aws

User Guide

# Tagging AWS Resources

**Version 1.0**

# Tagging AWS Resources: User Guide

# Table of Contents

# Tagging your AWS resources

Tags are key and value pairs that act as metadata for organizing your AWS resources. With most AWS resources, you have the option of adding tags when you create the resource. Examples of resources include an Amazon Elastic Compute Cloud (Amazon EC2) instance, an Amazon Simple Storage Service (Amazon S3) bucket, or a secret in AWS Secrets Manager.

> **Important**
>
> Do not store personally identifiable information (PII) or other confidential or sensitive information in tags. We use tags to provide you with billing and administration services. Tags are not intended to be used for private or sensitive data.

Tags can help you manage, identify, organize, search for, and filter resources. You can create tags to categorize resources by purpose, owner, environment, or other criteria.

Each tag has two parts:

- A *tag key* (for example, `CostCenter`, `Environment`, or `Project`). Tag keys are case sensitive.
- A *tag value* (for example, `111122223333` or `Production`). Like tag keys, tag values are case sensitive.

## How to add tags to your AWS resources

There are three ways to add tags to your AWS resources:

- **AWS service API operation** – The tagging API operations supported directly an AWS service. To discover what tagging functionality each AWS service provides, see the service's documentation in the AWS documentation index.
- **Tag Editor console** – Some services also support tagging with the AWS Tag Editor console.
- **Resource Groups Tagging API** – Most services also support tagging using the AWS Resource Groups Tagging API.

> **Note**
>
> You can also use AWS Service Catalog TagOptions Library to easily manage tags on provisioned products. A *TagOption* is a key-value pair managed in Service Catalog. It is not an AWS tag, but serves as a template for creating an AWS tag based on the TagOption.

You can tag resources for all cost-accruing services in AWS. For the following services, AWS recommends newer alternative AWS services that support tagging to better meet customer use cases.

| | | |
|---|---|---|
| Amazon Cloud Directory | Amazon CloudSearch | Amazon Cognito Sync |
| AWS Data Pipeline | Amazon Elastic Transcoder | Amazon Machine Learning |
| AWS OpsWorks Stacks | Amazon S3 Glacier Direct | Amazon SimpleDB |
| Amazon WorkSpaces Application Manager | AWS DeepLens | |

## Best practices

As you create a tagging strategy for AWS resources, follow best practices:

- Do not add personally identifiable information (PII) or other confidential or sensitive information in tags. Tags are accessible to many AWS services, including billing. Tags are not intended to be used for private or sensitive data.
- Use a standardized, case-sensitive format for tags, and apply it consistently across all resource types.
- Consider tag guidelines that support multiple purposes, like managing resource access control, cost tracking, automation, and organization.
- Use automated tools to help manage resource tags. Tag Editor and the Resource Groups Tagging API enable programmatic control of tags, making it easier to automatically manage, search, and filter tags and resources.
- Use too many tags rather than too few tags.

- Remember that it is easy to change tags to accommodate changing business requirements, but consider the consequences of future changes. For example, changing access control tags means you must also update the policies that reference those tags and control access to your resources.
- You can automatically enforce the tagging standards that your organization chooses to adopt by creating and deploying tag policies using AWS Organizations. Tag policies let you specify tagging rules that define valid key names and the values that are valid for each key. You can choose to only monitor, giving you an opportunity to evaluate and clean up your existing tags. Once your tags are in compliance with your chosen standards, you can then turn on enforcement in the tag policies to prevent non-compliant tags from being created. For more information, see Tag policies in the *AWS Organizations User Guide*.

# Tagging categories

Companies that are most effective in their use of tags typically create business-relevant tag groupings to organize their resources along technical, business, and security dimensions. Companies that use automated processes to manage their infrastructure also include additional, automation-specific tags.

| Technical tags | Tags for automation | Business tags | Security tags |
| --- | --- | --- | --- |
| • **Name –** Identify individual resources<br>• **Application ID –** Identify resources that are related to a specific application<br>• **Application Role –** Describe the function of a particular resource (such as web server, message broker, database) | • **Date/Time –** Identify the date or time a resource should be started, stopped, deleted, or rotated<br>• **Opt in/Opt out –** Indicate whether a resource should be included in an automated activity such as starting, stopping, or resizing instances<br>• **Security –** Determine | • **Project –** Identify projects that the resource supports<br>• **Owner –** Identify who is responsible for the resource<br>• **Cost Center/ Business Unit –** Identify the cost center or business unit associated with a resource, typically for cost allocation and tracking | • **Confidentiality –** An identifier for the specific data confidentiality level a resource supports<br>• **Compliance –** An identifier for workloads that must adhere to specific compliance requirements |

| Technical tags | Tags for automation | Business tags | Security tags |
| --- | --- | --- | --- |
| • **Cluster –** Identify resource farms that share a common configuration and perform a specific function for an application<br>• **Environment –** Distinguish between development, test, and production resources<br>• **Version –** Help distinguish between versions of resources or applications | requirements, such as encryption or enabling of Amazon VPC flow logs; identify route tables or security groups that need extra scrutiny | • **Customer –** Identify a specific client that a particular group of resources serves | |

# Tag naming limits and requirements

The following basic naming and usage requirements apply to tags:

- Each resource can have a maximum of 50 user created tags.
- System created tags that begin with `aws:` are reserved for AWS use, and do not count against this limit. You can't edit or delete a tag that begins with the `aws:` prefix.
- For each resource, each tag key must be unique, and each tag key can have only one value.
- The tag key must be a minimum of 1 and a maximum of 128 Unicode characters in UTF-8.
- The tag value must be a minimum of 0 and a maximum of 256 Unicode characters in UTF-8.
- Allowed characters can vary by AWS service. For information about what characters you can use to tag resources in a particular AWS service, see its documentation. In general, the allowed characters are letters, numbers, spaces representable in UTF-8, and the following characters: _ . : / = + - @.

- Tag keys and values are case sensitive. As a best practice, decide on a strategy for capitalizing tags, and consistently implement that strategy across all resource types. For example, decide whether to use `Costcenter`, `costcenter`, or `CostCenter`, and use the same convention for all tags. Avoid using similar tags with inconsistent case treatment.

# Common tagging strategies

Use the following tagging strategies to help identify and manage AWS resources.

**Contents**



## Tags for resource organization

Tags are a good way to organize AWS resources in the AWS Management Console. You can configure tags to be displayed with resources, and can search and filter by tag. With the AWS Resource Groups service, you can create groups of AWS resources based on one or more tags or portions of tags. You can also create groups based on their occurrence in an AWS CloudFormation stack. Using Resource Groups and Tag Editor, you can consolidate and view data for applications that consist of multiple services, resources, and Regions in one place.

## Tags for cost allocation

AWS Cost Explorer and detailed billing reports let you break down AWS costs by tag. Typically, you use business tags such as *cost center/business unit*, *customer*, or *project* to associate AWS costs with traditional cost-allocation dimensions. But a cost allocation report can include any tag. This lets you associate costs with technical or security dimensions, such as specific applications, environments, or compliance programs. The following is an example of a partial cost allocation report.

| Total Cost | user:Owner | user:Stack | user:Cost Center | user:Application |
|---|---|---|---|---|
| 0.95 | DbAdmin | Test | 80432 | Widget2 |
| 0.01 | DbAdmin | Test | 80432 | Widget2 |
| 3.84 | DbAdmin | Prod | 80432 | Widget2 |
| 6.00 | DbAdmin | Test | 78925 | Widget1 |
| 234.63 | SysEng | Prod | 78925 | Widget1 |
| 0.73 | DbAdmin | Test | 78925 | Widget1 |
| 0.00 | DbAdmin | Prod | 80432 | Portal |
| 2.47 | DbAdmin | Prod | 78925 | Portal |

For some services, you can use an AWS-generated `createdBy` tag for cost allocation purposes, to help account for resources that might otherwise go uncategorized. The `createdBy` tag is available only for supported AWS services and resources. Its value contains data associated with specific API or console events. For more information, see AWS-Generated Cost Allocation Tags in the *AWS Billing and Cost Management User Guide*.

## Tags for automation

Resource or service-specific tags are often used to filter resources during automation activities. Automation tags are used to opt in or opt out of automated tasks or to identify specific versions of resources to archive, update, or delete. For example, you can run automated `start` or `stop` scripts that turn off development environments during nonbusiness hours to reduce costs. In this scenario, Amazon Elastic Compute Cloud (Amazon EC2) instance tags are a simple way to identify instances to opt out of this action. For scripts that find and delete stale, out-of-date, or rolling Amazon EBS snapshots, snapshot tags can add an extra dimension of search criteria.

## Tags for access control

IAM policies support tag-based conditions, letting you constrain IAM permissions based on specific tags or tag values. For example, IAM user or role permissions can include conditions to limit EC2 API calls to specific environments (such as development, test, or production) based on their tags. The same strategy can be used to limit API calls to specific Amazon Virtual Private Cloud (Amazon VPC) networks. Support for tag-based, resource-level IAM permissions is service specific. When you use tag-based conditions for access control, be sure to define and restrict who can modify the tags. For more information about using tags to control API access to AWS resources, see AWS services that work with IAM in the *IAM User Guide*.

# Tagging governance

An effective tagging strategy uses standardized tags and applies them consistently and programmatically across AWS resources. You can use both reactive and proactive approaches for governing tags in your AWS environment.

- **Reactive governance** is for finding resources that are not properly tagged using tools such as the Resource Groups Tagging API, AWS Config Rules, and custom scripts. To find resources manually, you can use Tag Editor and detailed billing reports.
- **Proactive governance** uses tools such as AWS CloudFormation, Service Catalog, tag policies in AWS Organizations, or IAM resource-level permissions to ensure standardized tags are consistently applied at resource creation.

  For example, you can use the AWS CloudFormation `Resource Tags` property to apply tags to resource types. In Service Catalog, you can add portfolio and product tags that are combined and applied to a product automatically when it is launched. More rigorous forms of proactive governance include automated tasks. For example, you can use the Resource Groups Tagging API to search an AWS environment's tags, or run scripts to quarantine or delete improperly tagged resources.

# Learn more

This page provides general information on tagging AWS resources. For more information about tagging resources in a particular AWS service, see its documentation. The following are also good sources of information about tagging:

- For information about the AWS Resource Groups Tagging API, see the *Resource Groups Tagging API Reference Guide*.
- For information about Tag Editor, see Tag Editor in this guide.
- For information about the tagging functionality each AWS service provides, see the service's documentation in the AWS documentation index.
- For information about using tags in IAM policies to help control who can view and interact with your AWS resources, see Controlling access to and for IAM users and roles using tags in the *IAM User Guide*.

