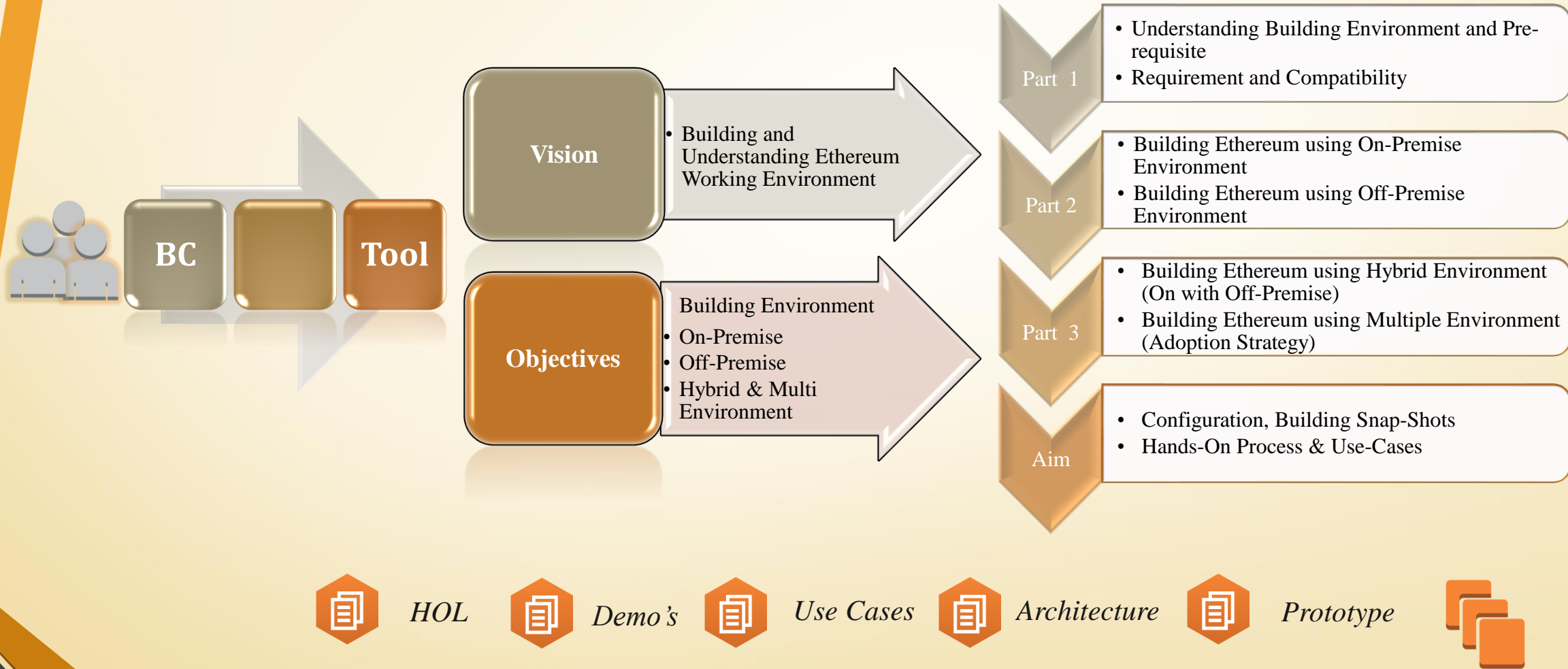


CONTENT DEVELOPMENT  
ON  
**BLOCKCHAIN**  
CONFIGURATION, BUILDING, & USE-CASE

By  
**Dr YOGESH G (PhD)**

# Blockchain : TOC

## Content Development



# On-Premise: Setting up the Development Environment

Solidity is the most popular smart contract language for Ethereum. So now we have chosen our language for smart contract development, let us see what is required to get started.

In order to set up the environment for developing a Smart Contract, two choices are available.

1. Remix Online IDE
2. Local Set Up

# On-Premise: Setting up the Development Environment (Cont.)

## **Remix IDE**

The first choice is to use online remix IDE (integrated development environment) to build and test the smart contract. This is a quick, easy and recommended way for beginners.

### **Pros:**

No installation required, completely online

Get started without any challenges, quick for prototyping and validating Smart Contract

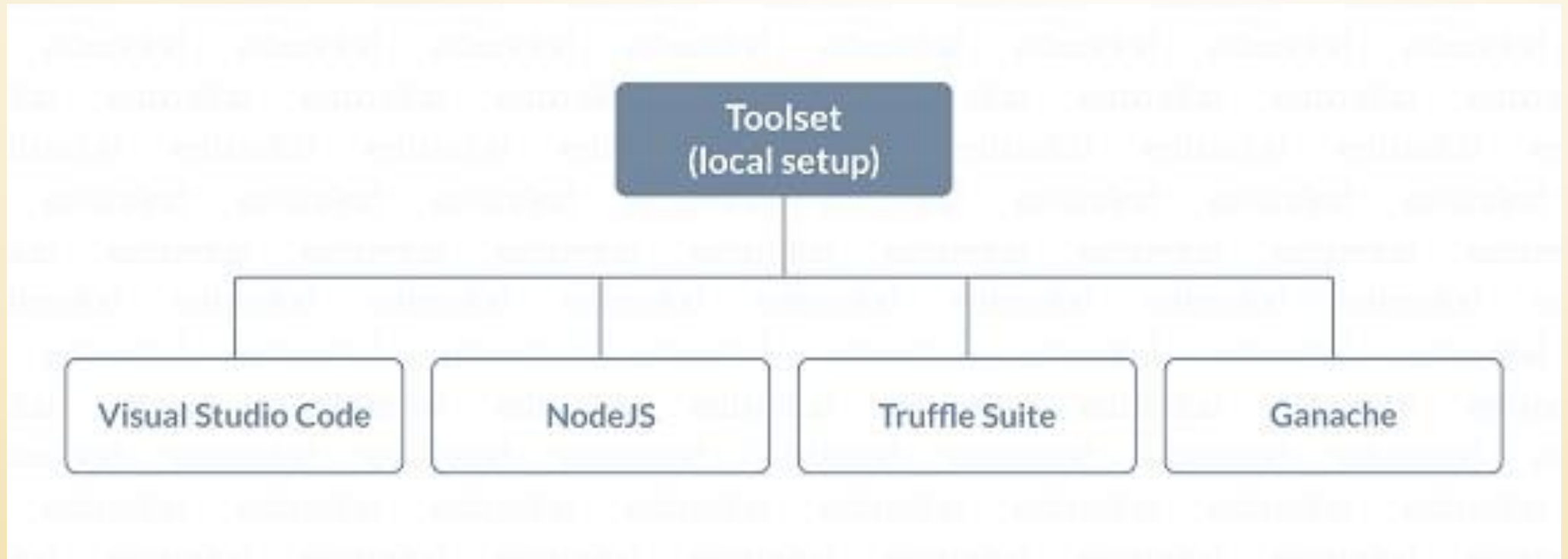
Provides a local Ethereum virtual network

### **Cons:**

As the progression goes from building Smart Contract, to consuming the smart contract in DApp (Distributed Application) Remix IDE has serious limitations.

# On-Premise: Setting up the Development Environment (Cont.)

## Local Setup



# On-Premise: Setting up the Development Environment (Cont.)

Toolset required for smart contract development is as follows:

## **Node**

The latest version of the node can be downloaded [here](#).

Using NVM (Node Version Manager)

## **Windows Installation**

Download installer from [here](#)

Download and run the installer, this should install the nvm

Open up a windows terminal and run “nvm –version”

nvm --version

This should return the current version on nvm.

## **Mac Installation**

Run the following script, this should automatically install nvm on the machine.

