aws

Developer Guide

AWS Blockchain Templates

AWS Blockchain Templates: Developer Guide

Table of Contents

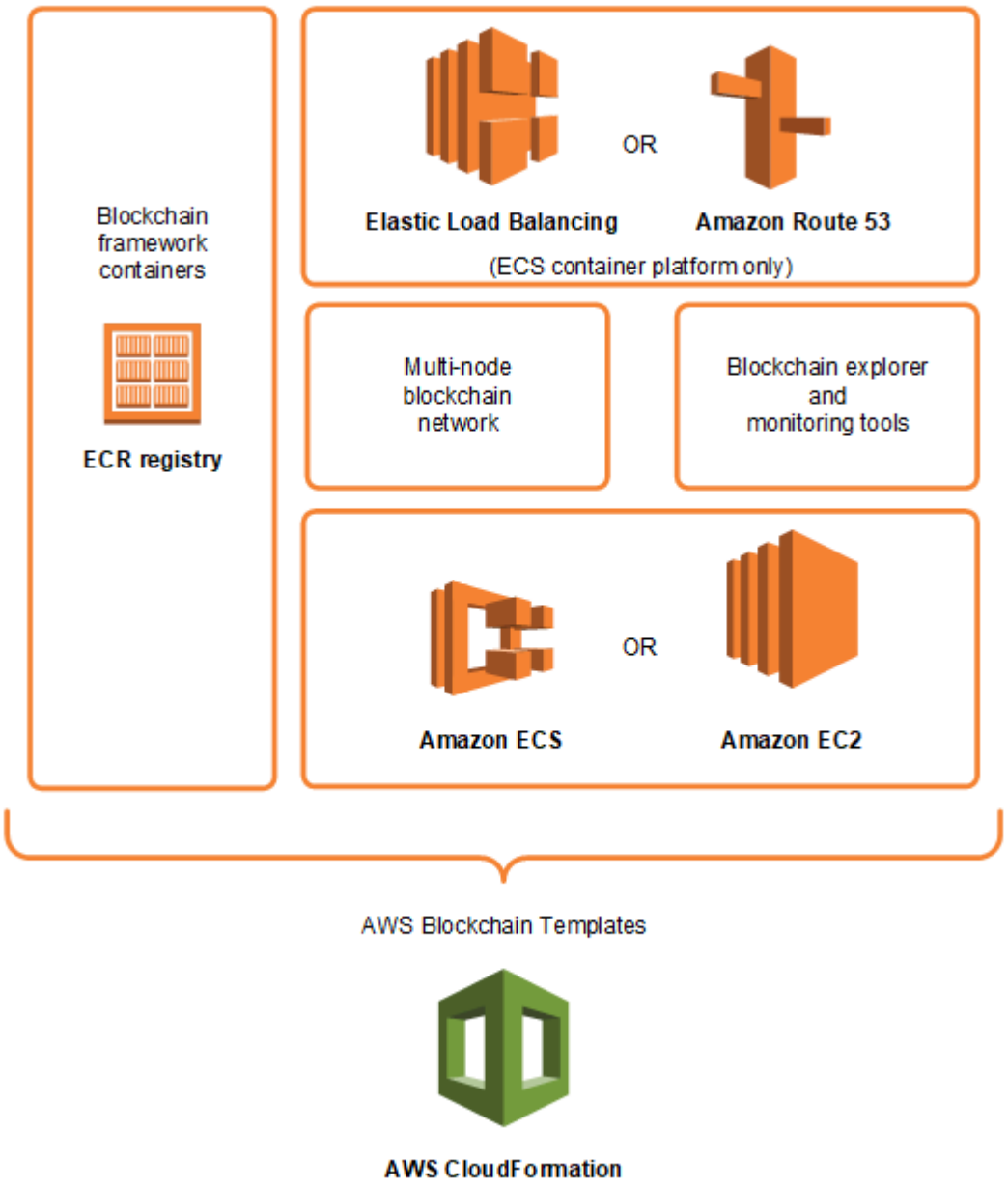
AWS Blockchain Templates was discontinued on April 30, 2019. No further updates to this service or this supporting documentation will be made. For the best Managed Blockchain experience on AWS, we recommend that you use Amazon Managed Blockchain (AMB). To learn more about getting started with Amazon Managed Blockchain, see our workshop on Hyperledger Fabric, or our blog on deploying an Ethereum node. If you have questions about AMB or require further support, contact Support or your AWS account team.

What Is AWS Blockchain Templates?

AWS Blockchain Templates helps you quickly create and deploy blockchain networks on AWS using different blockchain frameworks. Blockchain is a decentralized database technology that maintains a continually growing set of transactions and smart contracts hardened against tampering and revision using cryptography.

A blockchain network is a peer-to-peer network that improves the efficiency and immutability of transactions for business processes like international payments, supply chain management, land registration, crowd funding, governance, financial transactions, and more. This allows people and organizations who may not know one another to trust and independently verify the transaction record.

You use AWS Blockchain Templates to configure and launch AWS CloudFormation stacks to create blockchain networks. The AWS resources and services you use depend on the AWS Blockchain Template you choose and the options that you specify. For information about available templates and their features, see AWS Blockchain Templates and Features. The fundamental components of a blockchain network on AWS created using AWS Blockchain Templates are shown in the following diagram.



How to Get Started

The best place to start depends on your level of expertise with blockchain and AWS—particularly the services related to AWS Blockchain Templates.

I'm proficient with AWS and blockchain

Start with the topic in AWS Blockchain Templates and Features about the framework you want to use. Use the links to launch the AWS Blockchain Template and configure the blockchain network, or download the templates to check them out on your own.

I'm proficient with AWS and new to blockchain

Start with the Getting Started with AWS Blockchain Templates tutorial. This walks you through creating an introductory Ethereum blockchain network with default settings. When you finish, see AWS Blockchain Templates and Features for an overview of blockchain frameworks and links to learn more about configuration choices and features.

I'm a beginner with AWS and proficient with blockchain

Start with Setting Up AWS Blockchain Templates. This helps you get set up with fundamentals on AWS, like an account and a user profile. Next, run through the Getting Started with AWS Blockchain Templates tutorial. This tutorial walks you through creating an introductory Ethereum blockchain network. Even if you won't ultimately use Ethereum, you get hands-on experience setting up related services. This experience is useful for all blockchain frameworks. Finally, see the topic in the AWS Blockchain Templates and Features section for your framework.

I'm new to AWS and blockchain

Start with Setting Up AWS Blockchain Templates. This helps you get set up with fundamentals on AWS, like an account and a user profile. Then run through the Getting Started with AWS Blockchain Templates tutorial. This tutorial walks you through creating an introductory Ethereum blockchain network. Take the time to explore the links to learn more about AWS services and Ethereum.

Related Services

Depending on the options you select, AWS Blockchain Templates can use the following AWS services to deploy blockchain:

- **Amazon EC2**—Provides compute capacity for your blockchain network. For more information, see the Amazon EC2 User Guide.
- **Amazon ECS**—Orchestrates container deployment among EC2 instances in a cluster for your blockchain network, if you choose to use it. For more information, see the Amazon Elastic Container Service Developer Guide.
- **Amazon VPC**—Provides network access for the Ethereum resources that you create. You can customize configuration for accessibility and security. For more information, see the Amazon VPC Developer Guide.
- **Application Load Balancing**—Serves as a single point of contact for access to available user interfaces and internal service discovery when using Amazon ECS as a container platform. For

more information, see What is an Application Load Balancer? in the *User Guide for Application Load Balancers.*.

Setting Up AWS Blockchain Templates

Before you start with AWS Blockchain Templates, complete the following tasks:

- Sign Up for AWS
- Create an IAM User
- Create a Key Pair

These are fundamental prerequisites for all blockchain configurations. In addition, the blockchain network that you choose may have prerequisites, which vary according to your desired environment and configuration choices. For more information, see the relevant section for your blockchain template in AWS Blockchain Templates and Features.

For step-by-step instructions to set up prerequisites for a private Ethereum network using an Amazon ECS cluster, see Getting Started with AWS Blockchain Templates.

Sign Up for AWS

When you sign up for AWS, your AWS account is automatically signed up for all services. You are charged only for the services that you use.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To create 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.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

Note your AWS account number. You need it when you create an IAM user in the next task.

Create an IAM User

Services in AWS require that you provide credentials when you access them, so that the service can determine whether you have permissions to access its resources. The console requires your password. You can create access keys for your AWS account to access the command line interface or API. However, we don't recommend that you access AWS using the credentials for your AWS account; we recommend that you use AWS Identity and Access Management (IAM) instead. Create an IAM user, and then add the user to an IAM group with administrative permissions or grant this user administrative permissions. You can then access AWS using a special URL and the credentials for the IAM user.

If you signed up for AWS but have not created an IAM user for yourself, you can create one using the IAM console. If you already have an IAM user, you can skip this step.

To create an administrator user, choose one of the following options.

Choose one way to manage your administrator	**To**	**By**	**You can also**
In IAM Identity Center (Recommended)	Use short-term credentials to access AWS. This aligns with the security best practices. For information about best practices, see Security best practices in IAM in the *IAM User Guide*.	Following the instructions in Getting started in the *AWS IAM Identity Center User Guide*.	Configure programmatic access by Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*.

Choose one way to manage your administrator	To	By	You can also
In IAM (Not recommended)	Use long-term credentials to access AWS.	Following the instructions in Create an IAM user for emergency access in the *IAM User Guide*.	Configure programmatic access by Manage access keys for IAM users in the *IAM User Guide*.

To sign in as this new IAM user, sign out of the AWS Management Console, then use the following URL, where *your_aws_account_id* is your AWS account number without the hyphens (for example, if your AWS account number is `1234-5678-9012`, your AWS account ID is `123456789012`):

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

Enter the IAM user name and password that you just created. When you're signed in, the navigation bar displays "*your_user_name @ your_aws_account_id*".

If you don't want the URL for your sign-in page to contain your AWS account ID, you can create an account alias. From the IAM dashboard, choose **Create Account Alias** and enter an alias, such as your company name. To sign in after you create an account alias, use the following URL:

```
https://your_account_alias.signin.aws.amazon.com/console/
```

To verify the sign-in link for IAM users for your account, open the IAM console and check under **IAM users sign-in link** on the dashboard.

For more information, see the AWS Identity and Access Management User Guide.

Create a Key Pair

AWS uses public-key cryptography to secure the login information for the instances in a blockchain network. You specify the name of the key pair when you use each AWS Blockchain Template. You can then use the key pair to access instances directly, for example, to log in using SSH.

If you already have a key pair in the right Region, you can skip this step. If you haven't created a key pair already, you can create one using the Amazon EC2 console. Create the key pair in the same Region that you use to launch the Ethereum network. For more information, see Regions and Availability Zones in the *Amazon EC2 User Guide*.

To create a key pair

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the navigation bar, select a Region for the key pair. You can select any Region that's available to you, regardless of your location, but key pairs are specific to a Region. For example, if you plan to launch an instance in the US East (Ohio) region, you must create a key pair for the instance in the same Region.
3. In the navigation pane, choose **Key Pairs**, **Create Key Pair**.
4. For **Key pair name**, enter a name for the new key pair. Choose a name that is easy for you to remember, such as your IAM user name, followed by `-key-pair`, plus the region name. For example, *me*-key-pair-*useast2*. Choose **Create**.
5. The private key file is automatically downloaded by your browser. The base file name is the name that you specified as the name of your key pair, and the file name extension is `.pem`. Save the private key file in a safe place.

> **Important**
>
> This is the only chance for you to save the private key file. You provide the name of your key pair when you launch the Ethereum network.

For more information, see Amazon EC2 Key Pairs in the *Amazon EC2 User Guide*. For more information about connecting to EC2 instances using the key pair, see Connect to Your Linux Instance in the *Amazon EC2 User Guide*.

Getting Started with AWS Blockchain Templates

This tutorial demonstrates how to use the AWS Blockchain Template for Ethereum to create a private blockchain network on AWS through AWS CloudFormation. The network that you create has two Ethereum clients and one miner running on Amazon EC2 instances in an Amazon ECS cluster. Amazon ECS runs these services in Docker containers pulled from Amazon ECR. Before you start this tutorial, it's helpful to know about blockchain networks and the AWS services involved, but not required.

This tutorial assumes that you have set up the general prerequisites covered in Setting Up AWS Blockchain Templates. In addition, you must set up some AWS resources, such as an Amazon VPC network and specific permissions for IAM roles, before you use the template.

The tutorial demonstrates how to set up those prerequisites. We made setup choices, but they are not prescriptive. As long as you meet the prerequisites, you can make other configuration choices based on the needs of your application and environment. For information about the features and general prerequisites for each template, and to download templates or launch them directly in AWS CloudFormation, see AWS Blockchain Templates and Features.

