Using Amazon Elastic File System (Amazon EFS) with AWS Lambda: Persistent Storage for Serverless Applications

**SPL-DD-300-STEFSD-10-EN - Version 1.0.6**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab Overview**

In this lab, you learn how to quickly create and configure an Amazon Elastic File System (Amazon EFS) file system based on a real-life scenario. In this scenario your company has chosen Amazon EFS as their storage solution. As part of the implementation, you mount your file system in an AWS Cloud9 instance. Also, you will setup EFS as a shared application package repository and test a serverless video processing workflow with AWS Lambda.

OVERVIEW OF THE AWS SERVICES USED IN THIS LAB

**Amazon Elastic File System (Amazon EFS)** provides a simple, serverless, set-and-forget elastic file system for use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, eliminating the need to provision and manage capacity to accommodate growth. Amazon EFS has a simple web services interface that allows you to create and configure file systems quickly and easily. The service manages all the file storage infrastructure for you, meaning that you can avoid the complexity of deploying, patching, and maintaining complex file system configurations.

Amazon EFS supports the Network File System version 4 (NFSv4.1 and NFSv4.0) protocol, so the applications and tools that you use today work seamlessly with Amazon EFS. Multiple compute instances, including Amazon EC2, Amazon ECS, and AWS Lambda, can access an Amazon EFS file system at the same time, providing a common data source for workloads and applications running on more than one compute instance or server.

 For more information, see [What is Amazon Elastic File System?](https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html).

**AWS Lambda** is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. With Lambda, you can run code for virtually any type of application or backend service. All you need to do is supply your code in one of the languages that Lambda supports.

Lambda is an ideal compute service for many application scenarios, as long as you can run your application code using the Lambda standard runtime environment and within the resources that Lambda provides. Lambda is best suited for shorter, event-driven workloads, since Lambda functions run for up to 15 minutes per invocation.

 For more information, see [What is AWS Lambda?](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html).

TOPICS COVERED

By the end of this lab, you will be able to:

* Create and configure an Amazon EFS file system.
* Add file system access point, and configure client permissions.
* Configure an AWS Lambda function to use an EFS file system for persistent storage.
* Install EFS utils client for NFS file access, and mount the file system to an AWS Cloud9 instance.
* Test a serverless video processing workflow with AWS Lambda and Amazon EFS.

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab, you should have a basic understanding of Amazon EFS, AWS Lambda, be familiar with basic navigation of the AWS Management Console, and be comfortable editing scripts using an AWS Cloud9 code editor.

ICON KEY

Various icons are used throughout this lab to call attention to certain aspects of the guide. The following list explains the purpose for each one:

* The keyboard icon specifies that you must run a command.
* The clipboard icon indicates that you can verify the output of a command or edited file by comparing it to the provided example.
* The note icon specifies important hints, tips, guidance, or advice.
* Calls attention to information of special interest or importance. Failure to read the note does not result in physical harm to the equipment or data, but could result in the need to repeat certain steps.
* The “i” circle icon specifies where to find more information.
* The person with a check mark icon indicates an opportunity to check your knowledge and test what you have learned.
* Suggests a moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.

DURATION

This lab requires **60** minutes to complete.

SCENARIO

Your company has built a serverless video processing application. It needs a shared content repository to host the application’s output. It also wants to store any application dependent packages on this persistent storage. Your company has chosen Amazon EFS as its storage solution and has asked you to set up and test the EFS implementation.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING:** **Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

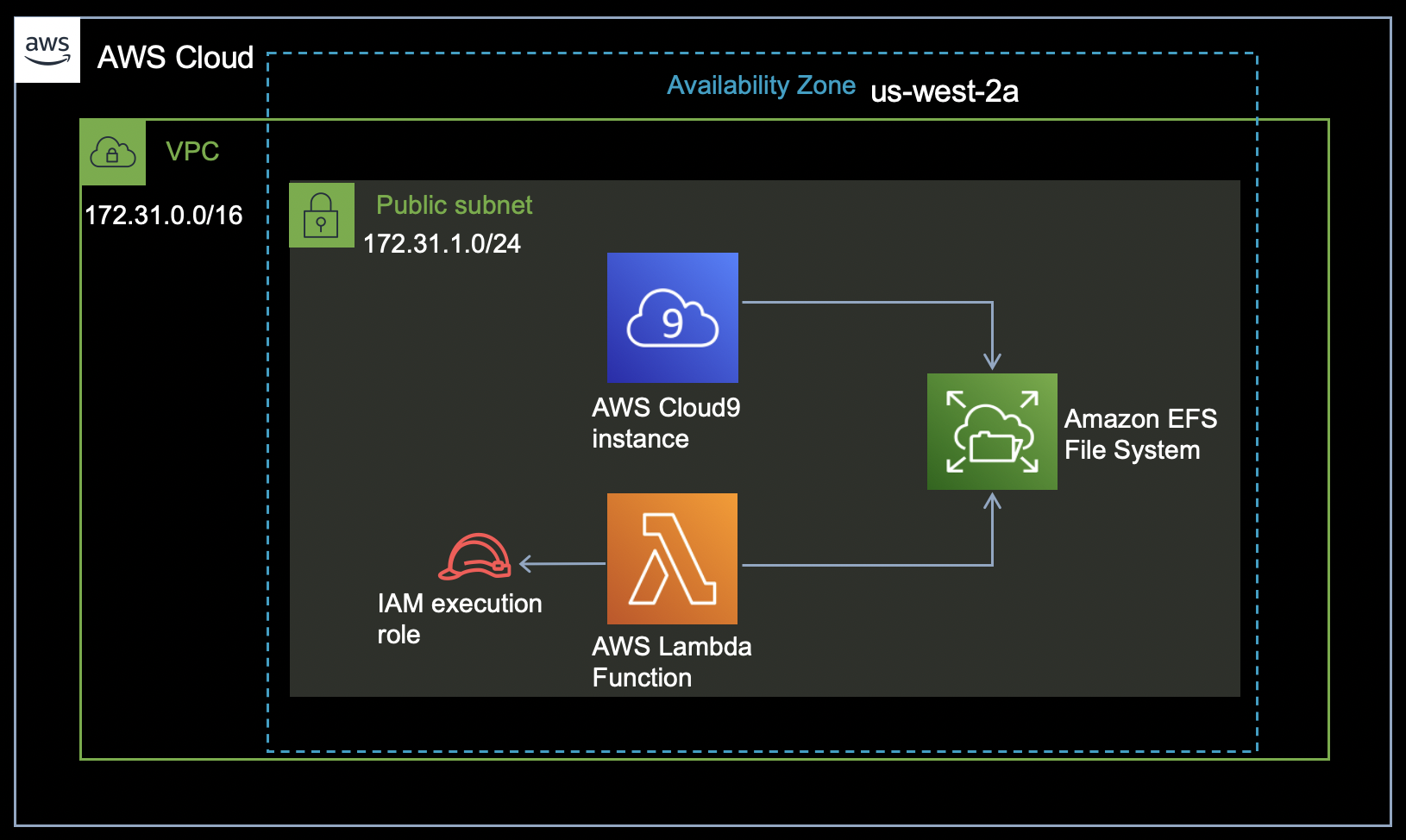
In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

**Task 1: Review the lab architecture**

Take a moment to familiarize yourself with the architecture of this lab environment. It begins with one VPC, one public subnet, one Cloud9 IDE instance, one security group, one Lambda function with ffmpeg deployed as a layer, one IAM execution role for Lambda function, and one sample video to copy to EFS file system.

The following diagram shows the architecture of the lab environment:



**Task 2: Review an Amazon EFS File System**

In this task, you review an existing Amazon EFS file system using the Amazon EFS console and learn about the different configuration settings along the way.

1. In the AWS management console, on the **Services** menu, choose **Storage** and then choose **EFS**.

**Note:** You can also search for

EFS

 at the top of the Services menu.

