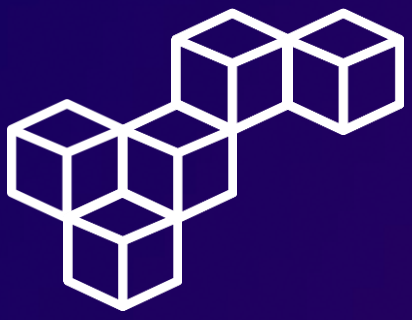
AWS re:Start LAB

Managing Resources with Tagging

WEEK 11

aws re/start



Overview

Tagging in AWS is a powerful practice that enhances the organization and management of cloud resources. By applying tags, you can categorize resources based on various attributes, such as environment, project, or owner, making it easier to identify and manage them. This organizational strategy is crucial for large-scale environments where tracking and maintaining numerous resources can become complex. Effective tagging allows for streamlined resource management, improved cost allocation, and enhanced operational efficiency.

Leveraging tags, you can efficiently locate and manage resources, enabling actions such as stopping or terminating EC2 instances based on specific criteria. Using tools like the AWS CLI or SDKs, these actions can be automated, ensuring that resources are used optimally and cost-effectively. Tagging thus plays a critical role in maintaining a well-organized and cost-efficient AWS environment, ensuring that resources are easily accessible and manageable for operational tasks and automated processes.

Topics covered

- Apply tags to existing AWS resources.
- Find resources based on tags.
- Use the AWS CLI or AWS SDK for PHP to stop and terminate Amazon EC2 instances based on certain attributes of the resource.

Task 1

Using Tags to Manage Resources

Step 1: Connect to the Command Host

Connect to the **Command Host** instance using SSH.

```
support@HP-Pavilion-Laptop:~/Downloads$ ssh -i labsuser.pem ec2-user@35.92.246.41
       ,     #_
       ~\_  ####_        Amazon Linux 2
      ~~  \_#####\
      ~~     \###|       AL2 End of Life is 2025-06-30.
      ~~       \#/ ___
       ~~       V~' '->
        ~~~         /    A newer version of Amazon Linux is available!
          ~~._.   _/
             _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
           _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-10-5-0-136 ~]$
```

Step 2: Find tagged instances

Find all instances in your account that are tagged with a tag of **Project** and a value of **ERPSystem**.

```
[ec2-user@ip-10-5-0-136 ~]$ aws ec2 describe-instances \
> --filter "Name=tag:Project,Values=ERPSystem"
{
    "Reservations": [
        {
            "Instances": [
                {
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "PublicDnsName": "",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "EbsOptimized": false,
                    "LaunchTime": "2024-06-02T23:33:03.000Z",
                    "PrivateIpAddress": "10.5.1.103",
                    "ProductCodes": [],
                    "VpcId": "vpc-0634b6ecadecbc5b9",
                    "CpuOptions": {
                        "CoreCount": 1,
                        "ThreadsPerCore": 2
```

aws re/start

Task 1

Using Tags to Manage Resources

Step 3: Limit the output

Use the --query parameter to limit the output of the previous command to only the instance ID of the discovered instance.

```
[ec2-user@ip-10-5-0-136 ~]$ aws ec2 describe-instances \
> --filter "Name=tag:Project,Values=ERPSystem" \
> --query 'Reservations[*].Instances[*].InstanceId'
[
    [
        "i-0d89071c1d6ad4e87"
    ],
    [
        "i-0c403f2bd8dcf2b2a"
    ],
    [
        "i-02d785f546afb6c58"
    ],
    [
```

Step 4: Include availability zones

Include both the instance ID and the Availability Zone of each instance in your return result.

```
[ec2-user@ip-10-5-0-136 ~]$ aws ec2 describe-instances \
> --filter "Name=tag:Project,Values=ERPSystem" \
> --query 'Reservations[*].Instances[*].{ID:InstanceId,AZ:Placement.AvailabilityZone}'
[
    [
        {
            "AZ": "us-west-2a",
            "ID": "i-0d89071c1d6ad4e87"
        }
    ],
    [
        {
            "AZ": "us-west-2a",
            "ID": "i-0c403f2bd8dcf2b2a"
        }
    ],
    [
```

