1

ขั้นตอนที่ 2 ตรวจสอบเงินในบัญชี

```
> eth.getBalance(eth.accounts[0])
0
> web3.fromWei(eth.getBalance(eth.accounts[0]), "ether")
0
```

แสดงว่าตอนนี้ node1 ยังไม่มีเงินเลย

หมายเหตุ 1 ether = 1.e+18 wei หน่วย wei เป็นหน่วยที่เล็กที่สุด

### **บัญชีของ Node2**

ขั้นตอนที่ 1 เปิด command prompt ของ Node2

ขั้นตอนที่ 2 ตรวจสอบเงินในบัญชี

```
> eth.getBalance(eth.accounts[0])
2150000000000000000000
> web3.fromWei(eth.getBalance(eth.accounts[0]), "ether")
215
```

## Lab 7: Transaction การโอนเงิน

คำอธิบาย จาก Lab 5 เราได้ทดลองให้ Node2 เป็น miner จึงมีเงินอยู่จำนวนหนึ่ง ในแล็บนี้เราจะทดลองสร้าง transaction การโอนเงินจาก Node2 ไปยัง Node1 โดยให้ Node3 เป็น miner ทั้งนี้ทุกครั้งก่อนการโอนเงิน เราจะต้อง unlock บัญชีที่ต้องการโอนเงินเสียก่อน (ในที่นี้คือบัญชีของ Node2)

ขั้นตอนที่ 1 หยุด miner เพื่อต้องการแสดง pending transaction ด้วยคำสั่ง **miner.stop()**

ขั้นตอนที่ 2 unlock บัญชีของ Node2 ด้วยคำสั่ง personal.unlockAccount() โดยต้องการกรอกรหัสผ่านด้วยจึงจะสามารถ unlock ได้สำเร็จ

```
> personal.unlockAccount(eth.accounts[0])
Unlock account 0xaa97b233dc3b903c064219a22ab1b762316b90f5
Passphrase: password2
True
```

ขั้นตอนที่ 3 โอนเงินด้วยคำสั่ง eth.sendTransaction() โดยเราใช้ address ของบัญชีผู้รับโอน (Node1) ด้วย ทั้งนี้เราสามารถทราบ address ของบัญชีได้จากตอนที่เราสร้างบัญชีด้วยคำสั่ง newAccount (ใน Lab1) และยังสามารถเรียกดูในภายหลังได้จากคำสั่ง eth.accounts[0]

3.1 เปิด command prompt ของ Node1 เรียกดู address ด้วยคำสั่ง  
eth.accounts[0]

3.2 เปิด command prompt ของ Node2 เพื่อโอนเงิน ด้วยคำสั่ง  
eth.sendTransaction()

```
>eth.sendTransaction({from:eth.accounts[0],
to:"0xd01cb382caf5eb782644b5e63ad3cdb100e313bf",
value:web3.toWei(10,"ether")})

"0x26b4c086d178023db79eca266ffbd8ddcaa925e7336cdd3caa89a59c0d8b5fad"
```

หมายเหตุ 1 ทุกครั้งที่ send transaction ต้อง unlock เสมอ เพราะทุกครั้งที่เราส่งเสร็จจะ lock โดยอัตโนมัติ

2. จะได้รับ Transaction hash เป็นคำตอบ ให้เก็บไว้เพื่อตรวจสอบต่อไป

ขั้นตอนที่ 4 การสำรวจ transaction

4.1 eth.getTransaction("Txhash")

```
>eth.getTransaction("0xeb795e43455dded5f2348a3bd222d6b6308050abfc42653bbffccb4f9301eb")
{
```

```

    blockHash:
    "0x0000000000000000000000000000000000000000000000000000000000000000
    0000",
    blockNumber: null,
    from: "0x0fc48c008e6743586eadd8e4bc821cea108c2121",
    gas: 90000,
    gasPrice: 20000000000,
    hash:
    "0xebed795e43455dded5f2348a3bd222d6b6308050abfc42653bbffccb4f9301e
    b",
    input: "0x",
    nonce: 1,
    r:
    "0x30ee929527ea09bbbba2998ea9586c578aabbbe61b566112e93824c314ae0
    c45",
    s:
    "0x6b8eeb0e362430eba841e9cceb8ba6fdc690e49cd3dc9862ce28c36430a4cdf
    3",
    to: "0x40fa9967135000124efdd62a90ebf08e1170793b",
    transactionIndex: 0,
    v: "0x1b",
    value: 15000000000000000000
}

```

## 4.2 eth.pendingTransaction

```

>eth.pendingTransactions
[
  {
    blockHash: null,
    blockNumber: null,
    from: "0x0fc48c008e6743586eadd8e4bc821cea108c2121",
    gas: 90000,
    gasPrice: 20000000000,
    hash:
    "0xebed795e43455dded5f2348a3bd222d6b6308050abfc42653bbffccb4f9301e
    b",

```

```

input: "0x",
nonce: 1,

r:
"0x30ee929527ea09bbba2998ea9586c578aabbbe61b566112e93824c314ae0
c45",

s:
"0x6b8eeb0e362430eba841e9cceb8ba6fdc690e49cd3dc9862ce28c36430a4cdf
3",

to: "0x40fa9967135000124efdd62a90ebf08e1170793b",
transactionIndex: 0,
v: "0x1b",
value: 15000000000000000000
}]