You should be prompted with the Amazon EFS File systems page, and you should see an existing file system

my\_lambda\_filesystem

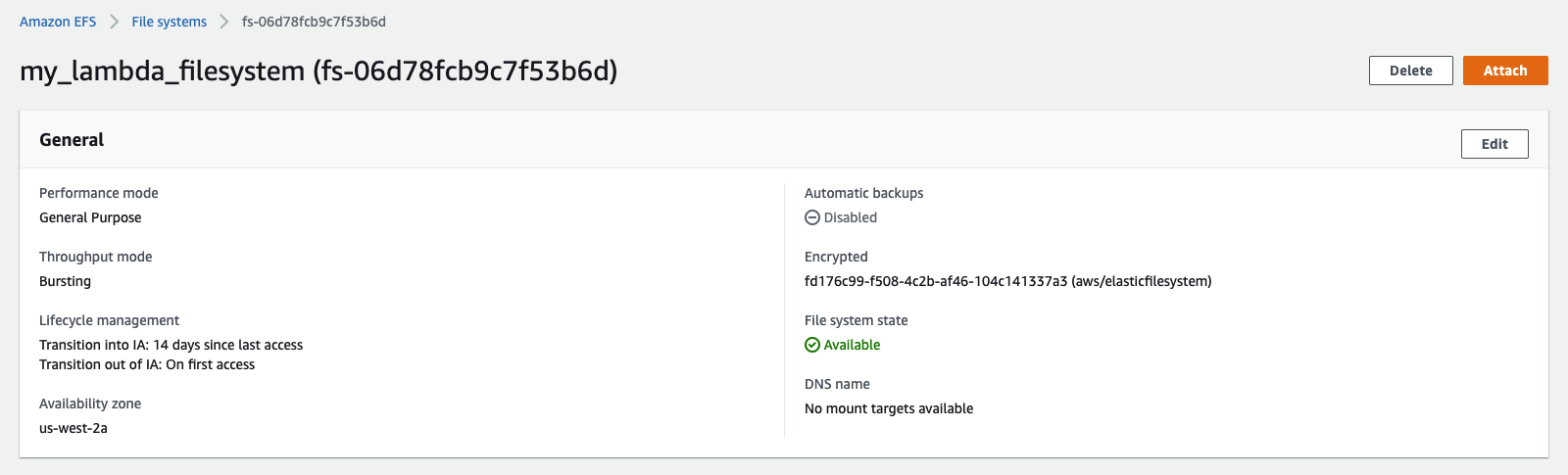
 already created for this lab.

1. Select  the

my\_lambda\_filesystem

 and choose **View details**.

Here you see the following configuration settings for your EFS file system:



* **Performance mode:**

General Purpose

* **Throughput mode:**

Bursting

 For more information, see [Storage classes and performance](https://docs.aws.amazon.com/efs/latest/ug/performance.html#storage-perf).

* **Lifecycle management:**
  + **Transition into IA:**

14 days since last access

* + **Transition out of IA:**

On first access

 Amazon EFS lifecycle management automatically manages cost-effective file storage for your file systems. When enabled, lifecycle management migrates files that have not been accessed for a set period of time to the EFS Standard–Infrequent Access (Standard-IA) or One Zone–Infrequent Access (One Zone-IA) storage class, depending on your file system. You define that period of time by using the Transition into IA lifecycle policy.

* **Availability zone:**

us-west-2a

 Since the file system was created using the One Zone Storage class, you see a single availability zone specified here. One Zone storage classes store file sytem data and metadata redundantly within a single Availability Zone which makes it less expensive than Standard storage classes.

* **Automatic backups:**

Disabled

 For simplicity, automated backups are disabled for this lab. However it is recommended to enable this option for Production workloads. For more information, see [Using AWS Backup to back up and restore Amazon EFS file systems](https://docs.aws.amazon.com/efs/latest/ug/awsbackup.html).

* **Encrypted:** Here you see the KMS KeyId (aws/elasticfilesystem) that was used for encrypting your EFS file system.

 Encryption of data at rest is enabled using the default key for Amazon EFS (aws/elasticfilesystem). For more information, see [Encrypting data at rest](https://docs.aws.amazon.com/efs/latest/ug/encryption-at-rest.html).

* **File system state:** Available
* **DNS name:**

No mount targets available

 You see *No mount targets available* since the mount targets are not yet configured for the EFS file system. The mount targets will be configured in the next task.

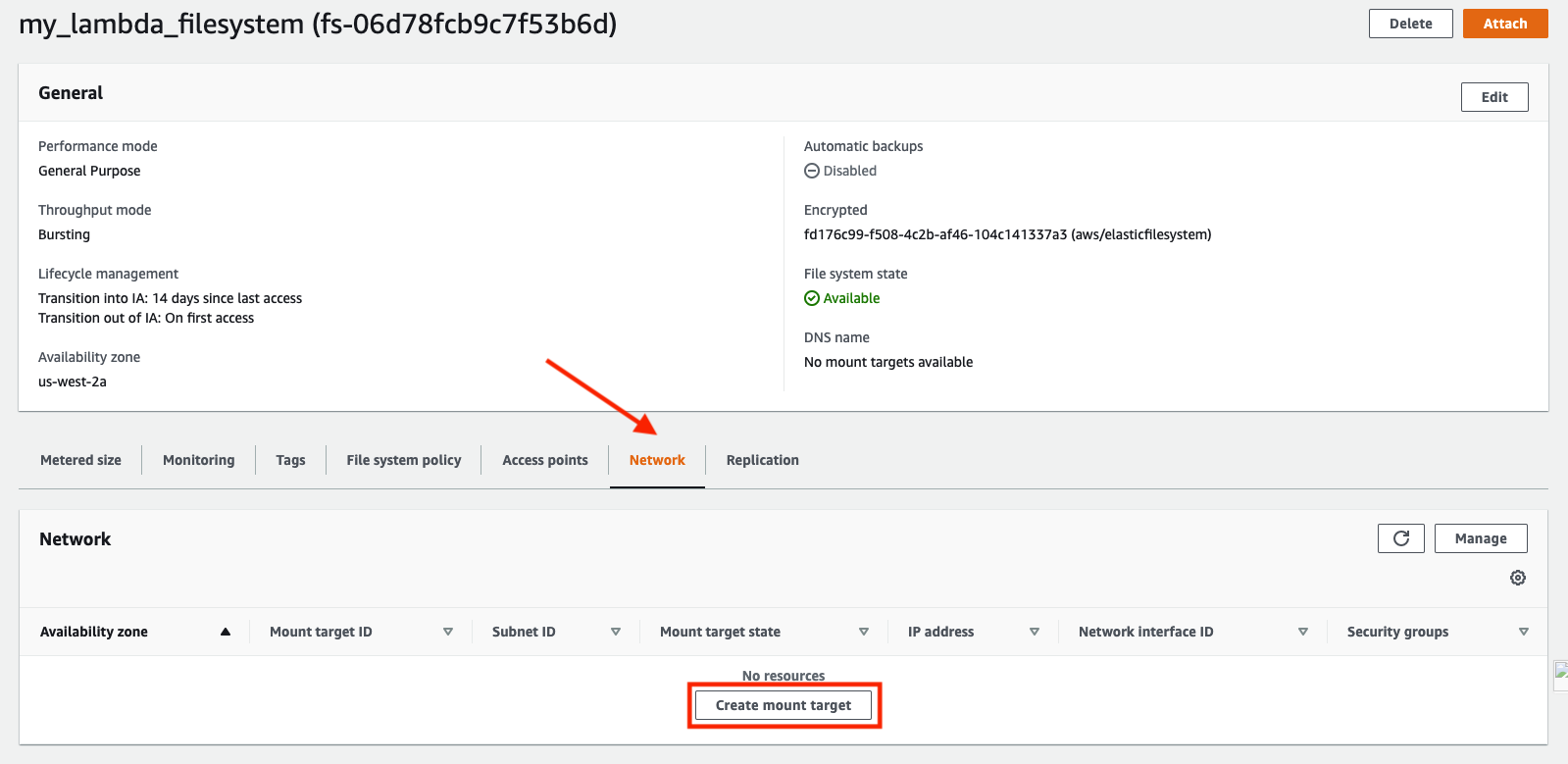
**Task 3: Create mount target for Amazon EFS File System**

In this task, you create a mount target for your Amazon EFS file system.

1. On the

my\_lambda\_filesystem

 page, choose the **Network** tab and then choose **Create mount target**.



 Should you encounter a warning that recommends enabling the EFS service-linked role using AWS IAM, you can safely ignore that as the EFS service-linked role is already configured as part of the lab setup.

1. Under the **Network** section, configure the following:

* **Virtual Private Cloud (VPC):** Select the

EFS-Lab-VPC

 from the drop-down list (Verify that it is the same value as *LabVPCId* from the list of parameters to the left of these instructions)

* **Mount targets:** Choose **Add mount target**
  + **Availability zone:** Select

us-west-2a

 from the drop-down list

* + **Subnet ID:** Select the

EFS-Lab-PublicSubnetA

 from the drop-down list (Verify that it is the same value as *LabSubnetId* from the list of parameters to the left of these instructions)

* + **IP address:** Leave it blank
  + **Security groups:** Select the default security group from the drop-down list (Verify that it is the same value as *LabSecurityGroupId* from the list of parameters to the left of these instructions)

