



STRENGTHENING AWS DATA SECURITY: A COMPLETE FRAMEWORK FOR EBS DATA PROTECTION AND ENCRYPTION



FEBRUARY 25, 2025
SHAISTHA KHANUM

Table of Contents

Introduction:	3
Amazon Elastic Block Storage.....	3
Encrypting EBS volumes.....	3
Understanding AWS KMS and EBS Encryption	4
Encrypt Existing Unencrypted EBS Volumes and Snapshots.....	4
Why EBS volumes/snapshots should be encrypted (and what that means).....	4
Flow Chart.....	4
Identify unencrypted volumes:.....	5
Create a snapshot of the unencrypted volume:	5
Copy the snapshot to an encrypted form:.....	6
Create a new encrypted volume from the encrypted snapshot:	8
Detach the unencrypted volume from the EC2 instance:.....	10
Attach the new encrypted volume to the EC2 instance:.....	11
Verify data integrity and encryption:	13
Clean up resources:	14
Automating AWS EBS Volume Encryption using Terraform, Bash, and Checkov.....	15
Tools Used	15
Terraform	15
Checkov.....	15
Flow Chart:	16
Automating EBS Creation using Terraform	16
AWS Configuration.....	16
Write Terraform Code.....	17
Execute Terraform Commands.....	20
Verify EBS Creation	21
Encrypting Unencrypted Volumes Using Bash Script	23
Verify Encrypted Volumes.....	24
Scanning Terraform Code Using Checkov.....	26
Clean Up Resources.....	27
Performance Impact of EBS Encryption.....	27
Compliance and Security Best Practices.....	27
Incident Response and Data Recovery	28
Additional Tips.....	28
Real-World Use Cases and Scenarios	29
Conclusion	29

References:	30
Appendices:	31

Introduction:

In the era of cloud computing, data security has become a cornerstone of resilient and compliant infrastructure. Amazon Elastic Block Store (EBS), a vital component of Amazon Web Services (AWS), offers high-performance block storage for EC2 instances, supporting a wide range of workloads—from critical databases to large-scale analytics. However, the sensitive nature of data stored on EBS volumes demands robust protection measures to safeguard against unauthorized access, data breaches, and compliance violations.

This report presents a comprehensive framework for strengthening AWS data security by focusing on EBS data protection and encryption. It explores native AWS encryption mechanisms, highlights the role of AWS Key Management Service (KMS) in securing data at rest, and outlines best practices for managing encryption keys. The report also provides a step-by-step guide for encrypting existing unencrypted EBS volumes, leveraging automation with Terraform and Bash scripts, and implementing security compliance checks using tools like Checkov.

Furthermore, it delves into the performance impact of EBS encryption, compliance considerations, and strategies for effective data recovery and incident response. By adopting this end-to-end approach, organizations can ensure that their AWS environments not only meet regulatory standards but also uphold the highest levels of data confidentiality, integrity, and availability.

Amazon Elastic Block Storage

Amazon Elastic Block Storage (EBS) is a block-level storage service designed for use with Amazon EC2 instances. Acting like a virtual SSD, EBS provides scalable and high-performance storage for various applications, including databases, big data analytics engines, and web applications. Given the sensitive nature of the data often stored on EBS volumes, it's critical to implement robust data protection strategies, including encryption.

Encrypting EBS volumes

By default, EBS volumes are not encrypted when created unless configured otherwise in the EC2 settings. Once an EBS volume is created, its encryption status cannot be altered. To change the encryption state, data must be migrated to a new volume with the desired encryption settings.

Key Points:

- Snapshots created from encrypted volumes are automatically encrypted.
- EBS encryption is transparent to end-users and applications.
- While EBS encryption protects against unauthorized access to physical disks, it doesn't safeguard data from threats within the EC2 instance.

Understanding AWS KMS and EBS Encryption

AWS Key Management Service (KMS) plays a central role in EBS encryption. It manages encryption keys and integrates with EBS for seamless encryption processes. Users can choose between AWS-managed keys or customer-managed keys (CMKs) for greater control.

- **AWS KMS Features:**

- Centralized key management for EBS and other AWS services.
- Fine-grained access control using AWS IAM policies.
- Automatic key rotation for enhanced security.
- Integration with AWS CloudTrail for auditing key usage and changes.
- Support for cross-account access and multi-region key replication.

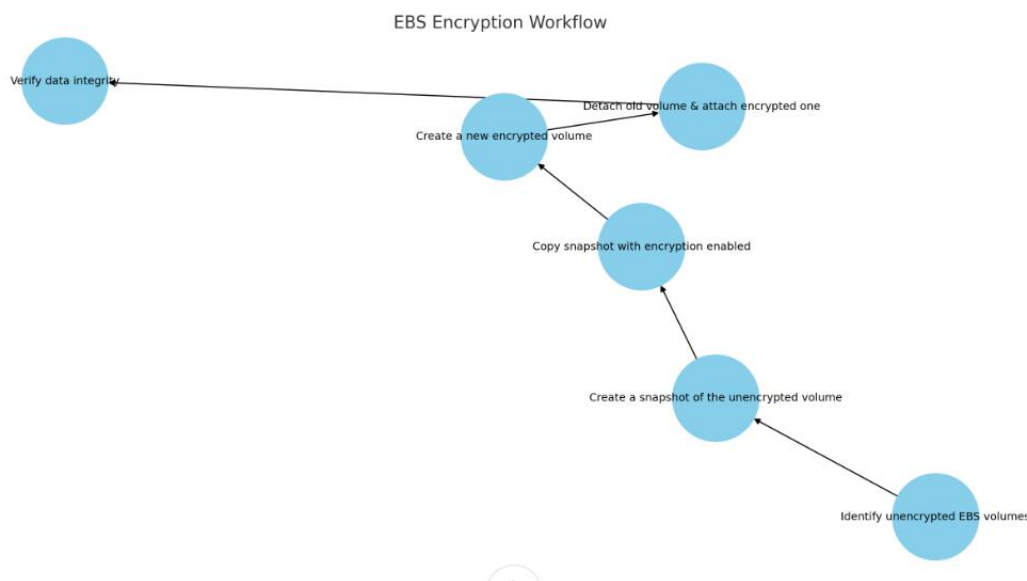
Encrypt Existing Unencrypted EBS Volumes and Snapshots

Learn how to manage unencrypted EBS volumes and snapshots attached to existing EC2 instances by following the correct process to encrypt them effectively.

Why EBS volumes/snapshots should be encrypted (and what that means)

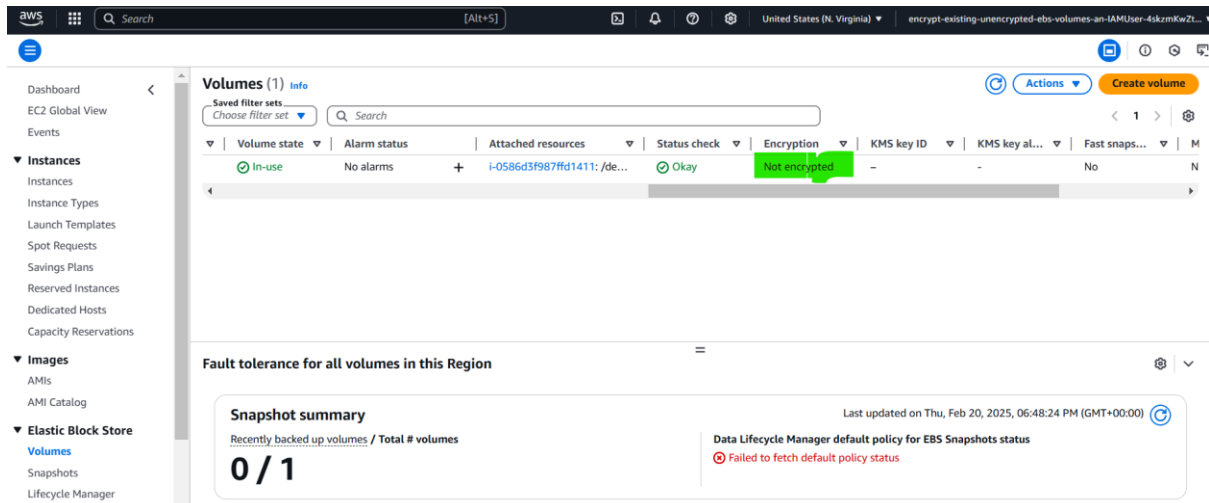
The TL;DR: This type of encryption protects against threats related to physical access to the actual disks. However, if a malicious actor gains access to your EC2 instance and successfully impersonates a legitimate user, they can still access the decrypted data—encryption at this level won't protect against that. Instead, encrypting volumes and snapshots primarily safeguards against rogue AWS employees or individuals with physical access to the underlying storage from viewing your data in plaintext.

Flow Chart



Identify unencrypted volumes:

1. Navigate to the EC2 Console.
2. In the navigation pane, select **Volumes** under **Elastic Block Store**.
3. Check the **Encryption** column for volumes marked as **Not Encrypted**.