```

ขั้นตอนที่ 5 ให้ Node 3 รันเป็น miner

5.1 เปิด command prompt ของ Node3

5.2 พิมพ์คำสั่ง miner.start(2) รอสักครู่หนึ่งและปิด miner ด้วยคำสั่ง miner.stop()

ขั้นตอนที่ 6 การตรวจสอบเงินในบัญชีของ Node1 และ Node2

6.1 เปิด command prompt ของ Node1 และพิมพ์คำสั่ง  
eth.getBalance(eth.accounts[0])

6.2 เปิด command prompt ของ Node2 และพิมพ์คำสั่ง  
eth.getBalance(eth.accounts[0])

6.3 ตรวจสอบ transaction ใน node2

**>eth.getTransaction("0xebed795e43455dded5f2348a3bd222d6b630  
8050abfc42653bbffccb4f9301eb")**

หมายเหตุ ต้องใช้ transaction hash ของ node2 ของตนเอง

**>eth.pendingTransactions**

หมายเหตุ ในแล็บนี้หากพบปัญหาเรื่องไม่ประมวลผล transaction ให้ลองตรวจสอบก่อนว่าทุกโหนดมี block number เดียวกัน (แสดงว่า sync กัน) โดยใช้คำสั่ง **eth.blockNumber** ในการตรวจสอบ

## **LAB 8: การเชื่อมต่อกับเครื่องอื่นผ่านระบบเครือข่าย**

ขั้นตอนที่ 0. การเตรียมการ

ให้ลบข้อมูล ethash เดิมทิ้ง (อยู่ใน ethash dir เช่น  
C:\Users\CIS2016\AppData\Ethash)

ขั้นตอนที่ 1. สร้าง folder ใหม่ d:\bc2 ที่มีเพียงแค่ genesis.json และ geth.exe

ขั้นตอนที่ 2. เปิด command prompt ขึ้นมาใหม่ และเข้าไปยัง folder ใหม่

ขั้นตอนที่ 3. สร้างโหนด

```
geth --networkid 100 --identity node1 --verbosity 3 --nodiscover --  
nat none --datadir node1 init genesis.json
```

ขั้นตอนที่ 4. เข้าสู่ geth prompt

```
geth --networkid 100 --identity node1 --verbosity 3 --nat none --  
datadir node1 --rpc --rpcapi "web3, eth, personal" --rpccorsdomain "*" --  
ipcpath node1/geth.ipc console
```

ขั้นตอนที่ 5. สร้างบัญชีใหม่

```
> personal.newAccount("password")  
"0x30af1b7513494af2fcea122fe4044cc515a5638"
```

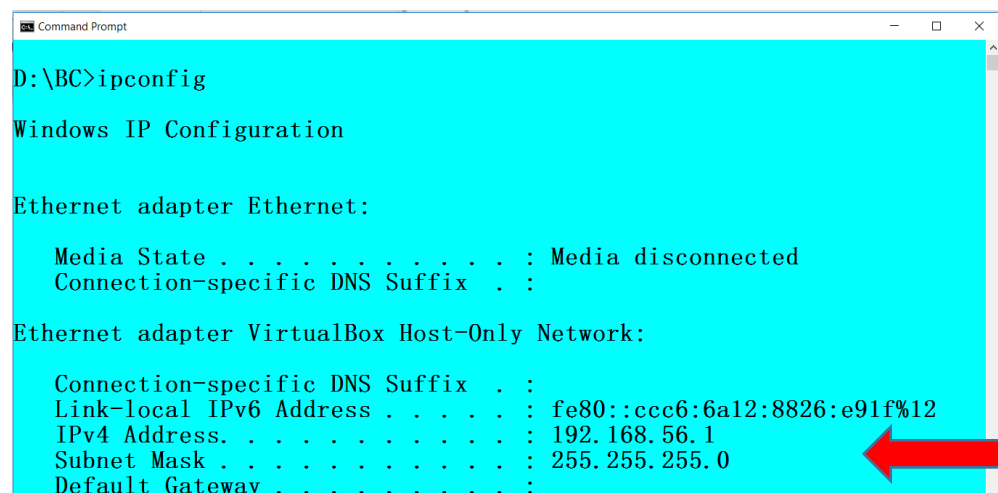
คือ address ของบัญชีใหม่นี้

ขั้นตอนที่ 6. เชื่อมต่อไปยังโหนดอื่น โดยเราต้องรู้ IP address ของเครื่องที่เราต้องการ  
เชื่อมต่อด้วยก่อน เช่น 192.168.1.207 เป็นต้น

6.1 สํารวจ ip address ของเครื่องตนเอง

6.1.1 เปิด command prompt ใหม่

6.1.2 พิมพ์คำสั่ง ipconfig



```
Command Prompt  
D:\BC>ipconfig  
  
Windows IP Configuration  
  
Ethernet adapter Ethernet:  
  
Media State . . . . . : Media disconnected  
Connection-specific DNS Suffix . :  
  
Ethernet adapter VirtualBox Host-Only Network:  
  
Connection-specific DNS Suffix . :  
Link-local IPv6 Address . . . . . : fe80::ccc6:6a12:8826:e91f%12  
IPv4 Address. . . . . : 192.168.56.1  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . :
```

