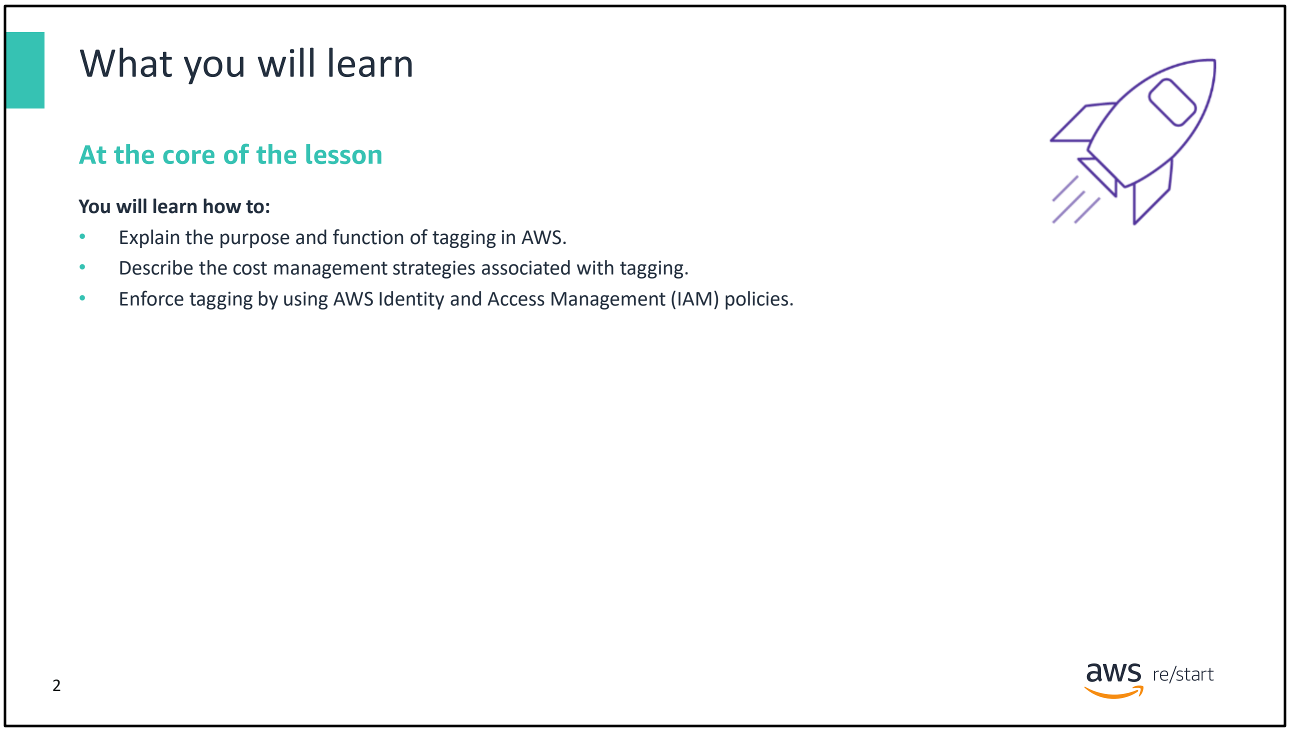
re/start

Tagging

Welcome to Tagging.



What you will learn

At the core of the lesson

You will learn how to:

- Explain the purpose and function of tagging in AWS.
- Describe the cost management strategies associated with tagging.
- Enforce tagging by using AWS Identity and Access Management (IAM) policies.

In this lesson, you will learn how to manage resource consumption in an AWS account by using tags. You will also review examples of common tagging use cases that use AWS Config and AWS Identity and Access Management (IAM).

Specifically, you will learn how to:

- Explain the purpose and function of tagging in AWS.
- Describe the cost management strategies associated with tagging.
- Enforce tagging by using AWS Identity and Access Management (IAM) policies.

Video

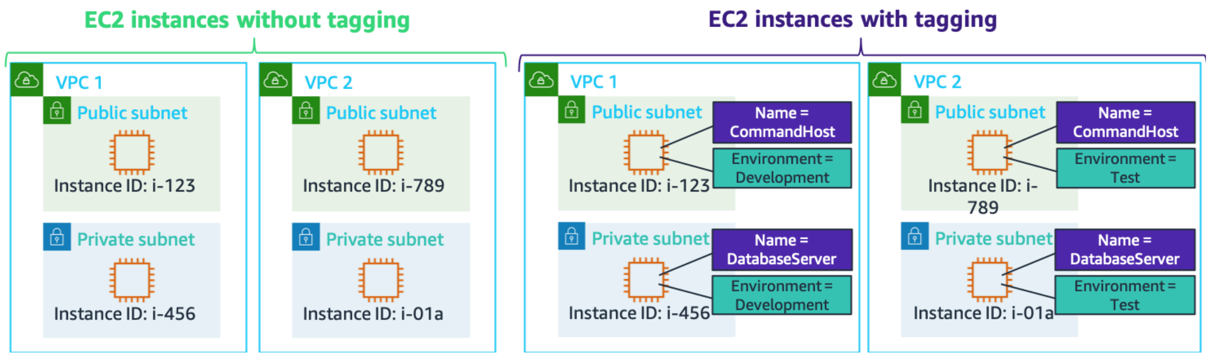
AWS Q&A with an SA:
What are tags and what can I do with them?