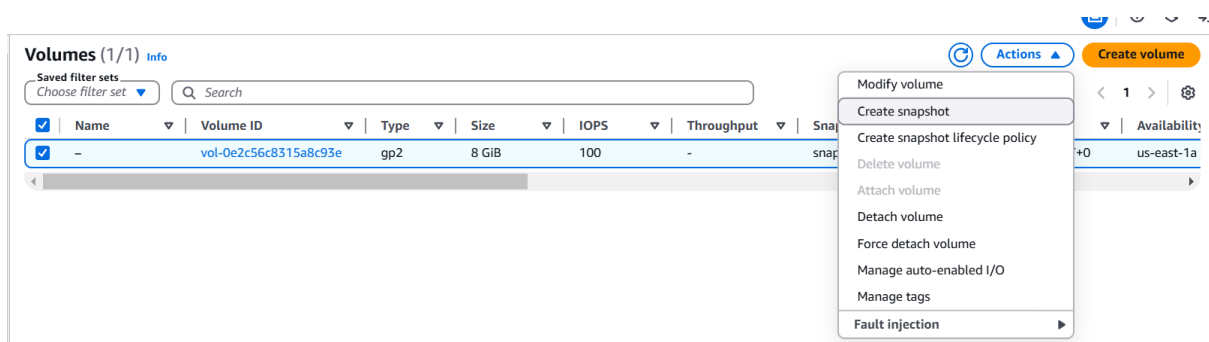
Create a snapshot of the unencrypted volume:

It's not possible to directly encrypt unencrypted volumes in AWS, so we first need to create a snapshot of the current volume so that we can then either create a new snapshot or volume from that snapshot and encrypt that new volume or snapshot.

In this lab, we'll demonstrate creating an encrypted snapshot from an unencrypted snapshot, and then creating an encrypted volume from that new encrypted snapshot. But keep in mind you could technically skip that step and create an encrypted volume from an unencrypted snapshot, if you wanted to.

Let's get started:

1. Select the unencrypted volume.
2. Choose **Actions > Create Snapshot**.



3. Enter a descriptive name and description for the snapshot.

EC2 > Volumes > vol-0e2c56c8315a8c93e > Create snapshot

Source volume
Volume ID
vol-0e2c56c8315a8c93e

Availability Zone
us-east-1a

Snapshot details
Description
Add a description for your snapshot
cybrsec-ec2-snapsho[
255 characters maximum.

Encryption [Info](#)
Not encrypted

Tags [Info](#)
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
No tags associated with the resource.
[Add tag](#)
You can add 50 more tags.

Cancel Create snapshot

4. Click Create Snapshot.

Note: If you're not familiar, EBS snapshots are point-in-time copies of EBS volumes which can be used to create incremental backups of your data. These snapshots are stored in S3 behind the scenes (you can't see the buckets in your account) and can be used to restore data.

5. You'll see the green banner once you select Create

Successfully created snapshot snap-0dfafd0eacdc1b3b5 from volume vol-0e2c56c8315a8c93e.
If you need your snapshot to be immediately available consider using Fast Snapshot Restore.

Manage fast snapshot restore

Volumes (1) [Info](#)

Saved filter sets
Choose filter set

Search

<input type="checkbox"/>	Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created	Availability
<input type="checkbox"/>	-	vol-0e2c56c8315a8c93e	gp2	8 GiB	100	-	snap-0f692dd...	2025/02/20 18:46 GMT+0	us-east-1a

Take note of the snapshot ID that starts with snap-123xx. You may click on the snapshot ID as well. (Mine look like this: [snap-0dfafd0eacdc1b3b5](#) and volume id: [vol-0e2c56c8315a8c93e](#).)

Copy the snapshot to an encrypted form:

1. Go to **Snapshots** in the EC2 Console.
2. Select the snapshot and choose **Actions > Copy Snapshot**.

Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs

Snapshots (1/1) Info

Owned by me Search

<input checked="" type="checkbox"/>	Name	Snapshot ID	Volume size	Description	Storage tier	Snapshot status
<input checked="" type="checkbox"/>	-	snap-0dfafd0eacdc1b3b5	8 GiB	cybrsec-ec2-snapshot	Standard	Completed

Snapshot ID: snap-0dfafd0eacdc1b3b5

Details
Snapshot settings
Storage tier
Tags

Snapshot ID snap-0dfafd0eacdc1b3b5 Owner 875496488523 Description cybrsec-ec2-snapshot Source volume Volume ID vol-0e2c56c8315a8c93e Encryption Encryption	Full snapshot size 1.67 GiB Started Thu Feb 20 2025 18:56:03 GMT+0000 (Greenwich Mean Time) Volume size 8 GiB KMS key ID	Progress 100% Product codes - KMS key alias	Snapshot status Completed Fast snapshot restore - KMS key ARN
--	--	--	--

- Set the **Destination Region** and check the **Encrypted** box.
- Use the AWS-managed key (default) or a customer-managed key.

Snapshots (1/1) Info

Owned by me Search

<input checked="" type="checkbox"/>	Name	Snapshot ID	Volume size	Description	Storage tier	Snapshot status
<input checked="" type="checkbox"/>	-	snap-0dfafd0eacdc1b3b5	8 GiB	cybrsec-ec2-snapshot	Standard	Completed

Recycle Bin
Actions
Create snapshot

Create volume from snapshot
Create image from snapshot
Copy snapshot
Launch copy duration calculator
Delete snapshot
Manage tags
Snapshot settings
Archiving

1

Snapshot ID: snap-0dfafd0eacdc1b3b5

Details
Snapshot settings
Storage tier
Tags

Snapshot ID snap-0dfafd0eacdc1b3b5 Owner	Full snapshot size 1.67 GiB Started	Progress 100% Product codes	Snapshot status Completed Fast snapshot restore
--	---	---	---

- Click **Copy Snapshot**.

The original snapshot that is to be copied.

📁 snap-0dfafd0eacdc1b3b5

us-east-1

Description

A description for the snapshot copy.

[Copied snap-0dfafd0eacdc1b3b5 from us-east-1] cybrsec-ec2-snapshot

255 characters maximum.

The Region in which to create the snapshot copy.

us-east-1

Specify a completion duration for the snapshot copy operation. Additional costs apply. [Learn more](#)

☐ Enable time-based copy

Use Amazon EBS encryption as an encryption solution for your EBS resources.

☒ Encrypt this snapshot

(default) sys/cha

```
(default) aws/ebs
```

- Default key that protects my EBS volumes when no other key is defined

Note: If you were to choose to use a customer-managed key in your own environment, just know CMS keys incur additional costs for AWS KMS usage, while AWS-managed keys are included in the standard EBS pricing.

Create a new encrypted volume from the encrypted snapshot:

1. Wait for the encrypted snapshot to complete.
2. Before moving on, take note of the new snapshot ID that you just created (the one with Copied ... in the description) and copy/paste it in your notes.

Snapshots (1/2) Info						
Owned by me ▾		<input type="text" value="Search"/>				
<input checked="" type="checkbox"/>	Name ▾	Snapshot ID ▾	Volume size ▾	Description ▾	Storage tier ▾	Snapshot status ▾
<input type="checkbox"/>	-	snap-0dfafd0eacdc1b3b5	8 GiB	cybrsec-ec2-snapshot	Standard	✔ Completed
<input checked="" type="checkbox"/>	-	snap-0ea07efbc76ec0d80	8 GiB	[Copied snap-0dfafd0eacdc1b3b5 from us-east-1] cybrsec-ec2-sna...	Standard	✔ Completed

3. In **Volumes**, click **Create Volume**.
4. For the Size (GiB), enter 8 instead of 100.

Create an Amazon EBS volume to attach to any EC2 instance in the same Availability Zone.

Volume settings

Volume type [Info](#)

General Purpose SSD (gp3)

Size (GiB) [Info](#)

8

Min: 1 GiB, Max: 16384 GiB.

IOPS [Info](#)

3000

Min: 3000 IOPS, Max: 16000 IOPS.

Throughput (MiB/s) [Info](#)

125

Min: 125 MiB, Max: 1000 MiB. Baseline: 125 MiB/s.

Availability Zone [Info](#)

us-east-1a

Snapshot ID - optional [Info](#)

snap-0e0a7efbc76ec0d80

Fast snapshot restore [Info](#)

Not enabled for selected snapshot

Encryption [Info](#)

Use Amazon EBS encryption as an encryption solution for your EBS resources associated with your EC2 instances.

☒ Encrypt this volume

KMS key [Info](#)

(default) aws/ebs

KMS key description

☐ Default key that protects my EBS volumes when no other key is defined

5. Select the encrypted snapshot, set the size and availability zone.
6. Ensure **Encrypt this volume** is selected.
7. Click **Create Volume**.

Successfully created volume vol-05b9d55763b9fd79b.

Volumes (1) [Info](#)

Choose filter set [Choose filter set](#)

Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created	Availability Zone	Volume state	Alarm status
-	vol-0e2c56c8315a8c93e	gp2	8 GiB	100	-	snap-0f692dd...	2025/02/20 18:46 GMT+0	us-east-1a	In-use	No alarms

Successfully created volume vol-05b9d55763b9fd79b.

Volumes (2) [Info](#)

Choose filter set [Choose filter set](#)

Throughput	Snapshot ID	Created	Availability Zone	Volume state	Alarm status	Attached resources	Status check	Encryption	KMS key
125	snap-0e0a7ef...	2025/02/20 19:17 GMT+0	us-east-1a	Available	No alarms	+	Okay	Encrypted	e4b40
-	snap-0f692dd...	2025/02/20 18:46 GMT+0	us-east-1a	In-use	No alarms	+	Okay	Not encrypted	-

Successfully created volume vol-05b9d55763b9fd79b.

Volumes (1) Info

Choose filter set

Search

	Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created	Availability Zone	Volume state	Alarm status
	-	vol-0e2c56c8315a8c93e	gp2	8 GiB	100	-	snap-0f692dd...	2025/02/20 18:46 GMT+0	us-east-1a	In-use	No alarms

Detach the unencrypted volume from the EC2 instance:

1. Go to the EC2 console and select the instance that has the unencrypted volume attached
2. From the “Storage” section, find the unencrypted volume and note its device name (e.g., /dev/xvda)

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

All states

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
		i-0586d3f987ffd1411	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-54-224-111-34.co...	54.224.111.34	You are not

i-0586d3f987ffd1411

Details Status and alarms Monitoring Security Networking **Storage** Tags

▼ Root device details

Root device name
/dev/xvda

Root device type
EBS

EBS optimization
disabled

▼ Block devices

Filter block devices

	Volume ID	Device name	Volume size (GiB)	Volume State	Attachment status	Attachment time	Encrypted	KMS key ID
	vol-0e2c56c8315a8c93e	/dev/xvda	8	In-use	Attached	2025/02/20 18:46 GMT+0	No	-

Volume monitoring (1)

3. **Stop** the EC2 instance, choose Instance State > Stop Instance. Confirm.

