AWS Cloud Operations & Migrations Blog

Automatically tag new AWS resources based on identity or role

by Bill Dry | on 02 NOV 2020 | in Advanced (300), Amazon CloudWatch, AWS Lambda, AWS Systems Manager, Management Tools, Python, Technical How-To | Permalink | Share

You might have heard the adage to "tag early, tag often" in infrastructure planning and design sessions. Using accurate, meaningful tags on your AWS resources is a best practice. Consistently applied resource tags deliver organizational benefits such as accurate cost allocation, granular access controls, precisely routed operation issues, and simplified resource operating state changes. This blog post provides steps for ensuring your new AWS resources are tagged appropriately.

Solution overview

The auto-tagging solution described in this post applies your organization's required tags to newly created resources using an automated workflow. It includes a rule created in Amazon CloudWatch Events, a resource tag repository such as AWS Systems Manager Parameter Store, and an AWS Lambda function.

By following the steps in this post, you create a CloudWatch event rule, Parameter Store entries, and a Lambda function to enable the auto-tagging solution explained in this post.

Figure 1 shows this solution's architecture and its five-step workflow.

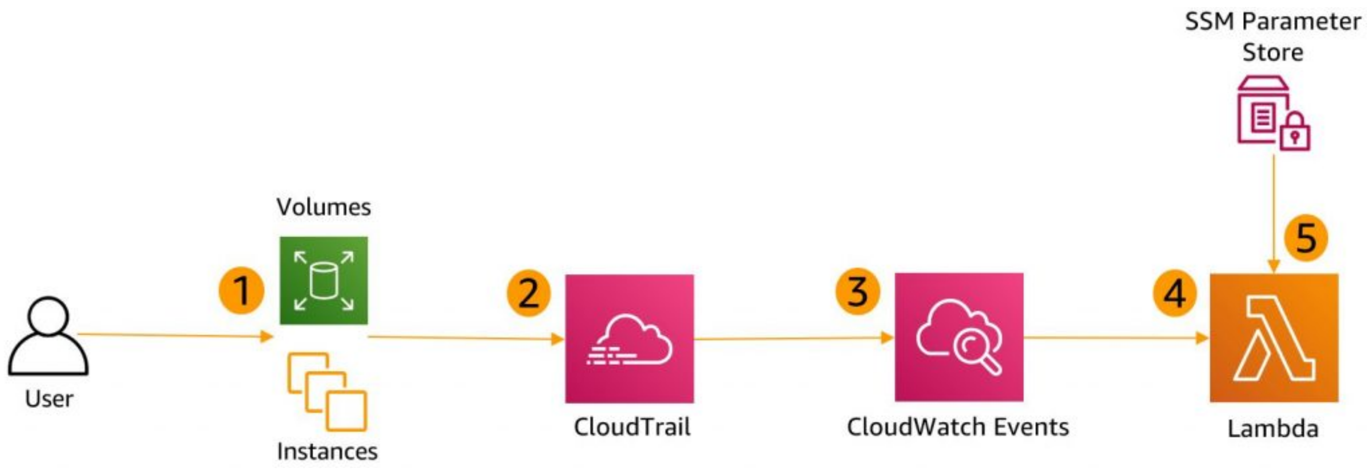


Figure 1: Auto-tagging solution workflow

Workflow steps

1. A user creates Amazon Elastic Compute Cloud (Amazon EC2) instances.
2. AWS CloudTrail logs a resource creation API event.
3. A CloudWatch event rule monitors and is triggered upon the creation of events like **RunInstances**.
4. The CloudWatch event rule detects an applicable event, and then invokes a Lambda function to tag the resources.
5. Lambda retrieves the required tags from Parameter Store and tags the new resource.

Solution setup

Follow these steps to set up the auto-tagging solution.

Step 1: Clone the solution repo

You'll find the AWS Identity and Access Management (IAM) permissions policy document, IAM trust policy document, and Lambda function in this GitHub repo. Run the `git clone` command to clone this GitHub repo to your local machine:

```
git clone git@github.com:aws-samples/resource-auto-tagger.git
```

Step 2: Select a CloudTrail trail

You need a CloudTrail trail to detect and respond to AWS resource creation API events. If you do not already have a trail, follow the steps in Creating a Trail in the AWS CloudTrail User Guide. Here is an example AWS CLI command for creating a trail for this auto-tagging solution:

```
aws cloudtrail create-trail --name resource-creation-events --s3-bucket-name blog-demos
```

The Amazon EC2 **RunInstances** API CloudTrail event provides a lot of tagging information. For example, you can extract:

- The single sign-on (SSO) user ID of the entity that created the resource from the `principalId` key
- The IAM role the entity assumed during resource creation from the `arn` key.
- The date/time of resource creation from the `eventTime` key.