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.0/install.sh | bash
```

In order to validate the installation run the following command “nvm --version”

# On-Premise: Setting up the Development Environment (Cont.)

## Post-Installation

1. Once the nvm is installed run the following command.
2. Run the following command "nvm list". This will list down the node version installed locally-> **nvm list**
3. To list down the node version available for download please run the following command-> **nvm ls-remote**
4. In order to install a specific version of node run the following command "nvm install <node-version" e.g. "nvm install 10.15.0" this will install node version 10.15.0-> **nvm install 10.15.0**
5. Check the current version of the node being used by following command  
node --version
1. For further details on nvm or node, please look at details [here](#).

# On-Premise: Setting up the Development Environment (Cont.)

## Visual Studio Code

Visual Studio Code has been one of the best IDE provided by Microsoft for solidity smart contract development. Visual Studio code can be downloaded from [here](#).

## Truffle Suite

Truffle suite provides an excellent framework for Smart Contract. Installation of truffle is done via npm (node package manager), which is installed with node. In order to install truffle, open up the terminal window or command prompt window. In the terminal window run the following command,

*npm install -g truffle*

(g flag in the command stands for global)

Post installation of truffle is completed, run the following command to verify the version installed

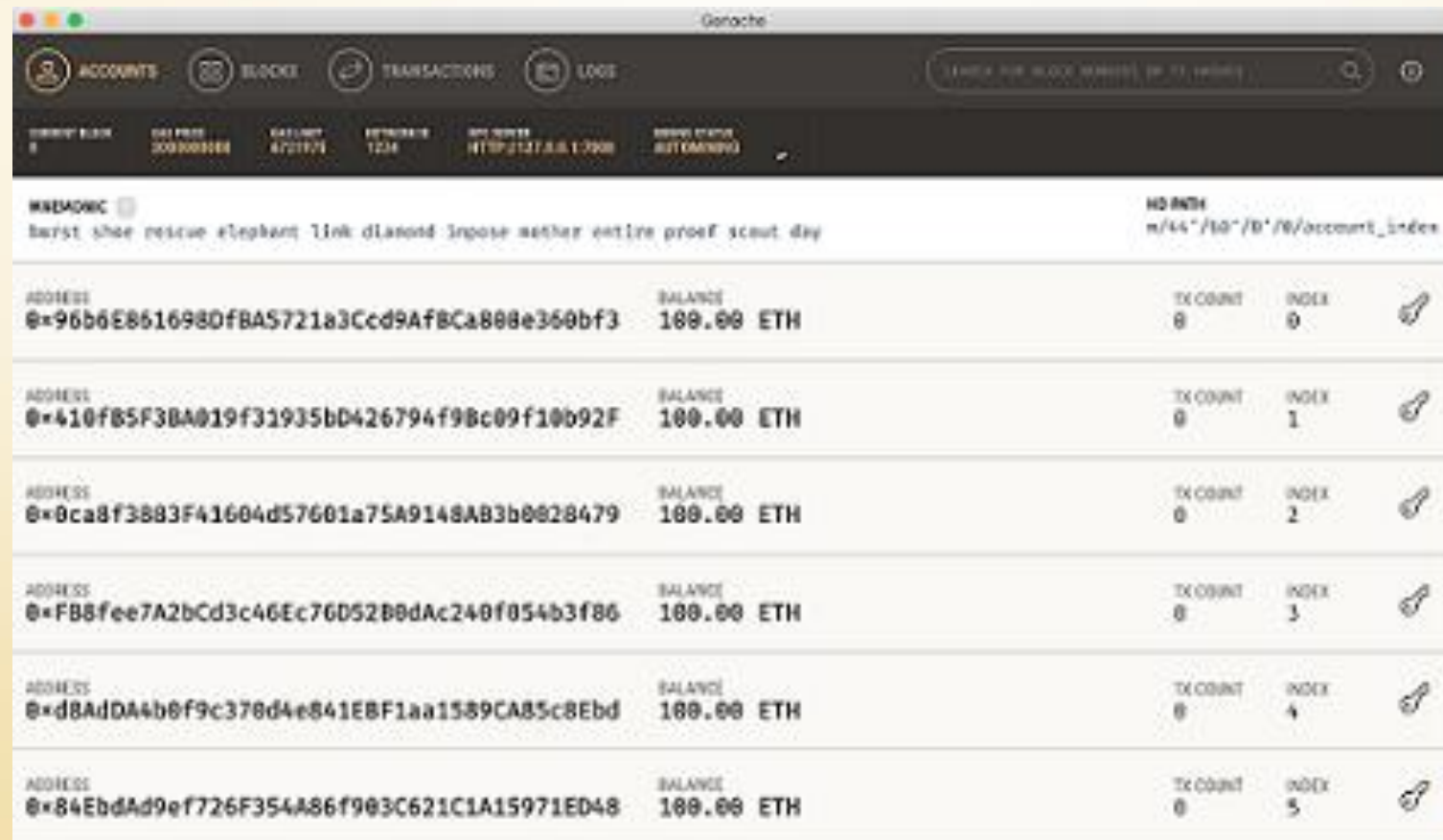
*truffle --version*



# On-Premise: Setting up the Development Environment (Cont.)

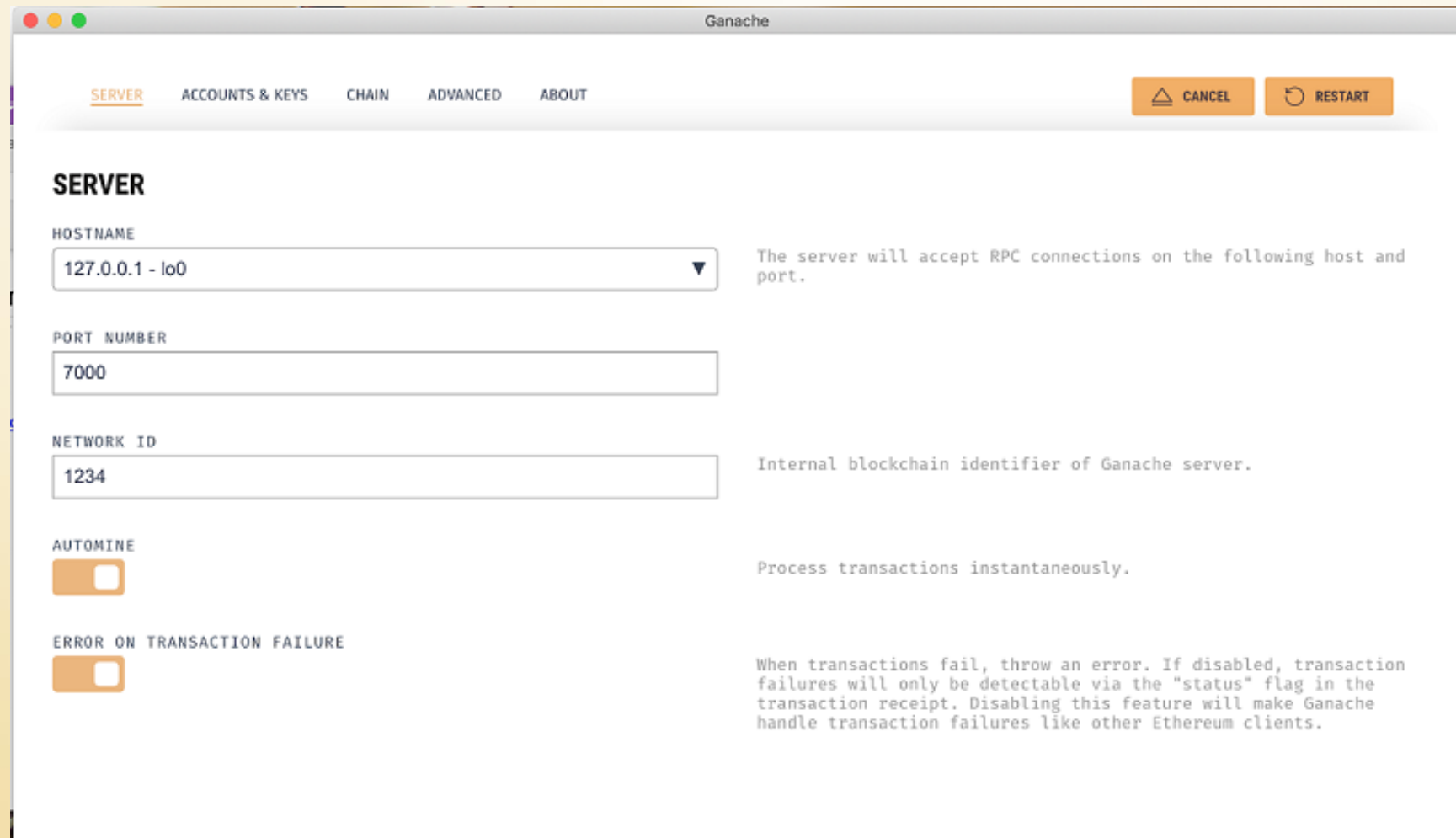
## Ganache (Single node local Ethereum network)

1. The setup file for ganache can be downloaded [here](#)
2. Select the installation package based on the operating system.
3. Run the installer, once the installation is completed following screen should appear



# On-Premise: Setting up the Development Environment (Cont.)

In order to customize the ganache test Ethereum node running, please click on the gear icon on the right corner of the ganache tool, the following screen should be visible.



The screenshot shows the 'Ganache' application window with the 'SERVER' tab selected. The window has a title bar with standard macOS window controls (red, yellow, green buttons). The 'SERVER' tab is highlighted in orange, with other tabs 'ACCOUNTS & KEYS', 'CHAIN', 'ADVANCED', and 'ABOUT' visible. In the top right corner, there are two orange buttons: 'CANCEL' with a triangle icon and 'RESTART' with a circular arrow icon. The main content area is titled 'SERVER' and contains several configuration options:

- HOSTNAME:** A dropdown menu showing '127.0.0.1 - lo0'. To its right is a text description: 'The server will accept RPC connections on the following host and port.'
- PORT NUMBER:** A text input field containing '7000'.
- NETWORK ID:** A text input field containing '1234'. To its right is a text description: 'Internal blockchain identifier of Ganache server.'
- AUTOMINE:** A toggle switch that is currently turned on (orange). To its right is a text description: 'Process transactions instantaneously.'
- ERROR ON TRANSACTION FAILURE:** A toggle switch that is currently turned on (orange). To its right is a text description: 'When transactions fail, throw an error. If disabled, transaction failures will only be detectable via the "status" flag in the transaction receipt. Disabling this feature will make Ganache handle transaction failures like other Ethereum clients.'

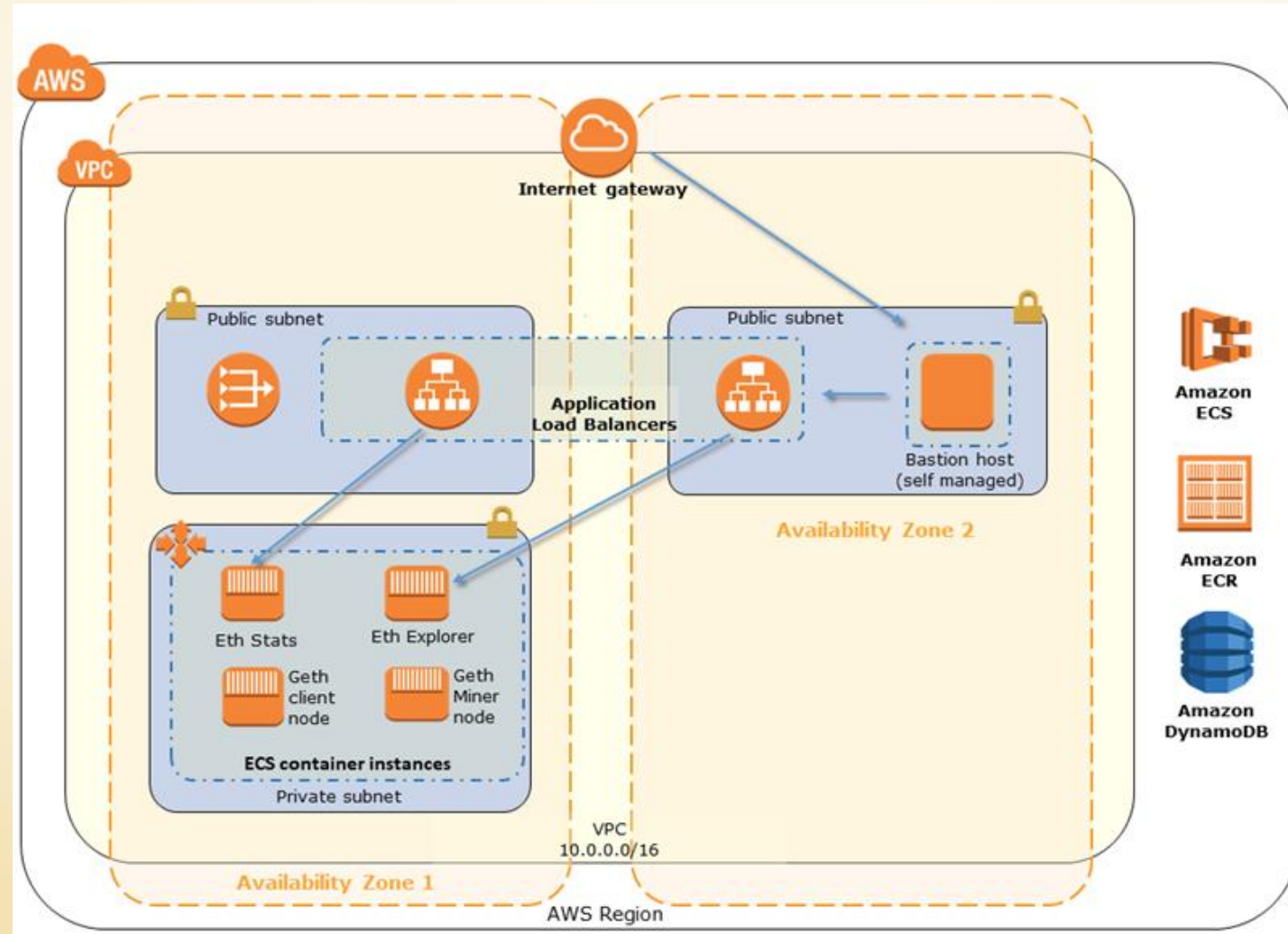
# On-Premise: Setting up the Development Environment (Cont.)

- Port and network id can be customized, post customization, please click on the restart button on the top left corner.
- This will reinitialize the node and RPC location will be `http://[host-name]:[port]`. In case of the setting as seen in the above screen the RPC endpoint will be `http://127.0.0.1:7000`
- Ganache installation is running and can be interacted with using the above-mentioned RPC endpoint.

# AWS Blockchain Framework for Ethereum

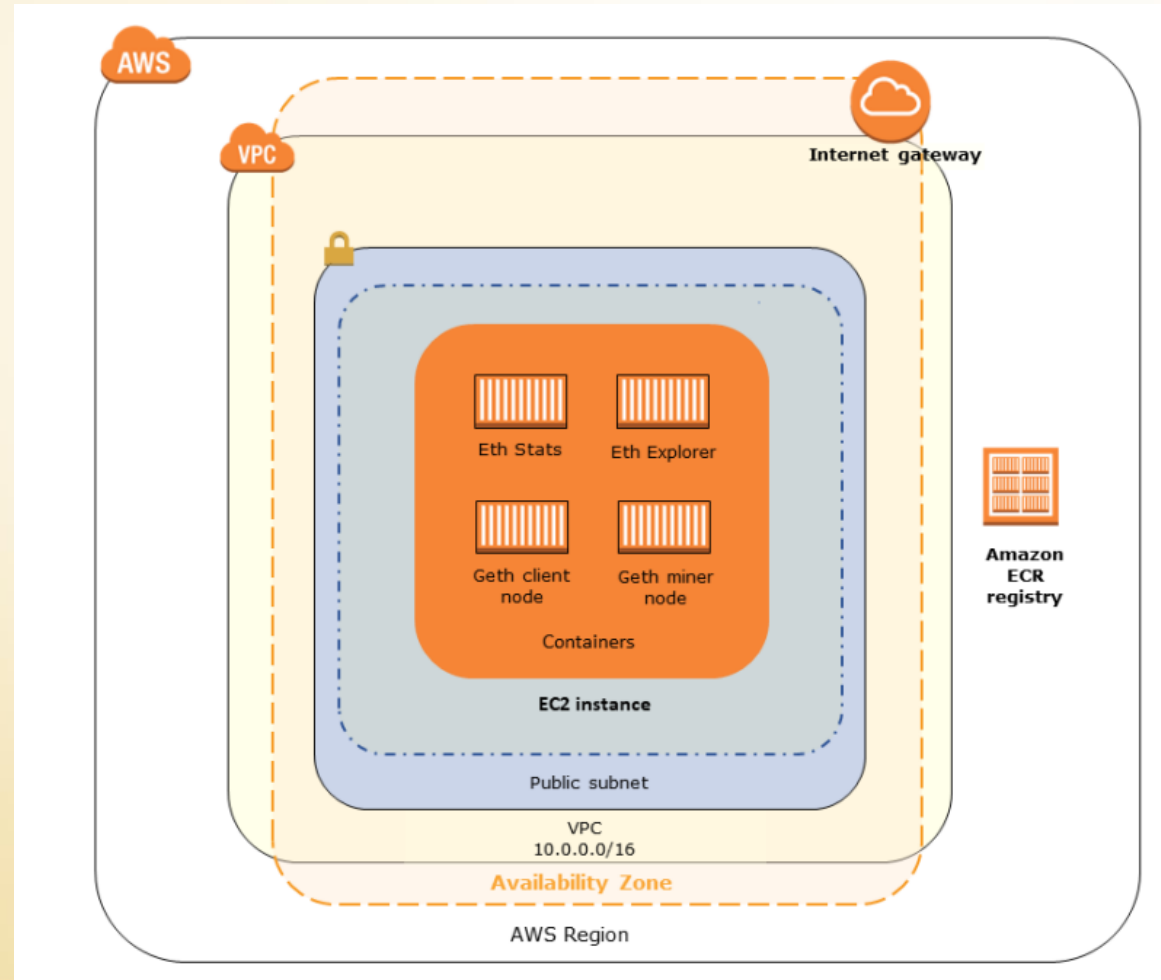
With Amazon ECS, you create your Ethereum network on an ECS cluster composed of multiple EC2 instances, with an Application Load Balancer and related resources.

The following diagram depicts an Ethereum network created using the template with the ECS container platform option:



# Using the Docker-Local Platform

Alternatively, you can launch Ethereum containers within a single Amazon EC2 instance. All containers run on a single EC2 instance. This is a simplified setup. The following diagram depicts an Ethereum network created using the template with the docker-local container platform option:



# AWS: Building Ethereum Environment

1. Create an IAM User
2. Create a Key Pair
3. Getting Started: Set Up Prerequisites
4. Create a VPC and Subnets
5. Create Security Groups
6. Create an IAM Role for Amazon ECS and an EC Instance Profile
7. Create a Bastion Host
8. Create the Ethereum Network
9. Connect to EthStats and EthExplorer Using the Bastion Host
10. Clean Up Resources
11. AWS Blockchain Templates and Features
12. AWS Blockchain Template for Ethereum
13. Links to Launch
14. Ethereum Options
15. Prerequisites
16. Connecting to Ethereum Resources
17. AWS Blockchain Template for Hyperledger Fabric
18. Links to Launch
19. AWS Blockchain Template for Hyperledger Fabric Components
20. Prerequisites
21. Connecting to Hyperledger Fabric Resources

## Create a VPC and Subnets

Use the Amazon VPC console (<https://console.aws.amazon.com/vpc/>) to create the Elastic IP address, the VPC, and the subnet as described below.

### **To create an Elastic IP address**

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. Choose Elastic IPs, Allocate new address, Allocate.
3. Make a note of the Elastic IP address that you create and choose Close.
4. In the list of Elastic IP addresses, find the Allocation ID for the Elastic IP address created earlier. You use this when you create the VPC.

## Create a VPC and Subnets

### To create the VPC

1. From the navigation bar, select a Region for the VPC. VPCs are specific to a Region, so select the same Region in which you created your key pair in and where you are launching the Ethereum stack.

For more information, see [Create a Key Pair](#) (p. 5).

2. On the VPC dashboard, choose Start VPC Wizard.

3. On the Step 1: Select a VPC Configuration page, choose VPC with Public and Private Subnets, Select.

4. On the Step 2: VPC with Public and Private Subnets page, leave IPv4 CIDR block and IPv6 CIDR

block to their default values. For VPC name, enter a friendly name.



## Create a VPC and Subnets

5. For Public subnet's IPv4 CIDR, leave the default value. For Availability Zone, choose a zone. For Public subnet name, enter a friendly name.

You specify this subnet as one of the first of two subnets for the Application Load Balancer when you use the template.

Note the Availability Zone of this subnet because you select the same Availability Zone for the private subnet, and a different one for the other public subnet.

6. For Private subnet's IPv4 CIDR, leave the default value. For Availability Zone, select the same Availability Zone as in the previous step. For Private subnet name, enter a friendly name.

7. For Elastic IP Allocation ID, select the Elastic IP address that you created earlier.

8. Leave the default values for other settings.

9. Choose Create VPC

# Create a VPC and Subnets

### VPC with Public and Private Subnets

---

**IPv4 CIDR block:**  (65531 IP addresses available)

**IPv6 CIDR block:** ☒ No IPv6 CIDR Block  
☐ Amazon provided IPv6 CIDR block

**VPC name:**

---

**Public subnet's IPv4 CIDR:**  (251 IP addresses available)

**Availability Zone:**  ▼

**Public subnet name:**

**Private subnet's IPv4 CIDR:**  (251 IP addresses available)

**Availability Zone:**  ▼

**Private subnet name:**

You can add more subnets after AWS creates the VPC.

---

Specify the details of your NAT gateway ([NAT gateway rates apply](#)). [Use a NAT instance instead](#)

**Elastic IP Allocation ID:**

---

**Service endpoints**

---

**Enable DNS hostnames:** ☒ Yes ☐ No

**Hardware tenancy:**  ▼

---

[Cancel and Exit](#)

## Create the second public subnet in a different Availability Zone

To create the second public subnet in a different Availability Zone

1. Choose Subnets and then select the public subnet that you created earlier from the list. Select the Route Table tab and note the Route table ID. You specify this same route table for the second public subnet below.
2. Choose Create Subnet.
3. For Name tag, enter a name for the subnet. You use this name later when you create the bastion host in this network.
4. For VPC, select the VPC that you created earlier.
5. For Availability Zone, select a different zone from the zone that you selected for the first public subnet.
6. For IPv4 CIDR block, enter 10.0.2.0/24.
7. Choose Yes, Create. The subnet is added to the list of subnets.
8. With the subnet selected from the list, choose Subnet Actions, Modify auto-assign IP settings. Select Auto-assign IPs, Save, Close. This allows the bastion host to obtain a public IP address when you create it in this subnet.
9. On the Route Table tab, choose Edit. For Change to, select the route table ID that you noted earlier and choose Save.



## To create the IAM role for Amazon ECS

To create the IAM role for Amazon ECS

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose Roles, Create Role.
3. Under Select type of trusted entity, choose AWS service.
4. For Choose the service that will use this role, choose Elastic Container Service.
5. Under Select your use case, choose Elastic Container Service, Next: Permissions.
6. For Permissions policy, leave the default policy (AmazonEC2ContainerServiceRole) selected, and choose Next: Review.
7. For Role name, enter a value that helps you identify the role, such a ECSRoleForEthereum. For Role Description, enter a brief summary. Note the role name for later.
8. Choose Create role.
9. Select the role that you just created from the list. If your account has many roles, you can search for the role name.

## To create the IAM role for Amazon ECS

**Create role**

Select type of trusted entity

- AWS service**  
EC2, Lambda and others
- Another AWS account  
Belonging to you or 3rd party
- Web identity  
Cognito or any OpenID provider
- SAML 2.0 federation  
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

**EC2**  
Allows EC2 instances to call AWS services on your behalf.

**Lambda**  
Allows Lambda functions to call AWS services on your behalf.

API Gateway	DMS	Elastic Transcoder	Machine Learning	SageMaker
Application Auto Scaling	Data Pipeline	ElasticLoadBalancing	MediaConvert	Service Catalog
Auto Scaling	DeepLens	Glue	OpsWorks	Step Functions
Batch	Directory Service	Greengrass	RDS	Storage Gateway
CloudFormation	DynamoDB	GuardDuty	Redshift	
CloudHSM	EC2	Inspector	Rekognition	
CloudWatch Events	EMR	IoT	S3	
CodeBuild	ElastiCache	Kinesis	SMS	
CodeDeploy	Elastic Beanstalk	Lambda	SNS	
Config	<b>Elastic Container Service</b>	Lex	SWF	

Select your use case

**EC2 Role for Elastic Container Service**  
Allows EC2 instances in an ECS cluster to access ECS.

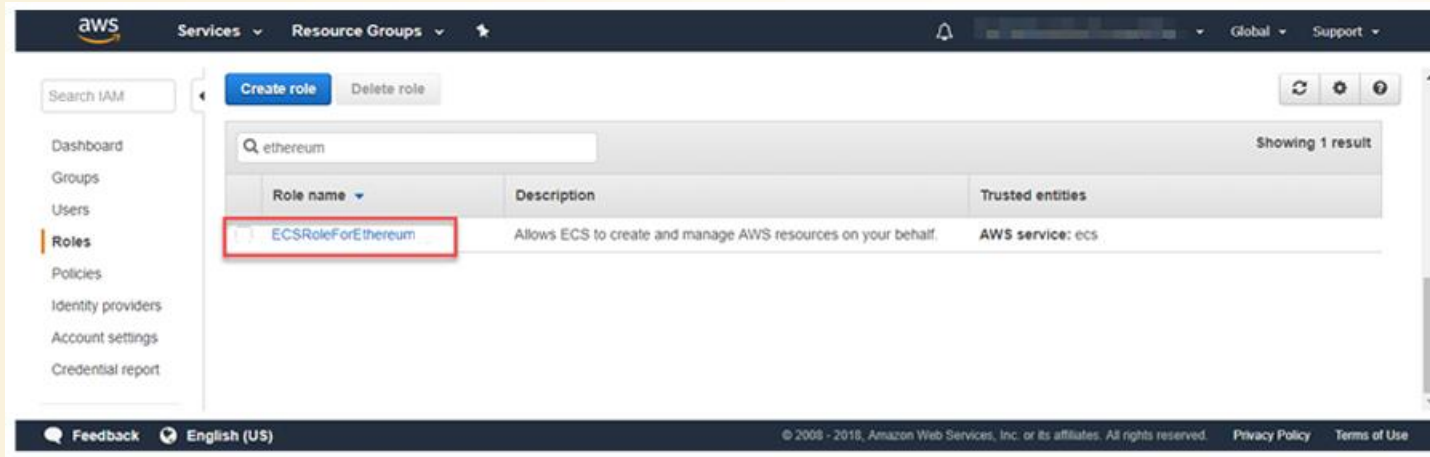
**Elastic Container Service**  
Allows ECS to create and manage AWS resources on your behalf.

\* Required

Cancel **Next: Permissions**

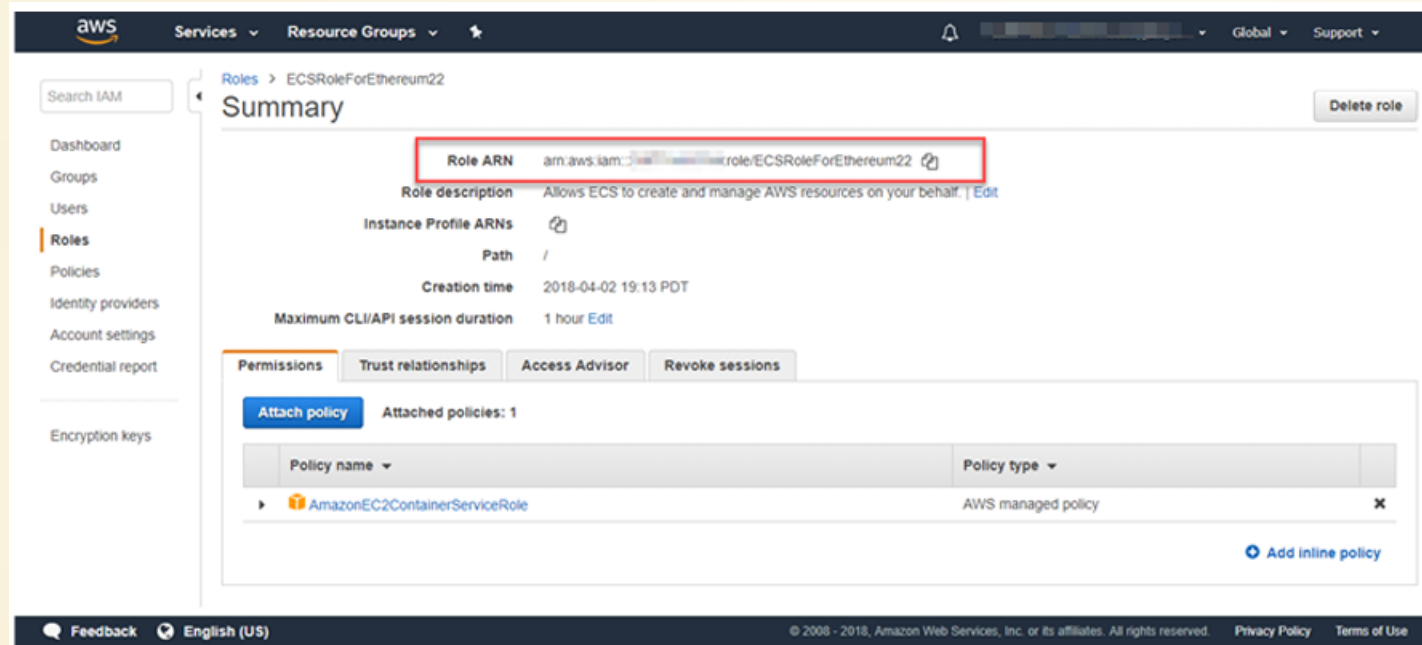
Feedback English (US)

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)



Select the role that you just created from the list. If your account has many roles, you can search for the role name.





Select the role that you just created from the list. If your account has many roles, you can search for the role name.



## To create an EC2 instance profile

1. In the navigation pane, choose Policies, Create policy.
2. Choose JSON and replace the default policy statement with the following JSON policy:

