10 BEST PRACTICES FOR AWS COST OPTIMIZATION
Gartner’s key finding on *Cloud Service Expense Management* (CSEM) says,

> Organizations utilizing IaaS without CSEM processes will almost certainly have unnecessary spending.

**Gartner**

In addition to this key finding, IDC predicts,

> The worldwide public cloud services spending will double to more than **$141 billion** by 2019.

**IDC**

But, do we actually need to prove these implications, right?

Consider this eBook as your ultimate guide for AWS Cost Optimization.
The following 10 best practices would help you cut through all the unnecessary spends in your AWS cloud:

1. **“Reserved Instances” is your first big step to AWS Cost Optimization**

Reserved Instances (RIs) are one of the most perplexing yet a great option AWS offers its users, to save big on their resources. Reserved Instances are nothing but similar to coupons or a long-term commitment to AWS where you pay a low, one-time fee with a good discount on the hourly charge for your running instances. To reduce your usage costs, AWS pricing framework automatically applies the rates for Reserved Instances.

The most evident approach to control your computing cost is to buy reserved EC2 instances for a time period of 1 year or 3 years. With Amazon's three different payment options available for reserved instances, you can either pay all upfront, partially upfront, or no upfront. Customers can realize cost savings up to 70% on reserved instances as compared to on-demand instances.

What makes RIs exceptional and value for money is that they provide organizations with loads of flexibility in terms of repurposing, reselling, and payment options. But, with perks, comes challenges in RIs. If you purchase an instance but not use it at all or is kept turned off for long, you are still paying a fixed charge for reservation regardless of usage.

**RIs can easily turn into an investment disaster if not supervised and planned properly.**

To begin with, it is not an easy task to analyze and envisage one or three years of utilization of a particular instance when an organization has been on AWS for the same time period or even less. Secondly, organizations that are pulled in to the pay-as-you-go cloud model are cagey about capital expenses that drag them back to long-term contracts and sunk expenses. Businesses can also have a tough time discovering extra reservation capacity of certain instance types across the marketplace. Enterprises, in any scenario, might find this a muddled and exorbitant method to execute.

Organizations can make the most out of reserved instances by implementing the following best practices:

- One should purchase standard reserved capacity to meet the basic or “average sustained usage” for the minimum number of instances required to keep the application(s) running or, the existing instances that are constantly running.

- First, anticipate and focus on what can get you maximum savings with speedy ROIs. This brings down the potential impact of unused or underutilized resources in future. Applications with extremely stable usage trends & patterns are considered to be the best use cases for reserved instances.

- For large-scale organizations, using a single centralized account to purchase and manage reserved instances across the entire organization prevents RI wastage, thus eliminating unnecessary charges. A unified hub for RIs allows resources that are not utilized by one team/application to be taken up by other projects/teams internally. Having multiple accounts for instance purchases, dedicated to every small-big team does not offer this flexibility of moving reservations between these accounts as and when required.
Proper Allocation & Tracking of RIs

Several EC2 Instances remain underutilized due to lack of proper allocation and planning.

It is always better to experiment with RIs on stable applications. You may drive better value by selecting smaller instance sizes and strategically using those instances without any upfront fees.

To figure out whether your RIs are keeping up with average sustained utilization or costing you big, leverage cost management & governance tools like Centilytics to review your usage history and billing detail with hourly-level granularity. Centilytics’ RI Planner gives recommendations on reserved instance purchases based on your historical data, which can be particularly useful if you would prefer not to go over years of data across numerous applications.

Repurpose Unused RIs

Since RIs are all about estimation & planning to make your investment worthwhile, despite their sufficient utilization, your organization may still be left with some idle reservations. These could be repurposed and put to good use elsewhere.

What you can do is reallocate them to totally new projects and applications where they fit in. Else, you could repurpose them to deal with existing workloads, which are using expensive on-demand instances in your cloud. Or, if repurposing doesn’t work out for you; if you have over reserved instance capacity than required or; your requirements change over time, unused or underutilized RIs can be sold on AWS Marketplace. A quick tip for aged, unpopular instance types; they might be hard to sell as the volume on this market varies.