1. Choose **Save**.
2. You should see the following message on the top of your screen:

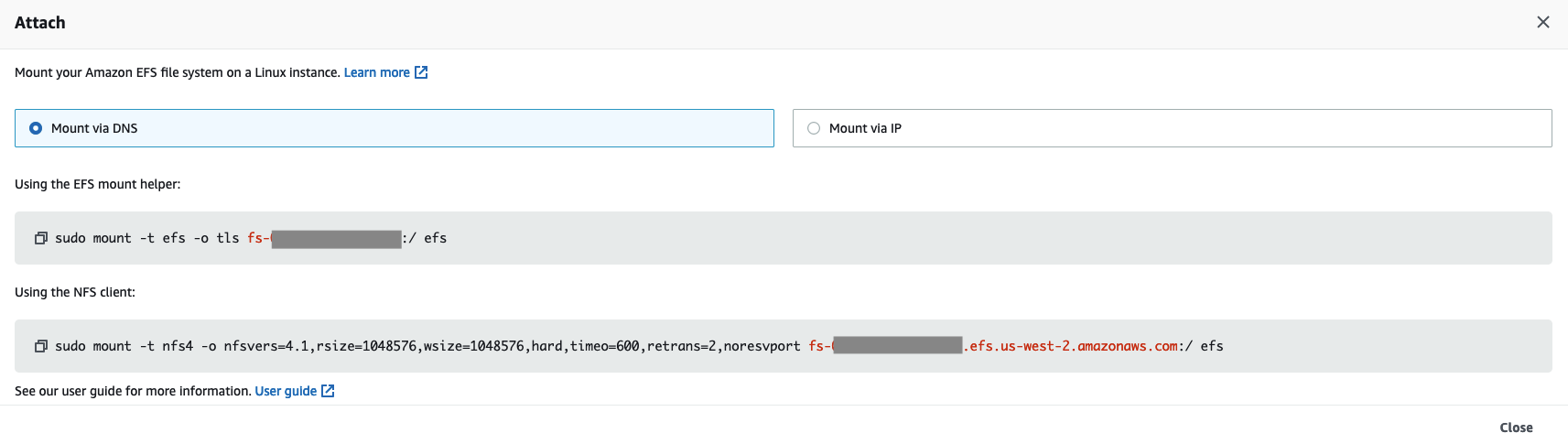
 Success!

Submitted all mount target changes successfully for file system (fs-xxxx)

After few minutes, the mount target state will change from *Creating* to *Available*. You might have to use the refresh option to reflect the status change.

1. From the top-right corner of the screen, choose **Attach**.

Here you see the commands for mounting the file system on to the Cloud9 instance as seen below:

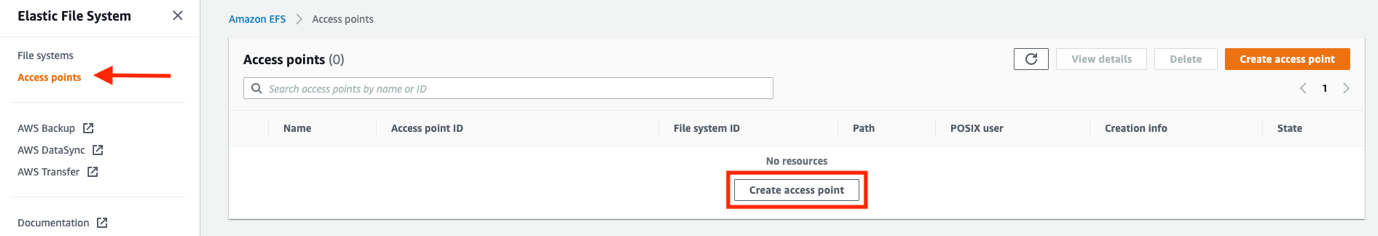


1. From the **Attach** pop-up window, make a note of the mount command under **Using the EFS mount helper**. You use it in the next task.
2. Choose **Close**.

**Task 4: Add File System Access Point**

In this task, you create a file system access point for your Amazon EFS file system.

1. From the left menu select **Access points**, and then choose **Create access point**.



1. On the **Create access point** screen, configure the following:

* **Details:**
  + **File system:** Select the existing file system id

my\_lambda\_filesystem

 from the dropdown list

* + **Name:** Enter

LambdaEFS

* + **Root directory path:** Enter

/prod

* **POSIX user:**
  + **User ID:** Enter

1000

* + **Group ID:** Enter

1000

* **Root directory creation permissions:**
  + **Owner user ID:** Enter

1000

* + **Owner group ID:** Enter

1000

* + **Access point permissions:** Enter

0755

 For more information, see [Working with Amazon EFS access points](https://docs.aws.amazon.com/efs/latest/ug/efs-access-points.html).

1. Choose **Create access point**.
2. You see the following message on the top of your screen if the access point creation is successful:

Success!

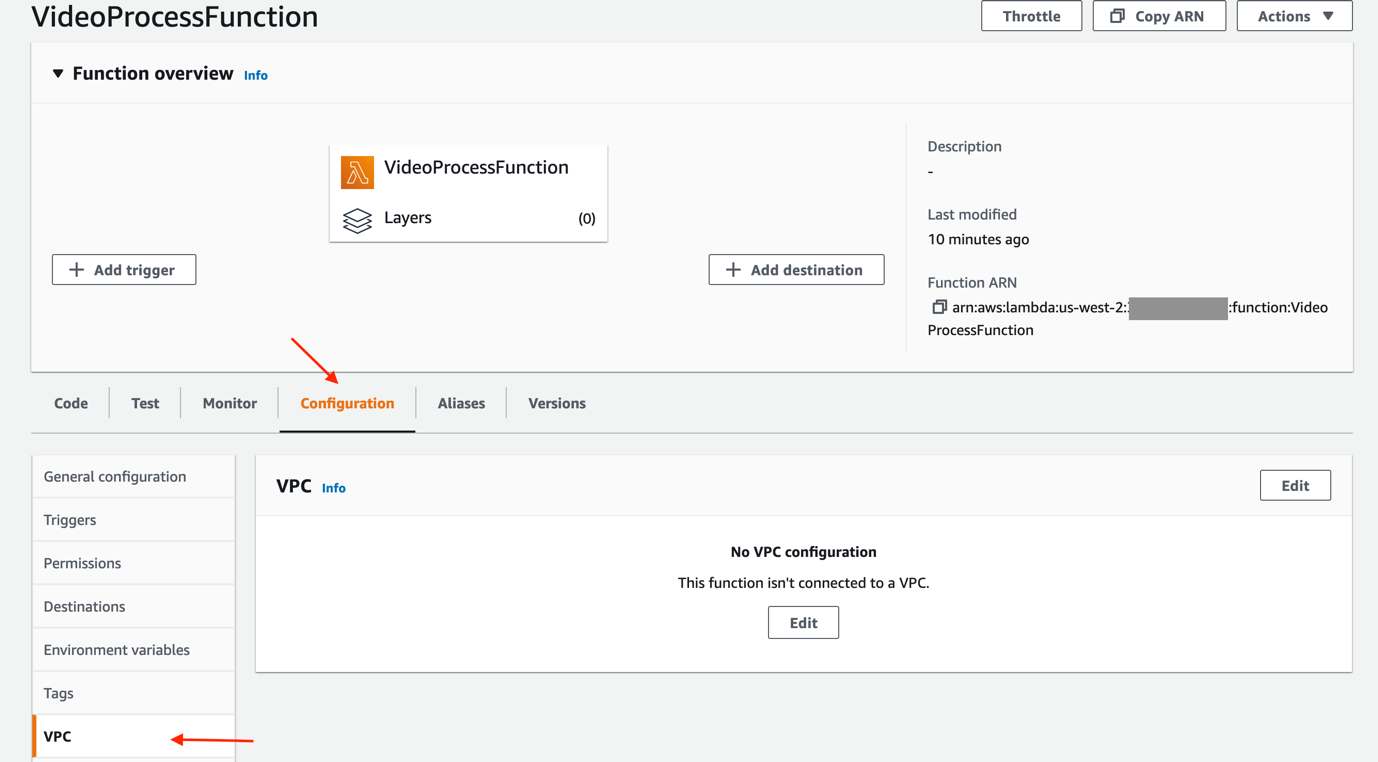
Access point (fsap-xxxxxx) is available

**Task 5: Configure an AWS Lambda function to use EFS**

AWS Lambda functions that makes use of EFS file systems must run from within a VPC. Further, the IAM execution role for Lambda function must provide access to the VPC and EFS. For the purpose of this lab, the IAM execution role for Lambda function is already configured with the required permissions.

In this task, you connect an existing AWS Lambda function to the EFS file system by modifying the required configuration parameters (VPC, File systems, Environment variables, and Layers).

1. At the top-left corner of the windows, on the **Services** menu, choose **Compute** and then choose **Lambda**.
2. On the AWS Lambda Dashboard, choose **Functions** from the left navigation pane.
3. On the **Functions** screen, select **VideoProcessFunction**, choose **Actions** and then choose **View details**.
4. Select the **Configuration** tab within the Lambda console, and then choose **VPC** from the left navigation pane.



1. Under **VPC**, choose **Edit** and configure the following:

* **VPC:** Select

EFS-Lab-VPC

 (Verify that it is the same value as *LabVPCId* from the list of parameters to the left of these instructions)

* **Subnets:** Select

EFS-Lab-PublicSubnetA

 (Verify that it is the same value as *LabSubnetId* from the list of parameters to the left of these instructions)

At this step you see the following warning:

 We recommend that you choose at least 2 subnets for Lambda to run your functions in high availability mode.