```
{  "Version": "Date",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:DeregisterContainerInstance",
        "ecs:DiscoverPollEndpoint",
        "ecs:Poll",
        "ecs:RegisterContainerInstance",
        "ecs:StartTelemetrySession",
        "ecs:Submit*",
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "dynamodb:BatchGetItem",
        "dynamodb:BatchWriteItem",
        "dynamodb:PutItem",
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:Scan",
        "dynamodb:Query",
        "dynamodb:UpdateItem"
      ],
      "Resource": "*"
    }
  ]
}
```

aws

Services

Resource Groups

Global

Support

Create policy

1

2

Review policy

Name\*

EthereumPolicyForEC2

Use alphanumeric and '\*+=, @-\_' characters. Maximum 128 characters.

Description

Permissions policy for EC2 instances in the Ethereum network.

Maximum 1000 characters. Use alphanumeric and '\*+=, @-\_' characters.

Summary

Q Filter

Service	Access level	Resource	Request condition
Allow (4 of 134 services) <a>Show remaining 130</a>			
<a>CloudWatch Logs</a>	Limited: Write	All resources	None
<a>DynamoDB</a>	Limited: Read, Write	All resources	None
<a>EC2 Container Registry</a>	Limited: Read	All resources	None
<a>EC2 Container Service</a>	Limited: Write	All resources	None

\* Required

Cancel

Previous

Create policy

Feedback

English (US)

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

In the above process, choose Review policy.  
For Name, enter a value that helps you identify this permissions policy, for example EthereumPolicyForEC2. For Description, enter a brief summary. Choose Create policy

aws Services Resource Groups

## Create role

1 2 3

### Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy Refresh

Filter: Policy type Q EthereumPolicyForEC2 Showing 1 result

	Policy name	Attachments	Description
<input checked="" type="checkbox"/>	EthereumPolicyForEC2	0	Permissions policy for EC2 instances in the Ethereum network.

\* Required Cancel Previous Next: Review

Feedback English (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

aws Services Resource Groups

Search IAM

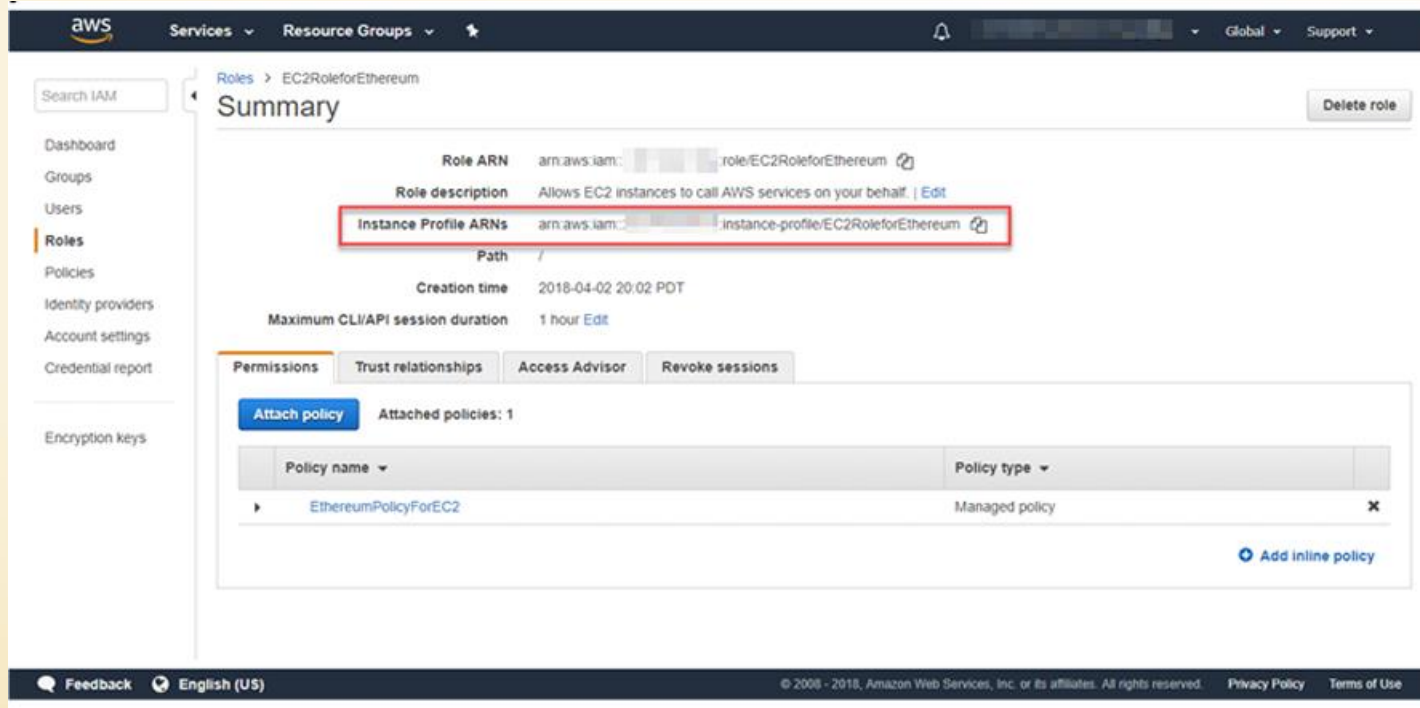
Dashboard Groups Users Roles Policies Identity providers Account settings

Create role Delete role

Q EC2RoleforEthereum Showing 1 result

	Role name	Description	Trusted entities
<input type="checkbox"/>	EC2RoleforEther...	Allows EC2 instances to call AWS ser...	AWS service: ec2

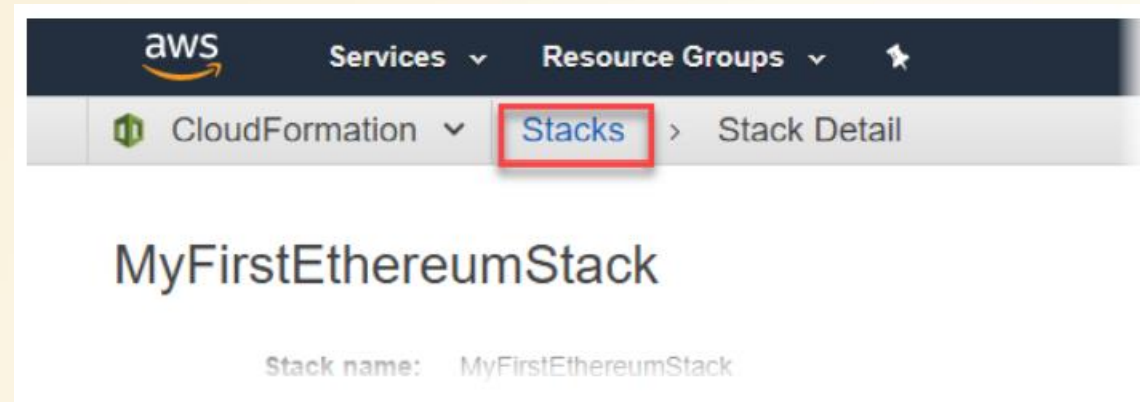
Feedback English (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



# To Create a Bastion Host

1. Follow the first five steps to Launch an Instance in the Amazon EC2 User Guide for Linux Instances.
2. Choose Edit Instance Details. For Network, choose the VPC you created earlier, for Subnet select the second public subnet that you created earlier. Leave all other settings to their defaults.
3. Confirm the change when prompted, and then choose Review and Launch.
4. Choose Edit Security Groups. For Assign a security group, choose Select an existing security group.
5. From the list of security groups, select the security group for the Application Load Balancer that you created earlier, and then choose Review and Launch.
6. Choose Launch.
7. Note the instance ID. You need it later when you

# Create the Ethereum Network



# Cloud Formation: Stack

When all stacks show `CREATE_COMPLETE` for Status, you can connect to Ethereum user interfaces to verify that the network is running and accessible. When you use the ECS container platform, URLs for connecting to EthStats, EthExplorer, and EthJsonRPC through the Application Load Balancer are available on the Outputs tab of the root stack.

## **Important**

You won't be able to connect directly to these URLs or SSH directly until you set up a proxy connection through the bastion host on your client computer

# Cloud Formation: Stack

The screenshot shows the AWS CloudFormation console. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', a search icon, a notification bell, a user profile, and region/account info ('Oregon', 'Support'). Below this is a breadcrumb trail: 'CloudFormation' > 'Stacks'. The main area has a 'Create Stack' button, an 'Actions' dropdown, and a 'Design template' button. A filter bar shows 'Filter: Active' and a search box 'By Stack Name'. It indicates 'Showing 4 stacks'. A table lists the stacks:

	Stack Name	Created Time	Status	Description
<input type="checkbox"/>	MyFirstEthereumStack-Ether... NESTED	2018-04-12 13:26:46 UTC-0700	CREATE_COMPLETE	This template creates an AutoScalingGroup of EC2 I...
<input type="checkbox"/>	MyFirstEthereumStack-Ether... NESTED	2018-04-12 13:26:38 UTC-0700	CREATE_COMPLETE	This template creates the ECS cluster and Ethereu...
<input type="checkbox"/>	MyFirstEthereumStack-Ether... NESTED	2018-04-12 13:25:59 UTC-0700	CREATE_COMPLETE	This template deploys an Ethereum cluster on an ex...
<input checked="" type="checkbox"/>	MyFirstEthereumStack	2018-04-12 13:25:54 UTC-0700	CREATE_COMPLETE	This template creates an Ethereum network on an A...

Below the table is a tabbed interface with 'Overview' selected. The 'Outputs' tab is active, showing a table of stack outputs:

Key	Value	Description	Export Name
EthStatsURL	http://MyFir-...us-west-2.elb.amazonaws.com	Visit this URL to see the status of your ...	
EthExplorerURL	http://MyFir-...us-west-2.elb.amazonaws.com:8080	Visit this URL to view transactions on yo...	
EthJsonRPCURL	http://MyFir-...us-west-2.elb.amazonaws.com:8545	Use this URL to access the Geth JSON ...	

The footer contains 'Feedback', 'English (US)', copyright information '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.', and links to 'Privacy Policy' and 'Terms of Use'.



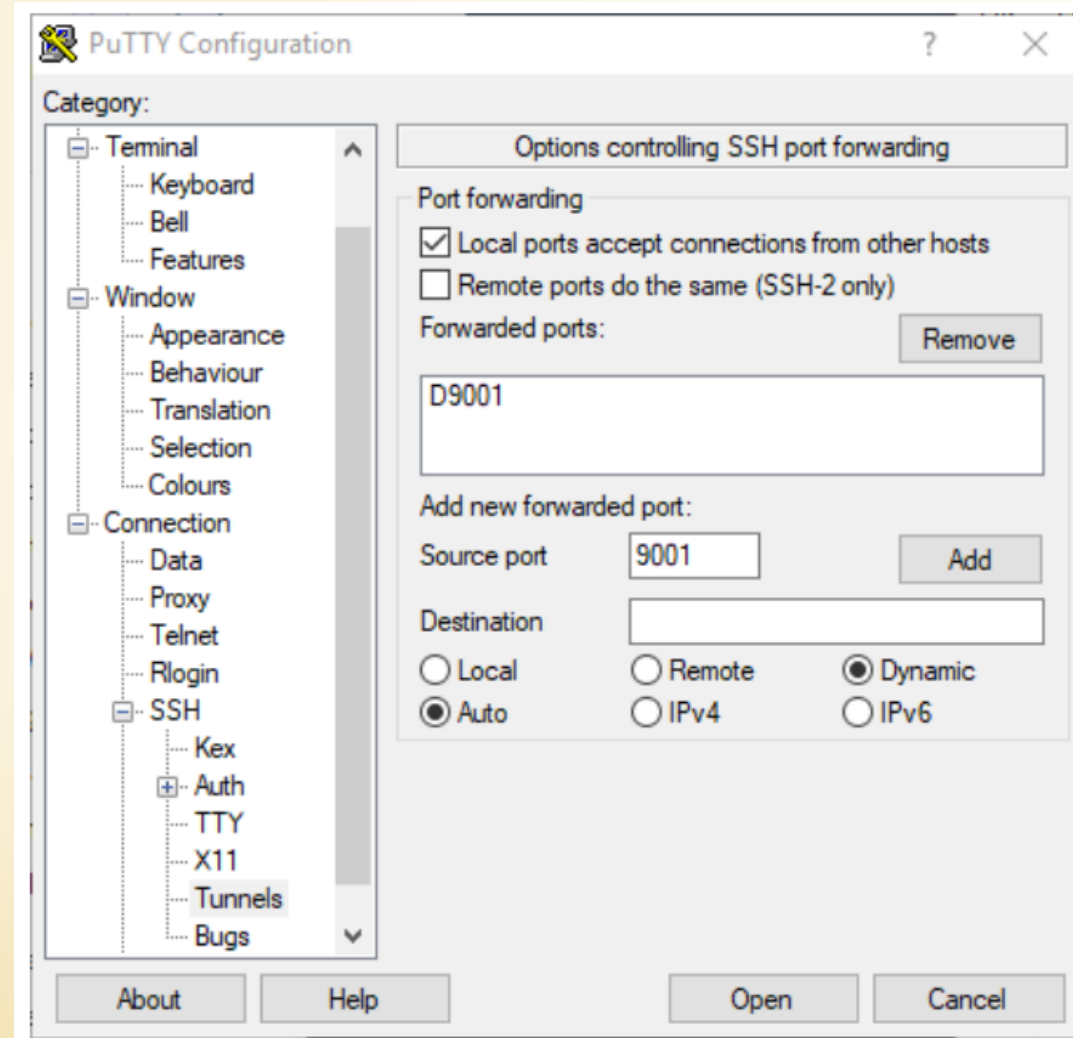
# Connect to EthStats and EthExplorer Using the Bastion Host

To connect to the bastion host with SSH port forwarding using PuTTY (Windows)

1. Follow the procedures in Connecting to Your Linux Instance from Windows Using PuTTY in the Amazon EC2 User Guide for Linux Instances through step 7 of the Starting a PuTTY Session procedure, using the same key pair that you specified in the AWS Blockchain Template for Ethereum configuration.
2. In PuTTY, under Category, choose Connection, SSH, Tunnels.
3. For Port forwarding, choose Local ports accept connections from other hosts.
4. Under Add new forwarded port:
  - a. For Source port, enter 9001. This is an arbitrary unused port that we chose, and you can choose a different one if necessary.
  - b. Leave Destination blank.
  - c. Select Dynamic.
  - d. Choose Add.

To connect to the bastion host with SSH port forwarding using PuTTY (Windows)

# Connect to EthStats and EthExplorer Using the Bastion Host



To connect to the bastion host with SSH port forwarding using PuTTY (Windows)

# Connect to EthStats and EthExplorer Using the Bastion Host


Choose Open and then authenticate to the bastion host as required by your key configuration. Leave the connection open. With the PuTTY connection open, you now configure your system or a browser extension to use the forwarded port for your Ethereum network URLs.

The following instructions are based on using FoxyProxy Standard to forward connections based on the URL pattern of EthStats and EthExplorer and port 9001, which you established earlier as the forwarded port, but you can use any method that you prefer.

## To configure FoxyProxy to use the SSH tunnel for Ethereum network URLs

This procedure was written based on Chrome. If you use another browser, translate the settings and sequence to the version of FoxyProxy for that browser.

1. Download and install the FoxyProxy Standard browser extension, and then open **Options** according to the instructions for your browser.
2. Choose **Add New Proxy**.
3. On the **General** tab, make sure that the proxy is **Enabled** and enter a **Proxy Name** and **Proxy Notes** that help you identify this proxy configuration.
4. On the **Proxy Details** tab, choose **Manual Proxy Configuration**. For **Host or IP Address** (or **Server or IP Address** in some versions), enter *localhost*. For **Port**, enter *9001*. Select **SOCKS Proxy?**.