This video introduces what a tag is and what you can do with them. It can be viewed on the Amazon Web Services YouTube channel at [AWS Q&A with an SA: What are tags and what can I do with them?]

What is a tag?

A tag:

- Is a key-value pair that can be attached to an AWS resource.
- Enables you to identify and categorize resources.

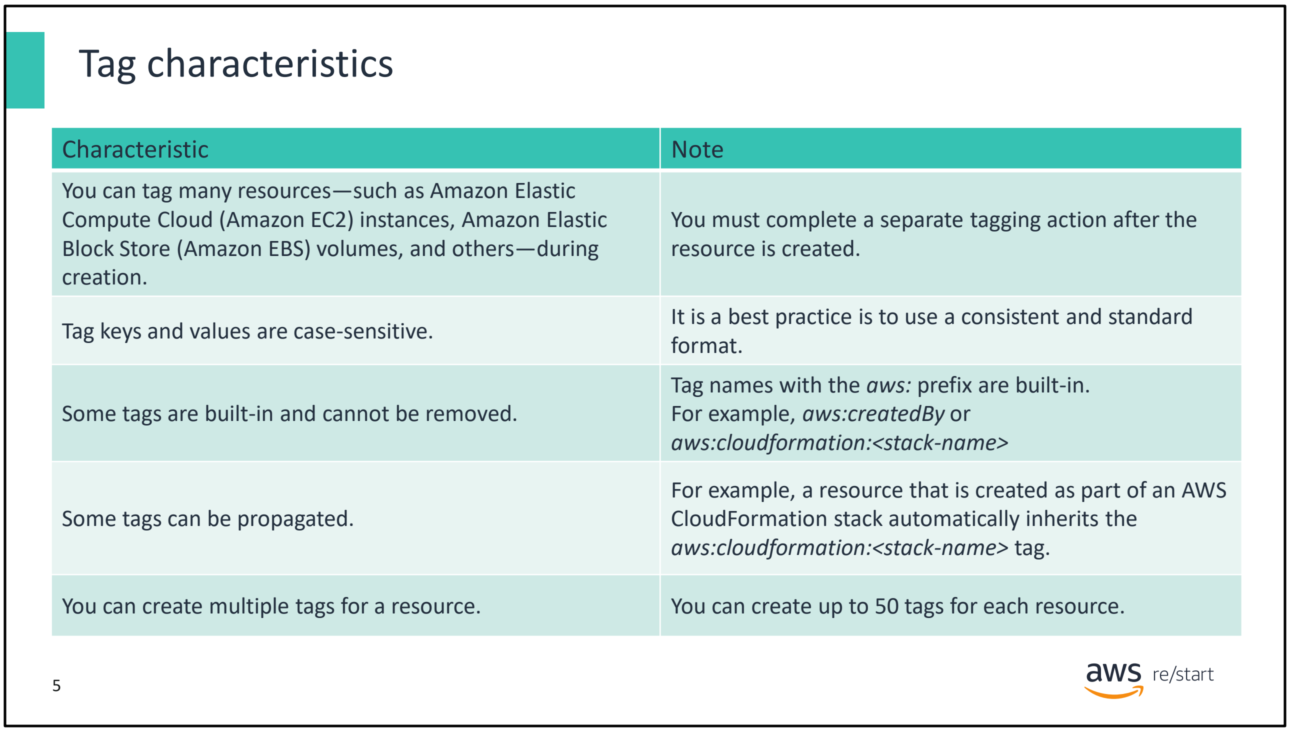


aws re/start

A tag is a label that you assign to an AWS resource. It enables you to identify it or categorize it in a meaningful way. A tag consists of a *key* and a *value*, both of which you define.

For example, if you have two Amazon Elastic Compute Cloud (Amazon EC2) instances in a development environment, you might assign a tag to both instances with a key of *Name* (essentially, a name tag). You could then assign a value to each key, such as *CommandHost* for the first instance and *DatabaseServer* for the second instance. With these tags, you could quickly identify each instance and their purpose, which would be more difficult if you used only their instance IDs. In addition, if you created similar instances in another environment—such as *Test*—you could assign another tag to each instance with a key of *Environment*. You could then assign a value of *Development* or *Test* to each instance so that you could distinguish between them and categorize them by environment.