Throughout this tutorial, examples use the US West (Oregon) Region (us-west-2), but you can use any region that supports AWS Blockchain Templates:

- US West (Oregon) Region (us-west-2)
- US East (N. Virginia) Region (us-east-1)
- US East (Ohio) Region (us-east-2)

> **Note**
>
> Running a template in a Region not listed above launches resources in the US East (N. Virginia) Region (us-east-1).

The AWS Blockchain Template for Ethereum that you configure using this tutorial creates the following resources:

- On-Demand EC2 instances of the type and number that you specify. The tutorial uses the default t2.medium instance type.

- An internal Application Load Balancer.

Following the tutorial, steps are provided to clean up resources that you create.

Topics

Set Up Prerequisites

The AWS Blockchain Template for Ethereum configuration that you specify in this tutorial requires that you do the following:

- Create a VPC and Subnets
- Create Security Groups
- Create an IAM Role for Amazon ECS and an EC2 Instance Profile
- Create a Bastion Host

Create a VPC and Subnets

The AWS Blockchain Template for Ethereum launches resources into a virtual network that you define using Amazon Virtual Private Cloud (Amazon VPC). The configuration you specify in this tutorial creates an Application Load Balancer, which requires two public subnets in different Availability Zones. In addition, a private subnet is required for the container instances, and the subnet must be in the same Availability Zone as the Application Load Balancer. You first use the VPC Wizard to create one public subnet and a private subnet in the same Availability Zone. You then create a second public subnet within this VPC in a different Availability Zone.

For more information, see What is Amazon VPC? in the *Amazon VPC User Guide*.

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

To create an Elastic IP address

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

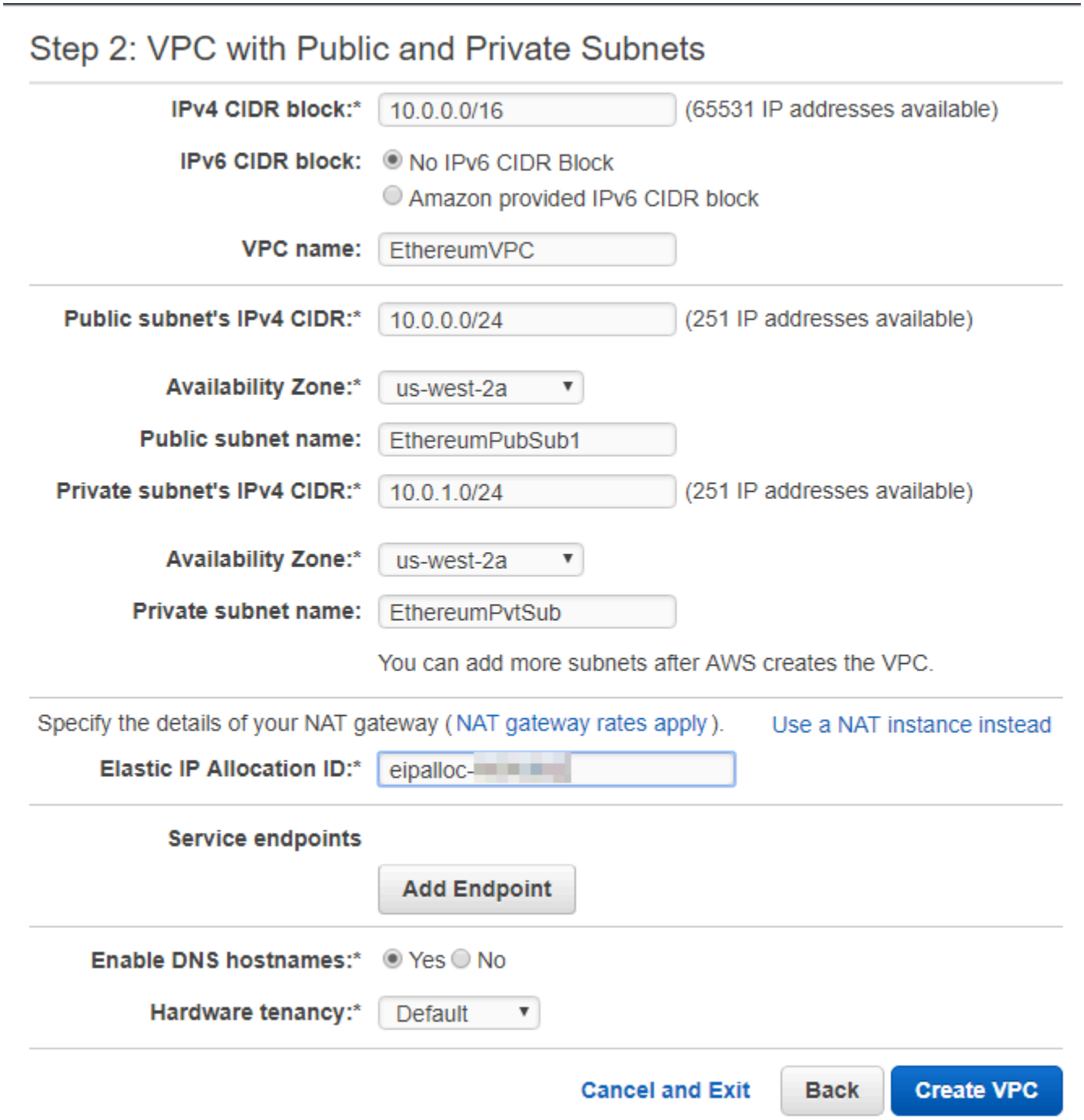
To create the VPC

1. From the navigation bar, select a Region for the VPC. VPCs are specific to a Region, so select the same Region in which you created your key pair in and where you are launching the Ethereum stack. For more information, see Create a Key Pair.
2. On the VPC dashboard, choose **Start VPC Wizard**.
3. On the **Step 1: Select a VPC Configuration** page, choose **VPC with Public and Private Subnets**, **Select**.
4. On the **Step 2: VPC with Public and Private Subnets** page, leave **IPv4 CIDR block** and **IPv6 CIDR block** to their default values. For **VPC name**, enter a friendly name.
5. For **Public subnet's IPv4 CIDR**, leave the default value. For **Availability Zone**, choose a zone. For **Public subnet name**, enter a friendly name.

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

   Note the Availability Zone of this subnet because you select the same Availability Zone for the private subnet, and a different one for the other public subnet.
6. For **Private subnet's IPv4 CIDR**, leave the default value. For **Availability Zone**, select the same Availability Zone as in the previous step. For **Private subnet name**, enter a friendly name.
7. For **Elastic IP Allocation ID**, select the Elastic IP address that you created earlier.
8. Leave the default values for other settings.
9. Choose **Create VPC**.

   The example below shows a VPC **EthereumNetworkVPC** with a public subnet **EthereumPubSub1** and a private subnet **EthereumPvtSub1**. The public subnet uses Availability Zone **us-west-2a**.

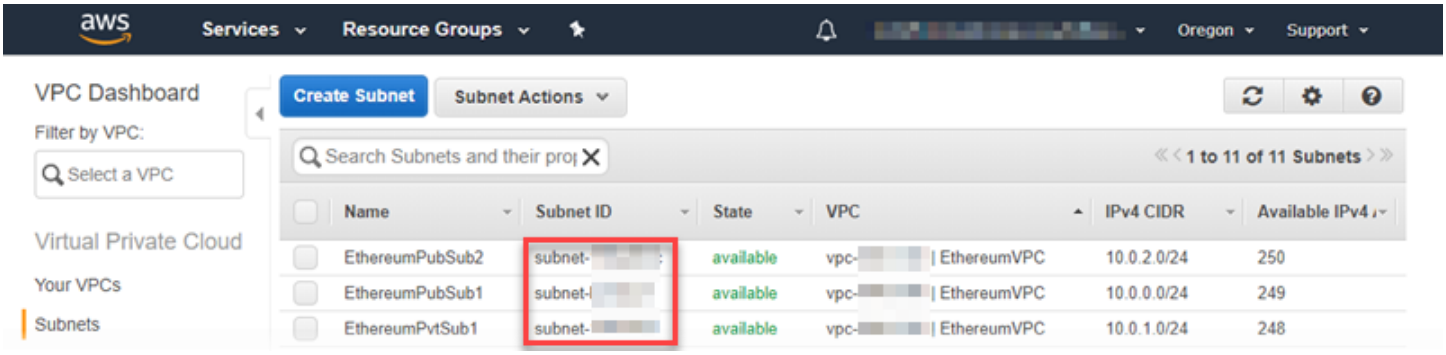


To create the second public subnet in a different Availability Zone

1. Choose **Subnets** and then select the public subnet that you created earlier from the list. Select the **Route Table** tab and note the **Route table** ID. You specify this same route table for the second public subnet below.
2. Choose **Create Subnet**.

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

You should now see three subnets for the VPC that you created earlier. Make a note of the subnet names and IDs so that you can specify them using the template.



Create Security Groups

Security groups act as firewalls, controlling inbound and outbound traffic to resources. When you use the template to create an Ethererum network on an Amazon ECS cluster, you specify two security groups:

- A security group for EC2 instances that controls traffic to and from EC2 instances in the cluster
- A security group for the Application Load Balancer that controls traffic between the Application Load Balancer, EC2 instances, and the bastion host. You associate this security group with the bastion host as well.

Each security group has rules that allow communication between the Application Load Balancer and the EC2 instances, as well as other minimum rules. This requires that the security groups reference one another. For this reason, you first create the security groups and then update them with appropriate rules.

To create two security groups

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Security Groups**, **Create Security Group**.
3. For **Security group name**, enter a name for the security group that's easy to identify and will differentiate it from the other, such as *EthereumEC2-SG* or *EthereumALB-SG*. You use these names later. For **Description**, enter a brief summary.
4. For **VPC**, select the VPC that you created earlier.
5. Choose **Create**.
6. Repeat the steps above to create the other security group.

Add inbound rules to the security group for EC2 instances

1. Select the security group for EC2 instances that you created earlier
2. On the **Inbound** tab, choose **Edit**.
3. For **Type**, choose **All traffic**. For **Source**, leave **Custom** selected, and then choose the security group you are currently editing from the list, for example, *EthereumEC2-SG*. This allows the EC2 instances in the security group to communicate with one another.
4. Choose **Add Rule**.
5. For **Type**, choose **All traffic**. For **Source**, leave **Custom** selected, and then choose the security group for the Application Load Balancer from the list, for example, *EthereumALB-SG*. This allows the EC2 instances in the security group to communicate with the Application Load Balancer.
6. Choose **Save**.

Add inbound and edit outbound rules for the security group for the Application Load Balancer

1. Select the security group for Application Load Balancers that you created earlier
2. On the **Inbound** tab, choose **Edit** and then add the following inbound rules:

a. For **Type**, choose **All traffic**. For **Source**, leave **Custom** selected, and then choose the security group you are currently editing from the list, for example, *EthereumALB-SG*. This allows the Application Load Balancer to communicate with itself and with the bastion host.

b. Choose **Add Rule**.

c. For **Type**, choose **All traffic**. For **Source**, leave **Custom** selected, and then choose the security group for EC2 instances from the list, for example, *EthereumEC2-SG*. This allows the EC2 instances in the security group to communicate with the Application Load Balancer and the bastion host.

d. Choose **Add Rule**.

e. For **Type**, choose **SSH**. For **Source**, select **My IP**, which detects your computer's IP CIDR and enters it.

> **Important**
>
> This rule allows the bastion host to accept SSH traffic from your computer, enabling your computer to use the bastion host to view web interfaces and connect to EC2 instances on the Ethereum network. To allow others to connect to the Ethereum network, add them as sources to this rule. Only allow inbound traffic to trusted sources.

f. Choose **Save**.

