Amazon Managed Blockchain

Management Guide

Amazon Managed Blockchain: Management Guide

Table of Contents

What Is Amazon Managed Blockchain?

Amazon Managed Blockchain is a fully managed service for creating and managing blockchain networks using open source frameworks. Currently, the Hyperledger Fabric open source framework is supported. Blockchain allows you to build applications where multiple parties can securely and transparently run transactions and share data without the need for a trusted, central authority.

You can use Managed Blockchain to create a scalable blockchain network quickly and efficiently using the AWS Management Console, the AWS CLI, or the Managed Blockchain SDK. Managed Blockchain scales to meet the demands of thousands of applications running millions of transactions. After the blockchain network is up and running, Managed Blockchain also simplifies network management tasks. Managed Blockchain manages your certificates, lets you easily create proposals for a vote among network members, and tracks operational metrics such as compute, memory, and storage resources.

This guide covers the fundamentals of creating and working with resources in a Managed Blockchain network.

How to Get Started with Managed Blockchain

We recommend the following resources to get started developing blockchain applications using Managed Blockchain:

- Key Concepts: Managed Blockchain Networks, Members, and Peer Nodes (p. 2)

  This overview helps you understand the fundamental building blocks of a Managed Blockchain network. It also tells you how to identify and communicate with resources, regardless of the blockchain framework that you're using.
- Get Started Creating a Hyperledger Fabric Blockchain Network Using Amazon Managed Blockchain (p. 6)

  This tutorial lets you try Managed Blockchain and get a Hyperledger Fabric blockchain application running in a short time. You create your first network, set up a Hyperledger Fabric client, and use the open source Hyperledger Fabric peer CLI to query and update the ledger. You then invite another member to the network. The member can be from a different AWS account, or you can invite a new member in your own account to simulate a multi-account network. The new member then queries and updates the ledger.
- Hyperledger Fabric Documentation (v1.2)

  The open source documentation for Hyperledger Fabric is a starting point for key concepts and the architecture of the Hyperledger Fabric blockchain network that you build using Managed Blockchain. As you develop your blockchain application, you can reference this document for key tasks and code samples. Use the documentation version that corresponds to the version of Hyperledger Fabric that you use.

Key Concepts: Managed Blockchain Networks, Members, and Peer Nodes

A blockchain network is a peer-to-peer network running a decentralized blockchain framework. A network includes one or more *members*, which are unique identities in the network. For example, a member might be an organization in a consortium of banks. Each member runs one or more blockchain *peer nodes* to run chaincode, endorse transactions, and store a local copy of ledger.

Amazon Managed Blockchain creates and manages these components for each member in a network, and it also creates components shared by all members in a network, such as the Hyperledger Fabric ordering service and the general networking configuration.

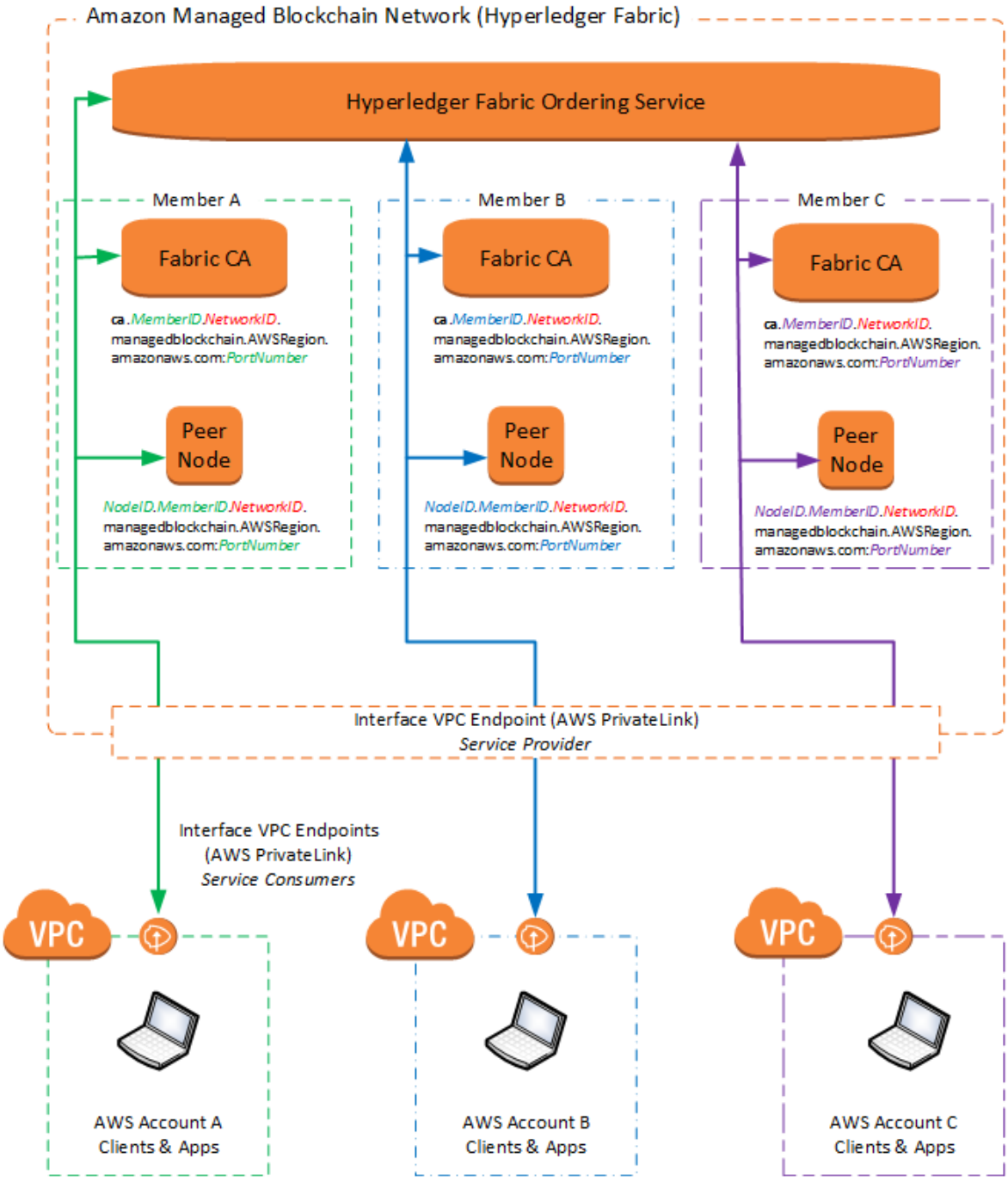
Managed Blockchain Networks and Editions

When creating a Managed Blockchain network, the creator chooses the blockchain framework and the *edition* of Amazon Managed Blockchain to use. The edition determines the capacity and capabilities of the network as a whole.

The creator also must create the first Managed Blockchain network member. Additional members are added through a proposal and voting process. There is no charge for the network itself, but each member pays an hourly rate (billed per second) for their network membership. Charges vary depending on the edition of the network. Each member also pays for peer nodes, peer node storage, and the amount of data that the member writes to the network. For more information about available editions and their attributes, see Managed Blockchain Pricing. For more information about the number of networks that each AWS account can create and join, see Managed Blockchain Limits in the *AWS General Reference*.

The blockchain network remains active as long as there are members. The network is deleted only when the last member deletes itself from the network. No member or AWS account, even the creator's AWS account, can delete the network until they are the last member and delete themselves.

The following diagram shows the basic components of a Hyperledger Fabric blockchain running on Managed Blockchain.



Inviting and Removing Members

A Managed Blockchain network is decentralized. An AWS account initially creates a Managed Blockchain network, but the network is not owned by that AWS account, or any other AWS account. To make changes to the network, members make *proposals* that all other members in the network vote on. For another AWS account to join the network, for example, an existing member creates a proposal to invite the account. Other members than vote Yes or No on the proposal. If the proposal is approved, an

invitation is sent to the AWS account. The account then accepts the invitation and creates a member to join the network. A similar proposal process is required to remove a member in a different AWS account. A principal in an AWS account with sufficient permissions can remove a member that the account owns at any time by deleting that member directly, without submitting a proposal.

The network creator also defines a *voting policy* for the network when they create it. The voting policy determines the basic rules for all proposal voting on the network. The voting policy includes the percentage of votes required to pass the proposal, and the duration before the vote expires.

Note
Different frameworks use slightly different terms for the identities that we call *members* in Managed Blockchain. For example, Hyperledger Fabric uses the term *organizations*.

Peer Nodes

When a member joins the network, one of the first things they must do is create at least one *peer node* in the membership.

Blockchain networks contain a distributed, cryptographically secure ledger that maintains the history of transactions in the network that is immutable—it can't be changed after-the fact. Each peer node stores a local copy of the ledger. Each peer node also holds the global state of the network for the channels in which they participate, which gets updated with each new transaction. The peer nodes also interact to create and endorse the transactions that are proposed on the network. Members define the rules in the endorsement process based on their business logic and the blockchain framework being used. In this way, every member can independently verify the transaction history without a centralized authority

To configure blockchain applications on peer nodes and to interact with other network resources, members use a client configured with open source tools such as a CLI or SDK. The applications and tools that you choose and your client setup depend on the blockchain framework that you use and your preferred development environment. For example, in the Getting Started (p. 6) tutorial, you configure an Amazon EC2 instance in a VPC with open source Hyperledger Fabric CLI tools. Regardless of the framework, the way that you identify and connect to Managed Blockchain resources using framework tools is the same.

Identifying Managed Blockchain Resources and Connecting from a Client

Because the blockchain network is decentralized, members must interact with each other's peer nodes and network-wide resources to make transactions, endorse transactions, verify members, and so on. When a network is created, Managed Blockchain gives the network a unique ID. Similarly, when an AWS account creates a member on the network and peer nodes, Managed Blockchain gives unique IDs to those resources.

Each network resource has a unique, addressable endpoint that Managed Blockchain creates from these IDs. Other members in the Managed Blockchain network, blockchain applications, and tools use these endpoints to identify and interact with resources on the Managed Blockchain network.

Resource endpoints on the Managed Blockchain network are in the following format:

```
ResourceID.MemberID.NetworkID.managedblockchain.AWSRegion.amazonaws.com:PortNumber
```

For example, to refer to a peer node with ID nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y, owned by a member with ID m-K46ICRRXJRCGRNNS4ES4XUUS5A, in a Hyperledger Fabric network with ID n-MWY63ZJZU5HGNCMBQER7IN6OIU, you use the following peer node endpoint:

```
nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y.m-K46ICRRXJRCGRNNS4ES4XUUS5A.n-
MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-east-1.amazonaws.com:30003
```

The port that you use with an endpoint depends on the blockchain framework, and the blockchain service that you are calling. Managed Blockchain supports only `us-east-1` for *AWSRegion*.

Within the blockchain network, access and authorization for each resource is governed by processes defined within the network. Outside the confines of the network—that is, from member's client applications and tools—Managed Blockchain uses AWS PrivateLink to ensure that only network members can access required resources. In this way, each member has a private connection from a client in their VPC to the Managed Blockchain network. The interface VPC endpoint uses private DNS, so you must have a VPC in your account that is enabled for Private DNS. For more information, see Create an Interface VPC Endpoint to Connect to Managed Blockchain Network Resources (p. 45).

Get Started Creating a Hyperledger Fabric Blockchain Network Using Amazon Managed Blockchain

This tutorial guides you through creating your first Hyperledger Fabric network using Amazon Managed Blockchain. It shows you how to set up the network and create a member in your AWS account, set up chaincode and a channel, and then invite members from other AWS accounts to join a channel. Instructions for invitees is also provided.

Steps

Prerequisites and Considerations

To complete this tutorial, you must have the resources listed in this section. Unless specifically stated otherwise, the requirements apply to both network creators and invited members.

Topics

An AWS account

Before you use Managed Blockchain for the first time, you must sign up for an Amazon Web Services (AWS) account.

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

A Linux Client (EC2 Instance)

You must have a Linux computer with access to resources in the VPC to serve as your Hyperledger Fabric client. This computer must have a version of the AWS CLI installed that includes the `managedblockchain` command. We recommend creating an Amazon Elastic Compute Cloud (Amazon EC2) instance in the same VPC and AWS region as the VPC endpoint for the Managed Blockchain network. This is the setup that the tutorial uses. For instructions to set up a Hyperledger Fabric client using this configuration, see Step 3: Create an Amazon EC2 Instance and Set Up the Hyperledger Fabric Client (p. 11).

An AWS CloudFormation template to create a Hyperledger Fabric client is available in amazon-managed-blockchain-client-templates repository on Github. For more information, see the readme.md in that repository. For more information about using AWS CloudFormation, see Getting Started in the *AWS CloudFormation User Guide*.

A VPC

You must have a VPC with an IPv4 CIDR block, and the `enableDnsHostnames` and `enableDnsSupport` options must be set to `true`. If you will connect to the Hyperledger Fabric client using SSH, the VPC must have an internet gateway, and the security group configuration associated with the Hyperledger Framework client must allow inbound SSH access from your SSH client.

- For more information about creating a suitable network, see Getting Started with IPv4 for Amazon VPC tutorial in the *Amazon VPC User Guide*.
- For information about using SSH to connect to an Amazon EC2 Instance, see Connecting to Your Linux Instance Using SSH in the *Amazon EC2 User Guide for Linux Instances*.
- For instructions about how to verify if DNS options are enabled, see Using DNS with Your VPC in the *Amazon VPC User Guide*.

Permissions to Create an Interface VPC Endpoint

The IAM principal (user) identity that you are using must have sufficient IAM permissions to create an interface VPC endpoint in your AWS account. For more information, see Controlling Access - Creating and Managing VPC Endpoints in the *Amazon VPC User Guide*.

EC2 Security Groups That Allow Communication on Required Ports

The EC2 security groups associated with the Hyperledger Fabric client Amazon EC2 instance and the Interface VPC Endpoint that you create during this tutorial must have rules that allow traffic between them for required Hyperledger Fabric services. EC2 security groups are restrictive by default, so you need to create security group rules that allow required access. In addition, a security group associated with the Hyperledger Fabric client Amazon EC2 instance must have an inbound rule that allows SSH traffic (Port 22) from trusted SSH clients.

For the purposes of simplicity in this tutorial, we recommend that you create an EC2 security group that you associate only with the Hyperledger Fabric client Amazon EC2 instance and the Interface VPC Endpoint. Then create an inbound rule that allows all traffic from within the security group. In addition,

create another security group to associate with the Hyperledger Fabric client Amazon EC2 instance that allows inbound SSH traffic from trusted clients.

> **Important**
> This security group configuration is recommended for this tutorial only. Carefully consider security group settings for your desired security posture. For information about the minimum required rules, see Configuring Security Groups (p. 67).

To create a security group that allows traffic between the Hyperledger Fabric client and the interface VPC endpoint for use in this tutorial

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Choose **Security groups** in the navigation pane, and then choose **Create security group**.
3. Enter a **Security group name** and **Description** for the security group that helps you find it. For example, **`HFClientAndEndpoint`**.
4. Make sure that the VPC you select is the default VPC for your account. This is the VPC in which Hyperledger Fabric network resources and the interface VPC endpoint are created.
5. Choose **Create**.
6. Select the security group that you just created from the list, choose **Inbound**, and then choose **Edit**.
7. Under **Type**, select **All traffic** from the list.
8. Under **Source**, leave **Custom** selected, and then begin typing the name or ID of this same security group—for example, **HFClientAndEndpoint**—and then select the security group so that its ID appears under **Source**.
9. Choose **Save**.

   You reference this security group later in this tutorial in Step 2: Create and Configure the Interface VPC Endpoint (p. 11) and Step 3: Create an Amazon EC2 Instance and Set Up the Hyperledger Fabric Client (p. 11).