Different Payment Options for RIs: Play It Right

AWS basically give you three different methods for pricing: All Upfront where you pay the entire amount in advanced (at one go); No Upfront where you pay a fixed amount of money on a monthly basis with no advanced payment or; Partial Upfront which is paying some money up front while the rest is paid in monthly or periodic installments. It also allows you to reserve resources so that you can start them whenever required.

As per the AWS’ Reserved Instance pricing model, if a reserved resource isn’t in use for a while or has been not launched yet, you are charged the fixed price for your reserved capacity regardless of whether being used or not. This puts you into a great loss and can even cost you your business.

If you have no cash flow limitations, opt for the All Upfront option for most of your RI purchases. This will get you multiple discounts.

Nonetheless, if we calculate the discounted rates between the All Upfront and Partial Upfront payment options the difference comes out to be very small. Depending on your financial state, you can switch between All Upfront and Partial Upfront payments without letting it affect the discounted rate much. On the contrary, the discounted rate is very low in the case of No Upfront option.

EC2 and RDS are not the only resources in AWS that can be reserved. You can also save money on other services i.e. Redshift and ElastiCache by purchasing reserved capacities for them.
Orphaned Snapshots

Snapshots are a point-in-time backup copy of an EBS volume, stored in Amazon S3. These snapshots are not only used as recovery points in case of any data loss or disaster but, can also be used as the starting point of new Amazon EBS volumes. The very snapshot can be utilized to instantiate as many volumes as required. You can make copies of these snapshots across various AWS regions for geographical expansion, disaster recovery and data center migrations.

But, your EBS snapshots remain on S3 and keep on racking up monthly charges if not monitored regularly. Individual Snapshots do not account for much spikes in your bills as multiple snapshots combinedly do.

An intensifying element that encourages this issue is that customers can configure settings that automatically create snapshots on a regular basis, thereby neglecting to schedule the deletion of aged, unwanted snapshots.

Snapshots that have no attached volumes to it are good contenders for deletion. Make it a habit or a part of your Cloud housekeeping routine but, you regularly need to clean-up the orphaned snapshots until and unless they are required for creating new EBS volumes in future.

This is way easier if every volume is tagged the moment it is created containing all the essential metadata like who created it and the purpose of creating the same.

When an EC2 instance terminates, EBS root device volumes are automatically erased by default. However, any additional EBS volumes that you attach at launch, or any EBS volumes that you attach to an existing instance persist even after the instance terminates.

When you delete a snapshot, only the data exclusive to that snapshot is removed. Deleting previous snapshots of a volume does not affect your ability to restore volumes from later snapshots of that volume.

If you make periodic snapshots of a volume, the snapshots are incremental so that only the blocks on the device that have changed since your last snapshot are saved in the new snapshot. Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot in order to restore the volume. - AWS

Try to retain a limited number of snapshots per instance as the EBS volumes are recovered from the latest snapshot(s) saved.
3 Rightsize your Resources/Instances

Comprehend rightsizing as modifying or considering to modify your existing AWS cloud infrastructure to balance and meet the exact demand of your business. Amongst a wide range of services that AWS allows, to rightsize, Elastic Compute Cloud (EC2) and Relational Database Service (RDS) are most commonly referred for they incur up to 50% of the total monthly costs.

Rightsizing helps you identify underutilized resources and settle on an effective decision of either vertically adjusting those resources allotted to the instance or releasing them. The two things you can most effectively rightsize are instance size and volume types. If an instance in your AWS account is being underutilized, you can switch down to a size or two. Another two elements that you can rightsize are the disk storage type and performance.

Your compute instances can be underutilized for two main reasons:

a. Either a workload is no longer resource-intensive (memory or CPU-intensive) as it was earlier,
b. or (b) the compute instance has accidentally or purposely been provisioned more than required.

Understanding your CPU and Memory Utilization is the kick-start point for you.