5. On the **URL Pattern** tab, choose **Add New Pattern**.



# Using the *AWS* Blockchain Template for Ethereum

# Using the AWS Blockchain Template for Ethereum

- AWS Blockchain Templates helps you quickly create and deploy blockchain networks on AWS using different blockchain frameworks. Blockchain is a decentralized database technology that maintains a continually growing set of transactions and smart contracts hardened against tampering and revision using cryptography.
- A blockchain network is a peer-to-peer network that improves the efficiency and immutability of transactions for business processes like international payments, supply chain management, land registration, crowd funding, governance, financial transactions, and more. This allows people and organizations who may not know one another to trust and independently verify the transaction record.
- You use AWS Blockchain Templates to configure and launch AWS CloudFormation stacks to create blockchain networks

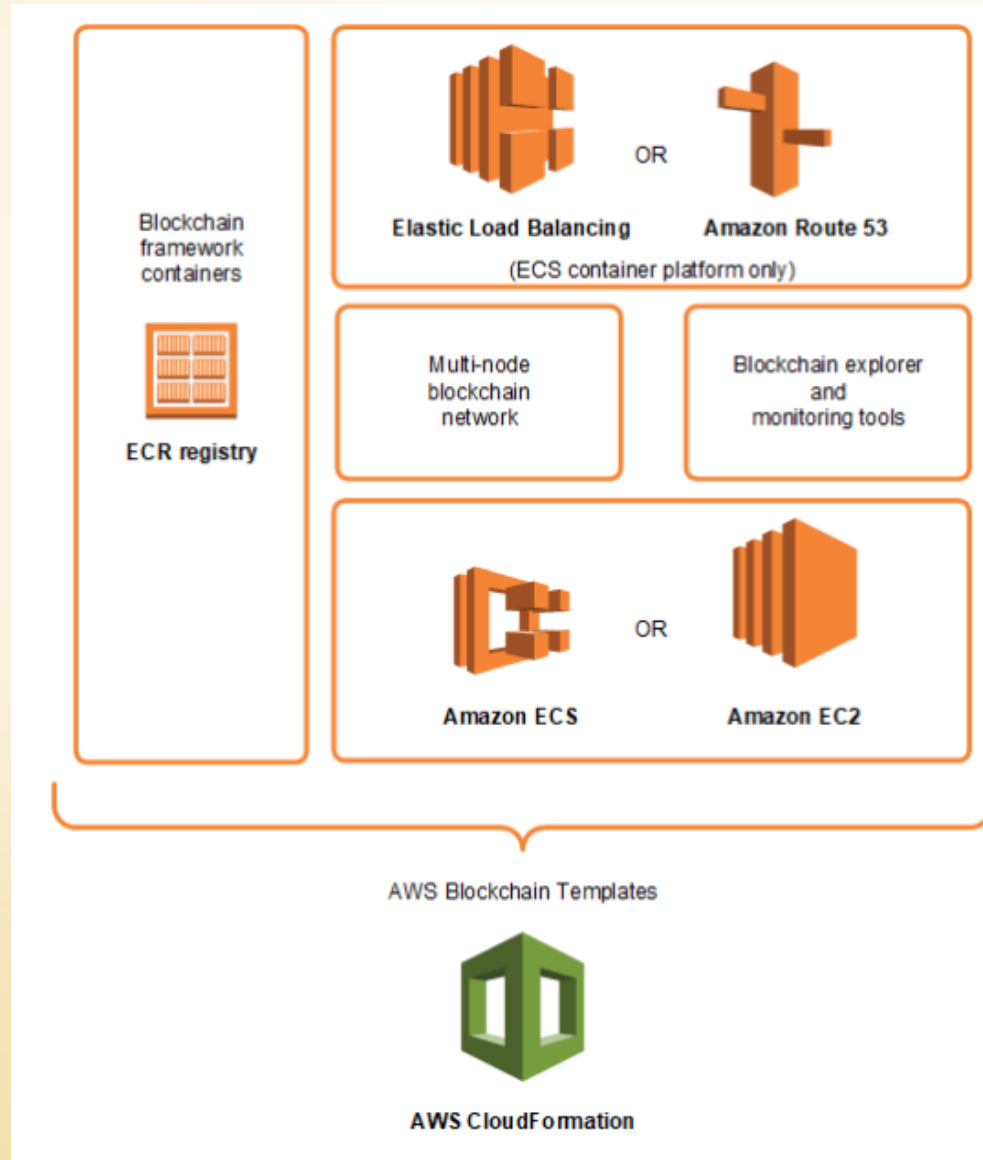
# Using the AWS Blockchain Template for Ethereum

## Requirements and Process:

- Choosing the Container Platform
- Choosing a Private or Public Ethereum Network
- Changing the Default Accounts and Mnemonic Phrase

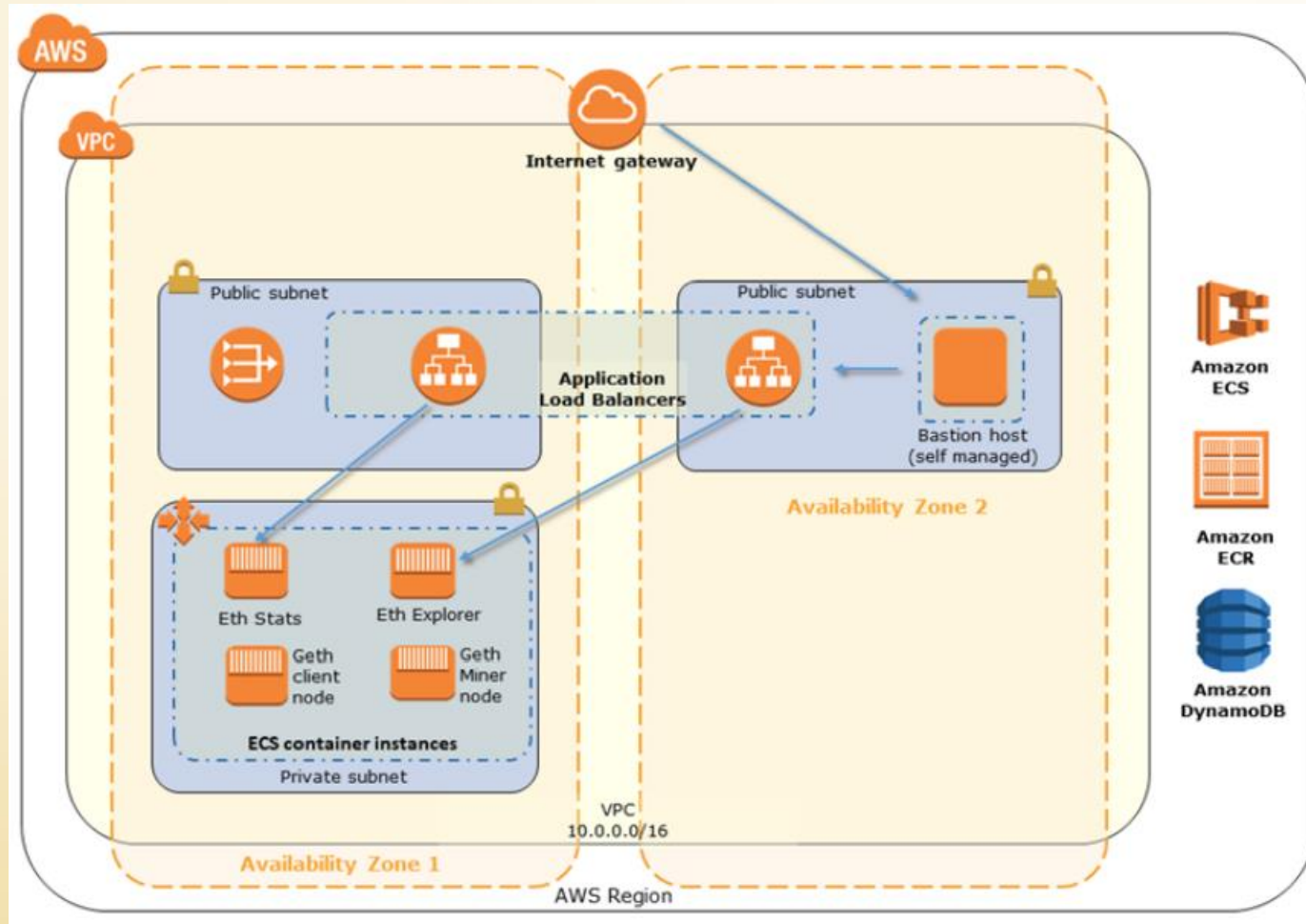
# Using the AWS Blockchain Template for Ethereum

The fundamental components of a blockchain network on AWS created using AWS Blockchain Templates are shown in the following diagram.



# Choosing the Container Platform

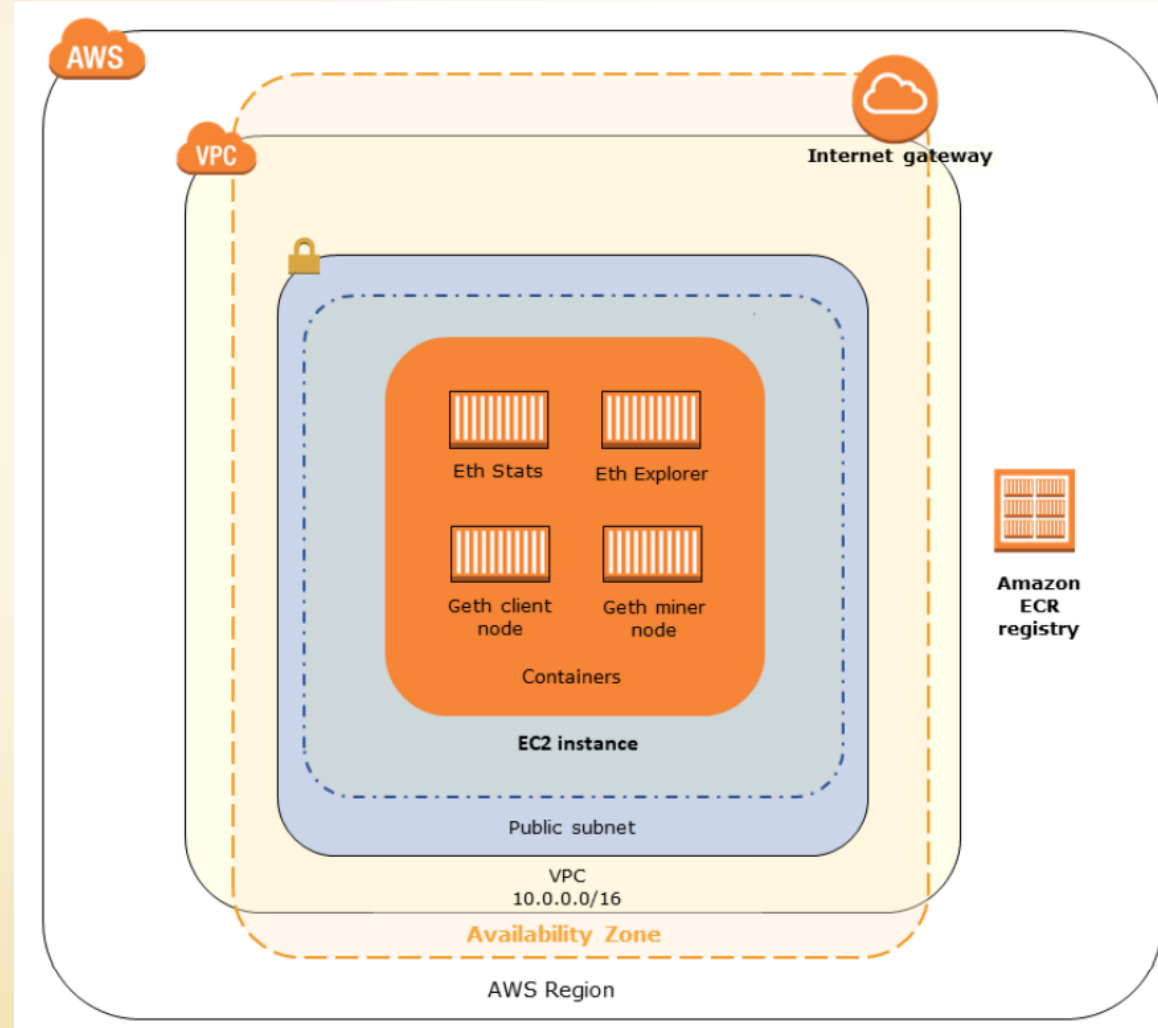
## Using the Amazon ECS Container Platform





# Choosing the Container Platform

## Using the Docker-Local Platform



# Creating the ECS Role and Permissions

**Create role**

Select type of trusted entity

**AWS service**  
EC2, Lambda and others

Another AWS account  
Belonging to you or 3rd party

Web identity  
Cognito or any OpenID provider

SAML 2.0 federation  
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

**EC2**  
Allows EC2 instances to call AWS services on your behalf.

**Lambda**  
Allows Lambda functions to call AWS services on your behalf.

API Gateway	DMS	Elastic Transcoder	Machine Learning	SageMaker
Application Auto Scaling	Data Pipeline	ElasticLoadBalancing	MediaConvert	Service Catalog
Auto Scaling	DeepLens	Glue	OpsWorks	Step Functions
Batch	Directory Service	Greengrass	RDS	Storage Gateway
CloudFormation	DynamoDB	GuardDuty	Redshift	
CloudHSM	EC2	Inspector	Rekognition	
CloudWatch Events	EMR	IoT	S3	
CodeBuild	ElastiCache	Kinesis	SMS	
CodeDeploy	Elastic Beanstalk	Lambda	SNS	
Config	<b>Elastic Container Service</b>	Lex	SWF	

Select your use case

**EC2 Role for Elastic Container Service**  
Allows EC2 instances in an ECS cluster to access ECS.

**Elastic Container Service**  
Allows ECS to create and manage AWS resources on your behalf.

\* Required

Cancel **Next: Permissions**

Feedback English (US)

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

# Create Role and ARN

This screenshot shows the AWS IAM console interface. On the left is a navigation menu with options like Dashboard, Groups, Users, Roles, Policies, Identity providers, Account settings, and Credential report. The 'Roles' section is selected. At the top, there are buttons for 'Create role' and 'Delete role'. A search bar contains the text 'ethereum', and a message indicates 'Showing 1 result'. Below the search bar is a table with columns 'Role name', 'Description', and 'Trusted entities'. One role is listed: 'ECSRoleForEthereum' with the description 'Allows ECS to create and manage AWS resources on your behalf.' and 'Trusted entities' as 'AWS service: ecs'. The role name is highlighted with a red box.

This screenshot shows the details page for the role 'ECSRoleForEthereum22' in the AWS IAM console. The breadcrumb navigation shows 'Roles > ECSRoleForEthereum22'. The page title is 'Summary'. A 'Delete role' button is in the top right. The role's details are listed: 'Role ARN' is 'arn:aws:iam::[account-id]:role/ECSRoleForEthereum22' (highlighted with a red box), 'Role description' is 'Allows ECS to create and manage AWS resources on your behalf.', 'Instance Profile ARNs' is empty, 'Path' is '/', 'Creation time' is '2018-04-02 19:13 PDT', and 'Maximum CLI/API session duration' is '1 hour'. Below this are tabs for 'Permissions', 'Trust relationships', 'Access Advisor', and 'Revoke sessions'. The 'Permissions' tab is active, showing 'Attach policy' and 'Attached policies: 1'. A table lists the attached policy: 'AmazonEC2ContainerServiceRole' (AWS managed policy). An 'Add inline policy' button is at the bottom right.

# Ethereum Blockchain

Solidity is a contract-oriented, high-level language for implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).

# Ethereum Blockchain

Blockchain technologies consist of three main components:

1. An immutable ledger database
2. A consensus mechanism
3. A smart-contract execution environment.

Every member in a blockchain network owns a copy of the ledger database, allowing the transaction history to be independently verifiable.

The ledger database contains two components: a journal that holds the cryptographically immutable history of all transactions and the world state that shows the current state of the data derived from these transactions.

# Ethereum Blockchain Network on AWS

**Scenario:** Deploy smart contracts to your private Ethereum blockchain network on AWS

# Building Steps: Ethereum Blockchain Network on AWS

Step 1  
& 2

- Deploy the CloudFormation template
- Review the genesis block

Step 3  
& 4

- Create a static node mapping
- Initialize the Ethereum client

Step 5  
& 6

- Start geth
- Interact with your network by using the Ethereum Wallet app

Step 7  
& 8

- Add an account
- Start mining

Step 9  
& 10

- See the accounts that you have registered
- Start mining with two threads

Step 11

- Create, deploy, and test your smart contract

# AWS Blockchain Templates

- AWS Blockchain Templates provide a fast and easy way to create and deploy blockchain networks using popular open-source frameworks.
- Components required in the building Blockchain Template are as below:
- Ethereum Network Status – a web-based application to monitor the health of your Ethereum network.
- Ethereum Explorer – a block explorer for Ethereum.
- Ethereum JSON-RPC – a stateless, lightweight remote procedure call (RPC) protocol.

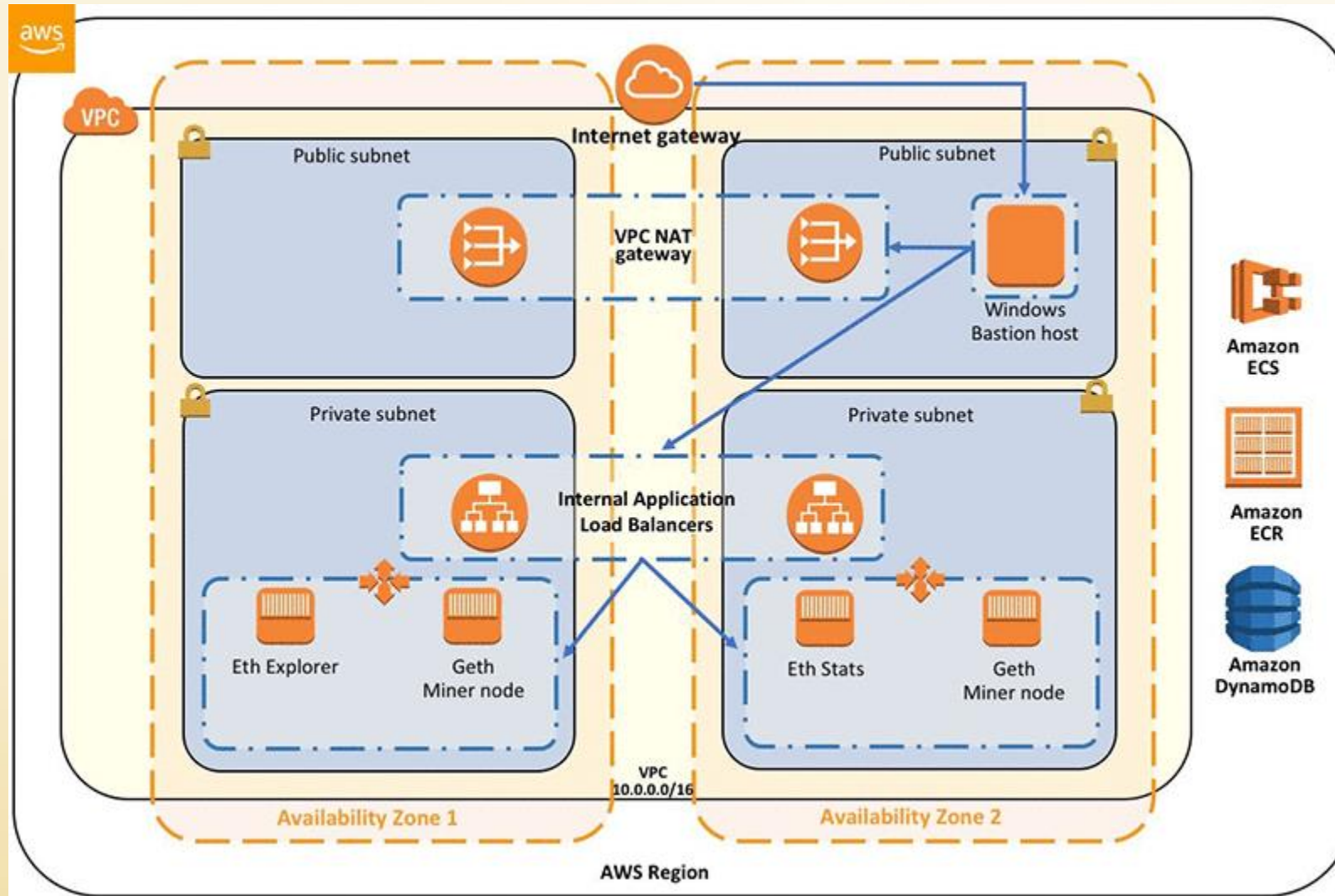
## Prerequisites: Building Ethereum blockchain

Below topic knowledge is required on:

1. AWS Blockchain Templates
2. Ethereum
3. Solidity
4. Smart contracts.



# AWS Blockchain Templates with a Windows Bastion Host.



## Step 1: Deploy the CloudFormation template

To start, deploy the CloudFormation template from below Link: Click [Here](#)

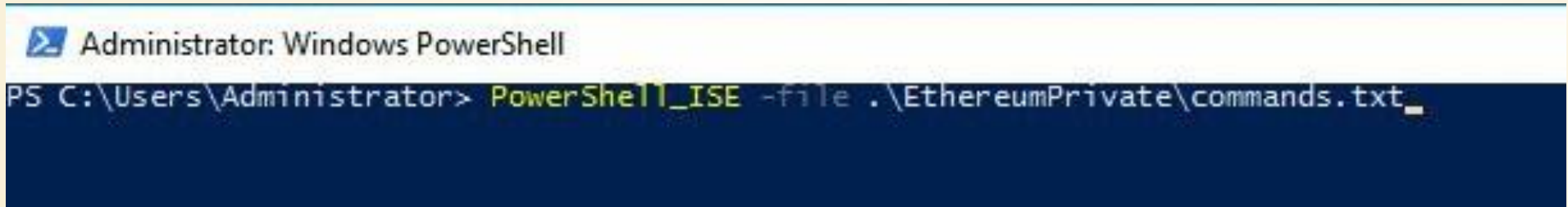
- This template which installs a Private Ethereum Blockchain network with preconfigured defaults and associated software for you to use for further implementation process.
- After the templates are deployed, log in to your Windows Bastion Host by using Remote Desktop Protocol (RDP).

The Windows Bastion Host has two client tools preinstalled that we use to connect to the blockchain:

1. **Ethereum Wallet** – the Ethereum GUI that enables you to hold and secure ether and other cryptoassets built on Ethereum, and also write, deploy, and use smart contracts.
2. **geth** – the command line interface for running a full Ethereum node implemented in Go. You can launch geth with an interactive console that provides a JavaScript runtime environment, exposing a JavaScript API to interact with your node.

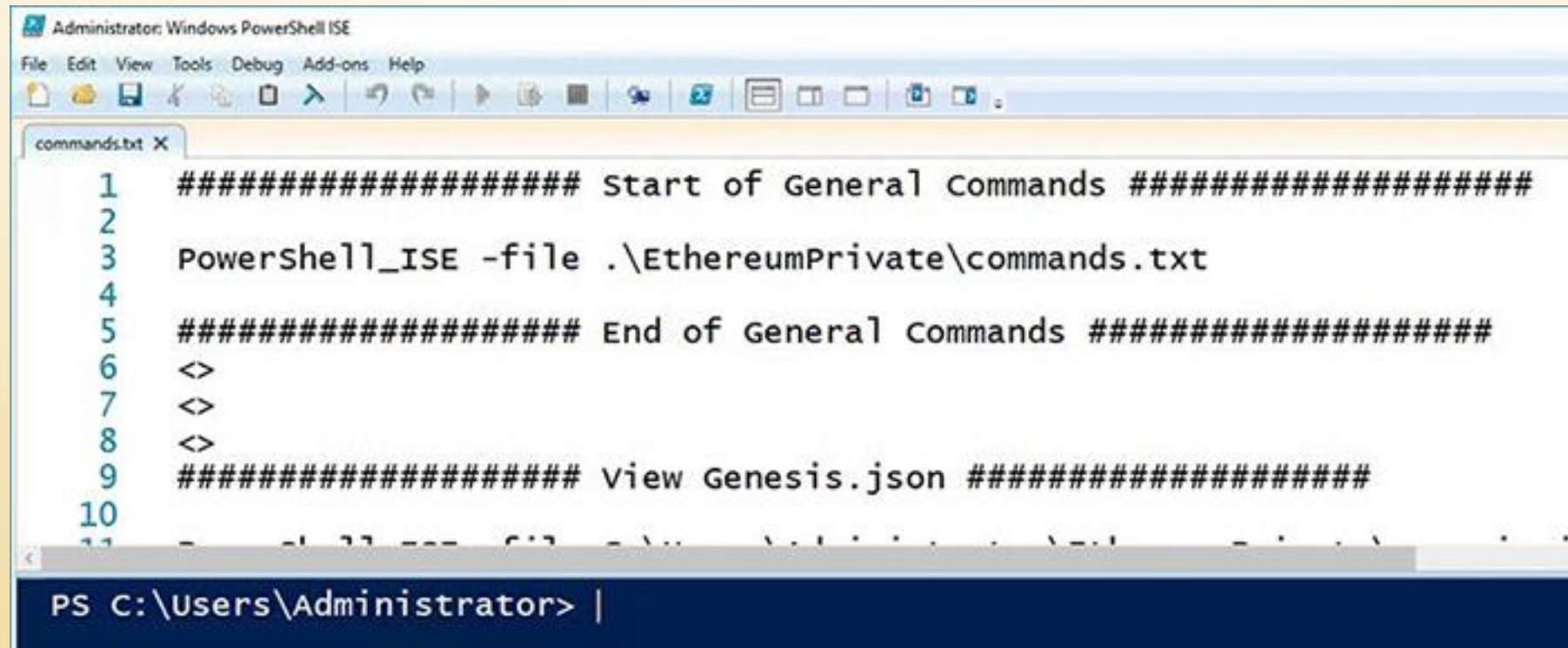
# Step 1: Deploy the CloudFormation template (Cont.)

## Running commands

A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell". The command prompt shows the command `PowerShell_ISE -file .\EthereumPrivate\commands.txt` being entered at the prompt `PS C:\Users\Administrator>`.

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> PowerShell_ISE -file .\EthereumPrivate\commands.txt
```