This situation is illustrated in the diagram. In the two virtual private clouds (VPCs) on the left, the EC2 instances do not have tags. It would not be easy to identify their purpose or which environment—*Development* or *Test*—they are running in. In contrast, the instances in the two VPCs on the right have both a *Name* tag and an *Environment* tag. You can clearly determine their purpose and running environment.

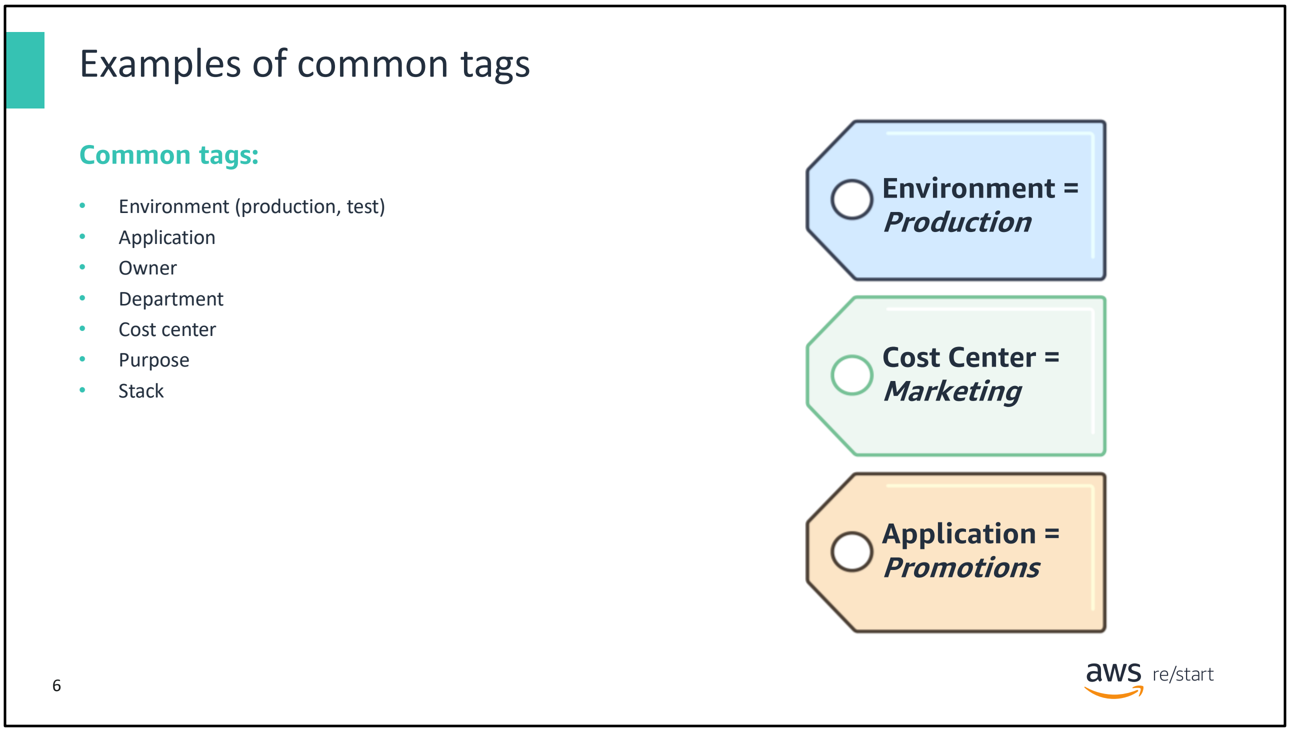


Tag characteristics

Characteristic	Note
You can tag many resources—such as Amazon Elastic Compute Cloud (Amazon EC2) instances, Amazon Elastic Block Store (Amazon EBS) volumes, and others—during creation.	You must complete a separate tagging action after the resource is created.
Tag keys and values are case-sensitive.	It is a best practice is to use a consistent and standard format.
Some tags are built-in and cannot be removed.	Tag names with the *aws:* prefix are built-in. For example, *aws:createdBy* or *aws:cloudformation:<stack-name>*
Some tags can be propagated.	For example, a resource that is created as part of an AWS CloudFormation stack automatically inherits the *aws:cloudformation:<stack-name>* tag.
You can create multiple tags for a resource.	You can create up to 50 tags for each resource.

Tags have a number of important characteristics, which include:

- You can only assign a tag to a resource after it has been created.
- Tag keys and values are case-sensitive. A best practice is to use a consistent and standardized format.
- You cannot edit or delete tag keys or values with the *aws:* prefix. These tags are assigned by AWS and are reserved for AWS use. For example, the `aws:createdBy` tag is automatically generated by AWS for cost allocation purposes. It is assigned to a resource to identify its creator. Another example of a generated tag is the `aws:cloudformation:<stack-name>` tag, which AWS CloudFormation automatically assigns to every stack to identify its name.
- Tags are sometimes inherited or propagated. Some services, such as AWS CloudFormation, and AWS Elastic Beanstalk can create other resources, such as Amazon Relational Database (Amazon RDS) instances or EC2 instances. Generally, when one of these services creates a resource, it will tag that resource with a reference to itself. For example, a resource that is created as part of an AWS CloudFormation stack automatically inherits the stack's `aws:cloudformation:<stack-name>` tag.
- You can create up to 50 tags for each resource. Tags with the `aws:` prefix do not count towards this number.