You can safely ignore the above warning for this lab.

* **Security groups:** Select

default VPC security group

 (Verify that it is the same value as *LabSecurityGroupId* from the list of parameters to the left of these instructions)

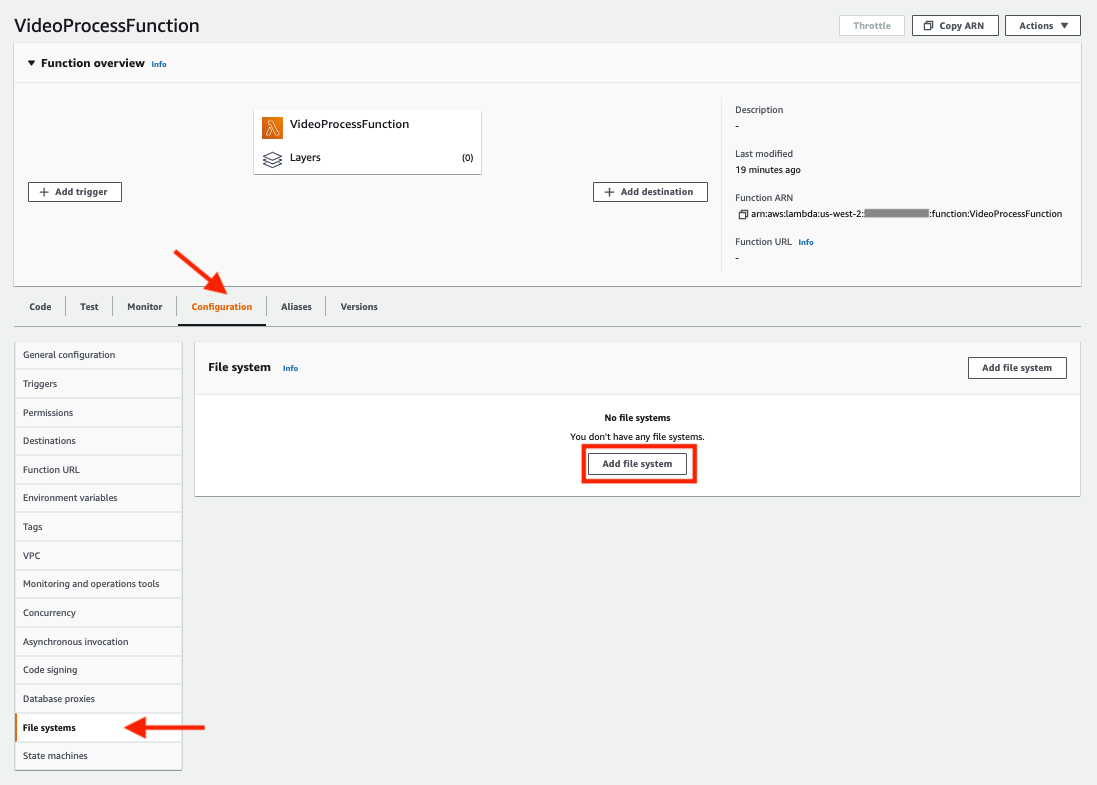
1. Choose **Save**.

 You might have to wait for few minutes for the Lambda function to update before proceeding with the next steps.

After successful update, you see the following message on top of the Lambda console:



1. From the **Configuration** tab within the Lambda console, choose **File systems** and then choose **Add file system**.



1. Under **File system**, configure the following:

* **EFS file system:** Select

my\_lambda\_filesystem

 from the dropdown list

* **Access point:** Select

LambdaEFS

 from the dropdown list

* **Local mount path:** Enter

/mnt/prod

 Local mount path mentioned above is the location where the file system is mounted on the Lambda function, starting with

/mnt/

.

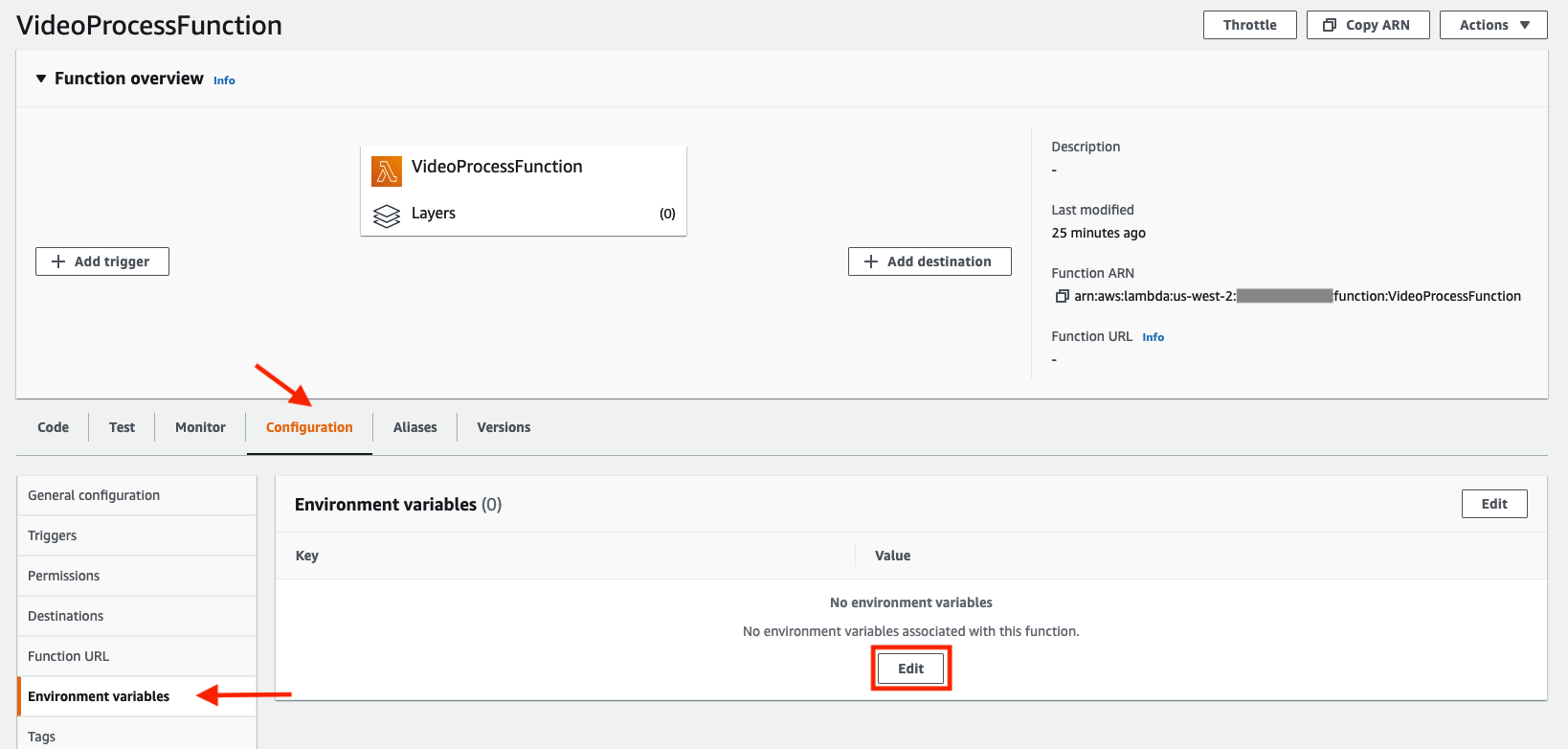
1. Choose **Save**.

 You might have to wait for few minutes for the Lambda function to update before proceeding with the next steps.

After successful update, you see the following message on top of the Lambda console:



1. From the **Configuration** tab within the Lambda console, choose **Environment variables** and then choose **Edit**.



1. Under **Environment variables**, choose **Add environment variable** and configure the following:

* **Key:** Enter

EFS\_PATH

**Value:** Enter

/mnt/prod

* **Key:** Enter

INPUT\_FILE

**Value:** Enter

Newvanlife.mp4

 The above two environment variables are referred within the **VideoProcessFuntion** Lambda function.

