User Guide

AWS Migration Hub Refactor Spaces

AWS Migration Hub Refactor Spaces: User Guide

Table of Contents

What is AWS Migration Hub Refactor Spaces?

AWS Migration Hub Refactor Spaces is the starting point for incremental application refactoring to microservices in AWS. Refactor Spaces helps reduce the undifferentiated heavy lifting of building and operating AWS infrastructure for incremental refactoring. You can use Refactor Spaces to help reduce risk when you develop applications into microservices or extend existing applications with new features written in microservices.

A Refactor Spaces environment provides the infrastructure, multi-account networking, and routing that you need to modernize incrementally. With Refactor Spaces environments, you can transparently add new services to an external HTTPS endpoint and incrementally route traffic to the new services. Refactor Spaces bridges networking across AWS accounts so that legacy and new services can communicate while they maintain the independence of separate accounts.

Refactor Spaces creates these resources in your account. This gives you the flexibility to apply your own configurations, like API Gateway custom domains, after Refactor Spaces creates them.

Refactor Spaces provides an application that models the Strangler Fig pattern for incremental refactoring. A Refactor Spaces application orchestrates Amazon API Gateway, Network Load Balancer, and resource-based AWS Identity and Access Management (IAM) policies so that you can transparently add new services to an external HTTP endpoint. For more information about the Strangler Fig pattern, see Strangler Fig Application.

You can also incrementally route traffic to the new services. Refactor Spaces periodically resolves Domain Name System (DNS) names for these services. This keeps underlying architecture changes hidden from your application consumers.

Topics

Are you a first-time Refactor Spaces user?

If you are a first-time user of Refactor Spaces, we recommend that you begin by reading the following sections:

- Refactor Spaces concepts
- How Refactor Spaces works
- Setting up
- Getting started with Refactor Spaces

Pricing

All Refactor Spaces orchestrated resources (for example, Transit Gateway) are provisioned in your AWS account. Therefore, you pay for usage of Refactor Spaces plus any costs associated with provisioned resources. You are charged for Refactor Spaces usage based on the number of hours that you run your refactor environments and API requests to Refactor Spaces. For more information, see AWS Migration Hub pricing.

Refactor Spaces concepts

This section describes the key components that you can create and manage when using AWS Migration Hub Refactor Spaces.

Topics

- Environment
- Applications
- Services
- Route

Environment

The Refactor Spaces environment contains Refactor Spaces applications and services. The environment provides the infrastructure, multi-account networking, and routing that you need to modernize incrementally, along with a unified view of networking, applications, and services across multiple AWS accounts. Refactor Spaces also includes a multi-account network fabric consisting of bridged virtual private clouds (VPCs). This way, resources within the fabric can interact through private IP addresses.

The *environment owner* is the account that the Refactor Spaces environment is created in. The environment owner has cross-account visibility into applications, services, and routes created in the environment, regardless of the account that creates the resource.

Applications

A Refactor Spaces application contains services and routes and provides a single external endpoint to expose the application to external callers. The application provides a Strangler Fig proxy for incremental application refactoring. For information about Strangler Fig, see Strangler Fig Application.

The Refactor Spaces application models the Strangler Fig pattern and orchestrates Amazon API Gateway, API Gateway VPC links, Network Load Balancer, and resource-based AWS Identity and Access Management (IAM) policies so that you can transparently add new services to the application's HTTP endpoint. It also incrementally routes traffic away from your existing application to the new services. This keeps the underlying architecture changes transparent to the application consumer.

Services

Refactor Spaces services provide your application's business capabilities and are reachable through unique endpoints. Service endpoints are one of two types: an HTTP/HTTPS URL, or an AWS Lambda function.

After the services are created, Refactor Spaces handles automatic Domain Name System (DNS) resolution for these services. This periodic DNS resolution ensures that the route configuration remains up-to-date and it enables users to create services with DNS names in the URL. Refactor Spaces automatically updates the DNS name when the DNS time-to-live (TTL) expires (or every 60 seconds for TTLs less than 60 seconds).

Route

A Refactor Spaces route is a proxy matching rule that forwards a request to a service. Each request is run against the set of routes configured in the application. If a rule matches, the request is sent to the target service configured for that rule. Applications have a default route that forwards requests to a default service if they don't match any of the rules.

Routes can be toggled between either an active or inactive state as you prepare and release the routes for traffic. Routes are configured on the application's Amazon API Gateway proxy.

You can create routes with path parameters. Refactor Spaces forwards the path parameters that you define to the target service.

How Refactor Spaces works

When you start to use AWS Migration Hub Refactor Spaces, you can use one or more AWS accounts. You can use a single account for testing. However, when you're ready to refactor, we recommend that you start with the following three accounts:

- One account for the existing application.
- One account for the first new microservice.
- One account to act as the refactor *environment owner* where Refactor Spaces configures cross-account networking and routes traffic.

First, you create a Refactor Spaces environment in the account that you choose as the environment owner. Then, you share the environment with the other two accounts with AWS Resource Access Manager. The Refactor Spaces console does this for you. Refactor Spaces automatically shares the resources that it creates within the environment with the other accounts. To do this, it orchestrates AWS Identity and Access Management (IAM) resource-based policies.

The refactor environment provides unified networking across accounts. To do this, it orchestrates Amazon API Gateway, a Network Load Balancer, AWS Transit Gateway, AWS Resource Access Manager, and security groups. The refactor environment contains your existing application and new microservices. After you create a refactor environment, you create a Refactor Spaces application within the environment. The Refactor Spaces application contains services and routes, and it provides a single endpoint to expose the application to external callers.

An application supports routing to services that run in containers, serverless compute, and Amazon Elastic Compute Cloud (Amazon EC2) with public or private visibility. Services within an application can have one of two endpoint types: a URL (HTTP and HTTPS) in a VPC, or an AWS Lambda function. After an application contains a service, you add a default route to direct all traffic from the application proxy to the service that represents the existing application. As you break out or add new capabilities in containers or serverless compute, you add new services and routes to redirect traffic to the new services.

When you create the default route, it defaults to an active state. You can toggle the route to the inactive state to stop sending traffic to the default route. To send the traffic to a new route, create a new route in an active state, or activate a route that is inactive.

For services with URL endpoints in a VPC, Refactor Spaces uses Transit Gateway to automatically bridge all service VPCs within the environment. This means that any AWS resources that you launch in a service VPC can communicate directly with all other service VPCs that you add to the environment. To route traffic to service endpoints with private URLs when you create environments without a transit gateway, you must configure VPC to VPC connectivity between your network infrastructure and the Refactor Spaces application VPC. You can apply additional cross-account routing constraints with VPC security groups. When you create routes that point to services with Lambda endpoints, Refactor Spaces orchestrates Amazon API Gateway Lambda integration to call the function across AWS accounts.

Environments without a network bridge

You can create environments without a network bridge so that you can use your existing network infrastructure to bridge communications across multiple AWS accounts. When you create an application in an environment without a bridge, you must connect your network infrastructure to the application proxy to route traffic to service endpoints with private URLs. You can connect VPCs across accounts in several ways, including VPC Peering, Transit Gateway, or AWS PrivateLink. For more information, see the Amazon VPC-to-Amazon VPC connectivity options in the *Amazon Virtual Private Cloud Connectivity Options* AWS whitepaper.

Setting up

Topics

Sign up for an 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 or text message 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.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Assign permissions to users

You must give users permissions to interact with Refactor Spaces. To give a user full access to Refactor Spaces, the user must assume a role that you create with the following two attached policies:

- The `AWSMigrationHubRefactorSpacesFullAccess` managed policy.
- A policy created for using extra required permissions that is described in Extra required permissions for Refactor Spaces.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.
- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

Grant programmatic access

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in

Which user needs programmatic access?	To	By
		the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Getting started with Refactor Spaces

This section describes how to get started with AWS Migration Hub Refactor Spaces.

Topics

Prerequisites

The following are the prerequisites for using AWS Migration Hub Refactor Spaces.

- You must have one or more AWS accounts, and IAM users set up for these accounts. For more information, see Setting up.
- Designate one of the IAM users as the Refactor Spaces environment owner.

The following steps describe how to use AWS Migration Hub Refactor Spaces in the Migration Hub console.

Step 1: Create an environment

This step describes how to create an environment as part of the Refactor Spaces **Getting started** wizard. You can also create an environment by choosing **Environments** under **App Refactor** in the Refactor Spaces navigation pane.

A Refactor Spaces environment contains the applications, services, and the AWS infrastructure used to safely and incrementally refactor. Refactor Spaces provides a unified view across multiple AWS accounts and simplifies multi-account use cases to accelerate application refactoring. You can create an environment that orchestrates AWS Transit Gateway to bridge cross-account

service communication. Alternatively, you can create an environment without a bridge and manage service-to-service communication with your own infrastructure. If you choose to use an environment's network bridge, you can achieve greater deployment independence across accounts and teams.

After an environment is created, you can share the environment with other AWS accounts, organizational units (OUs) in AWS Organizations, or an entire AWS organization. By sharing the environment with other AWS accounts, users in those accounts can create applications, services, and routes within the environment, unless you use AWS Identity and Access Management (IAM) to restrict access.

To create an environment

1. Using the AWS account that you created in Setting up, sign in to the AWS Management Console and open the Migration Hub console at https://console.aws.amazon.com/migrationhub/.
2. In the Migration Hub console navigation pane, choose **Refactor Spaces**.
3. Choose **Getting started**.
4. Select **Create a refactor environment to start to incrementally modernize to microservices in multiple AWS accounts**.
5. Choose **Start**.
6. Enter a name for the environment.
7. (Optional) Add a description for the environment.
8. (Optional) If you select the **Provision Multi-account network bridge** check box, Refactor Spaces creates a Transit Gateway for your microservices to communicate with your monolith directly across AWS accounts. If you want to use your own network infrastructure for communication across accounts, don't select the check box.
9. Refactor Spaces uses a service-linked role to connect to AWS services to orchestrate them on your behalf. When you use Refactor Spaces for the first time, the service-linked role is created for you with the correct permissions. For more information about the service-linked role, see Using service-linked roles for Refactor Spaces.
10. Choose **Next** to move to the **Create application** page.

Step 2: Create an application

This step describes how to create an application as part of the Refactor Spaces **Getting started** wizard. You can also create an application by choosing **Create application** under **Quick actions** in the Refactor Spaces navigation pane.

Applications are containers of services and routes that provide multi-account traffic routing for services in the application. When you create an application, you must provide Refactor Spaces with a VPC to route traffic from the application to services with private URL endpoints. For each application, Refactor Spaces orchestrate a proxy with an Amazon API Gateway connected to a Network Load Balancer through a VPC link.

To create an application

1. On the **Create application** page, enter a name for your application.
2. Under **Proxy VPC**, choose a proxy virtual private cloud (VPC) or choose **Create VPC**.

   An application's proxy needs a VPC. The proxy's Network Load Balancer is launched in the VPC and an API Gateway VPC link is configured for the VPC and Network Load Balancer.

   When creating an application in an environment without a network bridge, you need to configure VPC to VPC connectivity between your service VPCs and the Refactor Spaces application proxy VPC.
3. Under **Proxy endpoint type** select **Regional** or **Private**.

   The proxy's endpoint can be either Regional or private. Regional API Gateway endpoints are accessible through the public internet, and private API Gateway endpoints are only accessible through VPCs.
4. Choose **Next** to move to the **Share environment** page.