- The EC2 instance ID from `instanceId`.

This CloudTrail event also provides detail about other resources that were created or updated when the EC2 instance was created. This means you can extract and automatically tag your instances with detail like the VPC ID and subnet ID.

Step 3: Store your required AWS resource tags

There are two options for storing your required resource tag keys and values. You can use either or both of these options.

Option 1: Apply your required resource tags to the resource creator's IAM role. When the resource creator assumes the role to create a resource, the Lambda function described in this blog post retrieves the resource tags assigned to that IAM role. The Lambda function then applies those retrieved IAM role tags to the newly created AWS resource.

Option 2: Use AWS Systems Manager Parameter Store to store tags per resource creator role or SSO user ID using a hierarchy called a path. For this example, use this path partitioning scheme:

/auto-tag/*IAM_Role*/*User_ID*/tag/*tag_key*/*tag_value*

Here is an example AWS CLI command to create a tag entry in Parameter Store. This example path sets this tag's key to `team` and its value to `SouthEast-migration`.

```
aws ssm put-parameter --name /auto-tag/sso-role1/sso-user1/tag/team --value SouthEast-migration --type SecureString
```

Step 4: Authorize the Lambda function

The resource-auto-tagger Lambda function used in this solution needs permission to interact with other AWS services on your behalf. Create an IAM permissions policy that allows the Lambda function to invoke the service actions shown in the following table.

Required IAM Service Actions	Purpose
CloudWatch:PutMetricData	Every time this Lambda function is executed, it must record an entry in CloudWatch.
Logs: CreateLogGroup, CreateLogStream, DescribeLogGroups, DescribeLogStreams, GetLogEvents, PutLogEvents	This Lambda function must create, list, and write CloudWatch log groups and streams to record its actions and outcomes to the log stream at `/aws/lambda/function name`
IAM:ListRoleTags	The Lambda function must list and retrieve the tags assigned to the IAM role that created the new AWS resource.
EC2:CreateTags, DescribeInstances, DescribeVolumes	In this example, the Lambda function tags EC2 instances and associated Amazon Elastic Block Store (Amazon EBS) volumes, so it must receive IAM permissions to describe instances and volumes and to create and apply tags to instances and volumes.
SSM:GetParametersByPath	The Lambda function needs permission to retrieve your required tags entered in Parameter Store by hierarchical path.

Create an IAM permissions policy for the AWS services and actions shown in the table, and then assign that policy to an IAM role the resource-auto-tagger Lambda function assumes every time it is run. The GitHub repo you cloned in step 1 of this post contains an example IAM permissions policy and an example IAM trust policy.

The following example AWS CLI commands create an IAM permissions policy and link it to the IAM role. (Replace 123456789012 in the example command with your AWS account number.)

```
aws iam create-policy --policy-name resource-auto-tagger-lambda-permissions-policy --policy-document file://iam-lambda-role-permissions-policy.json
```

```
aws iam create-role --role-name resource-auto-tagger-lambda-role --assume-role-policy-document file://iam-lambda-role-trust-policy.json
```

```
aws iam attach-role-policy --role-name resource-auto-tagger-lambda-role --policy-arn arn:aws:iam::123456789012:policy/resource-auto-tagger-lambda-permissions-policy
```

Step 5: Create the resource-auto-tagger Lambda function

Now, use the AWS CLI to create the Lambda function that performs the resource tagging when it is triggered by the CloudWatch event rule. This Lambda function uses the Python 3.8 runtime.

The following example AWS CLI command creates the resource-auto-tagger Lambda function. (Replace 123456789012 in the example command with your AWS account number.)

```
aws lambda create-function --function-name resource-auto-tagger --runtime python3.8 --role arn:aws:iam::123456789012:role/resource-auto-tagger-lambda-role --timeout 6 --code S3Bucket=blog-demos,S3Key=auto-tag/resource-auto-tagger.zip,S3ObjectVersion=null --handler lambda_handler
```

How resource-auto-tagger works

Now, let's dig into the mechanics of the Lambda function to understand how it correctly tags the newly created AWS resources when it's triggered.