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

All states

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
		i-0586d3f987ffd1411	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-54-224-111-34.co...	54.224.111.34	You are not

Stop instance

Start instance

Reboot instance

Hibernate instance

Terminate (delete) instance

4. Once the instance is stopped, navigate back to the Volumes dashboard
5. You may have to refresh this page to view both volumes if you only see one
6. Detach the unencrypted volume via **Actions > Detach Volume**.
7. Confirm detachment.

Volumes (1/2) [Info](#)

Save filter sets [Choose filter set](#)

	Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created
<input type="checkbox"/>	-	vol-05b9d55763b9fd79b	gp3	8 GiB	3000	125	snap-0e0a7ef...	2025/02/20 19:17 GMT+0
<input checked="" type="checkbox"/>	-	vol-0e2c56c8315a8c93e	gp2	8 GiB	100	-	snap-0f692dd...	2025/02/20 18:46 GMT+0

Volume ID: vol-0e2c56c8315a8c93e

Details | Status checks | Monitoring | Tags

Volume ID vol-0e2c56c8315a8c93e AWS Compute Optimizer finding This user is not authorized to call AWS Compute Optimizer. Retry Fast snapshot restored No Attached resources i-0586d3f987ff14111 : /dev/xvda (attached) Source Snapshot ID snap-0f692dda8e6a338fc Encryption Not encrypted	Size 8 GiB Volume state In-use Availability Zone us-east-1a Outposts ARN -	Type gp2 IOPS 100 Created Thu Feb 20 2025 18:46:14 GMT+0000 (Greenwich Mean Time) Managed false	Status check Okay Throughput - Multi-Attach enabled No Operator -
Encryption Not encrypted	KMS key ID -	KMS key alias -	KMS key ARN -

Actions [Create volume](#)

- Modify volume
- Create snapshot
- Create snapshot lifecycle policy
- Delete volume
- Attach volume
- Detach volume**
- Force detach volume
- Manage auto-enabled I/O
- Manage tags
- Fault injection

Detach vol-0e2c56c8315a8c93e?

After you detach a volume, you might still be charged for volume storage. If you no longer need the volume, delete it to stop incurring charges.

Are you sure that you want to detach volume vol-0e2c56c8315a8c93e?

[Cancel](#) [Detach](#)

Attach the new encrypted volume to the EC2 instance:

1. Back in the EC2 console, select the stopped instance.
2. You'll now see that there are no Block devices attached.

Successfully initiated stopping of i-09a0bb7171f955462

Instances (1/1) [Info](#)

Find Instance by attribute or tag (case-sensitive) [All states](#)

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input checked="" type="checkbox"/>	-	i-09a0bb7171f955462	Stopped	t2.micro	-	View alarms	us-east-1a	-	-	You are not

i-09a0bb7171f955462

Details | Status and alarms | Monitoring | Security | Networking | **Storage** | Tags

Root device details

Root device name /dev/xvda	Root device type EBS	EBS optimization disabled
---	-------------------------	------------------------------

Block devices

	Volume ID	Device name	Volume size (GiB)	Volume State	Attachment status	Attachment time	Encrypted	KMS key ID
No block devices attached to this instance								

3. Go back to the Volumes dashboard
4. Select the new encrypted volume

5. Click on Actions > Attach Volume

The screenshot shows the AWS Management Console 'Volumes' page. At the top, there's a table of volumes. The first volume, 'vol-05bad712e264e5412', is selected. An 'Actions' dropdown menu is open, showing various options. The 'Attach volume' option is highlighted. Below the table, the details for the selected volume are shown. The volume is of type 'gp3', size '8 GiB', and is in the 'Available' state. It was created on 'Thu Feb 20 2025 20:45:13 GMT+0000 (Greenwich Mean Time)'. The 'Attach volume' button is visible at the bottom right of the details section.

6. Select the instance from the dropdown

7. Choose the device name that was previously used by the unencrypted volume (it should be /dev/xvda).

8. Click Attach Volume.

The screenshot shows the 'Attach volume' page in the AWS Management Console. The page title is 'Attach volume'. Below the title, there's a section for 'Basic details'. In this section, the 'Instance' dropdown is set to 'i-09a0bb7171f955462' and the 'Device name' is set to '/dev/xvda'. The 'Attach volume' button is visible at the bottom right of the page.

Now we can go back to the EC2 instance, refresh our EC2 page, and see that our volume is attached.

Start the EC2 instance:

1. In the EC2 console, select the instance
2. Choose Instance State > Start Instance

Instances (1/1) [Info](#) Last updated less than a minute ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) [All states](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
i-09a0bb7171f955462	i-09a0bb7171f955462	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-3-95-254-206.com...	3.95.254.206	You are not

i-09a0bb7171f955462 [Info](#)

Root device name: [/dev/xvda](#) Root device type: EBS EBS optimization: disabled

▼ **Block devices**

Filter block devices

Volume ID	Device name	Volume size (GiB)	Volume State	Attachment status	Attachment time	Encrypted	KMS key ID
vol-05bad712e264e5412	/dev/xvda	8	In-use	Attached	2025/02/20 20:52 GMT+0	Yes	e4b40a26-8d94-4111-93df-36096618

Volume monitoring (1)

Verify data integrity and encryption:

You'll need to wait a minute or so for the instance to boot up before you can connect to it, and you'll know it's ready when the Status check no longer says Initializing and instead shows a green 2/2 checks passed.

1. Connect to the EC2 instance using your preferred method (e.g., SSH or [SSM](#)).

[EC2](#) > [Instances](#) > [i-09a0bb7171f955462](#) > [Connect to instance](#)

Connect to instance [Info](#)
Connect to your instance i-09a0bb7171f955462 using any of these options

[EC2 Instance Connect](#) [Session Manager](#) [SSH client](#) [EC2 serial console](#)

Instance ID: [i-09a0bb7171f955462](#)

Connection Type

☒ **Connect using EC2 Instance Connect**
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

☐ **Connect using EC2 Instance Connect Endpoint**
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

☒ **Public IPv4 address**
[3.95.254.206](#)

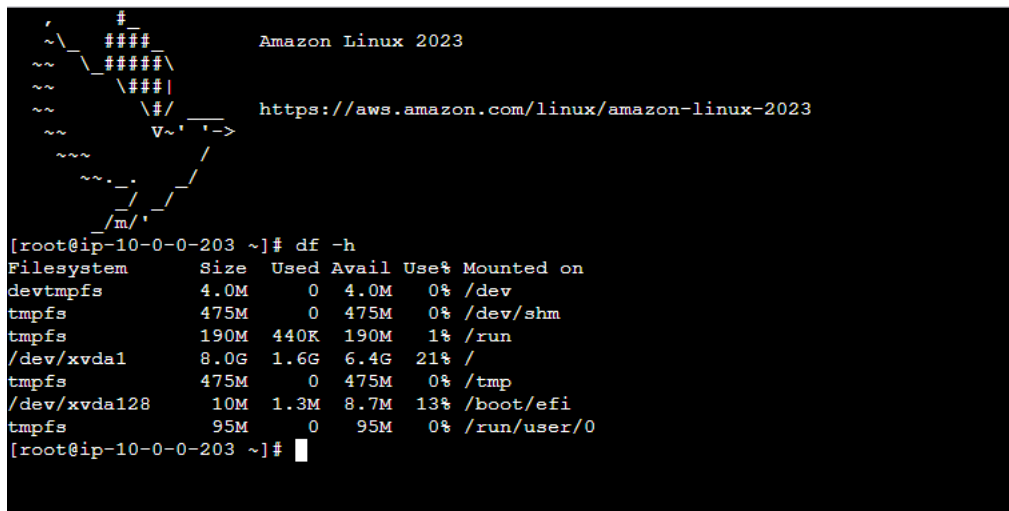
☐ **IPv6 address**
-

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, root.

Note: In most cases, the default username, root, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

[Cancel](#) [Connect](#)

2. On the EC2 Instance Connect tab, leave the defaults, ignore the warning and click on Connect.
3. Check if the data on the new encrypted volume is intact and accessible.
4. To start, you can use the df utility like this in the web-based terminal:
df -h

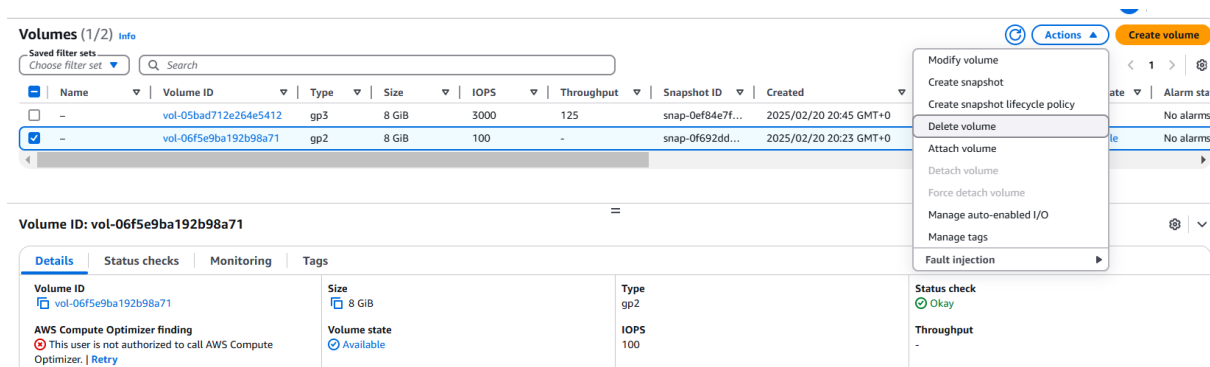


You can also use `lsblk` like this:

```
[root@ip-10-0-0-203 ~]# lsblk
NAME        MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
xvda        202:0    0  8G  0 disk
├─xvda1     202:1    0  8G  0 part /
├─xvda127   259:0    0  1M  0 part
└─xvda128   259:1    0 10M  0 part /boot/efi
[root@ip-10-0-0-203 ~]#
```

Clean up resources:

1. **Navigate** to the **EC2 Console** → **Volumes**.
2. **Select** the unencrypted volume.
3. Click **Actions** → **Delete Volume**.
4. **Confirm** by typing **delete** and then click **Delete**.
5. Ensure the volume's **State** shows as **Available** (not **In-use**) before deletion.