Accessing your commands.txt file.

A screenshot of the Windows PowerShell ISE editor window titled "Administrator: Windows PowerShell ISE". The editor shows a file named "commands.txt" with the following content:

```
1 ##### start of General Commands #####
2
3 PowerShell_ISE -file .\EthereumPrivate\commands.txt
4
5 ##### End of General Commands #####
6 <>
7 <>
8 <>
9 ##### view Genesis.json #####
10
11
```

The editor window includes a menu bar (File, Edit, View, Tools, Debug, Add-ons, Help) and a toolbar. At the bottom, a PowerShell prompt is visible: `PS C:\Users\Administrator> |`.

```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
commands.txt X
1 ##### start of General Commands #####
2
3 PowerShell_ISE -file .\EthereumPrivate\commands.txt
4
5 ##### End of General Commands #####
6 <>
7 <>
8 <>
9 ##### view Genesis.json #####
10
11
PS C:\Users\Administrator> |
```

## Step 2: Review the genesis block

Every blockchain starts with the genesis block. When you run **geth** with default settings for the first time, the main net genesis block is committed to the database. For a private network, you usually want a different genesis block.

- ❑ A folder called C:\Users\Administrator\EthereumPrivate has been created to store your private blockchain files. Inside this folder is a file called genesis.json.
- ❑ This file is used to create a genesis block that is compatible with the private network you've built using the AWS Blockchain Templates.
- ❑ Doing this enables you to synchronize your node with the private network hosted on Amazon ECS.

Open the file and review its contents. It should look similar to the following.

```

{
  "config":{
    "chainId":1234,
    "homesteadBlock":0,
    "eip155Block":0,
    "eip158Block":0},
    "nonce":"0x0000000000000001",
    "mixhash":"0x0000000000000000000000000000000000000000000000000000000000000000",
    "difficulty":"0x1",
    "coinbase":"0x000000000000000000000000000000000000000000000000",
    "timestamp":"0x00",
    "parentHash":"0x0000000000000000000000000000000000000000000000000000000000000000",
    "gasLimit":"0x7a1200",

    "alloc":{
      "0x0ADfCCa4B2a1132F82488546AcA086D7E24EA324":
        {"balance":"100000000000000000000"},

      "0x0bd5EebDC3E53973dDF236D43906C776a5fE3784":
        {"balance":"100000000000000000000"},

      "0x9537cb86f5a03C8CCB52c44b49757861eCA0004b":
        {"balance":"100000000000000000000"},

      "0x1Fbc353788338F902630E5494aD7FaC7dF8dBb29":
        {"balance":"100000000000000000000"},

      "0x5ccBe3B9B15eFB62bB2696051091Ee7C1Eb4c7E6":
        {"balance":"100000000000000000000"}

    }
  }
}

```

## Step 3: Create a Static Node Mapping

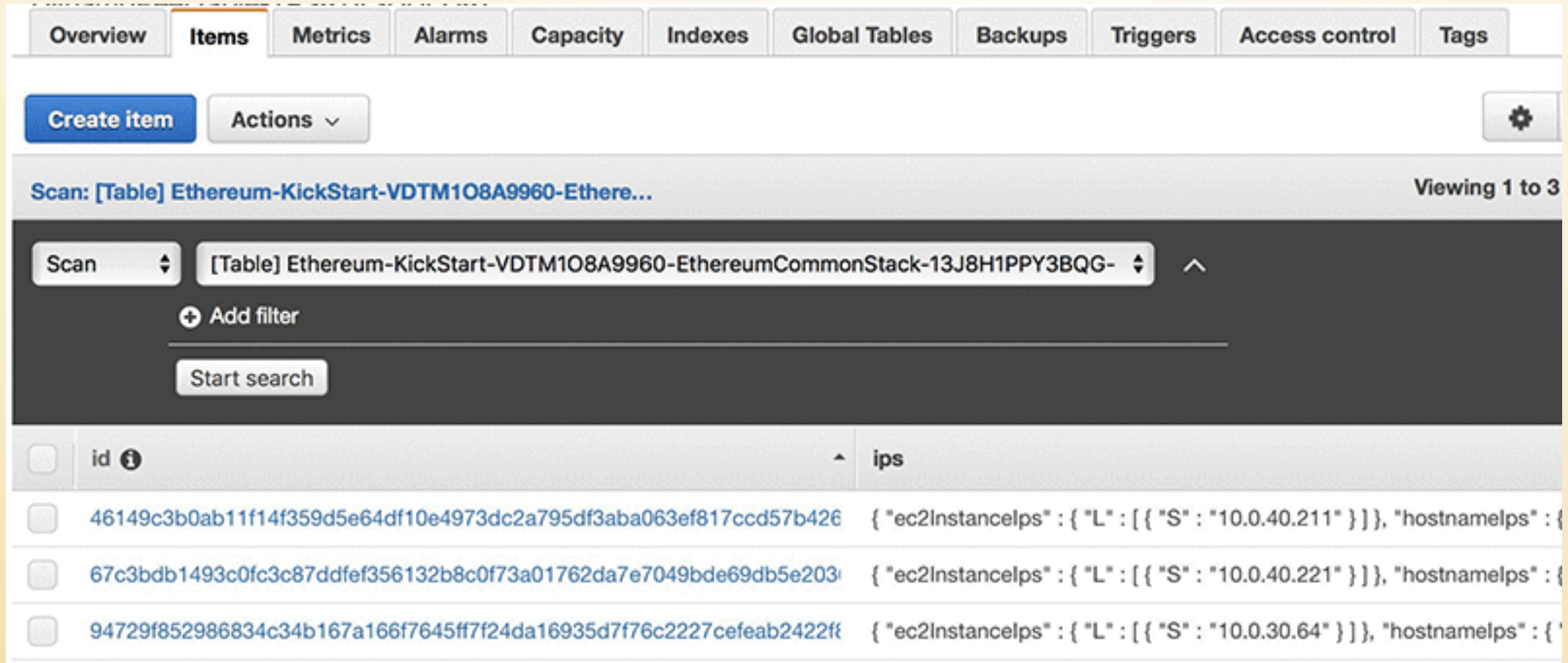
A static node mapping tells your node what other nodes to communicate with in the network. A file for doing this has already been created for you, but you still need to edit it.

1. In our setup, we set the static nodes as the ones currently running as tasks in Amazon ECS. Doing this allows your local node to participate in the same network as the one hosted on AWS.
2. Browse to the following file directory on your Windows Bastion Server:
3. `C:\Users\Administrator\EthereumPrivate\geth\`
4. Now open the file called `static-nodes.json` in this directory and populate it with the following node information from your Amazon DynamoDB table.



## Step 3: Create a Static Node Mapping (Cont.)

The following screenshot shows an Ethereum node and IP information.



The screenshot displays the AWS Management Console interface for a specific resource. The top navigation bar includes tabs for Overview, Items, Metrics, Alarms, Capacity, Indexes, Global Tables, Backups, Triggers, Access control, and Tags. Below the navigation bar, there are buttons for 'Create item' and 'Actions'. The main content area shows a table titled 'Scan: [Table] Ethereum-KickStart-VDTM1O8A9960-Ethere...'. The table has columns for 'id' and 'ips'. Three rows of data are visible, each representing an Ethereum node with its unique ID and associated IP information.

id	ips
46149c3b0ab11f14f359d5e64df10e4973dc2a795df3aba063ef817ccd57b426	{ "ec2InstanceIps" : { "L" : [ { "S" : "10.0.40.211" } ] }, "hostnamesIps" : {
67c3bdb1493c0fc3c87ddfef356132b8c0f73a01762da7e7049bde69db5e203	{ "ec2InstanceIps" : { "L" : [ { "S" : "10.0.40.221" } ] }, "hostnamesIps" : {
94729f852986834c34b167a166f7645ff7f24da16935d7f76c2227cefeab2422f	{ "ec2InstanceIps" : { "L" : [ { "S" : "10.0.30.64" } ] }, "hostnamesIps" : {

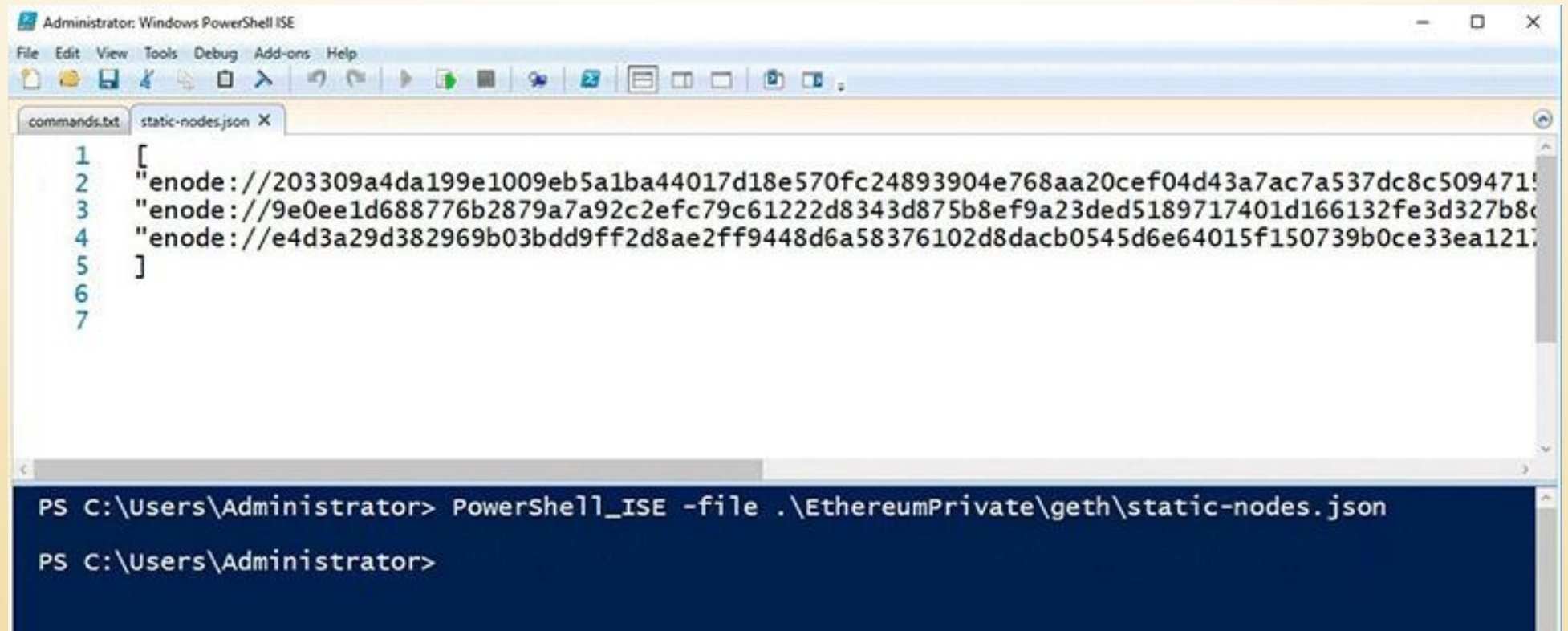
```
[  
  "enode://<id>@<ip address>:30303",  
  "enode://<id>@<ip address>:30303",  
  "enode://<id>@<ip address>:30303"  
]
```

## Step 3: Create a Static Node Mapping (Cont.)

We assume that you are still logging in to Windows with your Administrator account. As a best practice, we recommend that you set up an alternate non-Administrator user after you are familiar with the steps outlined in this blog post.

To edit the file static-nodes.json, run the following PowerShell command:

PowerShell\_ISE -file .\EthereumPrivate\geth\static-nodes.json

The screenshot shows the Windows PowerShell ISE interface. The title bar reads "Administrator: Windows PowerShell ISE". The menu bar includes "File", "Edit", "View", "Tools", "Debug", "Add-ons", and "Help". The toolbar contains various icons for file operations and execution. Two tabs are open: "commands.txt" and "static-nodes.json". The "static-nodes.json" tab is active, displaying a JSON array of three enode URLs. The content is as follows:

```
1  [  
2  "enode://203309a4da199e1009eb5a1ba44017d18e570fc24893904e768aa20cef04d43a7ac7a537dc8c509471!  
3  "enode://9e0ee1d688776b2879a7a92c2efc79c61222d8343d875b8ef9a23ded5189717401d166132fe3d327b8c  
4  "enode://e4d3a29d382969b03bdd9ff2d8ae2ff9448d6a58376102d8dacb0545d6e64015f150739b0ce33ea121!  
5  ]  
6  
7
```

The bottom of the window shows the PowerShell command prompt with the command `PowerShell_ISE -file .\EthereumPrivate\geth\static-nodes.json` entered and executed. The prompt is `PS C:\Users\Administrator>`.



## Step 3: Create a Static Node Mapping (Cont.)

As part of the CloudFormation stack deployment, you look up your enode ID and accompanying IP address in the DynamoDB table created. In addition, make the necessary changes to your static-node.json file. Remember to save your changes before closing the file.

You should now have a file with corresponding configuration information relevant to your environment. Your file is located here:

C:\Users\Administrator\EthereumPrivate\geth\static-nodes.json

Note: Don't use the configuration file following, which we provide only as an example.

An example static-nodes.json file is shown following.

```
[  
  "enode://18c1c4869d9e54b75ce968c1726b4ada18732ab5b577c2d0a35d85384e03def3e61a948dd0a1ba968d675c849  
  "enode://681867148e14ee9c6f1fb839e9f0674651e3e27eb1f1e15d3b2ae7e3de1640e6af3441cc06ea686db7f75b07a  
  "enode://8ac49be3a5da121ca7dfffd74e613a2d490771d0328da05f8c686192d6dbf8f9055e5935101498451a48c13d7e  
]
```

## Step 4: Initialize the Ethereum client

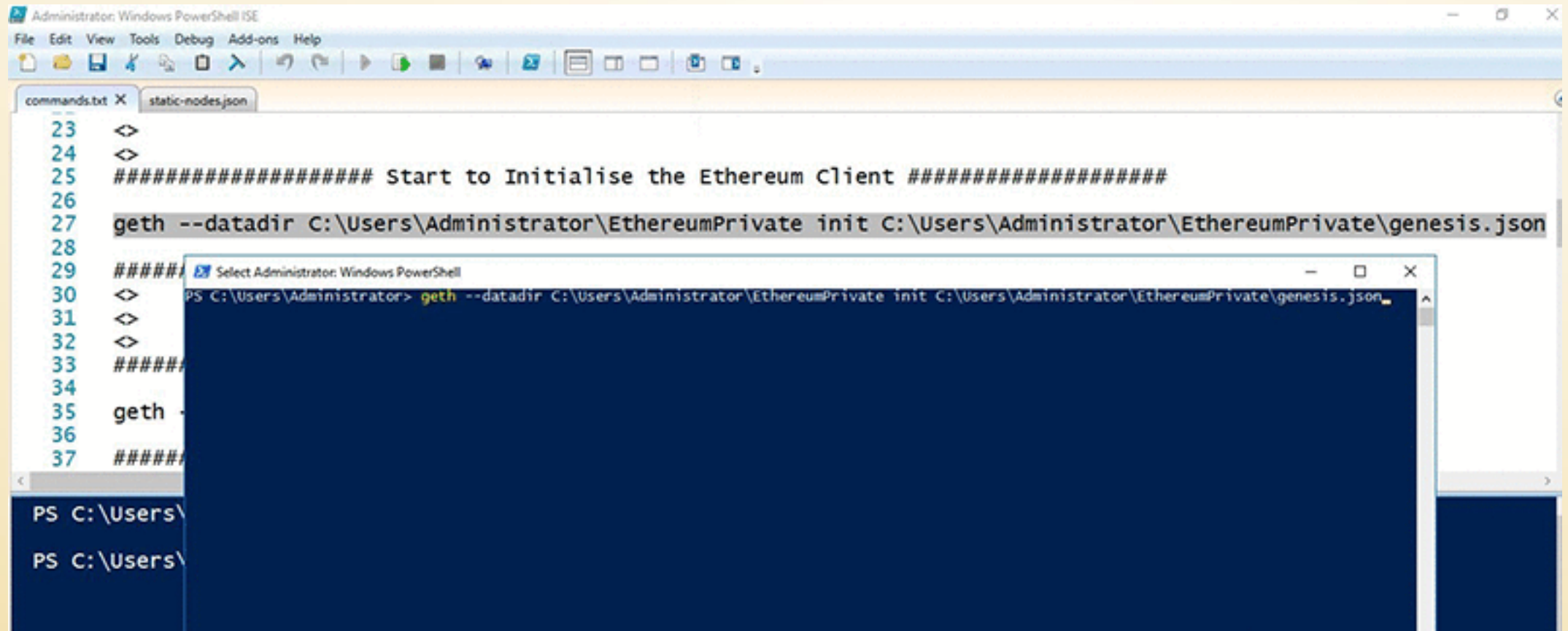
Now with your genesis.json and static-nodes.json files in place, it's time to initialize the Ethereum client.

First, you need to initialize your geth client to use the genesis block defined in genesis.json. To do so, open PowerShell and run the following command:

```
geth --datadir C:\Users\Administrator\EthereumPrivate init C:\Users\Administrator\EthereumPrivate\
```

**Note:** Don't run geth commands within PowerShell\_ISE but only within your standard PowerShell interface.

## Step 4: Initialize the Ethereum client (Cont.)



```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
commands.txt X static-nodes.json
23 <>
24 <>
25 ##### Start to Initialise the Ethereum Client #####
26
27 geth --datadir C:\Users\Administrator\EthereumPrivate init C:\Users\Administrator\EthereumPrivate\genesis.json
28
29 #####
30 <>
31 <>
32 <>
33 #####
34
35 geth -
36
37 #####
PS C:\Users\
PS C:\Users\
```

## Step 4: Initialize the Ethereum client (Cont.)

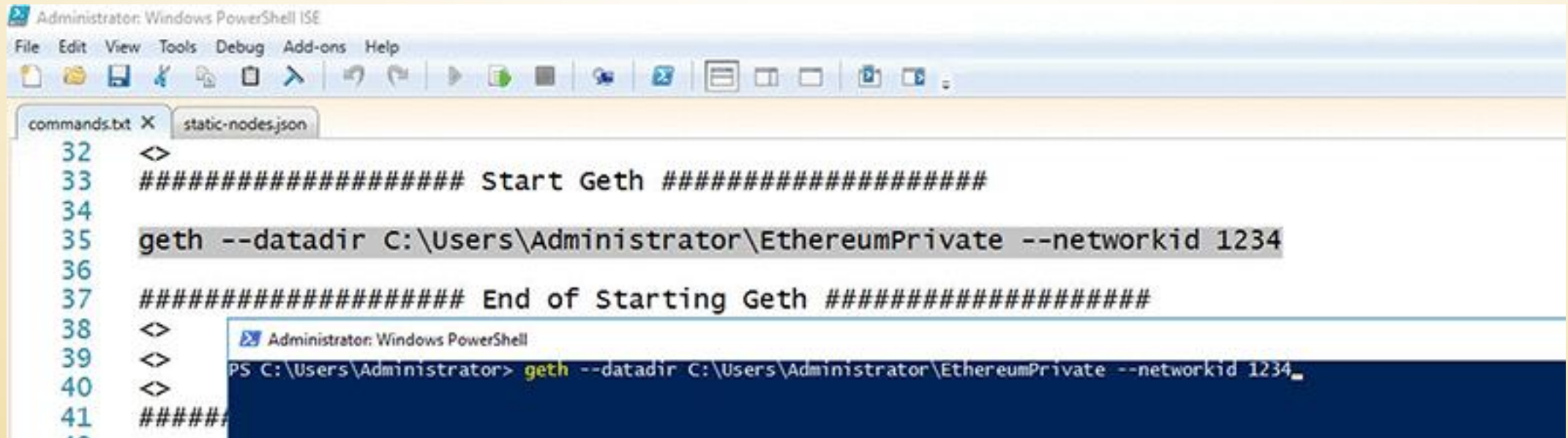
```
Administrator: Windows PowerShell
PS C:\Users\Administrator> geth --datadir C:\Users\Administrator\EthereumPrivate init C:\Users\Administrator\EthereumPrivate\genesis.json
INFO [08-09|20:40:47.875] Maximum peer count          ETH=25 LES=0 total=25
INFO [08-09|20:40:47.883] Allocated cache and file handles database=C:\\Users\\Administrator\\EthereumPrivate\\geth\\chaindata cache=
dies=16
INFO [08-09|20:40:47.916] Persisted trie from memory database nodes=6 size=951.00B time=0s gcnodes=0 gcsiz=0.00B gctime=0s livenodes=1
ze=0.00B
INFO [08-09|20:40:47.927] Successfully wrote genesis state database=chaindata hash=0
525ffb
INFO [08-09|20:40:47.936] Allocated cache and file handles database=C:\\Users\\Administrator\\EthereumPrivate\\geth\\lightchaindata c
6 handles=16
INFO [08-09|20:40:47.963] Persisted trie from memory database nodes=6 size=951.00B time=0s gcnodes=0 gcsiz=0.00B gctime=0s livenodes=1
ze=0.00B
INFO [08-09|20:40:47.974] Successfully wrote genesis state database=lightchaindata
1215_525ffb
PS C:\Users\Administrator> .
```

## Step 5: Start geth

Now your genesis block and peer nodes are configured. It's time to start the Ethereum client and configure it to communicate with your private network. You do so by running the following command.

```
geth --datadir C:\Users\Administrator\EthereumPrivate --networkid 1234
```

Note: Don't run geth commands within PowerShell\_ISE but only within your standard PowerShell interface.