To create a security group for the Hyperledger Fabric client that allows inbound SSH connections from the computer that you are working with

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Choose **Security groups** in the navigation pane, and then choose **Create security group**.
3. Enter a **Security group name** and **Description** for the security group that helps you find it. For example, **`HFClientSSH`**.
4. Make sure that the VPC you select is the same VPC that you will select for the interface VPC endpoint.
5. Choose **Inbound**, and then choose **Add rule**.
6. Under **Type**, select **SSH** from the list.
7. Under **Source**, select **My IP**. This adds the detected IP address of your current computer. Optionally, you can create additional rules for SSH connections from additional IP addresses or sources if required.
8. Choose **Create**.

   You will reference this security group later in this tutorial in Step 3: Create an Amazon EC2 Instance and Set Up the Hyperledger Fabric Client (p. 11).

Additional Considerations

- All commands in the tutorial assume that you are using an Amazon EC2 instance with an Amazon Linux AMI. Unless noted otherwise, instructions also assume that you are running commands in the

default home directory (`/home/ec2-user`). If you have a different configuration, modify instructions to fit your home directory as necessary.

- Command examples are preceded by a prompt (`[ec2-user@ip-192-0-2-17 ~]$`) where appropriate for clarity. Remove the prompt if you copy and paste examples.
- Hyperledger Fabric 1.2 requires that a channel ID contain only lowercase ASCII alphanumeric characters, dots (.), and dashes (-). It must start with a letter, and must be fewer than 250 characters.

Step 1: Create the Network and First Member

When you create the network, you specify the following parameters along with basic information such as names and descriptions:

- The open-source framework and version. This tutorial uses Hyperledger Fabric version 1.2.
- The voting policy for proposals on the network. For more information, see Work with Proposals (p. 47).
- The first member of the network, including the administrative user and administrative password that are used to authenticate to the member's certificate authority (CA).

Create the network using the AWS CLI or Managed Blockchain management console according to the following instructions. It may take a few minutes for Managed Blockchain to provision resources and bring the network online.

To create a Managed Blockchain network using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Create network**.
3. Under **Blockchain framework**:
   a. Select the blockchain framework to use. This tutorial is based on **Hyperledger Fabric version 1.2**.
   b. Select the **Network edition** to use. The network edition determines attributes of the network, such as the maximum number of members, nodes per member, and transaction throughput. Different editions have different rates associated with the membership. For more information, see Amazon Managed Blockchain Pricing.
4. Enter a **Network name and description**.
5. Under **Voting Policy**, choose the following:
   a. Enter the **Approval threshold percentage** along with the comparator, either **Greater than** or **Greater than or equal to**. For a proposal to pass, the Yes votes cast must meet this threshold before the vote duration expires.
   b. Enter the **Proposal duration in hours**. If enough votes are not cast within this duration to either approve or reject a proposal, the proposal status is `EXPIRED`, no further votes on this proposal are allowed, and the proposal does not pass.
6. Choose **Next**, and then, under **Create member**, do the following to define the first member for the network, which you own:
   a. Enter a **Member name** that will be visible to all members and an optional **Description**.
   b. Under **Hyperledger Fabric certificate authority (CA) configuration** specify a username and password to be used as the administrator on the Hyperledger Fabric CA. Remember the user

name and password. You need them later any time that you create users and resources that need to authenticate.

c. Choose **Create member and join network**.

7. Review **Network options** and **Member options**, and then choose **Create network and member**.

   The **Networks** list shows the name and **Network ID** of the network you created, with a **Status** of **Creating**. It may take a minute or two for Managed Blockchain to create your network, after which the **Status** is **Active**.

To create a Managed Blockchain network using the AWS CLI

Use the `create-network` command as shown in the following example. Consider the following:

- The example shows `HYPERLEDGER_FABRIC` as the `Framework` and `1.2` as the `FrameworkVersion`. The `FrameworkConfiguration` properties for `--network-configuration` and `--member-configuration` options may be different for other frameworks and versions.
- The `AdminPassword` must be at least 8 characters long and no more than 32 characters. It must contain at least one uppercase letter, one lowercase letter, and one digit. It cannot have a single quote(‘), double quote(“), forward slash(/), backward slash(\), @, percent sign (%), or a space.
- Remember the user name and password. You need them later any time you create users and resources that need to authenticate.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain create-network \
--network-configuration '{"Name":"MyTestTaigaNetwork", \
"Description":"MyTaigaNetDescription", \
"Framework":"HYPERLEDGER_FABRIC","FrameworkVersion": "1.2", \
"FrameworkConfiguration": {"Fabric": {"Edition": "STARTER"}}, \
"VotingPolicy": {"ApprovalThresholdPolicy": {"ThresholdPercentage": 50, \
"ProposalDurationInHours": 24, \
"ThresholdComparator": "GREATER_THAN"}}}' \
--member-configuration '{"Name":"org1", \
"Description":"Org1 first member of network",\
"FrameworkConfiguration":{"Fabric":
{"AdminUsername":"AdminUser","AdminPassword":"Password123"}}}'
```

The command returns the Network ID and the Member ID, as shown in the following example:

```
{
    "NetworkId": "n-MWY63ZJZU5HGNCMBQER7IN6OIU",
    "MemberId": "m-K46ICRRXJRCGRNNS4ES4XUUS5A"
}
```

The **Networks** page on the console shows a **Status** of **Active** when the network is ready. Alternatively, you can use the `list-networks` command, as shown in the following example, to confirm the network status.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain list-networks
```

The command returns information about the network, including an `AVAILABLE` status.

```
{
    "Networks": [
        {
```

```
            "Id": "n-MWY63ZJZU5HGNCMBQER7IN6OIU",
            "Name": "MyTestNetwork",
            "Description": "MyNetDescription",
            "Framework": "HYPERLEDGER_FABRIC",
            "FrameworkVersion": "1.2",
            "Status": "AVAILABLE",
            "CreationDate": 1541497086.888,
        }
    ]
}
```

Step 2: Create and Configure the Interface VPC Endpoint

Now that the network is up and running in your VPC, you set up an interface VPC endpoint (AWS PrivateLink) for your member. This allows the Amazon EC2 instance that you use as a Hyperledger Fabric client to interact with the Hyperledger Fabric endpoints that Amazon Managed Blockchain exposes for your member and network resources. For more information, see Interface VPC Endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*. Applicable charges for interface VPC endpoints apply. For more information, see AWS PrivateLink Pricing.

The AWS Identity and Access Management (IAM) principal (user) identity that you use must have sufficient IAM permissions to create an interface VPC endpoint in your AWS account. For more information, see Controlling Access - Creating and Managing VPC Endpoints in the *Amazon VPC User Guide*.

You can create the interface VPC endpoint using a shortcut in the Managed Blockchain console.

To create an interface VPC endpoint using the Managed Blockchain console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Networks**, select your network from the list, and then choose **View details**.
3. Choose **Create VPC endpoint**.
4. Choose a **VPC**.
5. For **Subnets**, choose a subnet from the list, and then choose additional subnets as necessary.
6. For **Security groups**, choose an EC2 security group from the list, and then choose additional security groups as necessary. We recommend that you select the same security group that your framework client EC2 instance is associated with.
7. Choose **Create**.

Step 3: Create an Amazon EC2 Instance and Set Up the Hyperledger Fabric Client

To complete this step, you launch an Amazon EC2 instance using the Amazon Linux AMI. Consider the following requirements and recommendations when you create the Hyperledger Fabric client Amazon EC2 instance:

- We recommend that you launch the client Amazon EC2 instance in the same VPC and using the same security group as the VPC Endpoint that you created in Step 2: Create and Configure the Interface VPC

Endpoint (p. 11). This simplifies connectivity between the Amazon EC2 instance and the Interface VPC Endpoint.

- We recommend that the EC2 security group shared by the VPC Endpoint and the client Amazon EC2 instance have rules that allow all inbound and outbound traffic between members of the security group. This also simplifies connectivity. In addition, ensure that this security group or another security group associated with the client Amazon EC2 instance has a rule that allows inbound SSH connections from a source that includes your SSH client's IP address. For more information about security groups and required rules, see Configuring Security Groups (p. 67).
- Make sure that the client Amazon EC2 instance is configured with an automatically assigned public IP address and an Amazon EC2 key pair so that you can connect to it using SSH.

For more information, see Getting Started with Amazon EC2 Linux Instances.

Note
An AWS CloudFormation template to create a Hyperledger Fabric client is available in amazon-managed-blockchain-client-templates repository on Github. For more information, see the readme.md in that repository. For more information about using AWS CloudFormation, see Getting Started in the *AWS CloudFormation User Guide*.

Verify That Version 1.16.149 or Later of The AWS CLI is Installed

After you create an Amazon EC2 instance, connect to it using SSH and verify that AWS CLI version 1.16.149 or later is installed. For more information about connecting to an Amazon EC2 instance, see Connect to Your Linux Instance in the *Amazon EC2 User Guide for Linux Instances*.

```
[ec2-user@ip-192-0-2-17 ~]$ aws --version
aws-cli/1.16.245 Python/2.7.16 Linux/4.14.138-114.102.amzn2.x86_64 botocore/1.12.235
```

We recommend that you update the AWS CLI to the latest version using `pip`.

Important
Using sudo to complete a command grants the command full access to your system. The examples below use it for simplicity. We recommend using it only when a more secure option is not available. For more information about installing and updating the AWS CLI, see Install the AWS CLI on Amazon Linux in the *AWS Command Line Interface User Guide*.

To upgrade your Amazon EC2 instance to the latest version of the AWS CLI

1. Connect to the instance.
2. Install `pip`.
   a. Use `curl` to download the `get-pip.py` script.

      ```
      curl -o get-pip.py https://bootstrap.pypa.io/get-pip.py
      ```

   b. Use `python` to run `get-pip.py`.

      ```
      sudo python get-pip.py
      ```

3. Use `pip` to upgrade the AWS CLI.

   ```
   sudo pip install awscli --upgrade
   ```

4. Use `aws --version` to verify that the version of the AWS CLI you use has been upgraded.

Step 3.1: Install Packages

Your Hyperledger Fabric client needs some packages and samples installed so that you can work with the Hyperledger Fabric resources. In this step, you install Go, Docker, Docker Compose, and some other utilities. You also create variables in the `~/.bash_profile` for your development environment. These are prerequisites for installing and using Hyperledger tools.

While connected to the Hyperledger Fabric client using SSH, run the following commands to install utilties, install docker, and configure the Docker user to be the default user for the Amazon EC2 instance:

```
[ec2-user@ip-192-0-2-17 ~]$ sudo yum update -y
[ec2-user@ip-192-0-2-17 ~]$ sudo yum install -y telnet
[ec2-user@ip-192-0-2-17 ~]$ sudo yum -y install emacs
[ec2-user@ip-192-0-2-17 ~]$ sudo yum install -y docker
[ec2-user@ip-192-0-2-17 ~]$ sudo service docker start
[ec2-user@ip-192-0-2-17 ~]$ sudo usermod -a -G docker ec2-user
```

Log out and log in again for the `usermod` command to take effect.

Run the following commands to install Docker Compose:

```
[ec2-user@ip-192-0-2-17 ~]$ sudo curl -L \
https://github.com/docker/compose/releases/download/1.20.0/docker-compose-`uname \
-s`-`uname -m` -o /usr/local/bin/docker-compose
[ec2-user@ip-192-0-2-17 ~]$ sudo chmod a+x /usr/local/bin/docker-compose
[ec2-user@ip-192-0-2-17 ~]$ sudo yum install libtool -y
```

Run the following commands to install golang:

```
[ec2-user@ip-192-0-2-17 ~]$ wget https://dl.google.com/go/go1.10.3.linux-amd64.tar.gz
[ec2-user@ip-192-0-2-17 ~]$ tar -xzf go1.10.3.linux-amd64.tar.gz
[ec2-user@ip-192-0-2-17 ~]$ sudo mv go /usr/local
[ec2-user@ip-192-0-2-17 ~]$ sudo yum install libtool-ltdl-devel -y
[ec2-user@ip-192-0-2-17 ~]$ sudo yum install git -y
```

Use a text editor to set up variables such as `GOROOT` and `GOPATH` in your `~/.bashrc` or `~/.bash_profile` and save the updates. The following example shows entries in `.bash_profile`.

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs
PATH=$PATH:$HOME/.local/bin:$HOME/bin

# GOROOT is the location where Go package is installed on your system
export GOROOT=/usr/local/go

# GOPATH is the location of your work directory
export GOPATH=$HOME/go

# Update PATH so that you can access the go binary system wide
```

```
export PATH=$GOROOT/bin:$PATH
export PATH=$PATH:/home/ec2-user/go/src/github.com/hyperledger/fabric-ca/bin
```

After you update `.bash_profile`, apply the changes:

```
[ec2-user@ip-192-0-2-17 ~]$ source ~/.bash_profile
```

After the installation, verify that you have the correct versions installed:

- Docker–17.06.2-ce or later
- Docker-compose–1.14.0 or later
- Go–1.10.x

To check the Docker version, run the following command:

```
[ec2-user@ip-192-0-2-17 ~]$ sudo docker version
```

The command returns output similar to the following:

```
Client:
 Version: 18.06.1-ce
 API version: 1.38
 Go version: go1.10.3
 Git commit: CommitHash
 Built: Tue Oct 2 18:06:45 2018
 OS/Arch: linux/amd64
 Experimental: false

Server:
 Engine:
 Version: 18.06.1-ce
 API version: 1.38 (minimum version 1.12)
 Go version: go1.10.3
 Git commit: e68fc7a/18.06.1-ce
 Built: Tue Oct 2 18:08:26 2018
 OS/Arch: linux/amd64
 Experimental: false
```

To check the version of Docker Compose, run the following command:

```
[ec2-user@ip-192-0-2-17 ~]$ sudo /usr/local/bin/docker-compose version
```

The command returns output similar to the following:

```
docker-compose version 1.22.0, build f46880fe
docker-py version: 3.4.1
CPython version: 3.6.6
OpenSSL version: OpenSSL 1.1.0f  25 May 2017
```

To check the version of go, run the following command:

```
[ec2-user@ip-192-0-2-17 ~]$ go version
```

The command returns output similar to the following:

```
go version go1.10.3 linux/amd64
```

Step 3.2: Set Up the Hyperledger Fabric CA Client

In this step, you verify that you can connect to the Hyperledger Fabric CA using the VPC endpoint you configured in Step 2: Create and Configure the Interface VPC Endpoint (p. 11). You then install the Hyperledger Fabric CA client. The Fabric CA issues certificates to administrators and network peers.

To verify connectivity to the Hyperledger Fabric CA, you need the `CAEndpoint`. Use the `get-member` command to get the CA endpoint for your member, as shown in the following example. Replace the values of `--network-id` and `--member-id` with the values returned in Step 1: Create the Network and First Member (p. 9).

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain get-member \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-K46ICRRXJRCGRNNS4ES4XUUS5A
```

Use `curl` or `telnet` to verify that the endpoint resolves. In the following example, replace *CAEndpoint* with the **CAEndpoint** returned by the `get-member` command.

```
[ec2-user@ip-192-0-2-17 ~]$ curl https://CAEndpoint/cainfo -k
```

The command should return output similar to the following:

```
{"result":
{"CAName":"abcd1efghijkllmn5op3q52rst","CAChain":"LongStringOfCharacters","Version":"1.2.1-
snapshot-"}
,"errors":[],"messages":[],"success":true}
```

Alternatively, you can connect to the Fabric CA using Telnet as shown in the following example. Use the same endpoint in the `curl` example, but separate the endpoint and the port as shown in the following example.

```
[ec2-user@ip-192-0-2-17 ~]$ telnet CaEndpoint-Without-Port CaPort
```

The command should return output similar to the following:

```
Trying 10.0.1.228...
Connected to ca.m-K46ICRRXJRCGRNNS4ES4XUUS5A.n-
MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-east-1.amazonaws.com.
Escape character is '^]'.
```

If you are unable to connect to the Fabric CA, double-check your network settings to ensure that the client Amazon EC2 instance has connectivity with the VPC Endpoint. In particular, ensure that the security groups associated with both the VPC Endpoint and the client Amazon EC2 instance have inbound and outbound rules that allow traffic between them.