# Using Tag Editor

Tags are key and value pairs that act as metadata for organizing your AWS resources. With most AWS resources, you have the option of adding tags when you create the resource. Examples of resources include an Amazon Elastic Compute Cloud (Amazon EC2) instance, an Amazon Simple Storage Service (Amazon S3) bucket, or a secret in AWS Secrets Manager. However, you can also add tags to multiple, supported resources at once by using Tag Editor. You build a query for resources of various types, and then add, remove, or replace tags for the resources in your search results. Tag-based queries assign an AND operator to tags, so any resource that matches the specified resource types and all specified tags is returned by the query.

> **Important**
>
> Do not store personally identifiable information (PII) or other confidential or sensitive information in tags. We use tags to provide you with billing and administration services. Tags are not intended to be used for private or sensitive data.

To add tags to—or edit or delete tags of—multiple resources at once, use Tag Editor. With Tag Editor, you search for the resources that you want to tag, and then manage tags for the resources in your search results.

**To start Tag Editor**

1. Sign in to the AWS Management Console.
2. Perform either of the following steps:
   - Choose **Services**. Then, under **Management & Governance**, choose **Resource Groups & Tag Editor**. In the navigation pane on the left, choose **Tag Editor**.
   - Use the direct link: AWS Tag Editor console.

Not all resources can have tags applied. For information about which resources Tag Editor supports, see the **Tag Editor tagging**column at Supported resource types in the *AWS Resource Groups User Guide*. If a resource type that you want to tag isn't supported, let AWS know by choosing **Feedback** in the lower left corner of the console window.

For information about permissions and roles that are required to tag resources, see Set up permissions.

**Topics**



# Tags and attribute-based access control

Tags can be an important part of your AWS access control strategy. For information about using tags as the attributes in an attribute-based access control (ABAC) strategy, see Controlling access to AWS resources using tags and Controlling access to and for IAM users and roles using tags, both in the *IAM User Guide*.

There is a comprehensive tutorial that shows how to grant access to different projects and groups using tags at IAM tutorial: Define permissions to access AWS resources based on tags in the *AWS Identity and Access Management User Guide*.

If you use a SAML-based identity provider (IdP) for single sign-in, you can attach tags to the assumed roles providing access to your users. For more information, see IAM tutorial: Use SAML session tags for ABAC in the *AWS Identity and Access Management User Guide*.

# Best practices for tag names

These are some best practices and naming conventions that we recommend that you use with your tags.

Key names for AWS tags are case sensitive so ensure that they are used consistently. For example, the tags keys `CostCenter` and `costcenter` are different. One tag key might be configured

as a cost allocation tag for financial analysis and reporting, and the other tag key might not be configured for the same use.

A number of tags are predefined by AWS or created automatically by various AWS services. Many *AWS generated tags* use key names that are all lowercase, with hyphens separating words in the name, and prefixes followed by colons to identify the source service for the tag. For example, see the following:

- `aws:ec2spot:fleet-request-id` is a tag that identifies the Amazon EC2 Spot Instance Request that launched the instance.
- `aws:cloudformation:stack-name` is a tag that identifies the AWS CloudFormation stack that created the resource.
- `elasticbeanstalk:environment-name` is a tag that identifies the application that created the resource.

Consider naming your tags using the following rules:

- Use all lowercase for the words.
- Use hyphens to separate words.
- Use a prefix followed by a colon to identify the organization name or abbreviated name.

For example, for a fictitious company named *AnyCompany*, you might define tags such as:

- `anycompany:cost-center` to identify the internal Cost Center code.
- `anycompany:environment-type` to identify whether the environment is development, test, or production.
- `anycompany:application-id` to identify the application that the resource was created for.

The prefix ensures that tags are clearly recognizable as defined by your organization and not by AWS or a third-party tool that you might be using. Using all lowercase with hyphens for separators avoids confusion about how to capitalize a tag name. For example, `anycompany:project-id` is simpler to remember than `ANYCOMPANY:ProjectID`, `anycompany:projectID`, or `Anycompany:ProjectId`.

# Getting started with Tag Editor

Tag Editor is one way to tag your resources. View the sections below to understand the prerequisites you must satisfy to use it.

## Prerequisites for working with Tag Editor

Before you get started working to tag your resources, be sure you have an active AWS account with existing resources and appropriate rights to tag resources and create groups.

**Topics**



### Sign up for an AWS account

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

**To sign up for an AWS account**

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

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

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

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

## Create a user with administrative access

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

### Secure your AWS account root user

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

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

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

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

### Create a user with administrative access

1. Enable IAM Identity Center.

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

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

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

### Sign in as the user with administrative access

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

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

### Assign access to additional users

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

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

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

## Create resources

You must have resources in your AWS account to tag. For more information about the supported resource types, see the **Tag Editor Tagging** column under Supported resource types in the *AWS Resource Groups User Guide*.

## Set up permissions

To make full use of Tag Editor, you might need additional permissions to tag resources or to see a resource's tag keys and values. These permissions are in the following categories:

- Permissions for individual services so that you can tag resources from those services and include them in resource groups.
- Permissions that are required to use the Tag Editor console.

If you're an administrator, you can provide permissions for your users by creating policies through the AWS Identity and Access Management (IAM) service. You first create IAM roles, users, or groups, and then apply the policies with the permissions that they need. For information about creating and attaching IAM policies, see Working with policies.

### Permissions for individual services

> **Important**
>
> This section describes permissions that are required if you want to tag resources **from other AWS service consoles and APIs**.

To add tags to a resource, you need the permissions required for the service to which the resource belongs. For example, to tag Amazon EC2 instances, you must have permissions to the tagging operations in that service's API, such as the Amazon EC2 CreateTags operation.

### Permissions required to use the Tag Editor console

To use the Tag Editor console to list and tag resources, the following permissions must be added to a user's policy statement in IAM. You can add either AWS managed policies that are maintained and kept up to date by AWS, or you can create and maintain your own custom policy.

### Using AWS managed policies for Tag Editor permissions

Tag Editor supports the following AWS managed policies that you can use to provide a predefined set of permissions to your users. You can attach these managed policies to any role, user, or group just as you would any other policy that you create.

**ResourceGroupsandTagEditorReadOnlyAccess**

This policy grants the attached IAM role or user permission to call the read-only operations for both AWS Resource Groups and Tag Editor. To read a resource's tags, you must also have permissions for that resource through a separate policy. Learn more in the following **Important** note.

**ResourceGroupsandTagEditorFullAccess**

This policy grants the attached IAM role or user permission to call any Resource Groups operation and the read and write tag operations in Tag Editor. To read or write a resource's tags, you must also have permissions for that resource through a separate policy. Learn more in the following **Important** note.

**Important**

The two previous policies grant permission to call the Tag Editor operations and use the Tag Editor console. However, you must also have permissions not only to invoke the operation, but also appropriate permissions to the specific resource whose tags you're trying to access. To grant that access to the tags, you must also attach one of the following policies:

- The AWS managed policy ReadOnlyAccess grants permissions to the read-only operations for every service's resources. AWS automatically keeps this policy up to date with new AWS services as they become available.
- Many services provide service-specific read-only AWS managed policies that you can use to limit access to only the resources provided by that service. For example, Amazon EC2 provides AmazonEC2ReadOnlyAccess.
- You can create your own policy that grants access to only the specific read-only operations for the few services and resources you want your users to access. This policy uses either an allowlist strategy or a denylist strategy.

  An allowlist strategy takes advantage of the fact that access is denied by default until you ***explicitly allow*** it in a policy. So, you can use a policy like the following example.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [ "tag:*" ],
            "Resource": "<ARNs of resources to allow tagging>"
        }
    ]
}
```

  Alternatively, you could use a denylist strategy that allows access to all resources except those that you explicitly block. This requires a separate policy that applies to the relevant users that allows access. The following example policy then denies access to the specific resources listed by the Amazon Resource Name (ARN).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
             "Action": [ "tag:*" ],
            "Resource": "<ARNs of resources to disallow tagging>"
        }
    ]
```

```
}
```

### Adding Tag Editor permissions manually

- `tag:*` (This permission allows all Tag Editor actions. If you instead want to restrict actions that are available to a user, you can replace the asterisk with a specific action, or with a comma-separated list of actions.)
- `tag:GetResources`
- `tag:TagResources`
- `tag:UntagResources`
- `tag:getTagKeys`
- `tag:getTagValues`
- `resource-explorer:*`
- `resource-groups:SearchResources`
- `resource-groups:ListResourceTypes`

> **Note**
>
> The `resource-groups:SearchResources` permission allows Tag Editor to list resources when you filter your search using tag keys or values.
> The `resource-explorer:ListResources` permission allows Tag Editor to list resources when you search resources without defining search tags.

### Granting permissions for using Tag Editor

To add a policy for using AWS Resource Groups and Tag Editor to a role, do the following.

1. Open the IAM console to the **Roles** page.
2. Find the role to which you want to grant Tag Editor permissions. Choose the role's name to open the role's **Summary** page.
3. On the **Permissions** tab, choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. Choose **Create policy**.

6. On the **JSON** tab, paste the following policy statement.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "tag:GetResources",
        "tag:TagResources",
        "tag:UntagResources",
        "tag:getTagKeys",
        "tag:getTagValues",
        "resource-explorer:*",
        "resource-groups:SearchResources",
        "resource-groups:ListResourceTypes"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> This example policy statement grants permissions to perform only Tag Editor actions.

7. Choose **Next: Tags** and then choose **Next: Review**.
8. Enter a name and description for the new policy. For example, **`AWSTaggingAccess`**.
9. Choose **Create policy**.

Now that the policy is saved in IAM, you can attach it to other principals, such as roles, groups, or users. For more information about how to add a policy to a principal, see Adding and removing IAM identity permissions in the *IAM User Guide*.

**Authorization and access control based on tags**

AWS services support the following:

- **Action-based policies** – For example, you can create a policy that allows users to perform `GetTagKeys` or `GetTagValues` operations, but no others.

- **Resource-level permissions in policies** – Many services support using ARNs to specify individual resources in the policy.
- **Authorization based on tags** – Many services support using resource tags in the condition of a policy. For example, you can create a policy that allows users full access to a group that has the same tag as the users. For more information, see What is ABAC for AWS? in the *AWS Identity and Access Management User Guide*.
- **Temporary credentials** – Users can assume a role with a policy that allows Tag Editor operations.

Tag Editor doesn't use any service-linked roles.

For more information about how Tag Editor integrates with AWS Identity and Access Management (IAM), see the following topics in the *AWS Identity and Access Management User Guide*:

- AWS services that work with IAM
- Actions, resources, and condition keys for Tag Editor
- Controlling access to AWS resources using policies

# Finding resources to tag

With Tag Editor, you build a query to find resources in one or more AWS Regions that are available for tagging. You can choose up to 20 individual resource types, or build a query on **All resource types**. Your query can include resources that already have tags, or resources that have no tags. For more information, see the **Tag Editor Tagging** column at Supported resource types in the *AWS Resource Groups User Guide*.

After you find resources to tag, you can use Tag Editor to add tags, or view, edit, or delete tags.