Examples of common tags

Common tags:

- Environment (production, test)
- Application
- Owner
- Department
- Cost center
- Purpose
- Stack

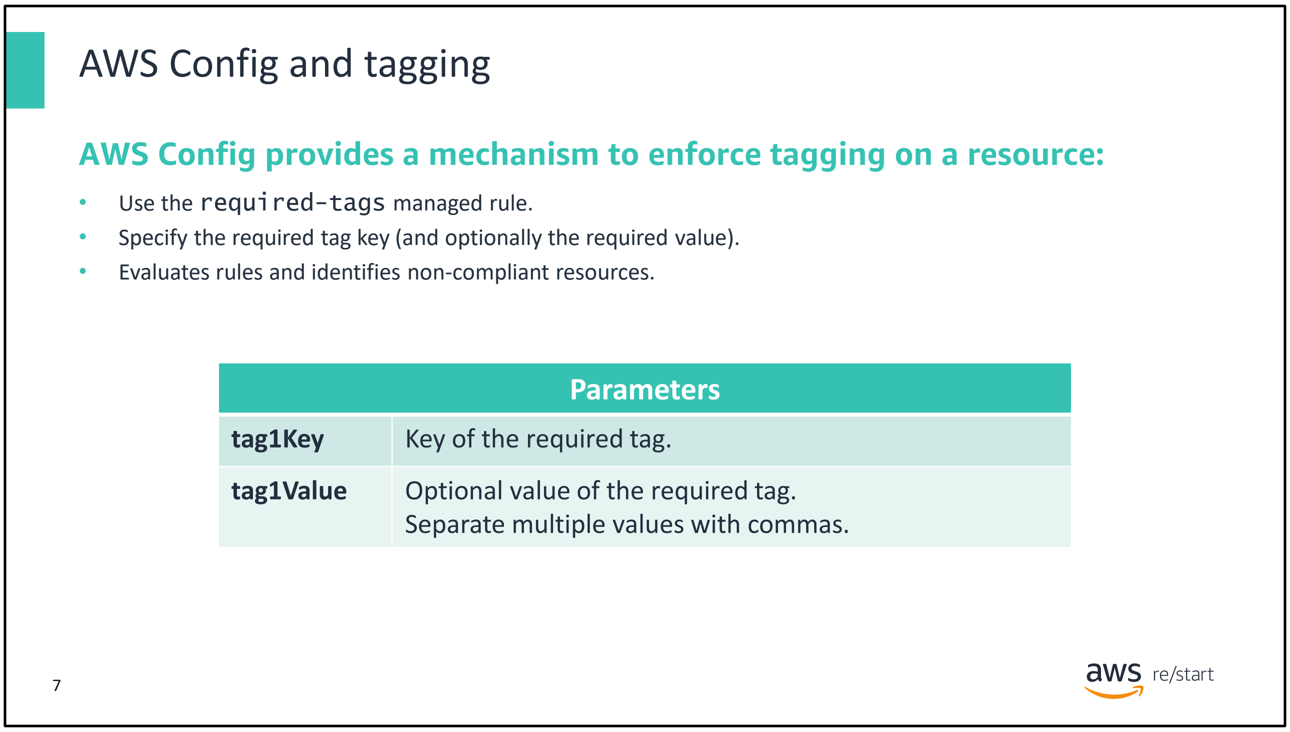
Environment = ***Production***

Cost Center = ***Marketing***

Application = ***Promotions***

aws re/start

Tags should represent organizationally relevant dimensions. This slide lists some examples of meaningful tags because they support the ability to manage resource inventory, access control, cost tracking, automation, and organization.



