

# Investigation Report on Why drberg AWS is consuming unusual Data Transfer Cost?

Jahidul Arafat, DevOps Engineer, Harris Web Works

April 24, 2020

# Contents

<b>1</b>	<b>DevOps</b>	<b>1</b>
1.1	Problem-1: The Dr.Berg site is having unusual Data Transfer Cost . . . .	1
1.1.1	Problem Description . . . . .	1
1.1.2	Earlier Discussion between @harris,@emiraj and @warren . . . . .	1
1.1.3	Dr.Berg AWS Feature Usage Bill . . . . .	3
1.1.4	How I debugged the huge Data Transfer cost on drberg AWS . . .	3
1.1.4.1	Step-1: Compare the DrBerg AWS and HWW AWS Cost breakdown and analyze what went wrong . . . . .	7
1.1.4.2	Step-2: Use AWS Cost Explorer to get more information	8
1.1.4.3	Step-3: AWS-Business Support Suggestion . . . . .	10
1.1.4.4	Step-4: Using CloudWatch to identify the instances in- flating the Data Transfer Bill . . . . .	12
1.1.4.5	Step-5: VPC Flow Log Analysis using CloudWatch . . .	13
1.1.4.6	Step-6: Try to dig out the IPs 18.233.43.190, 104.26.8.179 and 104.26.9.179 . . . . .	18
1.1.5	Findings and Recommendations . . . . .	23
1.1.5.1	Findings 01: On IP behind the CloudFlare and the reverse proxy security . . . . .	23
1.1.5.2	Findings 02: On drberg AWS architecture (see Figure 1.4)	23
1.1.5.3	Findings 03: From CloudWatch Network In/Out Bytes: EC2, RDS, ElastiCache . . . . .	24
1.1.5.4	Findings 04: Did I find any inconsistant Data Flow? . .	24
1.1.5.5	Findings 05: On VPC Log Analysis using CloudWatch .	24
1.1.5.6	Findings 06- Major misconfiguration found in the drberg aws architecture . . . . .	25
1.1.5.7	Recommendations . . . . .	26
1.1.5.8	Proposed Cost to take over the drberg site . . . . .	26
1.1.5.9	Learning . . . . .	26
1.1.5.10	Tools used, in chronological order . . . . .	27

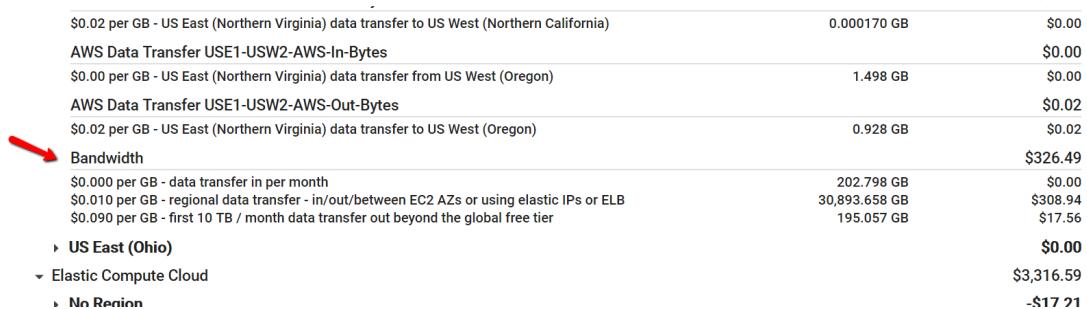
# Chapter 1

## DevOps

### 1.1 Problem-1: The Dr.Berg site is having unusual Data Transfer Cost

#### 1.1.1 Problem Description

- i Reporting time: 16 April, 2020 at around 2.00 AM, BDT by @emirajbbd
- ii Problem Area: Dr.Berg site is consuming more bandwidth then an usual Magento2 site would consume and cost  $\geq$  \$350 for the data transfer since January, 2020 (see figure 1.1).
- iii Under Bandwidth Cost, *the regional data transfer - in/out/between EC2 AZs or using elastic IPs or ELB* consumed around 30TB = 30,000GB, means 1000GB daily, which is unusual.



\$0.02 per GB - US East (Northern Virginia) data transfer to US West (Northern California)	0.000170 GB	\$0.00
AWS Data Transfer USE1-USW2-AWS-In-Bytes		\$0.00
\$0.00 per GB - US East (Northern Virginia) data transfer from US West (Oregon)	1.498 GB	\$0.00
AWS Data Transfer USE1-USW2-AWS-Out-Bytes		\$0.02
\$0.02 per GB - US East (Northern Virginia) data transfer to US West (Oregon)	0.928 GB	\$0.02
<b>Bandwidth</b>		<b>\$326.49</b>
\$0.000 per GB - data transfer in per month	202.798 GB	\$0.00
\$0.010 per GB - regional data transfer - in/out/between EC2 AZs or using elastic IPs or ELB	30,893.658 GB	\$308.94
\$0.090 per GB - first 10 TB / month data transfer out beyond the global free tier	195.057 GB	\$17.56
▸ US East (Ohio)		<b>\$0.00</b>
▾ Elastic Compute Cloud		\$3,316.59
▸ No Region		-\$17.21

Figure 1.1: Reported Problem: Bandwidth Consumption (March,2020, Dr.Berg)

#### 1.1.2 Earlier Discussion between @harris,@emiraj and @warren

Together they had identified the following probable issues as listed below:

- i Data Transfer Cost (average)  $\geq$  \$350
- ii Reverse Proxy Used: CloudFlare
- iii Number of products in the Dr.Berg Webserver  $\leq$  100
- iv EC2 and RDS Connection issue in Dr.Berg current development team as mentioned by @warren
  - Probably RDS not in the same availability zone as their ec2 instance..... so they get charged for regional data transfer.
  - Team Dr.Berg probably has RDS in one availability zone... ec2 in a different one..... so all data between magento and mysql cost money
  - Data transferred "in" to and "out" from Amazon EC2, Amazon RDS, Amazon Redshift , Amazon DynamoDB Accelerator (DAX), and Amazon ElastiCache instances or Elastic Network Interfaces across Availability Zones or VPC Peering connections in the same AWS Region is charged at \$0.01/GB in each direction.
  - If you **wget** between Availability Zones then you are charged. 10GB wget AZ 1 to AZ 2 cost USD \$0.10
  - What did the Team Dr.Berg transferred? its too huge for RDS and Elastic cache? \*\*\*\*\**(Solve this mystery)*
  - Database backup is 3.3GB per day (Internal). *My Note: \*\*\*(What about the RDS Active backup? Check it.)*
- v Crontab under the webserver user: ubuntu and under the root user
  - Under webserver user @ubuntu \*\*\* Problem detected by @emirajbbd: magento cron should run under apache/nginx user
 

```
#~ MAGENTO START 69dd2b02e1f3a65918182048ea4e29979a849d8942e8f53ed20a4bf10e529b36
* * * * * /usr/bin/php7.1 /var/www/html/bin/magento cron:run 2>&1 | grep -v "Ran jobs by schedule" >> /
var/www/html/var/log/magento.cron.log
* * * * * /usr/bin/php7.1 /var/www/html/update/cron.php >> /var/www/html/var/log/update.cron.log
* * * * * /usr/bin/php7.1 /var/www/html/bin/magento setup:cron:run >> /var/www/html/var/log/
setup.cron.log
#~ MAGENTO END 69dd2b02e1f3a65918182048ea4e29979a849d8942e8f53ed20a4bf10e529b3600 02 * * * _ /home/
ubuntu/scripts/db_backup.sh
```
  - Under root user: No cron job found
- vi HWW Practices in data transfer as mentioned by @emirajbbd
  - HWW usually use wget to transfer files from one server to another like from old aws2 to new server while migrating container. but it is not outside of the region and also used same aws elastic ip to aws elastic ip and so we were not charged
- vii Approximate cost (if HWW undertake this project) as suggested by @Warren:

**Server with disk space= \$575**  
**Bandwidth = \$350**  
**Backup\Monitoring\Others= \$100**

---

**Total Cost =\$1025**

viii @warren suggested the following existing AWS facilities of DrBerg will no longer be used:

- RDS for database
- AWS Elastic Search Service
- AWS ElastiCache (Redis)

ix The BW Billing has the following 03 major sections:

- \$0.000 per GB - data transfer in per month <sup>1</sup>
- \$0.010 per GB - regional data transfer - in/out/between EC2 AZs or using elastic IPs or ELB <sup>2</sup>
- \$0.090 per GB - first 10 TB / month data transfer out beyond the global free tier <sup>3</sup>

### 1.1.3 Dr.Berg AWS Feature Usage Bill

This section provides a comparative monthly billing information for the dr.berg site on the following 05 AWS features: Data Transfer cost, Elastic File System cost, ElastiCache/Redis billing, RDS and ElasticSearch Usage cost (see Table 1.1).

Table 1.1: AWS Feature Usage Cost of Dr.Berg.

AWS Target Features/Month	Data Transfer(\$)	EFS(\$)	ElastiCache/Redis(\$)	RDS(\$)	ElasticSearch(\$)
December/2019	63.22	8.04	569.77	458.32	N/A
January/2020	348.33	13.01	1017.79	1200.64	N/A
February/2020	347.61	13.03	952.13	1134.98	N/A
March/2020	326.51	13.09	1017.79	1200.64	N/A

### 1.1.4 How I debugged the huge Data Transfer cost on drberg AWS

During the month of January, February, March and April, 2020, the cost on DRBERG AWS was sky-rocketing (see Figure 1.2):

<sup>1</sup>First 1GB bandwidth = \$0.00 ("Free Tier")

<sup>2</sup>i.e. Under region us-east-1 and data transfer between EC2<->EC2, EC2<->RDS, EC2<->Redis, Redis<->Redis, EC2<->Third part app etc

<sup>3</sup>means: data transfer out to the internet. i.e. A user visiting your site and viewing an image.

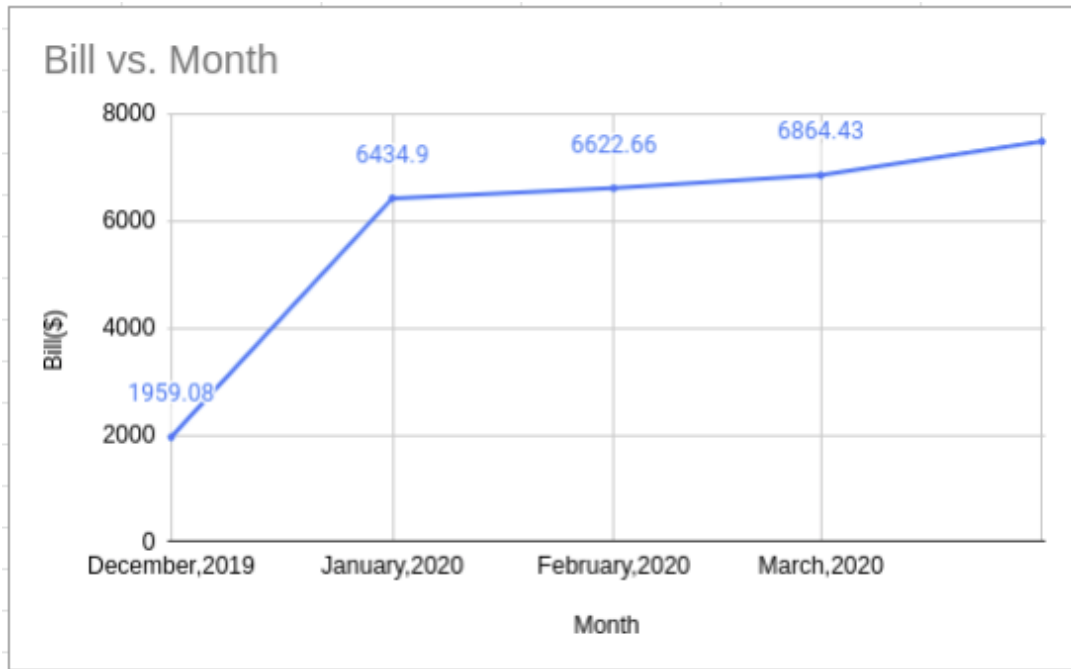


Figure 1.2: DrBerg AWS Billing

Table 1.2: AWS Ec2 instance details

Instance Info	EC2 Instances			
	EC#1 i-03453cdae379ba6cf	EC#2 i-03453cdae379ba6cf	EC#3 i-03453cdae379ba6cf	EC#4 i-03453cdae379ba6cf
Type/name	prod1-MVP-ebs-optimized-1.1-23-12-2019 c5.9xlarge	new prod inst - current production c5.4xlarge	NewUX_ec2_clone c5.4xlarge	drberg.newcoveb.dev t3.medium
VPC/Subnet/Private/Public IP	172.31.*.* Having 6X subnets and all are hosted into same subnet: 172.31.80.0/20 172.31.94.191 18.212.199.43 (Non Elastic)			
Hosted Zone/Creation Date	25 Dec, 2019	17 Jan, 2020	6 April, 2020	26 Dec, 2019

The projected costs were also increasing every day at a rapid pace. It seems like, something was going very wrong. This section explains how I debugged this in under 5 days. it is meant to be a pointers guide for HWW in future, if we encounter high Data Transfer costs on AWS. Hopefully, it saves us time. Below are the steps I took to debug and finally fix the leakage.

