AWS Cost Optimization Template for Recommendations and Diagnosis:

Expenses on the AWS can be divided into the following categories

	Cost type:	**Examples:**
1	Compute Costs	EC2 instances (billed per second/hour) Lambda invocations (per request + execution time) ECS/EKS tasks (when using EC2 or Fargate) Lightsail VMs (flat monthly pricing)
2	Storage Costs	Amazon S3 (based on storage class and GB used) EBS volumes and snapshots EFS (file system) and throughput provisioning Glacier/Deep Archive (archival storage)
3	Data Transfer Costs	Outbound data to the internet Inter-AZ or inter-region data transfer NAT Gateway data processing Inbound data (mostly free)
4	Networking and Security	Elastic Load Balancer (ELB) usage VPC NAT/Transit Gateway hourly & data charges AWS WAF and Shield Advanced PrivateLink endpoints
5	API calls, hosted zones and other services	Route 53 (hosted zones + DNS queries) CloudWatch logs, metrics, alarms AWS Config and Systems Manager Step Functions, EventBridge, etc.
6	AMIs and third party licences	Windows Server / SQL Server license fees Marketplace AMIs with commercial software BYOL (Bring Your Own License) models

Pillars of AWS Cost Optimization:

1. Right size
2. Increase elasticity
3. Leverage the right pricing model
4. Optimize storage
5. Measure, monitor, and improve

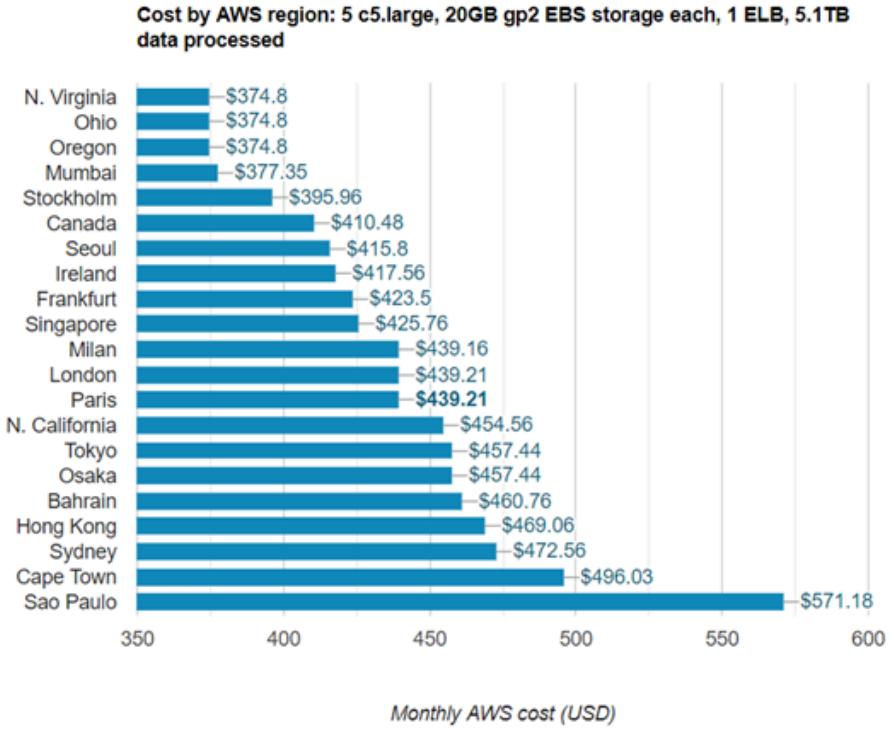
Cost optimization tools:

1. AWS Cost Explorer
2. AWS Budgets
3. AWS Pricing Calculator

1. Compute Costs:

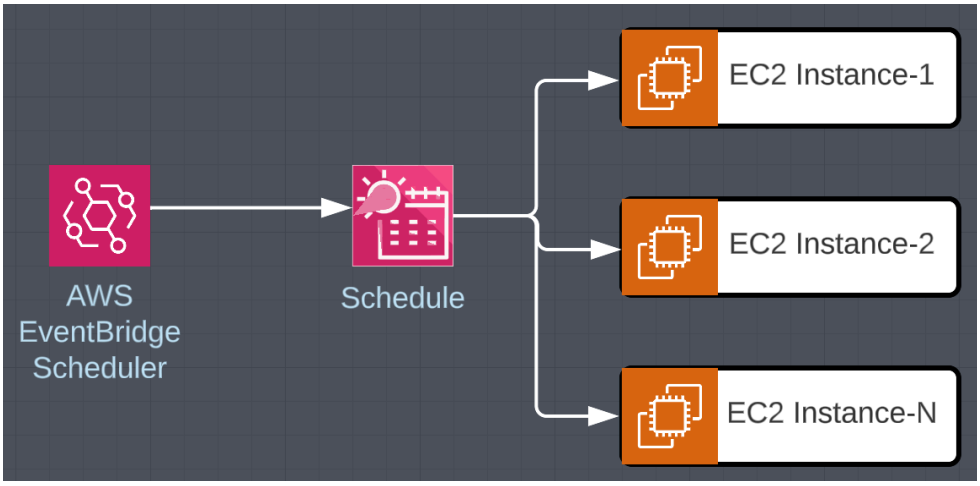
a. Choose the Appropriate AWS Region:

Compute pricing heavily depends on the AWS region



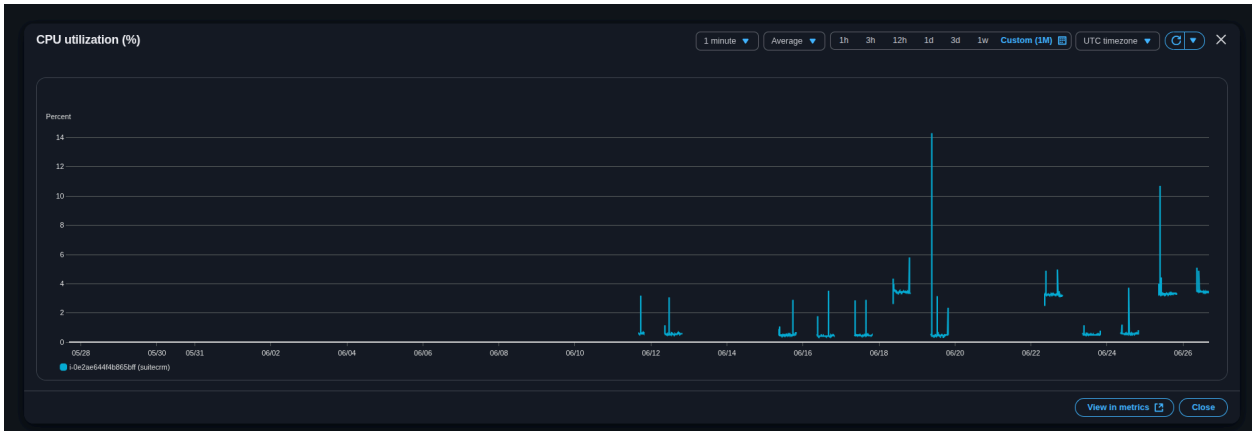
Analyze AWS Region pricing and select regions with lower costs for your workloads, provided there are no data residency or latency constraints. Best case scenario is finding the sweet spot between latency, cost and compute according to the region that the service is based on.

b. Create Schedules to Turn Off Unused Instances:



Use tools or automation (like **tagging** with cron schedules) to stop non-essential EC2 and RDS instances during off-hours, reducing unnecessary compute spend. Additionally, Use Auto Scaling to match demand and schedule on/off times for development or testing environments to avoid paying for idle resources.

c. Identify and Downsize Underutilized EC2 Instances (Right sizing):