6.2 ให้แต่ละคน addPeer ของคนด้านซ้ายของตนเอง (ต้องรู้ enode และ ip address  
ของคนทางซ้าย)



```
admin.addPeer("enode://0a4d41575465371c6af86c706f554cb3366feb0c7d629cf6ce65e3fb959d8f23c26523bcf6bf473a13e7d1884c3b938ef5404d3ea68b6a41f1de656e957056f8@192.168.1.101:30303?discport=0")
```

6.3 ให้แต่ละคน addPeer ของคนด้านขวาของตนเอง (ต้องรู้ enode และ ip address ของคนทางขวา)

```
admin.addPeer("enode://0a4d41575465371c6af86c706f554cb3366feb0c7d629cf6ce65e3fb959d8f23c26523bcf6bf473a13e7d1884c3b938ef5404d3ea68b6a41f1de656e957056f8@192.168.1.102:30303?discport=0")
```

ขั้นตอนที่ 7. ทุกคนเปิด miner

```
>miner.start(2)
```

ขั้นตอนที่ 8. ตรวจสอบเงินในบัญชี

```
>eth.getBalance(eth.accounts[0])
```

ขั้นตอนที่ 9. โอนเงิน

9.1 unlock บัญชี

```
>personal.unlockAccount(eth.accounts[0])
```

9.2. โอนเงิน

```
>eth.sendTransaction({from:eth.accounts[0],  
to:"0x347f55a82f2b9115d7924f43a36bea7d1106daf6",value:web3.toWei(2,"ether")})  
  
I0206 14:38:53.275720 eth/api.go:1193]  
Tx(0x0f2b651d02c582ee4433727873e92b0a082a5bb805652d56ddcb1b5aa6717301) to: 0x347f55a82f2b9115d7924f43a36bea7d1106daf6  
"0x0f2b651d02c582ee4433727873e92b0a082a5bb805652d56ddcb1b5aa6717301"
```

## Lab 9: การติดตั้งโหนดและใช้งานบล็อกเชนบนคลาวด์

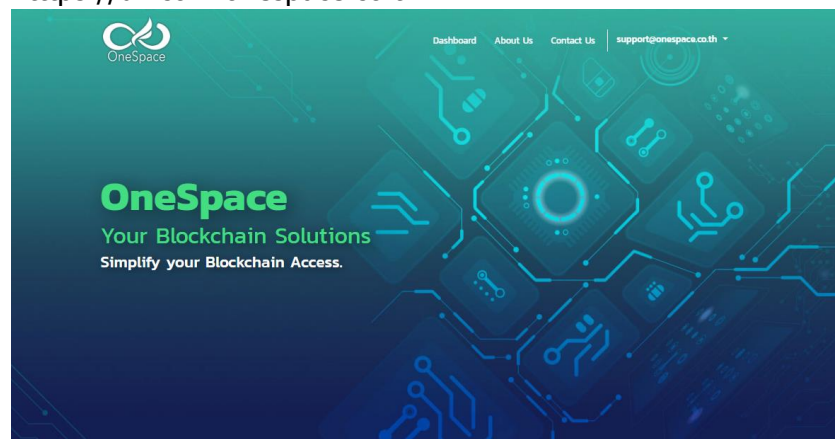
คำอธิบาย ในแล็บนี้ เราจะแนะนำให้ทุกท่านได้รู้จัก OneSpace ระบบบล็อกเชนบนคลาวด์ ที่ถูกพัฒนาขึ้นมาเพื่อให้ผู้ใช้สามารถใช้บล็อกเชนได้สะดวกยิ่งขึ้น โดยเราจะทดลองใช้งาน OneSpace แทนที่ใช้ geth ตามแล็บ 1-8

หมายเหตุ มีเอกสารประกอบเป็น powerpoint

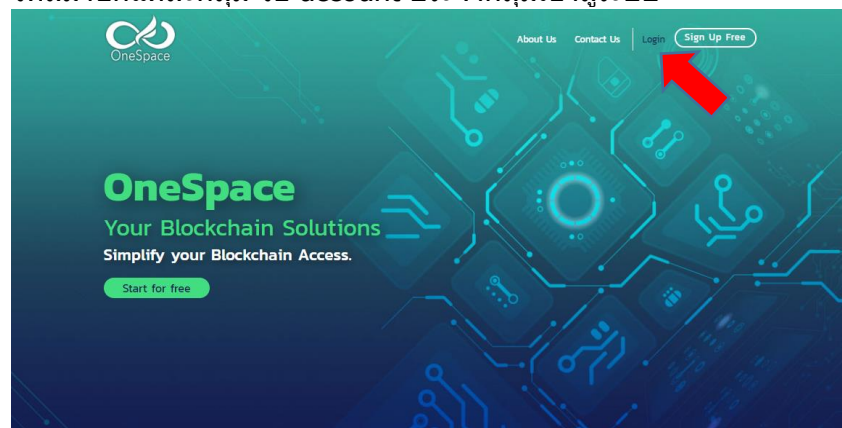
### Lab 9.1 การเข้าใช้งาน

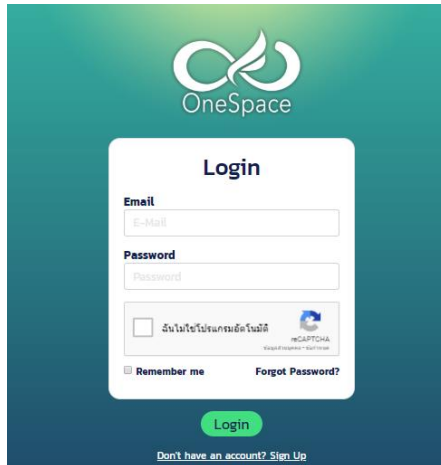
เพื่อให้สะดวกกับการทดลองใช้งานในวันนี้ ทีมผู้จัดได้จำลองระบบ OneSpace และ User account ให้แต่ละกลุ่ม โดยเริ่ม Workshop ดังนี้

