Tagging Best Practices

AWS Whitepaper

Tagging Best Practices: AWS Whitepaper

Table of Contents

Tagging Best Practices: Implement an Effective AWS Resource Tagging Strategy

Publication date: **December 2018** (*Document Revisions* (p. 20))

AWS allows customers to assign metadata to their AWS resources in the form of tags. Each tag is a simple label consisting of a customer-defined key and an optional value that can make it easier to manage, search for, and filter resources by purpose, owner, environment, or other criteria. AWS tags can be used for many purposes.

Introduction: Tagging Use Cases

Amazon Web Services allows customers to assign metadata to their AWS resources in the form of tags. Each tag is a simple label consisting of a customer-defined key and an optional value that can make it easier to manage, search for, and filter resources by purpose, owner, environment, or other criteria. AWS tags can be used for many purposes.

Tags for AWS Console Organization and Resource Groups

Tags are a great 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. By default, the AWS Management Console is organized by AWS service. However, the Resource Groups tool allows customers to create a custom console that organizes and consolidates AWS resources based on one or more tags or portions of tags. Using this tool, customers can consolidate and view data for applications that consist of multiple services and resources in one place.

Tags for Cost Allocation

AWS Cost Explorer and Cost and Usage Report support the ability to break down AWS costs by tag. Typically, customers use business tags such as cost center, business unit, or project to associate AWS costs with traditional financial reporting dimensions within their organization. However, a cost allocation report can include any tag. This allows customers to easily associate costs with technical or security dimensions, such as specific applications, environments, or compliance programs. Table 1 shows a partial cost allocation report.

Table 1: 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

Tags for Automation

Resource or service-specific tags are often used to filter resources during infrastructure automation activities. Tags can be used to opt into or out of automated tasks, or to identify specific versions of

resources to archive, update, or delete. For example, many customers run automated start/stop scripts that turn off development environments during non-business hours to reduce costs. In this scenario, Amazon Elastic Compute Cloud (Amazon EC2) instance tags are a simple way to identify the specific development instances to opt into or out of this process.

Tags for Operations Support

Tags can be used to integrate support for AWS resources into day-to-day operations including IT Service Management (ITSM) processes such as Incident Management. For example, Level 1 support teams could use tags to direct workflow and perform business service mapping as part of the triage process when a monitoring system triggers an alarm. Many customers also use tags to support processes such as backup/restore and operating system patching.

Tags for Access Control

AWS Identity and Access Management (IAM) policies support tag-based conditions, enabling customers to constrain permissions based on specific tags and their values. For example, IAM user or role permissions can include conditions to limit access to specific environments (for example, development, test, or production) or Amazon Virtual Private Cloud (Amazon VPC) networks based on their tags.

Tags for Security Risk Management

Tags can be assigned to identify resources that require heightened security risk management practices, for example, Amazon EC2 instances hosting applications that process sensitive or confidential data. This can enable automated compliance checks to ensure that proper access controls are in place, patch compliance is up to date, and so on.

The sections that follow identify recommended best practices for developing a comprehensive tagging strategy.

Best Practices for Identifying Tag Requirements

Employ a Cross-Functional Team to Identify Tag Requirements

As noted in the introduction, tags can be used for a variety of purposes. To develop a comprehensive strategy, it's best to assemble a cross-functional team to identify tagging requirements. Tag stakeholders in an organization typically include IT Finance, Information Security, application owners, cloud automation teams, middleware and database administration teams, and process owners for functions such as patching, backup/restore, monitoring, job scheduling, and disaster recovery.

Rather than meeting with each of these functional areas separately to identify their tagging needs, conduct tagging requirements workshops with representation from all stakeholder groups, so that each can hear the perspectives of the others and integrate their requirements more effectively into the overall strategy.

Use Tags Consistently

It's important to employ a consistent approach in tagging your AWS resources. If you intend to use tags for specific use cases, as illustrated by the examples in the introduction, you will need to rely on the consistent use of tags and tag values. For example, if a significant portion of your AWS resources are missing tags used for cost allocation, your cost analysis and reporting process will be more complicated and time-consuming, and probably less accurate. Likewise, if resources are missing a tag that identifies the presence of sensitive data, you may have to assume that all such resources contain sensitive data, as a precautionary measure.