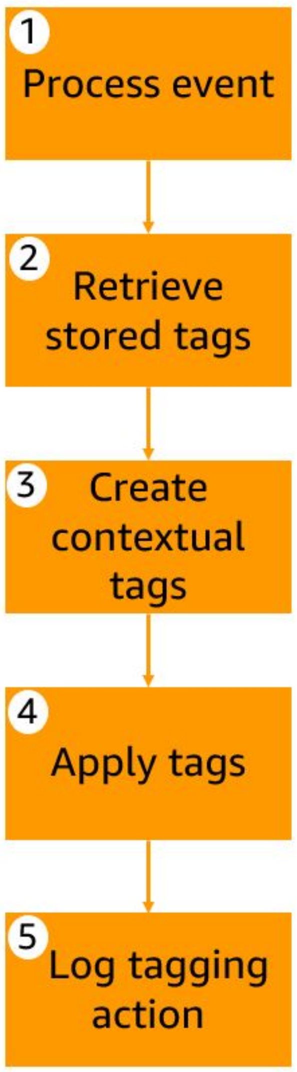


Figure 2: Resource-auto-tagger Lambda function workflow

The resource-auto-tagger Lambda function performs the following five tasks shown in Figure 2.

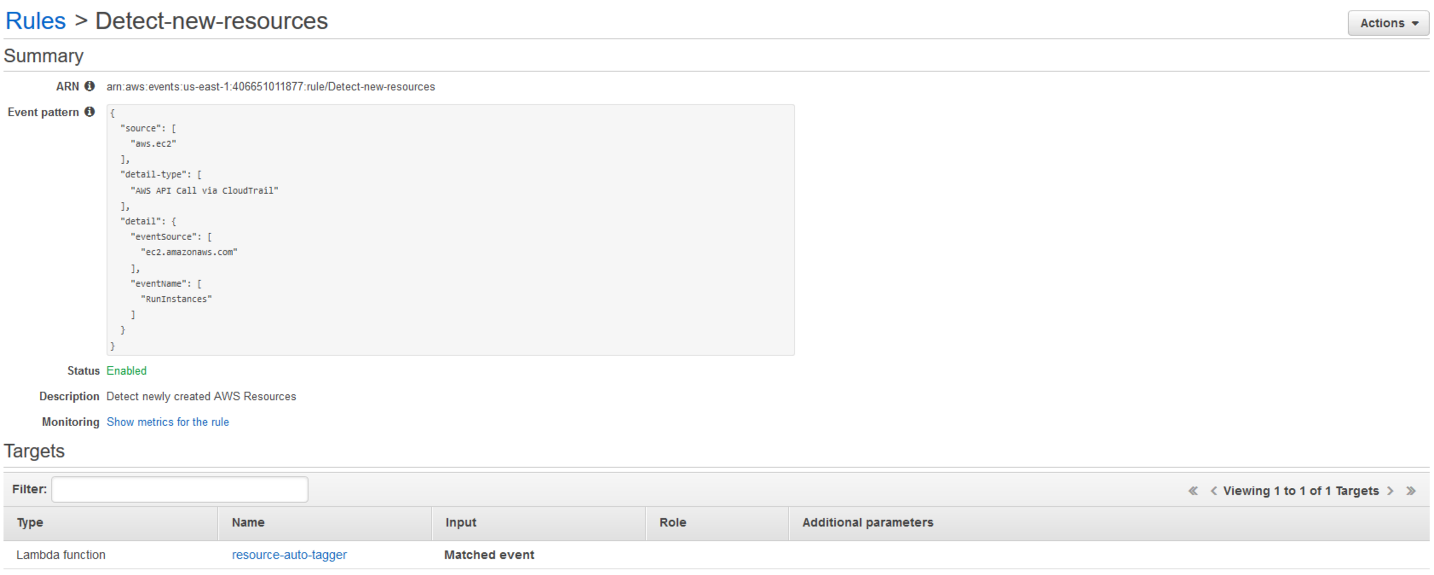
1. Receives, extracts, and formats the CloudTrail event into a Python data dictionary.
2. Retrieves the required resources tags to apply to the newly created AWS resources.
3. Creates contextual resource tags such as **Date created**.
4. Applies required resource tags to the newly created AWS resources.
5. Places an entry into CloudWatch Logs to memorialize the resource tagging event.