- i. The drberg (toyne) aws account has 4X EC2 instance running in US-NV (see Table 1.2).
- ii. All the EC2 instances on AWS are from same Availability Region (us-east-1d) and thereby of the same subnet 172.31.80.0/20. But the interesting fact is that , EC#1,EC#2 and EC#3 are pointing to the same service: <https://shop.drberg.com> with similar UX and with similar range of product information and other features. But EC#4 is seems to be the old site, with old UX design. **Check Investigation 1a.**
- iii. **ElastiCache** - For caching, the AWS account is using ElastiCache which hosts

Table 1.3: AWS ElastiCache Cluster being used

Redis Cluster	No	AZ	Private IP	Port	Master/Slave	Active Backup/snapshot	Data Sync
<b>Cache 0001</b> Created on: 17 Dec, 2019	001	us-east-1d	172.31.83.216	6379	Primary/Master	No	in-sync
	002	us-east-1a	172.31.28.42	6379	Replica	Yes (daily)	
	003	us-east-1e	172.31.77.50	6379	Replica	No	
<b>Session 0001</b> Create on: 11 Dec, 2019	001	us-east-1b	172.31.43.114	6379	Primary/Master	No	in-sync
	002	us-east-1a	172.31.22.65	6379	Replica	Yes (daily)	
	003	us-east-1d	172.31.82.114	6379	Replica	No	

their Redis Clusters for caching and queuing. Here are some information related to the ElastiCache Cluster being used (see Table 1.3) and Check the investigations 2a,2b,2c,2d.

- iv. **Amazon RDS** - Database being used is RDS. The database cluster is running in US-EAST-1 (NV). The details RDS architecture being used is in Table 1.4. **Check investigation 3a,3b,3c,3d**

Table 1.4: AWS RDS Cluster being used

RDS	AZ	Type	Size	Max Threshold	Public/Private IP	EC2 Hosting RDS	Lead Load Generator EC2 AAS	External Application Access	DB Backup
drberg Creation date: 11 Dec, 2019	us-east-1a	db.m5.4xlarge	1590GiB	3000GiB	52.0.178.138 172.31.17.213	172.31.94.191 EC2#1 172.31.81.143 EC2#2	172.31.81.143 (0.62) 172.31.94.191 (0.38)	198.255.52.130	Daily
database-new Creation Date: 6 Apr, 2020, has NFS:2049 in its security group	us-east-1f	db.m5.4xlarge	100GiB	1000GiB	No Public IP 172.31.55.52	172.31.93.238 EC2#3	172.31.93.238 (0.47)	No external application access	Daily

**EC2 Data Sync Operations:**

**Investigate 1a (Same Region-Same AZ: using private IP)** are there any data sync operations going on among these EC2 instances (EC2#1,2,3) using private IP/public IP and does it cost BW?<sup>a,b</sup>

**ElastiCache Investigations:**

**Investigate 2a (Same Region-Different AZs)** Does the data sync operation between Redis Master/Slave cost additional BW in different AZs?<sup>c</sup>

**Investigate 2b** Why there is only one backup for each active Redis Cluster (cache and session)?<sup>d</sup>

**Investigate 2c (Same Region-Different AZs)** Does the data in-sync operation from EC2 to Redis cluster in same region-different AZs incur additional data transfer cost?<sup>e</sup>

**Investigate 2d** Does data in/out from the ElastiCache node i.e. cache-002 to itself cost additional bandwidth?<sup>f</sup>

**RDS Investigations:**

**Investigate 3a (One RDS- accessed by 2X EC2)** RDS: drberg is being accessed by two EC2 instances (EC2#1 & EC2#2). Does the Data in/out from EC2 from and to RDS in Same Region-Different AZs cost additional data transfer cost?<sup>g</sup>

**Investigate 3b** RDS #drberg might be accessed by an external application at 198.255.52.130 as this ip is found in its security group. **Could it be a potential cause of Data Transfer Cost hike?**<sup>h</sup>

**Investigate 3c** RDS #Database-new has NFS access in its SG. Does it cost additional charge for the RDS-EFS data transfer?<sup>i</sup>

**Investigate 3d** Both RDS #drberg & #Database-new has daily backup. Does this snapshot creation add additional BW cost?<sup>j</sup>

<sup>a</sup>with Private IP, No BW cost. Data transferred between Amazon EC2 and Elastic Network Interfaces in the same Availability Zone is free.

<sup>b</sup>with Public IP, Yes, BW Cost. Data transferred between EC2 instances or containers, or Elastic Network Interfaces in the same availability zone and same VPC via public or Elastic IPv4 addresses are charged at \$0.01 per GB for egress traffic and \$0.01 per GB for ingress traffic.

<sup>c</sup>Yes!!! Data transferred "in" to and "out" from Amazon ElastiCache instances or Elastic Network Interfaces across Availability Zones in the same AWS Region is charged at \$0.01/GB in each direction.

<sup>d</sup>because 1 active backup is free, Storage space for additional backups is charged at a rate of \$0.085/GB per month for all AWS regions. *No additional data transfer fees for creating a backup*

<sup>e</sup>Yes, similar to footnote (c)

<sup>f</sup>No. There is no Amazon ElastiCache Data Transfer charge for traffic in or out of the Amazon ElastiCache Node itself

<sup>g</sup>Yes!!! Data transferred "in" to and "out" from Amazon EC2, Amazon RDS and Elastic Network Interfaces across Availability Zones in the same AWS Region is charged at \$0.01/GB in each direction.

<sup>h</sup>Could be, but I cant check it until having the permission to create flowlogs and to have ssh access to the node to trace the traffic using **IPTraf** and **iftop**

<sup>i</sup>Yes!!! Bidirectional charges applies

<sup>j</sup>No. But if the snapshot is being copied to another regions, data transfer cost incur and the rate varies within the regions.



#### 1.1.4.1 Step-1: Compare the DrBerg AWS and HWW AWS Cost breakdown and analyze what went wrong

The analysis of the drberg and HWW billing details give me a significant understanding on the price hick due to the extreme data transfer in the drberg account in Dec,2019/Jan/Feb/March/April,2020 which is 5 times higher than the data transferred in HWW during these period. A comparative analysis is shown in Table 1.5 along with the events happened (i.e. new EC2 launched, RDS creation, ElasticCache cluster deployment) during this period to have a better understanding on what event might have an impact on the higher BW cost.

Table 1.5: deberg and HWW AWS Data Transfer cost comparison

AWS ACCOUNT	Data Transfer / Cost				
	Dec,2019	Jan,2020	Feb,2020	March,2020	April (Till 21 Apr,2020)
Toync(drberg)	6322GB=6.322TB (a) <sup>4</sup> 67.563 GB \@\$0.00 (b) <sup>5</sup> 5,772.427 GB \@\$0.010 (c) <sup>6</sup> 61.039 GB \@\$0.090 \$63.22	34831 GB=34.831 TB (a) 237.301 GB \@\$0.00 (b) 32,780.143 GB \@\$0.010 (c) 227.874 GB \@\$0.090 \$348.33	34759 GB=34.759 TB (a) 183.394 GB \@\$0.00 (b) 33,059.358 GB \@\$0.010 (c) 188.822 GB \@\$0.090 \$347.59	32649 GB=32.649 TB (a) 202.789 GB \@\$0.00 (b) 30,893.658 GB \@\$0.010 (c) 195.057 GB \@\$0.090 \$326.49	23491 GB=23.491 TB (a) 136.820 GB \@\$0.00 (b) 22,275.648 GB \@\$0.010 (c) 135.063 GB \@\$0.090 \$234.91
Toync Creation Happened ** I will explain later how these has an impact on BW cost	11 Dec,2019 RDS#drberg launched ->\$us-east-1a ElasticCache Cluster #Cache launched ->us-east-1b,1a,1d  17 Dec,2019 ElasticCache Cluster #Session launched ->us-east-1d,1a,1e  25 Dec,2019 EC2#1 launched ->us-east-1d  26 Dec,2019 EC2#4 launched ->us-east-1d	17 Jan,2020 EC2#2 launched ->us-east-1d			6 Apr,2020 RDS#Database-new launched ->us-east-1f  EC2#3 launched ->us-east-1d
HWW	No Data found No Data found	6693 GB=6.693TB \$66.93	6730GB=6.730 TB \$67.30	7294 GB=7.294 TB \$72.94	4489 GB=4.489 TB \$44.89

**Finding** - Major difference is found in the Data Transfer Cost. Monthly Data transfer cost is 5 Times higher than that of the HWW BW cost.