A consistent approach is warranted even for tags identified as optional. For example, if you employ an opt-in approach for automatically stopping development environments during non-working hours, identify a single tag for this purpose rather than allowing different teams or departments to use their own; resulting in many different tags all serving the same purpose.

Assign Owners to Define Tag Value Propositions

Consider tags from a cost/benefit perspective when deciding on a list of required tags. While AWS does not charge a fee for the use of tags, there may be indirect costs (for example, the labor needed to assign and maintain correct tag values for each relevant AWS resource).

To ensure tags are useful identify an owner for each one. The tag owner has the responsibility to clearly articulate its value proposition. Having tag owners may help avoid unnecessary costs related to maintaining tags that are not used.

Focus on Required and Conditionally Required Tags

Tags can be required, conditionally required, or optional. Conditionally required tags are only mandatory under certain circumstances (for example, if an application processes sensitive data, you may require a tag to identify the corresponding data classification, such as Personally Identifiable Information or Protected Health Information).

When identifying tagging requirements, focus on required and conditionally required tags. Allow for optional tags, as long as they conform to your tag naming and governance policies, to empower your organization to define new tags for unforeseen or bespoke application requirements.

Start Small; Less is More

Tagging decisions are reversible, giving you the flexibility to edit or change as needed in the future. However, there is one exception—cost allocation tags—which are included in AWS monthly cost allocation reports. The data for these reports is based on AWS services utilization and captured monthly. As a result, when you introduce a new cost allocation tag it takes effect starting from that point in time. The new tag will not apply to past cost allocation reports.

Tags help you identify sets of resources. Tags can be removed when no longer needed. A new tag can be applied to a set of resources in bulk, however, you need to identify the resources requiring the new tag and the value to assign those resources.

Start with a smaller set of tags that are known to be needed and create new tags as the need arises. This approach is recommended over specifying an overabundance of tags that are anticipated to be needed in the future.

Best Practices for Naming Tags and Resources

Topics

Adopt a Standardized Approach for Tag Names

Keep in mind that names for AWS tags are case sensitive so ensure that they are used consistently. For example, the tags **CostCenter** and **costcenter** are different, so one might be configured as a cost allocation tag for financial analysis and reporting and the other one might not be. Similarly, the **Name** tag appears in the AWS Console for many resources, but the **name** tag does not.

A number of tags are predefined by AWS or created automatically by various AWS services. Many AWS-defined tags are named using all lowercase, with hyphens separating words in the name, and prefixes to identify the source service for the tag. For example:

- **aws:ec2spot:fleet-request-id** identifies the Amazon EC2 Spot Instance Request that launched the instance
- **aws:cloudformation:stack-name** identifies the AWS CloudFormation stack that created the resource
- **lambda-console:blueprint** identifies blueprint used as a template for an AWS Lambda function
- **elasticbeanstalk:environment-name** identifies the application that created the resource

Consider naming your tags using all lowercase, with hyphens separating words, and a prefix identifying 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 the resource was created for

The prefix ensures that tags are clearly identified as having been defined by your organization and not by AWS or a third-party tool that you may 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**.

Standardize Names for AWS Resources

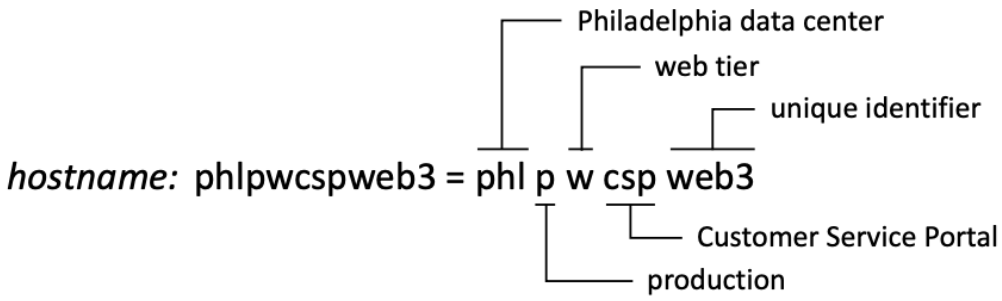
Assigning names to AWS resources is another important dimension of tagging that should be considered. This is the value that is assigned to the predefined AWS **Name** tag (or in some cases by other means), and is mainly used in the AWS Management Console. To understand the idea here, it's probably not

helpful to have dozens of EC2 instances all named *MyWebServer*. Developing a naming standard for AWS resources will help you keep your resources organized, and can be used in AWS Cost and Usage Reports for grouping related resources together (see also Propagate Tag Values Across Related Resources below).

