

Step 1 :- Create EBS Volume in EC2 instance

The screenshot shows the AWS Management Console interface for the Elastic Block Store (EBS) service. On the left, there is a navigation sidebar with various options like Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, and more. The main area displays a table titled "Volumes (1/3) Info" with three entries:

Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created
	vol-0e4f39cfed698391	gp3	8 GiB	3000	125	snap-05c5a3c...	2025/09
	vol-0852ec69a4331fbf6	gp3	8 GiB	3000	125	snap-05bbbeb8...	2025/10
	vol-02fe99c007fbf9cf3	gp3	8 GiB	3000	125	snap-0e78b08...	2025/09

Below the table, a modal window is open for the selected volume, showing its details:

Details	Status checks	Monitoring	Tags
Volume ID vol-02fe99c007fbf9cf3	Size 8 GiB	Type gp3	Status check Okay
AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations.	Volume state In-use	IOPS 3000	Throughput 125

Key Points:

- Replace the placeholder '`'your-volume-id-here'`' with the actual EBS volume ID to snapshot.
- Ensure your Lambda's IAM role has permissions like `ec2:CreateSnapshot`, `ec2:DeleteSnapshot`, `ec2:DescribeSnapshots`.
- You can schedule this Lambda to run periodically (e.g., weekly) using EventBridge (CloudWatch Events).
- The function creates a snapshot and deletes any snapshots older than 30 days owned by your AWS account for that volume.
- Logs include snapshot creation and deletion details for easy auditing.

Step 2 :- Create lambda function

Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Container image
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.
EBSSnapshotAndCleanup

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.13

Architecture Info
Choose the instruction set architecture you want for your function code.
arm64

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lambda > Functions > Create function

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).
 Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
service-role/Lambda_function-role-9i5yg9oh

Additional configurations
Use additional configurations to set up networking, security, and governance for your function. These settings help secure and customize your Lambda function deployment.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lambda > Functions > EBSSnapshotAndCleanup

Configuration

Code Test Monitor Configuration Aliases Versions

General configuration Info

General configuration

Description
-

Memory
128 MB

Ephemeral storage
512 MB

Edit

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

RDS databases

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

```
import boto3
import datetime

ec2 = boto3.client('ec2')
RETENTION_DAYS = 30 # Snapshots older than this will be deleted

def lambda_handler(event, context):
    volume_id = event.get('volume_id') # Get volume_id from event
    if not volume_id:
        volume_id = 'vol-02fe99c007fbf9cf3' # Replace with your actual
volume ID
        print(f"No volume_id in event; using fallback: {volume_id}")

    # Calculate cutoff date for deletion
    delete_date = datetime.datetime.now() -
datetime.timedelta(days=RETENTION_DAYS)

    # List all snapshots for the volume owned by self
    snapshots = ec2.describe_snapshots(
        Filters=[{'Name': 'volume-id', 'Values': [volume_id]}],
        OwnerIds=['self']
    )['Snapshots']

    # Delete snapshots older than retention period first
    for snap in snapshots:
        start_time = snap['StartTime'].replace(tzinfo=None) # naive
datetime
        if start_time < delete_date:
            snap_id = snap['SnapshotId']
            ec2.delete_snapshot(SnapshotId=snap_id)
            print(f"Deleted snapshot {snap_id} created on {start_time}")

    # Now create a new snapshot for the volume
    snapshot = ec2.create_snapshot(
        VolumeId=volume_id,
        Description=f"Automated snapshot of volume {volume_id} on
{datetime.datetime.now()}"
    )
    print(f"Created snapshot {snapshot['SnapshotId']} for volume
{volume_id}")
```

```

return {
    'created_snapshot': snapshot['SnapshotId'],
    'message': f"Snapshots older than {RETENTION_DAYS} days deleted"
for volume {volume_id}"
}

```

The screenshot shows the AWS Lambda Function Editor interface. The top navigation bar includes 'Lambda > Functions > EBSSnapshotAndCleanup'. Below the navigation is a tab bar with 'Code' (selected), 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The main area is titled 'Code source' with tabs for 'Info' and 'Code'. The code editor displays `lambda_function.py` with the following content:

```

EXPLORER
EBSSNAPSHOTANDCLEANUP
lambda_function.py

def lambda_handler(event, context):
    volume_id = event.get('volume_id') # Get volume_id from event
    if not volume_id:
        volume_id = 'vol-02fe99c007fbf9cf3' # Replace with your actual volume ID
        print(f"No volume_id in event; using fallback: {volume_id}")

# Calculate cutoff date for deletion

```

The 'PROBLEMS' tab shows no issues. At the bottom, there are buttons for 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'. The status bar at the bottom right indicates '© 2025, Amazon Web Services, Inc. or its affiliates.' and other links.