Task 1

Using Tags to Manage Resources

Step 5: Include a tag

Include the value of the **Project** tag in your output.

```
[ec2-user@ip-10-5-0-136 ~]$ aws ec2 describe-instances \
> --filter "Name=tag:Project,Values=ERPSystem" \
> --query 'Reservations[*].Instances[*].{ID:InstanceId,AZ:Placement.AvailabilityZone,
> Project:Tags[?Key==`Project`] | [0].Value}'
[
    [
        {
            "Project": "ERPSystem",
            "AZ": "us-west-2a",
            "ID": "i-0d89071c1d6ad4e87"
        }
    ],
    [
        {
            "Project": "ERPSystem",
            "AZ": "us-west-2a",
            "ID": "i-0c403f2bd8dcf2b2a"
        }
    ],
    [
```

Step 6: Include more tags

Include the **Environment** and **Version** tags in your output.

```
[ec2-user@ip-10-5-0-136 ~]$ aws ec2 describe-instances \
> --filter "Name=tag:Project,Values=ERPSystem" \
> --query 'Reservations[*].Instances[*].{ID:InstanceId,AZ:Placement.AvailabilityZone,
> Project:Tags[?Key==`Project`] | [0].Value,
> Environment:Tags[?Key==`Environment`] | [0].Value,
> Version:Tags[?Key==`Version`] | [0].Value}'
[
    [
        {
            "Project": "ERPSystem",
            "Environment": "production",
            "AZ": "us-west-2a",
            "Version": "1.0",
            "ID": "i-0d89071c1d6ad4e87"
        }
    ],
    [
        {
```

Task 1

Using Tags to Manage Resources

Step 7: Add a second tag filter

Add a second tag filter to see only the instances associated with the **project** named **ERPSystem** that belong to the **Environment** named **development**.

```
[ec2-user@ip-10-5-0-136 ~]$ aws ec2 describe-instances \
> --filter "Name=tag:Project,Values=ERPSystem" "Name=tag:Environment,Values=development" \
> --query 'Reservations[*].Instances[*].{ID:InstanceId,AZ:Placement.AvailabilityZone,
> Project:Tags[?Key==`Project`] | [0].Value,
> Environment:Tags[?Key==`Environment`] | [0].Value,
> Version:Tags[?Key==`Version`] | [0].Value}'
[
    [
        {
            "Project": "ERPSystem",
            "Environment": "development",
            "AZ": "us-west-2a",
            "Version": "1.0",
            "ID": "i-0c403f2bd8dcf2b2a"
        }
    ],
    [
        {
```

Step 8: Review the Shell script

Examine the contents of the **change-resource-tags.sh** script.

```
[ec2-user@ip-10-5-0-136 ~]$ nano change-resource-tags.sh
[ec2-user@ip-10-5-0-136 ~]$
```

```
 GNU nano 2.9.8                                    change-resource-tags.sh

#!/bin/bash

ids=$(aws ec2 describe-instances \
--filter "Name=tag:Project,Values=ERPSystem" "Name=tag:Environment,Values=development" \
--query 'Reservations[*].Instances[*].InstanceId' \
--output text)

aws ec2 create-tags \
--resources $ids \
--tags 'Key=Version,Value=1.1'
```

aws re/start

Task 1

Using Tags to Manage Resources

Step 9: Run the Shell script

Run the **change-resource-tags.sh** script to overwrite tags.

```
[ec2-user@ip-10-5-0-136 ~]$ ./change-resource-tags.sh
[ec2-user@ip-10-5-0-136 ~]$
```

Step 10: Verify version numbers

Verify that the version number on these instances has been incremented and that other non-development boxes in the **ERPSystem** project have been unaffected.

```
[ec2-user@ip-10-5-0-136 ~]$ aws ec2 describe-instances \
> --filter "Name=tag:Project,Values=ERPSystem" \
> --query 'Reservations[*].Instances[*].{ID:InstanceId, AZ:Placement.AvailabilityZone,
> Project:Tags[?Key==`Project`] |[0].Value,
> Environment:Tags[?Key==`Environment`] | [0].Value,
> Version:Tags[?Key==`Version`] | [0].Value}'
[
    [
        {
            "Project": "ERPSystem",
            "Environment": "production",
            "AZ": "us-west-2a",
            "Version": "1.0",
            "ID": "i-0d89071c1d6ad4e87"
        }
    ],
    [
        {
            "Project": "ERPSystem",
            "Environment": "development",
            "AZ": "us-west-2a",
            "Version": "1.1",
            "ID": "i-0c403f2bd8dcf2b2a"
        }
    ],
    [
        {
```