**To find resources to tag**

1. Open the Tag Editor console.
2. *(Optional)* Choose the AWS Regions in which to search for resources to tag. By default, your current Region is used. For this procedure, choose **us-east-1** and **us-west-2**.
3. Choose at least one resource type from the **Resource types** dropdown list. You can add or edit tags for up to 20 individual resource types at a time, or choose **All resource types**. For this procedure, choose **AWS::EC2::Instance** and **AWS::S3::Bucket**.

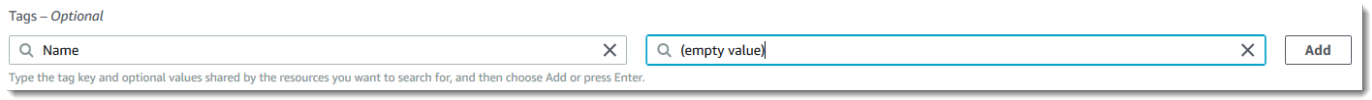
4. *(Optional)* In the **Tags** fields, enter a tag key, or a tag key and value pair, to limit the resources in the current AWS Region to only those that are tagged with your specified values. As you type a tag key, matching tag keys in the current Region appear in a list below. You can choose a tag key from the list. Tag Editor auto-completes the tag key for you as you type enough characters to match an existing key. Choose **Add** or press **Enter** when you've finished your tag. In this example, filter for resources that have a tag key of **Stage**. The tag value is optional but narrows the results of the query further. To add more tags, choose **Add**. Queries assign an AND operator to tags, so only resources that match both the specified resource type and all specified tags are returned by the query.

   > **Note**
   >
   > The Tag Editor console doesn't currently support wildcards.

   To find resources with multiple values for a tag key, add another tag with the same key to the query, but specify a different value. The results include all resources that are tagged with the same tag key and that have any of the selected values. The search is case sensitive.

   Leave the **Tags** boxes empty to find all resources of the specified type in the selected AWS Regions. This query returns resources that have any tags, and includes those that have no tags. To remove a tag from your query, choose **X** on the tag's label.

   To find resources that have a tag, but with an empty value, choose **(empty value)**, as shown below, when your cursor is in the tag value box.

   > **Note**
   >
   > Before you can find resources with the specified tags, they must have been applied to at least one resource of the specified type in the current AWS Region.

5. When your query is ready, choose **Search resources**. Results are displayed as a table in the **Resource search results** area.

   To filter a large number of resources, enter any filter text, such as part of the name of a resource, in **Filter resources**.

> **Note**
>
> You can use substrings to filter your results.

6. *(Optional)* To configure the columns that Tag Editor displays in your resource search results, choose the **Preferences** gear icon 
in the **Resource search results**.

   On the **Preferences** page, choose the number of rows that you want displayed in your search results. If you'd like to see all the text in the table, select the **Wrap lines** check box.

   Turn on columns that you want Tag Editor to display in your results. You can show a column for each tag that occurs in your search results or a selected subset of your search results. You can do this anytime after you find resources to tag. To enable a column, choose the switch icon next to the tag and change it from off 
to on 
.

   When you are finished configuring visible columns and number of displayed rows, choose **Confirm**.

## View and edit tags for a selected resource

Tag Editor shows you the existing tags on selected resources that are in the results of your **Find resources to tag** query.

If you enabled any **Tag** columns as described in the previous section, you can see the current value of that tag for each resource in the search results.

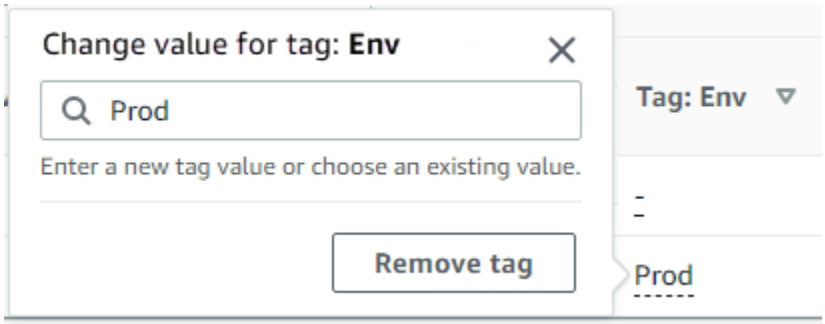
> **Note**
>
> This topic explains how to edit the tag for an *individual* resource. You can also bulk edit tags for several selected resources at the same time. For more information, see Managing tags with Tag Editor.

**To edit tags inline in the search results table**

1. Choose the value for the tag on the resource that you want to edit.

   > **Note**
   >
   > - If the chosen resource currently does not have a tag with the chosen key, the value displays as **(not tagged)**.
   > - If the chosen resource does have a tag with the chosen key but without a value, the value displays as '–'.

   In the following example, the column for the tag **Env** and with the current value of **Prod** was chosen.

2. You can enter a new value or choose from any of the values already present on other resources with this tag. You can also delete the tag from this one resource by choosing **Remove tag**.

**To view all tags for an individual resource**

1. In the results of your **Find resources to tag** query, choose the number in the **Tags** column for any resource for which you want to view existing tags. Resources with a dash in the **Tags** column do not have existing tags.
2. View existing tags in **Resource tags**. You can also open this window by choosing **Manage tags of selected resources**, when you're changing or removing tags from the **Manage tags** page.

   > **Note**
   >
   > If you don’t see a tag that you recently applied to a resource, try refreshing your browser window.

## Export results to .csv file

You can export the results of a **Find resources to tag** query to a comma-separated values (.csv) file. The .csv file includes the resource names, services, Region, resource IDs, the total number of tags, and a column for each unique tag key in the collection. The .csv file can help you develop a tagging strategy for resources in your organization, or determine where there are overlaps or inconsistencies in tagging across resources.

1. In the results of your **Find resources to tag** query, choose **Export resources to CSV**.
2. When you're prompted by your browser, choose to open the .csv file, or save it to a convenient location.

## Related information

- Using cost allocation tags in the *AWS Billing User Guide*

# Managing tags with Tag Editor

After you find resources that you want to tag, you can add, remove, and edit the tags for some or all of your search results. Tag Editor shows you any tags that are attached to resources. It also shows you whether those tags were added in Tag Editor, by the resource's service console, or by using the API.

**Important**

Do not store personally identifiable information (PII) or other confidential or sensitive information in tags. We use tags to provide you with billing and administration services. Tags are not intended to be used for private or sensitive data.

**Other ways to manage your tags**

This topic discusses tagging resources by using Tag Editor in the AWS Management Console. However, you can also manage the tags on your AWS resources by using the following tools:

- You can type or script commands at your shell prompt by using the `resourcegroupstaggingapi` commands in the AWS Command Line Interface (AWS CLI).
- You can create and run PowerShell scripts by using the AWS Resource Groups Tagging API in the AWS Tools for PowerShell Core.
- You can create and run programs with any of the available AWS SDKs by using the Resource groups tagging APIs, such as the tagging APIs for Python or the tagging APIs for Java.

When you add, remove, or edit existing tags, you're changing tags only on those resources that you select in the results of your **Find resources to tag** query. You can select up to 500 resources on which to manage tags.

**Topics**



## Add tags to selected resources

You can use Tag Editor to add tags to selected resources that are in the results of your **Find resources to tag** query.

> **Note**
>
> This topic describes how to bulk edit the tags for *multiple* resources. You can also edit the tag values for an individual resource. For more information, see View and edit tags for a selected resource.

1. Open the Tag Editor console, and submit a query that returns multiple resources that you want to tag.

2. In the results table of your **Find resources to tag** query, select the check boxes next to the resources that you want to add tags to. Enter a text string in **Filter resources** at the top of the table to filter for part of a resource's name, ID, tag keys, or tag values. In the **Tags** column, note that resources in the results already have tags applied to them.
3. Select the check box for one or more resources, and then choose **Manage tags of the selected resources**.
4. On the **Manage tags** page, shown below, view the tags on the resources that you selected. Although your original query returned more resources, you're adding tags to only those resources that you selected in step 1. Choose **Add tag**.
5. Enter a tag key and an optional tag value. For this procedure, you'll add the tag key **`Team`** and the tag value **`Development`**.

   > **Note**
   >
   > A resource can have a maximum of 50 user-applied tags. You might not be able to add new tags to a resource if you're approaching 50 user-applied tags. AWS generated tags don't apply to the 50-tag limit. Tag keys must also be unique within your selected resources. You can't add a new tag with a key that matches a tag key that already exists in your selected resources.

6. When you're finished adding tags, choose **Review and apply changes**.
7. If you accept the changes, choose **Apply changes to all selected**.
8. Depending on the number of resources you select, applying new tags can take a few minutes. Don't leave the page or open a different page in the same browser tab. If changes were successful, a green success banner is displayed at the top of the page. Wait for a success or failure banner to appear on the page before you continue.

   If tag changes to some or all resources were not successful, see Troubleshooting tag changes. After you resolve the unsuccessful tag changes (such as insufficient permissions), you can retry the tag changes on resources for which tag changes failed. For more information, see the section called “Retry failed tag changes”.

## Edit tags of selected resources

You can use Tag Editor to change existing tag values on selected resources that are in the results of your **Find resources to tag** query. Editing a tag changes the tag's value on all selected resources

that have the same tag key. You can't rename a tag key, but you can delete a tag and create a tag with a new name to replace the original tag key. This deletes all tags with that key on selected resources.

> **Important**
>
> Do not store personally identifiable information (PII) or other confidential or sensitive information in tags. We use tags to provide you with billing and administration services. Tags are not intended to be used for private or sensitive data.

1. In the results of your **Find resources to tag** query, select the check boxes next to the resources for which you want to change existing tags. Enter a text string in **Filter resources** to filter for part of a resource's name or ID. In the **Tags** column, note that resources in the results already have tags applied to them.
2. Choose **Manage tags of the selected resources**.
3. On the **Manage tags** page, in **Edit tags of selected resources**, view the tags on the resource that you selected. Although your original query might have returned more resources, you are changing tags for only those resources that you selected in step 1.
4. Change, add, or delete tag values. Existing tags must have a tag key, but tag values are optional. In this procedure, we change the value of the **`Team`** tag to **`QA`**.

   If resources in your selection have different values for the same key, **Selected resources have different tag values** is displayed in the **Tag value** field. In this case, placing your cursor in the box opens a dropdown list of all available values for this tag key in your selected resources.

   If resources in your selection have the tag value you want, the tag value is highlighted as you type it. For example, if resources in your selection already have the tag value **`QA`**, the value is highlighted as you type **`Q`**. The values in the dropdown list help keep tag values consistent across resources. The tag value is changed on all selected resources. In this example, the tag value is changed to **`QA`** for all selected resources that had a **`Team`** tag key. For selected resources that don't have the **`Team`** tag, the **`Team`** tag with the value **`QA`** is added.
5. When you're finished changing tags, choose **Review and apply changes**.
6. If you accept the changes, choose **Apply changes to all selected**.
7. Depending on the number of resources you selected, editing tags can take a few minutes. Don't leave the page or open a different page in the same browser tab. If changes were

successful, a green success banner is displayed at the top of the page. Wait for a success or failure banner to appear on the page before you continue.