1. ให้ทุกคนเปิด browser Chrome และเปิด URL:  
<https://unicorn.onespace.co.th>



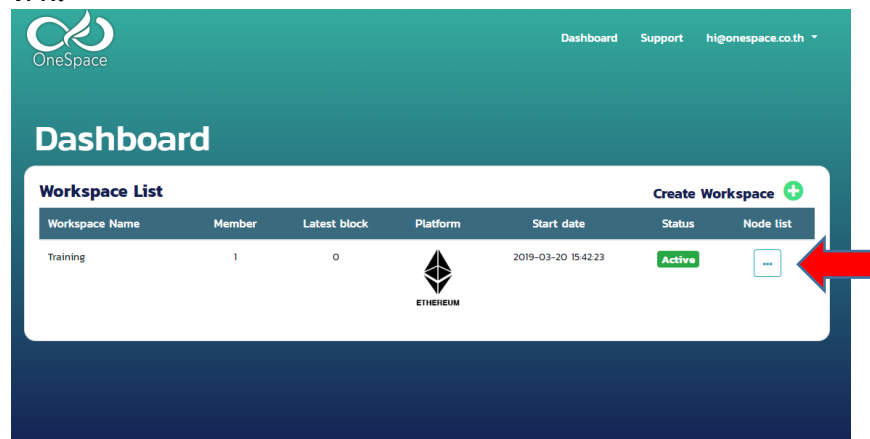
2. ให้สมาชิกแต่ละกลุ่ม ใช้ account ประจำกลุ่มเข้าสู่ระบบ



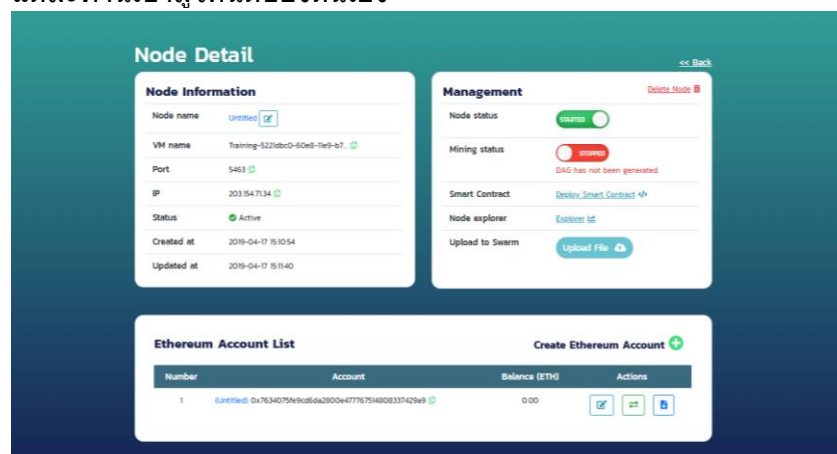


กลุ่มที่	Account	Password
1	group1@onespace.co.th	1q2w3e\$R
2	group2@onespace.co.th	1q2w3e\$R
3	group3@onespace.co.th	1q2w3e\$R
4	group4@onespace.co.th	1q2w3e\$R

3. เมื่อเข้าสู่ระบบแบบฟรี จะพบโหนดที่ถูกสร้างเตรียมให้ตามชื่อผู้เข้าอบรมแต่ละท่าน



4. แต่ละท่านเข้าสู่โหนดของตนเอง

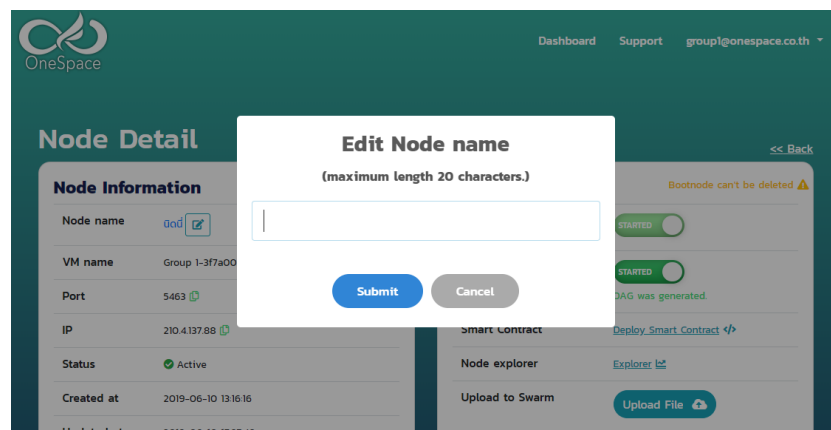
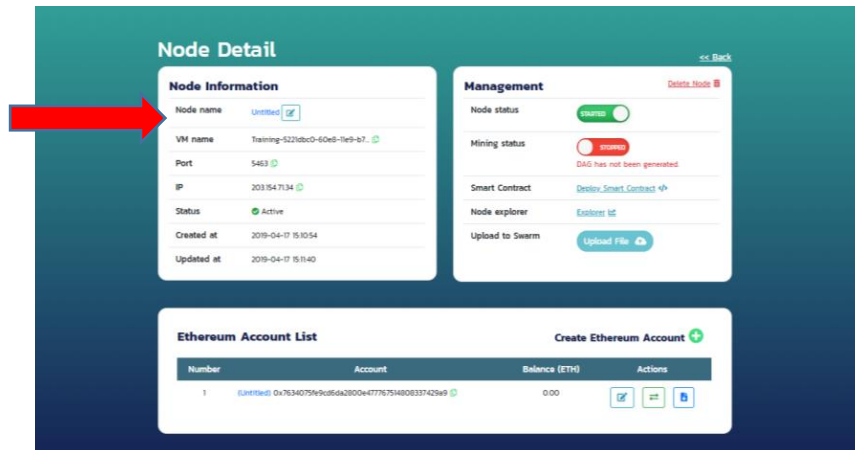