Topics

EC2 Instances

Naming for EC2 instances is a good place to start. Most organizations have already recognized the need to standardize on server hostnames, and have existing practices in effect. For example, an organization might create hostnames based on several components, such as physical location, environment type (development, test, production), role/purpose, application ID, and a unique identifier:



First, note that the various components of a hostname construction process like this are great candidates for individual AWS tags – if they were important in the past, they'll likely be important in the future. Even if these elements are captured as separate, individual tags, it's still reasonable to continue to use this style of server naming to maintain consistency, and substituting a different physical location code to represent AWS or an AWS region.

However, if you're moving away from treating your virtual instances like *pets* and more like *cattle* (which is recommended), you'll want to automate the assignment of server names to avoid having to assign them manually. As an alternative, you could simply use the AWS instance-id (which is globally unique) for your server names.

In either case, if you're also creating DNS names for servers, it's a good idea to associate the value used for the **Name** tag with the Fully Qualified Domain Name (FQDN) for the EC2 instance. So, if your instance name is *phlpwcspweb3*, the FQDN for the server could be *phlpwcspweb3.anycompany.com*. If you'd rather use the instance-id for the **Name** tag, then you should use that in your FQDN (for example, i-06599a38675.anycompany.com).

Other AWS Resource Types

For other types of AWS resources, one approach is to adopt a dot notation consisting of the following name components:

1. **account-name prefix:** for example, production, development, shared-services, audit, etc.
2. **resource-name:** free-form field for the logical name of the resource
3. **type suffix:** for example, subnet, sg, role, policy, kms-key, etc.

See Table 2 for examples of tag names for other AWS resource types.

Table 2: Sample tag names for other AWS resource types

Resource Type	Example AWS Resource Name	account-name	resource-name	type
Subnet	**prod.public-az1.subnet**	Production	public-az1	subnet
Subnet	**services.az2.subnet**	Shared Services	az2	subnet
Security Group	**prod.webserver.sg**	Production	webserver	sg
Security Group	**dev.webserver.sg**	Development	webserver	sg
Security Group	**services.dmz.sg**	Shared Services	dmz	sg
IAM Role	**prod.ec2-s3-access.role**	Production	ec2-s3-access	role
IAM Role	**dr.ec2-s3-access.role**	Disaster Recovery	ec2-s3-access	role
KMS Key	**prod.anycompany.kms-key**	Production	AnyCompany	kms-key

Some resource types limit the character set that can be used for the name. In such cases, the dot characters can be replaced with hyphens.

Best Practices for Cost Allocation Tags

Align Cost Allocation Tags with Financial Reporting Dimensions

AWS provides detailed cost reports and data extracts to help you monitor and manage your AWS spend. When you designate specific tags as cost allocation tags in the AWS Billing and Cost Management Console, billing data for AWS resources will include them. Remember, billing information is point-in-time data, so cost allocation tags appear in your billing data only after you have (1) specified them in the Billing and Cost Management Console and (2) tagged resources with them.

A natural place to identify the cost allocation tags you need is by looking at your current IT financial reporting practices. Typically, financial reporting covers a variety of dimensions, such as business unit, cost center, product, geographic area, or department. Aligning cost allocation tags with these financial reporting dimensions simplifies and streamlines your AWS cost management.

Use Both Linked Accounts and Cost Allocation Tags

AWS resources are created within accounts, and billing reports and extracts contain the AWS account number for all billable resources, regardless of whether or not the resources have tags. You can have multiple accounts, so creating different accounts for different financial entities within your organization is a way to clearly segregate costs.

AWS provides options for consolidated billing by associating payer accounts and linked accounts. You can also use AWS Organizations to create master accounts with associated member accounts to take advantage of the additional centralized management and governance capabilities.

Organizations may design their account structure based on a number of factors including fiscal isolation, administrative isolation, access isolation, blast radius isolation, engineering, and cost considerations (refer to the References (p. 18) section for links to relevant articles on AWS Answers). Examples include:

- Creating separate accounts for production and non-production to segregate communications and access for these environments
- Creating a separate account for shared services components and utilities
- Creating a separate audit account to capture log files for security forensics and monitoring
- Creating separate accounts for disaster recovery

Understand your organization's account structure when developing your tagging strategy since alignment of some of the financial reporting dimensions may already be captured by your account structure.