If tag changes to some or all resources were not successful, see Troubleshooting tag changes. After you resolve the root causes of unsuccessful tag changes (such as insufficient permissions), you can retry tag changes on resources for which tag changes failed. For more information, see the section called “Retry failed tag changes”.

## Remove tags from selected resources

You can use Tag Editor to remove tags from selected resources that are in the results of your **Find resources to tag** query. Removing a tag deletes the tag from all selected resources that have the tag. Because you can't edit tag keys, you can remove tags and replace them with new tags if you need to edit a tag key. This deletes all tags with that key on selected resources.

1. In the results of your **Find resources to tag** query, select the check boxes next to the resources you want to remove tags from. Enter a text string in **Filter resources** to filter for part of a resource's name or ID.
2. Choose **Manage tags of the selected resources**.
3. On the **Manage tags** page, in **Edit tags of selected resources**, view the tags on the resources that you selected. Although your original query might have returned more resources, you're changing tags for only those resources that you selected in step 1.
4. Choose **Remove tag** next to any tags that you want to delete. In this procedure, we remove the **`Team`** tag.

   > **Note**
   >
   > Choosing **Remove tag** removes a tag from all selected resources that have the tag.

5. Choose **Review and apply changes**.
6. On the confirmation page, choose **Apply changes to all selected**.
7. Depending on the number of resources you selected, removing tags can take a few minutes. Don't leave the page or open a different page in the same browser tab. If changes were successful, a green success banner is displayed at the top of the page. Wait for a success or failure banner to appear on the page before you continue.

If tag changes to some or all resources were not successful, see Troubleshooting Tag Changes. After you resolve the root causes of unsuccessful tag changes (such as insufficient permissions), you can retry tag changes on resources for which tag changes failed. For more information, see the section called "Retry failed tag changes".

### Retry failed tag changes

If tag changes fail on at least one of your selected resources, Tag Editor displays a red banner at the bottom of the page. The banner shows an error message for each type of failure that occurs. For each error, the banner identifies the specific resources on which Tag Editor couldn't make tag changes. After you review and troubleshoot the errors, choose **Retry failed tag changes on resources** to retry changes on only those resources on which tag changes failed.

### Related information

- Using cost allocation tags in the *AWS Billing User Guide*

## Using tags in IAM permission policies

AWS Identity and Access Management (IAM) is the AWS service that you use to create and manage permissions policies that determine who can access your AWS resources. Every attempt to access an AWS service or read or write an AWS resource is access controlled by an IAM policy.

These policies allow you to provide granular access to your resources. One of the features you can use to fine tune this access is the `Condition` element of the policy. This element lets you specify the conditions that must match the request to determine if the request can proceed. Among the things you can check with the `Condition` element are the following:

- Tags that are attached to the user or role making the request.
- Tags attached to the resource that is the object of the request.

### Tag-related condition keys

The following table describes the condition keys that you can use in an IAM permissions policy to control access based on tags. These condition keys let you do the following:

- Compare the tags on the principal calling the operation.
- Compare the tags provided to the operation as a parameter.
- Compare the tags attached to the resource that would be accessed by the operation.

For complete details about a condition key and how to use it, see the page linked in the **Condition key name** column.

| Condition key name | Description |
| --- | --- |
| aws:PrincipalTag | Compares the tag attached to the principal (IAM role or user) making the request with the tag that you specify in the policy. |
| aws:RequestTag | Compares the tag key-value pair that was passed to the request as a parameter with the tag key-value pair that you specify in the policy. |
| aws:ResourceTag | Compares the key-value pair that is attached to the resource with the tag key-value pair that you specify in the policy. |
| aws:TagKeys | Compares only the tag *keys* in the request with the keys that you specify in the policy. |

## Example IAM policies that use tags

### Example Example 1: Force users to attach a specific tag when they create a resource

The following example IAM permissions policy shows how to force the user who creates or modifies an IAM policy's tags to include a tag with the key `Owner`. Also, the policy requires that the value of the tag is set to the same value as the `Owner` tag currently attached to the calling principal. For this strategy to work, all principals must have an `Owner` tag attached, and users must be prevented from modifying that tag. If an attempt to create or modify a policy occurs without including the `Owner` tag, the policy doesn't match and the operation isn't allowed.

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

```
            "Action": [
                "iam:CreatePolicy",
                "iam:TagPolicy"
            ],
            "Resource": "arn:aws:iam::123456789012:policy/*",
            "Condition": {
                "StringEquals": {"aws:RequestTag/Owner": "${aws:PrincipalTag/Owner}"}
            }
        }
    ]
}
```

**Example Example 2: Use tags to limit access to a resource to its "owner"**

The following example IAM permissions policy lets the user stop a running Amazon EC2 instance only if the calling principal is tagged with the same `project` tag value as the instance.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "ec2:StopInstances"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:instance/*"
      ],
      "Condition": {
        "StringEquals": {"aws:ResourceTag/project": "${aws:PrincipalTag/project}"}
      }
    }
  ]
}
```

This example is an example of attribute-based access control (ABAC). For more information and additional examples of using IAM policies to implement a tag-based access control strategy, see the following topics in the *AWS Identity and Access Management User Guide*:

- Controlling access to AWS resources using tags
- Controlling access to and for IAM users and roles using tags

- IAM tutorial: Define permissions to access AWS resources based on tags – Shows how to grant access to different projects and groups using multiple tags.

# AWS Organizations tag policies

A *tag policy* is a type of policy that you create in AWS Organizations. You can use tag policies to help standardize tags across the resources in your organization's accounts. To use tag policies, we recommend that you follow the workflows described in Getting started with tag policies in the *AWS Organizations User Guide*. As mentioned on that page, the recommended workflows include finding and correcting noncompliant tags. To accomplish these tasks, you use the Tag Editor console.

**Topics**



## Prerequisites and permissions

Before you can evaluate compliance with tag policies in Tag Editor, you must meet the requirements and set the necessary permissions.

### Prerequisites for evaluating compliance with tag policies

Evaluating compliance with tag policies requires the following:

- You must first enable the feature in AWS Organizations, and create and attach tag policies. For more information, see the following pages in the *AWS Organizations User Guide*:
  - Prerequisites and permissions for managing tag policies
  - Enabling tag policies
  - Getting started with tag policies
- To **find noncompliant tags on an account's resources**, you need sign-in credentials for that account and the permissions listed in Permissions for evaluating compliance for an account.
- To **evaluate organization-wide compliance**, you need sign-in credentials for the organization's management account and the permissions listed in Permissions for evaluating organization-

wide compliance . You can request the compliance report from only the AWS Region US East (N. Virginia) .

## Permissions for evaluating compliance for an account

Finding noncompliant tags on an account's resources requires the following permissions:

- `organizations:DescribeEffectivePolicy` – To get the contents of the effective tag policy for the account.
- `tag:GetResources` – To get a list of resources that don't comply with the attached tag policy.
- `tag:TagResources` – To add or update tags. You also need service-specific permissions to create tags. For example, to tag resources in Amazon Elastic Compute Cloud (Amazon EC2), you need permissions for `ec2:CreateTags`.
- `tag:UnTagResources` – To remove a tag. You also need service-specific permissions to remove tags. For example, to untag resources in Amazon EC2, you need permissions for `ec2:DeleteTags`.

The following example AWS Identity and Access Management (IAM) policy provides permissions for evaluating tag compliance for an account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EvaluateAccountCompliance",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeEffectivePolicy",
                "tag:GetResources",
                "tag:TagResources",
                "tag:UnTagResources"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about IAM policies and permissions, see the IAM User Guide.

## Permissions for evaluating organization-wide compliance

Evaluating organization-wide compliance with tag policies requires the following permissions:

- `organizations:DescribeEffectivePolicy` – To get the contents of the tag policy that's attached to the organization, organizational unit (OU), or account.
- `tag:GetComplianceSummary` – To get a summary of noncompliant resources in all accounts in the organization.
- `tag:StartReportCreation` – To export the results of the most recent compliance evaluation to a file. Organization-wide compliance is evaluated every 48 hours.
- `tag:DescribeReportCreation` – To check the status of report creation.

The following example IAM policy provides permissions for evaluating organization-wide compliance.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EvaluateOrgCompliance",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeEffectivePolicy",
                "tag:GetComplianceSummary",
                "tag:StartReportCreation",
                "tag:DescribeReportCreation"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about IAM policies and permissions, see the IAM User Guide.

## Amazon S3 bucket policy for report storage

To create an organization-wide compliance report, you must grant access for the tag policies service principal to an Amazon Simple Storage Service (Amazon S3) bucket in the US East (N. Virginia) Region for report storage. This Amazon S3 bucket must be in the same account that is

requesting to generate a compliance report. The role you use must also have `GetBucketAcl` and `s3:PutObject` permissions to the bucket.

Attach the following bucket policy to the bucket, replacing each *`placeholder`* with your own information:

- Your S3 bucket name
- ID number of the organization
- Account ID number of the organization's management account for the organization in which you're applying the policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TagPolicyACL",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "tagpolicies.tag.amazonaws.com"
                ]
            },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::<your-bucket-name>",
            "Condition": {
                "StringLike": {
                    "aws:SourceAccount": "<organization-management-account-id>",
                    "aws:SourceArn": "arn:aws:tag:us-east-1:<organization-management-
account-id>:*"
                }
            }
        },
        {
            "Sid": "TagPolicyBucketDelivery",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "tagpolicies.tag.amazonaws.com"
                ]
            },
            "Action": [
```

```
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::<your-bucket-name>/AwsTagPolicies/<your-
organization-id>/*",
            "Condition": {
                "StringLike": {
                    "aws:SourceAccount": "<organization-management-account-id>",
                    "aws:SourceArn": "arn:aws:tag:us-east-1:<organization-management-
account-id>:*"
                }
            }
        }
    ]
}
```

## Evaluating compliance for an account

You can evaluate the compliance of an account in your organization with its effective tag policy.

> **Important**
>
> Untagged resources don't appear as noncompliant in results.
> To find untagged resources in your account, use AWS Resource Explorer with a query that uses **`tag:none`**. For more information, see Search for untagged resources in the *AWS Resource Explorer User Guide*.

The *effective tag policy* specifies the tagging rules that apply to an account. The effective tag policy is the aggregation of any tag policies that the account inherits, plus any tag policy directly attached to the account. When you attach a tag policy to the organization root, it applies to all accounts in your organization. When you attach a tag policy to an organizational unit (OU), it applies to all accounts and OUs that belong to the OU.

> **Note**
>
> If you haven't yet created tag policies, see Getting started with tag policies in the *AWS Organizations User Guide*.