Now that you have verified that you can connect to the Hyperledger Fabric CA, run the following commands to configure the CA client:

```
[ec2-user@ip-192-0-2-17 ~]$ mkdir -p /home/ec2-user/go/src/github.com/hyperledger/fabric-ca
[ec2-user@ip-192-0-2-17 ~]$ cd /home/ec2-user/go/src/github.com/hyperledger/fabric-ca
```

```
[ec2-user@ip-192-0-2-17 ~]$ wget https://github.com/hyperledger/fabric-ca/releases/
download/v1.2.1/hyperledger-fabric-ca-linux-amd64-1.2.1.tar.gz
[ec2-user@ip-192-0-2-17 ~]$ tar -xzf hyperledger-fabric-ca-linux-amd64-1.2.1.tar.gz
```

Step 3.3: Clone the Samples Repository

```
[ec2-user@ip-192-0-2-17 ~]$ cd /home/ec2-user
[ec2-user@ip-192-0-2-17 ~]$ git clone --branch v1.2.0 https://github.com/hyperledger/
fabric-samples.git
```

Step 3.4: Configure and Run Docker Compose to Start the Hyperledger Fabric CLI

Use a text editor to create a configuration file for Docker Compose named `docker-compose-cli.yaml` in the `/home/ec2-user` directory, which you use to run the Hyperledger Fabric CLI. You use this CLI to interact with peer nodes that your member owns. Copy the following contents into the file, and then save the file in your home directory.

```
version: '2'
services:
  cli:
    container_name: cli
    image: hyperledger/fabric-tools:1.2.0
    tty: true
    environment:
      - GOPATH=/opt/gopath
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_LOGGING_LEVEL=info # Set logging level to debug for more verbose logging
      - CORE_PEER_ID=cli
      - CORE_CHAINCODE_KEEPALIVE=10
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: /bin/bash
    volumes:
        - /var/run/:/host/var/run/
        - /home/ec2-user/fabric-samples/chaincode:/opt/gopath/src/github.com/
        - /home/ec2-user:/opt/home
```

Run the following command to start the Hyperledger Fabric peer CLI container:

```
[ec2-user@ip-192-0-2-17 ~]$ docker-compose -f docker-compose-cli.yaml up -d
```

If you restarted or logged out and back in after the `usermod` command in Step 3.1: Install Packages (p. 13), you shouldn't need to run this command with `sudo`. If the command fails, you can log out and log back in. Alternatively, you can run the command using `sudo`, as shown in the following example:

```
[ec2-user@ip-192-0-2-17 ~]$ sudo /usr/local/bin/docker-compose -f docker-compose-cli.yaml
 up -d
```

Step 4: Enroll an Administrative User

In this step, you use a pre-configured certificate to enroll a user with administrative permissions to your member's certificate authority (CA). To do this, you must create a certificate file. You also need the

endpoint for the CA of your member, and the user name and password for the user that you created in Step 1: Create the Network and First Member (p. 9).

Step 4.1: Create the Certificate File

Run the following command to copy the `managedblockchain-tls-chain.pem` to the `/home/ec2-user` directory:

```
aws s3 cp s3://us-east-1.managedblockchain/etc/managedblockchain-tls-chain.pem  /home/ec2-user/managedblockchain-tls-chain.pem
```

Run the following command to test that you copied the contents to the file correctly:

```
[ec2-user@ip-192-0-2-17 ~]$ openssl x509 -noout -text -in /home/ec2-user/managedblockchain-tls-chain.pem
```

The command should return the contents of the certificate in human-readable format.

Step 4.2: Enroll the Administrative User

Managed Blockchain registers the user identity that you specified when you created the member as an administrator. In Hyperledger Fabric, this user is known as the *bootstrap identity* because the identity is used to enroll itself. To enroll, you need the CA endpoint, as well as the user name and password for the administrator that you created in Step 1: Create the Network and First Member (p. 9). For information about registering other user identities as administrators before you enroll them, see Register and Enroll a User as an Administrator (p. 55).

Use the `get-member` command to get the CA endpoint for your membership as shown in the following example. Replace the values of `--network-id` and `--member-id` with the values returned in Step 1: Create the Network and First Member (p. 9).

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain get-member \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-K46ICRRXJRCGRNNS4ES4XUUS5A
```

The command returns information about the initial member that you created in the network, as shown in the following example. Make a note of the `CaEndpoint`. You also need the `AdminUsername` and password that you created along with the network.

The command returns output similar to the following:

```
{
    "Member": {
        "NetworkId": "n-MWY63ZJZU5HGNCMBQER7IN6OIU",
        "Status": "AVAILABLE",
        "Description": "MyNetDescription",
        "FrameworkAttributes": {
            "Fabric": {
                "CaEndpoint": "ca.m-K46ICRRXJRCGRNNS4ES4XUUS5A.n-MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-east-1.amazonaws.com:30002",
                "AdminUsername": "AdminUser"
            }
        },
        "StatusReason": "Network member created successfully",
        "CreationDate": 1542255358.74,
        "Id": "m-K46ICRRXJRCGRNNS4ES4XUUS5A",
```

```
        "Name": "org1"
    }
}
```

Use the CA endpoint, administrator profile, and the certificate file to enroll the member administrator using the `fabric-ca-client enroll` command, as shown in the following example:

```
[ec2-user@ip-192-0-2-17 ~]$ fabric-ca-client enroll \
-u https://AdminUsername:AdminPassword@SampleCAEndpointAndPort \
--tls.certfiles /home/ec2-user/managedblockchain-tls-chain.pem -M /home/ec2-user/admin-msp
```

An example command with fictitious administrator name, password, and endpoint is shown in the following example:

```
[ec2-user@ip-192-0-2-17 ~]$ fabric-ca-client enroll \
-u https://AdminUser:Password123@ca.m-K46ICRRXJRCGRNNS4ES4XUUS5A.n-
MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-east-1.amazonaws.com:30002 \
--tls.certfiles /home/ec2-user/managedblockchain-tls-chain.pem -M /home/ec2-user/admin-msp
```

The command returns output similar to the following:

```
2018/11/16 02:21:40 [INFO] Created a default configuration file at /home/ec2-user/.fabric-
ca-client/fabric-ca-client-config.yaml
2018/11/16 02:21:40 [INFO] TLS Enabled
2018/11/16 02:21:40 [INFO] generating key: &{A:ecdsa S:256}
2018/11/16 02:21:40 [INFO] encoded CSR
2018/11/16 02:21:40 [INFO] Stored client certificate at /home/ec2-user/admin-msp/signcerts/
cert.pem
2018/11/16 02:21:40 [INFO] Stored root CA certificate at /home/ec2-user/admin-msp/cacerts/
ca-abcd1efghijkllmn5op3q52rst-uqz2f2xakfd7vcfewqhckr7q5m-managedblockchain-us-east-1-
amazonaws-com-30002.pem
```

Step 4.3: Copy Certificates for the MSP

In Hyperledger Fabric, the Membership Service Provider (MSP) identifies which root CAs and intermediate CAs are trusted to define the members of a trust domain. Certificates for the administrator's MSP are in `$FABRIC_CA_CLIENT_HOME`, which is `/home/ec2-user/admin-msp` in this tutorial. Because this MSP is for the member administrator, copy the certificates from `signcerts` to `admincerts` as shown in the following example:

```
[ec2-user@ip-192-0-2-17 ~]$ cp -r admin-msp/signcerts admin-msp/admincerts
```

Important
It may take a minute or two after you enroll for you to be able to use your administrator certificate to create a channel with the ordering service.

Step 5: Create a Peer Node in Your Membership

Now that you are enrolled as an administrator for your member, you can use your client to create a peer node. Your member's peer nodes interact with other members' peer nodes on the blockchain to query and update the ledger, and store a local copy of the ledger.

Wait a minute or two for the administrative permissions from previous steps to propagate, and then use one of the following procedures to create a peer node.

To create a peer node using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Networks**, select the network from the list, and then choose **View details**.
3. Select a **Member** from the list, and then choose **Create peer node**.
4. Choose configuration parameters for your peer node according to the previous guidelines, and then choose **Create peer node**.

To create a peer node using the AWS CLI

- Use the `create-node` command, as shown in the following example. Replace the value of `--network-id`, `--member-id`, and `AvailabilityZone` as appropriate.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain create-node \
--node-configuration '{"InstanceType":"bc.t3.small","AvailabilityZone":"us-east-1a"}' \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-K46ICRRXJRCGRNNS4ES4XUUS5A
```

The command returns output that includes the peer node's `NodeID`, as shown in the following example:

```
{
     "NodeId": "nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y"
}
```

Step 6: Create a Hyperledger Fabric Channel

In Hyperledger Fabric, a ledger exists in the scope of a channel. The ledger can be shared across the entire network if every member is operating on a common channel. A channel also can be privatized to include only a specific set of participants. Members can be in your AWS account, or they can be members that you invite from other AWS accounts.

In this step, you set up a basic channel. Later on in the tutorial, in Step 8: Invite Another AWS Account to be a Member and Create a Joint Channel (p. 23), you go through a similar process to set up a channel that includes another member.

Step 6.1: Create configtx for Hyperledger Fabric Channel Creation

The `configtx.yaml` file contains details of the channel configuration. For more information, see Channel Configuration (configtx) in the Hyperledger Fabric documentation.

Use a text editor to create a file with the following contents and save it as `configtx.yaml` on your Hyperledger Fabric client. Replace *`MemberID`* with the MemberID you returned previously. For example *`m-K46ICRRXJRCGRNNS4ES4XUUS5A`*.

Important
This file is sensitive. Artifacts from pasting can cause the file to fail with marshalling errors. We recommend using `emacs` to edit it. You can also use `VI`, but before using `VI`, enter `:set paste`, press `i` to enter insert mode, paste the contents, press escape, and then enter `:set nopaste` before saving.

```
################################################################################
#
#   Section: Organizations
#
#   - This section defines the different organizational identities which will
#   be referenced later in the configuration.
#
################################################################################
Organizations:
    - &Org1
            # DefaultOrg defines the organization which is used in the sampleconfig
            # of the fabric.git development environment
        Name: MemberID
            # ID to load the MSP definition as
        ID: MemberID
        MSPDir: /opt/home/admin-msp
            # AnchorPeers defines the location of peers which can be used
            # for cross org gossip communication.  Note, this value is only
            # encoded in the genesis block in the Application section context
        AnchorPeers:
            - Host:
              Port:

################################################################################
#
#   SECTION: Application
#
#   - This section defines the values to encode into a config transaction or
#   genesis block for application related parameters
#
################################################################################
Application: &ApplicationDefaults
        # Organizations is the list of orgs which are defined as participants on
        # the application side of the network
    Organizations:

################################################################################
#
#   Profile
#
#   - Different configuration profiles may be encoded here to be specified
#   as parameters to the configtxgen tool
#
################################################################################
Profiles:
    OneOrgChannel:
        Consortium: AWSSystemConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
```

Run the following command to generate the configtx peer block:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec cli configtxgen \
-outputCreateChannelTx /opt/home/mychannel.pb \
-profile OneOrgChannel -channelID mychannel \
--configPath /opt/home/
```

Important
Hyperledger Fabric 1.2 requires that a channel ID contain only lowercase ASCII alphanumeric characters, dots (.), and dashes (-). It must start with a letter, and must be fewer than 250 characters.

Step 6.2: Set Environment Variables for Convenience

Set the following environment variables for convenience. Replace the following values as indicated:

- *m-K46ICRRXJRCGRNNS4ES4XUUS5A* is the `MemberID` returned by the `aws managedblockchain list-members` AWS CLI command and shown on the member details page of the Managed Blockchain console.
- *orderer.n-MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.amazonaws.com:30001* is the `OrderingServiceEndpoint` returned by the `aws managedblockchain get-network` command and listed on the network details page of the Managed Blockchain console.
- *nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y.m-K46ICRRXJRCGRNNS4ES4XUUS5A.n-MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-east-1.amazonaws.com:30003* is the `PeerEndpoint` returned by the `aws managedblockchain get-node` command and listed on the node details page of the Managed Blockchain console.

```
[ec2-user@ip-192-0-2-17 ~]$ export MSP_PATH=/opt/home/admin-msp
[ec2-user@ip-192-0-2-17 ~]$ export MSP=m-K46ICRRXJRCGRNNS4ES4XUUS5A
[ec2-user@ip-192-0-2-17 ~]$ export ORDERER=orderer.n-
MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.amazonaws.com:30001
[ec2-user@ip-192-0-2-17 ~]$ export PEER=nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y.m-
K46ICRRXJRCGRNNS4ES4XUUS5A.n-MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-
east-1.amazonaws.com:30003
```

These variables must be exported each time you log out of the client. To persist the variables across sessions, add the export statement to your `~/.bash_profile` as shown in the following example.

```
# .bash_profile
...other configurations
export MSP_PATH=/opt/home/admin-msp
export MSP=MemberID
export ORDERER=OrderingServiceEndpoint
export PEER=PeerNodeEndpoint
```

After updating `.bash_profile`, apply the changes:

```
[ec2-user@ip-192-0-2-17 ~]$source ~/.bash_profile
```

Step 6.3: Create the Channel

Run the following command to create a channel using the variables that you established and the configtx peer block that you created:

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer channel create -c mychannel \
-f /opt/home/mychannel.pb -o $ORDERER \
--cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

Step 6.4: Join Your Peer Node to the Channel

Run the following command to join the peer node that you created earlier to the channel:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer channel join -b mychannel.block \
-o $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

Step 7: Install and Run Chaincode

This section shows you how to install sample chaincode on your peer, instantiating the chaincode, querying the chaincode, and invoking the chaincode to update values.

Step 7.1: Install Chaincode

Run the following command to install example chaincode on the peer node:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
-e "CORE_PEER_ADDRESS=$PEER" \
cli peer chaincode install \
-n mycc -v v0 -p github.com/chaincode_example02/go
```

Step 7.2: Instantiate Chaincode

Run the following command to instantiate the chaincode:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
-e "CORE_PEER_ADDRESS=$PEER" \
cli peer chaincode instantiate \
-o $ORDERER -C mychannel -n mycc -v v0 \
-c '{"Args":["init","a","100","b","200"]}' \
--cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

You may have to wait a minute or two for the instantiation to propagate to the peer node. Use the following command to verify instantiation:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e  "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
-e "CORE_PEER_ADDRESS=$PEER"  \
cli peer chaincode list --instantiated \
-o $ORDERER -C mychannel \
--cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

The command returns the following when the chaincode is instantiated:

```
Get instantiated chaincodes on channel mychannel:
```

```
Name: mycc, Version: v0, Path: github.com/chaincode_example02/go, Escc: escc, Vscc: vscc
```

Step 7.3: Query the Chaincode

You may need to wait a brief moment for the instantiation from the previous step to complete before you run the following command to query a value:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer chaincode query -C mychannel \
-n mycc -c '{"Args":["query","a"]}'
```

The command should return the value of `a`, which you instantiated to a value of `100`.

Step 7.4: Invoke the Chaincode

In the previous steps, we instantiated the key `a` with a value of `100` and queried to verify. Using the `invoke` command in the following example, we remove `10` from that initial value:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER"  -e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer chaincode invoke -C mychannel \
-n mycc -c  '{"Args":["invoke","a","b","10"]}' \
-o $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

When we query again using the following command:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer chaincode query -C mychannel \
-n mycc -c '{"Args":["query","a"]}'
```

The command should return the value of `a` as the new value `90`.