If your infrastructure is not a part of auto-scaling groups, you need to have continuous monitoring and automated alerts in place to flag underutilized resources. Leverage SaaS tools like Centilytics that gives you actionable insights and recommendations on how & what you should rightsize in your cloud with real-time performance data and analytics.

However, before you get ahead in the “rightsizing game”, it is quite important to get a reality check on how your applications utilize resources (CPU and Memory). Choose the right instance type for your workloads and try to keep its utilization above 80% to get the most out of your application running on the AWS cloud.

To stay on the right track, continuously assessing utilization and efficiency of your resources and, fine-tuning them accordingly is recommended, in order to make data-driven decisions.
4 Cleanup Unnecessary EBS Volumes

Volumes act like unpolished, unformatted block devices, where user-provided device names and a block device interface are specified. Users leveraging Amazon EBS volumes can make a record system on top of them or, utilize the same in any other way a block device like a hard drive is used.

The Elastic Block Store volumes created in AWS accounts keep piling up, adding thousands of dollars to your monthly bill, no matter whether being utilized or not. Every time a new instance is launched, an EBS volume usually comes appended to it as the local block storage for the workload or the application. However, launching an instance via the AWS Console gives you the option in the settings to delete the attached EBS volume upon the decommissioning of the instance. If that setting is not enabled, the volume remains in your account even after its associated instance gets terminated.

These unattached block storage assets account for more than 50% of your AWS costs.

EBS volumes other than root volumes that are unattached to an EC2 instance or have low I/O processes should be erased. Organizing and deleting orphaned or unattached Elastic Block Store volumes on a regular basis will help you keep spiky surprises away from your AWS bill and restrict access to any confidential data stored in these volumes. You, therefore, must constantly check for unattached EBS volumes in your AWS accounts.

5 Cut Down Your Data Transfer Costs

Data Transfer costs could be one reason for your tangled AWS bill, contributing upwards of 30% to the total cost. Data transfer costs are the charges incurred for transferring data across AWS cloud services in & out, to & from an AWS service. You are also charged for transferring data into one service from another AWS service, and charged for transferring data out of the service to another one.

Data transfer into EC2 is usually free but, the outbound transfer has its fees. This is something that gets many new AWS customers to fall into the “unawareness” trap. Generally, data transfer “OUT from Amazon EC2 to the Internet” incurs substantial costs. These costs can rapidly mount up and keep on adding to your month to month cloud bills.

Irrespective of different fees for different AWS services and data capacities, pulling across Regions is expensive. You probably need to narrow down the number of Regions that your data flows across.

Try architecting or re-architecting your frameworks in a way so that the data transfer across various AWS regions or availability zones, is minimum.

Re-hosted applications that are not configured or aligned with AWS features needs are more prone to incur high data transfer costs.

They should be re-architected to ensure that data transfers are completed through the cheapest route possible.
Since IP addresses alone cannot save you much, you should enable caching at your origin servers or S3 for CloudFront edge locations to accelerate delivery of your websites, APIs, video content or other web resources. Compression of static as well as dynamic content such as HTML, Javascript, JSON etc. will save costs on bandwidth.

Also, when you deploy production changes, do not forget to automate server-side compression and client-side caching post deployment, in your release automation cycle.

For example, consider the following pricing range table that shows how much it costs to transfer up to 40 TB of data OUT from EC2 to the Internet from each Region per month:

<table>
<thead>
<tr>
<th>AWS Regions</th>
<th>Cost of Transferring up to 40 TB of Data OUT from EC2 to the Internet (per month)</th>
</tr>
</thead>
<tbody>
<tr>
<td>South America (Sao Paulo)</td>
<td>$0.23 per GB</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>$0.085 per GB</td>
</tr>
<tr>
<td>US East (Ohio)</td>
<td>$0.085 per GB</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>$0.085 per GB</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>$0.085 per GB</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>$0.085 per GB</td>
</tr>
<tr>
<td>EU (Frankfurt)</td>
<td>$0.085 per GB</td>
</tr>
<tr>
<td>EU (London)</td>
<td>$0.085 per GB</td>
</tr>
<tr>
<td>AWS GovCloud (US)</td>
<td>$0.115 per GB</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>$0.122 per GB</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>$0.135 per GB</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>$0.135 per GB</td>
</tr>
</tbody>
</table>

* The above data transfer rates are as per the latest AWS Reduced Data Transfer Prices effective from December 1, 2014 till date.