AWS Config and tagging

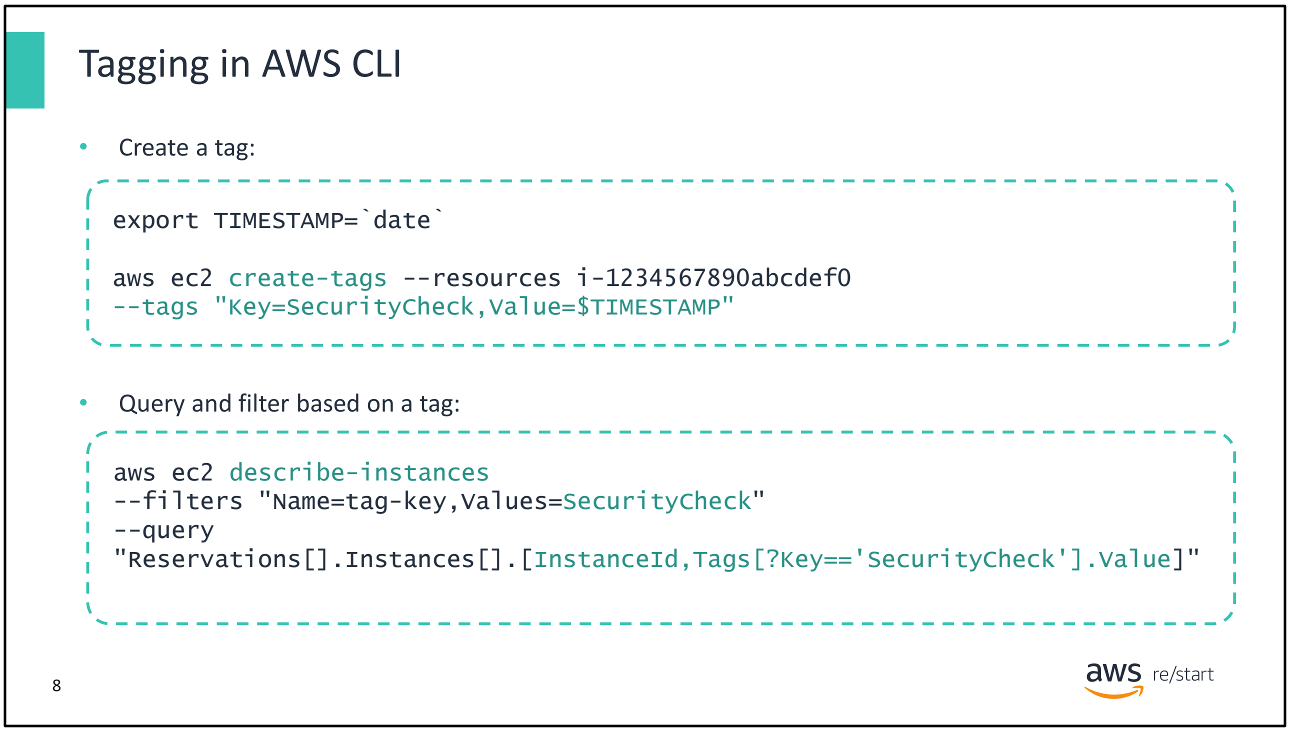
AWS Config provides a mechanism to enforce tagging on a resource:

- Use the `required-tags` managed rule.
- Specify the required tag key (and optionally the required value).
- Evaluates rules and identifies non-compliant resources.

Parameters	
tag1Key	Key of the required tag.
tag1Value	Optional value of the required tag. Separate multiple values with commas.

AWS Config provides *AWS managed rules*, which are predefined, customizable rules that AWS Config uses to evaluate whether your AWS resources comply with common best practices. You can customize the behavior of a managed rule to suit your needs. For example, you could use the *required-tags* managed rule to quickly assess whether a specific tag is applied to your resources. This rule enables you to specify the key of the required tag and, optionally, its value. After you activate the rule, AWS Config compares your resources to the defined conditions and reports any non-compliant resources. The evaluation of a managed rule can occur when a resource changes, or on a periodic basis.

For more information about the AWS Config required-tags managed rule, refer to the required-tags page in the AWS Config Developer Guide.



Tagging in AWS CLI

- Create a tag:

```
export TIMESTAMP=`date`

aws ec2 create-tags --resources i-1234567890abcdef0
--tags "Key=SecurityCheck,Value=$TIMESTAMP"
```