**Investigate-Step(1a)** See the Toync EC2/RDS/ElasticCache Creation timeline and since most of these are created in Mid Dec,2019, the Data transfer has experienced a hick in Jan/Feb/March. What could be the reasons?<sup>a</sup>

<sup>a</sup>Maybe the EC2/RDS/ElasticCache started data-sync among each other in different AZs since the creation date which results in data hick after Dec,25,2019

### 1.1.4.2 Step-2: Use AWS Cost Explorer to get more information

Cost Explorer is in the billing dashboard itself. I used “Usage Type Group” and used exclusion principle to eliminate the services not taking up costs. The only service which had increased costs was “EC2: Data Transfer–Inter AZ”. Check the graph in Figure 1.3 and in Table 1.6.

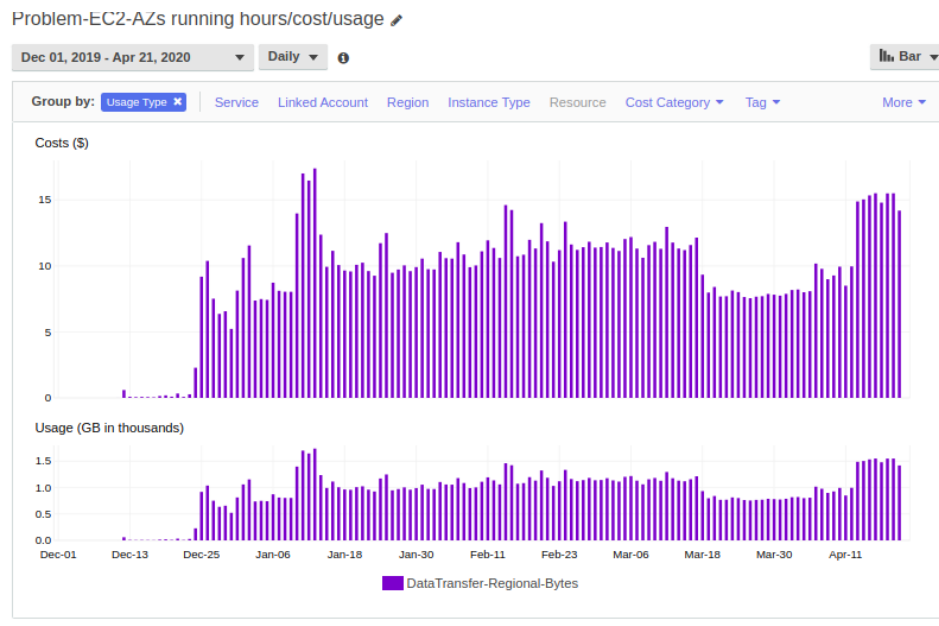


Figure 1.3: Cost Analyzer: EC2-Data Transfer–Inter AZ

Table 1.6: Data Usage- Major date breakdown

Category	Important Dates /Data Flow
Date with normal data flow/cost	23 Dec,2019/ \$0.2
Data flow significant raise/Date	25 Dec,2019/\$9.00
Maximum data flow cost observed/Date	11 Jan, 2020/\$17.01 12 Jan,2020/\$16.41 13 Jan,2020/\$17.36
Average Data Flow/Cost Since Dec 23, 2019	\$10.58/1058.100GB

**Finding** - The cost had gone up from \$0.2/day to \$10-\$17 per day since 25 Dec,2019.

I have gone through the Cost Analysis table generated in drberg AWS to check the root cause of this data transfer hike. You can find this in *Billings>Cost Explorer>Saved Report: Problem-EC2-AZs running hours/cost/usage*. Furthermore, on some

research, I found out that the “EC2:Data Transfer–Inter AZ” is the cost of *Data Transfer between instances located in different AZs*. Again, from Figure 1.3 , I have seen no inconsistent raise in the data flow from Dec,25,2019. Rather the data flow remains in the range of 1024GB to 1800GB per day since 25 Dec,2019 and that leads me to answer the following questions?

- (a) Why the data flow consistently remain in between 1024GB to 1800GB per day since 25 Dec,2019?
- (b) Have you observed the data transfer category? Its under **EC-Other** and the description is *Data Transfer-Regional-Bytes*<sup>a</sup>.

---

<sup>a</sup>means Same Region-Different AZs data transfer/sync/backup/snapshots are going on in between EC2-RDS-ElastiCache(Redis)-ElasticNetworkInterface

**A glimps of the DrBerg (AWS) Structure (see Figure 1.4)** which may help you identifying why data transfer consistently remains in between 1024GB to 1800GB per day since 25 Dec,2019, when before it was much lower.

- (a) Total EC2 Instances= 04 (EC2#1,EC2#2,EC2#3,EC#4) in us-east-1d.
- (b) All EC2 instances (1,2,3) loading the very similar contents and dbs and loads <https://shop.drberg.com>. Means, there might have a data sync operation among these EC2 instances and some of these EC2 instances might be working as dev site which are not managed in a cost effective manner. The data-sync might have used the public ip for the operation, results in bidirectional data charges applies @\$0.010.
- (c) RDS has two databases, (i)drberg and (ii) Database-new in us-east-1a and us-east-1f respectively.
- (d) EC2 Instances 1 and 2 are hosting the RDS #drberg while EC2#2 has maximum hosting load 0.62 on this RDS. The EC2 instances and RDS #drberg are in different AZs under same Region. Bidirectional data charges applies @\$0.010.
- (e) EC2 instance 3 is hosting the RDS #Database-new in different AZs again. Bidirectional data charges applies @\$0.010.
- (f) 2X redis cluster: Cache(001,002,003) in us-east-1b,1a,1d, Session(001,002,003) and are in the form Primary/Replica. The data-sync is going on among them, which results in additional BW cost across different Regions @\$0.010. But self data transfer operations inside any node does not cost BW. Moreover cross AZs data transfer charge applicable in between the EC2-Redis Clusters (bidirectional charges).
- (g) A single redis backup Cache-002 and Session-002 from each cluster, which is free of cost.
- (h) One interesting fact, a third party application or instance outside of the hosting VPC has access to the EC2 instances and the RDSs. That is: **198.255.52.130**. This ip is in my further investigation list to check how much beging transferred by this IP.