To find noncompliant tags, you must have the following permissions:

- `organizations:DescribeEffectivePolicy`
- `tag:GetResources`
- `tag:TagResources`
- `tag:UntagResources`

**To evaluate an account's compliance with its effective tag policy (console)**

1. While signed in to the account whose compliance you want to check, open the Tag Policies console.
2. The **Effective tag policy** section shows when the policy was last updated and the defined tag keys. You can expand a tag key to see information about its values, case treatment, and whether the values are enforced for specific resources types.

   > **Note**
   >
   > If you're signed in to the management account, you need to choose an account to see its effective policy and view compliance information.

3. In the **Resources with noncompliant tags** section, specify which AWS Region to search for noncompliant tags. Optionally, you can also search by resource type. Then choose **Search resources**.

   Real-time results are shown in the **Search results** section. To change the number of results returned per page or the columns to display, choose the settings icon (  ).
4. In the search results, select a resource with noncompliant tags.
5. In the dialog box that lists the resource's tags, choose the hyperlink to open the AWS service where the resource was created. From that console, correct the noncompliant tag.

   > **Tip**
   >
   > If you're not sure which tags are noncompliant, go to the **Effective tag policy** section for the account in the Tag Policies console. You can expand a tag key to view its tagging rules.

6. Repeat the process of finding and correcting tags until the account resources that you care about are compliant in each Region.

### To find noncompliant tags (AWS CLI, AWS API)

Use the following commands and operations to find noncompliant tags:

- AWS Command Line Interface (AWS CLI):
  - aws resourcegroupstaggingapi get-resources
  - aws resourcegroupstaggingapi tag-resources
  - aws resourcegroupstaggingapi untag-resources

  For the complete procedure for using tag policies in the AWS CLI, see Using tag policies in the AWS CLI in the *AWS Organizations User Guide*.
- AWS Resource Groups Tagging API:
  - GetResources
  - TagResources
  - UntagResources

### Next steps

We recommend that you repeat the process of finding and correcting compliance issues. Continue until the account's resources that you care about are compliant with the effective tag policy in each Region.

Finding and correcting noncompliant tags is an iterative process for multiple reasons, including the following:

- Your organization's use of tag policies can evolve over time.
- It takes time to effect change in your organization when creating resources.
- Compliance can change anytime that a new resource is created or when new tags are assigned to a resource.
- An account's effective tag policy is updated whenever a tag policy is attached to or detached from it. The effective tag policy is also updated whenever changes occur to tag the policies that the account inherits.

If you're signed in as the management account in the organization, you can also generate a report. This report shows information about all tagged resources in your organization's accounts. For more information, see Evaluating organization-wide compliance.

## Evaluating organization-wide compliance

You can evaluate your organization's compliance with its effective tag policy. You can generate a report that lists all tagged resources in accounts across your organization and whether each resource is compliant with the effective tag policy.

> **Important**
>
> Untagged resources don't appear as noncompliant in results.
> To find untagged resources in your account, use AWS Resource Explorer with a query that uses **`tag:none`**. For more information, see Search for untagged resources in the *AWS Resource Explorer User Guide*.

You can generate the report from your organization's management account in the `us-east-1` AWS Region only. The account generating the report must have access to an Amazon S3 bucket in the US East (N. Virginia) Region. The bucket must have an attached bucket policy as shown in Amazon S3 bucket policy for storing report.

To generate an organization-wide compliance report, you must have the following permissions:

- `organizations:DescribeEffectivePolicy`
- `tag:StartReportCreation`
- `tag:DescribeReportCreation`
- `tag:GetComplianceSummary`

**To generate an organization-wide compliance report (console)**

1. Open the Tag Policies console.
2. Choose the **This organization root** tab, and near the bottom of the page, choose **Generate report**.
3. On the **Generate report** screen, specify where to store the report.
4. Choose **Start exporting**.

When the report is complete, you can download it from the **Noncompliance report** section on the **Organization root** tab.

**Notes**

Organization-wide compliance is evaluated every 48 hours. This results in the following:

- It can take up to 48 hours for changes to a tag policy or resources to be shown in the organization-wide compliance report. For example, assume that you have a tag policy that defines a new standardized tag for a resource type. Resources of that type that don't have this tag can show as compliant in the report for up to 48 hours.
- Although you can generate the report at any time, report results aren't updated until the next evaluation is complete.
- The **NoncompliantKeys** column lists tag keys on the resource that are noncompliant with the effective tag policy.
- The **KeysWithNonCompliantValues** column lists keys defined in the effective policy that are on the resource with either incorrect case treatment or noncompliant values.
- If you close an AWS account that was a member of the organization, it can continue to appear in the tag compliance report for up to 90 days.

**To generate an organization-wide compliance report (AWS CLI, AWS API)**

Use the following commands and operations to generate an organization-wide compliance report, check on its status, and view the report:

- AWS Command Line Interface AWS CLI):
  - aws resourcegroupstaggingapi start-report-creation
  - aws resourcegroupstaggingapi describe-report-creation
  - aws resourcegroupstaggingapi get-compliance-summary

  For the complete procedure for using tag policies in the AWS CLI, see Using tag policies in the AWS CLI in the *AWS Organizations User Guide*.
- AWS API:
  - StartReportCreation
  - DescribeReportCreation

- GetComplianceSummary

# Monitor tag changes with serverless workflows and Amazon EventBridge

Amazon EventBridge supports tag changes on AWS resources. Using this EventBridge type, you can build EventBridge rules to match tag changes and route the events to one or more targets. For example, a target might be an AWS Lambda function to invoke automated workflows. This topic provides a tutorial for using Lambda to build a cost-effective serverless solution to securely process tag changes on your AWS resources.

## Tag changes generate EventBridge events

EventBridge delivers a near real-time stream of system events that describe changes in AWS resources. Many AWS resources support tags, which are custom, user-defined attributes to easily organize and categorize AWS resources. Common use cases for tags are cost allocation categorization, access-control security, and automation.

With EventBridge, you can monitor for changes to tags and track the tag state on AWS resources. Previously, to achieve similar functionality, you might have continuously polled APIs and orchestrated multiple calls. Now, any change to a tag including individual service APIs, Tag Editor, and the Tagging API will initiate the tag change on resource event. The following example shows a typical EventBridge event prompted by a tag change. It shows the new, updated, or deleted tag keys, and their associated values.

```
{
  "version": "0",
  "id": "bddcf1d6-0251-35a1-aab0-adc1fb47c11c",
  "detail-type": "Tag Change on Resource",
  "source": "aws.tag",
  "account": "123456789012",
  "time": "2018-09-18T20:41:38Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-0000000aaaaaaaaaa"
  ],
  "detail": {
    "changed-tag-keys": [
```

```
      "a-new-key",
      "an-updated-key",
      "a-deleted-key"
    ],
    "tags": {
      "a-new-key": "tag-value-on-new-key-just-added",
      "an-updated-key": "tag-value-was-just-changed",
      "an-unchanged-key": "tag-value-still-the-same"
    },
    "service": "ec2",
    "resource-type": "instance",
    "version": 3,
  }
}
```

All EventBridge events have the same top-level fields:

- **version** – By default, this value is set to `0` (zero) in all events.
- **id** – A unique value is generated for every event. This can be helpful in tracing events as they move through rules to targets and are processed.
- **detail-type** – Identifies, in combination with the `source` field, the fields and values that appear in the detail field.
- **source** – Identifies the service that was the source of the event. The source for tag changes is `aws.tag`.
- **time** – The timestamp of the event.
- **region** – Identifies the AWS Region where the event originated.
- **resources** – This JSON array contains Amazon Resource Names (ARNs) that identify resources that are involved in the event. This is the resource where tags have changed.
- **detail** – A JSON object, whose content is different depending on event type. For tag change on resource, the following detailed fields are included:
  - **changed-tag-keys** – The tag keys that changed by this event.
  - **service** – The service that the resource belongs to. In this example, the service is `ec2`, which is Amazon EC2.
  - **resource-type** – The type of resource of the service. In this example, it is an Amazon EC2 instance.
  - **version** – The version of the tag set. The version starts at 1 and increments when tags are changed. You can use the version to verify the order of tag change events.

- **tags** – The tags attached to the resource after the change.

For more information, see Amazon EventBridge event patterns in the *Amazon EventBridge User Guide*.

By using EventBridge, you can create rules that match specific event patterns based on the different fields. We demonstrate how to do this in the tutorial. Also, we show how an Amazon EC2 instance can be stopped automatically if a specified tag isn't attached to the instance. We use the EventBridge fields to create a pattern to match the tag events for the instance that launches a Lambda function.

## Lambda and serverless

AWS Lambda follows the serverless paradigm to run code in the cloud. You run code only when it's needed, without thinking about servers. You pay only for the exact compute time you use. Even though it's called *serverless*, it doesn't mean that there are no servers. Serverless in this context means that you don't have to provision, configure, or manage the servers that are used to run your code. AWS does all of that for you, so you can focus on your code. For more information about Lambda, see the AWS Lambda Product Overview.

## Tutorial: Automatically stopping Amazon EC2 instances that are missing required tags

**Topics**



As your pool of AWS resources and AWS accounts that you manage grows, you can use tags to make it easier to categorize your resources. Tags are commonly used for critical use cases such as cost allocation and security. To effectively manage AWS resources, your resources need to be consistently tagged. Often, when a resource is provisioned, it gets all the appropriate tags.

However, a later process can result in a tag change that results in drift from the corporate tag policy. By monitoring changes to your tags, you can spot tag drift and immediately respond. This gives you more confidence that the processes that depend on your resources being properly categorized will produce the desired results.

The following example demonstrates how to monitor for tag changes on Amazon EC2 instances to verify that a specified instance continues to have the required tags. If the instance's tags change and the instance no longer has the required tags, a Lambda function is invoked to shut down the instance automatically. Why would you want to do this? It ensures that all resources are tagged according to your corporate tag policy, for effective cost allocation, or to be able to trust security based on attribute-based access control (ABAC).

**Important**

We strongly recommend that you perform this tutorial in a non-production account where you can't inadvertently shut down important instances.
The example code in this tutorial intentionally limits the impact of this scenario to only the instances on a list of instance IDs. You must update the list with instance IDs that you are willing to shut down for the test. This helps ensure that you can't accidentally shut down every instance in a Region in your AWS account.
After testing, make sure that all of your instances are tagged according to your company's tagging strategy. Then, you can remove the code that limits the function to only the instance IDs on the list.

This example uses JavaScript and the 16.x version of Node.js. The example uses example AWS account ID 123456789012 and the AWS Region US East (N. Virginia) (`us-east-1`). Replace these with your own test account ID and Region.

**Note**

If your console uses a different Region for its default, ensure that you switch the Region you're using in this tutorial whenever you change consoles. A common cause of this tutorial failing is having the instance and function in two different Regions.

If you use a different Region than `us-east-1`, ensure that you change all references in the following code examples to your chosen Region.