Step 3: Share your environment

This step describes how to share an environment as part of the Refactor Spaces **Getting started** wizard. You can also share an environment by choosing **Share environment** under **Quick actions** in the Refactor Spaces navigation pane.

Environments are shared with other AWS accounts using AWS Resource Access Manager (AWS RAM). An environment share must be accepted by the invited account within twelve hours.

Otherwise, the environment must be shared again. If you're in an AWS organization, then you can enable auto-accepting of shares. AWS RAM supports sharing environments with other AWS accounts, organizational units (OUs) in AWS Organizations, or an entire AWS organization.

Because environments are containers of applications, services, routes, and orchestrated AWS resources, sharing the environment provides some access to these resources from the invited accounts. After sharing with other accounts, users in those accounts can create applications, services, and routes within the environment, unless you use AWS Identity and Access Management (IAM) to restrict access.

When sharing an environment with another AWS account, Refactor Spaces also shares the environment's transit gateway with the other account by orchestrating AWS RAM.

To share an environment

1. Select one of the following principal types to share your environment with:

   - AWS account
   - Organization - entire AWS organization
   - Organizational unit (OU)

   AWS RAM supports sharing environments with other AWS accounts, organizational units (OUs) in AWS Organizations, or an entire AWS organization.
2. Environments are shared with other AWS accounts using AWS Resource Access Manager (AWS RAM). AWS RAM supports sharing environments with other AWS accounts, organizational units (OUs) in AWS Organizations, or an entire AWS organization. If you want to share an environment with an entire AWS organization or OU, you must enable sharing with the organization in AWS RAM before trying to share in Refactor Spaces.
3. Enter the AWS account of the principal, and then choose **Add**.
4. Choose **Next** to move to the **Review** page.
5. Review the information you entered in the previous steps.
6. If everything looks correct, choose **Create environment**. If you want to change something, choose **Previous**.

Step 4: Create a service

Services provide the application's business capabilities. Your existing application is represented by one or more services. Each service has an endpoint (either an HTTP/HTTPS URL or an AWS Lambda function).

After your environment is created, you view information about the environment on the environment details page (the page with the environment's name as the heading). The environment details page shows a summary of your environment and lists the applications in your environment.

The following procedure describes how to create a service starting from the environment details page. You can also create a service by choosing **Create service** under **Quick actions** in the Refactor Spaces navigation pane.

To create a service from the environment details page

1. From the list of applications, choose the name of the application that you want to add the service to.
2. On the application details page (the page with the application's name as the heading), under **Services**, choose **Create service**.
3. Enter the name for the new service.
4. (Optional) Enter a description for the service.
5. Select one of the service endpoint types, **VPC** or **Lambda**.
   - Select **VPC** if the service is a URL endpoint in a VPC.
     a. Select a VPC to be added to the environment network bridge.
     b. Enter the service URL endpoint.

        If you are creating a service in an environment without a network bridge and the URL is private, you must configure VPC to VPC connectivity between your service VPC and the Refactor Spaces application proxy VPC.

        VPC endpoint URLs can contain publicly resolvable DNS names (http://www.example.com) or an IP address. Private DNS names are not supported in service URLs, but you can use private IP addresses that are in the service's VPC.
     c. (Optional) Enter a health check endpoint URL.

- Select **Lambda** if the service is a Lambda function.

  a. Choose a Lambda function from your account.

  b. (Optional) You can choose a Lambda function alias for the selected Lambda function, if one is available.

6. (Optional) Under **Route traffic to this service**, if you want to set this service as the application's default route, select the corresponding check box.

   When you create a service, you can optionally route application traffic to it at the same time. If the application the service is being created in does not have any routes, you can make the service the application's default route so that all traffic is routed to the service. If the application has existing routes, then you can add a route with a path to point to the service.

Step 5: Create a route

This section describes how to create a route.

An application is used to incrementally re-route traffic away from an existing application to new services. You can also use it to launch new features without touching the existing application.

If the selected application does not have any routes, the new route becomes the application's default route and all traffic is routed to the selected service. If the application has existing routes, then the route is scoped to a path and verb combination.

When created, the default route defaults to an active state. In this case, state is not a required input for default route at creation. However, like all other state values the state of the default route can be updated after creation to an inactive state, but only when all other routes are also inactive. Conversely, no route can be active without the default route also being active.

Routes in environments without a network bridge

When you create a route to a service with a private URL endpoint in an environment without a network bridge, route creation fails if you don't configure VPC to VPC connectivity between the service VPC and the application proxy VPC.

Routes with path parameters

Refactor Spaces supports creating routes with path parameters. To include a path parameter in a route, add the variable in curly braces. For example: `/users/{id}`, where `{id}` represents

the parameter that is forwarded to the service. For URL endpoints you must parse the path parameters.

When using path parameters with Lambda services, the path parameters are parsed by API Gateway and are added to the Lambda event object. Also, path parameters simplify creating service routes to a higher level of granularity. For example:

```
/users => lambda1
/users/{userId} => lambda2
/users/{userId}/projects => lambda3
/users/{userId}/projects/{projectId} => lambda4
```

When using path parameters in Refactor Spaces avoid the following scenarios:

- Duplicate path parameters in a route. For example: /users/{id}/version/{id}. Route creation fails due to duplicate '{id}' parameter value.
- Multiple parameters in same route level. For example: /users/{id}, /users/{name} causes path parameters to collide. Route creation fails because paths can only have 1 path parameter at the same route level.
- Parent-level paths with **Include child paths** selected. A route with path parameters will fail if the path parameter is at the same route level as another route with the same parent path. For example, creating a parent route of `/users/` with **Include child paths** selected, will collide if the desired nested route `/users/{id}` is at the same level as the parent route. A route at a different level than the parent route will not collide.

Creating a route in Refactor Spaces

1. On the application details page (the page with the application's name as the heading), under **Routes**, choose **Create route**.
2. Choose a service for the route.
3. Choose **Create route**.

Tutorials

The following tutorials present AWS Migration Hub Refactor Spaces use cases.

Tutorials for using Refactor Spaces with your own network infrastructure

When you use a Refactor Spaces environment without a network bridge, you must configure VPC to VPC connectivity between services with private URL endpoints and an application proxy VPC. The following tutorials show two common scenarios: connecting through VPC peering, and through your own transit gateway.

We use two accounts for the tutorials, an environment owner account and a service account. The service account contains the private service. In the tutorials, this is an Amazon EC2 web server. Its service VPC is configured to connect with the environment's application proxy VPC.

For the VPC peering scenario, traffic that targets services with private URL endpoints, flows from the application proxy VPC to your service VPC. When you use your own transit gateway, traffic that targets services with private URL endpoints, flows from the application proxy VPC to your network through the transit gateway.

For both scenarios, we provide sample CIDR ranges and port numbers. You can use the values that apply to your configuration.

To avoid connectivity errors, make sure that your VPC CIDR ranges don't overlap with the Refactor Spaces application proxy VPC.

Tutorial: Using your own VPC and VPC peering

This tutorial presents a scenario that contains two VPCs, both with public and private subnets, a network address translation (NAT) gateway, and an internet gateway.

This tutorial also contains an Amazon EC2 instance with a web server, security group, Refactor Spaces environment, application, service, and route. For more information about VPC peering, see Work with VPC peering connections in the *Amazon VPC Peering Guide*.

Step 1: Set up a VPC in the environment owner account

To set up the VPC in the environment owner account

1. Create a VPC with CIDR range 10.3.0.0/16 with one private subnet, one public subnet, and corresponding route tables.
2. Create and attach an internet gateway to your VPC and then add a route table entry for the public subnet.
3. Create a NAT gateway in the public subnet.
4. Create a route table entry for the private subnet to route to the NAT gateway. Use destination `0.0.0.0/0` and the target of the NAT gateway.
5. Create VPC peering in different accounts and in the same AWS Region. Share the VPC with the account that you want to share with the environment.

Step 2: Set up a VPC for the service running in the service account

To set up the VPC for the service running in the service account

1. Create a VPC with CIDR range 10.4.0.0/16 with one private subnet, one public subnet, and corresponding route tables.
2. Create and attach an internet gateway to your VPC and add a route table entry for the public subnet.
3. Create a NAT gateway in the public subnet.
4. Create a route table entry for the private subnet to route to the NAT gateway. Use destination 0.0.0.0/0 and target of the NAT gateway.
5. Accept VPC peering connection.
6. Edit route table to route to VPC peering. For example, when you add a route, for **Destination**, enter 10.3.0.0/16 and for **Target**, enter pcx-0a02261b9c4f051f7-EXAMPLE.

Step 3: Set up VPC peering in the environment owner account

To setup VPC peering in the environment owner account

- Edit route table to route to VPC peering. For example, when you add a route, for **Destination**, enter 10.4.0.0/16 and for **Target**, enter pcx-0a02261b9c4f051f7-EXAMPLE.

Step 4: Set up a web server in the service account

To setup a web server in the service account

1. Create an Amazon EC2 instance in the private subnet.
2. Install a web server on the Amazon EC2 instance. Run the web server on any port, for example, port 3000.
3. Create a security group in the VPC with an inbound rule that allows traffic from the environment owner account CIDR range to the server port, for example, 10.4.0.0/16 to port 3000.
4. Add the security group to the Amazon EC2 instance.

Step 5: Set up a Refactor Spaces environment and application in the environment owner account

Before you begin this step, make sure that you are using the AWS managed policy: AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess managed policy and the Extra required permissions policy for environments without a network bridge policy.

To set up an environment and an application in the environment owner account

1. In the environment account, create a Refactor Spaces environment with network fabric type NONE. Make sure to share the environment with the service account that serves as the environment account.
2. In the environment account, create an application with proxy VPC of the 10.3.0.0/16 CIDR range in environment owner account.

Step 6: Set up Refactor Spaces in the service account

To set up Refactor Spaces in the service account

1. In the service account, create a service pointing to the URL of your EC2 instance.
2. In the service account, create a default route to the EC2 instance.
3. To test that the route works, visit the Refactor Spaces API Gateway URL, as shown in the following example.

```
curl https://x8awx61hm3-EXAMPLE.execute-api.us-west-2.amazonaws.com/prod
```

Tutorial: Using your own AWS Transit Gateway

The following tutorial presents an example of how to use your own AWS Transit Gateway with Refactor Spaces.

In this tutorial, the VPC setup contains two VPCs, both with public and private subnets, a network address translation (NAT) gateway and an internet gateway. The tutorial also contains an Amazon EC2 instance with a web server, security groups, a Refactor Spaces environment, application, service, and route. Traffic flows to the private URL endpoint of a web server through your transit gateway. For more information, see VPC with public and private subnets (NAT).

Step 1: Set up a VPC in the environment owner account

To set up a VPC in the environment owner account

1. Create a VPC with CIDR range 10.1.0.0/16 with one private subnet and one public subnet, and corresponding route tables.
2. Create and attach an internet gateway to your VPC, and add a route table entry for the public subnet.
3. Create a NAT gateway in the public subnet.
4. Create a route table entry for the private subnet to route to the NAT gateway. Use destination 0.0.0.0/0 and the target of the NAT gateway.

Step 2: Set up a VPC in the service account

To set up a VPC in the service account

1. Create a VPC with a CIDR range of 10.2.0.0/16 with one private subnet and one public subnet, and corresponding route tables.
2. Create and attach an internet gateway to your VPC, and add a route table entry for the public subnet.
3. Create a NAT gateway in the public subnet.
4. Create a route table entry for the private subnet to route to the NAT gateway. Use destination 0.0.0.0/0 and the target of the NAT gateway.

