

FREE RESOURCE / UPDATED 19 OCTOBER 2026

Ten checks. A clearer AWS bill.

The checks I run first, with the CLI commands that find the waste and practical next steps.
Read-only discovery commands are separated from changes.

01 Delete EBS volumes nothing is attached to

A volume in the `available` state is billed for every GB, every hour, with no instance using it. They pile up when instances are terminated without deleting their volumes.

Find it

```
aws ec2 describe-volumes \
  --filters Name=status,Values=available \
  --query 'Volumes[][VolumeId,Size,VolumeType,CreateTime]' \
  --output table
```

Fix it

Snapshot anything you are unsure about, then delete the volume. Snapshots cost less per GB than the volume itself.

02 Move gp2 volumes to gp3

gp3 is cheaper per GB than gp2 and gives a baseline of 3,000 IOPS regardless of size.

Find it

```
aws ec2 describe-volumes \
  --filters Name=volume-type,Values=gp2 \
  --query 'Volumes[][VolumeId,Size]' \
  --output table
```

Fix it

`aws ec2 modify-volume --volume-id vol-0123 --volume-type gp3` changes the type in place while the volume stays attached. Check the volume needs no more than gp3's baseline throughput first.

Source: [AWS: gp3 is up to 20% lower price per GB than gp2](https://aws.amazon.com/ebs/general-purpose/) (<https://aws.amazon.com/ebs/general-purpose/>)

03 Prune old EBS snapshots

Snapshots taken for a one-off change, or by a backup job nobody reviews, stay billed until someone deletes them.

Find it

```
aws ec2 describe-snapshots --owner-ids self \
  --query 'sort_by(Snapshots,&StartTime)[].[SnapshotId,VolumeSize,StartTime,Description]' \
  --output table
```

Fix it

Keep what your backup policy needs (and anything an AMI uses), delete the rest, and set a Data Lifecycle Manager policy so snapshots expire on their own.

04 Release idle Elastic IPs and unneeded public IPv4

AWS charges for every public IPv4 address, in use or idle. An Elastic IP that is not associated with anything is pure cost.

Find it

```
aws ec2 describe-addresses \
  --query 'Addresses[?AssociationId==null].[PublicIp,AllocationId]' \
  --output table
```

Fix it

`aws ec2 release-address --allocation-id eipalloc-0123` for addresses you no longer need. For private workloads, turn off auto-assigned public IPs on the subnet.

Source: [AWS VPC pricing: \\$0.005 per public IPv4 address per hour](https://aws.amazon.com/vpc/pricing/) (<https://aws.amazon.com/vpc/pricing/>)

05 Route S3 and DynamoDB traffic around the NAT Gateway

A NAT Gateway bills per hour and per GB it processes. Traffic from private subnets to S3 or DynamoDB is often the bulk of that, and gateway endpoints carry it for free.

Find it

```
aws ec2 describe-nat-gateways \
  --query 'NatGateways[][NatGatewayId,VpcId,State]' \
  --output table

# Then in Cost Explorer, group by Usage type and look for NatGateway-Bytes.
```

Fix it

`aws ec2 create-vpc-endpoint --vpc-id vpc-0123 --service-name com.amazonaws.<region>.s3 --route-table-ids rtb-0123` adds an S3 gateway endpoint to the private route tables. Repeat with `dynamodb` if you use it.

Source: [AWS VPC pricing: NAT Gateway per hour and per GB; no charge for gateway endpoints](https://aws.amazon.com/vpc/pricing/) (<https://aws.amazon.com/vpc/pricing/>)

06 Set a retention period on CloudWatch log groups

New log groups default to never expire, so every log line is stored forever.

Find it

```
aws logs describe-log-groups \
  --query 'logGroups[?!retentionInDays].[logGroupName,storedBytes]' \
  --output table
```

Fix it

`aws logs put-retention-policy --log-group-name <name> --retention-in-days 30` (or whatever your compliance needs say).

07 Give S3 buckets a lifecycle policy

Logs, exports and old build artifacts sit in S3 Standard indefinitely unless a rule moves or expires them.

Find it

```
for b in $(aws s3api list-buckets --query 'Buckets[].Name' --output text); do
  aws s3api get-bucket-lifecycle-configuration --bucket "$b" >/dev/null 2>&1 \
  || echo "no lifecycle rule: $b"
done
```

Fix it

Add rules that expire temporary data and move rarely read data to a colder storage class. Also expire incomplete multipart uploads, which are billed but invisible in the console.

08 Rightsize idle and oversized EC2 instances

Instances sized for a launch-day peak keep that price long after the peak is gone.

Find it

```
# Opt in to Compute Optimizer once, then after it has collected data:
aws compute-optimizer get-ec2-instance-recommendations \
  --query 'instanceRecommendations[][instanceArn,finding,currentInstanceType]' \
  --output table
```

Fix it

Downsize instances marked over-provisioned one at a time, watching latency and memory. Where your software runs on ARM, Graviton types are usually cheaper for the same work.

Source: [AWS: Graviton instances cost up to 20% less than comparable x86 instances](https://aws.amazon.com/ec2/graviton/) (<https://aws.amazon.com/ec2/graviton/>)

09 Stop non-production environments outside working hours

Dev and staging usually run all 168 hours of the week while the team uses them for about 55.

Find it

```
aws ec2 describe-instances \
  --filters Name=instance-state-name,Values=running \
  --query 'Reservations[].Instances[][InstanceId,InstanceType,Tags[?Key==`Environment`][0].Value]' \
  --output table
```

Fix it

Tag non-production resources and stop them on a schedule. My open-source `aws-ec2-reaper` does this for EC2, RDS and ECS with Lambda and Terraform.

10 Commit your steady baseline with a Savings Plan

Compute that runs around the clock is cheapest under a commitment, and AWS will tell you how much to commit based on your own usage.

Find it

```
aws ce get-savings-plans-purchase-recommendation \
  --savings-plans-type COMPUTE_SP \
  --term-in-years ONE_YEAR \
  --payment-option NO_UPFRONT \
  --lookback-period-in-days THIRTY_DAYS
```

Fix it

Do the cleanup above first, then commit to the steady floor of what remains, not your peak. Turn on Cost Anomaly Detection so a new spike is flagged within a day.

Source: [AWS: Savings Plans save up to 72% compared to On-Demand](https://aws.amazon.com/savingsplans/) (<https://aws.amazon.com/savingsplans/>)