## Lab 9.2 การสำรวจโหนดในหน้า Node Detail

ในหน้า Node Detail ผู้ใช้สามารถสำรวจและปรับแต่งข้อมูลพื้นฐานได้ เช่น การเปิด-ปิด โหนด การเปิด-ปิด mining เป็นต้น

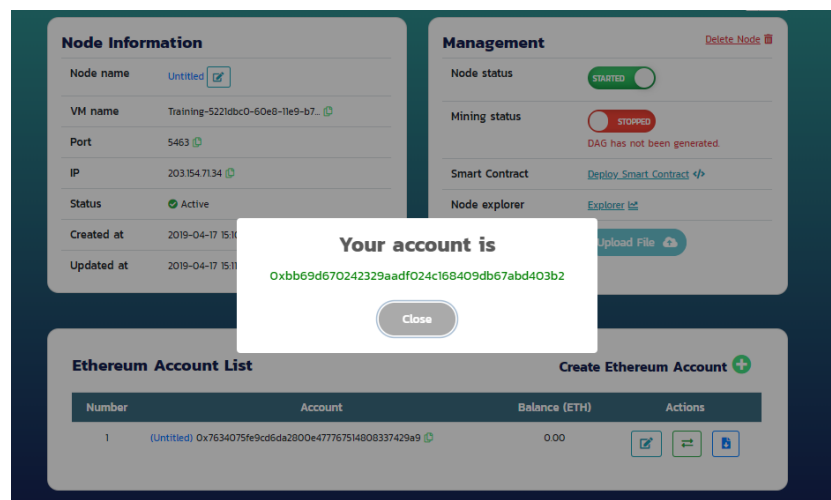
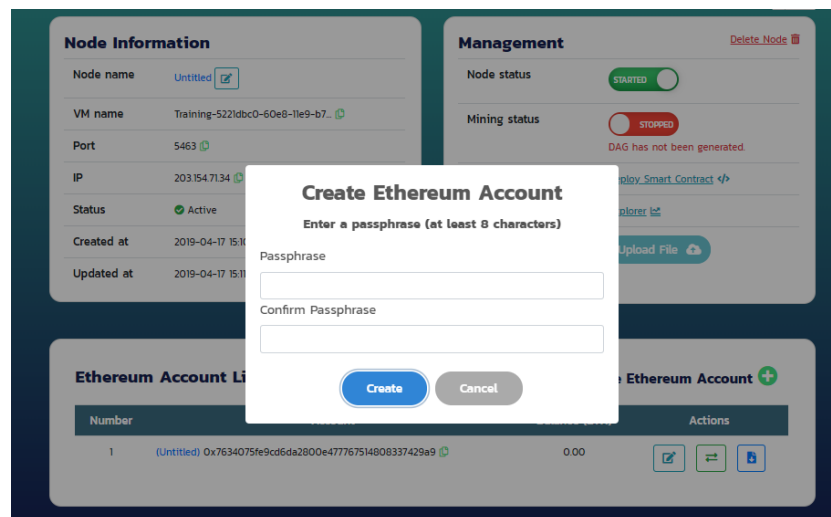
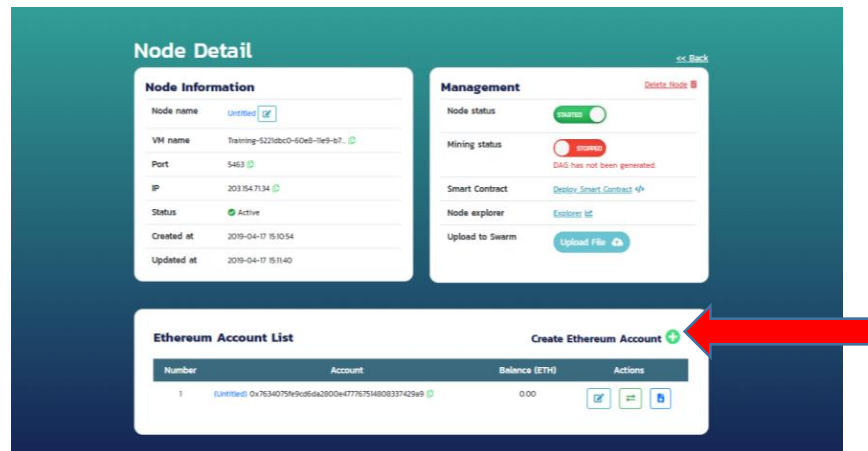
## Lab 9.3 การแก้ไขชื่อโหนด