Avoid Multi-Valued Cost Allocation Tags

For shared resources, you may need to allocate costs to several applications, projects, or departments. One approach to allocating costs is to create multi-valued tags that contain a series of allocation codes, possibly with corresponding allocation ratios, for example:

```
anycompany:cost-center = 1600|0.25|1625|0.20|1731|0.50|1744|0.05
```

If designated as a cost allocation tag, such tag values appear in your billing data. However, there are two challenges with this approach: (1) the data will have to be post-processed to parse the multi-valued tag values and produce more detailed records, and (2) you will need to establish a process to accurately set and maintain the tag values.

If possible, consider identifying existing cost sharing or chargeback mechanisms within your organization —or create new ones—and associate shared AWS resources to individual cost allocation codes defined by that mechanism.

Tag Everything

When developing a tagging strategy, be wary of focusing only on the set of tags needed for your EC2 instances. Remember that AWS allows you to tag most types of resources that generate costs on your billing reports. Apply your cost allocation tags across all resource types that support tagging to get the most accurate data for your financial analysis and reporting.

Best Practices for Tag Governance and Data Management

Topics

Integrate with Authoritative Data Sources

You may decide to include tags on your AWS resources for which data is already available within your organization. For example, if you are using a Configuration Management Database (CMDB), you may already have a process in place to store and maintain metadata about your applications, databases, and environments. Configuration Items (CIs) in your CMDB may have attributes including application or server owner, technical issue resolver groups, cost center or charge code, data classification, etc.

Rather than redundantly capturing and maintaining such existing metadata in AWS tags, consider integrating your CMDB with AWS. The integration can be bi-directional, meaning that data sourced from the CMDB can be copied to tags on AWS resources, and data that can be sourced from AWS (for example, IP addresses, instance IDs, and instance types) can be stored as attributes in your Configuration Items.

If you integrate your CMDB with AWS in this way, extend your AWS tag naming convention to include an additional prefix to identify tags that have externally-sourced values, for example:

- **anycompany:cmdb:application-id** – the CMDB Configuration Item ID for the application that owns the resource
- **anycompany:cmdb:cost-center** – the Cost Center code associated with the owning application, sourced from the CMDB
- **anycompany:cmdb:application-owner** – the individual or group that owns the application associated with this resource, sourced from the CMDB

This makes it clear that the tags are provided for convenience, and that the authoritative source of the data is the CMDB. Referencing authoritative data sources, rather than redundantly maintaining the same data in multiple systems, is a general data management best practice.

Use Compound Tag Values Judiciously

Initially, AWS limited the number of tags for a given resource to 10, resulting in some organizations combining several data elements into a single tag, using delimiters to segregate the different attributes, as in:

```
EnvironmentType = Development;Webserver;Tomcat-6.2;Tier 2
```

In 2016, the number of tags per resource was increased to 50 (with a few exceptions, such as S3 objects). Because of this, it's generally recommended to follow good data management practice by including only one data attribute per tag.

However, there are some situations where it may make sense to combine several related attributes together. Some examples include:

1. For contact information, as shown in Table 3.

   Table 3: Examples of compound and single tag values

Compound Tag Values	**anycompany:business-contact = John Smith;john.smith@anycompany.com; +12015551212**
	anycompany:technical-contact = Susan Jones;sue.jones@anycompany.com; +12015551213
Single Tag Values	anycompany:business-contact-name = John Smith
	anycompany:business-contact-email = john.smith@anycompany.com
	anycompany:business-contact-phone = +12015551212
	anycompany:technical-contact-name = Susan Jones
	anycompany:technical-contact-email = sue.jones@anycompany.com
	anycompany:technical-contact-phone = +12015551213

2. For multi-valued tags where a single attribute can have several homogenous values. For example, a resource supporting multiple applications might use a pipe-delimited list:

```
anycompany:cmdb:application-ids = APP012|APP045|APP320|APP450
```

   However, before introducing multi-valued tags, consider the source of the information, and how the information will be used if captured in an AWS tag. If there is an authoritative source for the data in question, then any processes requiring the information may be better served by referencing the authoritative source directly, rather than a tag. Also, as recommended in this paper, avoid multi-valued cost allocation tags if possible.