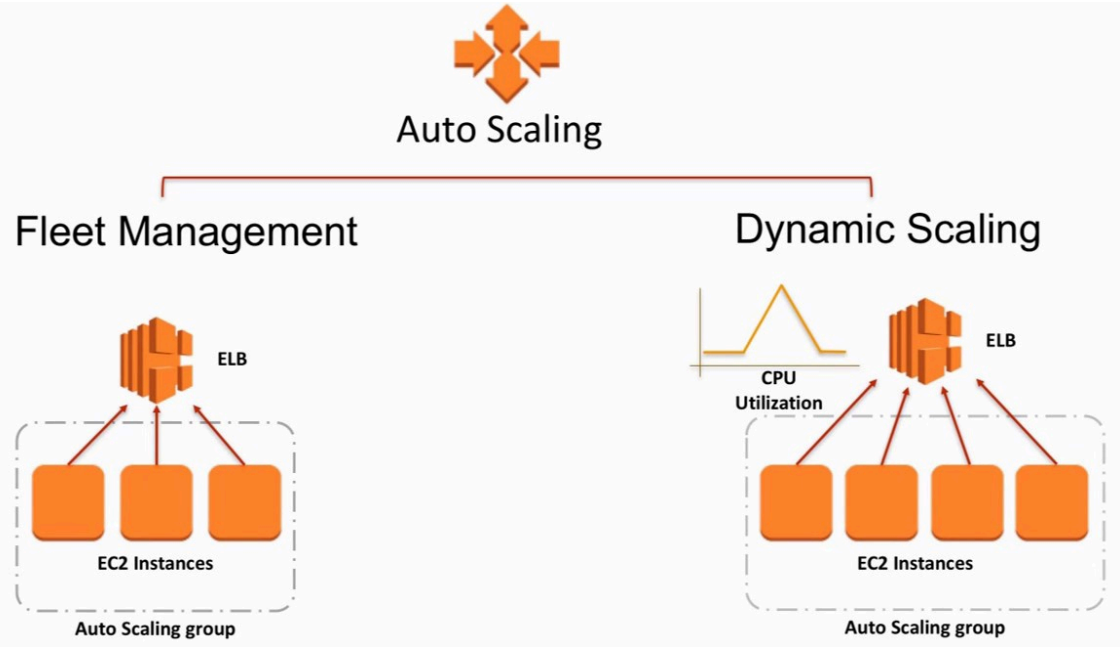


Use *AWS Cost Explorer* and *AWS Systems Manager* to monitor CPU and memory usage, and right-size or terminate underutilized instances based on recommendations.

d. Leverage EC2 Spot Instances:

Amazon EC2 spot instances are an important option for reducing AWS costs. Spot instances provide savings of **up to 90%** compared to normal on-demand prices. Spot instances are Amazon's way to sell off spare EC2 capacity. They can significantly reduce Amazon EC2 costs by allowing you to request the same EC2 instances, when they are in low demand, at a significantly discounted price.

e. **Optimize Auto Scaling Group (ASG) Configuration:**



ASG Configuration by setting up dynamic scaling policies that automatically add or remove instances based on real-time demand, such as CPU utilization or network traffic. Combine scheduled actions for predictable workload spikes and use a mix of On-Demand and Spot Instances to balance cost and availability. Regularly monitor and fine-tune your scaling policies using CloudWatch metrics to prevent over-provisioning and minimize idle resources, ensuring your infrastructure remains both cost-effective and responsive to changing workloads.

f. **Sell or Use Underutilized Reserved Instances:**
If you have excess Reserved Instances, sell them on the AWS Reserved Instance Marketplace or reallocate them to other workloads. But only ***Convertible Reserved Instance*** are resellable.

g. **Leverage Compute Savings Plans:**
For a long term running workload, commit to Savings Plans for predictable workloads to receive significant discounts over On-Demand rates. Compute Savings Plans provide the most flexibility and help to reduce your costs by **up to 66%.** These plans automatically apply to EC2 instance usage regardless of instance family, size, AZ, Region, OS or tenancy, and also apply to Fargate or Lambda usage. For example, with Compute Savings Plans, you can change from C4 to M5 instances, shift a workload from EU (Ireland) to EU (London), or move a workload from EC2 to Fargate or Lambda at any time and automatically continue to pay the Savings Plans price.