EFS\_PATH

 identifies the mount path where the file system is mounted on the Lambda function, and

INPUT\_FILE

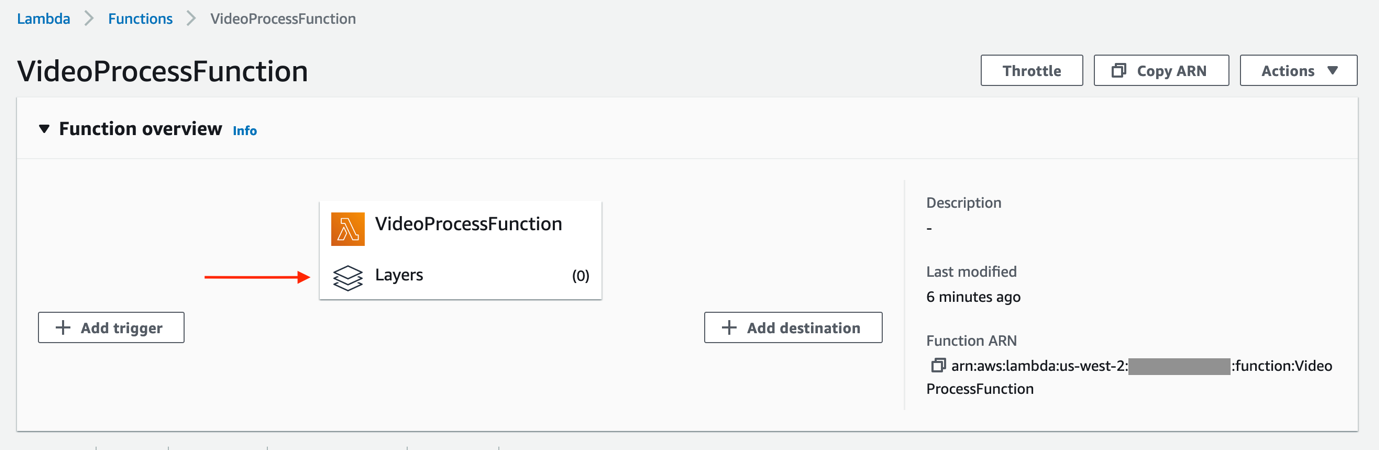
 identifies the sample video that will be used by Lambda function for processing.

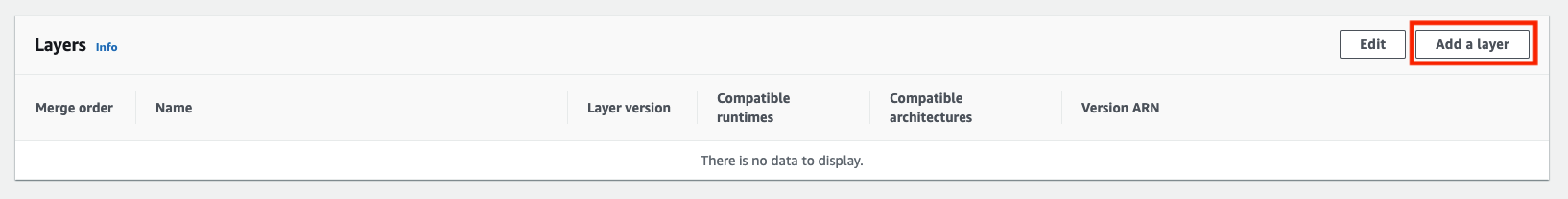
1. Choose **Save**.

After successful update, you see the following message on top of the Lambda console:



1. Under the **VideoProcessFunction** function overview, choose **Layers** or scroll down to the **Layers** section and choose **Add a layer**.





1. Under **Choose a layer** section, select  Specify an ARN, and paste the value of *LambdaLayerArn* from the list of parameters to the left of these instructions.
2. Choose **Verify** and then choose **Add**.

After successful update, you see the following message on top of the Lambda console:



Now that you have successfully configured the AWS Lambda function to use EFS, lets proceed to mount the file system on to the Cloud9 instance.

**Task 6: Install Amazon EFS client (amazon-efs-utils) and Mount EFS File Share in Cloud9**

The Amazon EFS client (

amazon-efs-utils

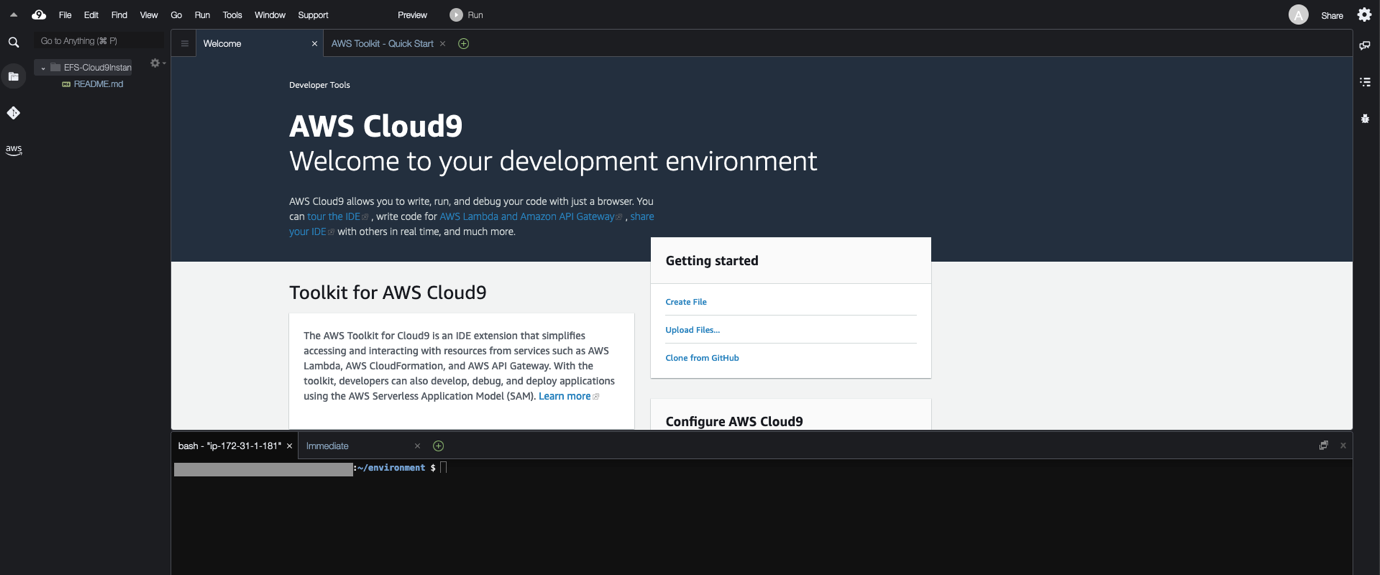
) is an open-source collection of Amazon EFS tools which includes a mount helper and tooling that makes it easier to perform encryption of data in transit for Amazon EFS file systems. A mount helper is a program that you use when you mount a specific type of file system. It is recommended to use the mount helper included with the Amazon EFS client to mount your Amazon EFS file systems. Using the Amazon EFS client simplifies mounting EFS file systems, and can provide improved file system performance.

You can mount EFS access points on Amazon EC2 instances or Cloud9 instances. This can be useful for browsing file systems contents and downloading files from other locations. The files that you write into the EFS file system are available to the Lambda functions that use the same EFS file system Similarly, the files written by Lambda function are available to the EC2 instance or Cloud9 instance.

For this task, you mount the EFS file system in AWS Cloud9 instance using the terminal window.

1. At the top-left corner of the windows, on the **Services** menu, choose **Developer Tools** and then choose **Cloud9**.
2. On the AWS Cloud9 environments screen, verify that the **Cloud9-EFSLab-IDE** is selected and choose **Open in Cloud9**.

A new tab opens to the AWS Cloud9 instance and you see the following screen:



1. Using the bash terminal available in Cloud9 IDE, run the following command to install the Amazon EFS utils package:

sudo yum install -y amazon-efs-utils

**Expected ouput:**

Loaded plugins: extras\_suggestions, langpacks, priorities, update-motd

247 packages excluded due to repository priority protections

Resolving Dependencies

--> Running transaction check

---> Package amazon-efs-utils.noarch 0:1.35.0-1.amzn2 will be installed

--> Processing Dependency: python3 for package: amazon-efs-utils-1.35.0-1.amzn2.noarch

--> Processing Dependency: stunnel5 for package: amazon-efs-utils-1.35.0-1.amzn2.noarch

--> Running transaction check

---> Package python3.x86\_64 0:3.7.16-1.amzn2.0.4 will be installed

--> Processing Dependency: python3-libs(x86-64) = 3.7.16-1.amzn2.0.4 for package: python3-3.7.16-1.amzn2.0.4.x86\_64

--> Processing Dependency: python3-pip for package: python3-3.7.16-1.amzn2.0.4.x86\_64

--> Processing Dependency: python3-setuptools for package: python3-3.7.16-1.amzn2.0.4.x86\_64

--> Processing Dependency: libpython3.7m.so.1.0()(64bit) for package: python3-3.7.16-1.amzn2.0.4.x86\_64

---> Package stunnel5.x86\_64 0:5.58-1.amzn2.0.1 will be installed

--> Running transaction check

---> Package python3-libs.x86\_64 0:3.7.16-1.amzn2.0.4 will be installed

---> Package python3-pip.noarch 0:20.2.2-1.amzn2.0.4 will be installed

---> Package python3-setuptools.noarch 0:49.1.3-1.amzn2.0.3 will be installed

--> Finished Dependency Resolution

Dependencies Resolved

====================================================================================================

Package Arch Version Repository Size

====================================================================================================