## Step 1. Create the Lambda function

### To create the Lambda function

1. Open the AWS Lambda management console.
2. Choose **Create function** and then choose **Author from scratch**.
3. For **Function name**, type **`AutoEC2Termination`**.
4. For **Runtime**, choose **Node.js 16.x**.
5. Keep all other fields at their default values, and choose **Create function**.
6. On the **Code** tab of the `AutoEC2Termination` detail page, open the **index.js** file to view its code.

   - If a tab with **index.js** is open, you can choose the edit box in that tab to edit its code.
   - If a tab with **index.js** isn't open, right-click the **index.js** file under the **AutoEC2Terminator** folder in the navigation pane. Then choose **Open**.
7. In the **index.js** tab, paste the following code in the editor box, replacing anything that is already present.

   Replace the value `RegionToMonitor` with the Region that you want to run this function in.

```
// Set the following line to specify which Region's instances you want to monitor
// Only instances in this Region are succesfully stopped on a match

const RegionToMonitor = "us-east-1"

// Specify the instance ARNs to check.
// This limits the function for safety to avoid the tutorial shutting down all
 instances in account
// The first ARN is a "dummy" that matches the test event you create in Step 3.
// Replace the second ARN with one that matches a real instance that you want to
 monitor and that you can
// safely stop

const InstanceList = [
    "i-0000000aaaaaaaaaa",
    "i-05db4466d02744f07"
];

// The tag key name and value that marks a "valid" instance. Instances in the
 previous list that
```

```
// do NOT have the following tag key and value are stopped by this function

const ValidKeyName = "valid-key";
const ValidKeyValue = "valid-value";

// Load and configure the AWS SDK
const AWS = require('aws-sdk');
// Set the AWS Region
AWS.config.update({region: RegionToMonitor});
// Create EC2 service object.
const ec2 = new AWS.EC2({apiVersion: '2016-11-15'});

exports.handler = (event, context, callback) => {

  // Retrieve the details of the reported event.
  var detail = event.detail;
  var tags = detail["tags"];
  var service = detail["service"];
  var resourceType = detail["resource-type"];
  var resource = event.resources[0];
  var resourceSplit = resource.split("/");
  var instanceId = resourceSplit[resourceSplit.length - 1];

  // If this event is not for an EC2 resource, then do nothing.
  if (!(service === "ec2")) {
    console.log("Event not for correct service -- no action (", service, ")" );
    return;
  }

  // If this event is not about an instance, then do nothing.
  if (!(resourceType === "instance")) {
    console.log("Event not for correct resource type -- no action (", resourceType,
")" );
    return;
  }

  // CAUTION - Removing the following 'if' statement causes the function to run
 against
  //           every EC2 instance in the specified Region in the calling AWS
 account.
  //           If you do this and an instance is not tagged with the approved tag
 key
  //           and value, this function stops that instance.
```

```
 // If this event is not for the ARN of an instance in our include list, then do
nothing.
 if (InstanceList.indexOf(instanceId)<0) {
   console.log("Event not for one of the monitored instances -- no action (",
resource, ")");
   return;
 }

 console.log("Tags changed on monitored EC2 instance (",instanceId,")");

 // Check attached tags for expected tag key and value pair
 if ( tags.hasOwnProperty(ValidKeyName) && tags[ValidKeyName] == "valid-value"){
   // Required tags ARE present
   console.log("The instance has the required tag key and value -- no action");
   callback(null, "no action");
   return;
 }

 // Required tags NOT present
 console.log("This instance is missing the required tag key or value -- attempting
to stop the instance");

 var params = {
   InstanceIds: [instanceId],
   DryRun: true
 };

 // call EC2 to stop the selected instances
 ec2.stopInstances(params, function(err, data) {
   if (err && err.code === 'DryRunOperation') {
     // dryrun succeeded, so proceed with "real" stop operation
     params.DryRun = false;
     ec2.stopInstances(params, function(err, data) {
       if (err) {
         console.log("Failed to stop instance");
         callback(err, "fail");
       } else if (data) {
         console.log("Successfully stopped instance", data.StoppingInstances);
         callback(null, "Success");
       }
     });
   } else {
     console.log("Dryrun attempt failed");
     callback(err);
```

```
    }
  });
};
```

8. Choose **Deploy** to save your changes and make the new version of the function active.

This Lambda function checks the tags of an Amazon EC2 instance, as reported by the tag change event in EventBridge. In this example, if the instance in the event is missing the required tag key `valid-key` or if that tag doesn't have the value `valid-value`, then the function tries to stop the instance. You can change this logical check or the tag requirements for your own specific use cases.

Keep the Lambda console window open in your browser.

## Step 2. Set up the required IAM permissions

Before the function can successfully run, you must grant the function the permission to stop an EC2 instance. The AWS provided role lambda_basic_execution doesn't have that permission. In this tutorial, you modify the default IAM permission policy that is attached to the function's execution role named `AutoEC2Termination-role-`*`uniqueid`*. The minimum additional permission required for this tutorial is `ec2:StopInstances`.

For more information about creating Amazon EC2 specific IAM policies, see Amazon EC2: Allows starting or stopping an EC2 Instance and modifying a security group, programmatically and in the console in the *IAM User Guide*.

**To create an IAM permission policy and attach it to the Lambda function's execution role**

1. In a different browser tab or window, open the Roles page of the IAM console.
2. Start typing the role name **`AutoEC2Termination`**, and when it appears in the list, choose the role name.
3. On the role's **Summary** page, choose the **Permissions** tab and choose the name of the one policy that is already attached.
4. On the policy's **Summary** page, choose **Edit policy**.
5. On the **Visual Editor** tab, choose **Add additional permissions**.
6. For **Service**, choose **EC2**.
7. For **Actions**, choose **StopInstances**. You can type **`Stop`** in the search bar, and then choose `StopInstances` when it appears.
8. For **Resources**, choose **All resources**, choose **Review policy**, and then choose **Save changes**.

This automatically creates a new version of the policy and sets that version as the default.

Your final policy should look similar to the following example.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "ec2:StopInstances",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:us-east-1:123456789012:*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:us-east-1:123456789012:log-group:/aws/lambda/
AutoEC2Termination:*"
        }
    ]
}
```

## Step 3. Do a preliminary test of your Lambda function

In this step, you submit a test event to your function. The Lambda test functionality works by submitting a manually provided test event. The function processes the test event just as if the event had come from EventBridge. You can define multiple test events with different values to exercise all of the different parts of your code. In this step, you submit a test event that indicates that an Amazon EC2 instance's tags changed, and the new tags don't include the required tag key and value.

**To test your Lambda function**

1. Return to the window or tab with the Lambda console and open the **Test** tab for your **AutoEC2Termination** function.
2. Choose **Create new event**.
3. For **Event name**, enter **`SampleBadTagChangeEvent`**.
4. In the **Event JSON**, replace the text with the sample event shown in the following example text. You don't need to modify the accounts, Region, or instance ID for this test event to work correctly.

```
{
  "version": "0",
  "id": "bddcf1d6-0251-35a1-aab0-adc1fb47c11c",
  "detail-type": "Tag Change on Resource",
  "source": "aws.tag",
  "account": "123456789012",
  "time": "2018-09-18T20:41:38Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-0000000aaaaaaaaaa"
  ],
  "detail": {
    "changed-tag-keys": [
      "valid-key"
    ],
    "tags": {
      "valid-key": "NOT-valid-value"
    },
    "service": "ec2",
    "resource-type": "instance",
    "version": 3
  }
}
```

5. Choose **Save**, and then choose **Test**.

   The test appears to fail, but that's OK.

   You should see the following error in the **Execution results** tab under **Response**.

```
{
```

```
  "errorType": "InvalidInstanceID.NotFound",
  "errorMessage": "The instance ID 'i-0000000aaaaaaaaaa' does not exist",
  ...
}
```

The error occurs because the instance specified in the test event doesn't exist.

The information on the **Execution results** tab, in the **Function Logs** section , demonstrates that your Lambda function successfully attempted to stop an EC2 instance. However, it failed because the code initially attempts a DryRun operation to stop the instance, which indicated that the instance ID was not valid.

```
START RequestId: 390c1f8d-0d9b-4b44-b087-8de64479ab44 Version: $LATEST
2022-11-30T20:17:30.427Z    390c1f8d-0d9b-4b44-b087-8de64479ab44    INFO    Tags
 changed on monitored EC2 instance ( i-0000000aaaaaaaaaa )
2022-11-30T20:17:30.427Z    390c1f8d-0d9b-4b44-b087-8de64479ab44    INFO    This
 instance is missing the required tag key or value -- attempting to stop the
 instance
2022-11-30T20:17:31.206Z    390c1f8d-0d9b-4b44-b087-8de64479ab44    INFO    Dryrun
 attempt failed
2022-11-30T20:17:31.207Z    390c1f8d-0d9b-4b44-b087-8de64479ab44    ERROR   Invoke
 Error     {"errorType":"InvalidInstanceID.NotFound","errorMessage":"The instance
 ID 'i-0000000aaaaaaaaaa' does not
 exist","code":"InvalidInstanceID.NotFound","message":"The instance ID
 'i-0000000aaaaaaaaaa' does not
 exist","time":"2022-11-30T20:17:31.205Z","requestId":"a5192c3b-142d-4cec-
bdbc-685a9b7c7abf","statusCode":400,"retryable":false,"retryDelay":36.87870631147607,"stack
["InvalidInstanceID.NotFound: The instance ID 'i-0000000aaaaaaaaaa' does
 not exist","    at Request.extractError (/var/runtime/node_modules/aws-sdk/
lib/services/ec2.js:50:35)","    at Request.callListeners (/var/runtime/
node_modules/aws-sdk/lib/sequential_executor.js:106:20)","    at Request.emit
 (/var/runtime/node_modules/aws-sdk/lib/sequential_executor.js:78:10)","    at
 Request.emit (/var/runtime/node_modules/aws-sdk/lib/request.js:686:14)","    at
 Request.transition (/var/runtime/node_modules/aws-sdk/lib/request.js:22:10)","
    at AcceptorStateMachine.runTo (/var/runtime/node_modules/aws-sdk/lib/
state_machine.js:14:12)","    at /var/runtime/node_modules/aws-sdk/lib/
state_machine.js:26:10","    at Request.<anonymous> (/var/runtime/node_modules/aws-
sdk/lib/request.js:38:9)","    at Request.<anonymous> (/var/runtime/node_modules/
aws-sdk/lib/request.js:688:12)","    at Request.callListeners (/var/runtime/
node_modules/aws-sdk/lib/sequential_executor.js:116:18)"]}
END RequestId: 390c1f8d-0d9b-4b44-b087-8de64479ab44
```

6. To prove that the code doesn't try to stop the instance when the correct tag is used, you can create and submit another test event.

   Choose the **Test** tab above **Code source**. The console displays your existing **SampleBadTagChangeEvent** test event.

7. Choose **Create new event**.
8. For **Event name**, type **`SampleGoodTagChangeEvent`**.
9. In line 17, delete **`NOT-`** to change the value to **`valid-value`**.
10. At the top of the **Test event** window, choose **Save**, and then choose **Test**.

    The output displays the following, which demonstrates that the function recognizes the valid tag and doesn't attempt to shut down the instance.