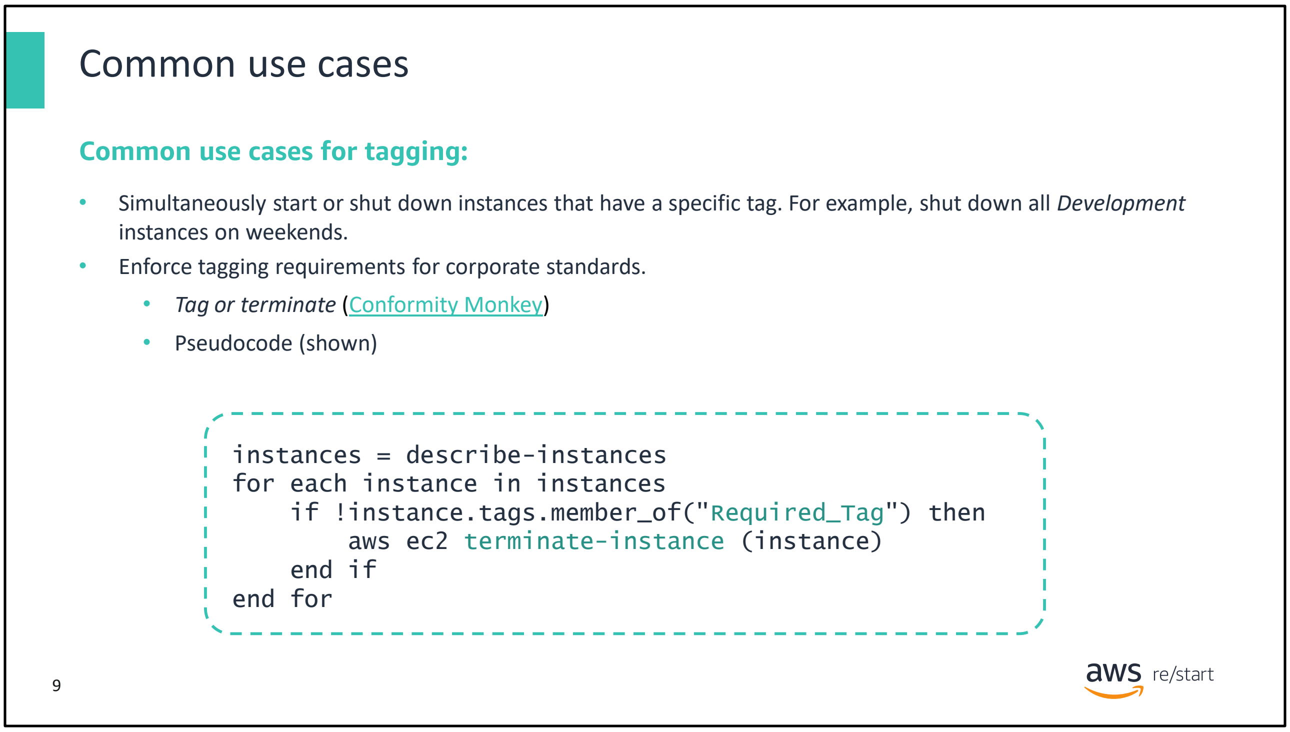
- Query and filter based on a tag:

```
aws ec2 describe-instances
--filters "Name=tag-key,Values=SecurityCheck"
--query
"Reservations[].Instances[].[InstanceId,Tags[?Key=='SecurityCheck'].Value]"
```

The examples show how to use the AWS Command Line Interface (AWS CLI) to create a tag and how to use a tag to query resources.

The first command creates a tag that is named *SecurityCheck* on the instance, which is identified by the ID of *i-1234567890abcdef0*. The command assigns the value of the *current timestamp* to the tag. The tag's key and value attributes are specified using the *`--tags`* option of the *`create-tags`* command. You can specify tags for EC2 instances and Amazon Elastic Block Store (Amazon EBS) volumes as part of the application programming interface (API) call that creates them. If the call creates both instances and volumes, you can specify distinct tags for the instance and for each associated volume.

The second command lists the instance ID and the tag value of all instances in the current Region that have a tag named *SecurityCheck*. The *`--filters`* option of the *`describe-instances`* command identifies the key that was used to filter the query results. The *`--query`* option identifies the data that is returned by the command.



Common use cases

Common use cases for tagging:

- Simultaneously start or shut down instances that have a specific tag. For example, shut down all *Development* instances on weekends.
- Enforce tagging requirements for corporate standards.
  - *Tag or terminate* (Conformity Monkey)
  - Pseudocode (shown)

```
instances = describe-instances
for each instance in instances
    if !instance.tags.member_of("Required_Tag") then
        aws ec2 terminate-instance (instance)
    end if
end for
```

Two common use cases for tagging include using a tag to shut down and restart all instances that have a specific tag, and the *tag or terminate* compliance check.