Step 6: Create a rule in CloudWatch Events

Create a rule in CloudWatch Events to trigger on the Amazon EC2 **RunInstances** API action. For information, see Creating a CloudWatch Events Rule That Triggers on an AWS API Call Using AWS CloudTrail in the Amazon CloudWatch Events User Guide. Use the following settings for the rule:

- For **Event source**, choose **Event Pattern**.
- For **Service Name**, choose EC2.
- For **Event Type**, choose **AWS API Call via CloudTrail**.
- Choose **Specific operation(s)**, and then enter **RunInstances**.
- For **Targets**, choose the Lambda function you created in Step 5.

Figure 3 shows the Detect-new-resources rule.


Rules > Detect-new-resources

Actions

Summary

ARN arn:aws:events:us-east-1:406651011877:rule/Detect-new-resources

Event pattern
```
{
  "source": [
    "aws.ec2"
  ],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "ec2.amazonaws.com"
    ],
    "eventName": [
      "RunInstances"
    ]
  }
}
```

Status Enabled

Description Detect newly created AWS Resources

Monitoring Show metrics for the rule

Targets

Filter:

Viewing 1 to 1 of 1 Targets

Type	Name	Input	Role	Additional parameters
Lambda function	resource-auto-tagger	Matched event		

Figure 3: CloudWatch Events rule reporting new AWS resources

Step 7: Verify the auto-tagging functionality

After you deploy your Lambda function and give it the appropriate IAM permissions through an assigned IAM role, create your Parameter Store repository. This repository will store your required resources tags by identity and role. Now you can test and verify the auto-tagging functionality by simply creating an EC2 instance in your AWS account.

The following example AWS CLI command creates an EC2 instance:

```
aws ec2 run-instances --image-id ami-0c94855ba95c71c99 --instance-type t2.micro --key-name your_key --tag-specifications
'ResourceType=instance,Tags=[{Key=Name,Value=auto-tag-example-1}]'
```

CloudTrail delivers API events within 15 minutes of their occurrence. After you create the instance, wait 15 minutes, and then check the tags assigned to the instance to verify the resource-auto-tagger Lambda function automatically applied the required tags.

The following AWS CLI command shows how to view the resource tags applied to the new EC2 instance:

```
aws ec2 describe-tags --filters "Name=resource-id,Values=i-your_resource_id"
```

```
{
    "Tags": [
        {
            "ResourceType": "instance",
            "ResourceId": "i-your_resource_id",
            "Value": "ct1-sso-user1",
            "Key": "Created by"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-your_resource_id",
            "Value": "2020-09-21T20:43:07Z",
            "Key": "Date created"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-your_resource_id",
            "Value": "Auto-tag-example-1",
            "Key": "Name"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-your_resource_id",
            "Value": " sso-role1",
            "Key": "Role Name"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-your_resource_id",
            "Value": "324-872-003",
            "Key": "cost_center"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-your_resource_id",
            "Value": "rehost-sql",
            "Key": "project"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-your_resource_id",
            "Value": "SouthEast-migration",
            "Key": "team"
        }
    ]
}
```

Cleanup

To avoid charges after you test the auto-tagging function, delete the Lambda function. The following example AWS CLI command deletes the Lambda function:

```
aws lambda delete-function --function-name resource-auto-tagger
```

If you entered resource tags into Systems Manager Parameter Store, here is an example AWS CLI command to delete them:

```
aws ssm delete-parameter --name /auto-tag/sso-role1/sso-user1/tag/team
```

Conclusion

In this post, I showed how when CloudTrail reports a watched API call, a CloudWatch event rule will trigger a Lambda function that automatically tags newly created EC2 instances with your required tags. The Parameter Store in AWS Systems Manager provides a hierarchical and secure way to store your required resource tags organized by user identity and IAM role. Using this auto-tagging technique with other service creation API calls, such as the **CreateBucket** action in Amazon Simple Storage Service (Amazon S3), you can create additional CloudWatch event rules and Lambda functions to automatically tag other AWS resource types as your builders and automation tools create them.

About the author

Bill Dry

Bill is a Senior Technical Account Manager for AWS Enterprise Support. He developed a variety of software applications ranging from digital signage to video banking over the past eleven years. Outside work, Bill enjoys coaching youth basketball and firing up the grill for backyard barbecues.