#### 1.1.4.3 Step-3: AWS-Business Support Suggestion

To minimize the data transfer cost, AWS Business support suggests the followings:

- (a) **Suggestion 1:** After having a back and forth about “EC2: Data Transfer —

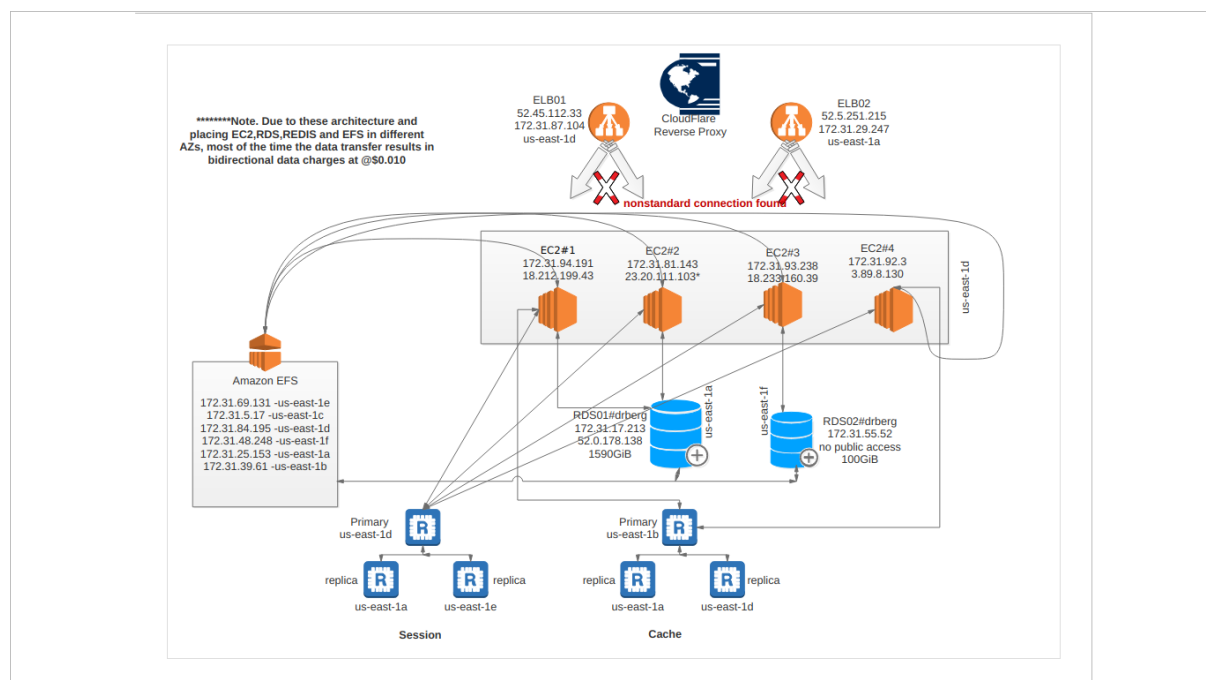


Figure 1.4: DrBerg AWS Architecture

Inter AZ”, they told these costs even include the cost of data transfer between drberg instances in one AZ and some other customer’s instances in another AZ.

The EC2 instances here are all in us-east-1d but there is a third party access from 198.255.52.130 out the current VPC. So data transfer cost might incur.

- (b) **Suggestion 2:** Use AWS VPC Flow Logs to monitor IP traffic coming in and going out from all my network interfaces. Use IPTrat to monitor the traffic going in and out of a particular interface. Use Iftop to monitor network flows onto the system.
- (c) **Suggestion 3:** Block port 22 and port 80 from the world where it wasn’t required.

drberg AWS already had that configuration done in their AWS Security Groups. So, this wasn’t a problem.

- (d) **Suggestion 4:** Use private IPs to communicate between my services as data transfer via public IP is also charged in “EC2: Data Transfer — Inter AZ” costs.

EC2{1,2,3}—pointing to  $\rightarrow$  <https://shop.drberg.com>. One of them could be the dev site or for someother testing and data might be synced using public ip, results in data transfer cost. Moreover, the Elastic Load Balancer, drberg AWS tried to implement is incomplete and useless as it even does not meet the minimal standard for a load balancer to have.

#### 1.1.4.4 Step-4: Using CloudWatch to identify the instances inflating the Data Transfer Bill

Here, I have explored this by creating graphs for EC2: “Network In”, EC2: “Network Out”, ElastiCache/redis: NetworkBytesIn, NetworkBytesOut and RDS: NetworkReceiveThroughput, NetworkTransmitThroughput as seen in Figure 1.5.



Figure 1.5: Using CloudWatch to identify the instances inflating the Data Transfer Bill

**Some interesting findings** : Here in Figure 1.5, I have analyzed the data for the last 3 days in the period of 19/4/2020 to 22/4/2020.

1. EC2 Network In: Everyday at around 2.15 UTC, the EC2 instance (i-085e01e0193867cef: new prod int-current production in us-east-1d) reaches to a pick (1.51GB to 1.93GB network inflow) for the last 3 consecutive days (see Figure 1.5a).
2. EC2 Network Out: Similarly, everyday at around 2.15UTC, the same EC2 instance produce more Network Outflow(around 500 MB to 751 MB) (see Figure 1.5b).
3. The ElastiCache cluster has seen to have a maximum hike for the cahce-001 NetworkBytesOut Primary Node, which is 121M on 20/4/2020 at 9.04UTC (see Figure 1.5c).
4. The RDS instances similar to EC2 Network-IN/Out, experienced a hike to 42MB at 2.15UTC (see Figure 1.5d).

**Note** - However these were using up data for sure, but the data spikes had no correlation with the consistent price increase. I did not believe this was the reason for high cost. *You will find this analytical report in CloudWatch -> Dashboards-> Problem\_debug.*

#### 1.1.4.5 Step-5: VPC Flow Log Analysis using CloudWatch

Here, I have used VPC flow logs. However setting up Flow Logs with CloudWatch consumed times . The VC Flow Log started on 2.10PM, 22 Apr,2020 BD Time and I collected data transferred in Bytes of **22 Apr, 2020 to 24 Apr,2020 unto 06.30PM BDT time** to identify who is the big player in this **HUGE!!!** data transfer flow. I take the following steps in analyzing the CloudWatch VPC Flow Logs.