ภายใน Node Explorer เราสามารถแก้ไขชื่อโหนดได้อย่างง่ายๆ ตามใจเราเอง  
ให้ทุกคนแก้ไขชื่อโหนดเป็นชื่อเล่นของตนเอง (ภาษาไทยก็ได้)

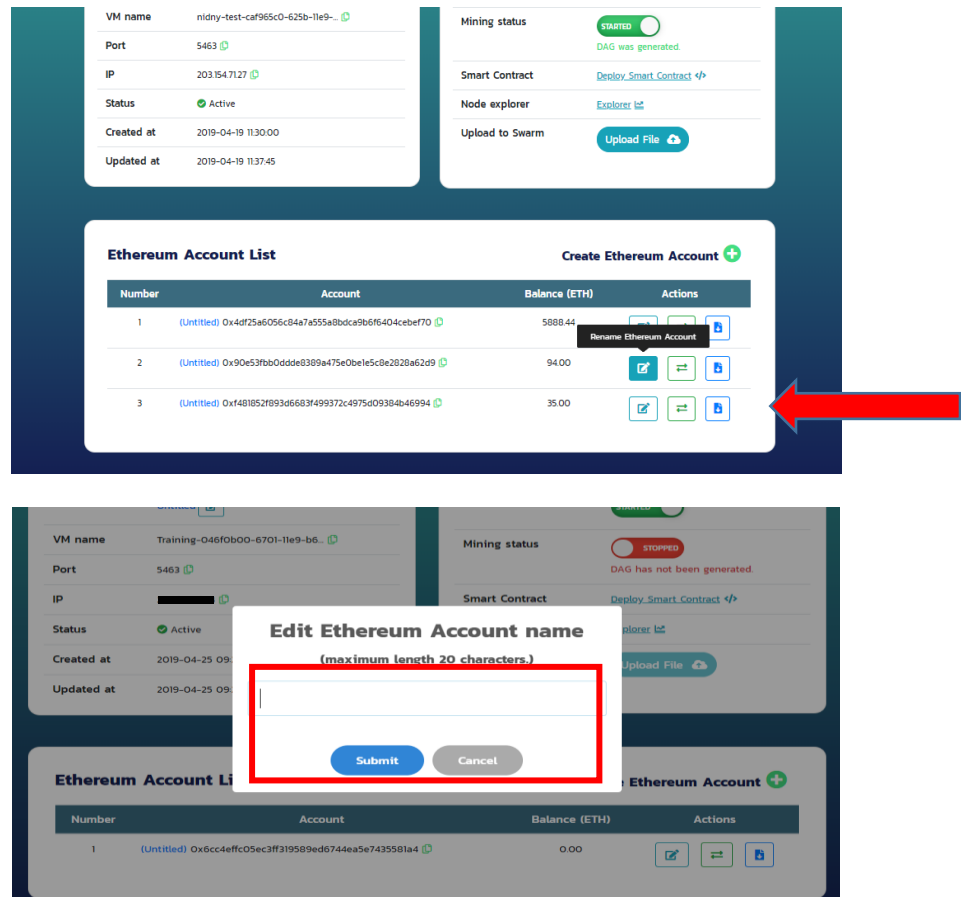


## Lab 9.4 การเพิ่มบัญชีภายในโหนด

ให้ทุกคนสร้างบัญชีเพิ่มอีกคนละ 1 บัญชี



## Lab 9.5 การแก้ไขข้อบกพร่อง



The screenshot shows the Ethereum interface with a VM configuration panel on the left and a mining status panel on the right. The main area displays the 'Ethereum Account List' table. A red arrow points to the 'Rename Ethereum Account' button in the 'Actions' column of the first row.

Number	Account	Balance (ETH)	Actions
1	(Untitled) 0x4df25a6056c847a555a8bdca9b6f6404cebf70	5888.44	[Rename Ethereum Account] [Copy] [Share]
2	(Untitled) 0x90e53fbb0ddde8389e475e0be1e5c8e282a62d9	94.00	[Copy] [Share] [Share]
3	(Untitled) 0xf481852f893d6683f499372c4975d09384b46994	35.00	[Copy] [Share] [Share]

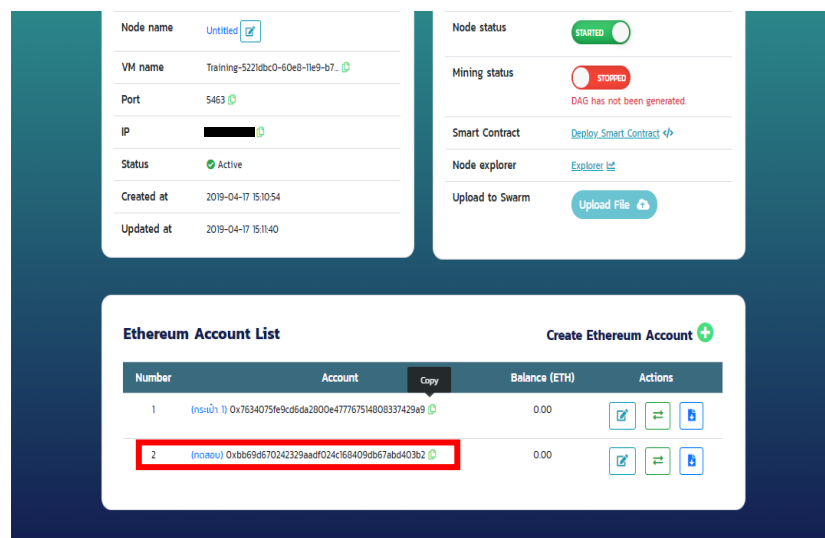
**Edit Ethereum Account name**  
(maximum length 20 characters)