Installing:

amazon-efs-utils noarch 1.35.0-1.amzn2 amzn2-core 57 k

Installing for dependencies:

python3 x86\_64 3.7.16-1.amzn2.0.4 amzn2-core 72 k

python3-libs x86\_64 3.7.16-1.amzn2.0.4 amzn2-core 9.8 M

python3-pip noarch 20.2.2-1.amzn2.0.4 amzn2-core 2.0 M

python3-setuptools noarch 49.1.3-1.amzn2.0.3 amzn2-core 1.1 M

stunnel5 x86\_64 5.58-1.amzn2.0.1 amzn2-core 165 k

Transaction Summary

====================================================================================================

Install 1 Package (+5 Dependent packages)

Total download size: 13 M

Installed size: 55 M

Downloading packages:

(1/6): python3-3.7.16-1.amzn2.0.4.x86\_64.rpm | 72 kB 00:00:00

(2/6): amazon-efs-utils-1.35.0-1.amzn2.noarch.rpm | 57 kB 00:00:00

(3/6): python3-pip-20.2.2-1.amzn2.0.4.noarch.rpm | 2.0 MB 00:00:00

(4/6): python3-libs-3.7.16-1.amzn2.0.4.x86\_64.rpm | 9.8 MB 00:00:00

(5/6): python3-setuptools-49.1.3-1.amzn2.0.3.noarch.rpm | 1.1 MB 00:00:00

(6/6): stunnel5-5.58-1.amzn2.0.1.x86\_64.rpm | 165 kB 00:00:00

----------------------------------------------------------------------------------------------------

Total 43 MB/s | 13 MB 00:00:00

Running transaction check

Running transaction test

Transaction test succeeded

Running transaction

Installing : python3-libs-3.7.16-1.amzn2.0.4.x86\_64 1/6

Installing : python3-setuptools-49.1.3-1.amzn2.0.3.noarch 2/6

Installing : python3-3.7.16-1.amzn2.0.4.x86\_64 3/6

Installing : python3-pip-20.2.2-1.amzn2.0.4.noarch 4/6

Installing : stunnel5-5.58-1.amzn2.0.1.x86\_64 5/6

Installing : amazon-efs-utils-1.35.0-1.amzn2.noarch 6/6

Verifying : amazon-efs-utils-1.35.0-1.amzn2.noarch 1/6

Verifying : stunnel5-5.58-1.amzn2.0.1.x86\_64 2/6

Verifying : python3-pip-20.2.2-1.amzn2.0.4.noarch 3/6

Verifying : python3-libs-3.7.16-1.amzn2.0.4.x86\_64 4/6

Verifying : python3-setuptools-49.1.3-1.amzn2.0.3.noarch 5/6

Verifying : python3-3.7.16-1.amzn2.0.4.x86\_64 6/6

Installed:

amazon-efs-utils.noarch 0:1.35.0-1.amzn2

Dependency Installed:

python3.x86\_64 0:3.7.16-1.amzn2.0.4 python3-libs.x86\_64 0:3.7.16-1.amzn2.0.4

python3-pip.noarch 0:20.2.2-1.amzn2.0.4 python3-setuptools.noarch 0:49.1.3-1.amzn2.0.3

stunnel5.x86\_64 0:5.58-1.amzn2.0.1

Complete!

For additional information, see [Using the amazon-efs-utils Tools](https://docs.aws.amazon.com/efs/latest/ug/using-amazon-efs-utils.html).

1. Run the following command to create a mount point directory:

sudo mkdir efs

1. Run the following command to mount the EFS file system. This is the same command that you noted as part of Step 16. Replace **fs-xxxxx** with the actual EFS file system id that was created earlier.

sudo mount -t efs -o tls fs-xxxxx:/ efs

**For example:**

sudo mount -t efs -o tls fs-02459383a89eeabed:/ efs

 This command times out after approximately five minutes because the security group is not configured to allow NFS traffic between the Cloud9 host and the EFS file system. Type Ctrl+C to interrupt the command, rather than waiting for it to time out. You add the appropriate security group next.

1. Return to the AWS management console.
2. At the top-left corner of the windows, on the **Services** menu, choose **Compute** and then choose **EC2**.
3. On the EC2 Dashboard, choose **Instances (running)**.
4. On the EC2 instances screen, select the **aws-cloud9-EFS-Cloud9Instance-xxx**, choose **Actions**, **Security** and then choose **Change security groups**.
5. Under the **Associated security groups** section, search and select the default security group (verify that it is the same value as *LabSecurityGroupId* from the list of parameters to the left of these instructions) and choose **Add security group**.
6. Choose **Save**.
7. Return to the Cloud9 terminal window and re-run the following command to mount the EFS file system. This is the same command that you noted as part of Step 16. Replace **fs-xxxxx** with the actual EFS file system id that was created earlier.

sudo mount -t efs -o tls fs-xxxxx:/ efs

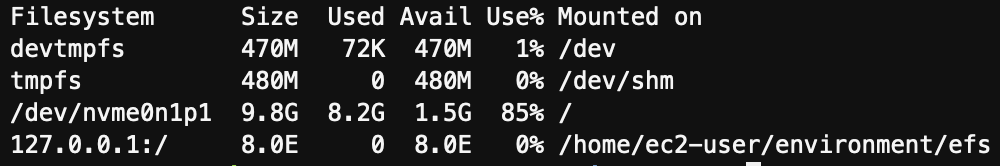
**For example:**

sudo mount -t efs -o tls fs-02459383a89eeabed:/ efs

1. Enter the following command to verify the file system has been mounted correctly:

df -h

**Expected ouput:**



1. Run the following command to download a sample video file (Newvanlife.mp4) that will be used for testing the serverless video processing workflow with AWS Lambda and Amazon EFS:

wget https://us-west-2-aws-training.s3-us-west-2.amazonaws.com/courses/SPL-DD-300-STEFSD/v1.0.6.prod-b403dd7e/scripts/Newvanlife.mp4

**Expected ouput:**

--2022-02-14 17:34:46-- https://us-west-2-aws-training.s3-us-west-2.amazonaws.com/courses/SPL-DD-300-STEFSD/v1.0.0.dev-da751db6/scripts/Newvanlife.mp4

Resolving us-west-2-aws-training.s3-us-west-2.amazonaws.com (us-west-2-aws-training.s3-us-west-2.amazonaws.com)... 52.218.184.105

Connecting to us-west-2-aws-training.s3-us-west-2.amazonaws.com (us-west-2-aws-training.s3-us-west-2.amazonaws.com)|52.218.184.105|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 2121806 (2.0M) [video/mp4]

Saving to: ‘Newvanlife.mp4’

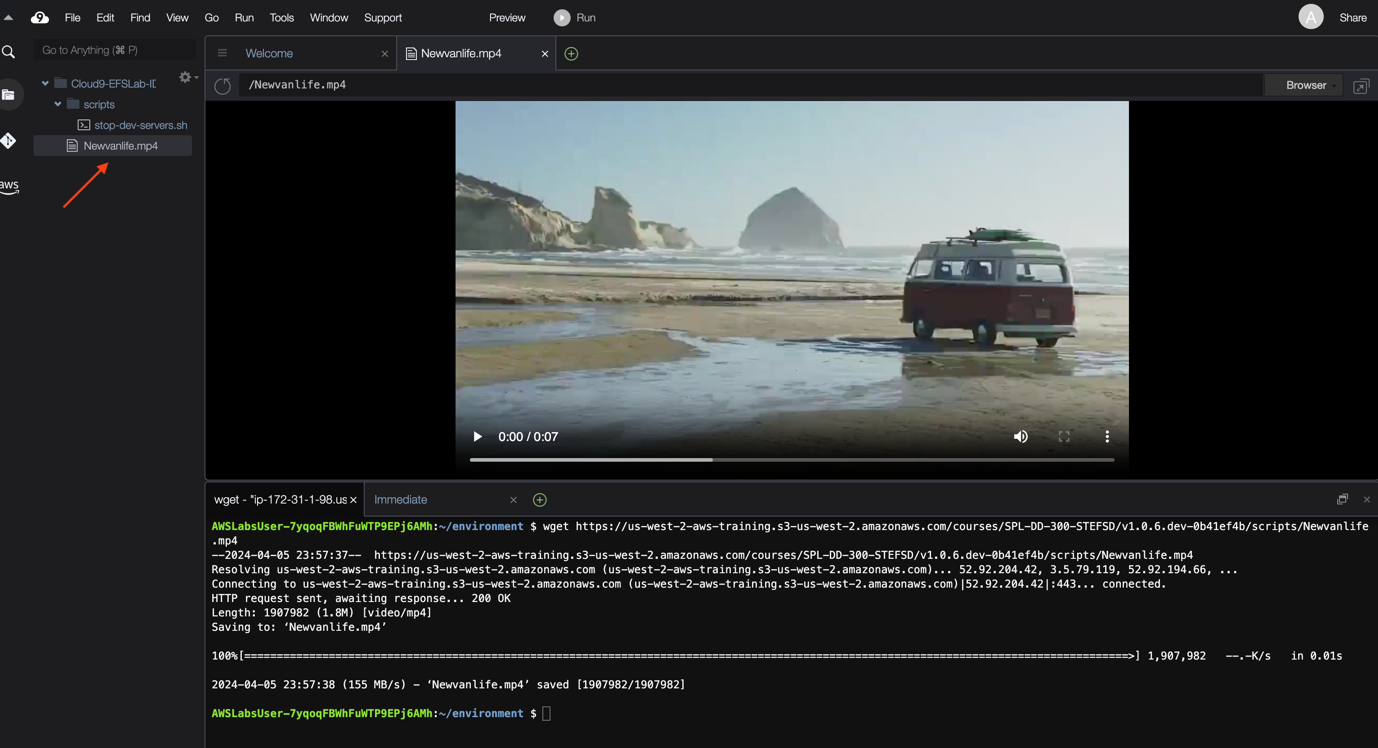
Newvanlife.mp4 100%[===============================================================================================================================>] 2.02M --.-KB/s in 0.06s