Step 8: Invite Another AWS Account to be a Member and Create a Joint Channel

Now that you have a Hyperledger Fabric network set up using Amazon Managed Blockchain, with an initial member in your AWS account and a VPC endpoint with a service name, you are ready to invite additional members. You invite additional members by creating a proposal for an invitation that existing members vote on. Since the blockchain network at this point consists of only one member, the first member always has the only vote on the invitation proposal for the second member. In the steps that follow, the network creator has an initial member named `org1` and the invited member is named `org2`. For proof of concept, you can create an invitation proposal for an additional member in the same AWS

account that you used to create the network, or you can create an invitation proposal for a different AWS account.

After the invitation proposal is approved, the invited account can create a member. Invited members are free to reject the invitation or ignore it until the invitation proposal expires. The invited account needs the network ID and VPC endpoint service name of the blockchain network to create a member. For more information, see Work with Invitations (p. 38). The invited account also needs to fulfill the prerequisites listed in Prerequisites and Considerations (p. 6).

Step 8.1: Create an Invitation Proposal

Create a proposal to invite an AWS account to create a member and join the network according to the following procedures. You need the AWS account ID of the member you want to invite. You can also invite your own account to create an additional member. If you are using the CLI, you also need the Network ID and Member ID that you created in Step 1: Create the Network and First Member (p. 9).

To create an invitation proposal using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. From the navigation pane, choose **Networks**, and then choose the network to which you want to invite an AWS account.
3. Choose **Proposals** and then choose **Propose invitation**.
4. For **Submit proposal as**, choose the member in your account that submits the proposal.

   Note
   The member who submits the proposal must also vote on it. A Yes vote is not automatically assumed.
5. Enter an optional **Description** . The description appears to other members. It's a good way to communicate key points or a reminder about the proposal before they vote.
6. For each AWS account that you want to invite, enter the account number in the space provided. Choose **Add** to enter additional accounts.

To create an invitation proposal using the AWS CLI

- Type a command similar to the following. Replace the value of `Principal` with the AWS account ID that you want to invite. Replace the value of `--member-id` with the value for the member in your account that submits the proposal.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain create-proposal \
--actions Invitations=[{Principal=123456789012}] \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-K46ICRRXJRCGRNNS4ES4XUUS5A
```

  The command returns the proposal ID, as shown in the following example:

```
{
    "ProposalId": "p-ZR7KUD2YYNESLNG6RQ33X3FUFE"
}
```

Step 8.2: Vote Yes on the Proposal

After you create the invitation proposal, use the first member that you created to vote Yes and approve the proposal. You must do this within the duration defined by the network voting policy.

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. From the navigation pane, choose **Networks**, and then choose the **Network** for which the proposal was made.
3. Choose **Proposals**.
4. Under **Active proposals**, choose the **Proposal ID** to vote on.
5. Under **Vote on proposal**, select the member in your account to vote as. If your account has multiple members, each member gets a vote.
6. Choose **Yes** to vote to approve the proposal. Voting yes is a requirement for the second member to be created in the next step. Choosing **No** rejects the proposal and an invitation is not created.
7. Choose to **Confirm** your vote.

Step 8.3: Create the New Member

To accept an invitation to create a member and join a network, the steps are similar whether you are creating a member in a Managed Blockchain network in a different AWS account or your own AWS account. You first create the member as shown in the following procedures. If you use the AWS CLI, make sure that you have the relevant information, including the Network ID and the Invitation ID that the network sent to your account. When you create a member, you specify the name that identifies your member on the network. You also specify the admin user and password to authenticate to your member certificate authority (CA).

To accept an invitation to create a member and join a network using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. From the navigation pane, choose **Invitations**.
3. Select the invitation that you want to accept from the list, and then choose **Accept invitation**. To view more information about the network you are invited to join, choose the network **Name** from the list
4. Under **Join network**, configure your network member according to the following guidelines:

   a. Enter a **Member name** that will be visible to all members and an optional **Description**.

   b. Under **Hyperledger Fabric certificate authority (CA) configuration** specify a username and password to be used as the administrator on the Hyperledger Fabric CA. Remember the user name and password. You need them later any time that you create users and resources that need to authenticate.

   c. Choose **Create member and join network**.
5. Choose **Create member**.

To accept an invitation to create a member and join a network using the AWS CLI

- Use the `create-member` command similar to the example below. Replace the value of `--network-id` with the Network ID that you are joining and `--invitation-id` with the Invitation ID sent to your account from the network.

```
aws managedblockchain create-member \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--invitation-id i-XL9MDD6LVWWDNA9FF94Y4TFTE \
--member-configuration 'Name=org2,Description=MyMemberDesc,\
FrameworkConfiguration={Fabric={AdminUsername=MyAdminUsername,\
```

```
AdminPassword=Password123}}'
```

The command returns output similar to the following:

```
{
"MemberId": "m-J46DNSFRTVCCLONS9DT5TTLS2A"
}
```

Additional Steps to Configure a Member

After you create the member, perform the following steps to configure the member. As you perform the steps, replace values with those specific to your member configuration, including the Member ID returned by the previous command. The Network ID and `OrderingServiceEndpoint` are the same for all members.

- Step 2: Create and Configure the Interface VPC Endpoint (p. 11)

  This step is only required if you are creating the second member in a different AWS account.
- Step 3: Create an Amazon EC2 Instance and Set Up the Hyperledger Fabric Client (p. 11)

  If you are creating an additional member in the same AWS account, and you already have a Hyperledger Fabric client, you can skip most of these steps. However, you should verify connectivity to the Hyperledger Fabric CA as described in Step 3.2: Set Up the Hyperledger Fabric CA Client (p. 15), using the new CA endpoint for the new member.
- Step 4: Enroll an Administrative User (p. 16)
- Step 5: Create a Peer Node in Your Membership (p. 18)

Step 8.4: Share Artifacts and Information with the Network Creator

Before a shared channel can be created, the following artifacts and information need to be shared with `org1` by `org2`:

- **org1 needs the org2 administrative certificate**—This certificate is saved the `/home/ec2-user/admin-msp/admincerts` directory on org2's Hyperledger Fabric client after Step 4: Enroll an Administrative User (p. 16). This is referenced in the following steps as `Org2AdminCertFile`
- **org1 needs the org2 root CA**—This certificate is saved to org2's `/home/ec2-user/admin-msp/cacerts` directory on org2's Hyperledger Fabric client after the same step as previous. This is referenced in the following steps as `Org2CACertFile`
- **org1 needs the `Endpoint` of the peer node that will join the channel**—This `Endpoint` value is output by the `get-node` command after Step 5: Create a Peer Node in Your Membership (p. 18) is complete.

Step 8.5: The Channel Creator (org1) Creates Artifacts for org2's MSP

In the following example, the channel creator is org1. The CA administrator for org1 copies the certificates from the step above to a location on the Hyperledger Fabric client computer. The Membership Service Provider (MSP) uses the certificates to authenticate the member.

On the channel creator's Hyperledger Fabric client, use the following commands to create directories to store the certificates, and then copy the certificates from the previous step to the new directories:

```
[ec2-user@ip-192-0-2-17 ~]$ mkdir /home/ec2-user/org2-msp
[ec2-user@ip-192-0-2-17 ~]$ mkdir /home/ec2-user/org2-msp/admincerts
[ec2-user@ip-192-0-2-17 ~]$ mkdir /home/ec2-user/org2-msp/cacerts

cp Org2AdminCerts /home/ec2-user/org2-msp/admincerts
cp Org2CACerts /home/ec2-user/org2-msp/cacerts
```

Org1 needs org2's member ID. You can get this by running the `list-members` command on org1's Hyperledger Fabric client as shown in the following example:

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain list-members \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU
```

The channel creator (org1) should verify that the required artifacts for channel creation are saved on the Hyperledger Fabric client as shown in the following list:

- Org1 MSP artifacts:
  - /home/ec2-user/admin-msp/signcerts/*certname*.pem
  - /home/ec2-user/admin-msp/admincerts/*certname*.pem
  - /home/ec2-user/admin-msp/cacerts/*certname*.pem
  - /home/ec2-user/admin-msp/keystore/*keyname*_sk
- Org2 MSP artifacts
  - /home/ec2-user/org2-msp/admincerts/*certname*.pem
  - /home/ec2-user/org2-msp/cacerts/*certname*.pem
- The TLS CA cert used for the Region:
  - /home/ec2-user/managedblockchain-tls-chain.pem
- Addresses of all peer nodes to join the channel for both org1 and org2.
- The respective member IDs of org1 and org2.
- A `configtx.yaml` file, which you create in the following step, saved to the `/home/ec2-user` directory on the channel creator's Hyperledger Fabric client.

  Note
  If you created this configtx file earlier, delete the old file, rename it, or replace it.

Step 8.6: Create configtx for the Joint Channel

The `configtx.yaml` file contains details of the channel configuration. For more information, see Channel Configuration (configtx) in the Hyperledger Fabric documentation.

The channel creator creates this file on the Hyperledger File client. If you compare this file to the file created in Step 6.1: Create configtx for Hyperledger Fabric Channel Creation (p. 19), you see that this `configtx.yaml` specifies two members in the channel.

Use a text editor to create a file with the following contents and save it as `configtx.yaml` on your Hyperledger File client. In the example below, replace *Member1ID* with the member ID of org1, which was created with the network in Step 1: Create the Network and First Member (p. 9). For example *m-K46ICRRXJRCGRNNS4ES4XUUS5A*. Replace *Member2ID* with the member ID of org2, which was created with Step 8.3: Create the New Member (p. 25).

Important
This file is sensitive. Artifacts from pasting can cause the file to fail with marshalling errors. We recommend using `emacs` to edit it. You can also use `VI`, but before using `VI`, enter `:set paste`, press `i` to enter insert mode, paste the contents, press escape, and then enter `:set nopaste` before saving.

```
################################################################################
#
#   Section: Organizations
#
#   - This section defines the different organizational identities which will
#   be referenced later in the configuration.
#
################################################################################
Organizations:
    - &Org1
            # member id defines the organization
        Name: Member1ID
            # ID to load the MSP definition as
        ID: Member1ID
            #msp dir of org1 in the docker container
        MSPDir: /opt/home/admin-msp
            # AnchorPeers defines the location of peers which can be used
            # for cross org gossip communication.  Note, this value is only
            # encoded in the genesis block in the Application section context
        AnchorPeers:
            - Host:
              Port:
    - &Org2
        Name: Member2ID
        ID: Member2ID
        MSPDir: /opt/home/org2-msp
        AnchorPeers:
            - Host:
              Port:

################################################################################
#
#   SECTION: Application
#
#   - This section defines the values to encode into a config transaction or
#   genesis block for application related parameters
#
################################################################################
Application: &ApplicationDefaults
        # Organizations is the list of orgs which are defined as participants on
        # the application side of the network
     Organizations:

################################################################################
#
#   Profile
#
#   - Different configuration profiles may be encoded here to be specified
#   as parameters to the configtxgen tool
#
################################################################################
Profiles:
    TwoOrgChannel:
        Consortium: AWSSystemConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
                - *Org2
```

Run the following command to generate the configtx peer block:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec cli configtxgen \
-outputCreateChannelTx /opt/home/ourchannel.pb \
```

```
-profile TwoOrgChannel -channelID ourchannel \
--configPath /opt/home/
```

Step 8.7: Create the Channel

The channel creator (org1) uses the following command on their Hyperledger Fabric client to create the channel `ourchannel`. The command example assumes that environment variables have been configured as described in Step 6.2: Set Environment Variables for Convenience (p. 21).

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer channel create -c ourchannel \
-f /opt/home/ourchannel.pb -o $ORDERER \
--cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

Step 8.8: Get Channel Genesis Block

A member who joins the channel must get the channel genesis block. In this example, org2 runs the following command from their Hyperledger Fabric client to get the genesis block.

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer channel fetch oldest /opt/home/ourchannel.block \
-c ourchannel -o $ORDERER \
--cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

Step 8.9: Join Peer Nodes to the Channel

Both org1 and org2 need to run the following command on their respective Hyperledger Fabric clients to join their peer nodes to the channel:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer channel join -b /opt/home/ourchannel.block \
-o $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

Step 8.10: Install Chaincode

Both org1 and org2 run the following command on their respective Hyperledger Fabric clients to install example chaincode on their respective peer nodes:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
-e "CORE_PEER_ADDRESS=$PEER" \
cli peer chaincode install  -n mycc -v v0 \
-p github.com/chaincode_example02/go
```

Step 8.11: Instantiate Chaincode

The channel creator (org1) runs the following command to instantiate the chaincode with an endorsement policy that requires both org1 and org2 to endorse all transactions. Replace *Member1ID* with the member ID of org1 and *Member2ID* with the member ID of org2. You can use the `list-members` command to get them.

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e  "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
-e "CORE_PEER_ADDRESS=$PEER" \
cli peer chaincode instantiate -o $ORDERER \
-C ourchannel -n mycc -v v0 \
-c '{"Args":["init","a","100","b","200"]}' \
--cafile /opt/home/managedblockchain-tls-chain.pem --tls \
-P "AND ('Member1ID.member','Member2ID.member')"
```

You may need to wait a brief moment for the instantiation from the previous step to complete before you run the following command to query a value:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer chaincode query -C ourchannel \
-n mycc -c '{"Args":["query","a"]}'
```

The command should return the value of `a`, which you instantiated to a value of `100`.

Step 8.12: Invoke Chaincode

With the channel created and configured with both members, and the chaincode instantiated with values and an endorsement policy, channel members can invoke chaincode. This example command is similar to the example in Step 7.4: Invoke the Chaincode (p. 23). However, the command uses the `--peerAddresses` option to specify the endpoints of peer nodes that belong to members in the endorsement policy. The example specifies *Org2PeerNodeEndpoint* in addition to `$PEER`, which indicates the command is run from an org1 Hyperledger Fabric client. If the command runs from the org1 Hyperledger Fabric client, *Org2PeerNodeEndpoint* is specified.