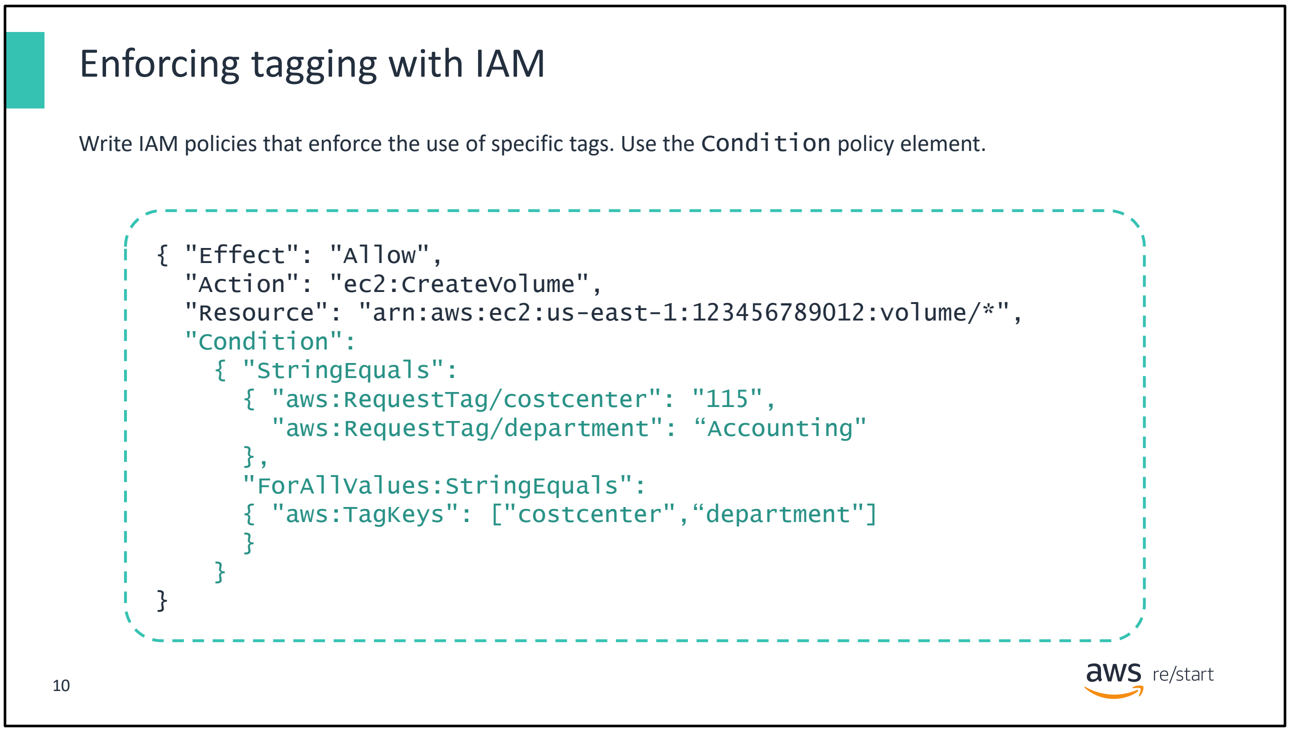
The first use case involves tagging all instances so they have an attribute that indicates the environment that they run in, such as *Development*, *Test*, or *Production*. To save costs, you can create a script that automatically shuts down *Development* instances on weekends and restarts them at the beginning of the week.

The *tag or terminate* strategy is shown in pseudocode. In this scenario, a company or division issues a set of policies regarding what tags must be placed on running resources. A script periodically examines all instances that run under an AWS account and checks that the required tags exist. If an instance does not have the required tags, the instance is terminated for being non-compliant.

In practice, companies that implement this strategy usually stagger deployment over several weeks. In Phase 1, machines are not immediately shut down. Instead, the *tag or terminate* script is written so that it sends an email message to the IAM user who created the instance. The message warns the IAM user that their instance might be shut down soon because it does not comply with corporate policies. In Phase 2 of the rollout, instances are actually shut down, and an explanation of the shutdown is sent to the IAM user who created the resource.

After instances are properly tagged to describe their role and function in an organization, companies can create other automated processes that implement company-wide cost-saving strategies. Companies can also use tags to organize billing reports that reflect internal cost structures and to achieve more accurate reporting for cost allocation.

For more information about the Conformity Monkey compliance check, refer to the following Netflix blog post: Conformity Monkey--Keeping your cloud instances following best practices.



Enforcing tagging with IAM

Write IAM policies that enforce the use of specific tags. Use the `Condition` policy element.

```
{ "Effect": "Allow",
  "Action": "ec2:CreateVolume",
  "Resource": "arn:aws:ec2:us-east-1:123456789012:volume/*",
  "Condition":
    { "StringEquals":
      { "aws:RequestTag/costcenter": "115",
        "aws:RequestTag/department": “Accounting"
      },
      "ForAllValues:StringEquals":
      { "aws:TagKeys": ["costcenter",“department"]
      }
    }
}
```

You can also write IAM policies that enforce the use of specific tags. For example, when you create a resource, you could use an IAM policy to enforce the use of the *department* and *costcenter* tags to help you achieve more accurate reporting for cost allocation. Other tag-related scenarios that you can enforce by using an IAM policy include:

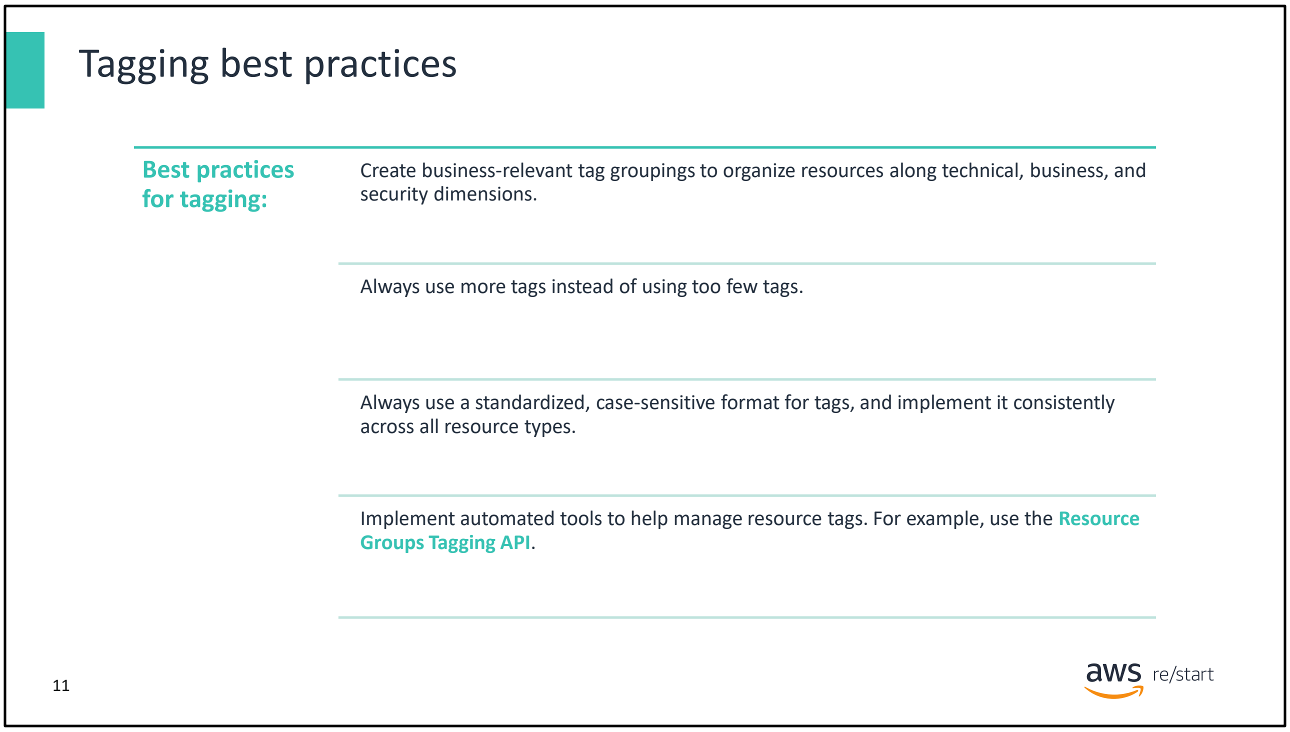
- Blocking the deletion of tags that are required by corporate standards
- Disallowing the creation of new tags for specific existing resources
- Requiring the use of encryption for any EBS volume that is created with a specific tag value

These tagging requirements are expressed in an IAM policy through the *Condition* policy element. In the example IAM policy, the requirements are enforced when a request to create an EBS volume is processed:

- The request must include a *costcenter* and *department* tag, and only those tags. This scenario is indicated in the *ForAllValues* modifier.
- The values for the *costcenter* and *department* tags in the request must be *115* and *Accounting*, respectively. This scenario is expressed in the keys for *aws:RequestTag/costcenter* and *aws:RequestTag/department*, respectively.

The net effect of this policy is that all newly created EBS volumes must have a *costcenter* tag and a *department* tag with the respective values of *115* and *Accounting*.

For more information on tagging EC2 instances and EBS volumes on creation, refer to the following AWS News Blog post: New - Tag EC2 Instances & EBS Volumes on Creation.



Tagging best practices

Best practices for tagging:	
	Create business-relevant tag groupings to organize resources along technical, business, and security dimensions.
	Always use more tags instead of using too few tags.
	Always use a standardized, case-sensitive format for tags, and implement it consistently across all resource types.
	Implement automated tools to help manage resource tags. For example, use the **Resource Groups Tagging API**.

aws re/start

This slides lists some best practices for building a useful and effective tagging strategy.

One best practice is to use automated tools to manage resource tags, such as the *Resource Groups Tagging application programming interface (API)*. This API enables the programmatic control of tags. This practice makes it easier to automatically manage, search, and filter tags and resources. For example, you can use the API to perform the following tasks:

- Tag and untag supported resources.
- Use tag-based filters to search for resources.
- List all existing tag keys.
- List all existing values for specified keys.

For more information about tagging strategies, refer to the Tagging Best Practices AWS whitepaper.

For more information about the Resource Groups Tagging API, see: Resource Groups Tagging API Reference documentation.

Key takeaways

- A tag is a **descriptive label** that is attached to an AWS resource.
- Use an **IAM policy** to **require the use of specific tags** on a resource, and **AWS Config** to periodically **verify that all resources are tagged**.
- A good tagging strategy **defines tags** along dimensions that **enable resource access control**, **cost-tracking**, **automation**, and **organization**.

Some key takeaways from this lesson include:

- A tag is a descriptive label that is attached to an AWS resource.
- Use an IAM policy to mandate the use of specific tags on a resource, and AWS Config to periodically verify that all resources are tagged.
- A good tagging strategy defines tags along dimensions that enable resource access control, cost-tracking, automation, and organization.