AWS data transfer costs in a nutshell

17 December 2025 - 10 min. read

You've meticulously calculated the cost of your EC2 instances, optimized your S3 storage tiers, and fine-tuned your RDS database IOPS. Everything seems under control. Then, the monthly bill arrives, and it's significantly higher than you expected. You scroll down, past the services you know and love, and find the culprit: a painfully large line item for Data Transfer.

If this sounds familiar, you're not alone. Data transfer is a cost that many engineers and architects often underestimate, treating it as a rounding error until it becomes a major, recurring expense.

The problem isn't just that moving data costs money. Figuring out when you're charged, how much you're charged, and why you're charged is notoriously complex.

Does moving data from an EC2 instance to an S3 bucket cost you?

What about replicating an RDS database across Availability Zones for high availability?

What happens if that traffic passes through a NAT Gateway?

This lack of clarity can lead to architectural blind spots, where a standard design decision for reliability or scalability silently balloons your operational costs.

In this article, the first in a series dedicated to this topic, we're going to demystify AWS data transfer. We'll cut through the dense documentation to give you the core principles you actually need to know. We will begin with the fundamental rules that govern all data movement and then demonstrate how they apply to the real-world scenarios you encounter every day.

Let's learn to predict, control, and optimize your AWS data transfer costs.

The Three Golden Rules of Data Transfer

The AWS pricing page for data transfer spans thousands of lines and looks intimidating. The good is that you don't need to memorize it. Almost every data transfer charge on your bill can be understood by remembering three fundamental baskets.

Data IN (Ingress) is (Almost) Always FREE

AWS wants your data. They make it as easy and cheap as possible to move your workloads, applications, and data into their ecosystem.

This means you are not charged for data transferred from the internet to your AWS services.

For example, a user uploads a 50MB video to your web application running on EC2. The data travels from their laptop, over the internet, to your Application Load Balancer (ALB). You pay $0.00 for that 50MB transfer.

If you are migrating a 2TB database from your on-premises data center to an Amazon RDS instance. You pay $0.00 for that 2TB ingress.

If your customers or IoT devices upload millions of small files (logs, photos, etc.) to an S3 bucket. You pay $0.00 for all that incoming data.

Like any good rule, there are a few niche exceptions (e.g., data transferred into a Global Accelerator). But for 99% of standard architectures, this rule holds: Ingress from the internet is free.

Data OUT (Egress) to the Internet is (Almost) Always PAID

This is the cost that everyone is aware of and typically accounts for the largest portion of the "Data Transfer" line item on your bill.

Any data leaving an AWS service and traveling out to the public internet is billable. This cost is calculated per Gigabyte (GB), and the price varies by AWS Region.

For example, if a user visits your website and your server (on EC2 or Fargate) sends back a 3MB webpage (HTML, CSS, JS, images). You pay for that 3MB of Data Transfer Out. Don’t forget to multiply this cost by each visitor, or worse, page refresh.

You host a 100MB PDF whitepaper on S3. Every time a customer downloads it directly from S3, you incur a 100MB Data Transfer Out charge.

Your mobile app calls your backend API (on API Gateway + Lambda). The API returns a 50KB JSON response. You pay for that 50KB transfer.

AWS is generous and provides a Free Tier that includes 100GB per month of free Data Transfer Out to the internet. This 100GB is aggregated across all services and regions (except for GovCloud and China). For small projects, this is fantastic. For any production application at scale, this free tier is typically consumed within the first few days of the month.

Data Transfer Inside AWS (or between AWS services)... It Depends

This is the category with the most hidden complexity.

What about data that never leaves the AWS network? Data moving between your own services? This is where the "hidden" costs live, and the answer is always "It depends."

The cost here depends entirely on the path the data takes. Is it staying in the same "data center"? Is it crossing to another? Is it going to another country? In AWS terms, we're referring to movement within an Availability Zone (AZ), across Availability Zones (inter-AZ), or across Regions (inter-Region).

Nearly every cost decision for internal traffic comes down to three concepts: Availability Zones (AZs), Regions, and service-specific routing (like VPC Endpoints).

Scenario A: Same Region, Same Availability Zone (Intra-AZ)

This refers to traffic between services in the same logical location.

Data transfer between services within the same AZ (e.g., from an EC2 instance in us-east-1a to an RDS instance in us-east-1a) using their Private IP addresses is FREE.

This rule only applies when using Private IPs. If your EC2 instance in us-east-1a communicates with another service in us-east-1a using its Public or Elastic IP address, the traffic must leave the VPC, hit the AWS public network border, and come back in (a process called "hairpinning").

In this "public IP" scenario, you are charged for Data Transfer OUT.

This is an expensive and common configuration mistake.

Scenario B: Same Region, Different Availability Zones (Inter-AZ)

This is the most common "hidden cost" and is directly tied to building highly available applications.

Any traffic that crosses an AZ boundary is PAID. This applies even if the services are in the same VPC (e.g., from a subnet in us-east-1a to a subnet in us-east-1b).

The Cost is typically $0.01 per GB in each direction. This means sending 1GB of data and receiving a 1GB response costs a total of $0.02.

This is the fee for using AWS's high-bandwidth, low-latency, redundant fiber that connects its data centers. When you run a Multi-AZ database, this is the cost of your data replication.

Scenario C: Different Regions (Inter-Region)

This is traffic for disaster recovery (DR) or global applications.

All data transferred from one AWS Region to another is PAID (e.g., from eu-west-1 (Ireland) to eu-central-1 (Frankfurt)).

The Cost is typically more expensive than Inter-AZ traffic, and the price varies depending on the source and destination regions.

The "Gotchas" & Service-Specific Rules

Sometimes, the cost doesn't just depend on where the services are, but how they're connected.