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer chaincode invoke \
-C ourchannel -n mycc -c '{"Args":["invoke","a","b","10"]}' \
--peerAddresses $PEER \
--tlsRootCertFiles /opt/home/managedblockchain-tls-chain.pem \
--peerAddresses Org2PeerNodeEndpoint \
--tlsRootCertFiles /opt/home/managedblockchain-tls-chain.pem \
-o $ORDERER --cafile /opt/home/managedblockchain-tls-chain.pem --tls
```

When we query again using the following command:

```
[ec2-user@ip-192-0-2-17 ~]$ docker exec -e "CORE_PEER_TLS_ENABLED=true" \
-e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \
```

```
-e "CORE_PEER_ADDRESS=$PEER" \
-e "CORE_PEER_LOCALMSPID=$MSP" \
-e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \
cli peer chaincode query -C ourchannel \
-n mycc -c '{"Args":["query","a"]}'
```

The command should return the value of `a` as the new value `90`.

Create an Amazon Managed Blockchain Network

When a user in an AWS account creates a blockchain network on Amazon Managed Blockchain, they also create the first member in the network. This first member has no peer nodes associated with it until you create them. After you create the network and the first member, you can use that member to create an invitation proposal for other members in the same AWS account or in other AWS accounts. Any member can create an invitation proposal.

When you create the network and the first member in your AWS account, the network exists. However, transactions cannot be conducted and the ledger does not exist because there are no peer nodes. Do the following tasks to make your network functional:

- Create an interface VPC endpoint based on the network's **VPC service name** so that you can privately connect to resources. For more information, see Create an Interface VPC Endpoint to Connect to Managed Blockchain Network Resources (p. 45).
- Create at least one peer node in your first membership to interact with the network and to create and endorse transactions. For more information, see Work with Peer Nodes in a Managed Blockchain Network (p. 42).
- Create an invitation proposal for other AWS accounts to be members of the network, or invite an additional member in your account to simulate a multi-AWS account network. Vote Yes on your own proposal to approve it and create the invitation. For more information about inviting members, see Create an Invitation Proposal (p. 52).

Create a Managed Blockchain Network

You can create a Managed Blockchain network using the AWS Management Console, the AWS CLI, or the Managed Blockchain SDK CreateNetwork action.

To create a Managed Blockchain network using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Create network**.
3. Under **Blockchain framework**:
   a. Select the blockchain framework to use. This tutorial is based on **Hyperledger Fabric version 1.2**.
   b. Select the **Network edition** to use. The network edition determines attributes of the network, such as the maximum number of members, nodes per member, and transaction throughput. Different editions have different rates associated with the membership. For more information, see Amazon Managed Blockchain Pricing.
4. Enter a **Network name and description**.
5. Under **Voting Policy**, choose the following:
   a. Enter the **Approval threshold percentage** along with the comparator, either **Greater than** or **Greater than or equal to**. For a proposal to pass, the Yes votes cast must meet this threshold before the vote duration expires.

b. Enter the **Proposal duration in hours**. If enough votes are not cast within this duration to either approve or reject a proposal, the proposal status is `EXPIRED`, no further votes on this proposal are allowed, and the proposal does not pass.

6. Choose **Next**, and then, under **Create member**, do the following to define the first member for the network, which you own:

   a. Enter a **Member name** that will be visible to all members and an optional **Description**.

   b. Under **Hyperledger Fabric certificate authority (CA) configuration** specify a username and password to be used as the administrator on the Hyperledger Fabric CA. Remember the user name and password. You need them later any time that you create users and resources that need to authenticate.

   c. Choose **Create member and join network**.

7. Review **Network options** and **Member options**, and then choose **Create network and member**.

   The **Networks** list shows the name and **Network ID** of the network you created, with a **Status** of **Creating**. It may take a minute or two for Managed Blockchain to create your network, after which the **Status** is **Active**.

To create a Managed Blockchain network using the AWS CLI

Use the `create-network` command as shown in the following example. Consider the following:

- The example shows `HYPERLEDGER_FABRIC` as the `Framework` and `1.2` as the `FrameworkVersion`. The `FrameworkConfiguration` properties for `--network-configuration` and `--member-configuration` options may be different for other frameworks and versions.
- The `AdminPassword` must be at least 8 characters long and no more than 32 characters. It must contain at least one uppercase letter, one lowercase letter, and one digit. It cannot have a single quote(‘), double quote(“), forward slash(/), backward slash(\), @, percent sign (%), or a space.
- Remember the user name and password. You need them later any time you create users and resources that need to authenticate.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain create-network \
--network-configuration '{"Name":"MyTestTaigaNetwork", \
"Description":"MyTaigaNetDescription", \
"Framework":"HYPERLEDGER_FABRIC","FrameworkVersion": "1.2", \
"FrameworkConfiguration": {"Fabric": {"Edition": "STARTER"}}, \
"VotingPolicy": {"ApprovalThresholdPolicy": {"ThresholdPercentage": 50, \
"ProposalDurationInHours": 24, \
"ThresholdComparator": "GREATER_THAN"}}}' \
--member-configuration '{"Name":"org1", \
"Description":"Org1 first member of network",\
"FrameworkConfiguration":{"Fabric":
{"AdminUsername":"AdminUser","AdminPassword":"Password123"}}}'
```

The command returns the Network ID and the Member ID, as shown in the following example:

```
{
    "NetworkId": "n-MWY63ZJZU5HGNCMBQER7IN6OIU",
    "MemberId": "m-K46ICRRXJRCGRNNS4ES4XUUS5A"
}
```

Delete an Amazon Managed Blockchain Network

A blockchain network on Amazon Managed Blockchain remains active as long as there are members. A network is deleted only when the last member deletes itself from the network. No member or AWS account, even the creator's AWS account, can delete the network until they are the last member and delete themselves. When you delete the last member, all resources for that member and the blockchain network are deleted. For more information, see Delete a Member in Your AWS Account (p. 37).

Invite or Remove Members

To invite and remove other network members, any member can create a *proposal* that is submitted for a vote to all network members. If a proposal is approved within the duration and with the percentage of Yes votes specified in the voting policy for the network, the appropriate action is carried out.

A member can only join the network through an approved invitation proposal. The exception is the first member, which is created along with the network. The first member then submits a proposal and is the sole voter on the proposal to invite the second member. An AWS account can delete members from the network that they own directly. A proposal is not required. To delete a member in a different AWS account, a proposal to remove the member is required. Information about all proposals, including the member who created the proposal, the current vote count, and more is available to all network members.

This topic provides basic information for creating proposals to invite or remove members, and to delete a member that your AWS account owns. For more detailed information about proposals, including how to vote on a proposal, see Work with Proposals (p. 47).

Create a Proposal to Invite an AWS Account to the Network

You can use the AWS Management Console, the AWS CLI, or the Managed Blockchain API to create an invitation proposal. When a proposal to invite a member is approved, an invitation is sent to the specified AWS accounts. An administrator with the appropriate permissions in that account can then choose to either create a member and join the network or reject the invitation.

To create an invitation proposal using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. From the navigation pane, choose **Networks**, and then choose the network to which you want to invite an AWS account.
3. Choose **Proposals** and then choose **Propose invitation**.
4. For **Submit proposal as**, choose the member in your account that submits the proposal.

   Note
   The member who submits the proposal must also vote on it. A Yes vote is not automatically assumed.
5. Enter an optional **Description** . The description appears to other members. It's a good way to communicate key points or a reminder about the proposal before they vote.
6. For each AWS account that you want to invite, enter the account number in the space provided. Choose **Add** to enter additional accounts.

To create an invitation proposal using the AWS CLI

- Type a command similar to the following. Replace the value of `Principal` with the AWS account ID that you want to invite. Replace the value of `--member-id` with the value for the member in your account that submits the proposal.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain create-proposal \
--actions Invitations=[{Principal=123456789012}] \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-K46ICRRXJRCGRNNS4ES4XUUS5A
```

The command returns the proposal ID, as shown in the following example:

```
{
    "ProposalId": "p-ZR7KUD2YYNESLNG6RQ33X3FUFE"
}
```

Create a Proposal to Remove a Member From the Network

You can use the AWS Management Console, the AWS CLI, or the Managed Blockchain API to create a proposal to remove a member owned by another AWS account.

To create a proposal to remove a member using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. From the navigation pane, choose **Networks**, and then choose the network.
3. Choose **Proposals** and then choose **Propose removal**.
4. For **Submit proposal as**, choose the member in your account that submits the proposal.

   Note
   The member who submits the proposal must also vote on it. A Yes vote is not automatically assumed.
5. Enter an optional **Description** . The description appears to other members. It's a good way to communicate key points or a reminder about the proposal before they vote.
6. For each member that you want to remove, enter the member ID in the space provided. Choose **Add** to enter additional members.

To create a removal proposal using the AWS CLI

- Type a command similar to the following. Replace the value of `Principal` with the AWS account ID that you want to invite. Replace the value of `--member-id` with the value for the member in your account that submits the proposal.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain create-proposal \
--actions Removals=[{MemberID=m-K46ICRRXJRCGRNNS4ES4XUUS5A}] \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-J46DNSFRTVCCLONS9DT5TTLS2A
```

The command returns the proposal ID, as shown in the following example:

```
{
    "ProposalId": "p-ZR7KUD2YYNESLNG6RQ33X3FUFE"
}
```

Delete a Member in Your AWS Account

You can use the AWS Management Console, the AWS CLI, or the Managed Blockchain API to directly remove members that your AWS account owns from a network.

Warning
Removing a member in your account deletes all associated resources, such as peer nodes. For your AWS account to rejoin the network, an existing member must create a proposal to invite your AWS account, and the proposal must be approved.

To delete a member in your account using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Networks**, choose the network **Name**, and then choose **Members**.
3. Under **Members owned by you**, select a member and then choose **Delete member**.
4. Choose **Delete** when prompted to confirm.

To delete a member in your account using the AWS CLI

- Use the `delete-member` command as shown in the following example. Replace the values of `--network-id` and `--member-id` as appropriate.

```
aws managedblockchain delete-member --network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU --member-id m-J46DNSFRTVCCLONS9DT5TTLS2A
```

Accept an Invitation to Create a Member and Join a Managed Blockchain Network

In Amazon Managed Blockchain, a member is a distinct identity within the Managed Blockchain network associated with an AWS account. An AWS account can have multiple members on the network. Every member in a Managed Blockchain network must be invited to participate through a proposal made by an existing member and approved according to the network's voting policy. The exception is the first member, which is created along with the network. For more information, see Work with Proposals (p. 47). After the invitation is approved, the invited AWS account can create a member and join the network using the invitation ID.

Each member pays an hourly rate, billed per second, for their network membership, peer nodes, and peer node storage. Charges also apply to the amount of data written to the network. Charges may vary depending on the network edition selected when the network was created. For more information, see Amazon Managed Blockchain Pricing. The resources associated with a member's account depend on the specific blockchain framework and application requirements, but each member must have the following resources:

- **An interface VPC endpoint in the account**–Managed Blockchain is a PrivateLink-powered service, so you must have an interface VPC endpoint in your account to communicate with the service endpoint that the Managed Blockchain network makes available. For more information, see Create an Interface VPC Endpoint to Connect to Managed Blockchain Network Resources (p. 45) and Key Concepts: Managed Blockchain Networks, Members, and Peer Nodes (p. 2).
- **One or more peer nodes**–Each member must have at least one peer node to actively participate in the blockchain network. When you create a member it has no peer nodes by default. You create peer nodes after you create the member. Peer nodes run on Managed Blockchain instances. Custom Amazon EC2 instances or on-premises instances cannot participate as peer nodes on a Managed Blockchain network. For more information, see Work with Peer Nodes in a Managed Blockchain Network (p. 42).

Topics

Work with Invitations

If you are invited to join a Managed Blockchain network, you can accept the invitation by creating a member using the invitation ID. You can also reject the invitation. After you reject an invitation, the invitation ID is no longer valid. A new invitation proposal must be approved, and a new invitation ID is required to create a member. If don't accept or reject an invitation before it expires, the invitation lapses. As with a rejected invitation, a new invitation ID is required.

You can see all pending, accepted, and rejected invitations for your AWS account in the AWS Management Console. Alternatively, you can use the AWS CLI or the Managed Blockchain SDK ListInvitations action.

You can set up Amazon CloudWatch Events along with Amazon Simple Notification Service so that you receive an alert when there is an invitation for your account. For more information, see Automating Managed Blockchain Proposal Notifications with CloudWatch Events (p. 53).

To list blockchain network member invitations for your AWS account using the console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Invitations**, and then do one of the following:

To...	Do this...
View details about the network, such as the network ID, the description, endpoints, voting policy details, and current members.	Select the invitation from the list and choose **View details**.
Use the invitation to create a member and join the network.	Select the invitation from the list and choose **Accept Invitation**. For next steps, see Create a Member and Join a Network (p. 40)
Reject the invitation.	Select the invitation from the list and choose **Reject Invitation**.

To list blockchain network member invitations for your AWS account using the AWS CLI

- Use the following command:

```
aws managedblockchain list-invitations
```

The command returns a list of invitations, along with detail for each invitation, as shown in the following example for an invitation in the `PENDING` status:

```
{
   "Invitations": [
      {
         "CreationDate": 2019-04-08T23:40:20.628Z,
         "ExpirationDate": 2019-04-09T23:40:20.628Z,
         "InvitationId": "i-XL9MDD6LVWWDNA9FF94Y4TFTE",
         "NetworkSummary": {
            "CreationDate": 2019-04-03T13:15:22.345Z,
            "Description": "Test network for supply chain blockchain.",
            "Framework": "HYPERLEDGER_FABRIC",
            "FrameworkVersion": "1.2",
            "Id": "n-MWY63ZJZU5HGNCMBQER7IN6OIU",
            "Name": "Example Corp.",
            "Status": "AVAILABLE"
         },
         "Status": "PENDING"
      }
   ]
}
```

You can use the `InvitationID` with the `create-member` command to create a member and join the network. For next steps, see Create a Member and Join a Network (p. 40).

Create a Member and Join a Network

You can use the Managed Blockchain console, the AWS CLI, or the Managed Blockchain SDK CreateMember action to create a member in a network that your account is invited to. If you created the Managed Blockchain network, you create the first member when you create the network. All subsequent members must be invited to join by way of a member proposal.

After you create the member, for the member to be functional on the network, your account must have a VPC endpoint associated with the VPC endpoint service name published by the network. For more information, see Create an Interface VPC Endpoint to Connect to Managed Blockchain Network Resources (p. 45). You also must create at least one peer node in your membership. For more information, see Work with Peer Nodes in a Managed Blockchain Network (p. 42).

To accept an invitation to create a member and join a network using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. From the navigation pane, choose **Invitations**.
3. Select the invitation that you want to accept from the list, and then choose **Accept invitation**. To view more information about the network you are invited to join, choose the network **Name** from the list
4. Under **Join network**, configure your network member according to the following guidelines:
   a. Enter a **Member name** that will be visible to all members and an optional **Description**.
   b. Under **Hyperledger Fabric certificate authority (CA) configuration** specify a username and password to be used as the administrator on the Hyperledger Fabric CA. Remember the user name and password. You need them later any time that you create users and resources that need to authenticate.
   c. Choose **Create member and join network**.
5. Choose **Create member**.

To accept an invitation to create a member and join a network using the AWS CLI

- Use the `create-member` command similar to the example below. Replace the value of `--network-id` with the Network ID that you are joining and `--invitation-id` with the Invitation ID sent to your account from the network.

```
aws managedblockchain create-member \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--invitation-id i-XL9MDD6LVWWDNA9FF94Y4TFTE \
--member-configuration 'Name=org2,Description=MyMemberDesc,\
FrameworkConfiguration={Fabric={AdminUsername=MyAdminUsername,\
AdminPassword=Password123}}'
```

The command returns output similar to the following:

```
{
```

```
"MemberId": "m-J46DNSFRTVCCLONS9DT5TTLS2A"
}
```

After you create the member, you can use the `get-member` command to return important details about the member configuration.

Work with Peer Nodes in a Managed Blockchain Network

Peer nodes are essential. They do the work for your member on the Managed Blockchain network. They keep a local copy of the shared ledger, let you query the ledger, and interact with clients and other peer nodes to perform transactions. A new member has no peer nodes. Create at least one peer node per member. Each peer node runs on an *Managed Blockchain instance type*. You cannot add a custom Amazon EC2 instance to your member, nor can you connect an on-premises machine. The number of peer nodes and the Managed Blockchain instance type of peer nodes available to each member depends on the network edition specified when the network was created. For more information, see Amazon Managed Blockchain Pricing.

When you create a peer node, you select the following characteristics:

- **Managed Blockchain instance type**

  This determines the computational and memory capacity allocated to this node for the blockchain workload. You can choose more CPU and RAM if you anticipate a more demanding workload for each node. For example, your nodes may need to process a higher rate of transactions. Different instance types are subject to different pricing.
- **Allocated storage**

  This is the amount of storage in GiB that is available to the peer node for storing local copies of the ledger. Storage rates apply.
- **Availability Zone**

  You can select the Availability Zone to launch the peer node in. The ability to distribute peer nodes in a member across different Availability Zones allows you to design your blockchain application for resiliency. For more information, see Regions and Availability Zones in the *Amazon EC2 User Guide for Linux Instances*.

Create a Peer Node

You can create a peer node in a member that is in your AWS account using the AWS Management Console, the AWS CLI, or the Managed Blockchain SDK CreateNode action.

To create a peer node using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Networks**, select the network from the list, and then choose **View details**.
3. Select a **Member** from the list, and then choose **Create peer node**.
4. Choose configuration parameters for your peer node according to the previous guidelines, and then choose **Create peer node**.