h. **Implement Auto Scaling and Schedule On/Off Times:**
Use Auto Scaling to match demand and schedule on/off times for development or testing environments to avoid paying for idle resources.

i. **Use AWS Trusted Advisor and Compute Optimizer:**
Regularly review recommendations for rightsizing and identifying idle resources.

j. **Create Separate Accounts and Use AWS Organizations:**
Separate production and development environments and consolidate billing for better cost tracking and discount sharing.

k. **Focus on optimizing workload rather than increasing the size of Infrastructure:**
Instead of simply scaling up hardware or adding more servers to handle growing demands, prioritize making your existing workloads more efficient. This approach can lead to significant cost savings and better resource utilization.

L. **Separate workloads into layers:**
Instead of running an entire application or workload as a monolithic entity on a single set of resources, it's far more effective to break it down into distinct layers. Though it may seem costly to use multiple services, each layer can be designed, optimized, scaled, and managed independently which leads to greater efficiency, easier maintenance, and better cost control.

2. Storage Costs:

a. **Monitor and Delete Unused EBS Volumes:**
Regularly audit and remove unattached EBS volumes to avoid ongoing storage charges.

b. **Identify and Delete Orphaned Snapshots:**
Use tagging and automation (e.g., Lambda functions) to find and delete snapshots not associated with active volumes or instances.

c. **Move Infrequently Accessed Data to Cheaper Tiers:**
Use S3 lifecycle policies or intelligent tiering to automatically transition data to lower-cost storage classes like S3 Glacier or Deep Archive.

d. **Eliminate Zombie Resources:**
Routinely identify and remove unused storage resources such as unattached volumes, old AMIs, and outdated backups.

e. **Implement Tagging Policies:**
Tag resources for better visibility and cost allocation, making it easier to identify and manage unused or underutilized storage.

f. **Automate Data Lifecycle Management:**
Set up automated policies to archive or delete data after a certain period.

g. **Use AWS Cost and Usage Reports (CUR):**
Generate detailed reports to analyze storage usage and costs for optimization.

h. **Leverage Compression and Deduplication:**
Reduce storage footprint by compressing and deduplicating data before storing.

i. **Regularly Review Storage Usage:**
Audit storage consumption and delete obsolete or duplicate data.

j. **Implement Tools for Cost Visibility:**
Use AWS-native or third-party tools to gain insights into storage cost drivers.

3. Data Transfer Costs:

a. **Keep Transfers Within the Same Region or Availability Zone:**
Minimize cross-region and cross-AZ transfers, which incur higher costs.

b. Use AWS CloudFront and Caching:
Deploy CloudFront to cache content closer to users, reducing outbound data transfer from origin servers and leveraging free outbound transfers from edge locations.

c. **Leverage Private Connectivity (VPC Endpoints, Direct Connect):**
Use private IPs, VPC endpoints, or Direct Connect to reduce data transfer charges compared to public internet routes.

d. **Optimize High-Bandwidth Workloads:**
Place high-bandwidth workloads in public subnets to bypass NAT gateway costs when appropriate, or use NAT instances for more control over bandwidth and cost.

e. **Monitor and Minimize Outbound Data:**
Regularly review data transfer patterns and limit unnecessary outbound transfers, as these are typically the most expensive.

f. **Choose Regions Based on Data Transfer Patterns:**
Factor in region-specific data transfer pricing when architecting your solution.

g. Schedule Large Transfers Strategically:
Time large transfers for off-peak hours to optimize usage and potentially take advantage of lower rates.

h. **Implement Data Compression:**
Compress data before transfer to reduce volume and costs.

i. **Use AWS Budgets for Data Transfer:**
Set up budgets and alerts to monitor and control data transfer spending.

j. **Use AWS Trusted Advisor:**
Identify opportunities to optimize data transfer costs with Trusted Advisor checks.

