AWS Migration Hub Refactor Spaces

User Guide

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 evolving applications into microservices or extending existing applications with new features written in microservices.

The Refactor Spaces environment simplifies cross-account networking by orchestrating AWS Transit Gateway, AWS Resource Access Manager, and virtual private clouds (VPCs). Refactor Spaces bridges networking across AWS accounts to permit earlier and newer services to communicate while maintaining the independence of separate AWS accounts.

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. You can also incrementally route traffic to the new services. This keeps underlying architecture changes transparent for your application consumers. For more information about the Strangler Fig pattern, see Strangler Fig Application.

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 (p. 2)
- How Refactor Spaces works (p. 3)
- Setting up (p. 4)
- Getting started with Refactor Spaces (p. 6)

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

The Refactor Spaces environment provides a unified view of networking, applications, and services across multiple AWS accounts.

A Refactor Spaces environment contains Refactor Spaces applications and services. It’s a multi-account network fabric consisting of bridged virtual private clouds (VPCs), which allows resources within it to interact through private IP addresses. The environment provides a unified view of networking, applications, and services across multiple AWS accounts.

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.

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.

How Refactor Spaces works

When starting to use AWS Migration Hub Refactor Spaces you can use one or more AWS accounts. You can use a single account for testing. However, once you're ready to start refactoring, 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*, in which Refactor Spaces configures cross-account networking and routes traffic.

First, you create a Refactor Spaces environment in the account chosen as the environment owner. Then, you share the environment with the other two accounts using AWS Resource Access Manager (the Refactor Spaces console does this for you). After you share the environment with another account, Refactor Spaces automatically shares the resources that it creates within the environment with the other accounts. It does so by orchestrating AWS Identity and Access Management (IAM) resource-based policies.

The refactor environment provides unified networking across accounts by orchestrating AWS Transit Gateway, AWS Resource Access Manager, and virtual private clouds (VPCs). 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 running 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's 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 created, the default route defaults to an active state. You can stop sending traffic to the default route by toggling the route to the inactive state. 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 you launch in a service VPC can communicate directly with all other service VPCs added to the environment. You can apply additional cross-account routing constraints using VPC security groups. When creating routes that point to services with Lambda endpoints, Refactor Spaces orchestrates Amazon API Gateway's Lambda integration to call the function across AWS accounts.

Setting up

Before you use AWS Migration Hub Refactor Spaces for the first time, complete the following tasks:

Sign up for AWS

In this section, you sign up for an AWS account. If you already have an AWS account, skip this step.

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all AWS services, including AWS Migration Hub Refactor Spaces. You are charged only for the services that you use.

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

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

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

Create IAM users

When you create an AWS account, you get a single sign-in identity that has complete access to all of the AWS services and resources in the account. This identity is called the AWS account *root user*. Signing in to the AWS Management Console using the email address and password that you used to create the account gives you complete access to all of the AWS resources in your account.

We strongly recommend that you *not* use the root user for everyday tasks, even the administrative ones. Instead, follow the security best practice Create Individual IAM Users and create an AWS Identity and Access Management (IAM) administrator user. Then, securely lock away the root user credentials and use them to perform only a few account and service management tasks.

In addition to creating an administrative user, you must also create non-administrative IAM users. The following topics explain how to create both types of IAM users.

Topics

Creating an IAM administrative user

By default, an administrator account inherits the `AWSMigrationHubRefactorSpacesFullAccess` managed policy required for accessing AWS Migration Hub Refactor Spaces.

To create an administrator user

- Create an administrator user in your AWS account. For instructions, see Creating Your First IAM User and Administrators Group in the *IAM User Guide*.

Creating an IAM non-administrative user

This section describes how to grant the necessary permissions required for using Refactor Spaces for a non-administrative user.

Before using Refactor Spaces, create a user with the `AWSMigrationHubRefactorSpacesFullAccess` managed policy and then attach the policy that grants the extra required permissions necessary to use Refactor Spaces to the user. This extra required permissions policy is described in Extra required permissions for Refactor Spaces (p. 21).

When creating non-administrative IAM users, follow the security best practice Grant Least Privilege and grant users minimum permissions.