Submit Cancel

## Lab 9.6 การโอนเงินข้ามบัญชีภายในโหนดเดียวกัน

เราสามารถดำเนินการได้อย่างง่ายดาย โดยอาศัยการทำงาน 3 ขั้นตอนคือ

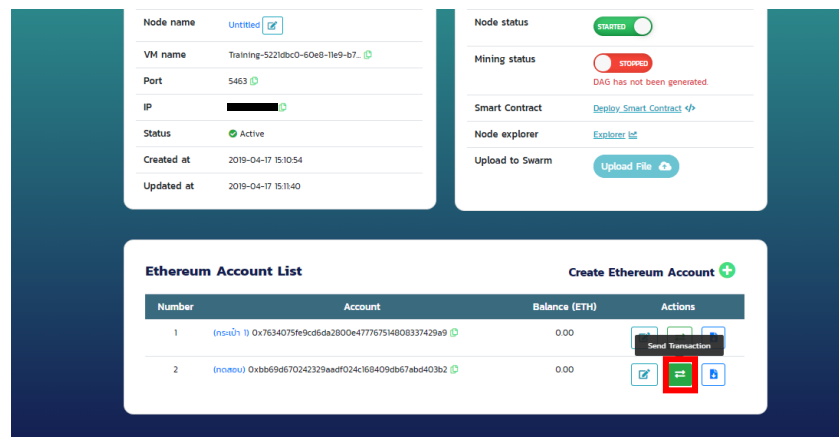
1. กดปุ่ม copy เลขบัญชีปลายทาง



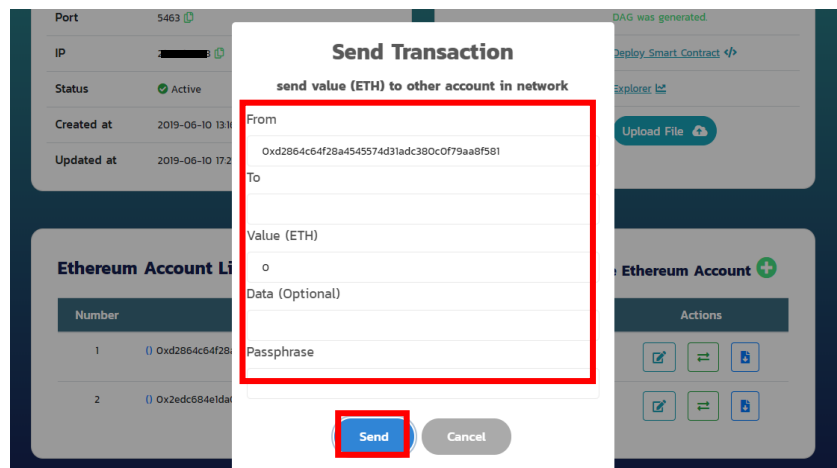
The screenshot shows the Ethereum interface with a VM configuration panel on the left and a mining status panel on the right. The main area displays the 'Ethereum Account List' table. A red box highlights the 'Copy' button in the 'Actions' column of the second row.

Number	Account	Balance (ETH)	Actions
1	(no name) 0x7634075fe9cd5d2800e47767514808337429a9	0.00	[Copy] [Share] [Share]
2	(no name) 0xb668d670242329aadf024c168409db67abd403b2	0.00	[Copy] [Share] [Share]

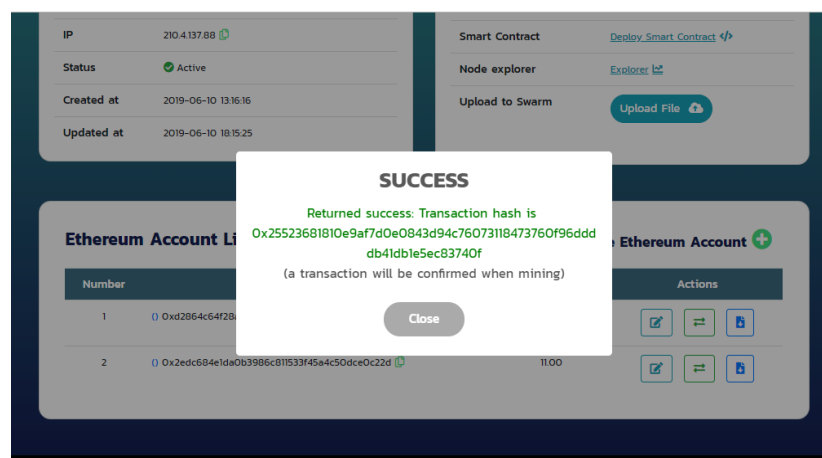
## 2. กดปุ่มโอนเงินจากบัญชีต้นทาง



## 3. กรอกข้อมูลเงินที่ต้องการโอน (และข้อมูลอื่นๆ ที่ต้องการ) พร้อมกดโอน



เมื่อเรียบร้อยแล้ว เราสามารถบันทึกเลขที่ Transaction ไว้เพื่อค้นหาได้ในภายหลัง โดยเราจะสำรวจ Transaction ได้ด้วย Node Explorer