1. VPC Peering: When you connect two VPCs:

- If the VPCs are in the same AZ, traffic is FREE.
- If the VPCs are in different AZs (but the same Region), you pay the Inter-AZ fee.
- If the VPCs are in different Regions, you pay the Inter-Region fee.

2. NAT Gateway: This is a major cost trap.

The Problem: An EC2 instance in a private subnet needs to access S3 (which is a regional service, but its access point is public). Without a VPC Endpoint, the traffic routes to your NAT Gateway.

You pay two fees:

The Inter-AZ fee if your NAT Gateway is in a different AZ (which it should be for HA!).

The NAT Gateway Data Processing Fee.

This traffic is not "Data Out to Internet" because S3 is on the AWS network, but it's often more expensive!

3. VPC Gateway Endpoints (S3 & DynamoDB)

You create a Gateway Endpoint for S3 in your VPC.

This creates a direct, private route from your VPC to S3.

All traffic from your instances (in any AZ in that region) to S3 via this endpoint is FREE.

This is a non-negotiable best practice for cost optimization.

The quick wins to reduce data transfer costs

Alright, now that we understand the rules and have seen why we get charged, let's get to the good part: how to optimize that bill.

1. Embrace Amazon CloudFront (Your CDN Shield)

This is your first, best, and most powerful defense against the “Data Out” fee.

Think about it. Every time a user, somewhere in the world, downloads an image, a video, a PDF, or even just the main JavaScript bundle for your web app, you are paying the premium price for data to leave your EC2 instance, ALB, or S3 bucket.

The fix? Stop serving it directly. Put Amazon CloudFront, AWS's global Content Delivery Network, in front of your application.

This is a triple win:

Cheaper Rates: The per-GB price for data served from CloudFront is just... cheaper. It's often up to 40% less expensive than serving that same file directly from EC2.

A Massive Free Tier: CloudFront comes with its own separate and incredibly generous, permanent free tier: 1TB of data out per month, completely separate from the 100GB AWS-wide free tier. For many applications, this is a game-changer.

The Real Magic (Free Origin Fetch): This is the crucial part. The traffic from your "origin" (your S3 bucket or ALB) to the CloudFront network to cache that file is completely, 100% free.

2. Master VPC Endpoints (and Starve Your NAT Gateway)

Now let's hunt down that sneaky "hidden" cost from internal traffic. The most notorious offender is the NAT Gateway.

Remember our EC2 instance in a private subnet? It has no direct route to the internet, but it still needs to talk to AWS services like S3 to download a file or DynamoDB to write data. By default, how does it do this? It routes its traffic to your NAT Gateway (which is in a public subnet), which then sends the traffic to S3.

This is a cost trap. You get billed twice for this one action:

The solution is to create a VPC **Gateway** Endpoint for S3 and DynamoDB.

The endpoint itself is free. And all the data that flows through it is absolutely free. You just completely bypassed the NAT Gateway and all its associated fees for that traffic.

Note: We are talking about Gateway endpoints. Interface endpoint traffic **is billed**; however is less costly than NAT gateway traffic.

3. Design with "Availability Zone-Awareness"

Every gigabyte that crosses an AZ boundary (Inter-AZ) costs $0.01 in each direction.

This is the "cost of resilience," and frankly, for things like database replication, it's a cost we should be happy to pay.

Where we get into trouble is with unnecessary cross-AZ traffic, often from very "chatty" applications that are constantly talking to each other.

Conclusions

While the AWS pricing page may appear to be an uncrackable code, the reality is that almost every charge on your bill boils down to a few simple principles.

Let's recap the Three Golden Rules one more time:

- Data IN from the Internet is (almost always) FREE.
- Data OUT to the Internet is (almost always) PAID.
- Data INSIDE the AWS network
  - Intra-AZ (using private IPs) is FREE.
  - Inter-AZ (for high availability) is PAID.
  - Inter-Region (for disaster recovery) is PAID (and costs more).

By just remembering these rules, you can start to "see" the data transfer costs before you even write a line of code. You can understand that a Multi-AZ RDS instance isn't just paying for compute; it's paying for the resilience of Inter-AZ data replication. You can recognize that serving a 1MB image from EC2 costs real money, while serving it from CloudFront is practically free.

Optimizing your AWS bill isn't just about turning things off; it's about making informed decisions. It's about building smarter, more efficient architectures. Using tools like CloudFront and VPC Gateway Endpoints isn't just a best practice for performance or security; it's one of the most powerful cost-control levers you have.

What's Next?

We've covered the fundamentals, the "101" of AWS data transfer. You now have the knowledge to analyze 90% of your data transfer bill.

But what about that last 10%? What about more complex, hybrid scenarios?

In our next article, we'll take a deep dive into the advanced topics

Stay tuned!

About Proud2beCloud

Proud2beCloud is a blog by beSharp, an Italian APN Premier Consulting Partner expert in designing, implementing, and managing complex Cloud infrastructures and advanced services on AWS. Before being writers, we are Cloud Experts working daily with AWS services since 2007. We are hungry readers, innovative builders, and gem-seekers. On Proud2beCloud, we regularly share our best AWS pro tips, configuration insights, in-depth news, tips&tricks, how-tos, and many other resources. Take part in the discussion!

Alessio Gandini

Cloud-native Development Line Manager @ beSharp, DevOps Engineer and AWS expert.Since I was still in the Alfa version, I'm a computer geek, a computer science-addicted, and passionate about electronics all-around.At this moment, I'm hanging out in the IoT world, also exploring the voice user experience, trying to do amazing (Io)Things.Passionate about cinema and great TV series consumer, Sunday videogamer

---