AWS Naming Convention Best Practices

Tagging

September 11, 14

Table of Contents

Tagging Overview

Tags provide identification and classification of AWS resources by the association of descriptive metadata, for example, application identifier, environment, or owner. Each tag consists of a key and a value, both of which are user-defined strings. Once defined, tags can be used as a filter when requesting resources, such as Amazon EC2 instances, based on tag keys or values. Tags facilitate resource management at scale and appear on the monthly invoice, which can be particularly useful for cost allocation and control.

Tagging Restrictions

The following basic restrictions apply to tags:

- Maximum number of tags per resource – 10
- Maximum key length – 127 Unicode characters
- Maximum value length – 255 Unicode characters
- Tag keys and values are case sensitive
- Tag keys must be unique per resource
- Do not use the "aws:" prefix in your tag name or values because it is reserved for AWS use.

The following table lists all Amazon EC2 resources which support tags:

Resource	Tagging Support
AMI	Yes
Bundle Task	No
Customer Gateway	Yes

DHCP Option	Yes
Amazon EBS Volume	Yes
Instance Store Volume	No
Elastic IP	No
Instance	Yes
Key Pair	No
Load Balancer	No
Network ACL	Yes
Network Interface	Yes
Placement Group	No
Reserved Instance	Yes
Reserved Instance Listing	No
Route Table	Yes
Spot Instance Request	Yes
Security Group	Yes
Subnet	Yes
Virtual Private Gateway	Yes
VPC	Yes
VPN Connection	Yes

For the latest information on using tags, see Tagging your Amazon EC2 Resources.

Tagging Naming Conventions

Due to the 10 tag per-resource limit, a best practice is to combine several of the tag keys and values into compound tags. For example, instead of creating 3 keys called "OwnerName", "OwnerPhone", and "OwnerEmail," the 3 keys could be combined into 1 key called "OwnerContact," which could contain the compound values of Name, Phone, and Email address using a pipe delimiter (e.g. John Doe|1-555-1212|jdoe@org.com).

Style Rules

- Tag key names are case-sensitive and can contain mixed-case letters, numbers, underscores, and hyphens.
- Tag key names should use upper CamelCase (a.k.a. Pascal case), a convention that combines words/abbreviations by beginning each word with a capital letter such as "MiscMetadata" and "SupportEndpoints".
- Externally managed tags -- for example, those supported by managed services partners -- should use a unique tag key name prefix not to exceed 12 characters followed by a hypen ("-") character (e.g. PartnerName-Contact or 3rdPartyName-SupportEndpoint).

- Tag values are case-sensitive and should not use the semi-colon (";"), equal sign ("="), or pipe ("|") characters as these are used as delimiters in compound values.
- Compound tag value key names should use Pascal case followed by an equal sign ("=") such as KeyName1=value1|value2|value3;KeyName2=value1|value2|value3

Example 1 – Using simple tag value:

Keys	Value
KeyName	value

Example 2 – Using compound tag value with pipe delimiter:

Keys	Value
KeyName	value1|value2|value3

Example 3 – Using compound tag value with semi-colon and pipe delimiter:

Keys	Value
KeyName	Key1=value1|value2|value3;Key2=value1|value2|value3

The following are several conventionally used tag keys with compound tag values:

Keys	Value
Application	Value contains application name with optional unique identifier and optional domain/resource name using pipe delimiter (e.g. Org Web Site|7231a74d|www.org.com)
Environment	Value contains name of environment with optional tier name or business critical level using pipe delimiter (e.g. Production|Business Critical Tier 1)
Contact	Value contains contact information (Name, Phone, Email) using semi-colon and pipe delimiter (e.g. OwnerContact=John Doe|1-555-1212|jdoe@org.com;IncidentContact=John Doe|1-555-1212|jdoe@org.com)
SupportEndpoints	Value contains support endpoints using semi-colon delimiter (e.g. AlarmEndPoint=arn::SNS::topic1;ChangeApprovalEndPoint= arn::SQS::changequeue)
MiscMetadata	Value contains miscellaneous metadata using semi-colon delimiter and pipe delimiter (e.g. key1=value1|value2|value3;key2=value1|value2|value3)

PartnerName-Contact	Value contains externally managed services partner contact information (Name, Phone, Email) using semi-colon and pipe delimiter (e.g. OwnerContact=John Doe|1-555-1212|jdoe@org.com;IncidentContact=John Doe|1-555-1212|jdoe@org.com)

Tagging API and CLI Commands

Use the following API and CLI commands to add, update, list, and delete the tags for your resources. The documentation for each command provides examples.