Automating AWS EBS Volume Encryption using Terraform, Bash, and Checkov

Also automated the process of creating unencrypted AWS EBS volumes and EC2 instances using **Terraform**, then encrypts the unencrypted volumes using a **Bash script**, and finally scans the Terraform code for security misconfigurations using the **Checkov** tool.

Tools Used

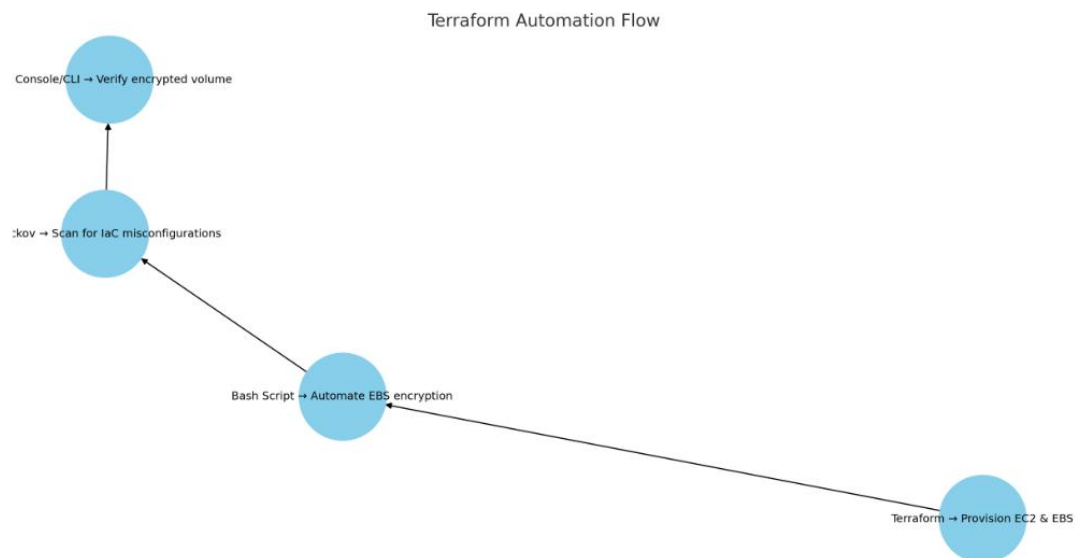
Terraform

Terraform is an open-source Infrastructure as Code (IaC) tool that enables users to define and provision infrastructure using a declarative configuration language. It simplifies cloud resource management and supports multi-cloud deployments.

Checkov

Checkov is a static code analysis tool for Infrastructure as Code (IaC). It scans Terraform, CloudFormation, Kubernetes, and other IaC frameworks for misconfigurations and security compliance violations.

Flow Chart:



Automating EBS Creation using Terraform

AWS Configuration

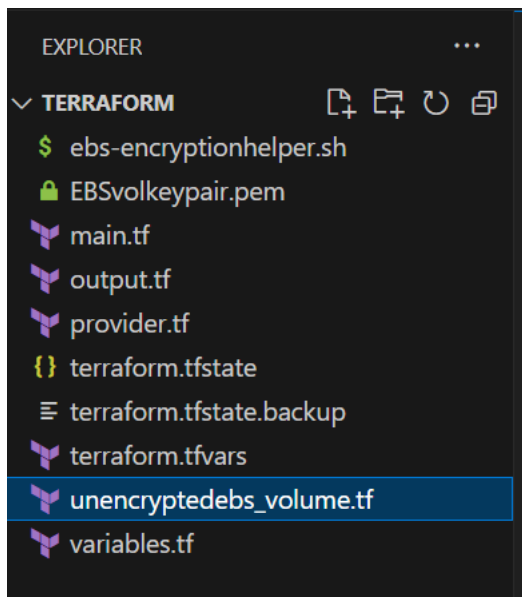
- **Run the following command to configure AWS credentials:**

aws configure

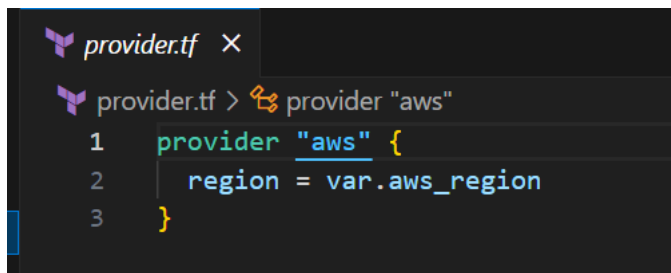
```
(kali㉿kali)-[~/EBSVolencryption]
└─$ aws configure
AWS Access Key ID [*****VRPT]:
AWS Secret Access Key [*****fgiw]:
Default region name [us-east-1]:
Default output format [json]:

(kali㉿kali)-[~/EBSVolencryption]
└─$ aws sts get-caller-identity
{
  "UserId": "AIDAJ54TPTVXXXXXX",
  "Account": "111111111111",
  "Arn": "arn:aws:iam::111111111111:user/iam-user"
}
```

Write Terraform Code



- **provider.tf** → Defines AWS as the provider.



- **main.tf** → Contains resources for EC2 instance and unencrypted EBS volumes.

```
main.tf ×
main.tf
1 terraform {
2   required_version = ">= 1.3.0"
3   required_providers {
4     aws = {
5       source = "hashicorp/aws"
6       version = "~> 5.0"
7     }
8   }
9 }
10
11 resource "aws_instance" "ec2_instance" {
12   ami           = var.ami_id
13   instance_type = var.instance_type
14   key_name      = var.key_name
15
16   tags = {
17     Name = "EBS-Encryption-Instance"
18   }
19 }
20
21 resource "aws_ebs_volume" "unencrypted_volume" {
22   availability_zone = var.availability_zone
23   size              = var.volume_size
24 }
25
26 resource "aws_volume_attachment" "attach_volume" {
27   device_name = "/dev/sdh"
28   volume_id   = aws_ebs_volume.unencrypted_volume.id
29   instance_id = aws_instance.ec2_instance.id
30 }
31
```

➤ **unencrypteddebs_volume.tf** → Defines the unencrypted EBS volume.

```
unencrypteddebs_volume.tf ×
unencrypteddebs_volume.tf > resource "aws_ebs_volume" "unencrypted_volume1"
1 resource "aws_ebs_volume" "unencrypted_volume1" {
2   availability_zone = var.availability_zone
3   size              = var.volume_size
4   encrypted         = false # ✓ Explicitly unencrypted
5
6   tags = {
7     Name = "Unencrypted-Volume1"
8   }
9 }
10
```

➤ **variables.tf** → Manages input variables.

```

variables.tf X
variables.tf > variable "aws_region"
1  variable "aws_region" {
2      description = "AWS Region"
3      default     = "us-east-1"
4      type       = string
5  }
6
7  variable "availability_zone" {
8      description = "The availability zone where the instance will be la
9      type       = string
10 }
11
12 variable "instance_type" {
13     description = "EC2 instance type"
14     default     = "t2.micro"
15 }
16
17 variable "ami_id" {
18     default     = "ami-05b10e08d247fb927"
19     description = "AMI ID for EC2 instances"
20     type       = string
21 }
22
23 variable "key_name" {
24     description = "The name of the key pair"
25     type       = string
26 }
27
28 variable "volume_size" {
29     description = "The size of the EBS volume in GB"
30     type       = number
31 }
32
33 variable "volume_type" {
34     description = "The type of the EBS volume (e.g. gp2, gp3, io1)"

```

➤ **output.tf** → Outputs key resource information.

```

output.tf X
output.tf > output "instance_id"
1  output "instance_id" {
2      value = aws_instance.ec2_instance.id
3  }
4
5  output "volume_id" {
6      value = aws_ebs_volume.unencrypted_volume.id
7  }

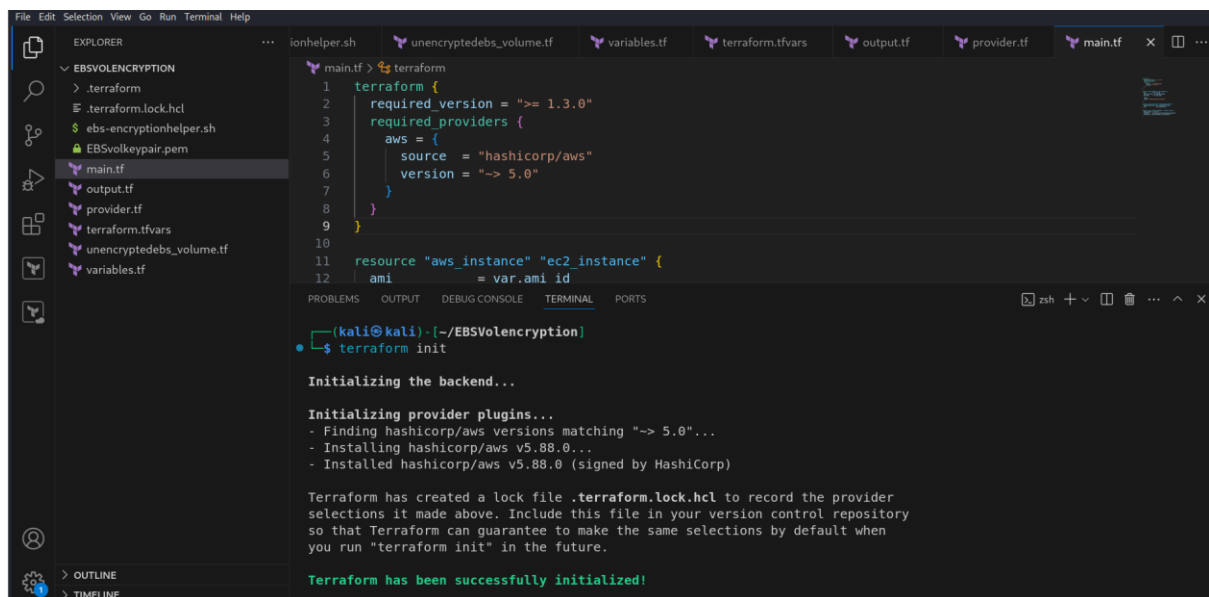
```