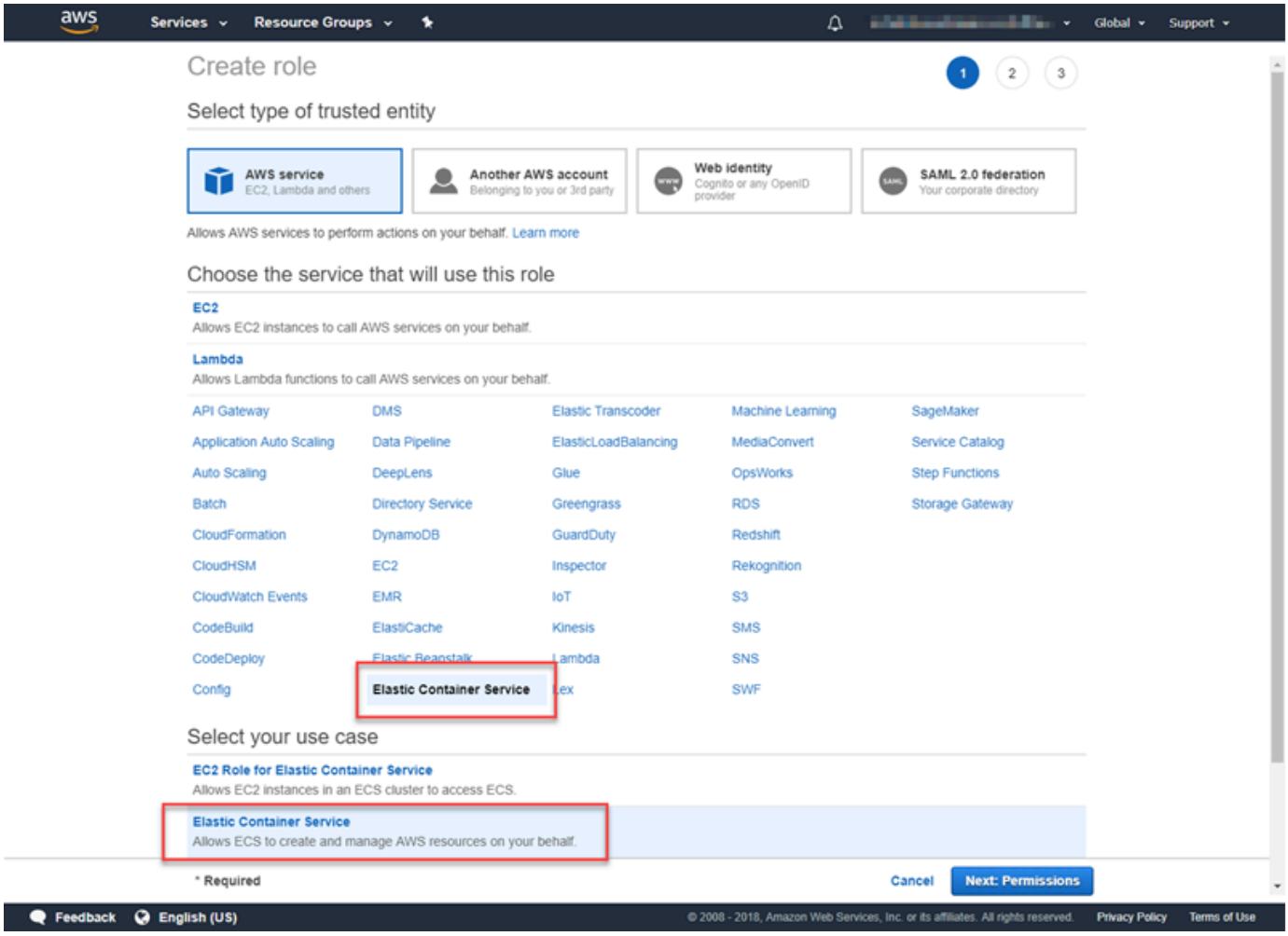
3. On the **Outbound** tab, choose **Edit** and delete the rule that was automatically created to allow outbound traffic to all IP addresses.
4. Choose **Add Rule**.
5. For **Type**, choose **All traffic**. For **Destination**, leave **Custom** selected, and then choose the security group for EC2 instances from the list. This allows outbound connections from the Application Load Balancer and the bastion host to EC2 instances in the Ethereum network.
6. Choose **Add Rule**.
7. For **Type**, choose **All traffic**. For **Destination**, leave **Custom** selected, and then choose the security group you are currently editing from the list, for example, *EthereumALB-SG*. This allows the Application Load Balancer to communicate with itself and with the bastion host.
8. Choose **Save**.

Create an IAM Role for Amazon ECS and an EC2 Instance Profile

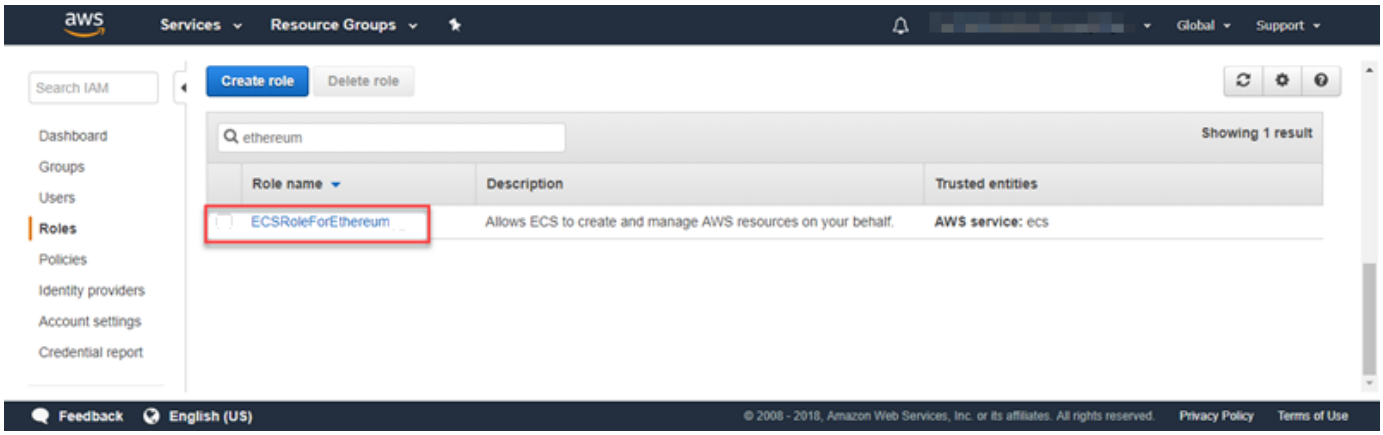
When you use this template, you specify an IAM role for Amazon ECS and an EC2 instance profile. The permissions policies attached to these roles allow the AWS resources and instances in your cluster interact with other AWS resources. For more information, see IAM Roles in the *IAM User Guide*. You set up the IAM role for Amazon ECS and the EC2 instance profile using the IAM console (https://console.aws.amazon.com/iam/).

To create the IAM role for Amazon ECS

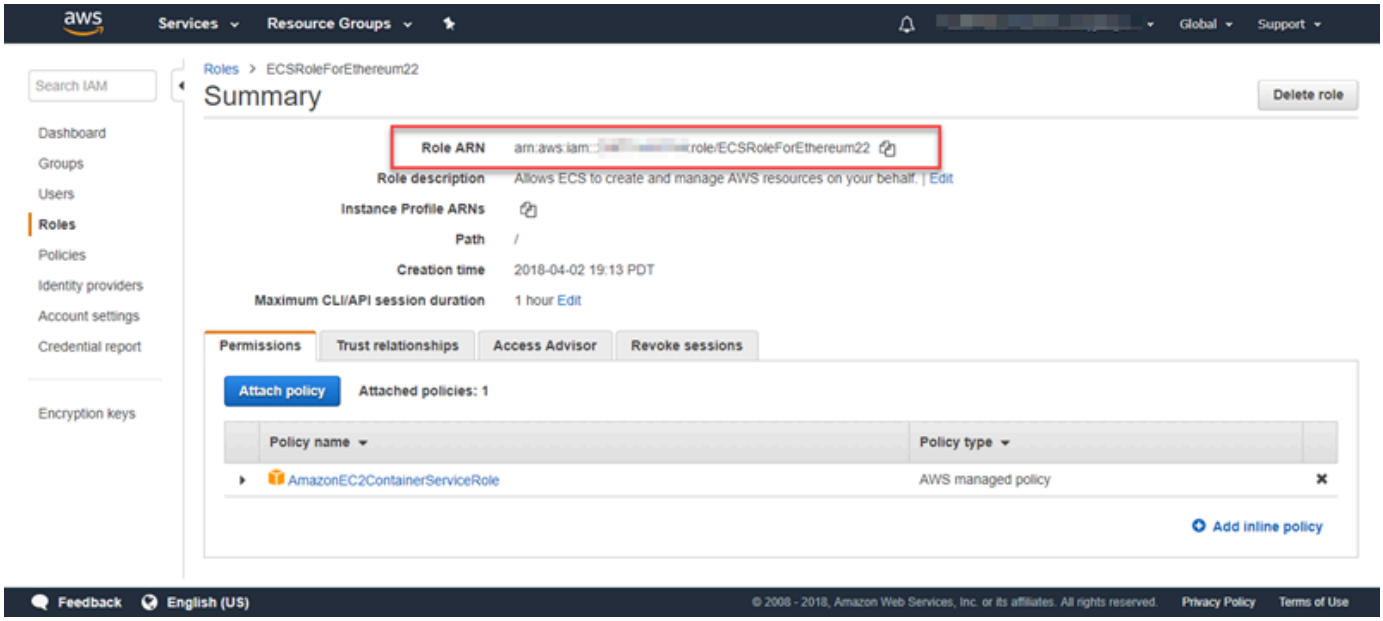
1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**, **Create Role**.
3. Under **Select type of trusted entity**, choose **AWS service**.
4. For **Choose the service that will use this role**, choose **Elastic Container Service**.
5. Under **Select your use case**, choose **Elastic Container Service**, **Next:Permissions**.



6. For **Permissions policy**, leave the default policy (**AmazonEC2ContainerServiceRole**) selected, and choose **Next:Review**.
7. For **Role name**, enter a value that helps you identify the role, such as *ECSRoleForEthereum*. For **Role Description**, enter a brief summary. Note the role name for later.
8. Choose **Create role**.
9. Select the role that you just created from the list. If your account has many roles, you can search for the role name.



10. Copy the **Role ARN** value and save it so that you can copy it again. You need this ARN when you create the Ethereum network.



The EC2 instance profile that you specify in the template is assumed by EC2 instances in the Ethereum network to interact with other AWS services. You create a permissions policy for the role,

create the role (which automatically creates an instance profile of the same name), and then attach the permissions policy to the role.