2022-02-14 17:34:46 (36.5 MB/s) - ‘Newvanlife.mp4’ saved [2121806/2121806]

You can also play the sample video by selecting

Newvanlife.mp4

 file via the left navigation pane on the Cloud9 console:



1. Run the following command to confirm the sample video is successfully downloaded in the current directory:

ls

**Expected ouput:**

efs README.md Newvanlife.mp4

1. Run the following command to go the efs/ directory and list its contents:

cd efs/ && ls

1. After running the above command, you won’t see any contents inside the

efs/

 directory.

Now, in order to generate the file system access point path

/prod

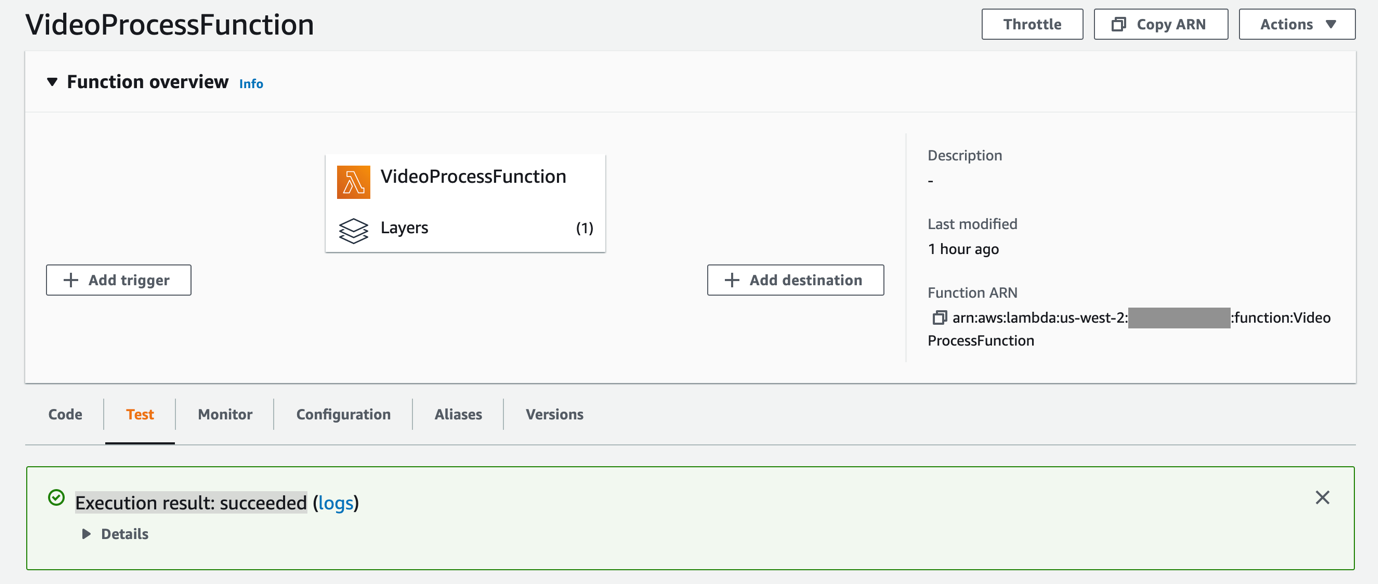
 inside the

efs/

 directory, you need to invoke the **VideoProcessFunction** Lambda function.

1. Return to the AWS Lambda console, and on the **VideoProcessFunction** Lambda function screen, choose the **Test** tab, and run a test event by choosing the **Test** option.

After the Lambda function ran successful, you see the following:



 If you expand the **Details** section above to view the Lambda logs, you will notice the following error:

START RequestId: 4d7053a2-8693-494f-97f0-c8cfb8f0030c Version: $LATEST

2022-02-21T15:49:29.152Z 4d7053a2-8693-494f-97f0-c8cfb8f0030c INFO /opt/bin/ffmpeg -loglevel error -i /mnt/prod/Newvanlife.mp4 -s 240x135 -vf fps=1 /mnt/prod/%d.jpg

2022-02-21T15:49:29.295Z 4d7053a2-8693-494f-97f0-c8cfb8f0030c INFO Finished: 1

2022-02-21T15:49:29.298Z 4d7053a2-8693-494f-97f0-c8cfb8f0030c INFO Error: Error: Command failed: /opt/bin/ffmpeg -loglevel error -i /mnt/prod/Newvanlife.mp4 -s 240x135 -vf fps=1 /mnt/prod/%d.jpg

/mnt/prod/Newvanlife.mp4: No such file or directory

at ChildProcess.exithandler (child\_process.js:308:12)

at ChildProcess.emit (events.js:314:20)

at maybeClose (internal/child\_process.js:1022:16)

at Process.ChildProcess.\_handle.onexit (internal/child\_process.js:287:5) {

killed: false,

code: 1,

signal: null,

cmd: '/opt/bin/ffmpeg -loglevel error -i /mnt/prod/Newvanlife.mp4 -s 240x135 -vf fps=1 /mnt/prod/%d.jpg'

}

2022-02-21T15:49:29.298Z 4d7053a2-8693-494f-97f0-c8cfb8f0030c INFO stdout:

2022-02-21T15:49:29.298Z 4d7053a2-8693-494f-97f0-c8cfb8f0030c INFO stderr: /mnt/prod/Newvanlife.mp4: No such file or directory

END RequestId: 4d7053a2-8693-494f-97f0-c8cfb8f0030c

REPORT RequestId: 4d7053a2-8693-494f-97f0-c8cfb8f0030c Duration: 149.18 ms Billed Duration: 150 ms Memory Size: 2048 MB Max Memory Used: 75 MB Init Duration: 150.78 ms

The above error is expected and you can safely ignore it for this step.

1. Return to the Cloud9 terminal and run the following command from inside the

efs/

 directory:

ls

**Expected ouput:**

prod

As you can see from the above output, the

prod

 directory is now created. You will notice that the directory name is same as the root directory path

/prod

 that was used while creating file system access point as part of Task 3.

1. Run the following command to go back to the

~/environment/

 directory:

cd ../ && pwd

1. Run the following command inside the

~/environment/

 directory to move the

Newvanlife.mp4

 file to the

efs/prod/

 directory:

sudo mv Newvanlife.mp4 efs/prod/

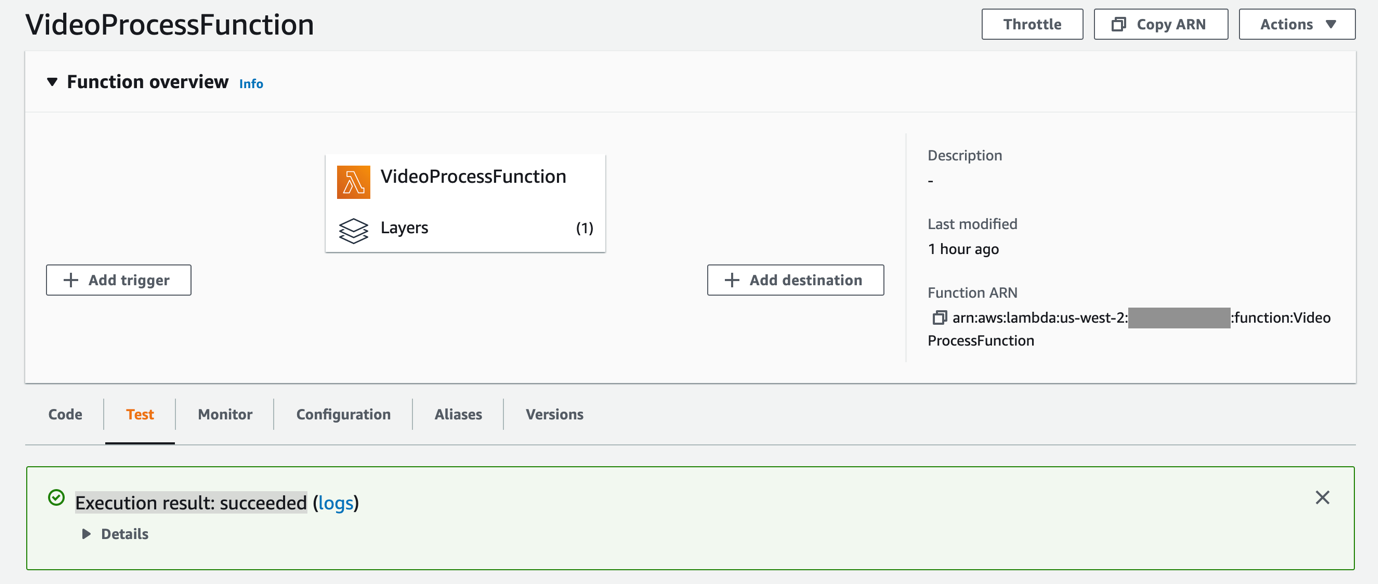
**Task 7: Test a serverless video processing workflow with AWS Lambda and Amazon EFS.**

This task use EFS to process the sample Newvanlife.mp4 video and create screenshots for each second of the recording. The Lambda function uses the FFmpeg Linux package to process the video.