A screenshot of the Windows PowerShell ISE (Integrated Scripting Environment) window. The title bar reads "Administrator: Windows PowerShell ISE". The menu bar includes "File", "Edit", "View", "Tools", "Debug", "Add-ons", and "Help". The toolbar contains various icons for file operations and execution. Two tabs are open: "commands.txt" and "static-nodes.json". The "commands.txt" tab is active, showing a script with line numbers 32 through 41. The script contains a section for starting Geth, with line 35 containing the command: `geth --datadir C:\Users\Administrator\EthereumPrivate --networkid 1234`. This line is highlighted. Below the script, a PowerShell console window is open, showing the prompt `PS C:\Users\Administrator>` followed by the command `geth --datadir C:\Users\Administrator\EthereumPrivate --networkid 1234_` being entered.



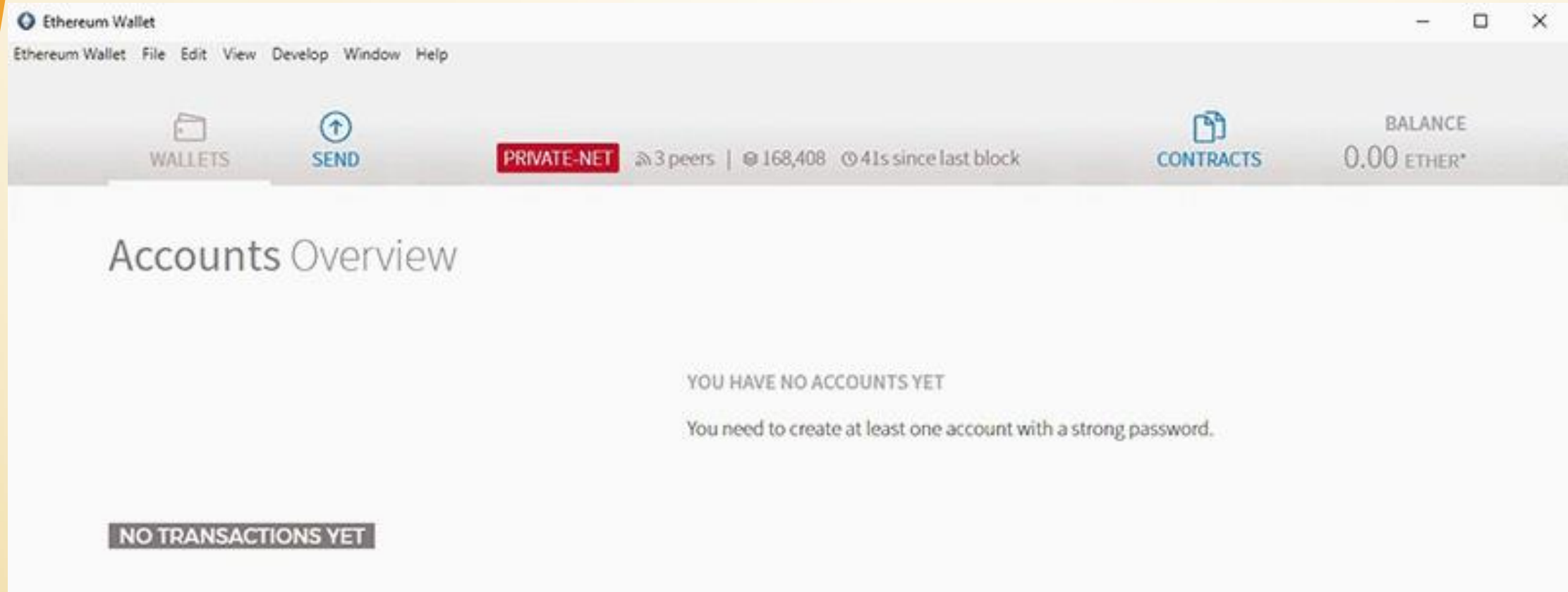
## Step 5: Start geth (Cont.)

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> geth --datadir C:\Users\Administrator\EthereumPrivate --networkid 1234
INFO [08-09|21:42:36.769] Maximum peer count
INFO [08-09|21:42:36.788] Starting peer-to-peer node
INFO [08-09|21:42:36.795] Allocated cache and file handles
INFO [08-09|21:42:36.833] Initialised chain configuration
<nil> Constantinople: <nil> Engine: unknown}
INFO [08-09|21:42:36.845] Disk storage enabled for ethash caches
INFO [08-09|21:42:36.853] Disk storage enabled for ethash DAGs
INFO [08-09|21:42:36.861] Initialising Ethereum protocol
INFO [08-09|21:42:36.867] Loaded most recent local header
INFO [08-09|21:42:36.874] Loaded most recent local full block
INFO [08-09|21:42:36.880] Loaded most recent local fast block
INFO [08-09|21:42:36.887] Regenerated local transaction journal
INFO [08-09|21:42:36.893] Starting P2P networking
INFO [08-09|21:42:39.018] UDP listener up
c4a86bb4be6397dfe21a69a34659560@[:]:30303
INFO [08-09|21:42:39.030] RLPx listener up
c4a86bb4be6397dfe21a69a34659560@[:]:30303
INFO [08-09|21:42:39.033] IPC endpoint opened
INFO [08-09|21:42:48.307] Block synchronisation started
INFO [08-09|21:42:48.315] Imported new state entries
INFO [08-09|21:42:49.279] Imported new block headers
INFO [08-09|21:42:49.301] Imported new block receipts
INFO [08-09|21:42:49.332] Imported new block headers
INFO [08-09|21:42:49.357] Imported new block receipts
INFO [08-09|21:42:49.462] Imported new block headers
INFO [08-09|21:42:49.590] Imported new block receipts
INFO [08-09|21:42:49.602] Imported new block headers
INFO [08-09|21:42:49.615] Imported new block receipts
INFO [08-09|21:42:49.628] Imported new state entries
INFO [08-09|21:42:49.638] Imported new block receipts
INFO [08-09|21:42:49.647] Committed new head block
INFO [08-09|21:42:49.667] Imported new chain segment
ETH=25 LES=0 total=25
instance=Geth/v1.8.13-stable-225171a4/windows-amd64/go1.10.3
database=C:\\Users\\Administrator\\EthereumPrivate\\geth\\chaindata cache=
config="{ChainID: 1234 Homestead: 0 DAO: <nil> DAOsupport: false EIP150:
dir=C:\\Users\\Administrator\\EthereumPrivate\\geth\\ethash count=3
dir=C:\\Users\\Administrator\\AppData\\Ethash count=2
versions="[63 62]" network=1234
number=0 hash=0b1215_525ffb td=1
number=0 hash=0b1215_525ffb td=1
number=0 hash=0b1215_525ffb td=1
transactions=0 accounts=0
self=enode://4751168b709c9ec3d34f504be701d69571de72e55452edb2fbb58977216b
self=enode://4751168b709c9ec3d34f504be701d69571de72e55452edb2fbb58977216b
url=\\\\.\\pipe\\geth.ipc
count=2 elapsed=929.1µs processed=2 pending=0 retry=0 duplicate=0 unexpect
count=192 elapsed=845.953ms number=192 hash=1f36b1_2b219a ignored=0
count=192 elapsed=7.997ms number=192 hash=1f36b1_2b219a size=768.00B ig
count=384 elapsed=40.013ms number=576 hash=1569f3_11d98b ignored=0
count=384 elapsed=4.000ms number=576 hash=1569f3_11d98b size=1.54kB ig
count=960 elapsed=108.983ms number=1536 hash=4e1562_ed047f ignored=0
count=960 elapsed=89ms number=1536 hash=4e1562_ed047f size=3.84kB ig
count=135 elapsed=124ms number=1671 hash=297fa6_f0649f ignored=0
count=69 elapsed=999.9µs number=1605 hash=3566bb_118451 size=276.00B ig
count=2 elapsed=0s processed=4 pending=0 retry=0 duplicate=0 une
count=1 elapsed=0s number=1606 hash=1bfca7_873bd4 size=4.00B ig
number=1606 hash=1bfca7_873bd4
blocks=65 txs=0 mgas=0.000 elapsed=14.006ms mgasps=0.000 number=1671 has
```

## Step 6: Interact with your network by using the Ethereum Wallet app

- To interact with your network, you start by opening the Ethereum Wallet app.
- A wallet is where you can store your keys and associated ether, contracts, and tokens. Your assets are not actually stored on your machine but rather on the blockchain.
- Your public key can have assets assigned to it. Your private key is used to sign transactions that enable you to create and interact with contracts and send assets to others.
- You should see an Ethereum Wallet icon on the desktop. Double-click this icon to open the app. Doing so connects to your private Ethereum network, and you see a screen like the following.

## Step 6: Interact with your network by using the Ethereum Wallet app

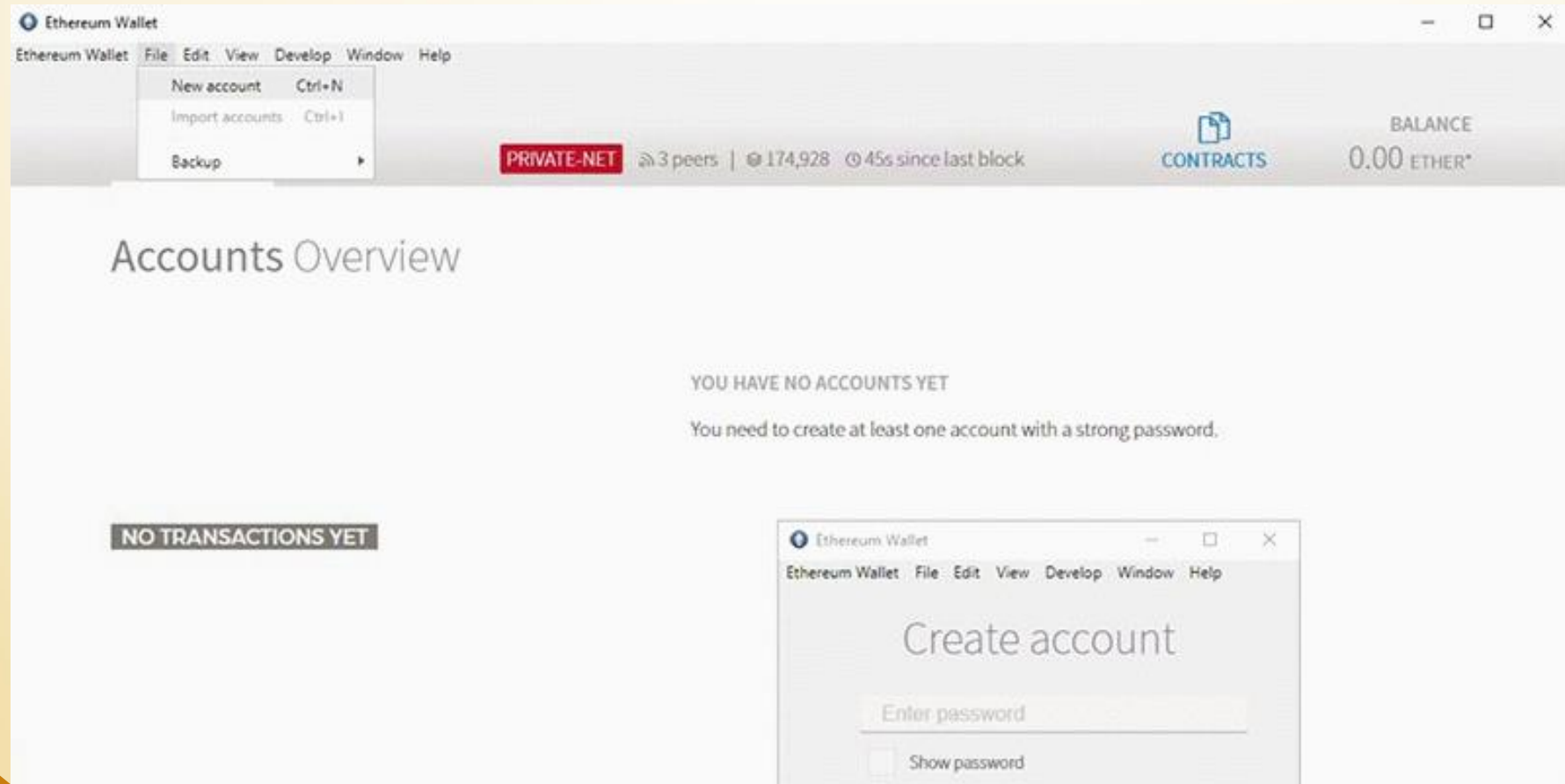


Private-Net indicates that you're using a nonpublic network ID such as 1234.



## Step 7: Add an account

- To start testing transactions and smart contracts, you add an account. Accounts are password-protected keys that can hold ether, secure ethereum-based tokens or coins, and control contracts.
- In the Ethereum Wallet, go to File, New Account, enter a password, and press Enter. Confirm your password, and press Enter. You have now created an account.



## Step 8: Start Mining

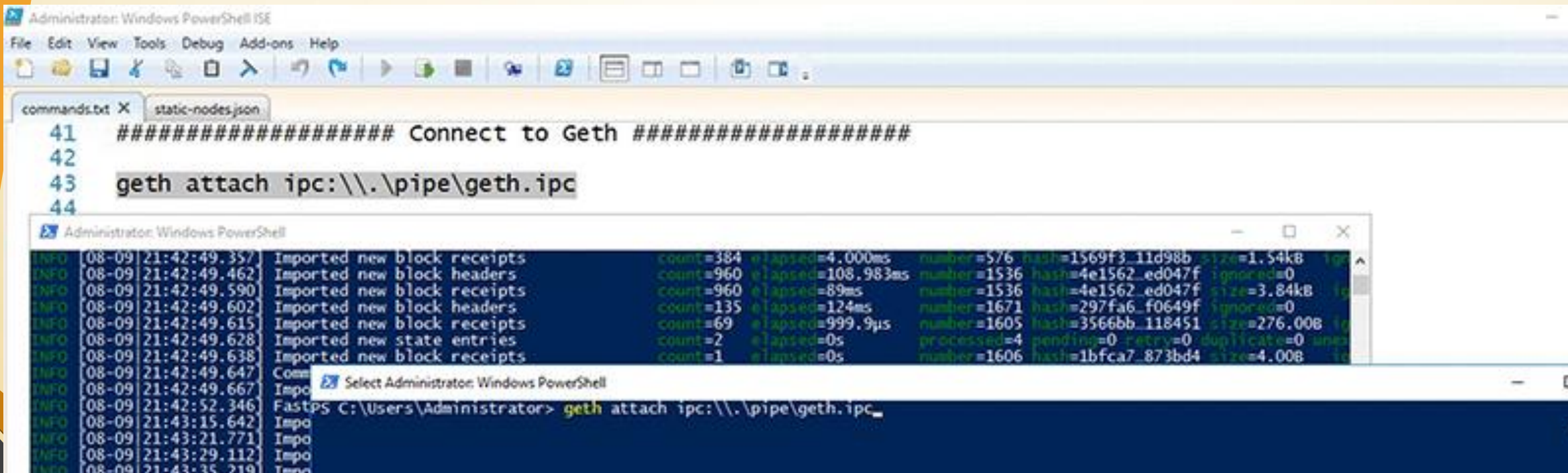
Mining is the process of confirming blocks to the network by solving a mathematical problem.

- With proof of work, the aim is to take the combination of the previous block hash, timestamp, and transactions, and a nonce value to produce a hash with a certain number of leading zeros.
- All members of the network have the first three variables, so it's essentially a race to find the nonce value that produces the desired result.
- The first node to find the answer broadcasts it to the network for confirmation. After the other nodes have verified the answer, the block is committed and ether is rewarded to the miner.
- Mining on the public Ethereum network is a complex task because it's only feasible using GPUs, requiring an OpenCL or CUDA enabled ethminer instance.
- In a private network setting, however, a single CPU miner instance is more than enough for practical purposes.
- A single instance can produce a stable stream of blocks at the correct intervals without needing heavy resources.

## Step 8: Start Mining (Cont.)

You need to start mining some ether so you have funds available for transactions such as creating smart contracts.

To do this, you need to open another PowerShell window and attach to the geth instance with `geth attach ipc:\\.\\pipe\\geth.ipc` so you can run commands.



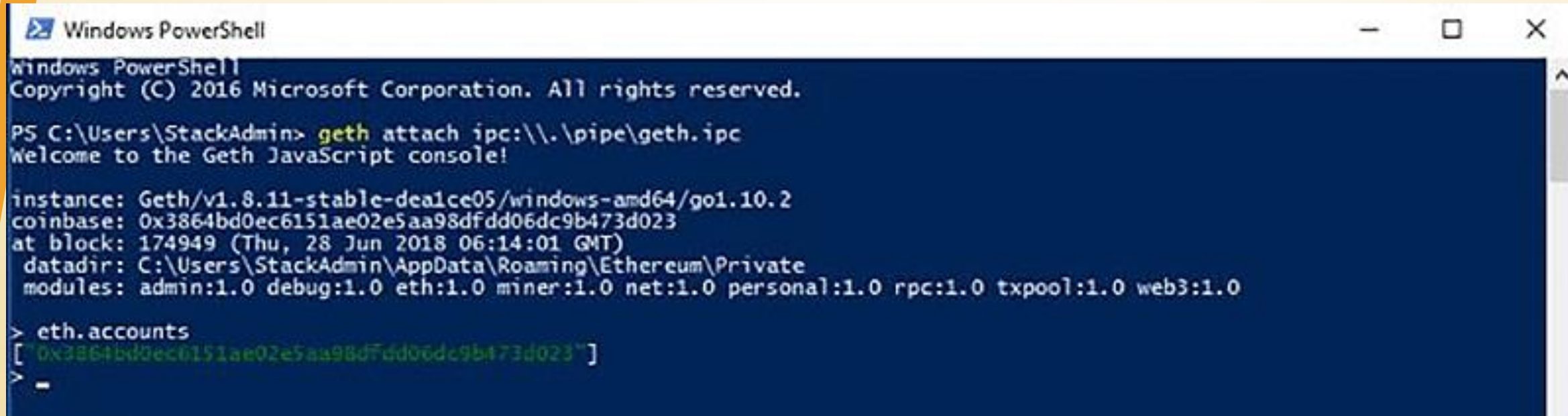
```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
commands.txt X static-nodes.json
41 ##### Connect to Geth #####
42
43 geth attach ipc:\\.\\pipe\\geth.ipc
44