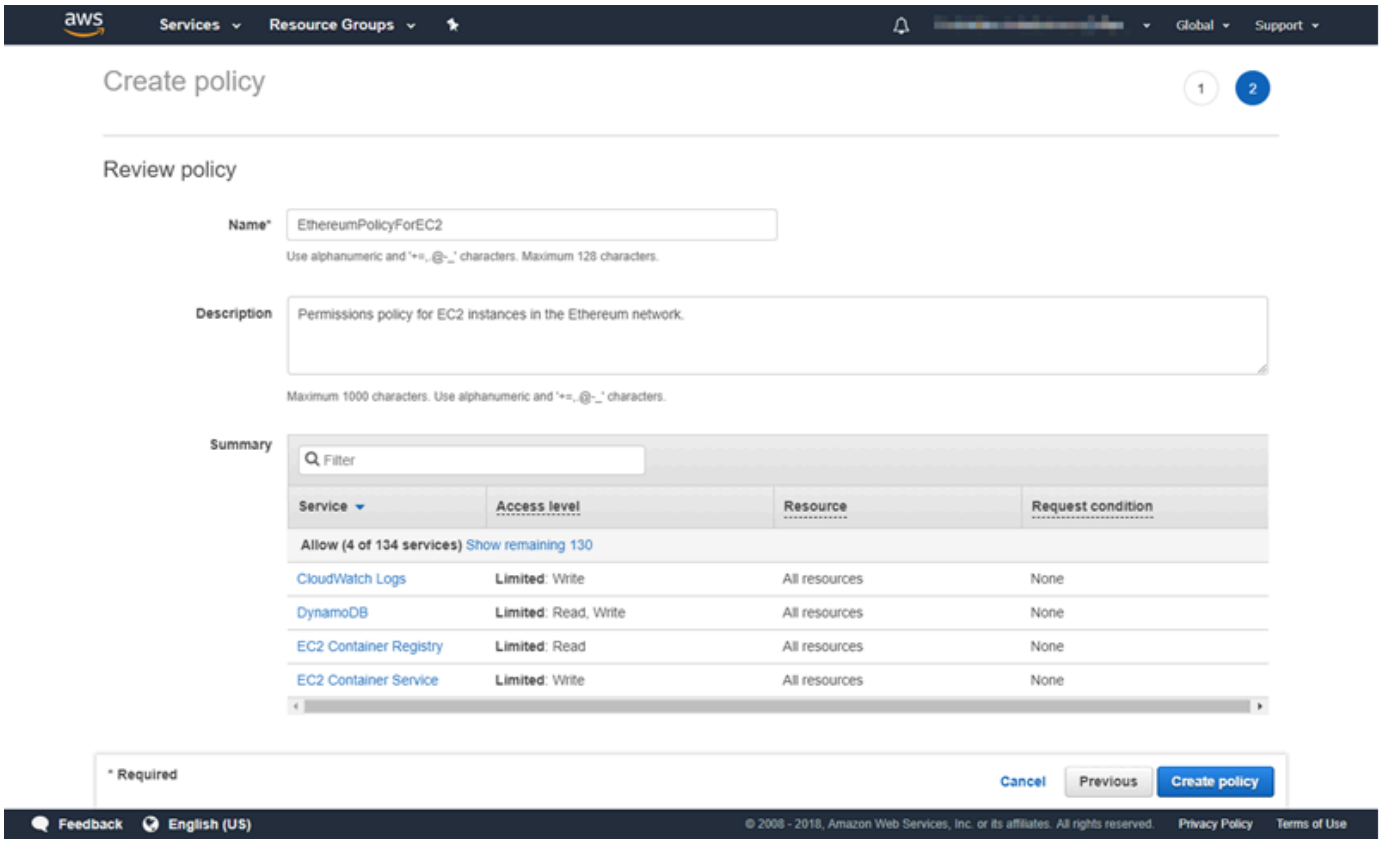
To create an EC2 instance profile

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

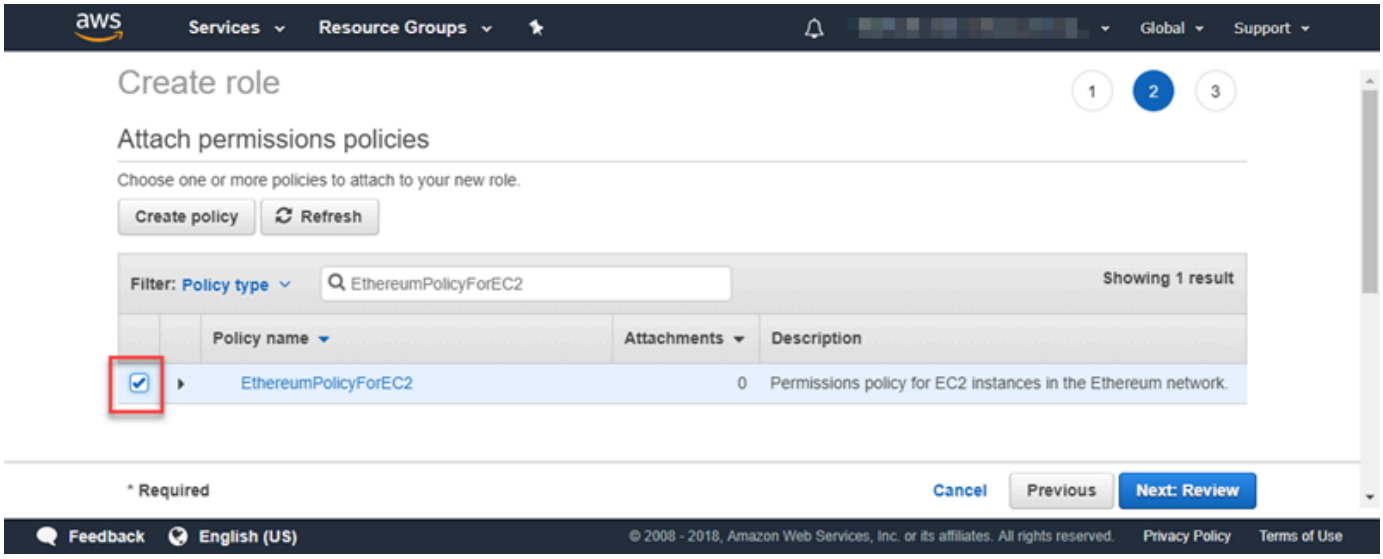
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:PutItem",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": "*"
        }
    ]
}
```

3. Choose **Review policy**.

4. For **Name**, enter a value that helps you identify this permissions policy, for example *EthereumPolicyForEC2*. For **Description**, enter a brief summary. Choose **Create policy**.

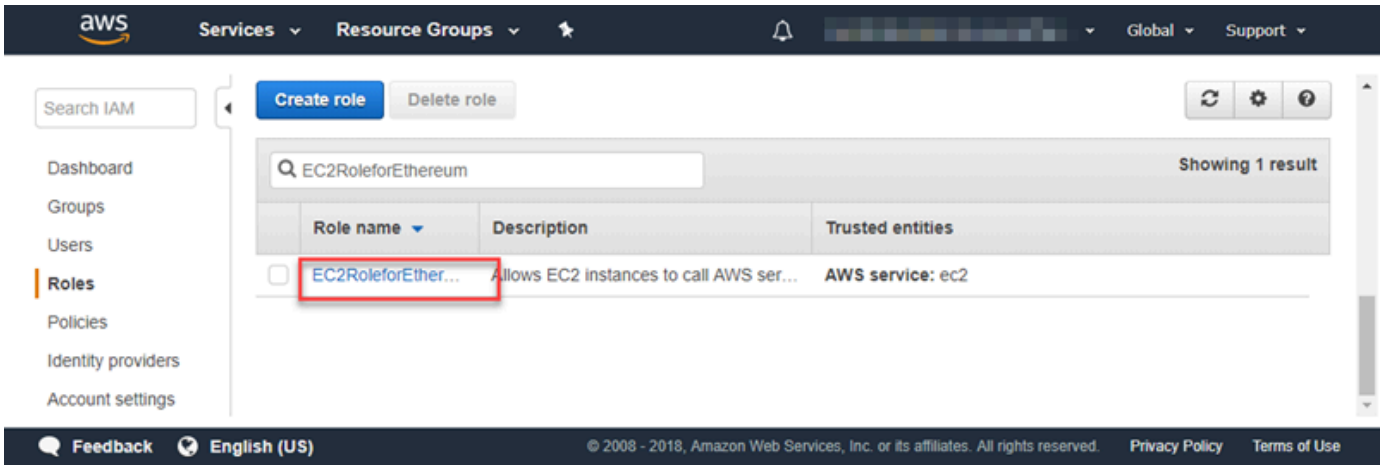


5. Choose **Roles**, **Create role**.
6. Choose **EC2**, **Next: Permissions**.
7. In the **Search** field, enter the name of the permissions policy that you created earlier, for example *EthereumPolicyForEC2*.
8. Select the check mark for the policy that you created earlier, and choose **Next: Review**.

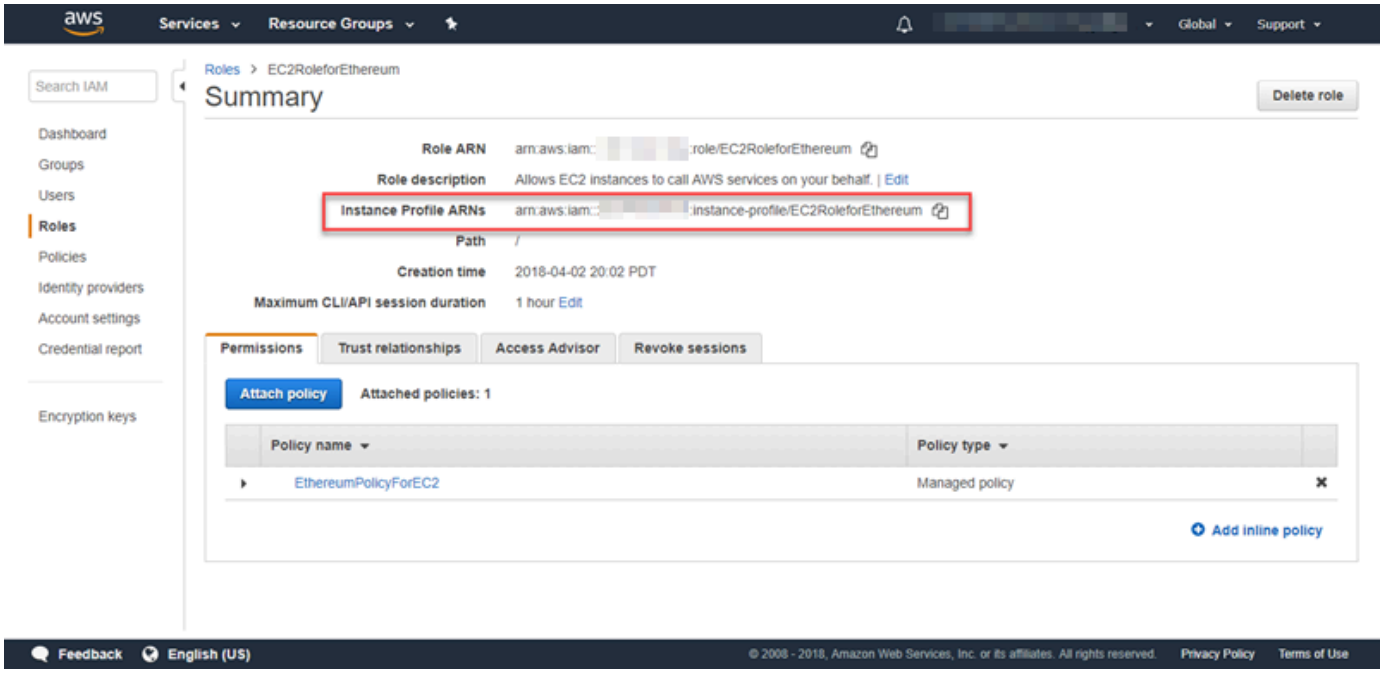


9. For **Role name**, enter a value that helps you identify the role, for example *EC2RoleForEthereum*. For **Role description**, enter a brief summary.Choose **Create role**.

10. Select the role that you just created from the list. If your account has many roles, you can enter the role name in the **Search** field.



11. Copy the **Instance Profile ARN** value and save it so you can copy it again. You need this ARN when you create the Ethereum network.



Create a Bastion Host

In this tutorial, you create a *bastion host*. This is an EC2 instance that you use to connect to the web interfaces and instances in your Ethereum network. Its sole purpose is to forward SSH traffic from trusted clients outside the VPC so that they can access Ethereum network resources.

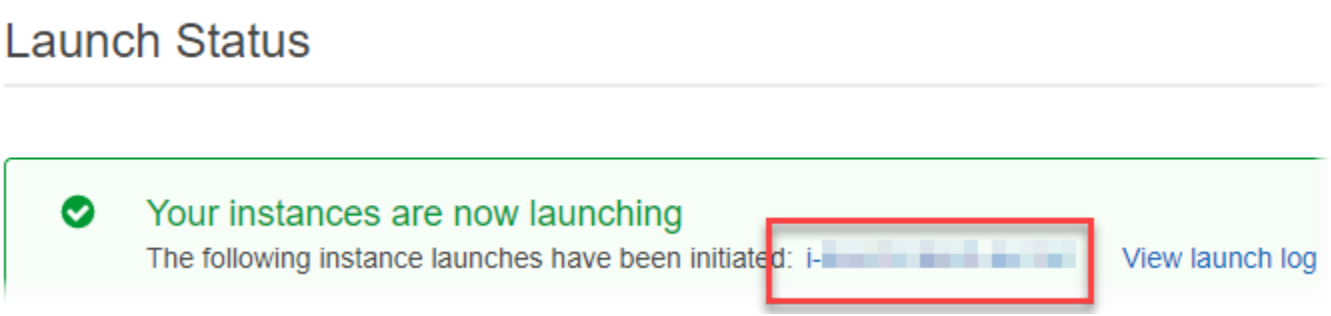
You set up the bastion host because the Application Load Balancer that the template creates is internal, meaning it only routes internal IP addresses. The bastion host:

- Has an internal IP address that the Application Load Balancer recognizes because you launch it in the second public subnet that you created earlier.
- Has a public IP address that the subnet assigns, which can be accessed by trusted sources outside the VPC.
- Is associated with the security group for the Application Load Balancer you created earlier, which has an inbound rule that allows SSH traffic (port 22) from trusted clients.

To be able to access the Ethereum network, trusted clients need to be set up to connect through the bastion host. For more information, see Connect to EthStats and EthExplorer Using the Bastion Host. A bastion host is one approach. You can use any approach that provides access from trusted clients to private resources within a VPC.

To create a bastion host

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



Create the Ethereum Network

The Ethereum network that you specify using the template in this topic launches an AWS CloudFormation stack that creates an Amazon ECS cluster of EC2 instances for the Ethereum network. The template relies on the resources that you created earlier in Set Up Prerequisites.

When you launch the AWS CloudFormation stack using the template, it creates nested stacks for some tasks. After they are complete, you can connect to resources served by the network's Application Load Balancer through the bastion host to verify that your Ethereum network is running and accessible.

To create the Ethereum network using the AWS Blockchain Template for Ethereum

1. See Getting Started with AWS Blockchain Templates, and open the latest AWS Blockchain Template for Ethereum in the AWS CloudFormation console using the quick-links for your AWS Region.
2. Enter values according to the following guidelines:

   - For **Stack name**, enter a name that is easy for you to identify. This name is used within the names of resources that the stack creates.
   - Under **Ethereum Network Parameters** and **Private Ethereum Network Parameters**, leave the default settings.

     > **Warning**
     >
     > Use the default accounts and associated mnemonic phrase for testing purposes only. Do not send real Ether using the default set of accounts because anyone with access to the mnemonic phrase can access or steal Ether from the accounts. Instead, specify custom accounts for production purposes. The mnemonic phrase associated with the default account is `outdoor father modify clever trophy abandon vital feel portion grit evolve twist`.

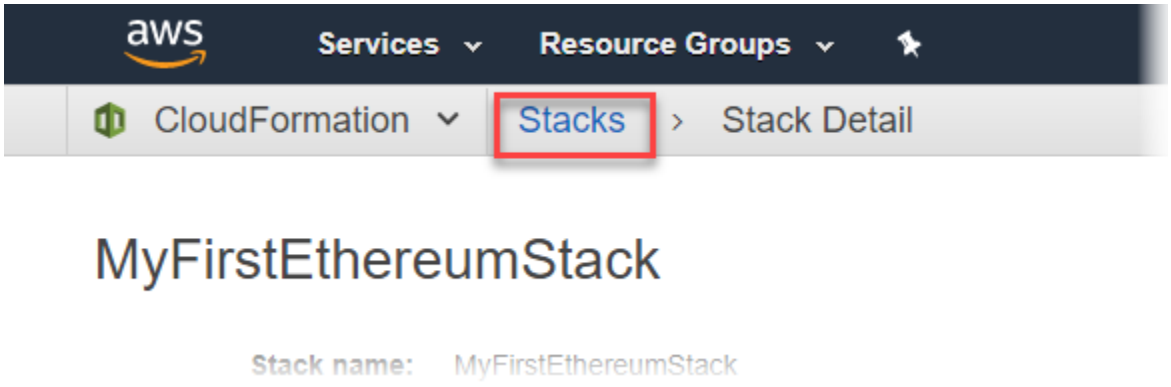
   - Under **Platform configuration**, leave the default settings, which creates an Amazon ECS cluster of EC2 instances. The alternative, **docker-local** creates an Ethereum network using a single EC2 instance.
   - Under **EC2 configuration**, select options according to the following guidelines:
     - For **EC2 Key Pair**, select a key pair. For information about creating a key pair, see Create a Key Pair.
     - For **EC2 Security Group**, select the security group you created earlier in Create Security Groups.
     - For **EC2 Instance Profile ARN**, enter the ARN of the instance profile that you created earlier in Create an IAM Role for Amazon ECS and an EC2 Instance Profile.
   - Under **VPC network configuration, select options according to the following guidelines:**
     - For **VPC ID**, select the VPC that you created earlier in Create a VPC and Subnets.
     - For **Ethereum Network Subnet IDs**, select the single private subnet that you created earlier in the procedure To create the VPC.

- Under **ECS cluster configuration**, leave the defaults. This creates an ECS cluster of three EC2 instances.
- Under **Application Load Balancer configuration (ECS only)**, select options according to the following guidelines:
  - For **Application Load Balancer Subnet IDs**, select two public subnets from the list of subnets that you noted earlier.
  - For **Application Load Balancer Security Group**, select the security group for the Application Load Balancer that you created earlier in Create Security Groups.
  - For **IAM Role**, enter the ARN of the ECS role that you created earlier in Create an IAM Role for Amazon ECS and an EC2 Instance Profile.
- Under **EthStats**, select options according to the following guidelines:
  - For **Deploy EthStats**, leave the default setting, which is *true*.
  - For **EthStats Connection Secret**, type an arbitrary value that is at least six characters.
- Under **EthExplorer**, leave the default setting for **Deploy EthExplorer**, which is *true*.
- Under **Other parameters**, leave the default value for **Nested Template S3 URL Prefix** and make a note of it. This is where you can find nested templates.

3. Leave all other settings to their defaults, select the acknowledgement check box, and choose **Create**.

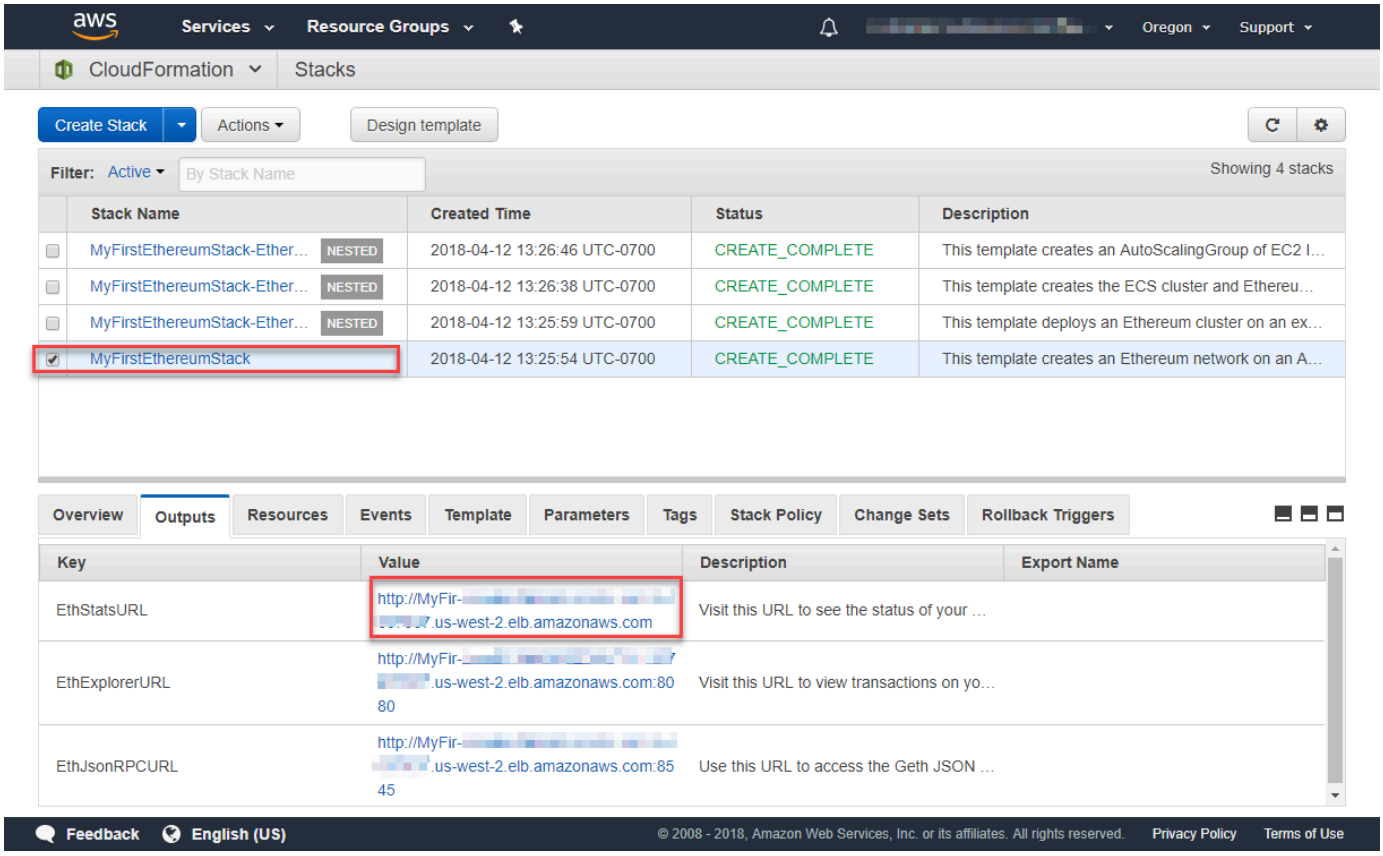
   The **Stack Detail** page for the root stack that AWS CloudFormation launches appears.

4. To monitor the progress of the root stack and nested stacks, choose **Stacks**.



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

> **Important**
>
> You won't be able to connect directly to these URLs or SSH directly until you set up a proxy connection through the bastion host on your client computer. For more information, see Connect to EthStats and EthExplorer Using the Bastion Host.



Connect to EthStats and EthExplorer Using the Bastion Host

To connect to Ethereum resources in this tutorial, you set up SSH port forwarding (SSH tunneling) through the bastion host. The following instructions demonstrate how to do this so that you can connect to EthStats and EthExplorer URLs using a browser. In the instructions below, you first set up a SOCKS proxy on a local port. You then use a browser extension, FoxyProxy, to use this forwarded port for your Ethereum network URLs.

If you use Mac OS or Linux, use an SSH client to set up the SOCKS proxy connection to the bastion host. If you are a Windows user, use PuTTY. Before you connect, confirm that the client computer

you are using is specified as an allowed source for inbound SSH traffic in the security group for the Application Load Balancer that you set up earlier.

To connect to the bastion host with SSH port forwarding using SSH

- Follow the procedures in Connecting to Your Linux Instance Using SSH in the *Amazon EC2 User Guide*. For step 4 of the Connecting to Your Linux Instance procedure, add `-D 9001` to the SSH command, specify the same key pair that you specified in the AWS Blockchain Template for Ethereum configuration, and specify the DNS name of the bastion host.