Step 3: Set up a web server in the service account VPC.

To set up the web server in the service account VPC

1. Create an Amazon EC2 instance in the private subnet.
2. Install a web server on the Amazon EC2 instance.
3. Create a security group in a member VPC with an inbound rule allowing traffic from the CIDR in Environment Owner Account 10.1.0.0/16.
4. Add the security group to the Amazon EC2 instance.

Step 4: Set up Transit Gateway in the environment owner account

To set up Transit Gateway in the environment owner account

1. Create a Transit Gateway in this account with all the defaults. For more information, see Getting started with transit gateways in the *Amazon VPC Transit Gateways user guide*.
2. Create a VPC attachment to the VPC with all the defaults.
3. Add a route in the main route table of the VPC. Direct the route to the CIDR range of the other VPC.
4. Associate the subnet route table of the VPC with the main route table.

Step 5: Set up Transit Gateway in the service account

To set up Transit Gateway in the service account

1. Share Transit Gateway with service account with the AWS RAM console from environment account.
2. Accept the resource share from service account.
3. Create a Transit Gateway attachment from the service account to the VPC with all the defaults and the two private subnets.
4. Accept the Transit Gateway attachment from environment account.
5. Add a route in the main route table of the VPC. Direct the route to the CIDR range of the other VPC.
6. Associate the subnet route table of the VPC with the main route table.

Now you should have two VPCs with Transit Gateway routing set up.

Step 6: Set up a Refactor Spaces environment and application in the environment owner account

Before you begin this step, make sure that you are using the AWS managed policy: AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess managed policy and the Extra required permissions policy for environments without a network bridge policy.

To set up a Refactor Spaces environment and application in the environment owner account

1. In the environment owner account, create a Refactor Spaces environment with network fabric type NONE. Make sure to share the environment with the service account.
2. In the environment owner account, create an application with proxy VPC of the 10.1.0.0/16 CIDR range in **Environment owner account**.

Step 7: Set up a Refactor Spaces service in the service account

To set up a Refactor Spaces service in the service account

1. In the service account, create a service that points to the URL of Amazon EC2 instance.
2. In the service development account, create a default route to the EC2 instance.

3. To test that the route works, visit the Refactor Spaces API Gateway URL, as shown in the following example.

```
curl https://x8awx61hm3-EXAMPLE.execute-api.us-west-2.amazonaws.com/prod
```

Tutorial: Using Refactor Spaces with AWS Application Migration Service

This tutorial shows how to automatically create a refactor environment and route traffic to your application after migrating an application using AWS Application Migration Service (AWS MGN), so that you can continue modernizing as soon as you've migrated an application. The AWS Migration Hub Refactor Spaces (Refactor Spaces) post-launch action in AWS MGN creates a Refactor Spaces environment, an application, a service, and a default route.

You can use a Refactor Spaces environment to quickly launch new features in AWS Lambda (Lambda), Amazon Elastic Container Service (Amazon ECS), or Amazon Elastic Kubernetes Service (Amazon EKS), or you can safely and incrementally move traffic to new services without modifying the existing application. For more information, see What is AWS Migration Hub Refactor Spaces? and Enable Refactor Spaces in the *AWS MGN User Guide*.

For a quick overview of a complete migration workflow, see Migration workflow in the *AWS MGN User Guide*.

Prerequisites

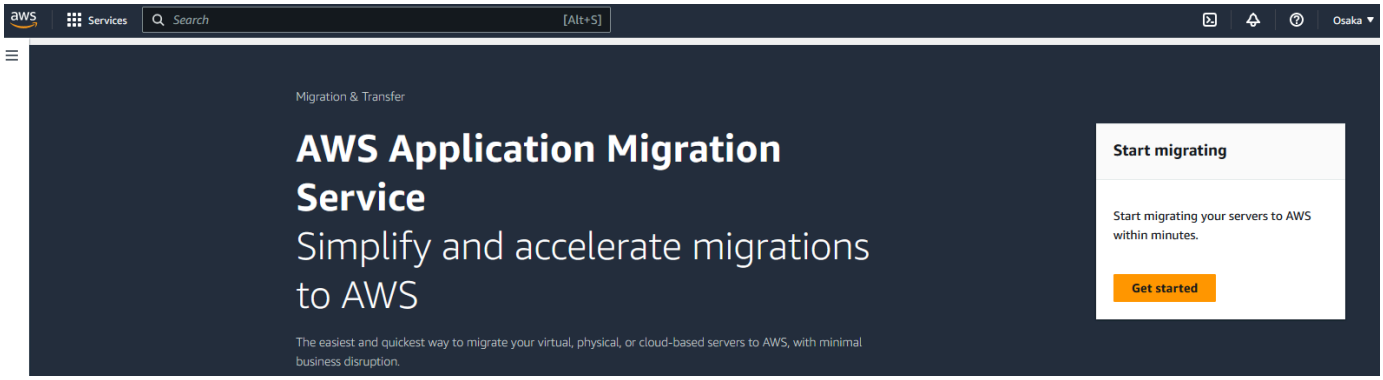
To do this tutorial, you need the following prerequisites:

- An AWS account. If you don't have an AWS account, see Getting started: Are you a first-time AWS user?.
- Familiarity with AWS MGN. For an introduction, see What Is AWS Application Migration Service?.
- Familiarity with Refactor Spaces. For an introduction, see What is AWS Migration Hub Refactor Spaces?

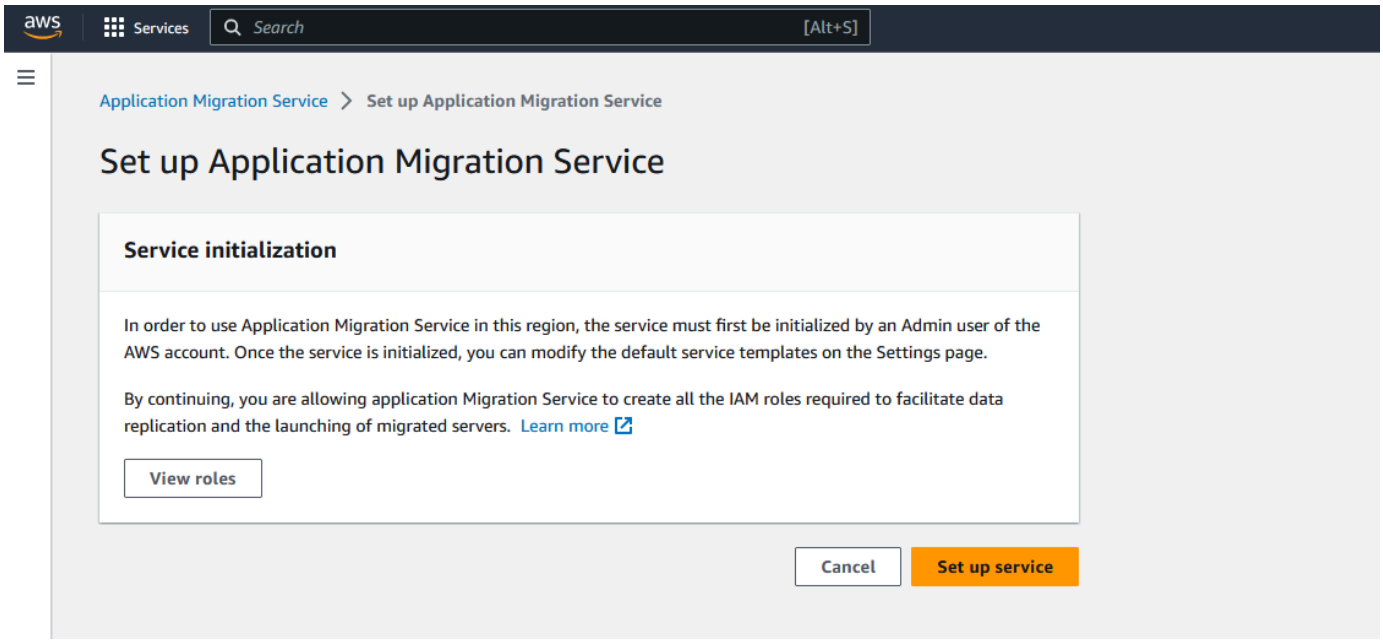
Step 1: Set up AWS MGN

To use AWS MGN, you must first set it up in the AWS Region in which you plan to use it.

1. Open the AWS MGN console at https://console.aws.amazon.com/mgn/home.
2. Choose the Region that you want to work in.
3. Choose the **Get started** button that appears in the following image.



4. If this is your first time using AWS MGN in the Region you chose, you will see the following screen. Choose the **Set up service** button.



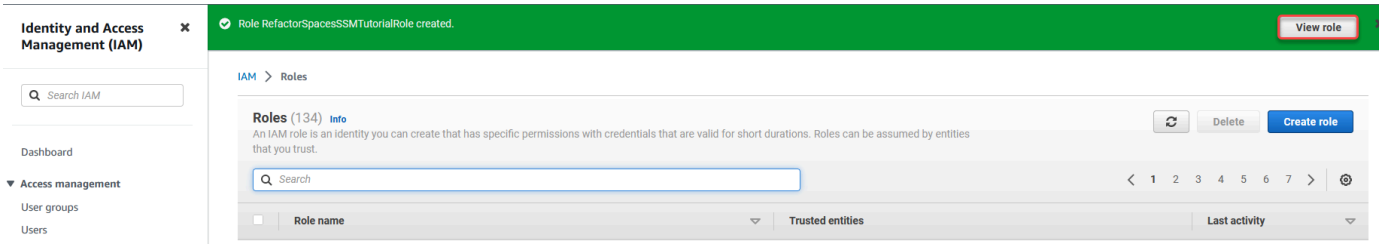
For more information, see Initializing AWS MGN via the console.

Step 2: Create an IAM role

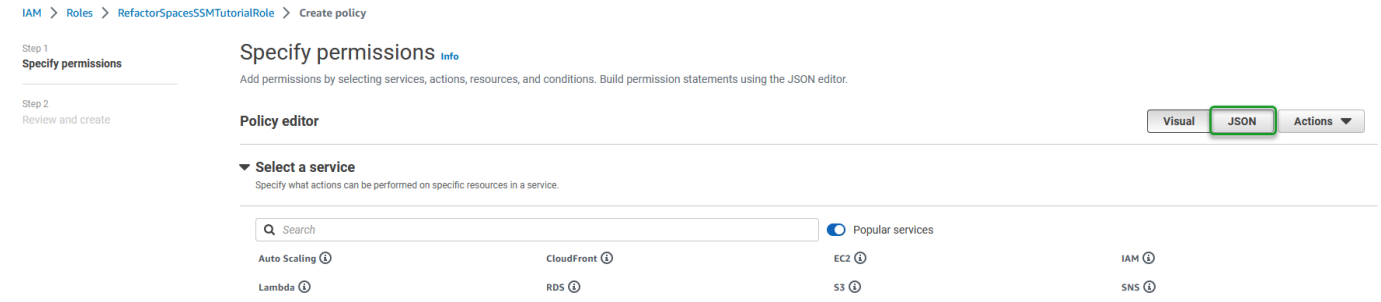
In this step you create a service role that the automation script of the MGN post-launch action will need to assume.