To create a peer node using the AWS CLI

- Use the `create-node` command, as shown in the following example. Replace the value of `--network-id`, `--member-id`, and `AvailabilityZone` as appropriate.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain create-node \
```

```
--node-configuration '{"InstanceType":"bc.t3.small","AvailabilityZone":"us-east-1a"}' \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-K46ICRRXJRCGRNNS4ES4XUUS5A
```

The command returns output that includes the peer node's `NodeID`, as shown in the following example:

```
{
     "NodeId": "nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y"
}
```

View Peer Node Configuration Properties

You can view information about each peer node in your member using the AWS Management Console, the AWS CLI or the Managed Blockchain API GetNode command. Details include basic information like the Managed Blockchain instance type, Availability Zone, and creation date, along with the following important properties:

- Status
  - **Creating**

    Managed Blockchain is provisioning and configuring the Managed Blockchain instance for the peer node.
  - **Available**

    The peer node is running and available on the Managed Blockchain network.
  - **Failed**

    The peer node has an issue that has caused Managed Blockchain to add it to the deny list on the network. This usually indicates that the peer node has reached memory or storage capacity. As a first step, we recommend that you delete the instance and provision an instance with more capacity.
  - **Create Failed**

    The node could not be created with the Managed Blockchain instance type and the Availability Zone specified. We recommend trying another availability zone, a different instance type, or both.
  - **Deleting**

    The node is being deleted. This can happen because the node was deleted by the member, the member was deleted by the AWS account, or the member was deleted through an approved removal proposal.
  - **Deleted**

    The node has been deleted. See the previous item for possible reasons.
- Endpoints

  Hyperledger Fabric uses endpoints associated with each peer node to identify the peer node on the network for different processes. Managed Blockchain assigns unique peer node endpoints to each peer node on each network when the peer node is created. The peer node endpoint consists of the applicable port and the domain name of the peer node derived from the network ID, member ID, and peer node ID. For more information, see Identifying Managed Blockchain Resources and Connecting from a Client (p. 4). Do not assume that the ports for a service are the same among members; different members may use different ports for the same service. Conversely, peer nodes in different networks may use the same ports, but their endpoints are always unique.
  - **Peer endpoint**

Use this endpoint, including the port, within Hyperledger Fabric to address the peer node when using all services other than peer channel-based event services.

- **Peer event endpoint**

  Use this endpoint, including the port, within Hyperledger Fabric to address the peer node for peer channel-based event services.

You can check the peer node status using the `get-node` command, as shown in the following example:

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain get-node \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-K46ICRRXJRCGRNNS4ES4XUUS5A \
--node-id nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y
```

The command returns output that includes the peer node's `PeerEndpoint` and `PeerEventEndpoint`, as shown in the following example. You need this endpoint and port when communicating with the node using your blockchain framework client or addressing the node within an application.

```
{
   "Node": {
      "AvailabilityZone": "us-east-1a",
      "CreationDate": 2019-04-08T23:40:20.628Z,
      "FrameworkAttributes": {
         "Fabric": {
            "PeerEndpoint": "nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y.m-K46ICRRXJRCGRNNS4ES4XUUS5A.n-
MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-east-1.amazonaws.com:30003",
            "PeerEventEndpoint": "nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y.m-
K46ICRRXJRCGRNNS4ES4XUUS5A.n-MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-
east-1.amazonaws.com:30004"
         }
      },
      "Id": "nd-6EAJ5VA43JGGNPXOUZP7Y47E4Y",
      "InstanceType": "bc.t3.small",
      "Status": "AVAILABLE"
   }
}
```

Create an Interface VPC Endpoint to Connect to Managed Blockchain Network Resources

Each member of a blockchain network on Managed Blockchain needs to privately access resource endpoints from their client applications and tools. Amazon Managed Blockchain uses Interface VPC Endpoints (AWS PrivateLink) to accomplish this.

Managed Blockchain creates a *VPC service name* for each network when it is created. Each Managed Blockchain network is a unique *endpoint service* with its own VPC service name. Each member then uses the VPC service name to create an interface VPC endpoint in their account. This interface VPC endpoint lets you access resources on the Managed Blockchain network through their endpoints. AWS accounts that are not invited to the network don't have access to the VPC service name and cannot set up an interface VPC endpoint for access.

The IAM principal (user) identity that you are using must have sufficient IAM permissions to create an interface VPC endpoint in your AWS account. For more information, see Controlling Access - Creating and Managing VPC Endpoints in the *Amazon VPC User Guide*.

Any blockchain framework clients that access resources on the network need access to the interface VPC endpoint. For example, if you use an Amazon EC2 instance as a blockchain framework client, you can create it in a subnet and security group that are shared with the interface VPC endpoint.

Applicable charges for interface VPC endpoints apply. For more information, see AWS PrivateLink Pricing.

The interface VPC endpoint that you set up to access a Managed Blockchain network must be enabled for private DNS names. This requires that you create the endpoint in a VPC that has the `enableDnsHostnames` and `enableDnsSupport` options set to `true`.

To create an interface VPC endpoint using the Managed Blockchain console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Networks**, select your network from the list, and then choose **View details**.
3. Choose **Create VPC endpoint**.
4. Choose a **VPC**.
5. For **Subnets**, choose a subnet from the list, and then choose additional subnets as necessary.
6. For **Security groups**, choose an EC2 security group from the list, and then choose additional security groups as necessary. We recommend that you select the same security group that your framework client EC2 instance is associated with.
7. Choose **Create**.

To create an interface VPC Endpoint for the Managed Blockchain network

1. Find the **VPC endpoint service name** of the network. This value is returned by `get-network` command using the Managed Blockchain CLI, and is available on the network **Details** page using the Managed Blockchain console (choose **Networks**, select the network from the list, and then choose **View details**).
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.

3. Choose **Endpoints**, **Create Endpoint**.
4. Choose **Find service by name**. For **Service Name**, enter the VPC Endpoint Service Name from step 1.
5. Choose **Verify** and then choose **Create endpoint**.
6. Make sure that **Enable Private DNS Name** is selected. This option is only available if the VPC you selected has **Enable DNS hostnames** and **Enable DNS support** set to true for the VPC. This is a requirement for the VPC.
7. We recommend that the EC2 security group that you specify for the VPC endpoint is the same as the EC2 security group for the blockchain client that you create to work with the Managed Blockchain network.

Work with Proposals

To make a change to the network in Amazon Managed Blockchain that requires consensus among network members, network members create a *proposal*. For example, members can create a proposal to invite another AWS account to become a member, to invite multiple accounts, or to remove one or more members in different AWS accounts.

A proposal is submitted to all network members to make a Yes or No vote. If the proposal is approved within the duration and with the percentage of Yes votes specified in the voting policy for the network, the proposed action is carried out. The *voting policy* is established when the network is created and governs votes on all proposals. It can't be updated after the network is created. For more information, see Create an Amazon Managed Blockchain Network (p. 32).

Understanding the Proposal Lifecycle

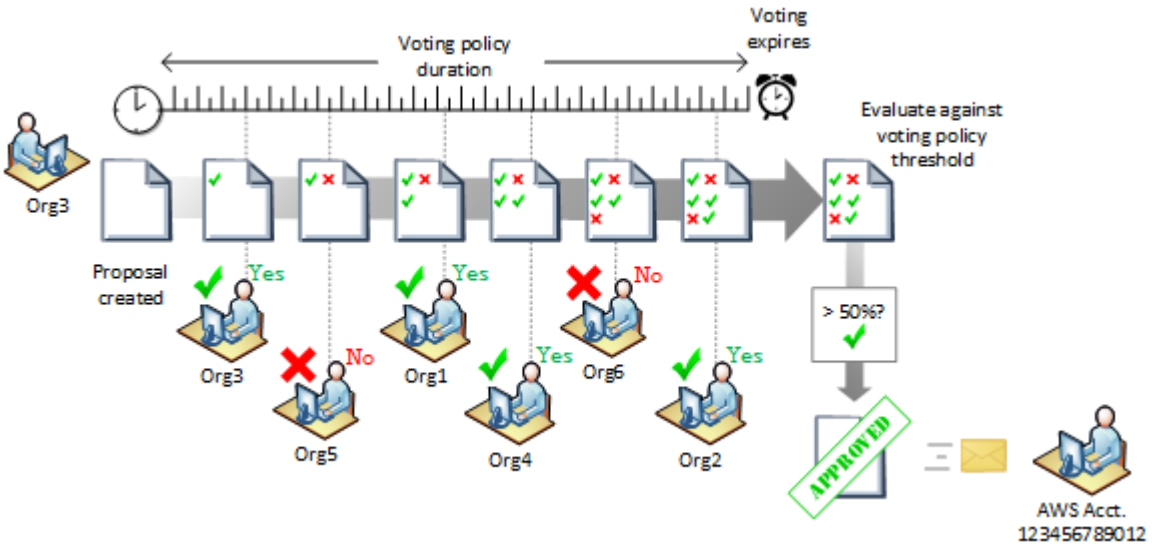
To understand the proposal lifecycle, consider a hypothetical proposal to invite AWS account 123456789012 to join a Managed Blockchain network made by a member named Org3. The Managed Blockchain network currently has six members: Org1, Org2, Org3, and so on. The network was created by Org1, who specified a voting policy with a **50% approval threshold**, a **greater than** comparator, and a proposal duration of 24 hours.

The following flow diagrams depict the possible outcomes of a proposal using this example:

- Approved with Full Vote (p. 47)
- Approved with Partial Vote (p. 48)
- Rejected with Full Vote (p. 48)
- Rejected with Partial Vote (p. 48)
- Expired, Does Not Pass (p. 49)

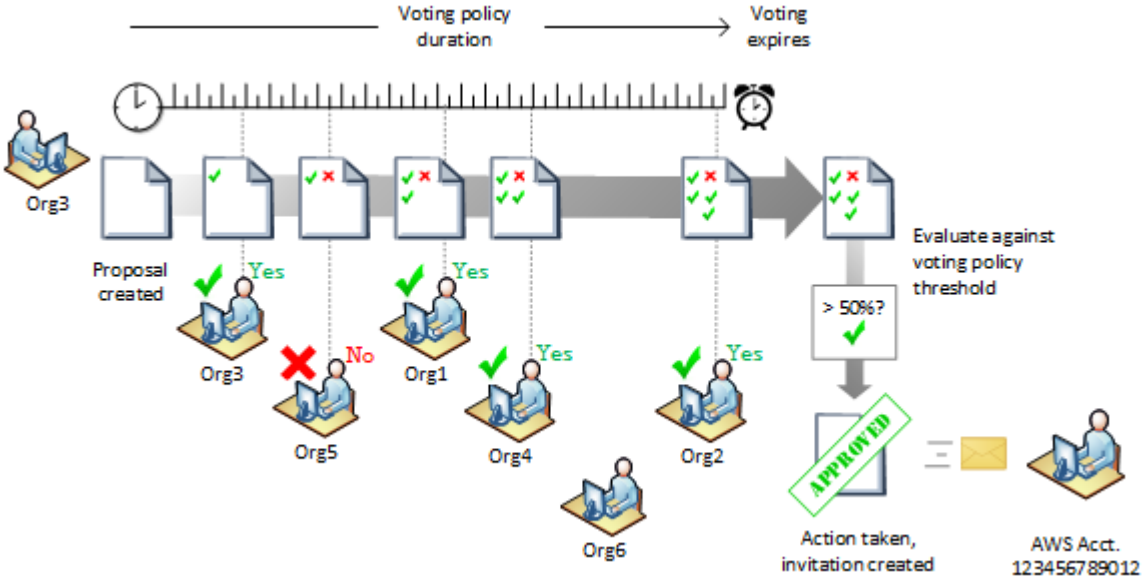
Example – Proposal approved with full member vote

For the following proposal, all members cast a vote before the duration expired. The proposal is `APPROVED`, and an invitation is extended to the AWS account.



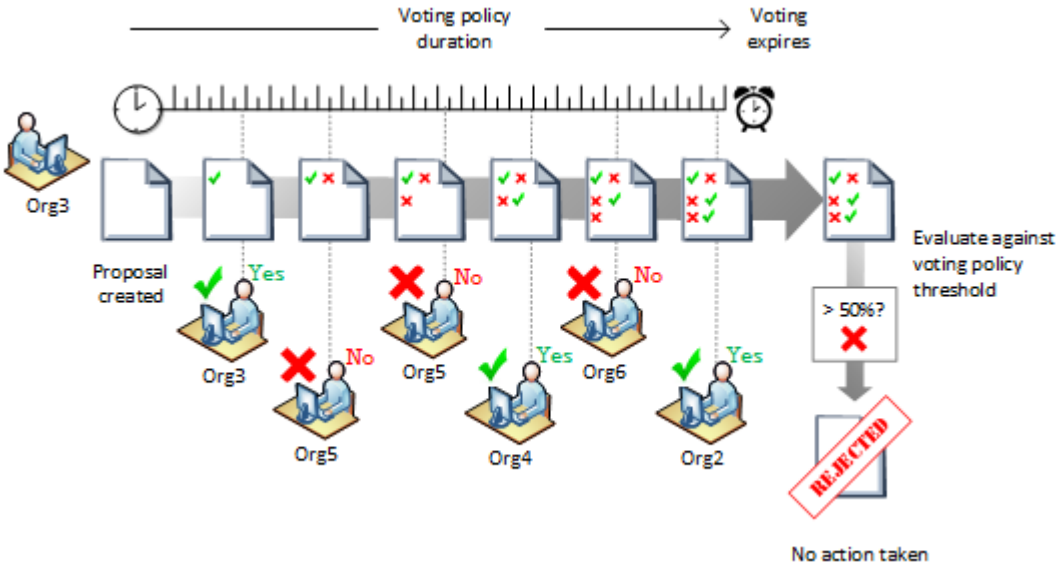
Example – Proposal approved with partial member vote

For the following proposal, not all members cast a vote before the duration expired. However, enough Yes votes were cast to approve the proposal according to the voting policy. The proposal is `APPROVED`, and an invitation is extended to the AWS account.



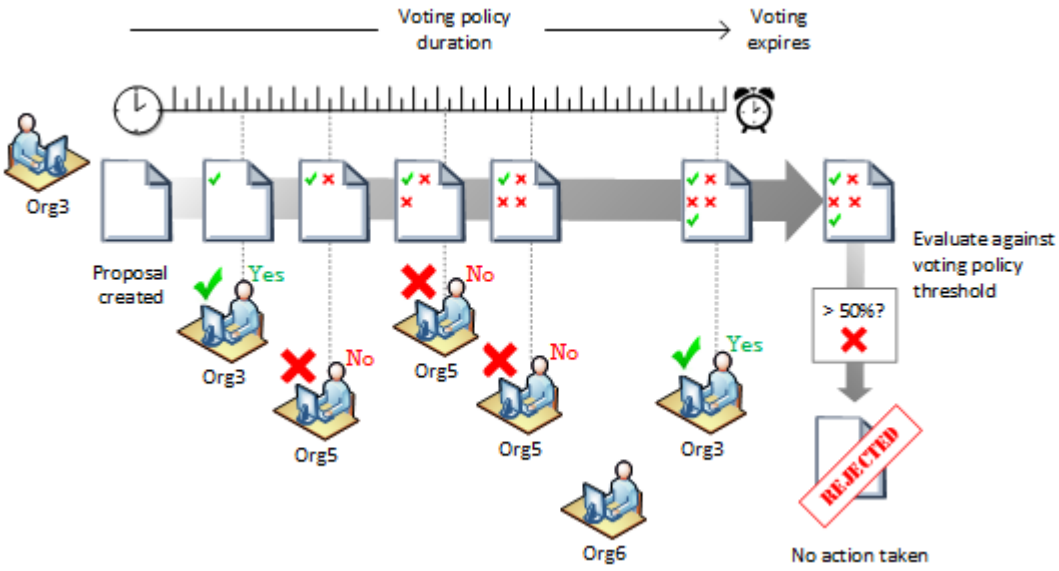
Example – Proposal rejected with full member vote

For the following proposal, all members cast a vote before the duration expired. Because the comparator in the voting policy is **greater than**, a three-to-three vote does not pass the threshold for approval. The proposal is `REJECTED`, and an invitation is not extended to the AWS account.