```
START RequestId: 53631a49-2b54-42fe-bf61-85b9e91e86c4 Version: $LATEST
2022-12-01T23:24:12.244Z    53631a49-2b54-42fe-bf61-85b9e91e86c4    INFO    Tags
 changed on monitored EC2 instance ( i-0000000aaaaaaaaaa )
2022-12-01T23:24:12.244Z    53631a49-2b54-42fe-bf61-85b9e91e86c4    INFO    The
 instance has the required tag key and value -- no action
END RequestId: 53631a49-2b54-42fe-bf61-85b9e91e86c4
```

    Keep the Lambda console open in your browser.

## Step 4. Create the EventBridge rule that launches the function

Now you can create an EventBridge rule that matches the event and points to your Lambda function.

### To create the EventBridge rule

1. In a different browser tab or window, open the EventBridge console to the **Create Rule** page.
2. For **Name**, enter **`ec2-instance-rule`**, and then choose **Next**.
3. Scroll down to **Creation method** and choose **Custom pattern (JSON editor)**.
4. In the editing box, paste the following pattern text, and then choose **Next**.

```
{
  "source": [
    "aws.tag"
```

```
  ],
  "detail-type": [
    "Tag Change on Resource"
  ],
  "detail": {
    "service": [
      "ec2"
    ],
    "resource-type": [
      "instance"
    ]
  }
}
```

This rule matches `Tag Change on Resource` events for Amazon EC2 instances and invokes whatever you specify as the **Target** in the next step.

5. Next, add your Lambda function as the target. In the **Target 1** box, under **Select a target**, choose **Lambda function**.
6. Under **Function**, choose the **AutoEC2Termination** function that you created previously, and then choose **Next**.
7. On the **Configure tags** page, choose **Next**. Then on the **Review and create** page, choose **Create rule**. This also automatically grants permission for EventBridge to invoke the specified Lambda function.

## Step 5. Test the complete solution

You can test your final result by creating an EC2 instance and watching what happens when you change its tags.

**To test the monitoring solution with a real instance**

1. Open the Amazon EC2 console to the **Instances** page.
2. Create an Amazon EC2 instance. Before you launch it, attach a tag with the key `valid-key` and the value `valid-value`. For information about how to create and launch an instance,see Step 1: Launch an instance in the *Amazon EC2 User Guide*. In the procedure *To launch an instance*, in step 3, where you enter the **Name** tag, also choose **Add additional tags**, choose **Add tag**, and then enter the **Key** of **`valid-key`** and **Value** of **`valid-value`**. You can **Proceed without a key pair** if this instance is solely for the purposes of this tutorial and you plan on

deleting this instance after you complete it. Return to this tutorial when you reach the end of **Step 1**; you don't need to do **Step 2: Connect to your instance**.

3. Copy the **InstanceId** from the console.
4. Switch from the Amazon EC2 console to the Lambda console. Choose your **AutoEC2Termination** function, choose the **Code** tab, and then choose the **index.js** tab to edit your code.
5. Change the second entry in the `InstanceList` by pasting the value you copied from the Amazon EC2 console. Ensure that the `RegionToMonitor` value matches the Region that contains the instance you pasted.
6. Choose **Deploy** to make your changes active. The function is now ready to be activated by tag changes to that instance in the specified Region.
7. Switch from the Lambda console to the Amazon EC2 console.
8. Change the **Tags** attached to the instance by either deleting the **valid-key** tag or by changing that key's value.

> **Note**
>
> For information about how to change the tags on a running Amazon EC2 instance, see Add and delete tags on an individual resource in the *Amazon EC2 User Guide*.

9. Wait a few seconds, and then refresh the console. The instance should change its **Instance state** to **Stopping** and then to **Stopped**.
10. Switch from the Amazon EC2 console to the Lambda console with your function, and choose the **Monitor** tab.
11. Choose the **Logs** tab, and in the **Recent invocations** table, choose the most recent entry in the **LogStream** column.

    The Amazon CloudWatch console opens to the **Log events** page for the last invocation of your Lambda function. The last entry should look similar to the following example.

```
2022-11-30T12:03:57.544-08:00    START RequestId: b5befd18-2c41-43c8-
a320-3a4b2317cdac Version: $LATEST
2022-11-30T12:03:57.548-08:00    2022-11-30T20:03:57.548Z b5befd18-2c41-43c8-
a320-3a4b2317cdac INFO Tags changed on monitored EC2 instance ( arn:aws:ec2:us-
west-2:123456789012:instance/i-1234567890abcdef0 )
```

```
2022-11-30T12:03:57.548-08:00    2022-11-30T20:03:57.548Z b5befd18-2c41-43c8-
a320-3a4b2317cdac INFO This instance is missing the required tag key or value --
 attempting to stop the instance
2022-11-30T12:03:58.488-08:00    2022-11-30T20:03:58.488Z b5befd18-2c41-43c8-
a320-3a4b2317cdac INFO Successfully stopped instance [ { CurrentState: { Code: 64,
 Name: 'stopping' }, InstanceId: 'i-1234567890abcdef0', PreviousState: { Code: 16,
 Name: 'running' } } ]
2022-11-30T12:03:58.546-08:00    END RequestId: b5befd18-2c41-43c8-
a320-3a4b2317cdac
```

### Summary

This tutorial demonstrated how to create an EventBridge rule to match against a tag change on a resource event for Amazon EC2 instances. The rule pointed to a Lambda function that automatically shuts down the instance if it doesn't have the required tag.

The Amazon EventBridge support for tag changes on AWS resources opens possibilities to build event-driven automation across many AWS services. Combining this capability with AWS Lambda provides you with tools to build serverless solutions that access AWS resources securely, scale on demand, and are cost effective.

Other possible use cases for the tag-change-on-resource EventBridge event include:

- **Launch a warning if someone accesses your resource from an unusual IP address** – Use a tag to store the source IP address of each visitor that accesses your resource. Changes to the tag generates a CloudWatch event. You can use that event to compare the source IP address to a list of valid IP addresses and activate a warning email if the source IP address isn't valid.
- **Monitor if there are changes to your tag-based access control for a resource** – If you have set up access to a resource using attribute (tag) based access control (ABAC), you can use EventBridge events generated by any changes to the tag to prompt an audit by your security team.

## Troubleshooting tag changes

The following checklist might be helpful if errors occur when you try to apply or change tags on selected resources in **Find resources to tag** query results.

- The resource might already have the maximum number of tags. Generally, resources can have a maximum of 50 user-defined tags. AWS generated tags don't count toward the 50-tag

maximum. Other users might also be adding tags to the same resource at the same time, which could raise the resource's tags to the maximum.

- Some services allow a different character set (or restrict the character set that is allowed) for creating tags. If you added or changed tags using special characters, review the tag requirements in the resource's service documentation to verify that those characters are allowed by the service.
- You might not have permissions to modify the tags for the resource. If you don't have permissions to view existing tags on a resource, you can't make changes to the resource's tags.
- You might not have permissions to change the resource. Changes to the resource's metadata might be restricted by another administrator.
- The resource might have been edited or deleted by another user or process. For example, assume that a resource was launched as part of the creation of an AWS CloudFormation stack. If the stack was deleted or is no longer in an active state, the resource might no longer be available.
- Tag changes might not be possible if a resource is offline or terminated, or if other updates (such as software upgrades) to the resource are in progress.
- Tag changes can fail if you close the browser tab or change the page before the tag changes complete. Let tag changes finish, and wait for the success or failure banner to appear on the page, before you leave the page.
- While there's a rate limit for the AWS Resource Groups Tagging API, the service you're tagging might impose a separate limit which you might hit before the Resource Groups Tagging API limit.

## Related information

- Using cost allocation tags in the *AWS Billing User Guide*

# Security in Tag Editor

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. For more information about the compliance programs that apply to Tag Editor, 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 Tag Editor. The following topics show you how to configure Tag Editor to meet your security and compliance objectives.

**Topics**



# Data protection in Tag Editor

The AWS shared responsibility model applies to data protection in Tag Editor. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud.

You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

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

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 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 text fields such as a **Name** field. This includes when you work with Tag Editor or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

## Data encryption

Tagging information is not encrypted. Although not encrypted, tags can contain information used as part of your security strategy, so it's important to control who can access tags on resources. It's especially critical that you control who can modify tags because such access could be used to elevate one's permissions.

### Encryption at rest

There are no additional ways of isolating service or network traffic that are specific to Tag Editor. If applicable, use AWS specific isolation. You can use the Tag Editor API and console in a virtual private cloud (VPC) to help maximize privacy and infrastructure security.

### Encryption in transit

Tag Editor data is encrypted in transit to the service's internal database for backup. This is not user-configurable.

### Key management

Tag Editor is not currently integrated with AWS Key Management Service and does not support AWS KMS keys.

## Internetwork traffic privacy

Tag Editor uses HTTPS for all transmissions between Tag Editor users and AWS. Tag Editor uses transport layer security (TLS) 1.3, but also supports TLS 1.2.

# Identity and access management for Tag Editor

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 Tag Editor 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 Tag Editor.

**Service user** – If you use the Tag Editor service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Tag Editor 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 Tag Editor, see Troubleshooting Tag Editor identity and access.

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

**IAM administrator** – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Tag Editor. To view example Tag Editor identity-based policies that you can use in IAM, see Tag Editor identity-based policy examples.

## Authenticating with identities

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

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

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

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If

you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

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

### AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

### Users and Groups

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

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

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

## 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:

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

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

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see 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 AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

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

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

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A

user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

### Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see 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 an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see 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 Tag Editor works with IAM

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

#### Topics

- Tag Editor identity-based policies
- Resource-based policies
- Authorization based on tags
- Tag Editor IAM roles

## Tag Editor identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources in addition to the conditions under which actions are allowed or denied. Tag Editor supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

### Actions

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.

Policy actions in Tag Editor use the following prefix before the action: `tag:`. Tag Editor actions are performed entirely in the console, but have the prefix `tag` in log entries.

For example, to grant someone permission to tag a resource with the `tag:TagResources` API operation, you include the `tag:TagResources` action in their policy. Policy statements must include either an `Action` or `NotAction` element. Tag Editor defines its own set of actions that describe tasks that you can perform with this service.

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

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

```
      "tag:action3"
```

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

```
"Action": "tag:Get*"
```

To see a list of Tag Editor actions, see Actions, resources, and condition keys for Tag Editor in the *Service Authorization Reference*.

**Resources**

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": "*"
```

Tag Editor does not have any resources of its own. Instead, it manipulates the metadata (tags) that are attached to resources created by other AWS services.

**Condition keys**

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*.

Tag Editor does not define any service-specific condition keys.