1. Open the IAM console at https://console.aws.amazon.com/iam.

2. In the left navigation pane, choose **Roles**.
3. Choose **Create role**.
4. Under **Use cases for other AWS services:**, choose **Systems Manager** from the dropdown list.
5. Below the dropdown list, choose the **Systems Manager option.**
6. Choose **Next**.
7. In the search field, enter **`Refactor`**, and then press the **`Enter`** key.
8. Choose the following two policies: **AWSMigrationHubRefactorSpacesFullAccess** and **AWSMigrationHubRefactorSpaces-SSMAutomationPolicy**.
9. Choose **Next**.
10. For the role name, enter **`RefactorSpacesSSMTutorialRole`**.
11. Choose **Create role**.
12. When you see a success message like the one in the following image, choose **View role**.



13. Choose **Add permissions**, then choose **Create inline policy**.
14. Choose the **JSON** button that appears in the following image.



15. Replace the JSON in the policy editor with the first block of JSON in the following section: Extra required permissions for Refactor Spaces.
16. Choose **Next**.
17. For the policy name, enter **`RefactorSpacesExtraRequiredPermissions`**.
18. Choose **Create policy**.
19. Copy the ARN that you see in the summary section of the role and save it because you need it to configure the post-launch action.

Step 3: Configure the launch and post-launch templates

In this step, you enable post-launch actions (if you haven't used them in this Region before) and you configure the Refactor Spaces post-launch action.

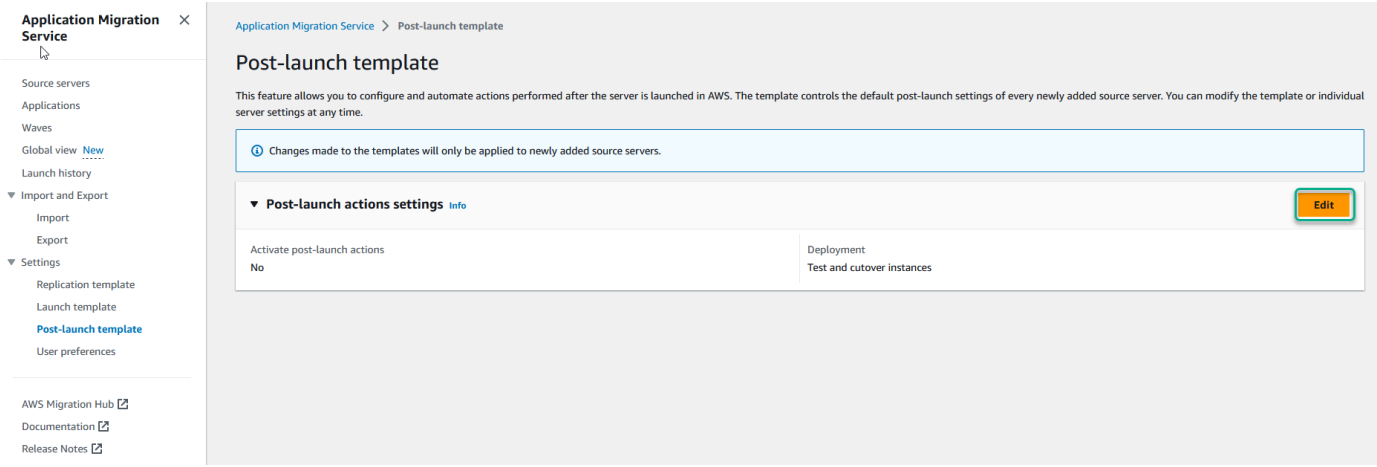
Configure the launch template to transfer server tags

1. In the left navigation pane, choose **Launch template**.
2. Choose **Edit**.
3. Turn on **Transfer server tags**.
4. Choose **Save template**.

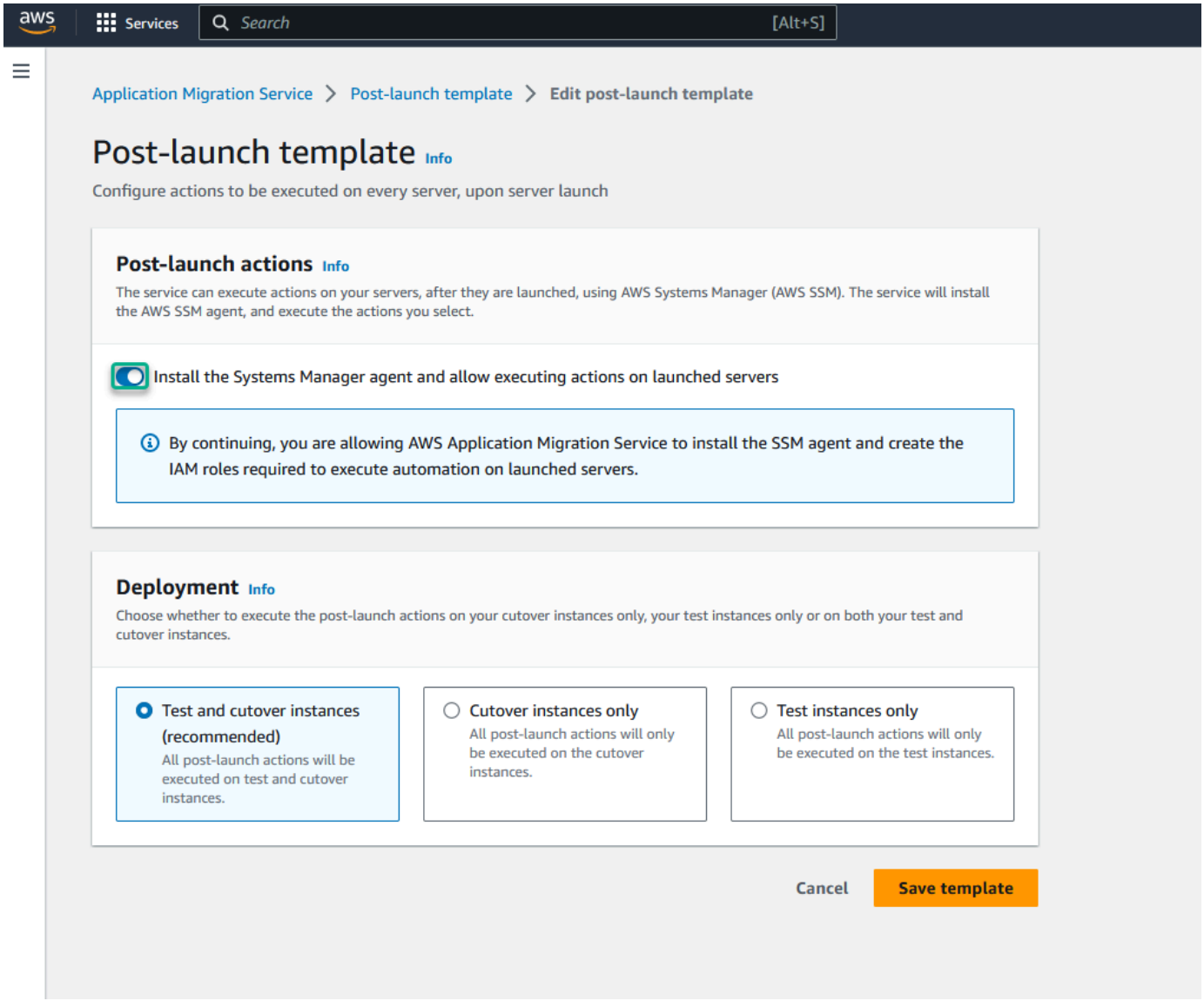
Now you must enable post-launch actions. If you've already enabled post-launch actions in this Region, skip the following procedure and go to the procedure for configuring the Refactor Spaces post-launch action.

Enable post-launch actions

1. In the left navigation pane, choose **Post-launch template**.
2. Choose the edit button that appears in the following image.



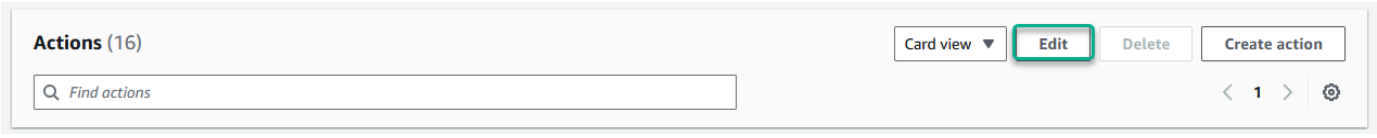
3. Turn on **Install the Systems Manager agent and allow executing actions on launched servers**, as shown in the following image.



4. Choose **Save template**.

Configure the Refactor Spaces post-launch action

1. In the left navigation pane, choose **Post-launch template**.
2. In the **Actions** section, choose the action with the title **Enable Refactor Spaces**.
3. In the **Actions** section, choose the **Edit** button that appears in the following image.



4. Specify the following values in the form:

Name of field or option	Value to use
EnvironmentName	Enter **`RefactorSpacesSSMTutorialEnvironment`** .
EnvironmentId	When you have an existing environment that you want to use, you enter its ID here and leave the **EnvironmentName** field blank. However, in this tutorial, to create a new environment, you enter a value for **EnvironmentName** and leave **EnvironmentId** blank.
ApplicationVpcId	Optional: Enter the ID of a VPC that you want to use for the Refactor Spaces application. If left blank, the application is created in the VPC of the launched Amazon EC2 instance. For information on how to create a VPC, see Create a VPC in the Amazon VPC User Guide.
NetworkFabricType	Choose **TRANSIT_GATEWAY**. To create an environment without a network bridge, enter **`NONE`**.
AccountIdsToShareEnvironment	You can leave this field blank or you can enter the IDs of any AWS accounts with which you want to share the Refactor Spaces environment.
ApplicationName	Enter **`RefactorSpacesSSMTutorialApplication`** .
ServiceName	Enter **`RefactorSpacesSSMTutorialService`** .
Protocol	Choose **http**.

Name of field or option	Value to use
Port	Enter **`4000`**.
UriPath	Enter **`/refactor-spaces/mgn/test`**
InstanceId	Choose **Use value: Launched EC2 InstanceId**. **Important**: For the script to work from AWS MGN, you must use the default value **Launched EC2 InstanceId**.
AutomationAssumeRole	Use the **`RefactorSpacesSSMTutorialRole`** role that you saved in Step 2: Create an IAM role.

5. Choose **Save action**.

Step 4: Create an EC2 instance to use as a source server

1. Open the EC2 console at https://console.aws.amazon.com/ec2.
2. Choose **Launch instance**.
3. For the name field, enter **`RefactorSpacesSSMTutorialEC2Instance`**.
4. Under **Amazon Machine Image (AMI)**, choose **Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type**.
5. Under the **Key pair (login)**, choose an existing key pair or choose **Create new key pair**. You need a key pair to connect to the EC2 instance. For information on how to connect to an EC2 instance, see Connect in the Amazon Elastic Compute Cloud User Guide for Linux Instances.
6. Create an IAM role and attach to it the AWSApplicationMigrationAgentInstallationPolicy policy.

Step 5: Add a source server and launch a test instance

1. Open the AWS MGN console at https://console.aws.amazon.com/mgn/home.
2. Follow the instructions in Linux in the AWS MGN User Guide to install the AWS Replication Agent on the EC2 instance that you created in the previous section.
3. In the left navigation pane of the MGN console, choose **Source servers**.

4. At the bottom of the screen, under **Source server name**, choose the name of the source server that you created for this tutorial. This action takes you to the source server's details page.
5. Choose the **Tags** tab.
6. Choose **Manage tags**, and then choose **Add new tag**.
7. For the key, enter **`refactor-spaces:ssm:optin`**. For the value, enter **`true`**.
8. Choose **Save**.
9. While still on the source server details page, choose **Test and cutover**, then choose **Launch test instances**.
10. After the test instance is launched, navigate to the **Refactor Spaces Environments** page. Choose **RefactorSpacesSSMTutorialEnvironment**, and then choose the application named **RefactorSpacesSSMTutorialApplication**.
11. On the application's page, the Proxy table contains the proxy URL. This URL becomes the migrated application's new front door and traffic is now routed to the migrated Amazon EC2 instance. Services and routes can be added to move traffic away from the migrated application to new microservices.