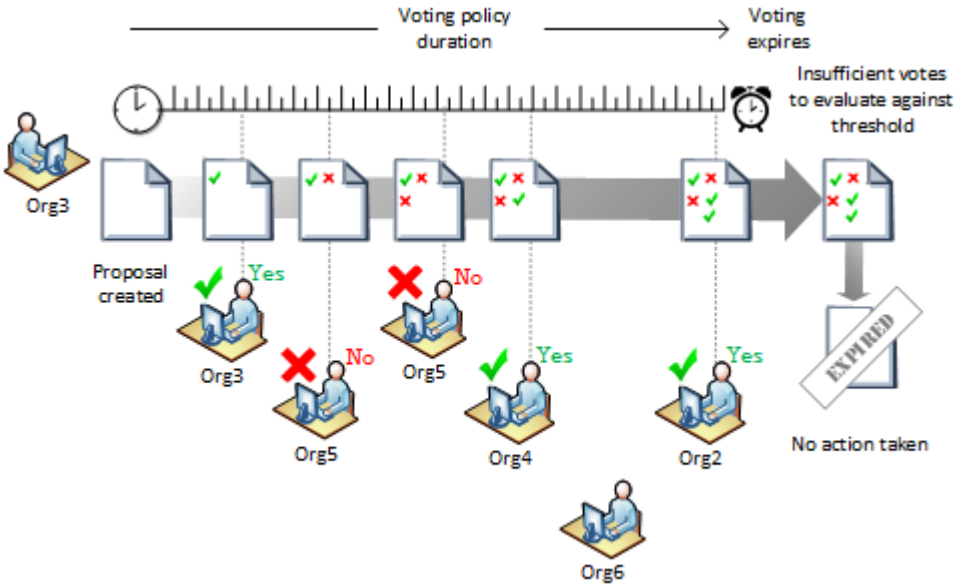
Example – Proposal rejected with partial member vote

For the following proposal, not all members cast a vote before the duration expired. However, enough No votes were cast to reject the proposal according to the voting policy. The proposal is `REJECTED`, and an invitation is extended to the AWS account.



Example – Proposal expires and is not approved

For the following proposal, not all members cast a vote before the duration expired, and neither the number of Yes nor No votes were cast to determine the outcome of the proposal. The proposal is `EXPIRED`, and an invitation is not extended to the AWS account.



View Proposals

All proposals made on a network are shown on the **Proposals** page for a network. Both **Active** proposals and **Completed** proposals are listed. Active proposals are still open for voting. You can also list proposals from the AWS CLI using the `list-proposals` command, or using the ListProposals action with the Managed Blockchain API.

The **Proposals** page for a Network shows both **Active** and **Completed** proposals, listing the **Proposal ID**, the name of the member that created the proposal, and the **Expiration Date (UTC)**, which is the creation time plus the proposal duration specified in the network's voting policy. You can choose a **Proposal ID** to vote on active proposals and to see more detail about any proposal, including the actions proposed and a voting summary by member.

Proposals have one of the following statuses:

- `IN_PROGRESS` - The proposal is active and open for member voting.
- `APPROVED` - The proposal was approved with sufficient `YES` votes among members according to the `VotingPolicy` specified for the `Network`. The specified proposal actions are carried out.
- `REJECTED` - The proposal was rejected with insufficient `YES` votes among members according to the `VotingPolicy` specified for the `Network`. The specified `ProposalActions` are not carried out.
- `EXPIRED` - Members did not cast the number of votes required to determine the proposal outcome before the proposal expired. The specified `ProposalActions` are not carried out.
- `ACTION_FAILED` - One or more of the specified `ProposalActions` in a proposal that was approved could not be completed because of an error. The `ACTION_FAILED` status occurs even if only one proposal action fails and other actions are successful.

To view proposals for a network using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Networks**, choose a network **Name**, and then choose **Proposals**.
3. Choose a **Proposal ID** from the list to view more detailed information, such as the description, a summary of **Actions**, and a **Voting Summary**.
4. Under **Voting Summary**, expand **Votes** to see individual member's votes on the proposal to date.

To view proposals for a network using the AWS CLI

- Enter a command similar to the following example. Replace *n-MWY63ZJZU5HGNCMBQER7IN6OIU* with the network ID for which you want to list proposals.

```
aws managedblockchain list-proposals --network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU
```

The command returns output similar to the following:

```
{
    "Proposals": [
        {
            "CreationDate": 2019-04-08T23:40:20.628Z,
            "Description": "Proposal to add Example Corp. member",
            "ExpirationDate": 2019-04-09T23:40:20.628Z,
            "ProposalId": "p-ZR7KUD2YYNESLNG6RQ33X3FUFE",
            "ProposedByMemberId": "m-J46DNSFRTVCCLONS9DT5TTLS2A",
            "ProposedByMemberName": "org1",
            "Status": "IN_PROGRESS"

        }
    ]
}
```

To view the details of a proposal using the AWS CLI

- Enter a command similar to the following. Replace *n-MWY63ZJZU5HGNCMBQER7IN6OIU* with the network ID and *p-ZR7KUD2YYNESLNG6RQ33X3FUFE* with the proposal ID to view.

```
aws managedblockchain get-proposal --network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU --
proposal-id p-ZR7KUD2YYNESLNG6RQ33X3FUFE
```

The command returns output similar to the following:

```
{
   "Proposal": {
      "Actions": {
         "Invitations": [
            {
               "Principal": "0123456789012"
            }
         ],
      "CreationDate": 2019-04-08T23:40:20.628Z,
      "Description": "Proposal to invite AWS Acct 0123456789012",
      "ExpirationDate": 2019-04-08T23:40:20.628Z,
      "NetworkId": "n-MWY63ZJZU5HGNCMBQER7IN6OIU",
      "NoVoteCount": 1,
      "OutstandingVoteCount": 3,
      "ProposalId": "p-ZR7KUD2YYNESLNG6RQ33X3FUFE",
      "ProposedByMemberId": "m-J46DNSFRTVCCLONS9DT5TTLS2A",
      "ProposedByMemberName": "org1",
      "Status": "IN_PROGRESS",
      "YesVoteCount": 2
   }
}
```

Vote on a Proposal

You can use the AWS Management Console, the AWS CLI `vote-on-proposal` command, or the VoteOnProposal action of the Managed Blockchain API to vote Yes or No on an active proposal. You cannot change a vote after you make it.

To vote on a proposal using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. Choose **Networks**, choose a network **Name**, and then choose **Proposals**.
3. From the list of **Active** proposals, choose a **Proposal ID**.
4. Under **Vote on proposal**, choose the member to vote as from the list, and then choose **Yes** or **No**.
5. When prompted, choose **Confirm**.

To vote on a proposal using the AWS CLI

- Use the `vote-on-proposal` command as shown in the following example. Replace the values of `--network-id`, `--member-id`, and `--vote` as appropriate.

```
aws managedblockchain vote-on-proposal --network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU --
member-id m-K46ICRRXJRCGRNNS4ES4XUUS5A --vote YES
```

Create an Invitation Proposal

You can use the AWS Management Console, the AWS CLI, or the Managed Blockchain API to create an invitation proposal.

To create an invitation proposal using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. From the navigation pane, choose **Networks**, and then choose the network to which you want to invite an AWS account.
3. Choose **Proposals** and then choose **Propose invitation**.
4. For **Submit proposal as**, choose the member in your account that submits the proposal.

   Note
   The member who submits the proposal must also vote on it. A Yes vote is not automatically assumed.
5. Enter an optional **Description** . The description appears to other members. It's a good way to communicate key points or a reminder about the proposal before they vote.
6. For each AWS account that you want to invite, enter the account number in the space provided. Choose **Add** to enter additional accounts.

To create an invitation proposal using the AWS CLI

- Type a command similar to the following. Replace the value of `Principal` with the AWS account ID that you want to invite. Replace the value of `--member-id` with the value for the member in your account that submits the proposal.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain create-proposal \
--actions Invitations=[{Principal=123456789012}] \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-K46ICRRXJRCGRNNS4ES4XUUS5A
```

The command returns the proposal ID, as shown in the following example:

```
{
    "ProposalId": "p-ZR7KUD2YYNESLNG6RQ33X3FUFE"
}
```

Create a Proposal to Remove a Network Member

To create a proposal to remove a member using the AWS Management Console

1. Open the Managed Blockchain console at https://console.aws.amazon.com/managedblockchain/.
2. From the navigation pane, choose **Networks**, and then choose the network.
3. Choose **Proposals** and then choose **Propose removal**.
4. For **Submit proposal as**, choose the member in your account that submits the proposal.

Note
The member who submits the proposal must also vote on it. A Yes vote is not automatically assumed.

5. Enter an optional **Description** . The description appears to other members. It's a good way to communicate key points or a reminder about the proposal before they vote.
6. For each member that you want to remove, enter the member ID in the space provided. Choose **Add** to enter additional members.

To create a removal proposal using the AWS CLI

- Type a command similar to the following. Replace the value of `Principal` with the AWS account ID that you want to invite. Replace the value of `--member-id` with the value for the member in your account that submits the proposal.

```
[ec2-user@ip-192-0-2-17 ~]$ aws managedblockchain create-proposal \
--actions Removals=[{MemberID=m-K46ICRRXJRCGRNNS4ES4XUUS5A}] \
--network-id n-MWY63ZJZU5HGNCMBQER7IN6OIU \
--member-id m-J46DNSFRTVCCLONS9DT5TTLS2A
```

The command returns the proposal ID, as shown in the following example:

```
{
    "ProposalId": "p-ZR7KUD2YYNESLNG6RQ33X3FUFE"
}
```

Automating Managed Blockchain Proposal Notifications with CloudWatch Events

Amazon CloudWatch Events enables you to automate your AWS services and respond automatically to system events. Events from AWS services are delivered to CloudWatch Events in near real time. You can write simple rules to indicate which events are of interest to you, and what automated actions to take when an event matches a rule. With Managed Blockchain, you can monitor CloudWatch Events events to respond to proposals, including invitations sent to your AWS account to join a network, and notification that proposals are `APPROVED` or `REJECTED`. Some examples include notifying an Amazon SNS topic or an AWS SMS queue when an invitation is sent or when a proposal made by a member in your account changes status.

For more information, see the Amazon CloudWatch Events User Guide.

Example Managed Blockchain Events

AWS Account Received an Invitation Event

The `detail-type` of these messages is `Managed Blockchain Invitation State Change`.

```
{
    "version": "0",
    "id": "abcd1234-eeee-4321-a1a2-123456789012",
    "detail-type": "Managed Blockchain Invitation State Change",
    "source": "aws.managedblockchain",
    "account": "123456789012",
```

```
    "time": "2019-04-08T23:40:20.628Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "invitationId": "i-XL9MDD6LVWWDNA9FF94Y4TFTE",
        "networkId": "n-MWY63ZJZU5HGNCMBQER7IN6OIU",
        "networkName": "ExampleCorpNetwork",
        "status": "PENDING",
        "expirationDate": "2019-04-09T23:40:20.628Z",
        "message": "You have received invitation i-XL9MDD6LVWWDNA9FF94Y4TFTE for Amazon
 Managed Blockchain Network n-MWY63ZJZU5HGNCMBQER7IN6OIU and it will expire at 2016-12-16
 20:42 UTC."
    }
}
```

Proposal State Change Event

The `detail-type` of these messages is `Managed Blockchain Proposal State Change`. The following example shows an event for a proposal that changed state to `APPROVED`.

```
{
    "version": "0",
    "id": "abcd1234-eeee-4321-a1a2-123456789012",
    "detail-type": "Managed Blockchain Proposal State Change",
    "source": "aws.managedblockchain",
    "account": "123456789012",
    "time": "2019-04-08T23:40:20.628Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "proposalId": "p-ZR7KUD2YYNESLNG6RQ33X3FUFE",
        "networkId": "n-MWY63ZJZU5HGNCMBQER7IN6OIU",
        "status": "APPROVED",
        "proposedByMemberId": "m-K46ICRRXJRCGRNNS4ES4XUUS5A",
        "proposedByMemberName": "NetworkMember1",
        "expirationDate": "2019-04-09T23:40:20.628Z",
        "description": "Proposal to remove AnyCompany from supply chain blockchain
 network.",
        "message": "Voting on proposal p-ZR7KUD2YYNESLNG6RQ33X3FUFE in Amazon Managed
 Blockchain Network n-MWY63ZJZU5HGNCMBQER7IN6OIU completed at 2016-19-16T20:10:50Z UTC and
 the proposal was approved."
    }
}
```

Work with Hyperledger Fabric

You access services and applications in the Managed Blockchain network from a blockchain framework client. The framework client runs tools and applications that you install for the blockchain framework version that you run on the Managed Blockchain network.

The client accesses Managed Blockchain network resource endpoints using an interface VPC endpoint that you set up in your account. For more information, see Create an Interface VPC Endpoint to Connect to Managed Blockchain Network Resources (p. 45). The client must have access to the interface VPC endpoint.

You can get the endpoints that networks, members, and clients make available using the AWS Management Console, or using `get` commands and actions with the AWS CLI or Managed Blockchain SDK. The available endpoints depend on the blockchain framework and may vary from client to client.

An AWS CloudFormation template to create a Hyperledger Fabric client is available in amazon-managed-blockchain-client-templates repository on Github. For more information, see the readme.md in that repository. For more information about using AWS CloudFormation, see Getting Started in the *AWS CloudFormation User Guide*.

Topics

Register and Enroll a User as an Administrator

When you create a member in a Managed Blockchain network, you specify the first user as an administrator. Managed Blockchain registers this user automatically with the Hyperledger Fabric CA as a *bootstrap identity*. You can then use this bootstrap identity to enroll the identity. Enrolling an identity sends the CA a Certificate Signing Request (CSR) so that the CA can validate that the identity is registered and otherwise valid. The CA then returns a signed certificate if the identity is valid. For more information, see the Fabric CA User's Guide.

To register an identity, you must have the following:

- The member CA endpoint
- The user name and password of an identity with permission to register identities with the member CA
- A valid certificate file and the path to the MSP directory of the identity that will register the new administrator

The following example command uses an existing user admin with a user name of `AdminUser` and a password of `Password123` in a member named org1 to register a new admin with a user name of `AdminUser2` and a password of `Password456`. The certificate file is saved to `/home/ec2-user/managedblockchain-tls-chain.pem` and the MSP directory for the AdminUser identity is `/home/ec2-user/admin-msp`.

The command specifies the `--id.attr 'hf.Admin=true'` option to register the user identity as an admin for the member.