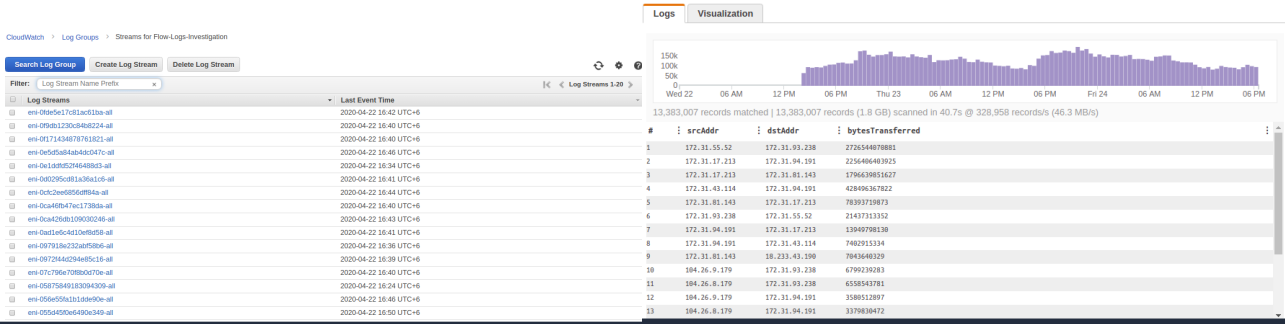
1. The log group named: **flow log investigation** which I have created under the *CloudWatch->Log section*, has collected data in 20 log streams in different Elastic-NetworkInterfaces(eni) as shown in Figure 1.6a.
2. Then I tried to dig the insights of the log streams using **VPC Flowlog Queries** which helped me to find out **Top 20 data transferred by source and destination IPs** with following query (See query result in Table 1.7). This results in the byte usage bar chart as in Figure 1.6c.

```
stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 20
```

Table 1.7: Top 20 Data Suckers of drberg VPC (22 Apr,2020-24 Apr,2020 (unto 6.30PM, BDT))

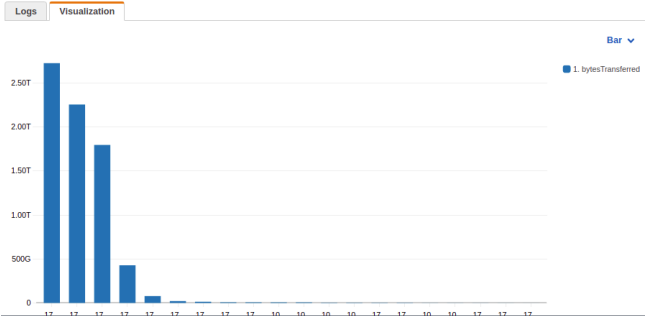
<b>srcAddr</b>	<b>dstAddr</b>	<b>bytesTransferred</b>	<b>Equivalent</b>
172.31.55.52	172.31.93.238	2726544070881	2.72TB
172.31.17.213	172.31.94.191	2256406403925	2.25TB
172.31.17.213	172.31.81.143	1796639851627	1.79TB
172.31.43.114	172.31.94.191	428496367822	428.49GB
172.31.81.143	172.31.17.213	78393719873	78.3GB
172.31.93.238	172.31.55.52	21437313352	21.4GB
172.31.94.191	172.31.17.213	13949798130	13.9GB
172.31.94.191	172.31.43.114	7402915334	7.40GB
172.31.81.143	18.233.43.190	7043640329	7.04GB
104.26.9.179	172.31.93.238	6799239283	6.79GB
104.26.8.179	172.31.93.238	6558543781	6.55GB
104.26.9.179	172.31.94.191	3580512897	3.58GB
104.26.8.179	172.31.94.191	3379830472	3.38GB
172.31.81.143	172.31.83.216	2991111672	2.99GB
172.31.83.216	172.31.81.143	2970194098	2.97GB
104.26.8.179	172.31.81.143	1234151889	1.23GB
104.26.9.179	172.31.81.143	1188223964	1.18GB
172.31.81.143	172.31.84.195	1077491991	1.07GB
172.31.84.195	172.31.81.143	774635668	774MB
172.31.81.143	162.158.78.93	400702725	400MB





(a) VPC Flow Log Streams

(b) Top 10 influential IPs



(c) TOP 10 Influential IP- Visualization

Figure 1.6: VPC Flowlog analysis using CloudWatch

**My Observation on Step-5**

1. The number of bytes being transferred were HUGE!! – close to 2.72TB in some cases in just 60 hours of time span.
2. The following IP List and their Usage in drberg may help to dig out the facts. See Table 1.8.
3. Below the top 11 interesting insights of data transfer from 22 Apr,2020 to 24 Apr,2020 till 6.30PM, BDT between (from Table 1.7) :
  - RDS02(172.31.55.52,us-east-1f)→EC#3(172.31.93.238,us-east-1d) around 2.72TB @\$0.010 per GB X2<sup>ab</sup>
  - RDS01(172.31.17.213,us-east-1a)→EC#1(172.31.94.191,us-east-1d) around 2.25TB @\$0.010 per GB X2
  - RDS01(172.31.17.213,us-east-1a)→EC#2(172.31.81.143,us-east-1d) around 1.79TB @\$0.010 per GB X2
  - Redis-Cache-001(172.31.43.114,us-east-1b)→EC#1(172.31.94.191, us-east-1d) around 428.49GB @\$0.010 per GB X2
  - EC#2(172.31.81.143,us-east-1d) → RDS01(172.31.17.213,us-east-1a) around 78.3GB @\$0.010 per GB X2
  - EC#3(172.31.93.238,us-east-1d) → RDS02(172.31.55.52,us-east-1f) around 21.4GB @\$0.010 per GB X2
  - EC#1(172.31.94.191, us-east-1d) → RDS01(172.31.17.213, us-east-1a) around 13.9GB @\$0.010 per GBX2
  - EC#1(172.31.94.191, us-east-1d) → Redis-Cache-001(172.31.43.114, us-east-1b) around 7.40GB @\$0.010 per GB X2
  - EC#2(172.31.81.143, us-east-1d) → **External IP(18.233.43.190)** around 7.04GB
  - ip:{104.26.9.179, CloudFlare Reverse Proxy} → EC#3(172.31.93.238,us-east-1d) around 6.79GB
  - ip:{104.26.8.179} → EC#3(172.31.93.238,us-east-1d) around 6.55GB
4. Some unknown ip has significant data transfer rate. So lets try to dig out these three ips: 18.233.43.190, 104.26.8.179, 104.26.9.179

<sup>a</sup>Bidirectional charge applies- (2.72TB Out from EC2\*0.010 per GB + 2.72TB into RDS\*\$0.010 per GB)

<sup>b</sup>For Calculation see: <https://medium.com/@mulupuru/your-comprehensive-guide-to-understanding-aws-data-transfer-costs-f5c8241d65ed>

Table 1.8: drberg AWS Infrastructure/IP/regions

<b>EC2 Instances</b>		
<b>IP</b>	<b>Description</b>	<b>Region</b>
172.31.94.191	EC2#1-Prod1-MVP-ebs-optimized1.1-23-12-2019	us-east-1d
172.31.81.143	EC2#2-New prod-inst-current-production	us-east-1d
172.31.93.238	EC2#3-NewUX-ec2-clone	us-east-1d
172.31.92.3	EC2#4-drberg.newcowebdev	us-east-1d
<b>ELB</b>		
172.31.87.104	ELB01- drberg	us-east-1d
172.31.29.247	ELB02-drberg2	us-east-1a
<b>RDS</b>		
172.31.17.213	RDS01-drberg	us-east-1a
172.31.55.52	RDS02-Database-New	us-east-1f
<b>ElastiCache/Redis</b>		
172.31.83.216	session-001	us-east-1d
172.31.28.42	session-002	us-east-1a
172.31.77.50	session-003	us-east-1e
172.31.43.114	cache-001	us-east-1b
172.31.22.65	cache-002	us-east-1a
172.31.82.114	cache-003	us-east-1d
<b>EFS</b>		
172.31.69.131	EFS01	us-east-1e
172.31.5.17	EFS02	us-east-1c
172.31.84.195	EFS03	us-east-1d
172.31.48.248	EFS04	us-east-1f
172.31.25.153	EFS05	us-east-1a
172.31.39.61	EFS06	us-east-1b