    Note: When migrating, you can hide the proxy's URL using Amazon API Gateway custom domains before cutting over to your new front door.

You have launched a test instance and a Refactor Spaces environment that includes the new application proxy URL, a default route that points to your migrated application, and the infrastructure necessary to incrementally route traffic to new services.

For more information on launching test instances, see Launching a test instance in the AWS MGN User Guide.

After testing, when you are ready for cutover, see Launching a cutover instance in the AWS MGN User Guide.

Related resources

- For information about IAM roles, see IAM roles in the AWS Identity and Access Management User Guide.
- For more information about EC2 instances, see What is Amazon EC2? in the Amazon Elastic Compute Cloud User Guide for Linux Instances.

Security in AWS Migration Hub Refactor Spaces

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Refactor Spaces, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS Migration Hub Refactor Spaces. It shows you how to configure Refactor Spaces to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Refactor Spaces resources.

Contents

Data protection in AWS Migration Hub Refactor Spaces

The AWS shared responsibility model applies to data protection in AWS Migration Hub Refactor Spaces. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and

management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Refactor Spaces or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at rest

Refactor Spaces encrypts all data at rest.

Encryption in transit

Refactor Spaces internetwork communications support TLS 1.2 encryption between all components and clients.

Identity and Access Management for AWS Migration Hub Refactor Spaces

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 Refactor Spaces 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 that you do in Refactor Spaces.

Service user – If you use the Refactor Spaces service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Refactor Spaces 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 Refactor Spaces, see Troubleshooting AWS Migration Hub Refactor Spaces identity and access.

Service administrator – If you're in charge of Refactor Spaces resources at your company, you probably have full access to Refactor Spaces. It's your job to determine which Refactor Spaces features and resources your service users 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 Refactor Spaces, see How AWS Migration Hub Refactor Spaces works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Refactor Spaces. To view example Refactor Spaces identity-based policies that you can use in IAM, see Identity-based policy examples for AWS Migration Hub Refactor Spaces.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might 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 Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one 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 don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

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 Use cases for IAM users 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. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). 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 Methods to assume a role in the *IAM User Guide*.

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

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **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 Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **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 Use an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) 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*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed 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 Choose 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. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do 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 an 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 Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) 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 AWS Migration Hub Refactor Spaces works with IAM

Before you use IAM to manage access to Refactor Spaces, learn what IAM features are available to use with Refactor Spaces.

IAM features you can use with AWS Migration Hub Refactor Spaces

IAM feature	Refactor Spaces support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level view of how Refactor Spaces and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Refactor Spaces

Supports identity-based policies: Yes

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

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. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Refactor Spaces

To view examples of Refactor Spaces identity-based policies, see Identity-based policy examples for AWS Migration Hub Refactor Spaces.

Resource-based policies within Refactor Spaces

Supports resource-based policies: Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for Refactor Spaces

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Refactor Spaces actions, see Actions defined by AWS Migration Hub Refactor Spaces in the *Service Authorization Reference*.

Policy actions in Refactor Spaces use the following prefix before the action:

```
refactor-spaces
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "refactor-spaces:action1",
      "refactor-spaces:action2"
         ]
```

To view examples of Refactor Spaces identity-based policies, see Identity-based policy examples for AWS Migration Hub Refactor Spaces.

Policy resources for Refactor Spaces

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

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

To see a list of Refactor Spaces resource types and their ARNs, see Resources defined by AWS Migration Hub Refactor Spaces in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Migration Hub Refactor Spaces.

To view examples of Refactor Spaces identity-based policies, see Identity-based policy examples for AWS Migration Hub Refactor Spaces.

Policy condition keys for Refactor Spaces

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Refactor Spaces condition keys, see Condition keys for AWS Migration Hub Refactor Spaces in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Migration Hub Refactor Spaces.

To view examples of Refactor Spaces identity-based policies, see Identity-based policy examples for AWS Migration Hub Refactor Spaces.

Access control lists (ACLs) in Refactor Spaces

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with Refactor Spaces

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using Temporary credentials with Refactor Spaces

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switch from a user to an IAM role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Refactor Spaces

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for Refactor Spaces

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Refactor Spaces functionality. Edit service roles only when Refactor Spaces provides guidance to do so.

Service-linked roles for Refactor Spaces

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

AWS managed policies for AWS Migration Hub Refactor Spaces

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

AWS managed policy: AWSMigrationHubRefactorSpacesFullAccess

You can attach the `AWSMigrationHubRefactorSpacesFullAccess` policy to a role that users can assume.

The `AWSMigrationHubRefactorSpacesFullAccess` policy grants full access to AWS Migration Hub Refactor Spaces, the Refactor Spaces console features and other related AWS services.

Permissions details

The `AWSMigrationHubRefactorSpacesFullAccess` policy includes the following permissions.

- `refactor-spaces` – Allows the user full access to Refactor Spaces.
- `ec2` – Allows the user to perform Amazon Elastic Compute Cloud (Amazon EC2) operations used by Refactor Spaces.
- `elasticloadbalancing` – Allows the user to perform Elastic Load Balancing operations used by Refactor Spaces.
- `apigateway` – Allows the user to perform Amazon API Gateway operations used by Refactor Spaces.
- `organizations` – Allows the user to perform AWS Organizations operations used by Refactor Spaces.
- `cloudformation` – Allows the user to perform AWS CloudFormation operations to create a one-click sample environment from the console.
- `iam` – Allows a service-linked role to be created for the user, which is a requirement for using Refactor Spaces.

Extra required permissions for Refactor Spaces

Before you can use Refactor Spaces, in addition to the `AWSMigrationHubRefactorSpacesFullAccess` managed policy provided by Refactor Spaces, the following extra required permissions must be attached to a role that users can assume.

- Grant permission to create a service-linked role for AWS Transit Gateway.
- Grant permission to attach a virtual private cloud (VPC) to a transit gateway for the calling account for all resources.
- Grant permission to modify the permissions for a VPC endpoint service for all resources.

- Grant permission to add or overwrite specified tags for Amazon EC2 resources.
- Grant permission to return tagged or previously tagged resources for the calling account for all resources.
- Grant permission to perform all AWS Resource Access Manager (AWS RAM) actions for the calling account on all resources.
- Grant permission to perform all AWS Lambda actions for the calling account on all resources.

You can get these extra permissions by creating an IAM policy using the following policy JSON, and attach it to a role.

The following policy grants the extra required permissions necessary to be able to use Refactor Spaces.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "transitgateway.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTransitGatewayVpcAttachment"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyVpcEndpointServicePermissions"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*"
        }
    ]
}
```

The following is the `AWSMigrationHubRefactorSpacesFullAccess` policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RefactorSpaces",
            "Effect": "Allow",
            "Action": [
                "refactor-spaces:*"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "EC2Describe",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcs",
                "ec2:DescribeTransitGatewayVpcAttachments",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeTags",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeInternetGateways"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RequestTagTransitGatewayCreate",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTransitGateway",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTransitGatewayVpcAttachment"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:environment-id": "false"
                }
            }
        },
        {
            "Sid": "ResourceTagTransitGatewayCreate",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTransitGateway",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTransitGatewayVpcAttachment"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
```

```
                "aws:ResourceTag/refactor-spaces:environment-id": "false"
            }
        }
    },
    {
        "Sid": "VpcEndpointServiceConfigurationCreate",
        "Effect": "Allow",
        "Action": [
            "ec2:CreateVpcEndpointServiceConfiguration"
        ],
        "Resource": "*"
    },
    {
        "Sid": "EC2NetworkingModify",
        "Effect": "Allow",
        "Action": [
            "ec2:DeleteTransitGateway",
            "ec2:AuthorizeSecurityGroupIngress",
            "ec2:RevokeSecurityGroupIngress",
            "ec2:DeleteSecurityGroup",
            "ec2:DeleteTransitGatewayVpcAttachment",
            "ec2:CreateRoute",
            "ec2:DeleteRoute",
            "ec2:DeleteTags"
        ],
        "Resource": "*",
        "Condition": {
            "Null": {
                "aws:ResourceTag/refactor-spaces:environment-id": "false"
            }
        }
    },
    {
        "Sid": "VpcEndpointServiceConfigurationDelete",
        "Effect": "Allow",
        "Action": "ec2:DeleteVpcEndpointServiceConfigurations",
        "Resource": "*",
        "Condition": {
            "Null": {
                "aws:ResourceTag/refactor-spaces:application-id": "false"
            }
        }
    },
    {
```

```
            "Sid": "ELBLoadBalancerCreate",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:CreateLoadBalancer"
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:loadbalancer/net/refactor-
spaces-nlb-*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:application-id": "false"
                }
            }
        },
        {
            "Sid": "ELBDescribe",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTags",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ELBModify",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:CreateLoadBalancerListeners",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteTargetGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/refactor-spaces:route-id": [
                        "*"
                    ]
                }
            }
```

```
        },
        {
            "Sid": "ELBLoadBalancerDelete",
            "Effect": "Allow",
            "Action": "elasticloadbalancing:DeleteLoadBalancer",
            "Resource": "arn:*:elasticloadbalancing:*:*:loadbalancer/net/refactor-
spaces-nlb-*"
        },
        {
            "Sid": "ELBListenerCreate",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:CreateListener"
            ],
            "Resource": [
                "arn:*:elasticloadbalancing:*:*:loadbalancer/net/refactor-spaces-nlb-
*",
                "arn:*:elasticloadbalancing:*:*:listener/net/refactor-spaces-nlb-*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:route-id": "false"
                }
            }
        },
        {
            "Sid": "ELBListenerDelete",
            "Effect": "Allow",
            "Action": "elasticloadbalancing:DeleteListener",
            "Resource": "arn:*:elasticloadbalancing:*:*:listener/net/refactor-spaces-
nlb-*"
        },
        {
            "Sid": "ELBTargetGroupModify",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:targetgroup/refactor-spaces-tg-
*"
        },
        {
```

```
            "Sid": "ELBTargetGroupCreate",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:CreateTargetGroup"
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:targetgroup/refactor-spaces-tg-
*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:route-id": "false"
                }
            }
        },
        {
            "Sid": "APIGatewayModify",
            "Effect": "Allow",
            "Action": [
                "apigateway:GET",
                "apigateway:DELETE",
                "apigateway:PATCH",
                "apigateway:POST",
                "apigateway:PUT",
                "apigateway:UpdateRestApiPolicy"
            ],
            "Resource": [
                "arn:aws:apigateway:*::/restapis",
                "arn:aws:apigateway:*::/restapis/*",
                "arn:aws:apigateway:*::/vpclinks",
                "arn:aws:apigateway:*::/vpclinks/*",
                "arn:aws:apigateway:*::/tags",
                "arn:aws:apigateway:*::/tags/*"
            ],
            "Condition": {
                "Null": {
                    "aws:ResourceTag/refactor-spaces:application-id": "false"
                }
            }
        },
        {
            "Sid": "APIGatewayVpcLinksGet",
            "Effect": "Allow",
            "Action": "apigateway:GET",
            "Resource": [
```

```
                "arn:aws:apigateway:*::/vpclinks",
                "arn:aws:apigateway:*::/vpclinks/*"
            ]
        },
        {
            "Sid": "OrganizationDescribe",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudformationStackCreate",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudformationStackTag",
            "Effect": "Allow",
            "Action": [
                "cloudformation:TagResource"
            ],
            "Resource": "arn:aws:cloudformation:*:*:stack/*"
        },
        {
            "Sid": "CreateRefactorSpacesSLR",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "refactor-spaces.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CreateELBSLR",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS managed policy: AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess

This AWS managed policy has reduced permissions when compared to the `AWSMigrationHubRefactorSpacesFullAccess` policy. You can attach the `AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess` policy to a role that users can assume.

You can use the `AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess` policy when you create environments without a network bridge. Since you are using your own network infrastructure, the modified policy removes Transit Gateway permissions and Amazon EC2 security groups related to Transit Gateway actions.

Permissions details

The `AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess` policy includes the following permissions.

- `refactor-spaces` – Allows the user full access to Refactor Spaces.
- `ec2` – Allows the user to perform Amazon Elastic Compute Cloud (Amazon EC2) operations used by Refactor Spaces.
- `elasticloadbalancing` – Allows the user to perform Elastic Load Balancing operations used by Refactor Spaces.
- `apigateway` – Allows the user to perform Amazon API Gateway operations used by Refactor Spaces.
- `organizations` – Allows the user to perform AWS Organizations operations used by Refactor Spaces.
- `cloudformation` – Allows the user to perform AWS CloudFormation operations to create a one-click sample environment from the console.

- `iam` – Allows a service-linked role to be created for the user, which is a requirement for using Refactor Spaces.

Extra required permissions policy for environments without a network bridge

The following policy is an example of a modified version of the extra required permissions policy that you must use together with the `AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess` policy when creating environments without a Transit Gateway. To use this policy, create a role and attach the policy to the role.

- Grant permission to modify the permissions for a virtual private cloud (VPC) endpoint for all resources.
- Grant permission to add or overwrite specified tags for Amazon EC2 resources.
- Grant permission to return tagged or previously tagged resources for the calling account for all resources.
- Grant permission to perform all AWS Resource Access Manager (AWS RAM) actions for the calling account on all resources.
- Grant permission to perform all AWS Lambda actions for the calling account on all resources.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyVpcEndpointServicePermissions"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ram:*"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*"
        }
    ]
}
```