3. For tags used for automation purposes. Such tags typically capture opt-in and automation status information. For example, if you implement an AWS Lambda function to automatically back up EBS volumes by taking snapshots, you might use a tag that contains a short JSON document:

```
anycompany:auto-snapshot = { “frequency”: “daily”, “last-backup”:
“2018-04-19T21:18:00.000+0000” }
```

There are many automation solutions available at AWS Labs (https://github.com/awslabs) and the AWS Marketplace (https://aws.amazon.com/marketplace) that make use of compound tag values in their implementations.

Use Automation to Proactively Tag Resources

AWS offers a variety of tools to help you implement proactive tag governance practices; by ensuring that tags are consistently applied when resources are created.

AWS CloudFormation provides a common language for provisioning all the infrastructure resources in your cloud environment. CloudFormation templates are simple text files that create AWS resources in an automated and secure manner. When you create AWS resources using AWS CloudFormation templates, you can use the CloudFormation Resource Tags property to apply tags to certain resource types upon creation.

AWS Service Catalog allows organizations to create and manage catalogs of IT services that are approved for use on AWS. These IT services can include everything from virtual machine images, servers, software, and databases to complete multi-tier application environments. AWS Service Catalog enables a self-service capability for users, allowing them to provision the services they need while also helping you to maintain consistent governance – including the application of required tags and tag values.

AWS Identity and Access Management (IAM) enables you to manage access to AWS services and resources securely. Using IAM, you can create and manage AWS users and groups, and use permissions to allow or deny their access to AWS resources. When you create IAM policies, you can specify resource-level permissions, which include specific permissions for creating and deleting tags. In addition, you can include condition keys, such as aws:RequestTag and aws:TagKeys, which will prevent resources from being created if specific tags or tag values are not present.

Constrain Tag Values with AWS Service Catalog

Tags are not useful if they contain missing or invalid data values. If tag values are set by automation, the automation code can be reviewed, tested, and enhanced to ensure that valid tag values are used. When tags are entered manually, there is the opportunity for human error.

One way to reduce human error is by using AWS Service Catalog. One of the key features of AWS Service Catalog is TagOption libraries. With TagOption libraries, you can specify required tags as well as their range of allowable values. AWS Service Catalog organizes your approved AWS service offerings, or products, into multiple portfolios. You can use TagOption libraries at the portfolio level, or even at the individual product level, to specify the range of allowable values for each tag.

Propagate Tag Values Across Related Resources

Many AWS resources are related. For example, an EC2 instance may have several Elastic Block Storage (EBS) volumes and one or more Elastic Network Interfaces (ENIs). For each EBS volume, many EBS snapshots may be created over time.

For consistency, best practice is to propagate tags and tag values across related resources. If resources are created by AWS CloudFormation templates, they are created together in groups called stacks from a common automation script, which can be configured to set tag values across all resources in the stack. For resources not created via AWS CloudFormation, you can still implement automation to automatically propagate tags from related resources. For example, when EBS snapshots are created, you can copy any tags present on the EBS volume to the snapshot. Similarly, you can use CloudWatch Events to trigger a Lambda function to copy tags from an S3 bucket to objects within the bucket any time S3 objects are created.

Lock Down Tags Used for Access Control

If you decide to use tags to supplement your access control policies, you will need to ensure that you restrict access to creating, deleting, and modifying those tags. For example, you can create IAM policies that use conditional logic to grant access to (1) EC2 instances for an IAM group created for developers and (2) for EC2 instances tagged as development. This could be further restricted to developers for a particular application based on a condition in the IAM policy that identifies the relevant application ID.

While the use of tags for this purpose is convenient, it can be easily circumvented if users have the ability to modify tag values in order to gain access that they should not have. Take preventative measures against this by ensuring that your IAM policies include deny rules for actions such as **ec2:CreateTags** and **ec2:DeleteTags**.

Even with this preventative measure, IAM policies that grant access to resources based on tag values should be used with caution and approved by your Information Security team. You may decide to use this approach for convenience in certain situations. For example, use strict IAM policies (without conditions based on tags) for restricting access to production environments; but for development environments, grant access to application-specific resources via tags to help developers avoid inadvertently affecting each other's work.