#### 1.1.4.6 Step-6: Try to dig out the IPs 18.233.43.190, 104.26.8.179 and 104.26.9.179

In Step-5, 03 unknown IP found in the data transfer top 10 chart. So I wanted to digout the information related to these IPs. The basic information that I had:

All the EC2#{1,2,3} instances pointing to a single site:  
**shop.drberg.com**

Steps I have taken to get the information of these two ip:

1. Used the tool **whatweb** to get the webserver info relating to 18.233.43.190 which has received around 7GB data from EC2#3-172.31.93.238(The Dev Site) over the 60 hours

```
> whatweb 18.233.43.190
Output
-----
http://18.233.43.190 [200 OK] Country[UNITED STATES]
[US], HTTPServer[Ubuntu Linux][nginx/1.10.3 (Ubuntu)],
IP[18.233.43.190], nginx[1.10.3]
```

2. Used the tool “**dig**” to reveal the DNS related information.
3. Used the following commands:

```
> dig +short -x 104.26.8.179
Return nothing
> dig +short -x 104.26.9.179
Return nothing
```

4. Since I didnt get anything from these **dig** operations, the only option left was to check whether these 2 IP has any correlation with the site:**shop.drberg.com**? I tried to dig information about this site and what revealed was just **SO INTERESTING!!!!**.

```
> dig shop.drberg.com
;; ANSWER SECTION:
shop.drberg.com. 299 IN A 104.26.9.179
shop.drberg.com. 299 IN A 104.26.8.179
```

5. But from @warren, I had the information that, drberg has used reverse proxy server in CloudFlare. So, I suspected, hence the dig site operation return these two IP, those for sure will be the CloudFlare Reverse Proxy of the site **shop.drberg.com**

```
> whois 104.26.9.179
```

```
> whois 104.26.8.179
```

Output

-----

```
# Copyright 1997-2020, American Registry for Internet Numbers, Ltd.
#
NetRange:      104.16.0.0 - 104.31.255.255
CIDR:          104.16.0.0/12
NetName:       CLOUDFLARENET
NetHandle:     NET-104-16-0-0-1
Parent:        NET104 (NET-104-0-0-0-0)
NetType:       Direct Assignment
OriginAS:      AS13335
Organization:  Cloudflare, Inc. (CLOUD14)
RegDate:       2014-03-28
Updated:       2017-02-17
Comment:       All Cloudflare abuse reporting can be done via
https://www.cloudflare.com/abuse
Ref:           https://rdap.arin.net/registry/ip/104.16.0.0
```

**Finding from Step-6:** The real ip of **shop.drberg.com** is hidden behind cloudflare reverse proxy. That's why "dig" command showing 104.26.8.179 and 104.26.9.179 as its hosting address and also consuming significant Bandwidth in data transfer between CloudFlare reverse proxy and the EC2 instances in drberg AWS account.

6. Lets try to byass the CloudFlare and see where the site is really hosted in using some specialized tools. This will help me to find the reason behind having too many EC2 instances pointing to the same site and where the main site really resides in.

a. **Hackers Way -01: Bypassing drberg CloudFlare using CloudFail**

```
> python3 cloudfail.py -t drberg.com --tor
```

Output

-----

```

/  _ _ _ _ _ | | _ _ _ _ _ | | _ _ _ _ _ | | _ _ _ _ _ |
| | | | | / _ \ | | | | / _ \ | | | | / _ \ | | | |

```

```

| |__| | ( ) | | | | ( | | _ | ( | | | |
\___|_| \___/ \_,_ \_,_ \_,_ \_,_ | | |
v1.0.1                                     by m0rtem

```

```

[18:01:08] Initializing CloudFail - the date is: 23/04/2020
[18:01:10] TOR connection established!
[18:01:10] New IP: 185.220.101.8
[18:01:10] Fetching initial information from: drberg.com...
[18:01:10] Server IP: 104.26.9.179
[18:01:10] Testing if drberg.com is on the Cloudflare network...
[18:01:21] Scanning 2897 subdomains (subdomains.txt), please wait...
[18:03:28] [FOUND:SUBDOMAIN] affiliates.drberg.com IP: 159.253.145.145 HTTP:
[18:03:51] [FOUND:SUBDOMAIN] app.drberg.com IP: 185.72.157.106 HTTP: 200
[18:04:22] [FOUND:SUBDOMAIN] autodiscover.drberg.com IP: 52.98.65.8 HTTP: 200
[18:07:34] [FOUND:SUBDOMAIN] courses.drberg.com IP: 185.72.157.106 HTTP: 200
[18:07:43] [FOUND:SUBDOMAIN] crm.drberg.com IP: 52.173.184.147 HTTP: 200
[18:25:20] [FOUND:SUBDOMAIN] shop.drberg.com ON CLOUDFLARE NETWORK!
[18:26:52] [FOUND:SUBDOMAIN] support.drberg.com ON CLOUDFLARE NETWORK!
[18:29:53] [FOUND:SUBDOMAIN] webmail.drberg.com IP: 173.203.187.14 HTTP: 200
[18:30:44] [FOUND:SUBDOMAIN] www.drberg.com ON CLOUDFLARE NETWORK!
[18:32:02] Scanning finished...

```

- b **Hackers Way -02: Gathering the information on drberg using dns-dumpster**<sup>7</sup> This further confirm and clarify that the above IPs which consuming a significant BW are the CloudFlare reverse proxy for drberg (See Figure 1.7 and Table 1.9 ).

---

<sup>7</sup><https://dnsdumpster.com/>

**Information Revealed** - 104.26.8.179 and 104.26.9.179 are the CloudFlare reverse proxy of drberg.com and the hacking tools revealed several of its subdomain, which is a weakness in their cloudFlare setup.

**Still What not get** - What is the real IP of drberg.com behind this cloudFlare is still not found. This is our next step to reveal. If We could discover it, then a confusion remains on,

“Why there are three EC2 instances pointing to the same site shop.drberg.com”, will be revealed.

**Highlight this IP** - The Table 1.9 revealed an ip **185.72.157.160** which found to be linked to several drberg subdomains: {test2.drberg.com,testapp.drberg.com,app.drberg.com}. We will have a look on it, as it may found to be useful later.