➤ **terraform.tfvars** → Holds variable values.

```
terraform.tfvars ×  
terraform.tfvars > aws_region  
1  aws_region      = "us-east-1"  
2  ami_id          = "ami-05b10e08d247fb927"  
3  instance_type   = "t2.micro"  
4  key_name        = "EBSvolkeypair"  
5  availability_zone = "us-east-1b"  
6  volume_size     = 8  
7  volume_type     = "gp3"  
8
```

Execute Terraform Commands

- terraform init # Initialize Terraform



```
File Edit Selection View Go Run Terminal Help  
EXPLORER  
EBSVOLENCRIPTION  
  .terraform  
  .terraform.lock.hcl  
  ebs-encryptionhelper.sh  
  EBSvolkeypair.pem  
  main.tf  
  output.tf  
  provider.tf  
  terraform.tfvars  
  unencryptedebs_volume.tf  
  variables.tf  
main.tf  
1 terraform {  
2   required_version = ">= 1.3.0"  
3   required_providers {  
4     aws = {  
5       source = "hashicorp/aws"  
6       version = "~> 5.0"  
7     }  
8   }  
9 }  
10  
11 resource "aws_instance" "ec2_instance" {  
12   ami = var.ami_id  
13 }  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
(kali@kali) - [~/EBSVolencryption]  
$ terraform init  
Initializing the backend...  
  
Initializing provider plugins...  
- Finding hashicorp/aws versions matching "~> 5.0"...  
- Installing hashicorp/aws v5.88.0...  
- Installed hashicorp/aws v5.88.0 (signed by HashiCorp)  
  
Terraform has created a lock file .terraform.lock.hcl to record the provider  
selections it made above. Include this file in your version control repository  
so that Terraform can guarantee to make the same selections by default when  
you run "terraform init" in the future.  
  
Terraform has been successfully initialized!
```

- terraform fmt # Format configuration files

```
(kali@kali) - [~/EBSVolencryption]  
$ terraform fmt  
unencryptedebs_volume.tf  
variables.tf
```

- terraform validate # Validate the code

```
(kali@kali) - [~/EBSVolencryption]  
$ terraform validate  
Success! The configuration is valid.
```

- terraform plan # Preview the changes

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(kali@kali) - [~/EBSVolencryption]
$ terraform plan
aws_ebs_volume.unencrypted_volume: Refreshing state... [id=vol-03091415f153dfac7]
aws_instance.ec2_instance: Refreshing state... [id=i-0888aabdfc4a1d3bc]
aws_ebs_volume.unencrypted_volume1: Refreshing state... [id=vol-0dbd5d1b4351d4d64]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
-/+ destroy and then create replacement

Terraform will perform the following actions:

# aws_ebs_volume.unencrypted_volume must be replaced
-/+ resource "aws_ebs_volume" "unencrypted_volume" {
  ~ arn          = "arn:aws:ec2:us-east-1:337909744329:volume/vol-03091415f153dfac7" -> (known after apply)
```

➤ terraform apply # Apply the infrastructure

```
(kali@kali) - [~/EBSVolencryption]
$ terraform apply
aws_ebs_volume.unencrypted_volume1: Refreshing state... [id=vol-0dbd5d1b4351d4d64]
aws_ebs_volume.unencrypted_volume: Refreshing state... [id=vol-03091415f153dfac7]
aws_instance.ec2_instance: Refreshing state... [id=i-0888aabdfc4a1d3bc]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
-/+ destroy and then create replacement

Terraform will perform the following actions:

# aws_ebs_volume.unencrypted_volume must be replaced
-/+ resource "aws_ebs_volume" "unencrypted_volume" {
  ~ arn          = "arn:aws:ec2:us-east-1:337909744329:volume/vol-03091415f153dfac7" -> (known after apply)
```

```
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_ebs_volume.unencrypted_volume1: Destroying... [id=vol-0dbd5d1b4351d4d64]
aws_ebs_volume.unencrypted_volume: Destroying... [id=vol-03091415f153dfac7]
aws_ebs_volume.unencrypted_volume1: Still destroying... [id=vol-0dbd5d1b4351d4d64, 10s elapsed]
aws_ebs_volume.unencrypted_volume: Still destroying... [id=vol-03091415f153dfac7, 10s elapsed]
aws_ebs_volume.unencrypted_volume: Destruction complete after 11s
aws_ebs_volume.unencrypted_volume1: Destruction complete after 11s
aws_ebs_volume.unencrypted_volume: Creating...
aws_ebs_volume.unencrypted_volume1: Creating...
aws_ebs_volume.unencrypted_volume: Still creating... [10s elapsed]
aws_ebs_volume.unencrypted_volume1: Still creating... [10s elapsed]
aws_ebs_volume.unencrypted_volume: Creation complete after 11s [id=vol-0e35390d8b6e88a8c]
aws_volume_attachment.attach_volume: Creating...
aws_ebs_volume.unencrypted_volume1: Creation complete after 11s [id=vol-0e2871ea10ec31a0d]
aws_volume_attachment.attach_volume: Still creating... [10s elapsed]
aws_volume_attachment.attach_volume: Still creating... [20s elapsed]
aws_volume_attachment.attach_volume: Creation complete after 22s [id=vai-416811575]

Apply complete! Resources: 3 added, 0 changed, 2 destroyed.

Outputs:

instance_id = "i-0888aabdfc4a1d3bc"
volume_id = "vol-0e35390d8b6e88a8c"
```

Deployment is successful, you can verify that manually by checking AWS management console:

Verify EBS Creation

- AWS Management Console:
 - Check EBS volumes, snapshots, and EC2 instances.

GMT+0	us-east-1b	Available	No alarms	+	-	Okay	Not encrypted	-	-	No	No
:21 GMT+0	us-east-1b	In-use	No alarms	+	-	Okay	Not encrypted	-	-	No	No

Volume ID: vol-0e2871ea10ec31a0d (Unencrypted-Volume1)

Details | Status checks | Monitoring | Tags

Volume ID
vol-0e2871ea10ec31a0d (Unencrypted-Volume1)

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendation s. | [Learn more](#)

Fast snapshot restored
No

Attached resources
-

Source
Snapshot ID
-

Encryption
Encryption
Not encrypted

Size
8 GiB

Volume state
Available

Availability Zone
us-east-1b

Outposts ARN
-

Type
gp2

IOPS
100

Created
Tue Feb 25 2025 00:29:40 GMT+0000 (Greenwich Mean Time)

Managed
false

Status check
Okay

Throughput
-

Multi-Attach enabled
No

Operator
-

KMS key ID
-

KMS key alias
-

KMS key ARN
-

○ Check in snapshots

snap-08580043db7a923f6

[Details](#) | [Delete](#) | [Actions](#)

Details

Snapshot ID
snap-08580043db7a923f6

Owner
107513503799

Description
Created by CreateImage(i-089b146125db92ee4) for ami-0676627ee43624fb2

Source volume

Volume ID
vol-04462a3562c7e6a15

Volume size
8 GiB

Encryption
Encryption
Not encrypted

KMS key ID
-

KMS key alias
-

KMS key ARN
-

Full snapshot size
8 GiB

Started
Mon Jun 26 2023 00:08:45 GMT+0100 (British Summer Time)

Progress
100%

Product codes
-

Snapshot status
Completed

Fast snapshot restore
-

Tags

Tags

Key | **Value**

The selected resource currently has no tags.

[Manage tags](#)

Instances (1/2)

[All states](#)

[Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4...	Elastic IP
EBS-Encrypto...	i-0888aabdfc4a1d3bc	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-18-206-154-137.co...	18.206.154.137	-
EBS-Encrypto...	i-0241e93b77b392e92	Terminated	t2.micro	-	View alarms +	us-east-1b	-	-	-

i-0888aabdfc4a1d3bc (EBS-Encryption-Instance)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary

Instance ID
i-0888aabdfc4a1d3bc

IPv6 address
-

Hostname type
IP name: ip-172-31-87-99.ec2.internal

Answer private resource DNS name
-

Auto-assigned IP address
18.206.154.137 [Public IP]

IAM Role
-

IMDSv2
Required

Operator
-

Public IPv4 address
18.206.154.137 | [open address](#)

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-87-99.ec2.internal

Instance type
t2.micro

VPC ID
vpc-0fe195bf8e0e5544d

Subnet ID
subnet-0e5753145411dd066

Instance ARN
arn:aws:ec2:us-east-1:337909744329:instance/i-0888aabdfc4a1d3bc

Private IPv4 addresses
172.31.87.99

Public IPv4 DNS
ec2-18-206-154-137.compute-1.amazonaws.com | [open address](#)

Elastic IP addresses
-

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | [Learn more](#)

Auto Scaling Group name
-

Managed
false

➤ Connecting to EC2 instance using SSH command as shown below:

```
(kali㉿kali)-[~/EBSVolencryption]
└─$ chmod 400 "EBSvolkeypair.pem"

(kali㉿kali)-[~/EBSVolencryption]
└─$ ssh -i "EBSvolkeypair.pem" ec2-user@ec2-18-206-154-137.compute-1.amazonaws.com
The authenticity of host 'ec2-18-206-154-137.compute-1.amazonaws.com (18.206.154.137)' can't be established.
ED25519 key fingerprint is SHA256:98dXF8+GEKUgNphbjRAJF6A56eoB3xQSpaiie6rHbRs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-206-154-137.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

      #
    _\#####_   Amazon Linux 2023
   ~~~ \#####\
       ~~~ \###|
           \#/   https://aws.amazon.com/linux/amazon-linux-2023
          V~' '->
         ~~~~~
        ~~~~_/_/_/
       ~~~~/_/_/_/
      ~~~~/_m/'
[ec2-user@ip-172-31-87-99 ~]$ whoami
ec2-user
```

Using Terraform code successfully created unencrypted volumes.