4. Networking and Security Costs:

a. **Delete Idle Load Balancers:**
Remove unused load balancers to stop incurring hourly charges.

b. **Optimize Bandwidth Use:**
Use direct connections and optimize network routes to avoid unnecessary data transfer and bandwidth costs.

c. **Disable Unessential Services:**
Turn off or remove unnecessary network and security services to reduce costs.

d. **Implement Tagging for Network Resources:**
Use consistent tagging for better tracking and cost allocation.

e. **Consolidate and Centralize Network Appliances:**
Reduce the number of NAT gateways, VPNs, and firewalls where possible.

f. **Use VPC Endpoints Over NAT Gateways:**
For AWS service access, VPC endpoints are often more cost-effective than NAT gateways.

g. **Monitor Network Traffic:**
Regularly analyze network flows to identify and eliminate unnecessary traffic.

h. **Leverage AWS Organizations for Centralized Management:**
Manage network and security costs across accounts with consolidated billing.

i. **Review Security Group and NACL Rules:**
Remove redundant or overly permissive rules to reduce potential attack surface and associated costs.

j. **Use AWS Trusted Advisor:**
Get recommendations for optimizing networking and security configurations.

5. API Calls, Hosted Zones, and Other Service Costs:

a. Monitor and Reduce API Calls:
Audit API usage to identify and eliminate unnecessary or redundant calls.

b. **Optimize Hosted Zone Configurations:**
Consolidate DNS records and hosted zones where possible to reduce Route 53 costs.

c. **Batch and Cache API Requests:**
Reduce the frequency and volume of API calls by batching requests and caching responses.

d. Implement Usage Plans and Quotas:
Control API usage with quotas to prevent cost overruns.

e. **Leverage AWS Cost and Usage Reports:**
Use CUR to track service usage and identify high-cost API or service patterns.

f. **Tag API and DNS Resources:**
Tag resources for better cost allocation and accountability.

g. **Use AWS Budgets for Monitoring:**
Set up budgets for API and other service usage to receive alerts on spending.

h. **Disable Unused Services:**
Regularly review and disable services that are no longer needed.

i. **Review Service Pricing by Region:**
Some services (like Route 53) may have different costs in different regions—choose wisely.

j. **Use AWS Trusted Advisor:**
Get recommendations for API and service usage optimization.

6. AMIs and Third-Party Licensing Costs:

a. **Delete Unused AMIs and Associated Snapshots:**
Regularly audit and remove obsolete AMIs and their snapshots to avoid storage charges.

b. **Track and Manage Third-Party Licenses:**
Maintain an inventory of third-party licenses to avoid over-provisioning and unnecessary renewals.

c. **Sell Unused Reserved Instances:**
Use the AWS Reserved Instance Marketplace to sell excess RIs and recoup some costs.

d. **Consolidate Licenses Across Accounts:**
Use consolidated billing and AWS Organizations to manage licenses centrally and benefit from volume discounts.

e. **Leverage Open-Source Alternatives:**
Where feasible, replace commercial software with open-source solutions to reduce licensing costs.

f. **Implement Tagging for AMIs and Licenses:**
Tag AMIs and licensed resources for better tracking and cost allocation.

g. Monitor License Utilization:
Use AWS and third-party tools to track actual usage and optimize license allocation.

h. **Review Marketplace Subscriptions:**
Regularly review and cancel unused marketplace subscriptions.

i. **Use AWS Budgets for License Costs:**
Set up budgets and alerts for third-party license spending.

j. **Leverage Trusted Advisor for Recommendations:**
Use Trusted Advisor to identify underutilized AMIs and licensed resources.

References:

https://www.cloudzero.com/blog/aws-cost-management-best-practices/

https://www.digitalocean.com/resources/articles/aws-cost-optimization#8-consider-ec2-spot-instances-for-flexible-workloads

https://pages.awscloud.com/rs/112-TZM-766/images/ebook-aws-cloud-financial-management-guide-032023.pdf

https://www.netapp.com/blog/3-ways-to-save-big-and-10-price-variations-to-know-aws-cvo-blg/