To create a non-administrator IAM user to use with Refactor Spaces

1. In AWS Management Console, navigate to the IAM console.
2. Create a non-administrator IAM user by following the instructions for creating a user with the console as described in Creating an IAM user in your AWS account in the *IAM User Guide*.

   While following the instructions in the *IAM User Guide*:

   - When on the step about selecting the type of access, select both **Programmatic access** and **AWS Management Console access**.
   - When on the step about the **Set permission** page, choose the option to **Attach existing policies to user directly**. Then, select the managed IAM policy **AWSMigrationHubRefactorSpacesFullAccess**.
   - When on the step about viewing the user's access keys (access key IDs and secret access keys), follow the guidance in the **Important** note about saving the user's new access key ID and secret access key in a safe and secure place.
3. After creating the user, add the extra required permissions policy to the user following the directions to embed an inline policy for a user described in Adding IAM identity permissions in the *IAM User Guide*. This extra required permissions policy is described in Extra required permissions for Refactor Spaces (p. 21).

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 AWS Identity and Access Management (IAM) users set up for these accounts. For more information, see Setting up (p. 4).
- Designate one of the IAM user accounts as the Refactor Spaces environment owner account.

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 environment simplifies multi-account use cases to accelerate application refactoring. When you create an environment, we orchestrate AWS Transit Gateway, virtual private clouds (VPCs), and AWS Resource Access Manager in your account.

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 are able to create applications, services, and routes within the environment, unless you use IAM to restrict access.

To create an environment

1. Using the AWS account that you created in Setting up (p. 4), 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. 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 (p. 30).
9. 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 provide multi-account traffic routing for services in the application. For each application, we orchestrate a proxy using Amazon API Gateway VPC links, a Network Load Balancer, and resource policies. Applications are containers of services and routes.

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.

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.
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 are able to create applications, services, and routes within the environment, unless you use 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.
6. 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.

        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.
7. a. Select Lambda if the service is a Lambda function.

   b. Choose a Lambda function from your account.

8. (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.

Note
A route is live immediately after being created and traffic is redirected away from the default route or an existing parent route.

To create a route

On the application details page (the page with the application's name as the heading), under **Routes**, choose **Create route**.

1. Choose a service for the route.
2. Choose **Create route**.

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. This content includes 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 user accounts with 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 recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.

- 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 personal data that is stored in Amazon S3.
- If you require FIPS 140-2 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) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form 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 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 (p. 28).

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 employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Refactor Spaces, see How AWS Migration Hub Refactor Spaces works with IAM (p. 15).

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 (p. 26).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see Signing in to the AWS Management Console as an IAM user or root user in the *IAM User Guide*.

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

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

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

AWS account root user

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

IAM users and groups

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

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

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

IAM roles

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

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

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated users and roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **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.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for AWS Migration Hub Refactor Spaces in the *Service Authorization Reference*.
  - **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 Creating 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 IAM 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 Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. 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**.

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

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

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, 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 Creating IAM policies in the *IAM User Guide*.

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

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. 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 entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

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

How 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 (p. 16)	Yes
Resource-based policies (p. 16)	Yes
Policy actions (p. 17)	Yes
Policy resources (p. 17)	Yes
Policy condition keys (p. 18)	Yes
ACLs (p. 18)	No
ABAC (tags in policies) (p. 18)	Partial

IAM feature	Refactor Spaces support
Temporary credentials (p. 19)	Yes
Principal permissions (p. 19)	Yes
Service roles (p. 19)	No
Service-linked roles (p. 19)	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 Creating IAM 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 (p. 26).

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 How IAM roles differ from resource-based policies 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 (p. 26).

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 (p. 26).

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 (p. 26).

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.

For more information about ABAC, see What is ABAC? 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 Switching to a 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 principal permissions	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, resources, and condition keys for AWS Migration Hub Refactor Spaces in the *Service Authorization Reference*.

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 Creating 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 IAM 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 your IAM identities.