```
fabric-ca-client register --url https://AdminUser:Password123@ca.m-
K46ICRRXJRCGRNNS4ES4XUUS5A.n-MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-
east-1.amazonaws.com:30002 \
```

```
--id.name AdminUser2 --id.secret Password456 \
--id.type user --id.affiliation org1 \
--id.attrs 'hf.Admin=true' --tls.certfiles /home/ec2-user/managedblockchain-tls-chain.pem \
--mspdir /home/ec2-user/admin-msp
```

After the user identity is registered as an admin, use the `fabric-ca-client enroll` command to enroll the new administrator as shown in the following example:

```
fabric-ca-client enroll \
-u https://AdminUser2:Password456@ca.m-K46ICRRXJRCGRNNS4ES4XUUS5A.n-
MWY63ZJZU5HGNCMBQER7IN6OIU.managedblockchain.us-east-1.amazonaws.com:30002 \
--tls.certfiles /home/ec2-user/managedblockchain-tls-chain.pem \
-M /home/ec2-user/admin-msp
```

Develop Chaincode

Smart contracts in Hyperledger Fabric are known as *chaincode*.

- For a conceptual overview of chaincode, see Smart Contracts and Developing Applications in the Hyperledger Fabric documentation.
- For links to Hyperledger Fabric SDKs, see Getting Started in the Hyperledger Fabric documentation.

Considerations and Limitations When Writing Chaincode for Managed Blockchain

- Managed Blockchain peer nodes use version 1.2 of the fabric-shim library to provide the low-level chaincode interface between applications, peers, and the Hyperledger Fabric system for chaincode applications using Node.js. All chaincode has a dependency on this library. Dependencies on other versions or other library packages are not supported because peer nodes currently do not have access to the NPM repository.
- The default limit for the size of a transaction payload is 1MB. To request a limit increase, create a case using the AWS Support Center.

Amazon Managed Blockchain Security

To provide data protection as well as authentication and access control, Amazon Managed Blockchain benefits from AWS features and the features of the open-source framework running on Managed Blockchain.

Topics

Data Protection for Amazon Managed Blockchain

Data encryption helps prevent unauthorized users from reading data from a blockchain network and the associated data storage systems. This includes data saved to persistent media, known as *data at rest*, and data that may be intercepted as it travels the network, known as *data in transit*.

Encryption at Rest

Amazon Managed Blockchain offers fully managed encryption at rest. Managed Blockchain encryption at rest provides enhanced security by encrypting all data at rest on peer nodes using Managed Blockchain owned encryption keys in AWS Key Management Service (AWS KMS). This functionality helps reduce the operational burden and complexity involved in protecting sensitive data. With encryption at rest, you can build security-sensitive blockchain applications that meet strict encryption compliance and regulatory requirements.

Encryption at rest integrates with AWS KMS for managing the encryption key that is used to encrypt your tables. A Managed Blockchain owned key is used to encrypt data at rest by default at no additional cost. No configuration is required. Using an AWS managed encryption key is not supported. For more information, see AWS owned CMKs in the *AWS Key Management Service Developer Guide*.

Encryption in Transit

The Hyperledger Fabric certificate authority (CA) in each membership provides a TLS certificate authority to secure Hyperledger Fabric communication channels in the network. For more information, see the Fabric CA's User Guide in Hyperledger Fabric documentation.

Authentication and Access Control

AWS Identity and Access Management (IAM) permission policies, VPC endpoint services powered by AWS PrivateLink, and Amazon EC2 security groups provide the primary means for you to control access to Amazon Managed Blockchain. In addition to these AWS services, open-source frameworks that run on Managed Blockchain have authentication and access control features that you can configure.

IAM permission policies are associated with AWS users in your account and determine who has access to what. Permission policies specify the actions that each user can perform using Managed Blockchain and other AWS services. VPC endpoint services allow each Managed Blockchain network member to connect privately to Managed Blockchain resources. Amazon EC2 security groups act as virtual firewalls and determine the inbound and outbound network traffic that is allowed between Managed Blockchain resources and other Amazon EC2 resources. In Managed Blockchain, these security groups are associated with the VPC endpoint in your account and with any framework clients that run on AWS, such as a Hyperledger Fabric client running on an Amazon EC2 instance.

Before you configure authentication and access control using AWS services and open-source features, we recommend that you review the following resources:

- For more information about IAM and IAM permission policies, see Identity and Access Management for Amazon Managed Blockchain (p. 58). We also recommend What is IAM? and IAM JSON Policy Reference in the *IAM User Guide*.
- For more information about VPC endpoints, see Create an Interface VPC Endpoint to Connect to Managed Blockchain Network Resources (p. 45) and VPC Endpoints in the *Amazon VPC User Guide*.
- For more information about Amazon EC2 security groups, see Configuring Security Groups (p. 67) and Amazon EC2 Security Groups for Linux Instances in the *Amazon EC2 User Guide for Linux Instances*.
- For more information about the Hyperledger Fabric Certificate Authority (CA), see Certificate Authority (CA) Setup in the Hyperledger Fabric documentation.
- For more information about Hyperledger Fabric application access control lists, see Application Access Control Lists in the Hyperledger Fabric documentation.

Identity and Access Management for Amazon Managed Blockchain

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Managed Blockchain resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in Managed Blockchain.

Service user – If you use the Managed Blockchain service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Managed Blockchain features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Managed Blockchain, see Troubleshooting Amazon Managed Blockchain Identity and Access (p. 66).

Service administrator – If you're in charge of Managed Blockchain resources at your company, you probably have full access to Managed Blockchain. It's your job to determine which Managed Blockchain features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Managed Blockchain, see How Amazon Managed Blockchain Works with IAM (p. 62).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Managed Blockchain. To view example Managed Blockchain identity-based policies that you can use in IAM, see Amazon Managed Blockchain Identity-Based Policy Examples (p. 64).

Authenticating With Identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see The IAM Console and Sign-in Page in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.

AWS Account Root User

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM Users and Groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing Access Keys for IAM Users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to

manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to Create an IAM User (Instead of a Role) in the *IAM User Guide*.

IAM Roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM Roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.
- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in the *IAM User Guide*.

To learn whether to use IAM roles, see When to Create an IAM Role (Instead of a User) in the *IAM User Guide*.

Managing Access Using Policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions.

AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON Policies in the *IAM User Guide*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM Policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing Between Managed Policies and Inline Policies in the *IAM User Guide*.

Resource-Based Policies

Resource-based policies are JSON policy documents that you attach to a resource such as an Amazon S3 bucket. Service administrators can use these policies to define what actions a specified principal (account member, user, or role) can perform on that resource and under what conditions. Resource-based policies are inline policies. There are no managed resource-based policies.

Access Control Lists (ACLs)

Access control policies (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they are the only policy type that does not use the JSON policy document format. Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access Control List (ACL) Overview in the *Amazon Simple Storage Service Developer Guide*.

Other Policy Types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions Boundaries for IAM Entities in the *IAM User Guide*.

- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs Work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session Policies in the *IAM User Guide*.

Multiple Policy Types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy Evaluation Logic in the *IAM User Guide*.

How Amazon Managed Blockchain Works with IAM

Before you use IAM to manage access to Managed Blockchain, you should understand what IAM features are available to use with Managed Blockchain. To get a high-level view of how Managed Blockchain and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

Topics

Managed Blockchain Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Managed Blockchain supports specific actions and resources. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

Actions

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in Managed Blockchain use the following prefix before the action: `managedblockchain:`. For example, to grant someone permission to vote on a proposal with the Managed Blockchain `VoteOnProposal` API operation, you include the `managedblockchain:VoteOnProposal` action in their policy. Policy statements must include either an `Action` or `NotAction` element. Managed Blockchain defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
      "managedblockchain:action1",
      "managedblockchain:action2"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `List`, include the following action:

```
"Action": "managedblockchain:List*"
```

To see a list of Managed Blockchain actions, see Actions Defined by Amazon Managed Blockchain in the *IAM User Guide*.

Resources

The `Resource` element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

Managed Blockchain resource types that can be used in IAM permission policy statements include the following:

- network
- member
- node
- proposal
- invitation

Members, nodes, and invitations are associated with your account. Networks and proposals, on the other hand, are scoped to the entire blockchain network and are not associated with a particular account.

For example an Managed Blockchain network resource has the following ARN:

```
arn:${Partition}:managedblockchain:${Region}::networks/${NetworkId}
```

For example, to specify the `n-MWY63ZJZU5HGNCMBQER7IN6OIU` network in your statement, use the following ARN:

```
"Resource": "arn:aws:managedblockchain:us-east-1::networks/n-MWY63ZJZU5HGNCMBQER7IN6OIU"
```

To specify any network that is visible to your account, use the wildcard (*):

```
"Resource": "arn:aws:managedblockchain:us-east-1::networks/*"
```

Some Managed Blockchain actions, such as `CreateNetwork`, `ListInvitations`, and `ListNetworks` cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

```
"Resource": "*"
```

To see a list of Managed Blockchain resource types and their ARNs, see Resources Defined by Amazon Managed Blockchain in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon Managed Blockchain.

Condition Keys

Managed Blockchain does not provide any service-specific condition keys, but it does support using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

Examples

To view examples of Managed Blockchain identity-based policies, see Amazon Managed Blockchain Identity-Based Policy Examples (p. 64).

Managed Blockchain Resource-Based Policies

Managed Blockchain does not support resource-based policies.

Authorization Based on Managed Blockchain Tags

Managed Blockchain does not support tagging resources or controlling access based on tags.

Amazon Managed Blockchain Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify Managed Blockchain resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

Topics

Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Managed Blockchain resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** – To start using Managed Blockchain quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get Started Using Permissions With AWS Managed Policies in the *IAM User Guide*.
- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to

specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

Allow Users to View Their Own Permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "ViewOwnUserInfo",
               "Effect": "Allow",
               "Action": [
                   "iam:GetUserPolicy",
                   "iam:ListGroupsForUser",
                   "iam:ListAttachedUserPolicies",
                   "iam:ListUserPolicies",
                   "iam:GetUser"
               ],
               "Resource": [
                   "arn:aws:iam::*:user/${aws:username}"
               ]
           },
           {
               "Sid": "NavigateInConsole",
               "Effect": "Allow",
               "Action": [
                   "iam:GetGroupPolicy",
                   "iam:GetPolicyVersion",
                   "iam:GetPolicy",
                   "iam:ListAttachedGroupPolicies",
                   "iam:ListGroupPolicies",
                   "iam:ListPolicyVersions",
                   "iam:ListPolicies",
                   "iam:ListUsers"
               ],
               "Resource": "*"
           }
       ]
   }
```

Using the Managed Blockchain Console

To access the Amazon Managed Blockchain console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Managed Blockchain resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy.

To ensure that those entities can still use the Managed Blockchain console, also attach the following AWS managed policy to the entities.

```
AmazonManagedBlockchainConsole FullAccess
```

For more information, see Adding Permissions to a User in the *IAM User Guide*.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

Performing All Managed Blockchain Actions on All Accessible Managed Blockchain Networks for an AWS Account

This example grants an IAM user in your AWS account access to all network and member resources in your account in the `us-east-1` Region for the AWS account `123456789012`. This includes the ability to create new networks, reject invitations to join other networks, and join other networks by creating a member.

```
{
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "ManageNetworkResources",
   "Effect": "Allow",
   "Action": [
    "managedblockchain:CreateProposal",
    "managedblockchain:GetProposal",
    "managedblockchain:DeleteMember",
    "managedblockchain:VoteOnProposal",
    "managedblockchain:ListProposals",
    "managedblockchain:GetNetwork",
    "managedblockchain:ListMembers",
    "managedblockchain:ListProposalVotes",
    "managedblockchain:RejectInvitation",
    "managedblockchain:GetNode",
    "managedblockchain:GetMember",
    "managedblockchain:DeleteNode",
    "managedblockchain:CreateNode",
    "managedblockchain:CreateMember",
    "managedblockchain:ListNodes"
   ],
   "Resource": [
    "arn:aws:managedblockchain:us-east-1::networks/*",
    "arn:aws:managedblockchain:us-east-1::proposals/*",
    "arn:aws:managedblockchain:us-east-1:123456789012:members/*",
    "arn:aws:managedblockchain:us-east-1:123456789012:invitations/*",
    "arn:aws:managedblockchain:us-east-1:123456789012:nodes/*"
   ]
  },
  {
   "Sid": "WorkWithNetworksForAcct",
   "Effect": "Allow",
   "Action": [
    "managedblockchain:ListNetworks",
    "managedblockchain:ListInvitations",
    "managedblockchain:CreateNetwork"
   ],
   "Resource": "*"
  }
 ]
}
```

Troubleshooting Amazon Managed Blockchain Identity and Access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Managed Blockchain and IAM.

Topics

I Am Not Authorized to Perform an Action in Managed Blockchain

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a *`widget`* but does not have `managedblockchain:`*`CreateMember`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 managedblockchain:CreateMember on resource: n-MWY63ZJZU5HGNCMBQER7IN6OIU
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`n-MWY63ZJZU5HGNCMBQER7IN6OIU`* resource using the `managedblockchain:`*`CreateMember`* action.

I Want to View My Access Keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing Access Keys in the *IAM User Guide*.

I'm an Administrator and Want to Allow Others to Access Managed Blockchain

To allow others to access Managed Blockchain, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Managed Blockchain.

To get started right away, see Creating Your First IAM Delegated User and Group in the *IAM User Guide*.

Configuring Security Groups

Security groups act as virtual firewalls. They control inbound and outbound traffic between your Hyperledger Fabric client and Managed Blockchain network resources through the VPC endpoint in your

account. By default, security group rules are restrictive, so you must add rules that allow traffic for any resources, such as client computers, that must access the Managed Blockchain network. The following tables list the minimum required security group rules that must be associated with the VPC endpoint and the Hyperledger Fabric client.

VPC Endpoint Security Group Minimum Rules

Inbound/Outbound	Type	Source/Destination	Purpose
Outbound	All traffic	0.0.0/0 (Anywhere)	Default. Allows unrestricted outbound traffic from the interface VPC endpoint to all recipients.
Inbound	Custom TCP, Port for Ordering Service (ranging between 30000 and 34000)—for example, 30001. The port is available within the **Ordering service endpoint** on the network details page using the console and returned within the `OrderingServiceEndpoint` property using the `get-network` command from the AWS CLI or using the `GetNetwork` API action.	The IPv4 address, an address range, or a security group that includes all members' Hyperledger Fabric clients.	Allows the Hyperledger Fabric ordering service to receive traffic from Hyperledger Fabric clients.
Inbound	Custom TCP, Port for the CA Service for a member (ranging between 30000 and 34000)—for example, 30002. This is unique to each member, and each member only needs access to their own CA. The port is available within the **Fabric certificate authority endpoint** on the member details page using the console and returned within the `CaEndpoint` property using the `get-member` command from the AWS CLI or using the `GetMember` API action.	The IPv4 address, an address range, or a security group that includes all members' Hyperledger Fabric clients.	Allows the Hyperledger Fabric certificate authority (CA) for each member to receive traffic from respective Hyperledger Fabric clients.
Inbound	Custom TCP, Ports, or Range of Ports for	The IPv4 address, an address range, or a	Allows the network to receive traffic from

Inbound/Outbound	Type	Source/Destination	Purpose
	Peer Event Services on Peer Nodes (ranging between 30000 and 34000). The port is available within the **Peer node endpoints** on the member details page using the console and returned as the `PeerEventPort` property using the `get-node` command from the AWS CLI or using the `GetNode` API action.	security group that includes all members' Hyperledger Fabric clients.	peer nodes as required. Each node in each membership has a unique port associated with its peer event service. Any node that might be a participant in an endorsement policy, regardless of membership, must be allowed communications in order to endorse transactions.

Hyperledger Fabric Client Security Group Minimum Rules

Inbound/Outbound	Type	Source/Destination	Purpose
Outbound	All traffic	0.0.0/0 (Anywhere)	Default. Allows unrestricted outbound traffic from the Hyperledger Fabric client to all recipients. If necessary, you can limit the destination to the interface VPC endpoint.
Inbound	SSH (Port 22)	The IP address, address range, or security group that includes trusted SSH clients that connect to the Hyperledger Fabric client.	Allows trusted clients to use SSH to connect to the Hyperledger Fabric client to interact—for example, to query and execute chaincode.

Document History for Amazon Managed Blockchain Management Guide

The following table describes important additions to the Amazon Managed Blockchain Management Guide. For notification about updates to this documentation, you can subscribe to the RSS feed.

update-history-change	update-history-description	update-history-date
Major updates for new proposal and voting work flow for member invitations and removals	Updated Getting Started tutorial, conceptual information, and procedures for new voting proposal design.	April 8, 2019
Added security group configuration guidance	Added prescriptive guidance for configuring security groups for the tutorial. Added references for minimum inbound and outbound security group rules required for Hyperledger Fabric client and interface VPC endpoint for reference and customization.	February 28, 2019
Updates to getting started steps	Removed redundant steps in 3.2. The step to update .bash_profile with path to fabric-ca was already covered in step 3.1.	December 3, 2018
Initial release of Amazon Managed Blockchain (Preview)	Initial documentation for Amazon Managed Blockchain.	November 28, 2018

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.