**KMS Encrypted EBS Volume: Attach to another instance and check the data**

Amazon EC2 sends a Create Grant request to AWS KMS so that it can decrypt the data key.

**EBS encryption works**

When you create an encrypted EBS volume and attach it to a supported instance type, the following types of data are encrypted:

• Data at rest inside the volume

• All data moving between the volume and the instance

• All snapshots created from the volume

• All volumes created from those snapshots

**Re-attaching an encrypted EC2 volume to a new instance**

• EC2 instance is running with a secondary EBS volume attached. This volume should hold persistent data.

• EC2 instance needs to be destroyed and recreated. Maybe I've changed the VPC or maybe the AMI I am using.

• Automation detaches the volume, terminates the instance, creates a new instance, then reattaches the volume

Connect to the New Instance:

• Connect to the instance where you attached the EBS volume using SSH or EC2 Instance Connect. Mount the EBS Volume on the Instance:

• available disk to find the newly attached volume:



**Detach:**

Detach the volume from the instance through the AWS Management Console or AWS CLI



**Modify the script to take inputs such as the starting number and ending number, ensuring the operation is performed within this range.**

**Create a Bash Script**:

This script will accept two inputs: a starting number and an ending number, then perform a sample operation within this range

usage() {

echo "Usage: $0 <start\_number> <end\_number>"

exit 1

}

if [ "$#" -ne 2 ]; then

usage

fi

START\_NUMBER=$1END\_NUMBER=$2

if ! [[ "$START\_NUMBER" =~ ^-?[0-9]+$ ]] || ! [[ "$END\_NUMBER" =~ ^-?[0-9]+$ ]]; then

echo "Error: Both start\_number and end\_number should be integers."

usage

fi

if [ "$START\_NUMBER" -gt "$END\_NUMBER" ]; then

echo "Error: start\_number should be less than or equal to end\_number."

usage

fi

for (( i = START\_NUMBER; i <= END\_NUMBER; i++ )); do

echo "Processing number: $i"

**Objective: The script should identify services with high CPU and memory utilization, capture these values, and send an email with the details to your Gmail account. Additionally, it should execute every minute. Details: CPU and Memory Monitoring: The script should track services with high CPU and memory usage, logging the CPU% and MEM%. Email Notification: Send a report to your Gmail with the captured data. Scheduled Execution: The script should run every minute.**

GMAIL\_USER='your-email@gmail.com'

GMAIL\_PASSWORD='your-password'

TO\_EMAIL='your-email@gmail.com'

EMAIL\_SUBJECT='High CPU and Memory Usage Report'

CPU\_THRESHOLD=50

MEM\_THRESHOLD=50

TMP\_FILE=$(mktemp)

get\_high\_usage\_services() {

ps aux --sort=-%cup | awk -v cpu\_thresh="$CPU\_THRESHOLD" -v mem\_thresh="$MEM\_THRESHOLD" '

NR>1 {

cpu=$3; mem=$4; cmd=$11;

if (cpu > cpu\_thresh || mem > mem\_thresh) {

print sprintf("PID: %s, USER: %s, CPU: %.2f%%, MEM: %.2f%%, COMMAND: %s", $2, $1, cpu, mem, cmd)

}

}' > "$TMP\_FILE"

}

send\_email() {

local body=$(cat "$TMP\_FILE")

echo -e "Subject:${EMAIL\_SUBJECT}\n\n${body}" | \

ssmtp -C /etc/ssmtp/ssmtp.conf "$TO\_EMAIL"

}

get\_high\_usage\_services

if [ -s "$TMP\_FILE" ]; then

send\_email

fi

rm "$TMP\_FILE"

**Objective: If memory usage exceeds 95%, the script should create a 5GB empty file, activate swap on it, and add it to the memory. Each step should be validated, including the final command using echo $?. Details: Memory Threshold: When memory usage goes beyond 95%, trigger the creation of a 5GB empty file. Swap Activation: Perform swapon and add the file to swap memory. Validation: Validate each step, especially for long-running commands. Ensure the final command’s success using echo** **$?. Command Validation: Include validation after each command, especially for those that take longer to execute.**

check\_success() {

local status=$?

if [ $status -ne 0 ]; then

echo "Command failed with exit code $status"

exit $status

fi

}

memory\_usage=$(free | awk '/Mem/{printf("%.0f"), $3/$2\*100}')

echo "Current memory usage: $memory\_usage%"

if [ -z "$memory\_usage" ]; then

echo "Failed to retrieve memory usage."

exit 1

fi

if [ "$memory\_usage" -ge 95 ]; then

echo "Memory usage exceeds 95%. Initiating swap file creation."

dd if=/dev/zero of=/swapfile bs=1M count=5120

check\_success

echo "5GB empty file created."

chmod 600 /swapfile

check\_success

echo "Permissions set for swap file."

mkswap /swapfile

check\_success

echo "Swap area created on /swapfile."

swapon /swapfile

check\_success

echo "Swap file enabled."

swapon --show | grep '/swapfile'

if [ $? -eq 0 ]; then

echo "Swap file is successfully added."

else

echo "Swap file not added."

exit 1

fi

else

echo "Memory usage is below 95%. No action needed."

fi