Remediate Untagged Resources

Automation and proactive tag management are important, but are not always effective. Many customers also employ reactive tag governance approaches to identify resources that are not properly tagged and correct them. Reactive tag governance approaches include (1) programmatically using tools such as the Resource Tagging API, AWS Config rules, and custom scripts; or (2) manually using Tag Editor and detailed billing reports.

Tag Editor is a feature of the AWS Management Console that allows you to search for resources using a variety of search criteria and add, modify, or delete tags in bulk. Search criteria can include resources with or without the presence of a particular tag or value. The AWS Resource Tagging API allows you to perform these same functions programmatically.

AWS Config enables you to assess, audit, and evaluate the configurations of your AWS resources. AWS Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against optimal configurations. With AWS Config, you can create rules to check resources for required tags and it will continuously monitor your resources against those rules. Any non-compliant resources are identified on the AWS Config Dashboard and via notifications. In the case where resources are initially tagged properly but their tags are subsequently changed or deleted, AWS Config will find them for you.

You can use AWS Config with CloudWatch Events to trigger automated responses to missing or incorrect tags. An extreme example would be to automatically stop or quarantine non-compliant EC2 instances.

The most suitable governance approach for an organization primarily depends on its AWS maturity model, but even experienced organizations use a combination of proactive and reactive governance techniques.

Implement a Tag Governance Process

Keep in mind that once you've settled on a tagging strategy for your organization, you will need to adapt it as you progress through your cloud journey. In particular, it's likely that requests for new tags will surface and need to be addressed. A basic tag governance process should include:

- impact analysis, approval, and implementation for requests to add, change, or deprecate tags;
- application of existing tagging requirements as new AWS services are adopted by your organization;
- monitoring and remediation of missing or incorrect tags; and
- periodic reporting on tagging metrics and key process indicators.

Conclusion

AWS resource tags can be used for a wide variety of purposes, from implementing a cost allocation process to supporting automation or authorizing access to AWS resources. Implementing a tagging strategy can be challenging for some organizations, due to the number of stakeholder groups involved and considerations such as data sourcing and tag governance. This white paper recommends a way forward based on a set of best practices, to get you started quickly with a tagging strategy that you can adapt as your organization's needs evolve over time.

Contributors

The following individuals and organizations contributed to this document:

- Brian Yost, Senior Consultant, AWS Professional Services

References

Tagging Use Cases

- AWS Tagging Strategies
- Tagging Your Amazon EC2 Resources
- Centralized multi-account and multi-Region patching with AWS Systems Manager Automation

Align Tags with Financial Reporting Dimensions

- Monthly Cost Allocation Report
- User-Defined Cost Allocation Tags
- Cost Allocation for EBS Snapshots
- AWS-Generated Cost Allocation Tags

Use Both Linked Accounts and Cost Allocation Tags

- Consolidated Billing for Organizations
- AWS Multiple Account Billing Strategy
- AWS Multiple Account Security Strategy
- What Is AWS Organizations?

Tag Everything

User-Defined Cost Allocation Tags

Integrate with Authoritative Data Sources

ITIL Asset and Configuration Management in the Cloud

Use Compound Tag Values Judiciously

Now Organize Your AWS Resources by Using up to 50 Tags per Resource

Use Automation to Proactively Tag Resources

- How can I use IAM policy tags to restrict how an EC2 instance or EBS volume can be created?
- How to Automatically Tag Amazon EC2 Resources in Response to API Events

- Supported Resource-Level Permissions for Amazon EC2 API Actions: Resource-Level Permissions for Tagging
- Example Policies for Working with the AWS CLI or an AWS SDK: Tagging Resources
- Resource Tag

Constrain Tag Values with AWS Service Catalog

- AWS Service Catalog Announces AutoTags for Automatic Tagging of Provisioned Resources
- AWS Service Catalog TagOption Library

Propagate Tag Values Across Related Resources

CloudWatch Events for EBS Snapshots

Lock Down Tags Used for Access Control

- AWS Services That Work with IAM
- How do I create an IAM policy to control access to Amazon EC2 resources using tags?
- Controlling Access to Amazon VPC Resources

Remediate Untagged Resources

- Resource Groups and Tagging for AWS
- AWS Resource Tagging API

Document Revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

update-history-change	update-history-description	update-history-date
Initial publication (p. 20)	First published.	December 1, 2018

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.