**Examples**

To view examples of Tag Editor identity-based policies, see Tag Editor identity-based policy examples.

## Resource-based policies

Tag Editor does not support resource-based policies because it doesn't define any of its own resources.

## Authorization based on tags

Authorization based on tags is part of the security strategy called attribute-based access control (ABAC).

To control access to a resource based on its 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. You can apply tags to a resource when you are creating or updating the resource.

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see Viewing groups based on tags. For more information about attribute-based access control (ABAC), see What is ABAC for AWS? in the *IAM User Guide*.

## Tag Editor IAM roles

An IAM role is an entity within your AWS account that has specific permissions. Tag Editor does not have or use service roles.

### Using temporary credentials with Tag Editor

In Tag Editor, you can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

### Service-linked roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf.

Tag Editor does not have or use service-linked roles.

### Service roles

This feature allows a service to assume a service role on your behalf.

Tag Editor does not have or use service roles.

## Tag Editor identity-based policy examples

By default, IAM principals, such as roles and users, don't have permission to create or modify tags. They also can't perform tasks using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS APIs. An IAM administrator must create IAM policies that grant principals permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the principals that require those permissions.

For instructions on creating an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

### Topics

- Policy best practices
- Using the Tag Editor console and Resource Groups Tagging API
- Allow users to view their own permissions
- Viewing groups based on tags

## Policy best practices

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

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

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

## Using the Tag Editor console and Resource Groups Tagging API

To access the Tag Editor console and the Resource Groups Tagging API, you must have a minimum set of permissions. These permissions must allow you to list and view details about the tags attached to resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console and API commands won't function as intended for the IAM principals with that policy.

To ensure that those principals can still use Tag Editor, attach the following policy (or a policy that contains the permissions listed in the following policy) to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "tag:GetResources",
        "tag:TagResources",
        "tag:UntagResources",
        "tag:getTagKeys",
        "tag:getTagValues",
        "resource-explorer:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

For more information about granting access to Tag Editor and Resource Groups Tagging API, see Granting permissions for using Tag Editor.

## 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": "*"
        }
    ]
}
```

## Viewing groups based on tags

You can use conditions in your identity-based policy to control access to Tag Editor resources based on tags. This example shows how you might create a policy that allows viewing a resource, in this example, a resource group. However, permission is granted only if the group tag `project` has the same value as the `project` tag attached to the calling principal.

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

```
            "Effect": "Allow",
            "Action": "resource-groups:ListGroups",
            "Resource": "arn:aws:resource-groups::region:account_ID:group/group_name"
        },
        {
            "Effect": "Allow",
            "Action": "resource-groups:ListGroups",
            "Resource": "arn:aws:resource-groups::region:account_ID:group/group_name",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/project": "${aws:PrincipalTag/
project}"}
            }
        }
    ]
}
```

You can attach this policy to the users in your account. If a user with the tag key `project` and tag value `alpha` attempts to view a resource group, the group must also be tagged `project=alpha`. Otherwise the user is denied access. The condition tag key `project` matches both `Project` and `project` because condition key names are not case-sensitive. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

## Troubleshooting Tag Editor identity and access

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

**Topics**

- I am not authorized to perform an action in Tag Editor
- I am not authorized to perform iam:PassRole

### I am not authorized to perform an action in Tag Editor

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your sign-in credentials.

The following example error occurs when the user `mateojackson` tries to use the console to view tags on a resource but does not have `tag:GetTagKeys` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 tag:GetTagKeys on resource: arn:aws:resource-groups::us-west-2:123456789012:resource-
type/my-test-resource
```

In this case, Mateo asks his administrator to update his policies to allow him to access the `my-test-resource` resource using the `tag:GetTagKeys` action.

### I am not authorized to perform iam:PassRole

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

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 Tag Editor. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

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

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

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

# Logging and monitoring in Tag Editor

All Tag Editor actions are logged in AWS CloudTrail.

## Logging Tag Editor API calls with CloudTrail

Tag Editor is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Tag Editor. CloudTrail captures all API calls for Tag Editor as events, including calls from the Tag Editor console and from code calls to the Resource Groups Tagging API. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Tag Editor. 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 Tag Editor, the IP address from which the request was made, who made the request, when it was made, and additional details.

For more information about CloudTrail, see the AWS CloudTrail User Guide.

## Tag Editor information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Tag Editor, or in the Tag Editor console, 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 Tag Editor, 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 resources:

- Creating a trail for your AWS account
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple Regions and Receiving CloudTrail log files from multiple accounts

All Tag Editor actions are logged by CloudTrail and are documented in the Tag Editor API Reference. Tag Editor actions in the console are logged by CloudTrail, and are shown as events with `tagging.amazonaws.com` as the `eventSource`.

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

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

For more information, see the CloudTrail `userIdentity` element.

## Understanding Tag Editor 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 are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the action `TagResources`.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAEXAMPLEEXAMPLE:botocore-session-1661372702",
        "arn": "arn:aws:sts::123456789012:assumed-role/cli-role/botocore-
session-1661372702",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAEXAMPLEEXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/cli-role",
                "accountId": "123456789012",
                "userName": "cli-role"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-08-24T20:25:03Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-08-24T20:27:14Z",
    "eventSource": "tagging.amazonaws.com",
    "eventName": "TagResources",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.65",
    "userAgent": "aws-cli/2.7.14 Python/3.9.11 Windows/10 exe/AMD64 prompt/off command/
resourcegroupstaggingapi.tag-resources",
    "requestParameters": {
        "resourceARNList": [
```

```
            "arn:aws:events:us-east-1:123456789012:rule/SecretsManagerMonitorRule"
        ],
        "tags": {
            "owner": "alice"
        }
    },
    "responseElements": {
        "failedResourcesMap": {}
    },
    "requestID": "8f9ea891-4125-460c-802f-26c11EXAMPLE",
    "eventID": "b2c9322a-aad7-424b-8f0b-423daEXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "tagging.us-east-1.amazonaws.com"
    }
}
```


# Compliance validation for Tag Editor

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

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

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

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance on Amazon Web Services – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

> **Note**
>
> Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.

- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

## Resilience in Tag Editor

Tag Editor performs automated backups to internal service resources. These backups are not user-configurable. Backups are encrypted, both at rest and in transit. Tag Editor stores customer data in Amazon DynamoDB.

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can

design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

If you delete tags accidentally, contact AWS Support Center.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

## Infrastructure security in Tag Editor

Tag Editor doesn't provide additional ways of isolating service or network traffic. If applicable, use AWS specific isolation. You can use the Tag Editor API and console in a virtual private cloud (VPC) to help maximize privacy and infrastructure security.

You use AWS published API calls to access Tag Editor through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TSL 1.2 and recommend TSL 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an AWS Identity and Access Management (IAM) principal. Or, you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Tag Editor does not support resource-based policies.

You can call Tag Editor API operations from any network location, but Tag Editor does support resource-based access policies, which can include restrictions based on the source IP address. You can also use Tag Editor policies to control access from specific Amazon Virtual Private Cloud (Amazon VPC) endpoints or specific VPCs. Effectively, this approach isolates network access to a given resource from only the specific VPC within the AWS network.

# Tag Editor reference information

Tag Editor reference information includes the applicable service quotas.

## Service quotas for Tag Editor

The following table provides information about the service quotas for Tag Editor.

These quotas are currently not adjustable using the Service Quotas console. Contact AWS Support.

| Name | Default |
| --- | --- |
| Tags attached per resource | 50 user-defined tags (AWS generated tags don't count against this limit.) |
| Tag key name | Minimum of 1, maximum 128 Unicode characters in UTF-8.<br><br>Allowed characters include letters, numbers, spaces, and the following characters:<br><br>`_ . : / = + - @`<br><br>Key names can't begin with `aws:` because that prefix is reserved for AWS use.<br><br>**Note**<br>Some AWS services have some additiona l character or length restrictions. For details, see the |

| Name | Default |
| --- | --- |
| | documentation for the specific service. |
| Tag values | Minimum of 0, maximum of 256 Unicode characters in UTF-8.<br><br>Allowed characters include letters, numbers, spaces, and the following characters:<br><br>`_ . : / = + - @`<br><br>**Note**<br>Some AWS services have some additional character or length restrictions. For details, see the documentation for the specific service. |
| Rate of calling the GetResources API operation | Maximum of 15 calls per second |
| Rate of calling the following API operations:<br><br>• TagResources<br>• UntagResources<br>• GetTagKeys<br>• GetTagValues | Maximum of 5 calls per second |

# Tag Editor document history

| Change | Description | Date |
| --- | --- | --- |
| Tagging content from AWS General Reference moved to this guide | The topics about tagging your AWS resources was moved from the AWS General Reference to this guide. | March 24, 2023 |
| IAM best practices update | Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM. | January 3, 2023 |
| Moving Tag Editor documentation to its own guide | Tag Editor documentation is now provided in its own user guide instead of being part of the AWS Resource Groups User Guide. | December 13, 2022 |
| Check for compliance with tag policies | After you create and attach tag policies to accounts using AWS Organizations, you can find noncompliant tags on resources in your organization's accounts. | November 26, 2019 |
| Tag Editor now supports finding untagged resources | You can now search for resources in Tag Editor that do not have tag values applied for a specific tag key. | June 18, 2019 |
| Tag Editor console moves out of AWS Systems Manager console | The Tag Editor console is now independent from the Systems Manager console. Although you can still find pointers to the Tag Editor | June 5, 2019 |

| | | |
|---|---|---|
| | console in the Systems Manager left navigation bar, you can open the Tag Editor console directly from the drop-down menu at the upper left of the AWS Management Console. | |
| Older, legacy Tag Editor tools are no longer available | Mentions of older, classic, or legacy Tag Editor have been removed; these tools are no longer available in AWS. Use Tag Editor instead. | May 14, 2019 |
| Tag Editor now supports tagging resources across multiple regions | Tag Editor now lets you search for and manage tags of resources across multiple regions, with your current region added to resource queries by default. | May 2, 2019 |
| Tag Editor now supports exporting query results to a CSV | You can export the results of a query on the **Find Resources to tag** page to a CSV-formatted file. A new Region column is shown in Tag Editor query results. Tag Editor now lets you search for resources that have empty values for a specific tag key. Tag key values auto-complete as you type a unique value among existing keys. | April 2, 2019 |

| | | |
|---|---|---|
| Tag Editor now supports adding all resource types to a query | You can apply tags to up to 20 individual resource types in a single operation, or you can choose **All resource types** to query all resource types in a region. Autocompletion has been added to the **Tag key** field of a query to help enable consistent tag keys among resources. If tag changes fail on some resources, you can retry tag changes on just resources for which tag changes failed. | March 19, 2019 |
| Tag Editor now supports multiple resource types in a search | You can apply tags to up to 20 resource types in a single operation. You can also choose the columns that are shown to you in search results, including columns for each unique tag key found in your search results or selected resources from results. | February 26, 2019 |

# AWS Glossary

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