The following is the `AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess` managed policy that you can use when creating environments without a Transit Gateway. To use this policy, create a role and attach the policy to the role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RefactorSpaces",
            "Effect": "Allow",
            "Action": [
                "refactor-spaces:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EC2Describe",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcs",
```

```
                "ec2:DescribeTags",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeInternetGateways"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VpcEndpointServiceConfigurationCreate",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpointServiceConfiguration"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EC2TagsDelete",
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteTags"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/refactor-spaces:environment-id": "false"
                }
            }
        },
        {
            "Sid": "VpcEndpointServiceConfigurationDelete",
            "Effect": "Allow",
            "Action": "ec2:DeleteVpcEndpointServiceConfigurations",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/refactor-spaces:application-id": "false"
                }
            }
        },
        {
            "Sid": "ELBLoadBalancerCreate",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:CreateLoadBalancer"
```

```
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:loadbalancer/net/refactor-
spaces-nlb-*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:application-id": "false"
                }
            }
        },
        {
            "Sid": "ELBDescribe",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTags",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ELBModify",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:CreateLoadBalancerListeners",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteTargetGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/refactor-spaces:route-id": [
                        "*"
                    ]
                }
            }
        },
        {
            "Sid": "ELBLoadBalancerDelete",
            "Effect": "Allow",
            "Action": "elasticloadbalancing:DeleteLoadBalancer",
```

```
            "Resource": "arn:*:elasticloadbalancing:*:*:loadbalancer/net/refactor-
spaces-nlb-*"
        },
        {
            "Sid": "ELBListenerCreate",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:CreateListener"
            ],
            "Resource": [
                "arn:*:elasticloadbalancing:*:*:loadbalancer/net/refactor-spaces-nlb-
*",
                "arn:*:elasticloadbalancing:*:*:listener/net/refactor-spaces-nlb-*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:route-id": "false"
                }
            }
        },
        {
            "Sid": "ELBListenerDelete",
            "Effect": "Allow",
            "Action": "elasticloadbalancing:DeleteListener",
            "Resource": "arn:*:elasticloadbalancing:*:*:listener/net/refactor-spaces-
nlb-*"
        },
        {
            "Sid": "ELBTargetGroupModify",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:targetgroup/refactor-spaces-tg-
*"
        },
        {
            "Sid": "ELBTargetGroupCreate",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:CreateTargetGroup"
```

```
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:targetgroup/refactor-spaces-tg-
*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:route-id": "false"
                }
            }
        },
        {
            "Sid": "APIGatewayModify",
            "Effect": "Allow",
            "Action": [
                "apigateway:GET",
                "apigateway:DELETE",
                "apigateway:PATCH",
                "apigateway:POST",
                "apigateway:PUT",
                "apigateway:UpdateRestApiPolicy"
            ],
            "Resource": [
                "arn:aws:apigateway:*::/restapis",
                "arn:aws:apigateway:*::/restapis/*",
                "arn:aws:apigateway:*::/vpclinks",
                "arn:aws:apigateway:*::/vpclinks/*",
                "arn:aws:apigateway:*::/tags",
                "arn:aws:apigateway:*::/tags/*"
            ],
            "Condition": {
                "Null": {
                    "aws:ResourceTag/refactor-spaces:application-id": "false"
                }
            }
        },
        {
            "Sid": "APIGatewayVpcLinksGet",
            "Effect": "Allow",
            "Action": "apigateway:GET",
            "Resource": [
                "arn:aws:apigateway:*::/vpclinks",
                "arn:aws:apigateway:*::/vpclinks/*"
            ]
        },
        {
```

```
            "Sid": "OrganizationDescribe",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudformationStackCreate",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudformationStackTag",
            "Effect": "Allow",
            "Action": [
                "cloudformation:TagResource"
            ],
            "Resource": "arn:aws:cloudformation:*:*:stack/*"
        },
        {
            "Sid": "CreateRefactorSpacesSLR",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "refactor-spaces.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CreateELBSLR",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
                }
            }
```

```
        }
    ]
}
```

AWS managed policy: AWSMigrationHubRefactorSpaces-SSMAutomationPolicy

To grant the permissions that are required to run SSM Automation, use this AWS managed policy in the IAM service role passed to the `AWSRefactorSpaces-CreateResources` automation document. This policy grants read and write access to tags to track the progress of the automation. When the Refactor Spaces environment's network bridge is enabled, the automation also adds the environment's security group to the Amazon EC2 instance to permit traffic from other Refactor Spaces services in the environment. This policy also grants access to the SSM parameters of the Application Migration Service post-launch action.

> **Important**
>
> When you use the `AWSMigrationHubRefactorSpaces-SSMAutomationPolicy` managed policy, the role must also use either AWSMigrationHubRefactorSpacesFullAccess or AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess, along with the extra required permissions that are described under these two policies.

Permissions details

The `AWSMigrationHubRefactorSpaces-SSMAutomationPolicy` policy includes the following permissions.

- `ec2:DescribeInstanceStatus` – required to validate that the Amazon EC2 instance exists.
- `ec2:CreateTags` and `ec2:DeleteTags` – required for tagging the Amazon EC2 instance. Tagging is needed for the automation to check if the script has already run against the Amazon EC2 instance. Deletion is needed for rollback in case of errors.
- `ec2:DescribeInstances` – required for the script to fetch all the security groups that are attached to an instance.
- `ec2:ModifyInstanceAttribute` – required when the Refactor Spaces environment's network bridge is enabled. This permission allows the script to add the environment's security group to the Amazon EC2 instance to permit traffic from other Refactor Spaces services in the environment.

- `ssm:GetParameters` – required to get the user-provided input values that are stored in the SSM parameter store.

The following is the `AWSMigrationHubRefactorSpaces-SSMAutomationPolicy` that you need to use in the IAM role that you pass to the SSM automation document `AWSRefactorSpaces-CreateResources` to grant the permissions that are required to run the automation.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstances"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyInstanceAttribute"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/refactor-spaces:ssm:optin": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyInstanceAttribute"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
```

```
                "ec2:DeleteTags"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/refactor-spaces:ssm:optin": "true"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "refactor-spaces:ssm:environment-id"
                }
            }
        },
        {
            "Action": "ssm:GetParameters",
            "Resource": "arn:aws:ssm:*:*:parameter/
ManagedByAWSApplicationMigrationService-*",
            "Effect": "Allow"
        }
    ]
}
```

Refactor Spaces updates to AWS managed policies

View details about updates to AWS managed policies for Refactor Spaces since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Refactor Spaces Document history page.

Change	Description	Date
Updated the AWSMigrationHubRefactorSpacesFullAccess and AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess policies to allow the `cloudformation:TagResource` action.	To accommodate a change in AWS CloudFormation, the two policies now allow the `cloudformation:TagResource` action.	April 11, 2024

Change	Description	Date
AWSMigrationHubRefactorSpaces-SSMAutomationPolicy – New policy	To grant the permissions that are required to run SSM Automation, use this AWS managed policy in the IAM service role passed to the `AWSRefactorSpaces-CreateResources` automation document.	August 10, 2023
Changed the resource element in statements that have the following action element: `"Action": ["elasticloadbalancing:AddTags", "elasticloadbalancing:CreateListener"]` This change affects AWSMigrationHubRefactorSpacesFullAccess, AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess, and the section called "MigrationHubRefactorSpacesServiceRolePolicy".	Updated NLB permissions to work with ELBv2 IAM changes.	July 20, 2023

Change	Description	Date
AWS managed policy: AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess – Added the `AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess` managed policy that you use when creating environments without a Transit Gateway.	Use the `AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess` policy when you create environments without a network bridge. Since you are using your own network infrastructure, the modified policy Transit Gateway permissions and Amazon EC2 security groups related to Transit Gateway actions.	April 3, 2023
MigrationHubRefactorSpacesServiceRolePolicy – Added the Elastic Load Balancing `DeregisterTargets` permission to the policy.	`MigrationHubRefactorSpacesServiceRolePolicy` provides access to AWS resources managed or used by AWS Migration Hub Refactor Spaces. The `AWSServiceRoleForMigrationHubRefactorSpaces` service-linked role uses this policy.	October 28, 2022
AWSMigrationHubRefactorSpacesFullAccess – Added Elastic Load Balancing tagging permissions.	The `AWSMigrationHubRefactorSpacesFullAccess` policy grants full access to Refactor Spaces, the Refactor Spaces console features and other related AWS services.	October 6, 2022

Change	Description	Date
AWSMigrationHubRefactorSpacesFullAccess – Removed the permission for creating tags for Amazon EC2 instances. This permission was added to Extra required permissions for Refactor Spaces.	The `AWSMigrationHubRefactorSpacesFullAccess` policy grants full access to Refactor Spaces, the Refactor Spaces console features and other related AWS services.	March 21, 2022
AWSMigrationHubRefactorSpacesFullAccess – New policy made available at launch	The `AWSMigrationHubRefactorSpacesFullAccess` policy grants full access to Refactor Spaces, the Refactor Spaces console features and other related AWS services.	November 29, 2021
MigrationHubRefactorSpacesServiceRolePolicy – New policy made available at launch	`MigrationHubRefactorSpacesServiceRolePolicy` provides access to AWS resources managed or used by AWS Migration Hub Refactor Spaces. The `AWSServiceRoleForMigrationHubRefactorSpaces` service-linked role uses this policy.	November 29, 2021
Refactor Spaces started tracking changes	Refactor Spaces started tracking changes for its AWS managed policies.	November 29, 2021

Identity-based policy examples for AWS Migration Hub Refactor Spaces

By default, users and roles don't have permission to create or modify Refactor Spaces resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Refactor Spaces, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Migration Hub Refactor Spaces in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Refactor Spaces 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 with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the Refactor Spaces console

To access the AWS Migration Hub Refactor Spaces console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Refactor Spaces 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 (users or roles) with that policy.