หมายเหตุ ในแล็บนี้ transaction จะยังไม่ถูกประมวลผล (ในหน้า transaction information –ข้อมูลของ Block Hash: **Transaction has not been confirmed**) เนื่องจาก Miner ยังไม่ได้ทำงาน

## Lab 9.7 ฟังก์ชัน Node Explorer

### Node Detail

#### Node Information

Node name: [Untitled](#)

VM name: [Training-527dbcd0-60ed-11e9-b7...](#)

Port: [5463](#)

IP: [203.154.71.34](#)

Status: Active

Created at: 2019-04-17 15:10:54

Updated at: 2019-04-17 15:11:40

#### Management

Node status: started

Mining status: stopped  
DAG has not been generated

Smart Contract: [Deploy Smart Contract](#)

Node explorer: [Explorer](#)

Upload to Swarm: [Upload File](#)

#### Ethereum Account List

Create Ethereum Account

Number	Account	Balance (ETH)	Actions
1	<a href="#">Untitled</a> <a href="#">0x76340759a9c6da2800e477767514808337425a9</a>	0.00	<a href="#">Edit</a> <a href="#">Reset</a> <a href="#">Delete</a>

### Ethereum Blockchain Explorer

Search by TransactionHash / BlockHash / BlockNumber

Search

IP: **203.154.71.36**

MINING STATUS: Start

LATEST BLOCK: **478**

TOTAL DIFFICULTY: **7848414**

#### Latest Blocks

Block#	Block Hash	Miner	Size	# of Tx
<a href="#">478</a>	<a href="#">0xb4d361ad149a...</a>	<a href="#">0x3410031df6f5...</a>	538	0
<a href="#">477</a>	<a href="#">0x48828ed108aa...</a>	<a href="#">0x3410031df6f5...</a>	538	0
<a href="#">476</a>	<a href="#">0x9291bccb6984...</a>	<a href="#">0x3410031df6f5...</a>	538	0
<a href="#">475</a>	<a href="#">0xe6516da429fa...</a>	<a href="#">0x3410031df6f5...</a>	538	0
<a href="#">474</a>	<a href="#">0x064127b3b062...</a>	<a href="#">0x3410031df6f5...</a>	647	1

#### Transactions

Tx Hash	Block#	From	To
<a href="#">0x58ee358b4ee6...</a>	<a href="#">474</a>	<a href="#">0x3410031df6f5...</a>	<a href="#">0x295A182653d5...</a>

### Transaction Detail

#### Transaction Information

IP: [210.4137174](#)

Transaction Hash: [0x7e9bec1c0b4a0127f7f96aeb3ceea8aadb0b8dc0f54a8fb432a162ee071](#)

Block Hash: Transaction has not been confirmed

Block: Transaction has not been confirmed

From: [0xfef0340f6530ff2bca66807cc814626f2521822](#)

To: [0xfef0340f6530ff2bca66807cc814626f2521822](#)

Value: 0 ETH (0 Wei)

Gas: 50000

Gas Price: 1000000000

Gas Used By Transaction: Transaction has not been confirmed

Contract Address: Transaction has not been confirmed

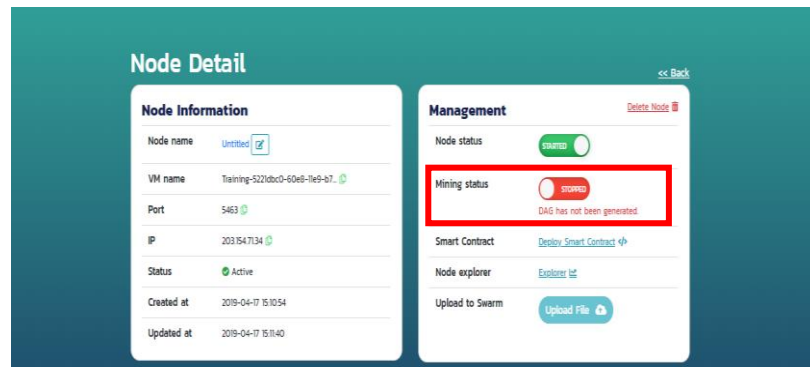
Nonce: 6

Input Data: [0x](#)



## Lab 9.8 การเปิด-ปิด Miner

ในหน้า Node Detail -> ในกล่อง Management เราสามารถเปิด-ปิด Miner ได้ที่ Mining status



เมื่อเราสั่งให้ Miner ทำงาน transaction ด้านบน ก็จะถูกประมวลผล (ในหน้า transaction information-ข้อมูลของ Block Hash: 0x25a4ada5862adadfadad1258526ad8526) และเราสามารถดูผลลัพธ์ได้ที่เงินในบัญชี

## Lab 9.9 การโอนเงินข้ามโหนด

ให้ทุกคนโอนเงินจากบัญชีตนเอง ไปยังบัญชีในโหนดอื่น