Run Bash Script


```
EBStvolkeypair.pem  $ ebs-encryptionhelper.sh X
$ ebs-encryptionhelper.sh
1  #!/bin/bash
2
3  REGION="us-east-1"
4  INSTANCE_ID="i-0888aabdfc4a1d3bc" # Replace with your EC2 Instance ID
5  AVAILABILITY_ZONE="us-east-1b" # Replace with your instance's AZ
6
7  # Stop the EC2 Instance
8  aws ec2 stop-instances --instance-ids $INSTANCE_ID --region $REGION
9  echo "Stopping instance $INSTANCE_ID..."
10 aws ec2 wait instance-stopped --instance-ids $INSTANCE_ID --region $REGION
11 echo "Instance $INSTANCE_ID stopped."
12
13 # Identify unencrypted EBS volumes
14 echo "Identifying unencrypted EBS volumes..."
15 unencrypted_volumes=$(aws ec2 describe-volumes --filters Name=attachment.instance-id,Values=$INSTANCE_ID -
16
17 for vol in $unencrypted_volumes; do
18     echo "Processing volume: $vol"
19
20     # Create Snapshot
21     snapshot_id=$(aws ec2 create-snapshot --volume-id $vol --description "Snapshot of $vol for encryption" -
22     echo "Snapshot created: $snapshot_id"
23
24     # Wait for Snapshot Completion
25     echo "Waiting for snapshot to complete..."
26     aws ec2 wait snapshot-completed --snapshot-ids $snapshot_id --region $REGION
27
28     # Copy Snapshot with Encryption
29     encrypted_snapshot=$(aws ec2 copy-snapshot --source-region $REGION --source-snapshot-id $snapshot_id --e
30     echo "Encrypted snapshot created: $encrypted_snapshot"
31
32     # Wait for Encrypted Snapshot Completion
33     echo "Waiting for encrypted snapshot to complete..."
34     aws ec2 wait snapshot-completed --snapshot-ids $encrypted_snapshot --region $REGION
```

```
(kali@kali)-[~/EBStvolencryption]
└─$ ./ebs-encryptionhelper.sh
{
  "StoppingInstances": [
    {
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "InstanceId": "i-0888aabdfc4a1d3bc",
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
Stopping instance i-0888aabdfc4a1d3bc...
Instance i-0888aabdfc4a1d3bc stopped.
Identifying unencrypted EBS volumes...
Processing volume: vol-0467be32dfa110f3b
```

Verify Encrypted Volumes

- **AWS Management Console:**
 - Check the updated EBS volume encryption status.

Instances (1/2) Info

Find Instance by attribute or tag (case-sensitive) All states

Last updated 1 minute ago Connect Instance state Actions Launch instances

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input checked="" type="checkbox"/>	EBS-Encryption...	i-0888aabdfc4a1d3bc	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-90-11-29.comput...	3.90.11.29	-
<input type="checkbox"/>	EBS-Encryption...	i-0241e93b77b392e92	Terminated	t2.micro	-	View alarms +	us-east-1b	-	-	-

i-0888aabdfc4a1d3bc (EBS-Encryption-Instance)

Root device name: [/dev/xvda](#) Root device type: EBS EBS optimization: disabled

▼ Block devices

Filter block devices

	Volume ID	Device name	Volume size (GiB)	Volume State	Attachment status	Attachment time	Encrypted	KMS key ID
<input type="checkbox"/>	vol-0f5db8d4c0ab6a46b	/dev/xvda	8	In-use	Attached	2025/02/25 01:43 GMT+0	Yes	11ad101d-6f79-46b9-8caf-042cc4b03

Volumes (1) Info

Volume ID = [vol-0f5db8d4c0ab6a46b](#) Clear filters

Volume state	Alarm status	Attached resources	Status check	Encryption	KMS key ID	KMS key al...	Fast snaps...	Multi-Atta...	Managed	Operator
In-use	No alarms	i-0888aabdfc4a1d3bc (EBS...	Okay	Encrypted	11ad101d-6f7...	aws/ebs	No	No	false	-

Volumes (6) Info

Snapshot ID	Created	Availability Zone	Volume state	Alarm status	Attached resources	Status check	Encryption
snap-000e4f2cebae29642	2025/02/25 00:21 GMT+0	us-east-1b	Available	No alarms	-	Okay	Not encrypted
snap-0f957171c3404738e	2025/02/25 01:26 GMT+0	us-east-1b	Available	No alarms	-	Okay	Encrypted
snap-0d98412f6a306aace	2025/02/25 01:31 GMT+0	us-east-1b	Available	No alarms	-	Okay	Encrypted
snap-02fe4233e6a6c6aa4	2025/02/25 01:43 GMT+0	us-east-1b	In-use	No alarms	i-0888aabdfc4a1d3bc (EBS...	Okay	Encrypted

- **AWS CLI:**

```
(kali㉿kali)-[~/EBSVolencryption]
aws ec2 describe-volumes --filters Name=attachment.instance-id,Values=i-0888aabdfc4a1d3bc --query "Volumes[*].{ID:VolumeId,Encrypted:Encrypted}" --region us-east-1 --output table
```

Encrypted	ID
True	vol-0f5db8d4c0ab6a46b

Try connecting to EC2 instance, to verify the volumes created or not using SSH

```
(kali㉿kali)-[~/EBSVolencryption]
$ ssh -i "EBSvolkeypair.pem" ec2-user@ec2-3-90-11-29.compute-1.amazonaws.com
The authenticity of host 'ec2-3-90-11-29.compute-1.amazonaws.com (3.90.11.29)' can't be established.
ED25519 key fingerprint is SHA256:98dXF8+GEKUGNphbjRAJF6A56eoB3xqSPaiei6rHbRs.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:6: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-90-11-29.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
```

Last login: Tue Feb 25 01:02:34 2025 from 90.251.127.120

ec2-user

Filesystem	Size	Used	Avail	Use%	Mounted on
devtmpfs	4.0M	0	4.0M	0%	/dev
tmpfs	475M	0	475M	0%	/dev/shm
tmpfs	190M	448K	190M	1%	/run
/dev/xvda1	8.0G	1.6G	6.4G	21%	/
tmpfs	475M	0	475M	0%	/tmp
/dev/xvda128	10M	1.3M	8.7M	13%	/boot/efi

```

tmpfs          95M      0     95M      0% /run/user/1000

```

```
NAME      MAJ:MIN  RM  SIZE  RO  TYPE  MOUNTPOINTS
xvda      202:0    0    8G    0  disk
└─xvda1    202:1    0    8G    0  part  /
└─xvda127 259:0    0    1M    0  part
└─xvda128 259:1    0   10M    0  part  /boot/efi
```

Scanning Terraform Code Using Checkov

```
checkov -d /home/kali/EBSVolencryption/
```

```
(checkov-env)-(kali@kali) [~/EBSVolencyption]
$ checkov -d /home/kali/EBSVolencyption/

2025-02-24 21:29:59,188 [MainThread ] [WARNI] An unsupported instruction IMPORT was used in /checkov-env/lib/pytho
n3.13/site-packages/checkov/common/util/dockerfile.py
2025-02-24 21:29:59,194 [MainThread ] [WARNI] An unsupported instruction DOCKERFILE_MASK was used in /checkov-env/
lib/python3.13/site-packages/checkov/common/util/dockerfile.py
2025-02-24 21:29:59,199 [MainThread ] [WARNI] An unsupported instruction DEF was used in /checkov-env/lib/python3.
13/site-packages/checkov/common/util/dockerfile.py
2025-02-24 21:29:59,204 [MainThread ] [WARNI] An unsupported instruction IF was used in /checkov-env/lib/python3.1
3/site-packages/checkov/common/util/dockerfile.py
2025-02-24 21:29:59,205 [MainThread ] [WARNI] An unsupported instruction RETURN was used in /checkov-env/lib/pytho
n3.13/site-packages/checkov/common/util/dockerfile.py
2025-02-24 21:29:59,214 [MainThread ] [WARNI] An unsupported instruction RETURN was used in /checkov-env/lib/pytho
n3.13/site-packages/checkov/common/util/dockerfile.py
[ dockerfile framework ]: 100% [REDACTED] [1/1], Current File Scanned=../../checkov-env/lib/python3.13/
[ terraform framework ]: 100% [REDACTED] [5/5], Current File Scanned=variables.tf~env/lib/python3.13/site
[ kubernetes framework ]: 100% [REDACTED] [1349/1349], Current File Scanned=checkov-env/lib/python3.13/sit
[ secrets framework ]: 1% [REDACTED] [15/1351], Current File Scanned=/home/kali/EBSVolencyption/checkov
```


- Apply AWS Config rules to enforce encryption compliance.
- Conduct periodic audits and vulnerability assessments.