Task 2

Stop and Start Resources by Tag

Step 1: Review the PHP script

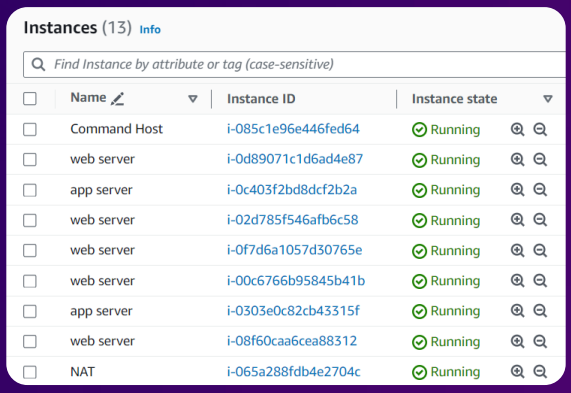
Examine the contents of the **stopinator.php** script.

```
[ec2-user@ip-10-5-0-136 ~]$ cd aws-tools/
[ec2-user@ip-10-5-0-136 aws-tools]$ nano stopinator.php
[ec2-user@ip-10-5-0-136 aws-tools]$
```

```
 GNU nano 2.9.8                                          stopinator.php

#!/usr/bin/php
<?php
# A simple PHP script to start all Amazon EC2 instances and Amazon RDS
# databases within all regions.
#
# USAGE: stopinator.php [-t stop-tags] [-nt exclude-tags]
#
# If no arguments are supplied, stopinator stops every Amazon EC2 and
# Amazon RDS instance running in an account.
#
# -t stop-tag: The tags to inspect to determine if a resource should be
# shut down. Format must follow the same format used by the AWS CLI.
#
# -e exclude-id: The instance ID of an Amazon EC2 instance NOT to terminate. Useful
# when running the stopinator from an Amazon EC2 instance.
#
# -p profile-name: The name of the AWS configuration section to use for
# credentials. Configuration sections are defines in your .aws/credentials file.
# If not supplied, will use the default profile.
#
# -s start: If present, starts instead of stops instances.
```

Step 2: Review EC2 instances

In the EC2 Management Console, review EC2 instances state.



Instances (13) Info

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state
Command Host	i-085c1e96e446fed64	Running
web server	i-0d89071c1d6ad4e87	Running
app server	i-0c403f2bd8dcf2b2a	Running
web server	i-02d785f546afb6c58	Running
web server	i-0f7d6a1057d30765e	Running
web server	i-00c6766b95845b41b	Running
app server	i-0303e0c82cb43315f	Running
web server	i-08f60caa6cea88312	Running
NAT	i-065a288fdb4e2704c	Running

aws re/start

Task 2

Stop and Start Resources by Tag

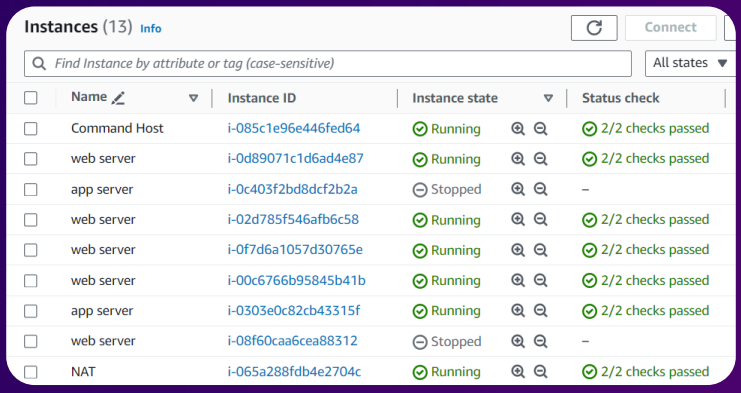
Step 3: Run the PHP script

Run the **stopinator.php** script to stop tagged instances.

```
[ec2-user@ip-10-5-0-136 aws-tools]$ ./stopinator.php -t"Project=ERPSystem;Environment=development"
Region is ap-south-1
PHP Notice:  Array to string conversion in /home/ec2-user/aws-tools/stopinator.php on line 120

        No instances to stop in Array.
Region is eu-north-1
PHP Notice:  Array to string conversion in /home/ec2-user/aws-tools/stopinator.php on line 120

        No instances to stop in Array.
Region is eu-west-3
PHP Notice:  Array to string conversion in /home/ec2-user/aws-tools/stopinator.php on line 120

        No instances to stop in Array.
Region is eu-west-2
```

Step 4: Review stopped instances

In the EC2 Management Console, verify that two instances are stopping or have already been stopped.