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 they're trying to perform.

To ensure that users and roles can still use the Refactor Spaces console, also attach the Refactor Spaces `ConsoleAccess` or `ReadOnly` AWS managed policy to the entities. For more information, see Adding permissions to a user 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": "*"
        }
    ]
}
```

Troubleshooting AWS Migration Hub Refactor Spaces identity and access

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

Topics

I am not authorized to perform an action in Refactor Spaces

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 sign-in credentials.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but does not have the fictional `refactor-spaces:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 refactor-spaces:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `refactor-spaces:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Refactor Spaces.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Refactor Spaces. However, the action requires the service to have

permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Refactor Spaces resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Refactor Spaces supports these features, see How AWS Migration Hub Refactor Spaces works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Using service-linked roles for Refactor Spaces

AWS Migration Hub Refactor Spaces uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Refactor Spaces. Service-linked roles are predefined by Refactor Spaces and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Refactor Spaces easier because you don't have to manually add the necessary permissions. Refactor Spaces defines the permissions of its service-linked roles, and unless defined otherwise, only Refactor Spaces can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your Refactor Spaces resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for Refactor Spaces

Refactor Spaces uses the service-linked role named **AWSServiceRoleForMigrationHubRefactorSpaces** and associates it with the **MigrationHubRefactorSpacesServiceRolePolicy** IAM policy – Provides access to AWS resources managed or used by AWS Migration Hub Refactor Spaces.

The AWSServiceRoleForMigrationHubRefactorSpaces service-linked role trusts the following services to assume the role:

- `refactor-spaces.amazonaws.com`

The following is the Amazon Resource Name (ARN) for AWSServiceRoleForMigrationHubRefactorSpaces.

```
arn:aws:iam::111122223333:role/aws-service-role/refactor-spaces.amazonaws.com/
AWSServiceRoleForMigrationHubRefactorSpaces
```

Refactor Spaces uses the **AWSServiceRoleForMigrationHubRefactorSpaces** service-linked role when performing cross-account changes. This role must be present in your account to use Refactor Spaces. If it is not present, Refactor Spaces creates it during the following API calls:

- `CreateEnvironment`
- `CreateService`
- `CreateApplication`

- `CreateRoute`

You must have `iam:CreateServiceLinkedRole` permissions to create the service-linked role. If the service-linked role doesn't exist in your account and cannot be created, the `Create` calls will fail. You must create the service-linked role in the IAM console before using Refactor Spaces, unless you are using the Refactor Spaces console.

Refactor Spaces does not use the service-linked role when making changes in the current signed-in account. For example, when an application is created, Refactor Spaces updates all of the VPCs in the environment so that they can communicate with the newly added VPC. If the VPCs are in other accounts, Refactor Spaces uses the service-linked role and the `ec2:CreateRoute` permission to update the route tables in other accounts.

To further expand on the create application example, when creating an application, Refactor Spaces updates the route tables that are in the virtual private cloud (VPC) provided in the `CreateApplication` call. This way, the VPC can communicate with other VPCs in the environment.

The caller must have the `ec2:CreateRoute` permission that we use to update the route tables. This permission exists in the service-linked role, but Refactor Spaces does not use the service-linked role in the caller's account to gain this permission. Instead, the caller must have the `ec2:CreateRoute` permission. Otherwise, the call fails.

You cannot use the service-linked role to escalate your privileges. Your account must already have the permissions in the service-linked role to make the changes in the calling account. The `AWSMigrationHubRefactorSpacesFullAccess` managed policy, together with a policy that grants the extra required permissions, defines all of the necessary permissions to create Refactor Spaces resources. The service-linked role is a subset of these permissions that is used for specific cross-account calls. For more information about `AWSMigrationHubRefactorSpacesFullAccess`, see AWS managed policy: AWSMigrationHubRefactorSpacesFullAccess.

Tags

When Refactor Spaces creates resources in your account, they are tagged with the appropriate Refactor Spaces resource ID. For example, the Transit Gateway created from `CreateEnvironment` is tagged with the `refactor-spaces:environment-id` tag with the environment ID as the value. The API Gateway API created from `CreateApplication` is tagged with `refactor-`

`spaces:application-id` with the application ID as the value. These tags allow Refactor Spaces to manage these resources. If you edit or remove the tags, Refactor Spaces can no longer update or delete the resource.

MigrationHubRefactorSpacesServiceRolePolicy

The role permissions policy named MigrationHubRefactorSpacesServiceRolePolicy allows Refactor Spaces to complete the following actions on the specified resources:

Amazon API Gateway actions

- `apigateway:PUT`
- `apigateway:POST`
- `apigateway:GET`
- `apigateway:PATCH`
- `apigateway:DELETE`

Amazon Elastic Compute Cloud actions

- `ec2:DescribeNetworkInterfaces`
- `ec2:DescribeRouteTables`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeVpcEndpointServiceConfigurations`
- `ec2:DescribeTransitGatewayVpcAttachments`
- `ec2:AuthorizeSecurityGroupIngress`
- `ec2:RevokeSecurityGroupIngress`
- `ec2:DeleteSecurityGroup`
- `ec2:DeleteTransitGatewayVpcAttachment`

```
ec2:CreateRoute

ec2:DeleteRoute

ec2:DeleteTags

ec2:DeleteVpcEndpointServiceConfigurations
```

AWS Resource Access Manager actions

```
ram:GetResourceShareAssociations

ram:DeleteResourceShare

ram:AssociateResourceShare

ram:DisassociateResourceShare
```

Elastic Load Balancing; actions

```
elasticloadbalancing:DescribeTargetHealth

elasticloadbalancing:DescribeListener

elasticloadbalancing:DescribeTargetGroups

elasticloadbalancing:RegisterTargets

elasticloadbalancing:CreateLoadBalancerListeners

elasticloadbalancing:CreateListener

elasticloadbalancing:DeleteListener

elasticloadbalancing:DeleteTargetGroup

elasticloadbalancing:DeleteLoadBalancer

elasticloadbalancing:DeregisterTargets

elasticloadbalancing:AddTags

elasticloadbalancing:CreateTargetGroup
```

The following is the full policy showing which resources the preceding actions apply to:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:DescribeTransitGatewayVpcAttachments",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeTargetGroups",
                "ram:GetResourceShareAssociations"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteTransitGatewayVpcAttachment",
                "ec2:CreateRoute",
                "ec2:DeleteRoute",
                "ec2:DeleteTags",
                "ram:DeleteResourceShare",
                "ram:AssociateResourceShare",
                "ram:DisassociateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/refactor-spaces:environment-id": "false"
                }
            }
        },
        {
            "Effect": "Allow",
```

```
            "Action": "ec2:DeleteVpcEndpointServiceConfigurations",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/refactor-spaces:application-id": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:CreateLoadBalancerListeners",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteTargetGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/refactor-spaces:route-id": [
                        "*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:PUT",
                "apigateway:POST",
                "apigateway:GET",
                "apigateway:PATCH",
                "apigateway:DELETE"
            ],
            "Resource": [
                "arn:aws:apigateway:*::/restapis",
                "arn:aws:apigateway:*::/restapis/*",
                "arn:aws:apigateway:*::/vpclinks/*",
                "arn:aws:apigateway:*::/tags",
                "arn:aws:apigateway:*::/tags/*"
            ],
            "Condition": {
                "Null": {
```

```
                    "aws:ResourceTag/refactor-spaces:application-id": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "apigateway:GET",
            "Resource": "arn:aws:apigateway:*::/vpclinks/*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:DeleteLoadBalancer",
            "Resource": "arn:*:elasticloadbalancing:*:*:loadbalancer/net/refactor-
spaces-nlb-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:CreateListener"
            ],
            "Resource": ["arn:*:elasticloadbalancing:*:*:loadbalancer/net/refactor-
spaces-nlb-*", "arn:*:elasticloadbalancing:*:*:listener/net/refactor-spaces-nlb-*"],
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:route-id": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:DeleteListener",
            "Resource": "arn:*:elasticloadbalancing:*:*:listener/net/refactor-spaces-
nlb-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:targetgroup/refactor-spaces-tg-
*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DeregisterTargets"
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:targetgroup/refactor-spaces-tg-
*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/refactor-spaces:route-id": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:CreateTargetGroup"
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:targetgroup/refactor-spaces-tg-
*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:route-id": "false"
                }
            }
        }
    ]
}
```

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

Creating a service-linked role for Refactor Spaces

You don't need to manually create a service-linked role. When you create Refactor Spaces environment, application, service, or route resources in the AWS Management Console, the AWS CLI, or the AWS API, Refactor Spaces creates the service-linked role for you. For more information about creating a service-linked role for Refactor Spaces, see Service-linked role permissions for Refactor Spaces.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create Refactor Spaces environment, application, service, or route resources, Refactor Spaces creates the service-linked role for you again.

Editing a service-linked role for Refactor Spaces

Refactor Spaces does not allow you to edit the AWSServiceRoleForMigrationHubRefactorSpaces service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

Deleting a service-linked role for Refactor Spaces

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the Refactor Spaces service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete the Refactor Spaces resources used by AWSServiceRoleForMigrationHubRefactorSpaces, use the Refactor Spaces console to delete the resources, or use the delete API operations for the resources. For more information about the delete API operations, see Refactor Spaces API Reference.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForMigrationHubRefactorSpaces service-linked role. For more information, see Deleting a Service-Linked Role in the *IAM User Guide*.

Supported Regions for Refactor Spaces service-linked roles

Refactor Spaces supports using service-linked roles in all of the Regions where the service is available. For more information, see AWS Regions and Endpoints.

Compliance validation for AWS Migration Hub Refactor Spaces

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.

- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Access Refactor Spaces using an interface endpoint (AWS PrivateLink)

You can use AWS PrivateLink to create a private connection between your VPC and AWS Migration Hub Refactor Spaces (Refactor Spaces). You can access Refactor Spaces as if it were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to access Refactor Spaces.

You establish this private connection by creating an *interface endpoint*, powered by AWS PrivateLink. We create an endpoint network interface in each subnet that you enable for the interface endpoint. These are requester-managed network interfaces that serve as the entry point for traffic destined for Refactor Spaces.

For more information, see Access AWS services through AWS PrivateLink in the *AWS PrivateLink Guide*.

Considerations for Refactor Spaces

Before you set up an interface endpoint for Refactor Spaces, review Considerations in the *AWS PrivateLink Guide*.

Refactor Spaces supports making calls to all of its API actions through the interface endpoint.

VPC endpoint policies are not supported for Refactor Spaces. By default, full access to Refactor Spaces is allowed through the interface endpoint. Alternatively, you can associate a security group with the endpoint network interfaces to control traffic to Refactor Spaces through the interface endpoint.

Create an interface endpoint for Refactor Spaces

You can create an interface endpoint for Refactor Spaces using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Create an interface endpoint in the *AWS PrivateLink Guide*.