Incident Response and Data Recovery

A comprehensive data recovery plan ensures data integrity in case of accidental deletions or breaches. Using AWS Backup and automated snapshots helps maintain up-to-date data copies.

- **Key Points:**
 - Schedule regular snapshots using AWS Data Lifecycle Manager.
 - Implement cross-region backups for disaster recovery.
 - Use AWS Backup for centralized backup management across services.
 - Configure CloudWatch alarms for snapshot failures.
 - Test recovery procedures regularly to ensure data availability.

Additional Tips

- **Enable EBS Encryption by Default:** Set this at the account level if all EBS volumes should be encrypted. Navigate to **EC2 Dashboard** → **Account Attributes** → **Data Protection and Security** → **EBS Encryption** and set “**Always encrypt new EBS volumes**” to “**Enabled**”.
- **Backup Regularly:** Regularly create snapshots of your volumes to ensure you have up-to-date backups in case of data loss or corruption. (*Automate this process for efficiency.*)
- **Automate Snapshot Management:** Use **AWS Data Lifecycle Manager** to automate the creation and retention of EBS snapshots, eliminating manual interventions.
- **Monitor Volume Usage:** Utilize **AWS CloudWatch** to track EBS volume performance and usage metrics, helping to identify potential issues early.
- **Cost Management:** Periodically review EBS usage and remove unused volumes or outdated snapshots to optimize costs.
- **Tagging:** Tag all resources (volumes, snapshots) for better organization, cost allocation, and easier identification of orphaned resources.
- **Enforce Encryption with AWS Config:** Besides enabling encryption by default, use **AWS Config** to enforce policies ensuring all new EBS volumes remain encrypted.
- **Instance Metadata Service (IMDS):** Ensure **IMDS** is enabled and securely configured to allow instances to retrieve necessary credentials for encryption and other operations.

Real-World Use Cases and Scenarios

- **Financial Services:** Encrypting sensitive transaction data to comply with PCI DSS.
- **Healthcare Organizations:** Ensuring HIPAA compliance by encrypting patient data.
- **Startups:** Using customer-managed keys for enhanced security and compliance.
- **E-commerce Platforms:** Protecting customer payment data and personal information.
- **Government Agencies:** Meeting strict data sovereignty and security requirements.

Conclusion

Securing Amazon EBS volumes with encryption is a vital step in protecting sensitive data within AWS environments. By leveraging AWS Key Management Service (KMS) for key control, implementing fine-grained IAM policies, and using security tools like Checkov for infrastructure compliance, organizations can significantly reduce the risk of data breaches and unauthorized access.

However, encryption alone is not a silver bullet. It must be part of a broader security strategy that includes continuous monitoring, regular backups, automated snapshot management, and performance tracking. Integrating Infrastructure as Code (IaC) practices with Terraform and incorporating security checks ensures consistency, scalability, and proactive risk mitigation.

By adopting this complete framework for EBS data protection and encryption, businesses can not only meet stringent compliance requirements like HIPAA, PCI DSS, and GDPR but also strengthen their overall cloud security posture. This layered approach fosters operational resilience, enhances data integrity, and builds trust with clients and stakeholders in an increasingly cloud-centric world.

References:

AWS. (n.d.). *Amazon EC2 instance connect prerequisites*. Available at: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-connect-prerequisites.html> [Accessed 25 Feb. 2025].

AWS. (n.d.). *Amazon EC2 instance storage: EBS-optimized instances*. Available at: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-optimized.html> [Accessed 25 Feb. 2025].

AWS. (n.d.). *AWS Backup Developer Guide*. Available at: <https://docs.aws.amazon.com/aws-backup/latest/devguide/whatisbackup.html> [Accessed 25 Feb. 2025].

AWS. (n.d.). *AWS compliance programs*. Available at: <https://aws.amazon.com/compliance/programs/> [Accessed 25 Feb. 2025].

AWS. (n.d.). *AWS Key Management Service (KMS) Developer Guide*. Available at: <https://docs.aws.amazon.com/kms/latest/developerguide/overview.html> [Accessed 25 Feb. 2025].

AWS. (n.d.). *AWS Well-Architected Framework: Security Pillar*. Available at: <https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html> [Accessed 25 Feb. 2025].

AWS. (n.d.). *Amazon EBS Encryption*. Available at: <https://docs.aws.amazon.com/ebs/latest/userguide/ebs-encryption.html> [Accessed 25 Feb. 2025].

AWS. (n.d.). *Amazon Elastic Block Store (EBS) volumes*. Available at: <https://docs.aws.amazon.com/ebs/latest/userguide/ebs-volumes.html> [Accessed 25 Feb. 2025].

Checkov. (n.d.). *Installing Checkov*. Available at: <https://www.checkov.io/2.Basics/Installing%20Checkov.html> [Accessed 25 Feb. 2025].

Cybr. (n.d.). *EBS Data Protection and Encryption*. Available at: <https://cybr.com/courses/introduction-to-aws-security/lessons/ebs-data-protection-and-encryption/> [Accessed 25 Feb. 2025].

Terraform. (n.d.). *AWS Provider Documentation*. Available at: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs> [Accessed 25 Feb. 2025].

Appendices:

```
└─(kali㉿kali)-[~/EBSVolencryption]
```

```
└─$ terraform plan
```

```
aws_ebs_volume.unencrypted_volume: Refreshing state... [id=vol-03091415f153dfac7]
```

```
aws_instance.ec2_instance:      Refreshing      state...      [id=i-0888aabdfc4a1d3bc]
```

```
aws_ebs_volume.unencrypted_volume1:      Refreshing      state...      [id=vol-0dbd5d1b4351d4d64]
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with

the following symbols:

+ create

-/+ destroy and then create replacement

Terraform will perform the following actions:

```
# aws_ebs_volume.unencrypted_volume must be replaced
-/+ resource "aws_ebs_volume" "unencrypted_volume" {
    ~ arn                        = "arn:aws:ec2:us-east-1:337909744329:volume/vol-03091415f153dfac7" -> (known after apply)
    ~ availability_zone         = "us-east-1a" -> "us-east-1b" # forces replacement
    ~ encrypted                 = false -> (known after apply)
    ~ id                       = "vol-03091415f153dfac7" -> (known after apply)
    ~ iops                     = 100 -> (known after apply)
    + kms_key_id               = (known after apply)
    - multi_attach_enabled     = false -> null
    + snapshot_id              = (known after apply)
    - tags                     = {} -> null
```



```

    ~ tags_all                = {} -> (known after apply)
    ~ throughput              = 0 -> (known after apply)
    ~ type                    = "gp2" -> (known after apply)
    # (2 unchanged attributes hidden)
  }

  # aws_ebs_volume.unencrypted_volume1 must be replaced
-/+ resource "aws_ebs_volume" "unencrypted_volume1" {
    ~ arn                    = "arn:aws:ec2:us-east-1:337909744329:volume/vol-0dbd5d1b4351d4d64" -> (known after apply)
    ~ availability_zone      = "us-east-1a" -> "us-east-1b" # forces replacement
    ~ id                    = "vol-0dbd5d1b4351d4d64" -> (known after apply)
    ~ iops                  = 100 -> (known after apply)
    + kms_key_id            = (known after apply)
    - multi_attach_enabled = false -> null
    + snapshot_id          = (known after apply)
    tags                    = {
        "Name" = "Unencrypted-Volume1"
    }
    ~ throughput            = 0 -> (known after apply)
    ~ type                  = "gp2" -> (known after apply)
    # (4 unchanged attributes hidden)
  }

  # aws_volume_attachment.attach_volume will be created
+ resource "aws_volume_attachment" "attach_volume" {
    + device_name = "/dev/sdh"
    + id          = (known after apply)
    + instance_id = "i-0888aabdfc4a1d3bc"

```

```
+ volume_id    = (known after apply)
}
```

Plan: 3 to add, 0 to change, 2 to destroy.

Changes to Outputs:

```
~ volume_id    = "vol-03091415f153dfac7" -> (known after apply)
```

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions

if you run "terraform apply" now.

```
└─(kali㉿kali)-[~/EBSVolencryption]
```

```
└─$ terraform apply
```

```
aws_ebs_volume.unencrypted_volume1: Refreshing state...
[id=vol-0dbd5d1b4351d4d64]
```

```
aws_ebs_volume.unencrypted_volume: Refreshing state... [id=vol-
03091415f153dfac7]
```

```
aws_instance.ec2_instance: Refreshing state... [id=i-
0888aabdfc4a1d3bc]
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with

the following symbols:

+ create

-/+ destroy and then create replacement

Terraform will perform the following actions:

```

# aws_ebs_volume.unencrypted_volume must be replaced
-/+ resource "aws_ebs_volume" "unencrypted_volume" {
    ~ arn = "arn:aws:ec2:us-east-1:337909744329:volume/vol-03091415f153dfac7" -> (known after apply)
    ~ availability_zone = "us-east-1a" -> "us-east-1b" # forces replacement
    ~ encrypted = false -> (known after apply)
    ~ id = "vol-03091415f153dfac7" -> (known after apply)
    ~ iops = 100 -> (known after apply)
    + kms_key_id = (known after apply)
    - multi_attach_enabled = false -> null
    + snapshot_id = (known after apply)
    - tags = {} -> null
    ~ tags_all = {} -> (known after apply)
    ~ throughput = 0 -> (known after apply)
    ~ type = "gp2" -> (known after apply)
    # (2 unchanged attributes hidden)
}

```

```

# aws_ebs_volume.unencrypted_volume1 must be replaced
-/+ resource "aws_ebs_volume" "unencrypted_volume1" {
    ~ arn = "arn:aws:ec2:us-east-1:337909744329:volume/vol-0dbd5d1b4351d4d64" -> (known after apply)
    ~ availability_zone = "us-east-1a" -> "us-east-1b" # forces replacement
    ~ id = "vol-0dbd5d1b4351d4d64" -> (known after apply)
    ~ iops = 100 -> (known after apply)
    + kms_key_id = (known after apply)
    - multi_attach_enabled = false -> null
    + snapshot_id = (known after apply)
}

```

```

tags                = {
    "Name" = "Unencrypted-Volume1"
}
~ throughput        = 0 -> (known after apply)
~ type               = "gp2" -> (known after apply)
# (4 unchanged attributes hidden)
}

```

```

# aws_volume_attachment.attach_volume will be created
+ resource "aws_volume_attachment" "attach_volume" {
    + device_name = "/dev/sdh"
    + id          = (known after apply)
    + instance_id = "i-0888aabdfc4a1d3bc"
    + volume_id   = (known after apply)
}

```

Plan: 3 to add, 0 to change, 2 to destroy.