Administrator: Windows PowerShell
INFO 08-09 21:42:49.357 Imported new block receipts count=384 elapsed=4.000ms number=576 hash=1569f3_11d98b size=1.54kB ig
INFO 08-09 21:42:49.462 Imported new block headers count=960 elapsed=108.983ms number=1536 hash=4e1562_ed047f ignored=0
INFO 08-09 21:42:49.590 Imported new block receipts count=960 elapsed=89ms number=1536 hash=4e1562_ed047f size=3.84kB ig
INFO 08-09 21:42:49.602 Imported new block headers count=135 elapsed=124ms number=1671 hash=297fa6_f0649f ignored=0
INFO 08-09 21:42:49.615 Imported new block receipts count=69 elapsed=999.9µs number=1605 hash=3566bb_118451 size=276.008 ig
INFO 08-09 21:42:49.628 Imported new state entries count=2 elapsed=0s processed=4 pending=0 retry=0 duplicate=0 unes
INFO 08-09 21:42:49.638 Imported new block receipts count=1 elapsed=0s number=1606 hash=1bfca7_873bd4 size=4.008 ig
INFO 08-09 21:42:49.647 Com
INFO 08-09 21:42:49.667 Impo
INFO 08-09 21:42:52.346 FastPS C:\Users\Administrator> geth attach ipc:\\.\\pipe\\geth.ipc_
INFO 08-09 21:43:15.642 Impo
INFO 08-09 21:43:21.771 Impo
INFO 08-09 21:43:29.112 Impo
INFO 08-09 21:43:35.219 Impo
```

## Step 9: See the accounts that you have registered

Type `eth.accounts` to see the accounts that you have registered.

You should see the same account you just created in Ethereum Wallet.

A screenshot of a Windows PowerShell window with a dark blue background. The title bar reads "Windows PowerShell". The text inside shows the Geth JavaScript console output after running `geth attach ipc:\\.pipe\geth.ipc`. It displays the Geth version, coinbase address, current block, datadir, and available modules. Finally, the `eth.accounts` command is executed, showing a single account address in green text.

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\StackAdmin> geth attach ipc:\\.pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.11-stable-dea1ce05/windows-amd64/go1.10.2
coinbase: 0x3864bd0ec6151ae02e5aa98dfdd06dc9b473d023
at block: 174949 (Thu, 28 Jun 2018 06:14:01 GMT)
datadir: C:\Users\StackAdmin\AppData\Roaming\Ethereum\Private
modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> eth.accounts
[
  "0x3864bd0ec6151ae02e5aa98dfdd06dc9b473d023"
]
```

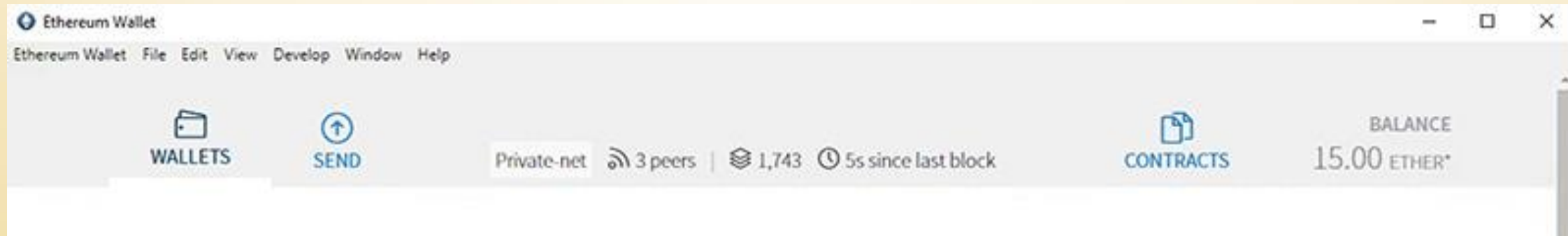


## Step 10: Start mining with two threads

Type `miner.start(2)` to start mining with two threads.

```
at block: 1733 (Thu, 09 Aug 2018 21:51:13 AEST)
datadir: C:\Users\Administrator\EthereumPrivate
modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0
> miner.start(2)_
```

You should now see your account within the Ethereum wallet start accumulating ether from the mining activities.



## Step 11: Create, deploy, and test your smart contract

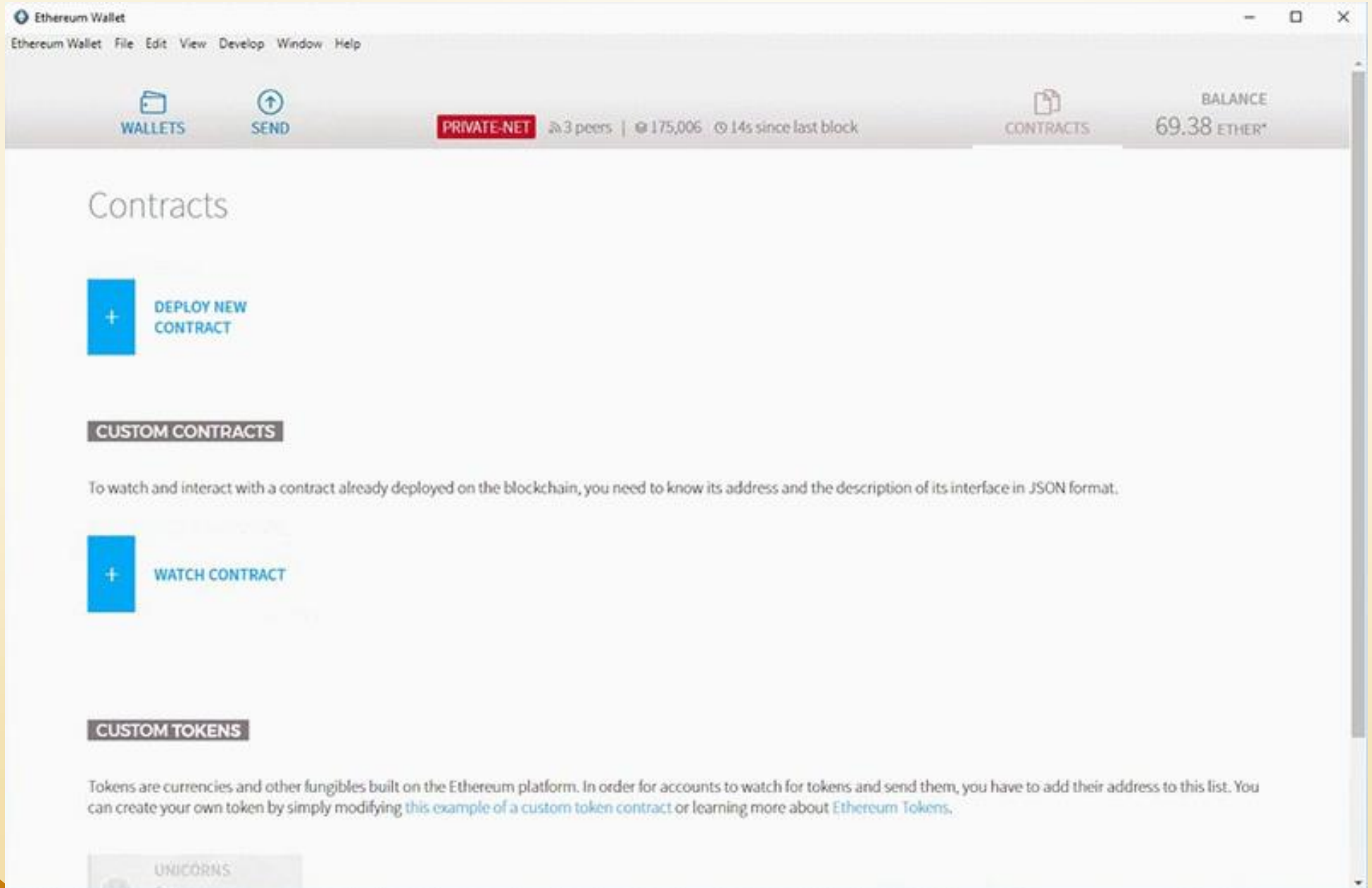
As a next step, create a smart contract.

Tokens in the Ethereum ecosystem can represent any fungible tradable good: coins, loyalty points, gold certificates, IOUs, in-game items, and so on. All tokens implement some basic features in a standard way. Because of this, your token is instantly compatible with the Ethereum wallet and any other client or contract that uses the same standards.

We're now going to build a basic contract that deploys a token or coin. For a bit of fun, I'm going to create my own basic currency called MattCoin. Feel free to modify the name from MattCoin to whatever makes sense for you. You can have a bit of fun and tell your friends you now have your own cryptocurrency too!

To deploy this contract, click **CONTRACTS** in the Ethereum Wallet, then select **DEPLOY NEW CONTRACT**.

## Step 11: Create, deploy, and test your smart contract (Cont.)



## Step 11: Create, deploy, and test your smart contract (Cont.)

Following is the code for the contract we're going to deploy to build MattCoin. Paste this code into the SOLIDITY CONTRACT SOURCE CODE section. Next, select the contract to deploy and also the initial number of tokens to create. Then, click DEPLOY and enter the password you created with the account.

This code creates a smart contract for a simple coin or token. First, it creates a mapping of all possible Ethereum addresses and gives them a balance. The variables allow any address to hold a value of MattCoin. Next, in the constructor we specify that the entire initial supply should go to the creator of the contract. Then finally, we define a function called **transfer**, which allows us to send MattCoin from one Ethereum address to another.



## Step 11: Create, deploy, and test your smart contract (Cont.)

```
pragma solidity ^0.4.18;
contract MattCoin {
    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;

    /* Initializes contract with initial supply tokens to the creator of the contract */
    constructor(
        uint256 initialSupply
    ) public {
        balanceOf[msg.sender] = initialSupply;          // Give the creator all initial tokens
    }

    /* Send coins */
    function transfer(address _to, uint256 _value) public {
        require(balanceOf[msg.sender] >= _value);        // Check if the sender has enough
        require(balanceOf[_to] + _value >= balanceOf[_to]); // Check for overflows
        balanceOf[msg.sender] -= _value;                  // Subtract from the sender
        balanceOf[_to] += _value;                          // Add the same to the recipient
    }
}
```

## Step 11: Create, deploy, and test your smart contract (Cont.)

The screenshot displays the Ethereum Wallet interface for a Private-Net. The top bar shows the wallet name, menu options, network status (3 peers, 168,931 nodes, 3s since last block), and the current balance of 472.66 ETH.

**Transaction Details:**

- FROM:** Account 1 - 472.66 ETH
- AMOUNT:** 0.0 ETH
- Send everything:** ☐
- You want to send:** 0 ETH

**Smart Contract Deployment Section:**

- SOLIDITY CONTRACT SOURCE CODE:**

```
1 pragma solidity ^0.4.18;
2 contract MattCoin {
3     /* This creates an array with all balances */
4     mapping (address => uint256) public balanceOf;
5
6     /* Initializes contract with initial supply tokens to the creator of the contract */
7     function MattCoin(
8         uint256 initialSupply
9     ) public {
10         balanceOf[msg.sender] = initialSupply; // Give the creator all initial tokens
11     }
12
13     /* Send coins */
14     function transfer(address _to, uint256 _value) public {
15         require(balanceOf[msg.sender] >= _value); // Check if the sender has enough
16         require(balanceOf[_to] + _value >= balanceOf[_to]); // Check for overflows
17         balanceOf[msg.sender] -= _value; // Subtract from the sender
18         balanceOf[_to] += _value; // Add the same to the recipient
19     }
20 }
```
- CONTRACT BYTE CODE:** (Empty)
- SELECT CONTRACT TO DEPLOY:** Matt Coin
- CONSTRUCTOR PARAMETERS:** Initial supply - 256 bits unsigned integer: 1000000000

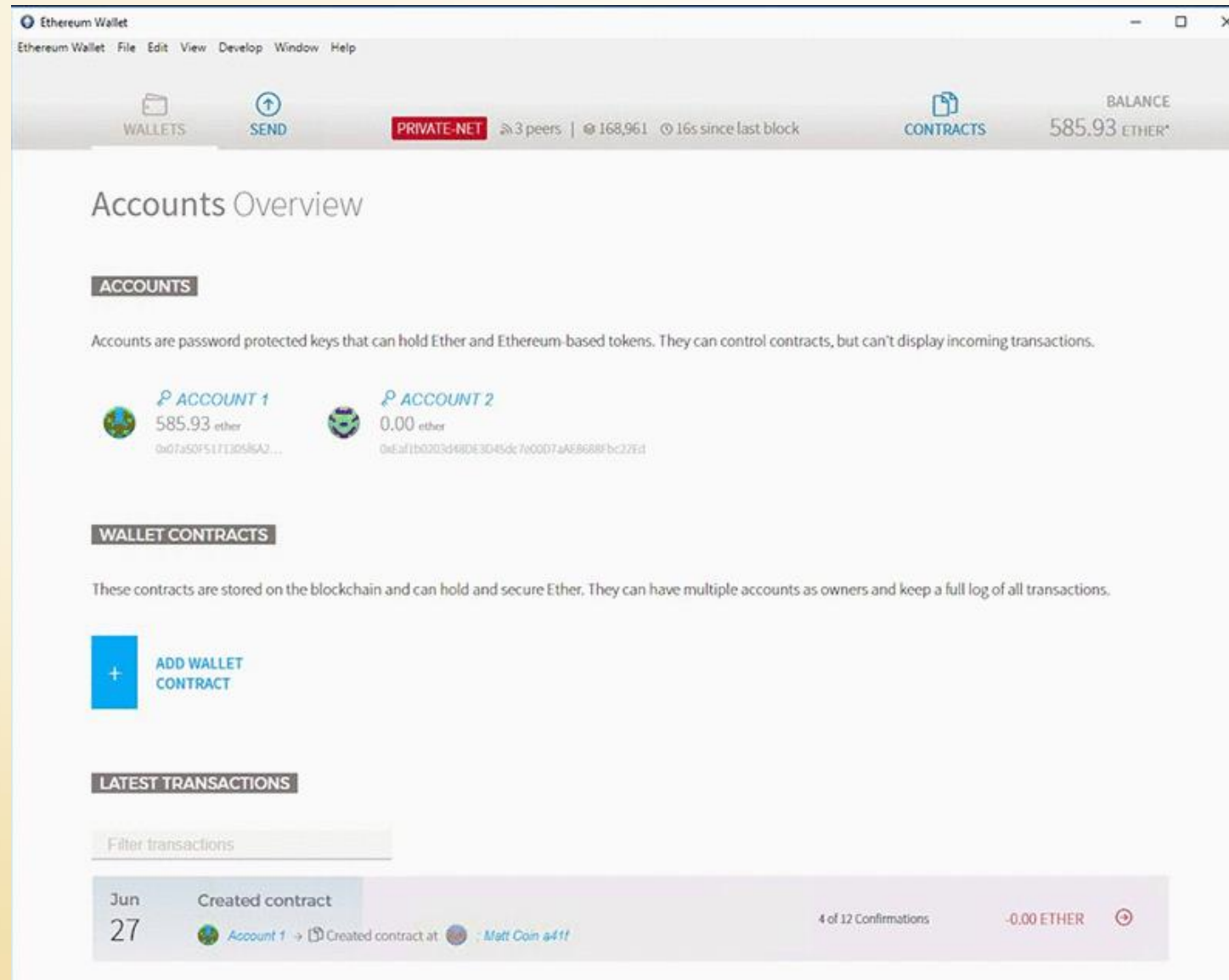
**Transaction Fee and Total:**

- SELECT FEE:** 0.003383928 ETH
- CHEAPER / FASTER:** Slider set towards FASTER.
- TOTAL:** 0.003383928 ETH

**Disclaimer:** This is the most amount of money that might be used to process this transaction. Your transaction will be mined **probably within 30 seconds**.

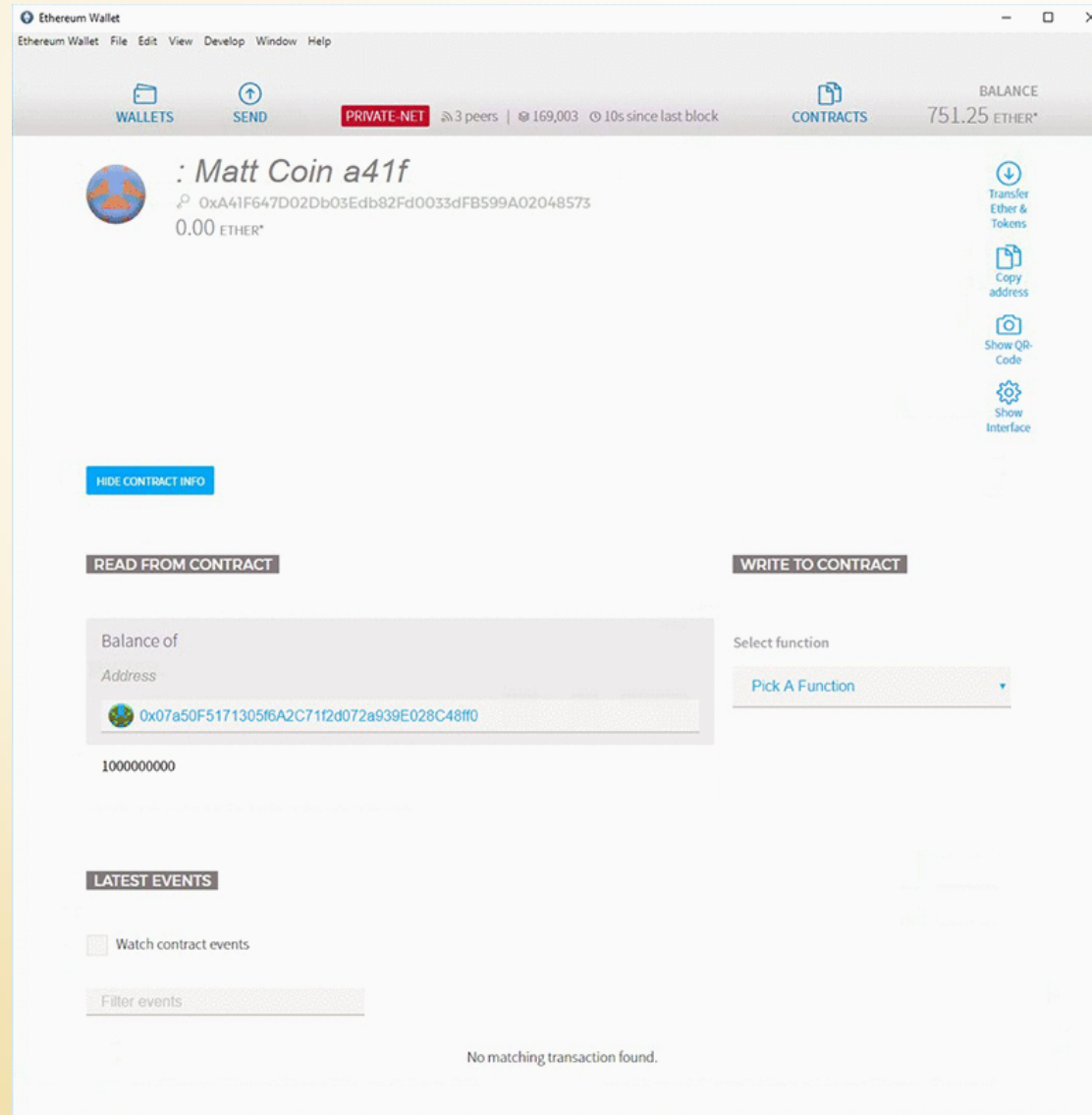
## Step 11: Create, deploy, and test your smart contract (Cont.)