aws re/start

Task 2

Stop and Start Resources by Tag

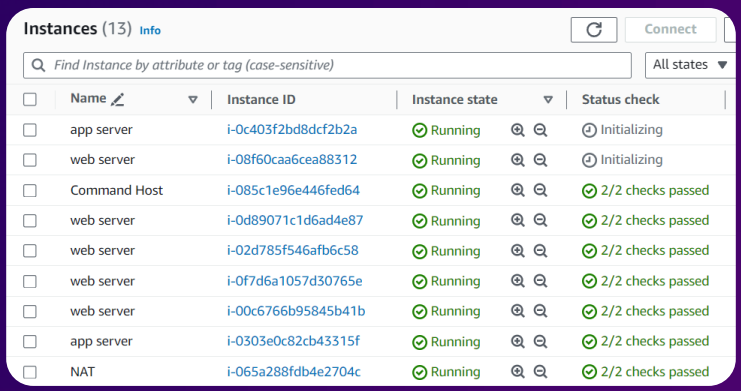
Step 5: Re-run the PHP script

Run the **stopinator.php** script to restart tagged instances.

```
[ec2-user@ip-10-5-0-136 aws-tools]$ ./stopinator.php -t"Project=ERPSystem;Environment=development" -s
Region is ap-south-1
PHP Notice:  Array to string conversion in /home/ec2-user/aws-tools/stopinator.php on line 110

        No instances to start in Array
Region is eu-north-1
PHP Notice:  Array to string conversion in /home/ec2-user/aws-tools/stopinator.php on line 110

        No instances to start in Array
Region is eu-west-3
PHP Notice:  Array to string conversion in /home/ec2-user/aws-tools/stopinator.php on line 110

        No instances to start in Array
Region is eu-west-2
```

Step 6: Review restarted instances

In the EC2 Management Console, verify that the two instances that were previously shut down are now restarting.



aws re/start

Task 3

Challenge: Terminate Non-Compliant Instances

Step 1: Open the Tag-Or-Terminate script

Open the file **terminate-instances.php** with the nano editor.

```
[ec2-user@ip-10-5-0-136 aws-tools]$ nano terminate-instances.php
[ec2-user@ip-10-5-0-136 aws-tools]$
```

Step 2: Review the script

Examine the contents of the **terminate-instances.php** script.

```
 GNU nano 2.9.8                                    terminate-instances.php

#!/usr/bin/php

<?php
require 'vendor/autoload.php';
use Aws\Ec2\Ec2Client;

$region = "us-west-2";
$subnetid = "";
$profile = "default"; # Only needed if not using IAM roles

# Necessary to quell a PHP error.
date_default_timezone_set('America/Los_Angeles');

array_shift($argv);
if (count($argv>0)) {
        do {
                $elem = array_shift($argv);
                if ($elem == "-region") {
                        $region = array_shift($argv);
                } elseif ($elem == "-subnetid") {
                        $subnetid = array_shift($argv);
                }
        } while (count($argv) > 0);
}

# Iterate through all available AWS regions.
$ec2 = Ec2Client::factory(array(
        'profile' =>$profile,
        'region' => $region
));
                                                        [ Read 81 lines ]
```