There are particular AWS services which cost more to send and/or receive data as compared to the others taking capacity as an additional factor.

Analyze which services incur higher costs for transferring data to and from within the selected Region, and re-design in a manner that your data transfers do not happen along costly routes. Go for lower-costing alternatives that equally cater to your needs.

Cost spikes in your AWS bill is also due to transferring large volumes of data via public or elastic IP addresses. AWS customers usually neglect or forget the fact that compressing or caching their data before transferring can save quite an amount.
Optimize Storage Costs with S3, S3 Reduced Redundancy Storage (RRS), and Glacier

AWS offers three types of services for your object storage, S3 being the most well-known. S3 guarantees you to provide with 99.999999999% redundancy and durability across all your availability zones within a region. In case, if you have some static data objects that are noncritical or reproducible, and you are willing to compromise on reduced redundancy, you can go for S3 Reduced Redundancy Storage (RRS). Reduced Redundancy Storage (RRS) costs around 15-20% less than that of S3 and offers lower levels of redundancy than Amazon S3’s standard storage i.e. 99.99% durability.

At last, comes Glacier that offers services for archiving your objects, something like magnetic tape backups in traditional data centers. The brighter side of Glacier is that it is extremely cheap, about one cent i.e. $0.01 per gigabyte every month and upto 80% cheaper than S3. The reason for which you might question considering this storage option is that the retrieval time of the object is around four to five hours.

Discard Assets that are more of Liabilities

It is common to have idle infrastructure components in your AWS account(s) that are running somewhere in the cloud environment but are not being used up to their potential or not being used for any purpose at all. We call such assets, obsolete resources or more of a liability. These could be your outdated EC2 instances, an idle Relational Database Service (RDS) instance or idle Elastic Load Balancers (ELB) that were once utilized for a specific purpose and hasn’t been in use since then, nor were terminated. Instances can also become obsolete due to some failure during their launch process or script errors failing to release those instances.

Regardless of what is causing your assets to sleepwalk, AWS will anyhow charge for them as long as they keep running. To cut some or thousands of dollars from your cloud bill, assets-turned liabilities must be detached, assessed, and immediately discarded.

However, do not forget to take a snapshot (point-in-time backup copy) of the resource before terminating it. This will help you recover the asset if you need it again in future.

Another Side of Obsolete Resources

Over time, your instances keep degrading and become deprecated enough to increase your cloud bill(s). These instances are being used just for the sake of generosity that AWS is offering still them and are there in your cloud from the beginning. So, why not upgrade the first-generation instances to the next generation that AWS releases every few years like a clockwork. Switching to the current generation of instances where cost-per-compute performance, networking, storage, the ability to attach new types of EBS volumes and every other configuration you need is an enhanced, optimized one with hourly cost savings. Not to mention the additional functionality it offers like clustering, upgradation of prior generation instances is usually a win-win situation for AWS customers.

For example, upgrading your instance from a c3.xlarge to a c4.xlarge will slice costs by up to 45-50% while offering significantly faster processing than that of the previous one.
8  Auto Scaling

Your organization probably have been into a situation of busy periods where there is a heavy load on your application but the number of servers is low than what is required. Autoscaling groups are a great option for acquiring extra capacity as and when required during heavy traffic.

Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups, specify the minimum & a maximum number of instances in each Auto Scaling group and specify scaling policies to automatically launch or terminate instances as per the demand of your application. Auto Scaling ensures that your group neither goes below this size nor above the maximum specified size and has as many instances as specified.

AWS explains auto-scaling with the following example:

The following Auto Scaling group has a minimum size of 1 instance, a desired capacity of 2 instances, and a maximum size of 4 instances. The scaling policies that you define adjust the number of instances, within your minimum and a maximum number of instances, based on the criteria that you specify.