  ```
  ssh -i /path/my-template-key-pair.pem ec2-user@bastion-host-dns -D 9001
  ```

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

1. Follow the procedures in Connecting to Your Linux Instance from Windows Using PuTTY in the *Amazon EC2 User Guide* through step 7 of the Starting a PuTTY Session procedure, using the same key pair that you specified in the AWS Blockchain Template for Ethereum configuration.
2. In PuTTY, under **Category**, choose **Connection**, **SSH**, **Tunnels**.
3. For **Port forwarding**, choose **Local ports accept connections from other hosts**.
4. Under **Add new forwarded port**:

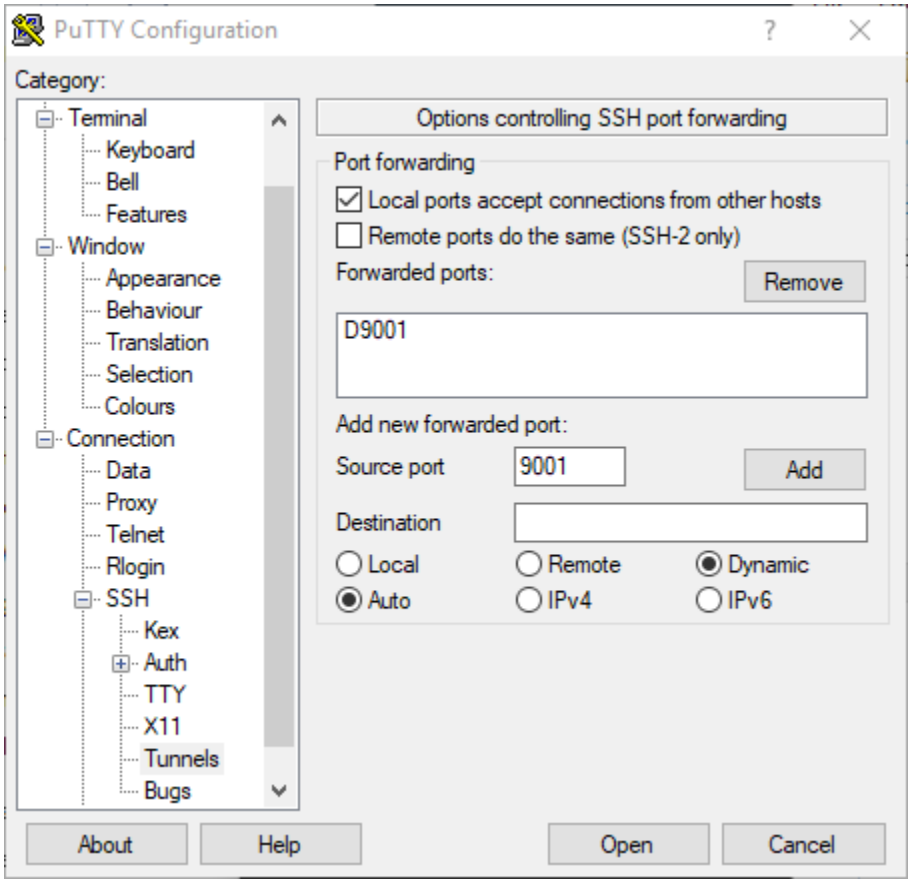
   a. For **Source port**, enter **9001**. This is an arbitrary unused port that we chose, and you can choose a different one if necessary.

   b. Leave **Destination** blank.

   c. Select **Dynamic**.

   d. Choose **Add**.

   For **Forwarded ports**, **D9001** should appear as shown below.



5. Choose **Open** and then authenticate to the bastion host as required by your key configuration. Leave the connection open.

With the PuTTY connection open, you now configure your system or a browser extension to use the forwarded port for your Ethereum network URLs. The following instructions are based on using FoxyProxy Standard to forward connections based on the URL pattern of EthStats and EthExplorer and port 9001, which you established earlier as the forwarded port, but you can use any method that you prefer.

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

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

1. Download and install the FoxyProxy Standard browser extension, and then open **Options** according to the instructions for your browser.
2. Choose **Add New Proxy**.

3. On the **General** tab, make sure that the proxy is **Enabled** and enter a **Proxy Name** and **Proxy Notes** that help you identify this proxy configuration.
4. On the **Proxy Details** tab, choose **Manual Proxy Configuration**. For **Host or IP Address** (or **Server or IP Address** in some versions), enter *localhost*. For **Port**, enter *9001*. Select **SOCKS Proxy?**.
5. On the **URL Pattern** tab, choose **Add New Pattern**.
6. For **Pattern name**, enter a name that's easy to identify, and for **URL Pattern**, enter a pattern that matches all Ethereum resource URLs you created with the template, for example **http://internal-MyUser-LoadB-***. For information on viewing URLs, see Ethereum URLs.
7. Leave the default selections for other settings and choose **Save**.

You are now able to connect to the Ethereum URLs, which are available on CloudFormation console using the **Outputs** tab of the root stack that you created with the template.

Clean Up Resources

AWS CloudFormation makes it easy to clean up resources that the stack created. When you delete the stack, all resources that the stack created are deleted.

To delete resources that the template created

- Open the AWS CloudFormation console, select the root stack that you created earlier, choose **Actions**, **Delete**.

  The **Status** of the root stack you created earlier and the associated nested stacks update to **DELETE_IN_PROGRESS**.

You may choose to delete the prerequisites you created for the Ethereum network.

Delete the VPC

- Open the Amazon VPC console, select the VPC you created earlier and then choose **Actions**, **Delete VPC**. This also deletes the subnets, security groups, and the NAT gateway associated with the VPC.

Delete the IAM role and EC2 instance profile

- Open the IAM console and choose **Roles**. Select the role for ECS and the role for EC2 that you created earlier and choose **Delete**.

Terminate the EC2 instance for the bastion host

- Open the Amazon EC2 dashboard, choose **Running instances**, select the EC2 instance that you created for the bastion host, choose **Actions**, **Instance State**, **Terminate**.

AWS Blockchain Templates and Features

This section provides links for you to begin creating a blockchain network right away, as well as information about configuration options and prerequisites for setting up the network on AWS.

The following templates are available:

- AWS Blockchain Template for Ethereum
- AWS Blockchain Template for Hyperledger Fabric

AWS Blockchain Templates is available in the following Regions:

- US West (Oregon) Region (us-west-2)
- US East (N. Virginia) Region (us-east-1)
- US East (Ohio) Region (us-east-2)

> **Note**
>
> Running a template in a Region not listed above launches resources in the US East (N. Virginia) Region (us-east-1).

Using the AWS Blockchain Template for Ethereum

Ethereum is a blockchain framework that runs smart contracts using Solidity, an Ethereum-specific language. Homestead is the most recent release of Ethereum. For more information, see the Ethereum Homestead Documentation and the Solidity documentation.

Links to Launch

See Getting Started with AWS Blockchain Templates for links to launch AWS CloudFormation in specific Regions using the Ethereum templates.

Ethereum Options

When you configure the Ethereum network using the template, you make choices that determine the subsequent requirements:

Choosing the Container Platform

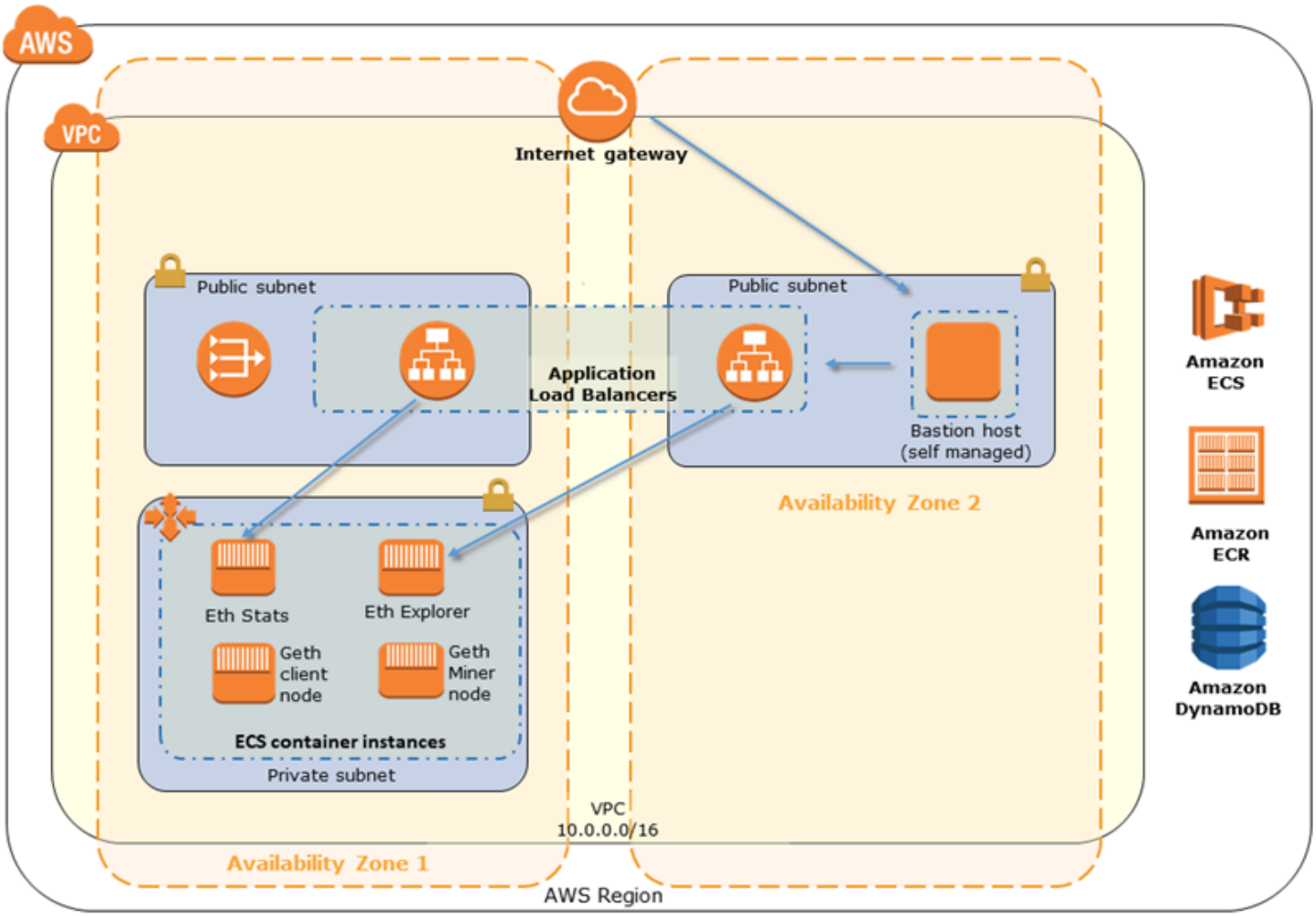
AWS Blockchain Templates use Docker containers stored in Amazon ECR to deploy blockchain software. The AWS Blockchain Template for Ethereum offers two choices for the **Container Platform** :

- **ecs**—Specifies that Ethereum runs on an Amazon ECS cluster of Amazon EC2 instances.
- **docker-local**—Specifies that Ethereum runs on a single EC2 instance.

Using the Amazon ECS Container Platform

With Amazon ECS, you create your Ethereum network on an ECS cluster composed of multiple EC2 instances, with an Application Load Balancer and related resources. For more information about using the Amazon ECS configuration, see the Getting Started with AWS Blockchain Templates tutorial.

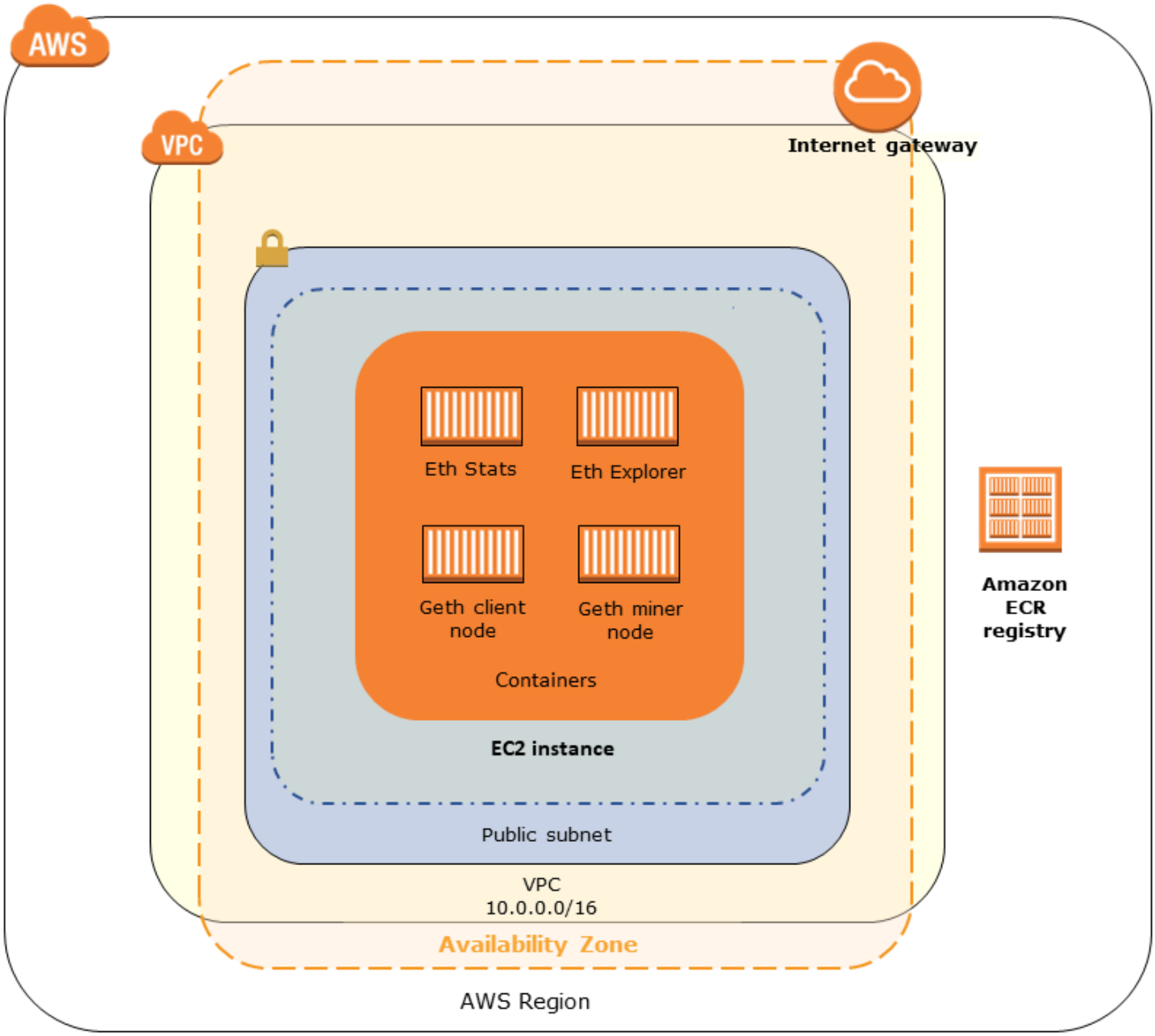
The following diagram depicts an Ethereum network created using the template with the ECS container platform option:



Using the Docker-Local Platform

Alternatively, you can launch Ethereum containers within a single Amazon EC2 instance. All containers run on a single EC2 instance. This is a simplified setup.

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



Choosing a Private or Public Ethereum Network

Choosing an **Ethereum Network ID** value other than 1–4 creates private Ethereum nodes that run within a network that you define, using the private network parameters that you specify.

When you choose an **Ethereum Network ID** from 1–4, the Ethereum nodes that you create are joined to the public Ethereum network. You can ignore private network settings and their defaults. If you choose to join Ethereum nodes to the public Ethereum network, ensure that the appropriate services in your network are internet-accessible.

Changing the Default Accounts and Mnemonic Phrase

A mnemonic phrase is a random set of words that you can use to generate Ethereum wallets (that is, private/public key pairs) for associated accounts on any network. The mnemonic phrase can be used to access Ether for associated accounts. We created a default mnemonic associated with the default accounts that the Ethereum template uses.

> **Warning**
>
> Use the default accounts and associated mnemonic phrase for testing purposes only. Do not send real Ether using the default set of accounts because anyone with access to the mnemonic phrase can access or steal Ether from the accounts. Instead, specify custom accounts for production purposes. The mnemonic phrase associated with the default account is `outdoor father modify clever trophy abandon vital feel portion grit evolve twist`.

Prerequisites

When you set up your Ethereum network using the AWS Blockchain Template for Ethereum, the minimum requirements listed below must be satisfied. The template requires the AWS components listed for each of the following categories:

Topics

Prerequisites for Accessing Ethereum Resources

Prerequisite	For ECS Platform	For Docker-Local
An Amazon EC2 key pair that you can use to access EC2	✓	✓