Create an interface endpoint for Refactor Spaces using the following service name:

```
com.amazonaws.region.refactor-spaces
```

If you enable private DNS for the interface endpoint, you can make API requests to Refactor Spaces using its default Regional DNS name. For example, `refactor-spaces.us-east-1.amazonaws.com`.

For information about the Refactor Spaces API, see the AWS Migration Hub Refactor Spaces API Reference.

Availability

Refactor Spaces currently supports VPC endpoints in the following AWS Regions.

Region Name	Region
US East (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2

Region Name	Region
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-north-1
South America (São Paulo)	sa-east-1

Working with other services

This section describes other AWS services that interact with Refactor Spaces.

Creating Refactor Spaces resources with CloudFormation

AWS Migration Hub Refactor Spaces is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you want (such as environments, applications, services, and routes), and AWS CloudFormation provisions and configures those resources for you.

When you use AWS CloudFormation, you can reuse your template to set up your Refactor Spaces resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

Refactor Spaces and CloudFormation templates

To provision and configure resources for Refactor Spaces and related services, you must understand AWS CloudFormation templates. Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your AWS CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use AWS CloudFormation Designer to help you get started with AWS CloudFormation templates. For more information, see What is AWS CloudFormation Designer? in the *AWS CloudFormation User Guide*.

Refactor Spaces supports creating environments, applications, services, and routes in AWS CloudFormation. For more information, including examples of JSON and YAML templates for environments, applications, services, and routes, see AWS Migration Hub Refactor Spaces in the *AWS CloudFormation User Guide*.

Template example

The following example template creates a virtual private cloud (VPC) and Refactor Spaces resources. When you choose to deploy an AWS CloudFormation template to create a demo refactor environment from the Getting started dialog box, the following template is deployed by the Refactor Spaces console.

Example YAML Refactor Spaces template

```
AWSTemplateFormatVersion: '2010-09-09'
Description: This creates resources in one account.
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.2.0.0/16
      Tags:
        - Key: Name
          Value: VpcForRefactorSpaces
  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: 10.2.1.0/24
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: RefactorSpaces Private Subnet (AZ1)
  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: 10.2.2.0/24
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: RefactorSpaces Private Subnet (AZ2)
  RefactorSpacesTestEnvironment:
    Type: AWS::RefactorSpaces::Environment
    DeletionPolicy: Delete
    Properties:
      Name: EnvWithMultiAccountServices
      NetworkFabricType: TRANSIT_GATEWAY
      Description: "This is a test environment"
  TestApplication:
    Type: AWS::RefactorSpaces::Application
    DeletionPolicy: Delete
    DependsOn:
      - PrivateSubnet1
```

```
      - PrivateSubnet2
    Properties:
      Name: proxytest
      EnvironmentIdentifier: !Ref RefactorSpacesTestEnvironment
      VpcId: !Ref VPC
      ProxyType: API_GATEWAY
      ApiGatewayProxy:
        EndpointType: "REGIONAL"
        StageName: "admintest"
  AdminAccountService:
    Type: AWS::RefactorSpaces::Service
    DeletionPolicy: Delete
    Properties:
      Name: AdminAccountService
      EnvironmentIdentifier: !Ref RefactorSpacesTestEnvironment
      ApplicationIdentifier: !GetAtt TestApplication.ApplicationIdentifier
      EndpointType: URL
      VpcId: !Ref VPC
      UrlEndpoint:
        Url: "http://aws.amazon.com"
  RefactorSpacesDefaultRoute:
    Type: AWS::RefactorSpaces::Route
    Properties:
      RouteType: "DEFAULT"
      EnvironmentIdentifier: !Ref RefactorSpacesTestEnvironment
      ApplicationIdentifier: !GetAtt TestApplication.ApplicationIdentifier
      ServiceIdentifier: !GetAtt AdminAccountService.ServiceIdentifier
      DefaultRoute:
        ActivationState: ACTIVE
  RefactorSpacesURIRoute:
    Type: AWS::RefactorSpaces::Route
    DependsOn: 'RefactorSpacesDefaultRoute'
    Properties:
      RouteType: "URI_PATH"
      EnvironmentIdentifier: !Ref RefactorSpacesTestEnvironment
      ApplicationIdentifier: !GetAtt TestApplication.ApplicationIdentifier
      ServiceIdentifier: !GetAtt AdminAccountService.ServiceIdentifier
      UriPathRoute:
        SourcePath: "/cfn-created-route"
        ActivationState: ACTIVE
        Methods: [ "GET" ]
```

Learn more about CloudFormation

To learn more about AWS CloudFormation, see the following resources:

- AWS CloudFormation
- AWS CloudFormation User Guide
- AWS CloudFormation API Reference
- AWS CloudFormation Command Line Interface User Guide

Logging Refactor Spaces API calls using AWS CloudTrail

AWS Migration Hub Refactor Spaces is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Refactor Spaces. CloudTrail captures all API calls for Refactor Spaces as events. The calls captured include calls from the Refactor Spaces console and code calls to the Refactor Spaces API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Refactor Spaces. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Refactor Spaces, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Refactor Spaces information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Refactor Spaces, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Refactor Spaces, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail

- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple Regions
- Receiving CloudTrail log files from multiple accounts

All Refactor Spaces actions are logged by CloudTrail and are documented in the Refactor Spaces API Reference. For example, calls to the `CreateEnvironment`, `GetEnvironment` and `ListEnvironments` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding Refactor Spaces log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

Sharing Refactor Spaces environments using AWS RAM

AWS Migration Hub Refactor Spaces integrates with AWS Resource Access Manager (AWS RAM) to enable resource sharing. AWS RAM is a service that enables you to share some Refactor Spaces resources with other AWS accounts or through AWS Organizations. With AWS RAM, you share resources that you own by creating a *resource share*. A resource share specifies the resources to share, and the consumers with whom to share them. Consumers can include:

- Specific AWS accounts inside or outside of its organization in AWS Organizations
- An organizational unit inside its organization in AWS Organizations

- Its entire organization in AWS Organizations

For more information about AWS RAM, see the *AWS RAM User Guide*.

For more information about sharing Refactor Spaces environments, see Step 3: Share your environment .

Quotas for AWS Migration Hub Refactor Spaces

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view a list of the quotas for AWS Migration Hub Refactor Spaces, see Refactor Spaces service quotas.

You can also view the quotas for Refactor Spaces by opening the Service Quotas console. In the navigation pane, choose **AWS services** and select **AWS Migration Hub Refactor Spaces**.

To request a quota increase, see Requesting a Quota Increase in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the limit increase form.

Document history for the Refactor Spaces User Guide

The following table describes the documentation releases for AWS Migration Hub Refactor Spaces (Refactor Spaces).

Change	Description	Date
IAM policy update	Update to the AWS Identity and Access Managemen t (IAM) `AWSMigrat ionHubRefactorSpac esFullAccess` and `AWSMigrationHubRef actorSpaces-Enviro nmentsWithoutBridg esFullAccess` policies. For more information, see Policy updates.	April 11, 2024
New AWS Migration Hub Refactor Spaces post-laun ch action in AWS Application Migration Service (AWS MGN)	AWS MGN now has a new post-launch action that you can use to create Refactor Spaces resources. For more information, see Tutorial: Using Refactor Spaces with AWS Application Migration Service.	August 29, 2023
New IAM managed policy	AWS Migration Hub Refactor Spaces now supports a new `AWSMigrationHubRef actorSpaces-SSMAut omationPolicy` AWS Identity and Access Management (IAM) managed policy. To grant the permissio ns that are required to run	August 10, 2023

	SSM Automation, use this AWS managed policy in the IAM service role passed to the `AWSRefactorSpaces-CreateResources` automation document. For more information, see AWSMigrationHubRefactorSpaces-SSMAutomationPolicy.	
New Regions	AWS Migration Hub Refactor Spaces now supports the following additional Regions: US West (N. California), Asia Pacific (Mumbai), Asia Pacific (Osaka), Asia Pacific (Seoul), Canada (Central), Europe (Paris), South America (São Paulo). For a complete list of the Refactor Spaces Regions and endpoints, see Refactor Spaces endpoints and quotas.	April 20, 2023
New Regions	AWS Migration Hub Refactor Spaces now supports the following additional Regions: US West (N. California), Asia Pacific (Mumbai), Asia Pacific (Osaka), Asia Pacific (Seoul), Canada (Central), Europe (Paris), South America (São Paulo). For a complete list of the Refactor Spaces Regions and endpoints, see Refactor Spaces endpoints and quotas.	April 20, 2023

New IAM managed policy	AWS Migration Hub Refactor Spaces now supports a new `AWSMigrationHubRefactorSpaces-EnvironmentsWithoutBridgesFullAccess` AWS Identity and Access Management (IAM) managed policy. You use this new managed policy when you create environments without a network bridge. For more information, see Policy updates.	April 3, 2023
Create environments without a network bridge	AWS Migration Hub Refactor Spaces now supports the creation of refactor environments without an AWS Transit Gateway based network bridge. With this flexibility, you can safely and incrementally refactor your application while you use your existing network infrastructure. For more information, see Tutorials on using your own network infrastructure with Refactor Spaces. Also, see CreateEnvironment and CreateRoute in the *AWS Migration Hub Refactor Spaces API Reference*.	March 6, 2023

Lambda function alias support	Refactor Spaces now supports the use of AWS Lambda function aliases as Refactor Spaces service endpoints . You can now use Lambda function aliases with Refactor Spaces to route traffic to specific versions of a Lambda function. For more informati on, see Create a service. Also, see CreateService and LambdaEndpointInput in the *AWS Migration Hub Refactor Spaces API Reference*.	December 15, 2022
Automatic DNS resolution	Refactor Spaces now supports automatic handling of IP address changes for Refactor Spaces services. With this feature, you can create services with Domain Name System (DNS) names in the URL. Refactor Spaces automatically re-resolves the DNS name when the DNS time-to-live (TTL) expires (or every 60 seconds for TTLs less than 60 seconds). For more information, see Refactor Spaces concepts and CreateRoute in the *AWS Migration Hub Refactor Spaces API Reference*.	November 16, 2022

IAM policy update	Update to the AWS Identity and Access Managemen t (IAM) `Migration HubRefactorSpacesS erviceRolePolicy` policy. For more information, see Policy updates.	October 28, 2022
IAM policy update	Update to the IAM `AWSMigrationHubRef actorSpacesFullAcc ess` managed policy. For more information, see Policy updates.	October 6, 2022
AWS PrivateLink support	You can now use AWS PrivateLink to securely access the Refactor Spaces API from your Amazon Virtual Private Cloud without using public IPs, and without requiring the traffic to traverse across the public internet. For more information, see Access Refactor Spaces using an interface endpoint (AWS PrivateLink). For information about the Refactor Spaces API, see the AWS Migration Hub Refactor Spaces API Reference.	July 19, 2022

Update Routes	You can now provision routes ahead of use by providing the option to activate and inactivate the route state. This allows you to fine-tune your routing approach based on the timing of increment ally refactoring your applicati ons. For more information, see How Refactor Spaces works in this user guide, and CreateRoute and UpdateRou te in the *AWS Migration Hub Refactor Spaces API Reference*.	June 22, 2022
IAM policy update	Update to the IAM `AWSMigrationHubRef actorSpacesFullAcc ess` managed policy. For more information, see Policy updates.	March 21, 2022
GA release	General availability release of Refactor Spaces.	February 9, 2022
Initial release	Initial preview release of the Refactor Spaces User Guide.	November 29, 2021