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 IAM user account full access to Refactor Spaces.
- `ec2` – Allows the IAM user account to perform Amazon Elastic Compute Cloud (Amazon EC2) operations used by Refactor Spaces.
- `elasticloadbalancing` – Allows the IAM user account to perform Elastic Load Balancing operations used by Refactor Spaces.
- `apigateway` – Allows the IAM user account to perform Amazon API Gateway operations used by Refactor Spaces.
- `organizations` – Allows the IAM user account to AWS Organizations operations used by Refactor Spaces.
- `cloudformation` – Allows the IAM user account 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 IAM user account, 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 assigned to an IAM user, group, or role in your account.

- 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 adding inline policies to your IAM user, group, or role. However, instead of using inline polices you can create an IAM policy using the following policy JSON, and attach it to the IAM user, group or 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": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcs",
                "ec2:DescribeTransitGatewayVpcAttachments",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeTags",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeInternetGateways"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTransitGateway",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTransitGatewayVpcAttachment"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:environment-id": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTransitGateway",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTransitGatewayVpcAttachment"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/refactor-spaces:environment-id": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpointServiceConfiguration"
            ],
            "Resource": "*"
        },
        {
            "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"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteVpcEndpointServiceConfigurations",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/refactor-spaces:application-id": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateLoadBalancer"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:application-id": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTags",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners"
            ],
            "Resource": "*"
        },
        {
            "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": "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-*",
            "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:AddTags",
                "elasticloadbalancing:CreateTargetGroup"
            ],
            "Resource": "arn:*:elasticloadbalancing:*:*:targetgroup/refactor-spaces-tg-*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/refactor-spaces:route-id": "false"
                }
            }
        },
        {
            "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"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "apigateway:GET",
            "Resource": [
                "arn:aws:apigateway:*::/vpclinks",
                "arn:aws:apigateway:*::/vpclinks/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "refactor-spaces.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
                }
            }
        }
    ]
}
```

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
AWSMigrationHubRefactorSpacesFullAccess (p. 20) – Removed the permission for creating tags for Amazon EC2 instances. This permission was added to Extra required permissions for Refactor Spaces (p. 21).	The `AWSMigrationHubRefactorSpacesFullAccess` policy grants full access to Refactor Spaces, the Refactor Spaces console features and other related AWS services.	March 21, 2022
AWSMigrationHubRefactorSpacesFullAccess (p. 20) – 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 (p. 30) – New policy made available at launch	`MigrationHubRefactorSpacesServiceRolePolicy` provides access to AWS resources managed or used by AWS Migration Hub Refactor Spaces. The policy is used by the AWSServiceRoleForMigrationHubRefactorSpaces service-linked role.	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, IAM users and roles don't have permission to create or modify Refactor Spaces resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform actions on the resources that they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

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

Topics

Policy best practices

Identity-based policies are very powerful. They 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 using AWS managed policies** – To start using Refactor Spaces quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get started using permissions with AWS managed policies in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant least privilege in the *IAM User Guide*.
- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

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 (IAM 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 you'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 user name and password.

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, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies 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 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 asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

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

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

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

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

I'm an administrator and want to allow others to access Refactor Spaces

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

To get started right away, see Creating your first IAM delegated user and group in the *IAM User Guide*.

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 (p. 15).
- 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 How IAM roles differ from resource-based policies 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 (p. 20).

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: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-*",
            "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: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 (p. 30).

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

Third-party auditors assess the security and compliance of AWS Migration Hub Refactor Spaces as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. 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 Refactor Spaces 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 and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – AWS Config; 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 that helps you check your compliance with security industry standards and best practices.

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 (Oregon)	us-west-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Stockholm)	eu-north-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 AWS Identity and Access Management (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 (p. 7).

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).

update-history-change	update-history-description	update-history-date
AWS PrivateLink support (p. 43)	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 (p. 43)	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 incrementally refactoring your applications. For more information, see How Refactor Spaces works in this user guide, and CreateRoute and UpdateRoute in the *AWS Migration Hub Refactor Spaces API Reference*.	June 22, 2022
IAM policy update (p. 43)	Update to the AWS Identity and Access Management (IAM) `AWSMigrationHubRefactorSpacesFullAccess` managed policy. For more information, see Policy updates.	March 21, 2022
GA release (p. 43)	General availability release of Refactor Spaces.	February 9, 2022
Initial release (p. 43)	Initial preview release of the Refactor Spaces User Guide.	November 29, 2021