Prerequisite	For ECS Platform	For Docker-Local
instances. The key must exist in the same Region as the ECS cluster and other resources.		
An internet-facing component, such as a bastion host or an internet-facing load balancer, with an internal address from which traffic is allowed into the Application Load Balancer. This is required with the ECS platform because the template creates an internal load balancer for security reasons. This is required with the docker-local platform when the EC2 instance is in a private subnet, which we recommend. For information about configuring a bastion host, see Create a Bastion Host.	✓	✓ (with private subnet)

IAM Prerequisites

Prerequisite	For ECS Platform	For Docker-Local
An IAM principal (user or group) that has permissio ns to work with all related services.	✓	✓
An Amazon EC2 instance profile with appropriate	✓	✓

Prerequisite	For ECS Platform	For Docker-Local
permissions for EC2 instances to interact with other services. For more informati on, see To create an EC2 instance profile.		
An IAM role with permissions for Amazon ECS to interact with other services. For more information, see Creating the ECS Role and Permissions.	✓	

Security Group Prerequisites

Prerequisite	For ECS Platform	For Docker-Local
A security group for EC2 instances, with the following requirements:	✓	✓
• Outbound rules that allow traffic to 0.0.0.0/0 (default) .	✓	✓
• An inbound rule that allows all traffic from itself (the same security group).	✓	✓
• An inbound rule that allows all traffic from the security group for the Application Load Balancer.	✓	
• Inbound rules that allow HTTP (port 80), EthStats (served on port 8080),		✓

Prerequisite	For ECS Platform	For Docker-Local
JSON RPC over HTTP (port 8545), and SSH (port 22) from trusted external sources, such as your client computer's IP CIDR.		

Prerequisite	For ECS Platform	For Docker-Local
A security group for the Application Load Balancer, with the following requirements: • An inbound rule that allows all traffic from itself (the same security group). • An inbound rule that allows all traffic from the security group for EC2 instances. • Outbound rules that allow all traffic only to the security group for EC2 instances. For more information, see Create Security Groups. • If associating this same security group with a bastion host, an inbound rule that allows SSH (port 22) traffic from trusted sources. • If the bastion host or other internet-facing component is in a different security group, an inbound rule that allows traffic from that component.	✓	

VPC Prerequisites

Prerequisite	For ECS Platform	For Docker-Local
An Elastic IP address, which is used for accessing Ethereum services.	✓	✓
A subnet to run EC2 instances . We strongly recommend a private subnet.	✓	✓
Two publicly accessible subnets. Each subnet must be in different Availability Zones from each other, with one in the same Availability Zone as the subnet for EC2 instances.	✓	

Example IAM Permissions for the EC2 Instance Profile and ECS Role

You specify an EC2 instance profile ARN as one of the parameters when you use the template. If you use the ECS container platform, you also specify an ECS role ARN. The permissions policies attached to these roles allow the AWS resources and instances in your cluster to interact with other AWS resources. For more information, see IAM Roles in the *IAM User Guide*. Use the policy statements and procedures below as a starting point for creating permissions.

Example Permissions Policy for the EC2 Instance Profile

The following permissions policy demonstrates allowed actions for the EC2 instance profile when you choose the ECS container platform. The same policy statements can be used in a docker-local container platform, with `ecs` context keys removed to limit access.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

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

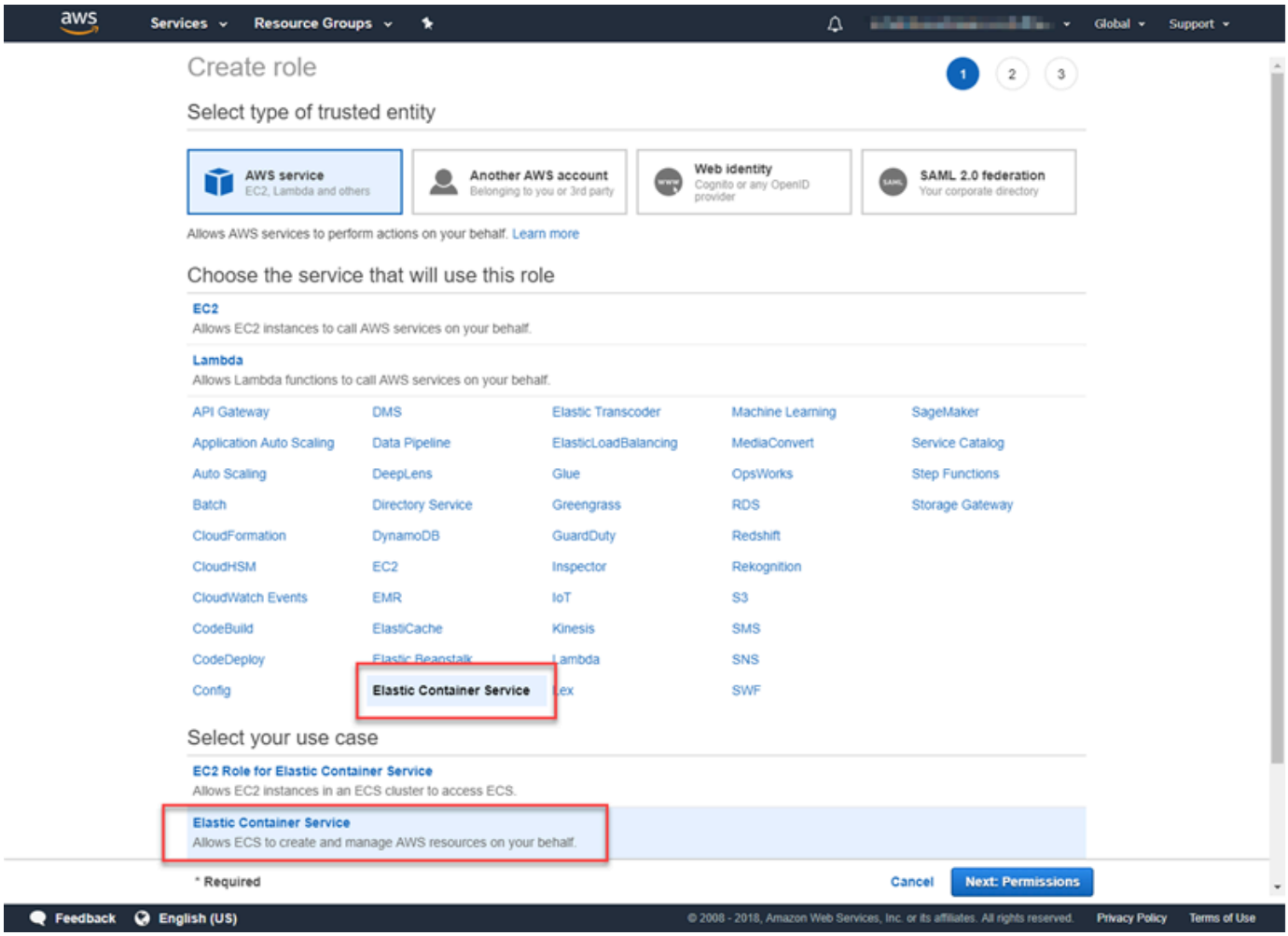
Creating the ECS Role and Permissions

For the permissions attached to the ECS role, we recommend that you start with the **AmazonEC2ContainerServiceRole** permissions policy. Use the following procedure to create a role and attach this permissions policy. Use the IAM console to view the most up-to-date permissions in this policy.

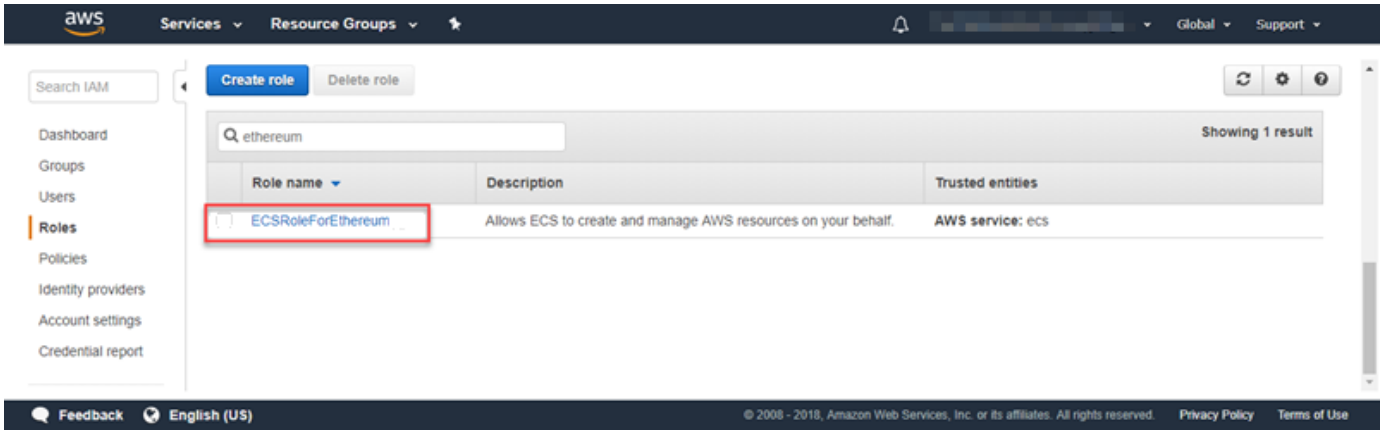
To create the IAM role for Amazon ECS

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**, **Create Role**.
3. Under **Select type of trusted entity**, choose **AWS service**.
4. For **Choose the service that will use this role**, choose **Elastic Container Service**.

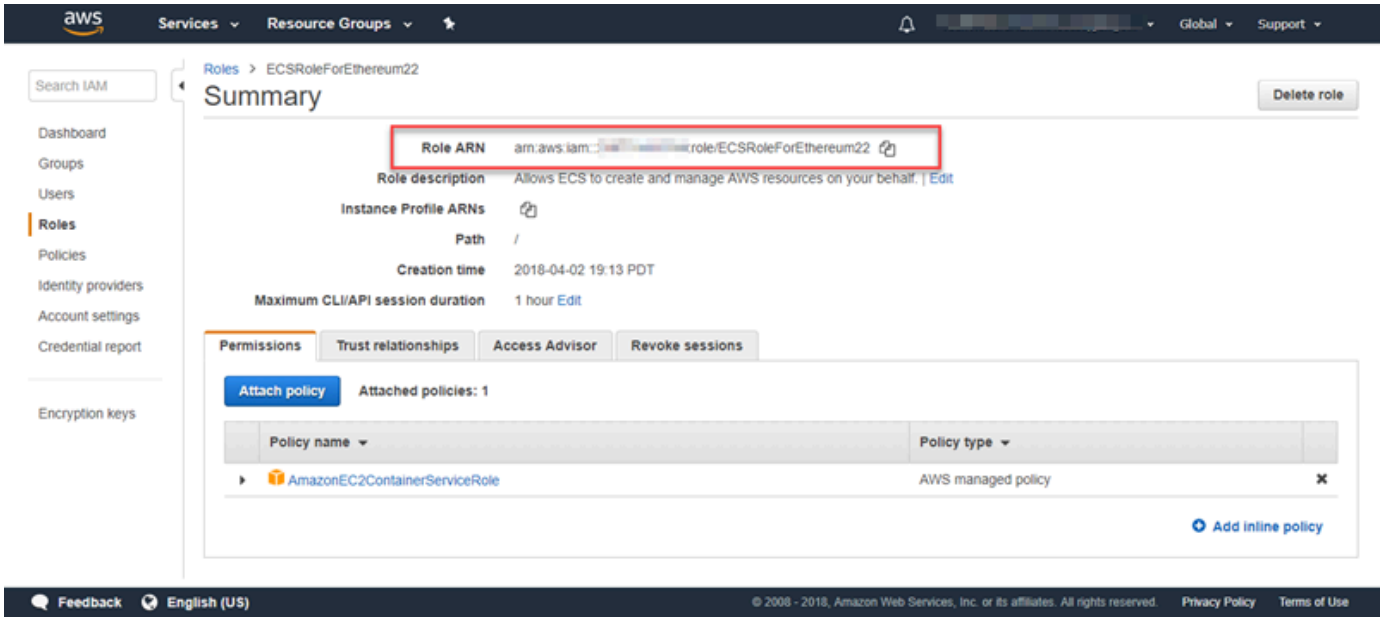
5. Under **Select your use case**, choose **Elastic Container Service**, **Next:Permissions**.



6. For **Permissions policy**, leave the default policy (**AmazonEC2ContainerServiceRole**) selected, and choose **Next:Review**.
7. For **Role name**, enter a value that helps you identify the role, such as *ECSRoleForEthereum*. For **Role Description**, enter a brief summary. Note the role name for later.
8. Choose **Create role**.
9. Select the role that you just created from the list. If your account has many roles, you can search for the role name.



10. Copy the **Role ARN** value and save it so that you can copy it again. You need this ARN when you create the Ethereum network.



Connecting to Ethereum Resources

After the root stack that you create with the template shows **CREATE_COMPLETE**, you can connect to Ethereum resources using the AWS CloudFormation console. How you connect depends on the container platform that you choose, ECS or docker-local:

- **ECS**—The **Output** tab of the root stack provides links to services running on the Application Load Balancer. These URLs are not directly accessible for security reasons. To connect, you can set up and use a *bastion host* to proxy connections to them. For more information, see Proxy Connections Using a Bastion Host below.

- **docker-local**—You connect using the IP address of the EC2 instance hosting Ethereum services as listed below. Use the EC2 console to find the *ec2-IP-address* of the instance that the template created.
  - **EthStats**—Use http://*ec2-IP-address*
  - **EthExplorer**—Use http://*ec2-IP-address*:8080
  - **EthJsonRpc**—Use http://*ec2-IP-address*:8545

  If you specified a public subnet for **Ethereum Network Subnet ID** (**List of VPC Subnets to use** within the template), you can connect directly. Your client must be a trusted source of inbound traffic for SSH (port 22), as well as the ports listed. This is determined by the **EC2 Security Group** that you specified using the AWS Blockchain Template for Ethereum.

  If you specified a private subnet, you can set up and use a *bastion host* to proxy connections to these addresses. For more information, see Proxy Connections Using a Bastion Host below.

Proxy Connections Using a Bastion Host

With some configurations, Ethereum services may not be publicly available. In those cases, you can connect to Ethereum resources through a *bastion host*. For more information about bastion hosts, see Linux Bastion Host Architecture in the *Linux Bastion Host Quick Start Guide*.

The bastion host is an EC2 instance. Make sure that the following requirements are met:

- The EC2 instance for the bastion host is within a public subnet with Auto-assign Public IP enabled and that has an internet gateway.
- The bastion host has the key pair that allows ssh connections.
- The bastion host is associated with a security group that allows inbound SSH traffic from the clients that connect.
- The security group assigned to the Ethereum hosts (for example, the Application Load Balancer if ECS is the container platform, or the host EC2 instance if docker-local is the container platform) allows inbound traffic on all ports from sources within the VPC.

With a bastion host set up, ensure that the clients that connect use the bastion host as a proxy. The following example demonstrates setting up a proxy connection using Mac OS. Replace *BastionIP* with the IP address of the bastion host EC2 instance and *MySshKey.pem* with the key pair file that you copied to the bastion host.

On the command line, type the following:

```
ssh -i mySshKey.pem  ec2-user@BastionIP -D 9001
```

This sets up port forwarding for port 9001 on the local machine to the bastion host.

Next, configure your browser or system to use SOCKS proxy for `localhost:9001`. For example, using Mac OS, select **System Preferences**, **Network**, **Advanced**, select **SOCKS proxy**, and type **localhost:9001**.

Using FoxyProxy Standard with Chrome, select **More Tools**, **Extensions**. Under **FoxyProxy Standard**, select **Details**, **Extension options**, **Add New Proxy**. Select **Manual Proxy Configuration**. For **Host or IP Address** type **localhost** and for **Port** type **9001**. Select **SOCKS proxy?**, **Save**.

You should now be able to connect to the Ethereum host addresses listed in the template output.

Using the AWS Blockchain Template for Hyperledger Fabric

Hyperledger Fabric is a blockchain framework that runs smart contracts called *chaincode*, which are written in Go. You can create a private network with Hyperledger Fabric, limiting the peers that can connect to and participate in the network. For more information about Hyperledger Fabric, see the Hyperledger Fabric documentation. For more information about chaincode, see the Chaincode for Developers topic in the Hyperledger Fabric documentation.

The AWS Blockchain Template for Hyperledger Fabric only supports a *docker-local* container platform, meaning the Hyperledger Fabric containers are deployed on a single EC2 instance.

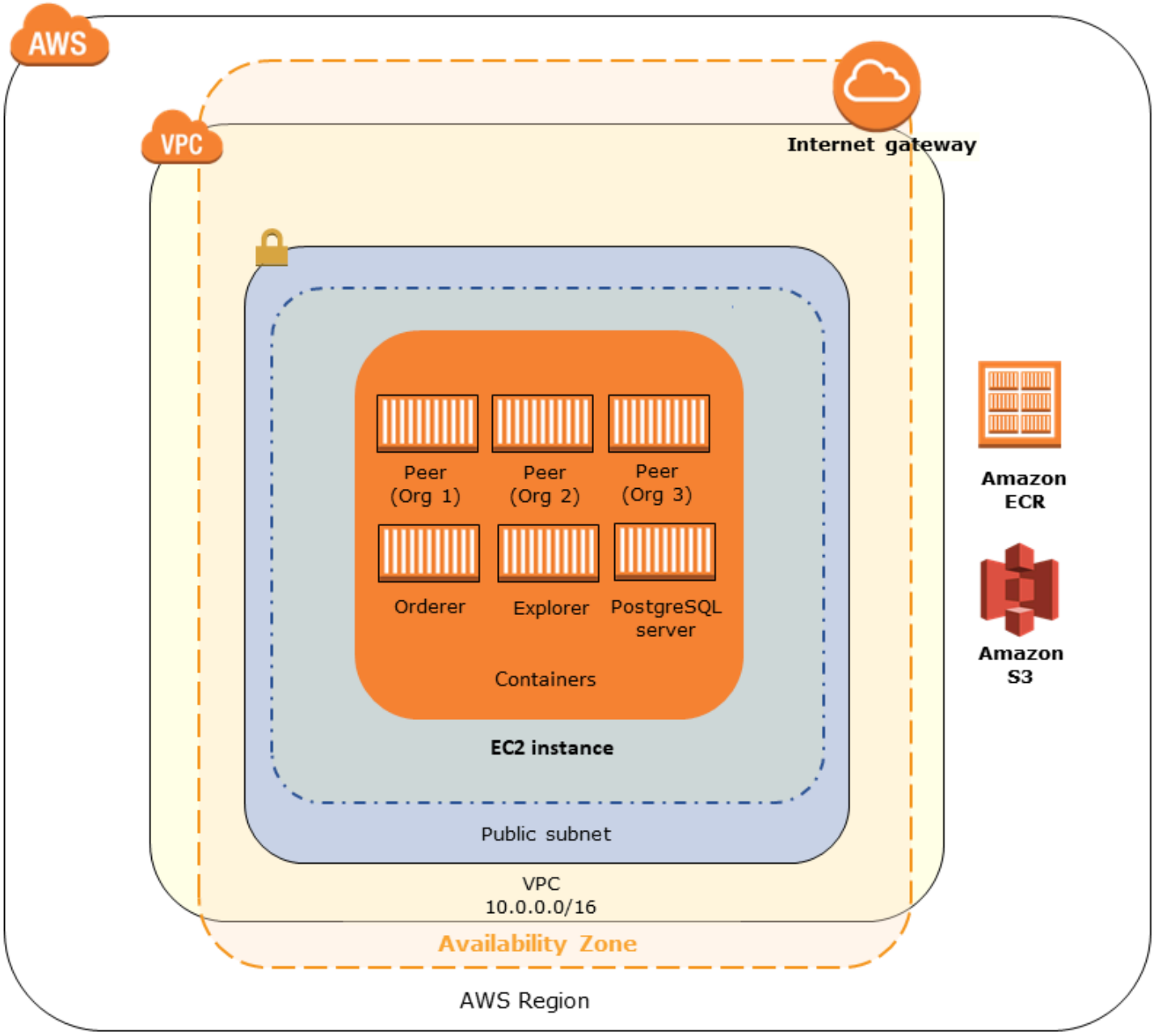
Links to Launch

See Getting Started with AWS Blockchain Templates for links to launch AWS CloudFormation in specific Regions using the Hyperledger Fabric templates.

AWS Blockchain Template for Hyperledger Fabric Components

The AWS Blockchain Template for Hyperledger Fabric creates an EC2 instance with Docker, and launches a Hyperledger Fabric network using containers on that instance. The network includes one order service and three organizations, each with one peer service. The template also launches a Hyperledger Explorer container, which allows you to browse blockchain data. A PostgreSQL server container is launched to support Hyperledger Explorer.

The following diagram depicts a Hyperledger Fabric network created using the template:



Prerequisites

Before you launch a Hyperledger Fabric network using template, make sure that the following requirements are satisfied:

- The IAM principle (user or group) that you use must have permission to work with all related services.

- You must have access to a key pair that you can use to access EC2 instances (for example, using SSH). The key must exist in the same region as the instance.
- You must have an EC2 instance profile with a permissions policy attached that allows access to Amazon S3 and to Amazon Elastic Container Registry (Amazon ECR) to pull containers. For an example permissions policy, see Example IAM Permissions for the EC2 Instance Profile.
- You must have a Amazon VPC network with a public subnet, or a private subnet with a NAT Gateway and Elastic IP address so that Amazon S3, AWS CloudFormation, and Amazon ECR can be accessed.
- You must have an EC2 security group with inbound rules that allow SSH traffic (port 22) from the IP addresses that need to connect to the instance using SSH, and the same for clients that need to connect to Hyperledger Explorer (port 8080).

Example IAM Permissions for the EC2 Instance Profile

You specify an EC2 instance profile ARN as one of the parameters when you use the AWS Blockchain Template for Hyperledger Fabric. Use the following policy statement as a starting point for the permissions policy attached to that EC2 role and instance profile.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

Connecting to Hyperledger Fabric Resources

After the root stack that you create with the template shows **CREATE_COMPLETE**, you can connect to Hyperledger Fabric resources on the EC2 instance. If you specified a public subnet, you can connect to the EC2 instance as would any other EC2 instance. For more information, see Connecting to Your Linux Instance Using SSH in the *Amazon EC2 User Guide*.

If you specified a private subnet, you can set up and use a *bastion host* to proxy connections to Hyperledger Fabric resources. For more information, see Proxy Connections Using a Bastion Host below.

> **Note**
>
> You may notice that the template allocates a public IP address to the EC2 instance hosting Hyperledger Fabric services; however, this IP address is not publicly accessible because routing policies in the private subnet you specify do not allow traffic between this IP address and public sources.

Proxy Connections Using a Bastion Host

With some configurations, Hyperledger Fabric services may not be publicly available. In those cases, you can connect to Hyperledger Fabric resources through a *bastion host*. For more information about bastion hosts, see Linux Bastion Host Architecture in the *Linux Bastion Host Quick Start Guide*.

The bastion host is an EC2 instance. Make sure that the following requirements are met:

- The EC2 instance for the bastion host is within a public subnet with Auto-assign Public IP enabled and that has an internet gateway.
- The bastion host has the key pair that allows ssh connections.
- The bastion host is associated with a security group that allows inbound SSH traffic from the clients that connect.
- The security group assigned to the Hyperledger Fabric hosts (for example, the Application Load Balancer if ECS is the container platform, or the host EC2 instance if docker-local is the container platform) allows inbound traffic on all ports from sources within the VPC.

With a bastion host set up, ensure that the clients that connect use the bastion host as a proxy. The following example demonstrates setting up a proxy connection using Mac OS. Replace *BastionIP* with the IP address of the bastion host EC2 instance and *MySshKey.pem* with the key pair file that you copied to the bastion host.

On the command line, type the following:

```
ssh -i mySshKey.pem  ec2-user@BastionIP -D 9001
```

This sets up port forwarding for port 9001 on the local machine to the bastion host.

Next, configure your browser or system to use SOCKS proxy for `localhost:9001`. For example, using Mac OS, select **System Preferences**, **Network**, **Advanced**, select **SOCKS proxy**, and type **localhost:9001**.

Using FoxyProxy Standard with Chrome, select **More Tools**, **Extensions**. Under **FoxyProxy Standard**, select **Details**, **Extension options**, **Add New Proxy**. Select **Manual Proxy Configuration**. For **Host or IP Address** type **localhost** and for **Port** type **9001**. Select **SOCKS proxy?**, **Save**.

You should now be able to connect to the Hyperledger Fabric host addresses listed in the template output.

Document History

The following table describes the documentation changes for this guide.

Latest documentation update: May 1, 2019

Change	Description	Date
Discontinuation of AWS Blockchain Templates.	AWS Blockchain Templates was discontinued on April 30, 2019. No further updates to this service or this supportin g documentation will be made. For the best Managed Blockchain experience on AWS, we recommend that you use Amazon Managed Blockchain (AMB).	May 1, 2019
Bastion host updates.	Modified getting started tutorial and Ethereum prerequisite requirements for the addition of a bastion host, which allows access to web resources served through the internal load balancer when using the ECS platform and the EC2 instance when using docker-local.	May 3, 2018
Created guide.	New developer guide to support initial release of AWS Blockchain Templates.	April 19, 2018

AWS Glossary

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