This screenshot shows the same Lambda function editor after deployment. The left sidebar shows the deployed function 'EBSSNAPSHOTANDCLEANUP' with 'lambda_function.py' selected. The 'DEPLOY' section has 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)' buttons. The main code editor window now shows the deployed code:

```

lambda_function.py
import boto3
import datetime
ec2 = boto3.client('ec2')
RETENTION_DAYS = 30 # Snapshots older than this will be deleted
def lambda_handler(event, context):
    volume_id = event.get('volume_id') # Get volume_id from event

```

The status bar at the bottom right shows 'Status: Succeeded'. The 'TEST EVENTS [SELECTED: EBS]' section shows a single event named 'ebs'. The 'Response' pane displays the JSON output of the test:

```

{
    "created_snapshot": "snap-0289cc30ef50cac1",
    "message": "Snapshots older than 30 days deleted for volume vol-02fe99c007fbf9cf3"
}

```

The 'Function Logs' pane shows the log output starting with 'START RequestId: be6e7859-8ede-42d6-aa33-861ce77a0c16 Version: \$LATEST'.

The screenshot shows the AWS Lambda Function Editor. The left sidebar lists the function's structure: EBSSnapshotAndCleanup (with lambda_function.py selected), DEPLOY, and TEST EVENTS [SELECTED: EBS]. Under TEST EVENTS, there are options to Create new test event, Private saved events (with ebs selected), and a link to CloudShell.

The main area displays the code for lambda_function.py:

```
lambda_function.py
1 import boto3
2 import datetime
3
4 ec2 = boto3.client('ec2')
5 RETENTION_DAYS = 30 # Snapshots older than this will be deleted
6
7 def lambda_handler(event, context):
8     volume_id = event.get('volume_id') # Get volume id from event
```

The right side shows the Execution Results tab with the following log output:

```
Function Logs:
START RequestId: be6e7859-8ede-42d6-aa33-861ce77a0c16 Version: $LATEST
No volume_id in event; using fallback: vol-02fe99c007fbf9cf3
Created snapshot snap-0289cce30ef58cac1 for volume vol-02fe99c007fbf9cf3
END RequestId: be6e7859-8ede-42d6-aa33-861ce77a0c16
REPORT RequestId: be6e7859-8ede-42d6-aa33-861ce77a0c16 Duration: 695.45 ms Billed Duration: 1148 ms Memory Size: 128 MB Max Memory Used: 100 MB Init Duration: 452.06 ms
Request ID: be6e7859-8ede-42d6-aa33-861ce77a0c16
```

At the bottom, the status bar shows: Ln 8 Col 14 Spaces: 4 UTF-8 LF Python Lambda Layout: US.

The screenshot shows the AWS EBS Volumes page. The left sidebar includes sections for Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager). The main content displays a table of volumes:

ID	Type	Size	IOPS	Throughput	Snapshot ID	Created	Availability
f39cfded698391	gp3	8 GiB	3000	125	snap-05c5a3c...	2025/09/16 11:37 GMT+5:...	eu-west-2b
2ec69a4331fbf6	gp3	8 GiB	3000	125	snap-05bbeb8...	2025/10/07 15:12 GMT+5:...	eu-west-2c
e99c007fbf9cf3	gp3	8 GiB	3000	125	snap-0e78b08...	2025/09/25 11:17 GMT+5:...	eu-west-2a

Below the table, a detailed view for Volume ID: vol-02fe99c007fbf9cf3 is shown with tabs for Details, Status checks, Monitoring, and Tags. The Details tab displays the following information:

Volume ID vol-02fe99c007fbf9cf3	Size 8 GiB	Type gp3	Status check Okay
AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more	Volume state In-use	IOPS 3000	Throughput 125