You now see the contract being created. The process finishes after it's been mined into the next block.



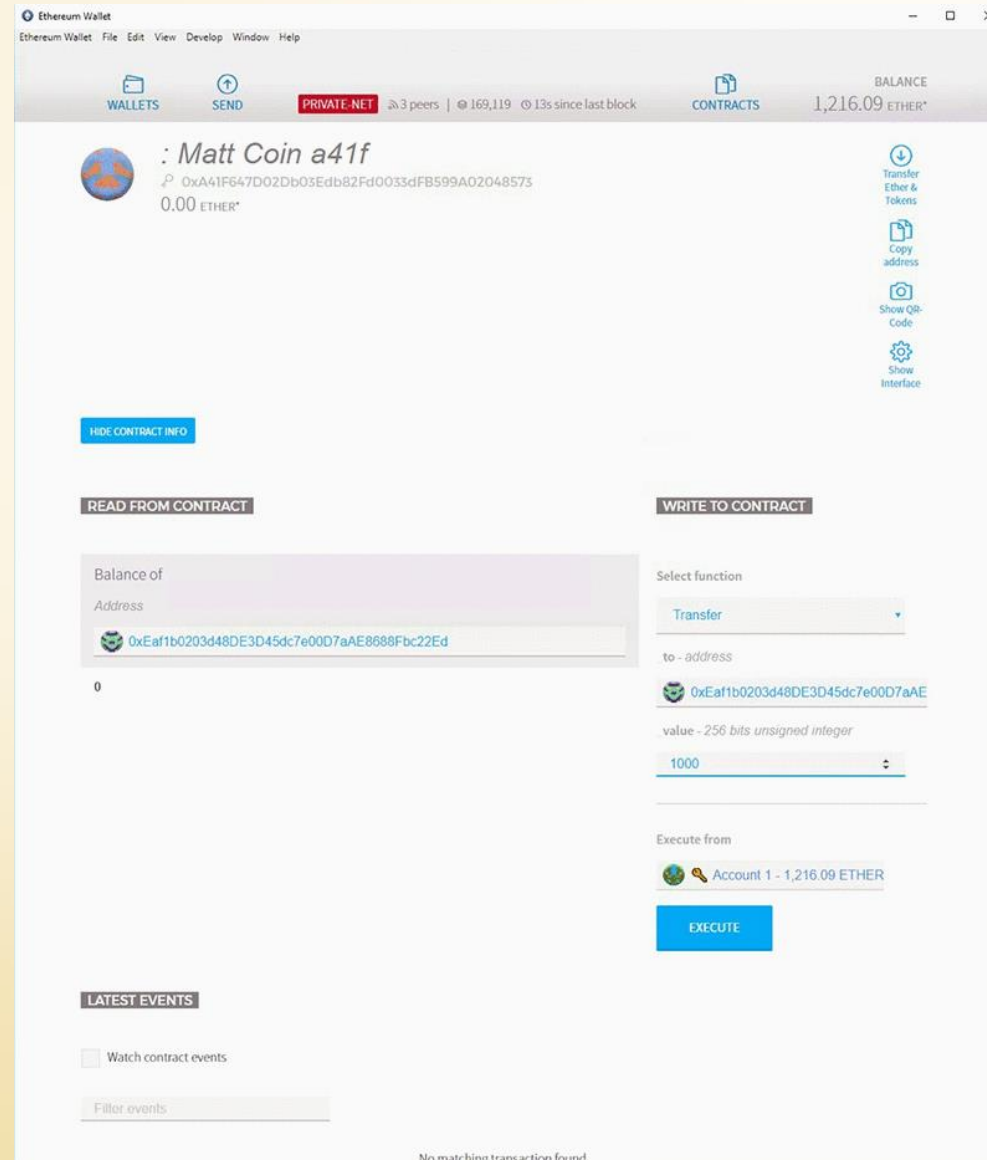
## Step 11: Create, deploy, and test your smart contract (Cont.)

After the contract has been created, click it and enter your account address into the Balance of field. You should then see a balance equal to the amount you created with the contract.



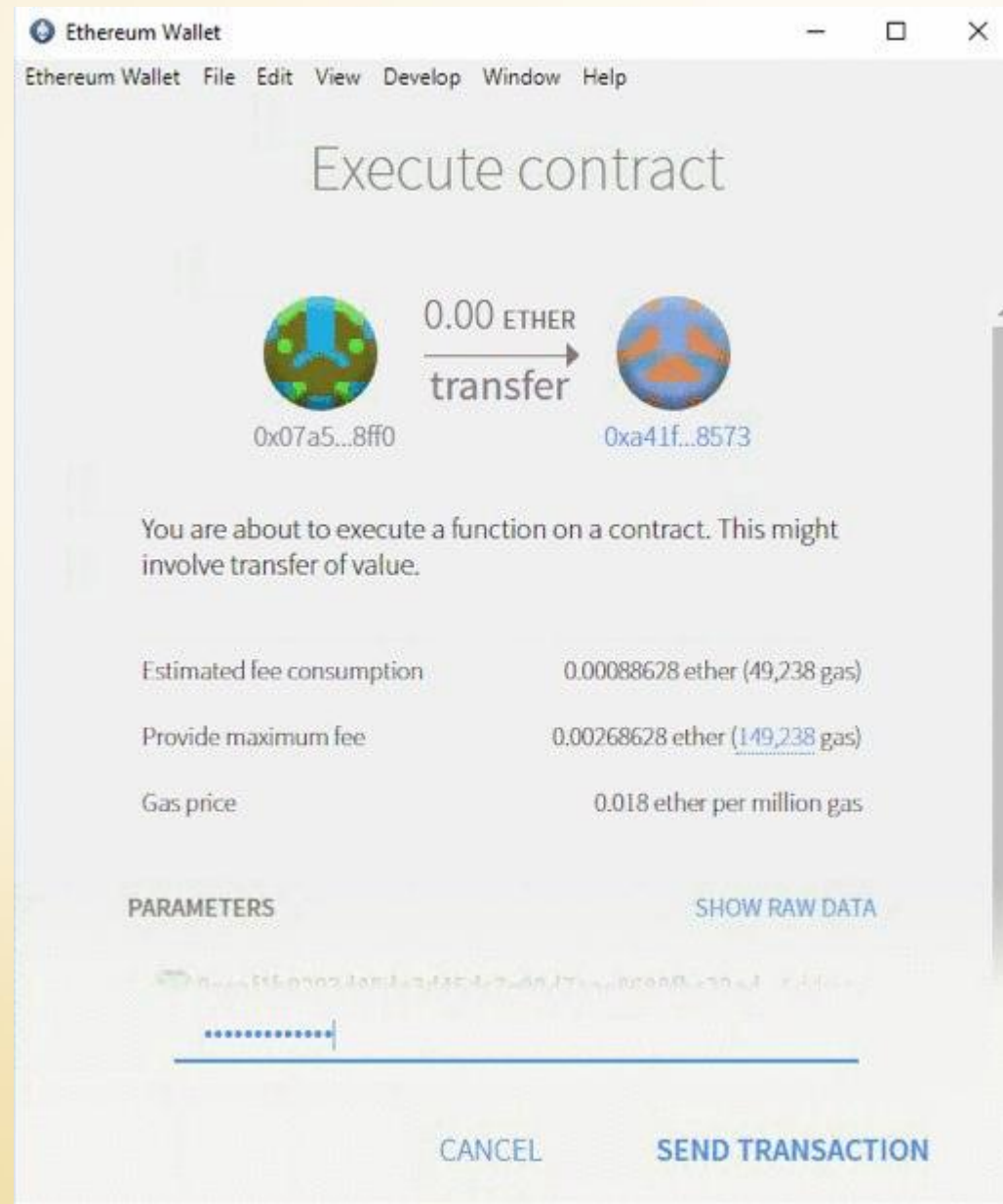
## Step 11: Create, deploy, and test your smart contract (Cont.)

If you want to test sending your tokens to another Ethereum account, first create a second account. Then select the transfer function, provide a to address and the number of tokens to transfer, and click EXECUTE.



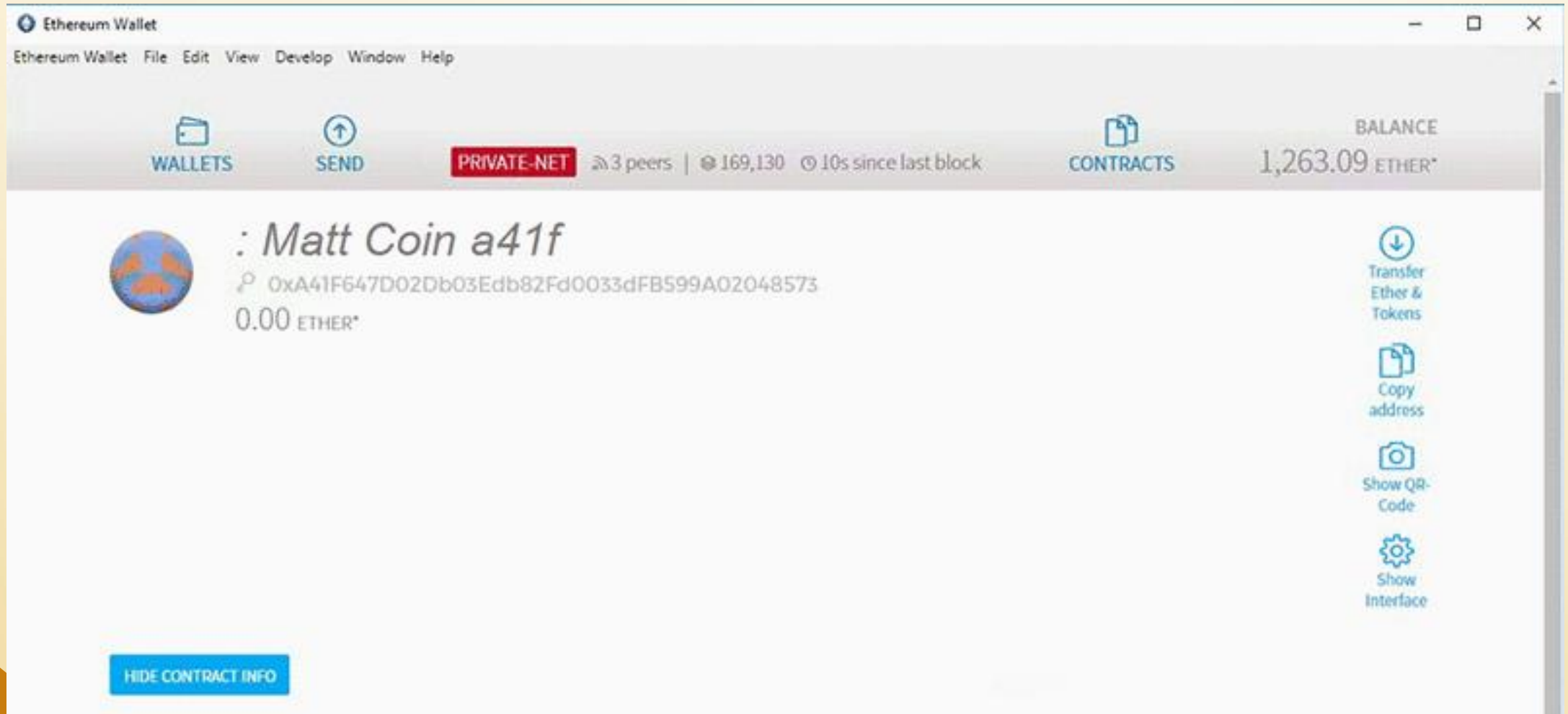
## Step 11: Create, deploy, and test your smart contract (Cont.)

You are again asked to enter your password to sign the transaction.



## Step 11: Create, deploy, and test your smart contract (Cont.)

After the transaction is sent, enter the to address value into the Balance of field and you should see the amount of value you sent to this address.





## Step 11: Create, deploy, and test your smart contract (Cont.)

After the transaction is sent, enter the to address value into the Balance of field and you should see the amount of value you sent to this address.

### READ FROM CONTRACT

Balance of

Address



0xEaf1b0203d48DE3D45dc7e00D7aAE8688Fbc22Ed

1000

### WRITE TO CONTRACT

Select function

Transfer

to - address



0xEaf1b0203d48DE3D45dc7e00D7aAE

value - 256 bits unsigned integer

1000

Execute from



Account 1 - 1,261.09 ETH

EXECUTE



# **Scenario:** Deploy smart contracts to your private Ethereum blockchain network on AWS

## **Conclusion**

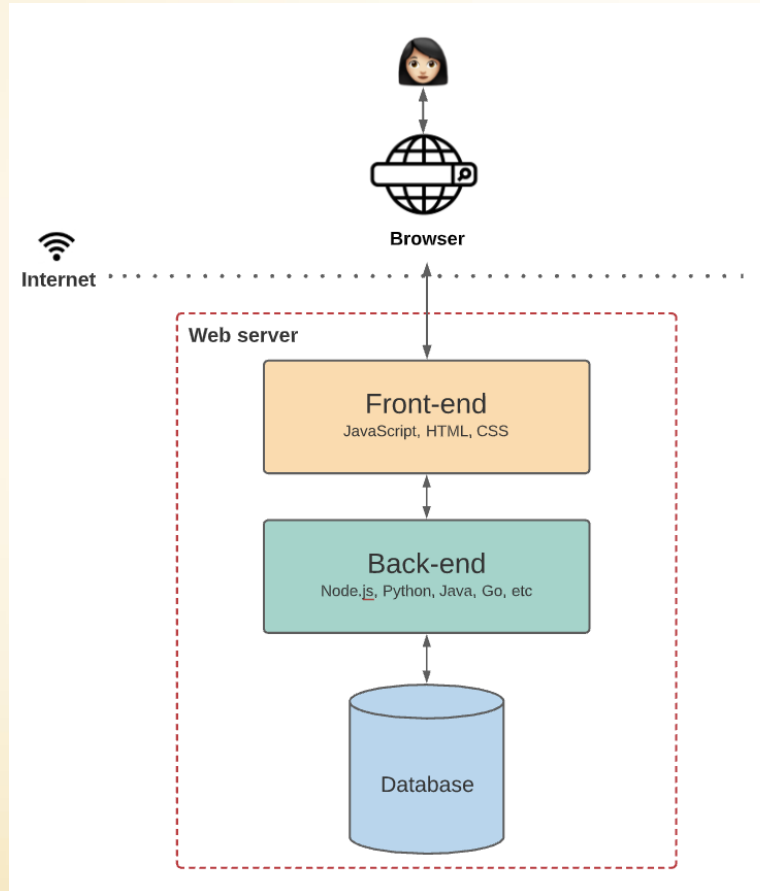
You've now set up your own private Ethereum network and deployed your first smart contract. By using the AWS Blockchain Template for Ethereum, it was easy to get started with your blockchain project and start testing use cases.

# Ethereum Blockchain Web Application

**Scenario:** Ethereum Blockchain using the  
Architecture of a Web 3.0 Application

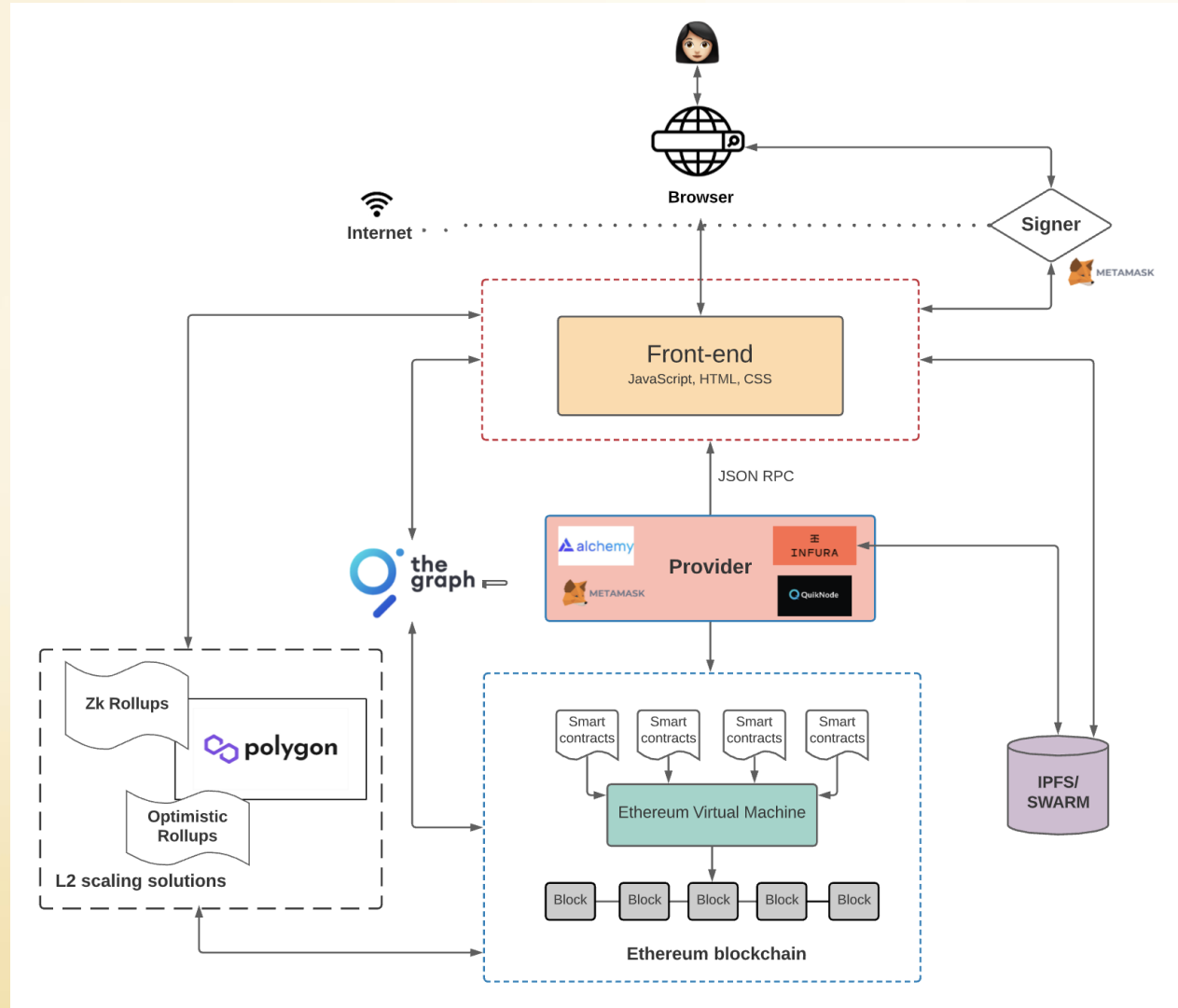
# Use-Case

## Building Block Chain Using Web 3.0 Application



# Use-Case

## Building Block Chain Using Web 3.0 Application



# Smart contracts

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.7.0 <0.9.0;
4
5 /**
6  * @title Storage
7  * @dev Store & retrieve value in a variable
8  */
9 contract Storage {
10
11     uint256 number;
12     struct Jon{
13         uint[] arr;
14     }
15
16     Jon private x;
17
18     function foo() public {
19         x.arr.push(5);
20     }
21
22     /**
23      * @dev Store value in variable
24      * @param num value to store
25      */
26     function store(uint256 num) public {
27         number = num;
28     }
29
30     /**
31      * @dev Return value
32      * @return value of 'number'
33      */
34     function retrieve() public view returns (uint256){
35         return number;
36     }
37 }
```

# Use-Case

## Building Block Chain Using Web 3.0 Application

For use-case solution please refer/click the below URL:  
Click [Here](#)

Thank You