After copying the Newvanlife.mp4 file to the EFS file location, you invoke the **VideoProcessFunction** Lambda function again to create a series of .jpg frames.

1. Return to the AWS Lambda console and close the logs section from the previous Lambda invocation.
2. On the **VideoProcessFunction** Lambda function screen, choose the **Test** tab, and run another test event by choosing the **Test** option.

After the Lambda function ran successful, you see the following:



If you expand the **Details** section above to view the Lambda logs, you should see the following output:

START RequestId: b70deddd-7efb-4c67-b4de-7e8d20832430 Version: $LATEST

2022-02-21T16:07:45.644Z b70deddd-7efb-4c67-b4de-7e8d20832430 INFO /opt/bin/ffmpeg -loglevel error -i /mnt/prod/Newvanlife.mp4 -s 240x135 -vf fps=1 /mnt/prod/%d.jpg

2022-02-21T16:07:52.167Z b70deddd-7efb-4c67-b4de-7e8d20832430 INFO Finished: 0

2022-02-21T16:07:52.168Z b70deddd-7efb-4c67-b4de-7e8d20832430 INFO stdout:

2022-02-21T16:07:52.168Z b70deddd-7efb-4c67-b4de-7e8d20832430 INFO stderr:

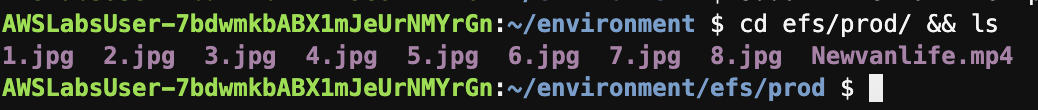
END RequestId: b70deddd-7efb-4c67-b4de-7e8d20832430

REPORT RequestId: b70deddd-7efb-4c67-b4de-7e8d20832430 Duration: 6527.05 ms Billed Duration: 6528 ms Memory Size: 2048 MB Max Memory Used: 115 MB Init Duration: 173.00 ms

1. Return to the Cloud9 terminal and run the following command:

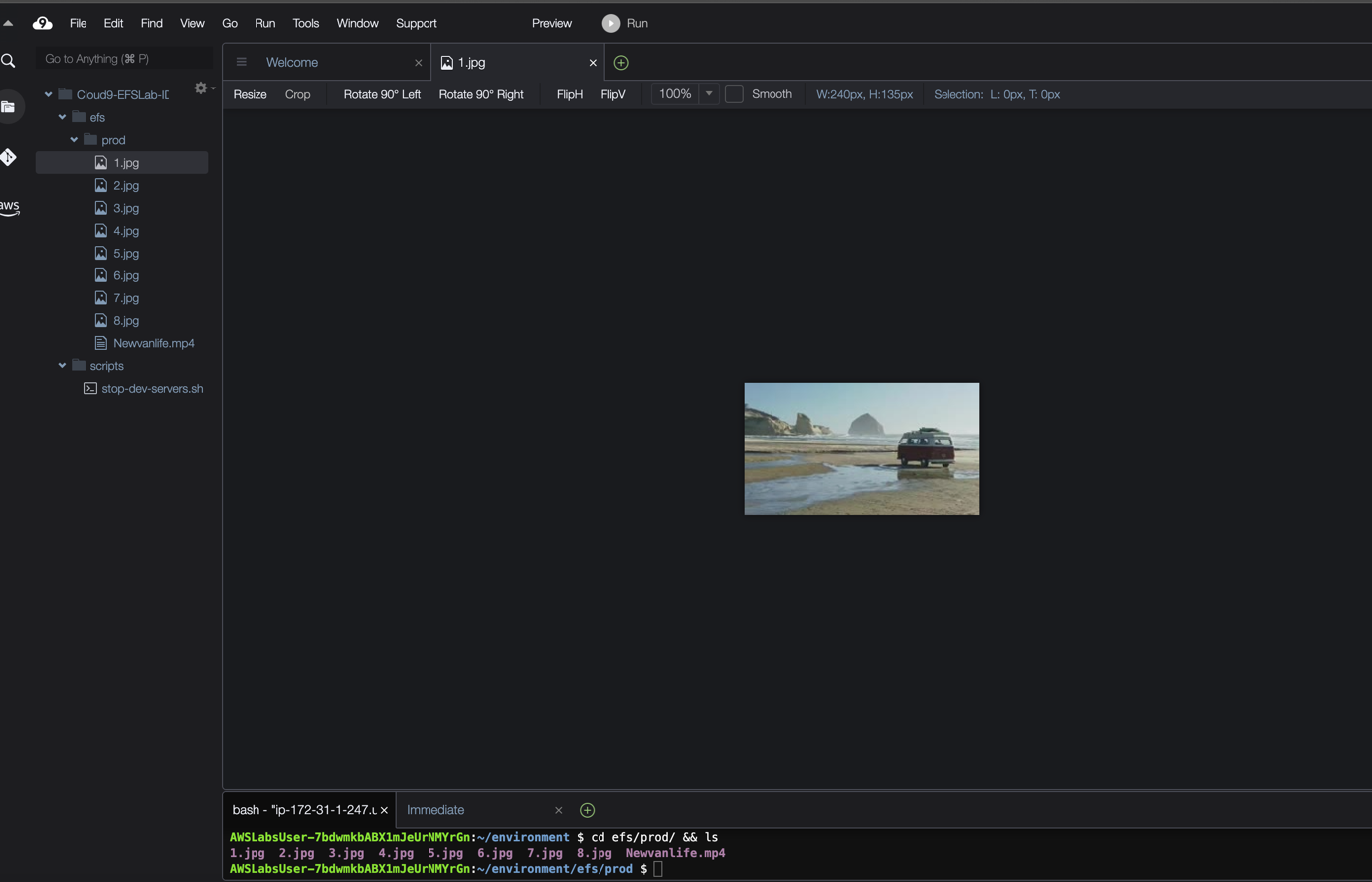
cd efs/prod/ && ls

**Expected ouput:**



As seen from the above output, the Lambda function processed a 51-second MP4 video and created screenshots for each second of the recording. These screenshots were then written back to the EFS file system.

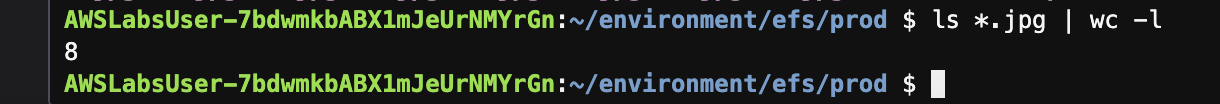
You can view these screenshots within the Cloud9 console as seen below:



1. You can also run the following command to confirm that the video was processed and was converted into 51 screenshots:

ls \*.jpg | wc -l

**Expected ouput:**



**Task 8: Challenge Task (Optional)**

 Challenge yourself! This challenge task reinforces your knowledge of the lab and gives you more practice if you have some time left.

* Create a second file system access point (e.g.

/test

) similar to step 12 in Task 4.

* Modify the Lambda function’s configuration to use this new access point mount (e.g.

/mnt/test

) similar to step 23 in Task 5.

* Update the Lambda function’s environment variable

EFS\_PATH

 with the new mount path (e.g.

EFS\_PATH=/mnt/test

) similar to step 26 in Task 5.

* Move the sample video to the

efs/test/

 directory. (e.g. command

sudo mv Newvanlife.mp4 efs/test/

) similar to step 51 in Task 6.

* Re-test the video processing functionality (by following similar steps in Task 7) and review the results.

**Conclusion**

 Congratulations! You now have successfully:

* Created and configured an Amazon EFS file system.
* Added file system access point, and configured client permissions.
* Configured an AWS Lambda function to use an EFS file system for persistent storage.
* Installed EFS utils client for NFS file access, and mounted the file system to an AWS Cloud9 instance.
* Tested a serverless video processing workflow with AWS Lambda and Amazon EFS.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional Resources**

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).