Changes to Outputs:

```

~ volume_id = "vol-03091415f153dfac7" -> (known after apply)

```

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

```

aws_ebs_volume.unencrypted_volume1:    Destroying...    [id=vol-0dbd5d1b4351d4d64]
aws_ebs_volume.unencrypted_volume:     Destroying...    [id=vol-03091415f153dfac7]

```

```
aws_ebs_volume.unencrypted_volume1:      Still      destroying...  
[id=vol-0dbd5d1b4351d4d64, 10s elapsed]  
  
aws_ebs_volume.unencrypted_volume: Still destroying... [id=vol-  
03091415f153dfac7, 10s elapsed]  
  
aws_ebs_volume.unencrypted_volume: Destruction complete after  
11s  
  
aws_ebs_volume.unencrypted_volume1: Destruction complete after  
11s  
  
aws_ebs_volume.unencrypted_volume: Creating...  
aws_ebs_volume.unencrypted_volume1: Creating...  
  
aws_ebs_volume.unencrypted_volume:      Still      creating...   [10s  
elapsed]  
  
aws_ebs_volume.unencrypted_volume1:      Still      creating...   [10s  
elapsed]  
  
aws_ebs_volume.unencrypted_volume: Creation complete after 11s  
[id=vol-0e35390d8b6e88a8c]  
  
aws_volume_attachment.attach_volume: Creating...  
  
aws_ebs_volume.unencrypted_volume1: Creation complete after 11s  
[id=vol-0e2871ea10ec31a0d]  
  
aws_volume_attachment.attach_volume:      Still      creating...   [10s  
elapsed]  
  
aws_volume_attachment.attach_volume:      Still      creating...   [20s  
elapsed]  
  
aws_volume_attachment.attach_volume: Creation complete after  
22s [id=vai-416811575]
```

Apply complete! Resources: 3 added, 0 changed, 2 destroyed.

Outputs:

```
instance_id = "i-0888aabdfc4a1d3bc"  
volume_id = "vol-0e35390d8b6e88a8c"
```

└─(kali㉿kali)-[~/EBSVolencryption]

```
└─$
```

```
└─(kali㉿kali)-[~/EBSVolencryption]
```

```
└─$ chmod 400 "EBSvolkeypair.pem"
```

```
└─(kali㉿kali)-[~/EBSVolencryption]
```

```
└─$ ssh -i "EBSvolkeypair.pem" ec2-user@ec2-18-206-154-137.compute-1.amazonaws.com
```

```
The authenticity of host 'ec2-18-206-154-137.compute-1.amazonaws.com (18.206.154.137)' can't be established.
```

```
ED25519 key fingerprint is  
SHA256:98dXF8+GEkUgNphbjRAJF6A56eoB3xqSPaie6rHbRs.
```

```
This key is not known by any other names.
```

```
Are you sure you want to continue connecting  
(yes/no/[fingerprint])? yes
```

```
Warning: Permanently added 'ec2-18-206-154-137.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
```

```
,      #_  
~\__ ####_      Amazon Linux 2023  
~~  \_#####\  
~~      \###|  
~~      \#/  ____  https://aws.amazon.com/linux/amazon-linux-2023  
~~      v~'  '->  
~~~      /
```

```
[ec2-user@ip-172-31-87-99 ~]$ whoami ec2-user [ec2-user@ip-172-31-87-99 ~]$ df -h  
Filesystem Size Used Avail Use% Mounted on  
devtmpfs 4.0M 0 4.0M 0% /dev  
tmpfs 475M 0 475M 0% /dev/shm  
tmpfs 190M 452K 190M 1% /run  
/dev/xvda1 8.0G 1.6G 6.4G 20% /  
tmpfs 475M 0 475M 0% /tmp  
/dev/xvda128 10M 1.3M 8.7M 13% /boot/efi  
tmpfs 95M 0 95M 0% /run/user/1000  
[ec2-user@ip-172-31-87-99 ~]$ lsblk  
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS  
xvda 202:0 0 8G 0 disk └─xvda1 202:1 0 8G 0 part / └─xvda127 259:0 0 1M 0 part  
└─xvda128 259:1 0 10M 0 part /boot/efi  
xvdh 202:112 0 8G 0 di
```

```
└─(kali㉿kali)-[~/EBSVolencryption]
```

```
└─$ ./ebs-encryptionhelper.sh
```

```
{
  "StoppingInstances": [
    {
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "InstanceId": "i-0888aabdfc4a1d3bc",
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}

Stopping instance i-0888aabdfc4a1d3bc...
Instance i-0888aabdfc4a1d3bc stopped.
Identifying unencrypted EBS volumes...
Processing volume: vol-0467be32dfa110f3b
Snapshot created: snap-079b69e791ed0145c
Waiting for snapshot to complete...
Encrypted snapshot created: snap-02fe4233e6a6c6aa4
Waiting for encrypted snapshot to complete...
Encrypted volume created: vol-0f5db8d4c0ab6a46b
Waiting for encrypted volume to become available...
{
  "AttachTime": "2025-02-25T00:21:55+00:00",
```

```
    "Device": "/dev/xvda",
    "InstanceId": "i-0888aabdfc4a1d3bc",
    "State": "detaching",
    "VolumeId": "vol-0467be32dfa110f3b"
}
```

Detached unencrypted volume: vol-0467be32dfa110f3b

```
{
    "AttachTime": "2025-02-25T01:43:40.838000+00:00",
    "Device": "/dev/xvda",
    "InstanceId": "i-0888aabdfc4a1d3bc",
    "State": "attaching",
    "VolumeId": "vol-0f5db8d4c0ab6a46b"
}
```

Encrypted volume vol-0f5db8d4c0ab6a46b attached to instance i-0888aabdfc4a1d3bc.

```
{
    "StartingInstances": [
        {
            "CurrentState": {
                "Code": 0,
                "Name": "pending"
            },
            "InstanceId": "i-0888aabdfc4a1d3bc",
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            }
        }
    ]
}
```

Starting instance i-0888aabdfc4a1d3bc...

Instance i-0888aabdfc4ald3bc is running.

```
—(kali㉿kali)-[~/EBSVolencryption]      L$      ssh      -i
"EBSvolkeypair.pem"      ec2-user@ec2-3-90-11-29.compute-
1.amazonaws.com  The authenticity of host 'ec2-3-90-11-
29.compute-1.amazonaws.com (3.90.11.29)' can't be established.
ED25519      key      fingerprint      is
SHA256:98dXF8+GEkUgNphbjRAJF6A56eoB3xqSPaie6rHbRs. This host
key is known by the following other names/addresses:
~/.ssh/known_hosts:6: [hashed name] Are you sure you want to
continue connecting (yes/no/[fingerprint])? yes Warning:
Permanently added 'ec2-3-90-11-29.compute-1.amazonaws.com'
(ED25519) to the list of known hosts. , #_ ~\_ #####_ Amazon
Linux 2023 ~ ~ \_#####\ ~ ~ \###| ~ ~ \#/ _____
https://aws.amazon.com/linux/amazon-linux-2023 ~ ~ V~' '-> ~~~ /
~~. _ _/ _/ _/ _/m/' Last login: Tue Feb 25 01:02:34 2025 from
90.251.127.120 [ec2-user@ip-172-31-87-99 ~]$ whoami ec2-user
[ec2-user@ip-172-31-87-99 ~]$ df -h Filesystem Size Used Avail
Use% Mounted on devtmpfs 4.0M 0 4.0M 0% /dev tmpfs 475M 0 475M
0% /dev/shm tmpfs 190M 448K 190M 1% /run /dev/xvda1 8.0G 1.6G
6.4G 21% / tmpfs 475M 0 475M 0% /tmp /dev/xvda128 10M 1.3M 8.7M
13% /boot/efi tmpfs 95M 0 95M 0% /run/user/1000 [ec2-user@ip-
172-31-87-99 ~]$ lsblk NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
xvda 202:0 0 8G 0 disk └─xvda1 202:1 0 8G 0 part / └─xvda127
259:0 0 1M 0 part └─xvda128 259:1 0 10M 0 part /boot/efi [ec2-
user@ip-172-31-87-99 ~]$ exit logout Connection to ec2-3-90-11-
29.compute-1.amazonaws.com closed. ┌─(kali㉿kali)-
[~/EBSVolencryption] L$ aws ec2 describe-volumes --filters
Name=attachment.instance-id,Values=i-0888aabdfc4ald3bc --query
"Volumes[*].{ID:VolumeId,Encrypted:Encrypted}" --region us-
east-1 --output table -----
| DescribeVolumes | +-----+-----+ |
Encrypted | ID | +-----+-----+ | True
| vol-0f5db8d4c0ab6a46b | +-----+-----+
--+
```