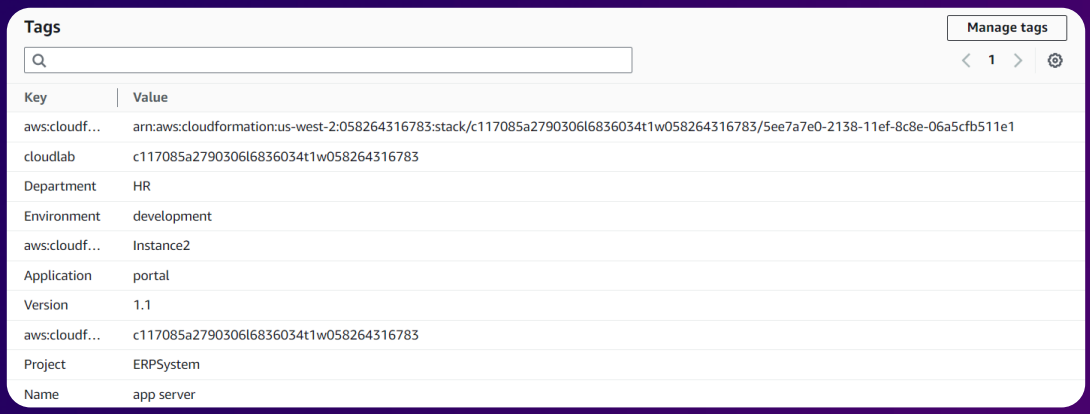
aws re/start

Task 3

Challenge: Terminate Non-Compliant Instances

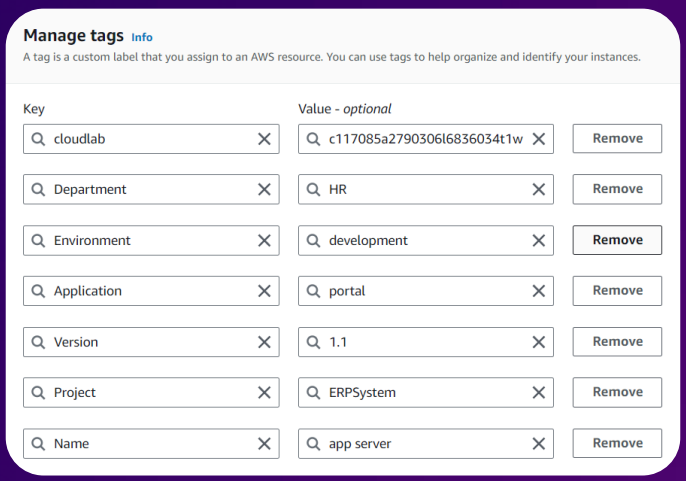
Step 3: Manage tags

Select one of the instances in your private subnet. On the **Tags** tab for the instance, click Manage tags.



Step 4: Remove tags

Find the **Environment** tag, and click Remove. Repeat this process for one other instance in your private subnet.



aws re/start

Task 3

Challenge: Terminate Non-Compliant Instances

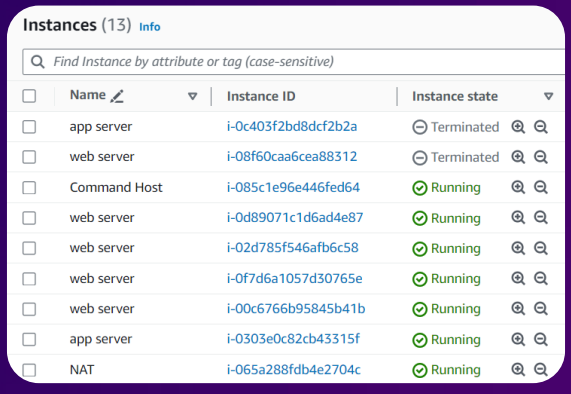
Step 5: Run the Tag-Or-Terminate script

Run the **terminate-instances.php** script. Provide the values for the region and subnetid parameters.

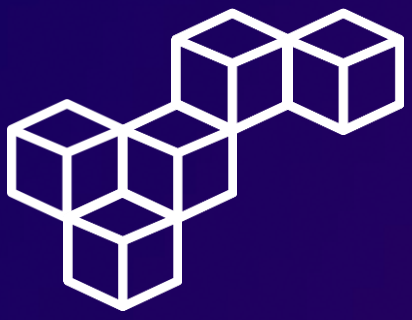
```
[ec2-user@ip-10-5-0-136 aws-tools]$ ./terminate-instances.php -region us-west-2 -subnetid subnet-0dcea2709d69cac70

Checking i-0d89071c1d6ad4e87
Checking i-0c403f2bd8dcf2b2a
Checking i-02d785f546afb6c58
Checking i-0f7d6a1057d30765e
Checking i-00c6766b95845b41b
Checking i-0303e0c82cb43315f
Checking i-08f60caa6cea88312
Terminating instances...
Instances terminated.


[ec2-user@ip-10-5-0-136 aws-tools]$
```

Step 6: Review terminated instances

In the EC2 Management Console, verify that two instances are terminating or have already been terminated.



aws re/start



Conclusions

Tagging
Tagging helps organize and manage AWS resources efficiently, improving resource visibility and operational effectiveness.

aws ec2 describe-instances
The aws ec2 describe-instances command enables detailed retrieval of EC2 instance information, facilitating better resource management and monitoring.

--filter
The --filter parameter allows precise querying of EC2 instances based on tags, streamlining resource identification and management.

--query
The --query parameter provides a flexible way to extract specific information from describe-instances output, enhancing data analysis capabilities.

JMESPath syntax
JMESPath syntax enables powerful and flexible querying of JSON data, making it easier to extract and manipulate information in AWS CLI commands.

aws re/start

Cristhian Becerra

cristhian-becerra-espinoza

+51 951 634 354

cristhianbecerra99@gmail.com

Lima, Peru