But how do you manage at times of low demand?

Since auto-scaling groups are configuration specific, you can manually adjust the settings and schedule for automatic scaling-down of your instance group which is mostly weekends when there is low demand or no demand. This would help you cut unnecessary costs being incurred.

9  Schedule your Usage - Turn off Instances on Weekends

One of the canny strategies for AWS cost optimization is to turn off the workloads on weekends or whenever they are in the non-production state. Saving costs in non-production workload environments would require scheduling the downtime of your EC2 servers in times of low demand. Your services running in the AWS environment also need to be bootstrapped so that they start & stop automatically in line with initiation & termination of your EC2 servers respectively.

Workload environments such as Dev-test, QA, sandpits or staging can save you big if turned-off when not in use. Schedule your EC2 servers to start and shut down by using instance tags to group resources. This makes it easier to apply policy-based actions to control shutdown of entire instance groups when they’re idle.

Shutting down resources outside working hours can unlock over 50% savings on the on-demand charges.
Elastic IP Addresses

An Elastic IP address is a static IPv4 address designed for dynamic cloud computing. An Elastic IP address is associated with your AWS account. With an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account. – AWS

With each running instance on AWS, you get one free Elastic IP address (EIP). EIPs are, however, a limited resource to rely upon. An unattached Elastic IP address remains allocated to your AWS account until explicitly released.

Amazon, in order to ensure effective consumption of addresses, imposes a small hourly charge on EIPs dissociated with a running instance, or EIPs associated with a halted instance or an unattached network interface. You are not charged for one Elastic IP address associated with the running instance but, Amazon levies charges for any additional Elastic IP addresses attached to the instance.

To cut down superfluous costs, you ought to release EIPs the very moment instances are halted or terminated with no immediate plans to restart them or attach it to an instance.
Worthy Nomination for AWS Cost Optimization

Spot Instances:

Amazon introduced the concept of Spot Instances to furnish its customers with unused EC2 compute capacity at very low rates. AWS customers can request for a Spot Instance by bidding on the maximum price they can pay every hour per instance. The user with the highest bid in spot marketplace gets the instance and pays the price he/she bade for.

But, to understand how the Spot Instance pricing works, you first need to understand the catch of bidding. Same EC2 instances are bade for different prices across different AWS Regions. You need to keep an eye on your AWS console for the pricing history of Spot Instances to choose a bid value that works for you. Another catch is to look for the region where the highest bid price of the instance you need and the number of bidders are the lowest amongst all the regions. Bidding there, increases your chances of getting the EC2 instance and run your applications or workloads that are neither mission-critical nor region-dependent.

Spot Instances can save you more than 50% and up to 90% if bid strategically. According to AWS, Spot Instances costs are usually over 75% lower than on-demand costs.

However, these savings with spot instances come with some challenges and risks you need to take care of.

When an EC2 instance is available at your bid price, you will be allotted that instance. As soon as the spot price of the instance increases and exceeds your bid, it will be cut off from your workloads and allotted to the highest bidder.

However, this kind of “fly-by-night” resource can be utilized effectively without putting your system at risk or affecting business operations. Keep the following aspects in mind:

• Since the entire concept of spot instances depends on the highest bid price, there is no surety that you would always get the instances when you need them.
• For those hoping to deliver stable online end-user experience; using spot instances for real-time applications or mission-critical workloads like Web applications is a “bad idea.” Since they work in a very unpredictable manner, consider using them for the applications or workloads that doesn’t mind downtime or even sudden failures.
• Spot instances can be best leveraged for high-performance computing (HPC) when fleets of machines need to be spun up for performing complex analysis within a short time. That many machines are practically not affordable to run at higher on-demand rates and instances at spot prices can drive you, massive cost savings.
• Spot instances are also a great option for running background jobs faster, conducting large-scale testing, and running non-interactive batch jobs such as report generation at the end of a business day, processing large data loads in off-peak times, video encoding, etc.
• For stateless applications that can still work even if some parts of the application are turned off or its state is lost at any given time, spot instances are a good option to consider.