Description	Amazon EC2 CLI	AWS CLI	AWS Tools for Windows PowerShell	API Action
Adds or overwrites one or more tags for the specified resource(s).	ec2-create-tags	create-tags	New-EC2Tag	CreateTags
Deletes the specified tags from the specified resource(s).	ec2-delete-tags	delete-tags	Remove-EC2Tag	DeleteTags
Describes one or more tags for your resources.	ec2-describe-tags	describe-tags	Get-EC2Tag	DescribeTags

Tagging Filters using AWS Tools for Windows PowerShell

Using AWS Tools for Windows PowerShell and the Get-EC2Tag cmdlet, Amazon EC2 Resource IDs from within a region can be returned on a filtered basis. The Get-EC2Tag cmdlet supports filtering on the "key", "resource-id", "resource-type", and "value." Basic wild-card filtering using * and ? are also supported. The maximum Amazon EC2 Resource

IDs that can be returned in a single call is 1,000. For more information on the Get-EC2Tag cmdlet, see the AWS Tools for Windows PowerShell documentation.

Filtering Amazon EC2 Resource Id example based on tag key name:

In this example, the Get-EC2Tag cmdlet will return the Resource Ids of each Amazon EC2 instance with a tag key name that matches the string "Contact"

```
$filter = New-Object Amazon.EC2.Model.Filter -Property @(Name="key"; Values="Contact")
Get-EC2Tag -Filter $filter
```

Filtering Amazon EC2 Resource Id example based on tag value using wild-cards:

In this example, the Get-EC2Tag cmdlet will return the Resource Ids of each Amazon EC2 instance with a tag value that contains the string "John Doe."

```
$filter = New-Object Amazon.EC2.Model.Filter -Property @(Name="value"; Values="*John Doe*")
Get-EC2Tag -Filter $filter
```

Filtering Amazon EC2 Resource Id example based on tag key name and tag value using wild-cards:

In this example, the Get-EC2Tag cmdlet will return the Resource Ids of each Amazon EC2 instance with a tag key name that matches the string "Contact" and a tag value that contains the string "John Doe."

```
$filter1 = New-Object Amazon.EC2.Model.Filter -Property @(Name="key"; Values="Contact")
$filter2 = New-Object Amazon.EC2.Model.Filter -Property @(Name="value"; Values="*John Doe*")
$filters = $filter1, $filter2
Get-EC2Tag -Filter $filters
```

Sample output from the Get-EC2Tag cmdlet when filtering using the criteria above.

ResourceId	ResourceType	Key	Value

i-2d477420	instance	Contact	OwnerContact=John Doe|1-55...
i-2d477421	instance	Contact	OwnerContact=John Doe|1-55...
i-2d477422	instance	Contact	OwnerContact=John Doe|1-55...

Filtering Amazon EC2 Resource Id example based on missing tag key name:

In this example, the Get-EC2Instance cmdlet will return the Instance Ids of each Amazon EC2 instance that have not been tagged with a tag key name that matches the string "Contact." This filter provides the ability to audit the Amazon EC2 instances for missing tags to enforce a tagging standard.

```
Get-EC2Instance | % {if (($_.Instances | % {$_.Tags | Where {$_.Key -eq "Contact"}}) -eq
$null) {$_}}
```

Tagging Summary

Tagging is an effective tool to help manage AWS resources at increasing scale, providing the ability to identify, classify and locate resources for management and billing purposes. Amazon EC2 filtering provides a way to both locate tagged resources and validate that the tagging standards in your organization are being properly implemented. Naming best practices can be leveraged to achieve consistency across your environment and maximize the benefits that tagging has to offer.