- c. **Hackers Way -03: Using censys.io<sup>89</sup> api along with CloudFlair<sup>10</sup> tool to dig the true IP of drberg behind CloudFlare**

```
> python3 cloudflair.py drberg.com
```

Output

-----

```
[*] Retrieving Cloudflare IP ranges from https://www.cloudflare.com/ips-v4
[*] The target appears to be behind CloudFlare.
[*] Looking for certificates matching "drberg.com" using Censys
[*] 40 certificates matching "drberg.com" found.
[*] Found 3 likely origin servers of drberg.com!
    - 23.20.111.103 (HTML content is 99 % structurally similar to drberg.com)
    - 185.72.157.106 (HTML content is 94 % structurally similar to drberg.com)
    - 18.212.199.43 (HTML content is 99 % structurally similar to drberg.com)
```

<sup>8</sup><https://censys.io/domain?q=cloudflare>

<sup>9</sup><https://www.shodan.io/host/104.17.98.21>

<sup>10</sup><https://github.com/christophetd/CloudFlair>

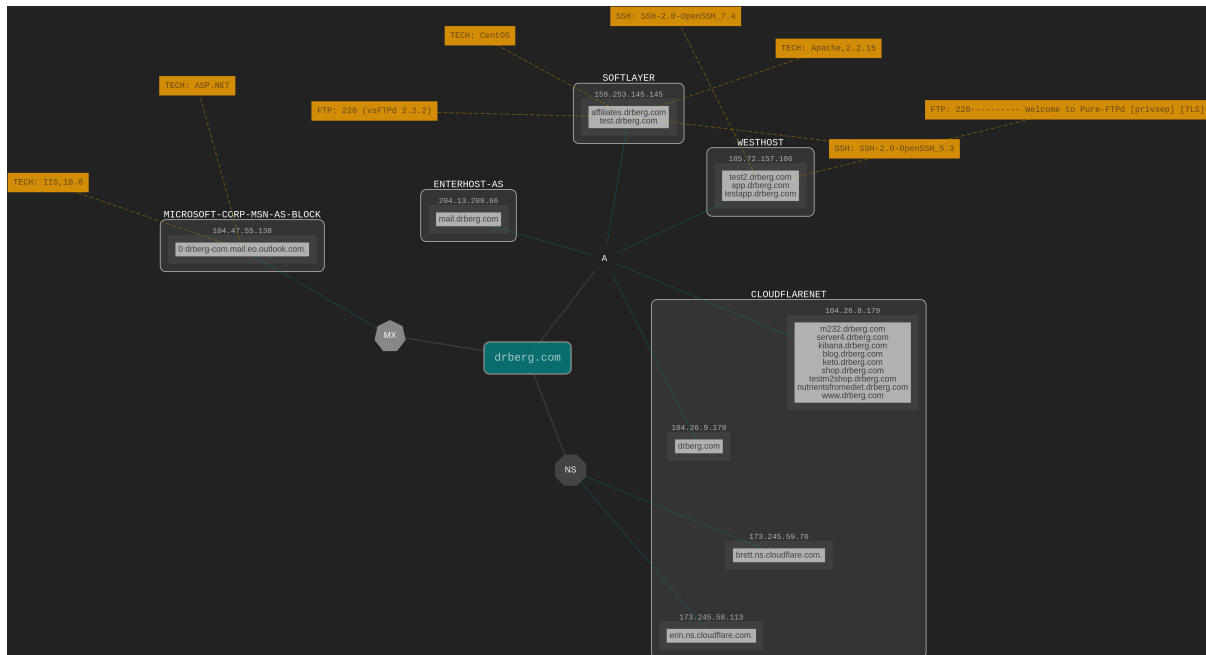


Figure 1.7: Gathering drberg network and hosting information using dnsdumpster

**Interesting Findings** - the censys.io api along with CloudFlair tool has reveals 03 most likely origin server of drberg.com. Means, the CloudFlare security of drberg has finally been breached and a **HUGE!!!!** finding discovered.

- 23.20.111.103 and 18.212.199.43 having 99% structural similarity to drberg.com.
- 23.20.111.103 -> EC2#2: New prod-instance-current-production under us-east-1d
- 18.212.199.43 -> EC2#1: Prod1-MVP-ebs-optimized1.1-23-12-2019 under us-east-1d
- 185.72.157.106 - this is the IP I asked earlier to have a look, this one is the old ip of drberg.com, as we get the following message from it

“The site may have moved to a different server. The URL for this domain may have changed or the hosting provider may have moved the account to a different server”.



Table 1.9: Gathering drberg network and hosting information using dnsdumpster

Hostname	IP Address	Type	Reverse DNS	Netblock Owner	Country	Tech / Apps	HTTP / Title	HTTPS / Title	FTP / SSH / Telnet	HTTP Other
drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare	cloudflare		cloudflare
m232.drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare			cloudflare
test2.drberg.com	185.72.157.106	A		WESTHOST	United States		Apache	CN: test1.drberg.com	ftp: 220 ----- Welcome to Pure-FTPd [privsep] [TLS] ssh: SSH-2.0-OpenSSH_7.4	
server4.drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare			cloudflare
klisana.drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare			cloudflare
blog.drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare			cloudflare
mail.drberg.com	204.13.209.66	A	66.204.13.209.reverse.enterehost.com	ENTERHOST-AS	United States					
helo.drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare			cloudflare
shop.drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare			cloudflare
testm2shop.drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare			cloudflare
app.drberg.com	185.72.157.106	A		WESTHOST	United States		Apache	CN: test1.drberg.com	ftp: 220 ----- Welcome to Pure-FTPd [privsep] [TLS] ssh: SSH-2.0-OpenSSH_7.4	
testapp.drberg.com	185.72.157.106	A		WESTHOST	United States		Apache	CN: test1.drberg.com	ftp: 220 ----- Welcome to Pure-FTPd [privsep] [TLS] ssh: SSH-2.0-OpenSSH_7.4	
alliances.drberg.com	159.253.145.145	A	91.91.159.145.reverse.com	SOFTLAYER	Netherlands	CentOS Apache:2.2.15			ftp: 220 (vsFTPd 2.2.2) ssh: SSH-2.0-OpenSSH_5.3	
unfriendedfromdiet.drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare			cloudflare
test.drberg.com	159.253.145.145	A	91.91.159.145.reverse.com	SOFTLAYER	Netherlands	CentOS Apache:2.2.15			ftp: 220 (vsFTPd 2.2.2) ssh: SSH-2.0-OpenSSH_5.3	
www.drberg.com	104.26.8.179	A		CLOUDFLARENET	United States		cloudflare			cloudflare

## 1.1.5 Findings and Recommendations

### 1.1.5.1 Findings 01: On IP behind the CloudFlare and the reverse proxy security

1. drberg.com is found to be under CloudFlare reverse proxy: 104.26.8.179 and 104.26.9.179
2. I have revealed the real ip behind this CloudFlare using several bypassing tools. The following ips found to have 99% structural similarities with drberg.com {23.20.111.103 and 18.212.199.43} and the ip 185.72.157.106 found to have 94% structural similarities with the site.
3. The bypass tool reveals so many subdomains and real ip of drberg.com (see Table 1.9), which means the further DDoS and XSS attack cant be prevented with their current setting. This reveals a poor CloudFlare configuration for drberg.

### 1.1.5.2 Findings 02: On drberg AWS architecture (see Figure 1.4)

1. The aws found to have 4X EC2 instances (in us-east-1d), 2X RDS databases (drberg:1590GB->us-east-1a and database-new:100GB->us-east-1f), 2X ElastiCache instances in (Primary/Replica) architecture, 6X EFS EndPoints in 6 us-east-AZs, 2X ELB -> {us-east-1d,us-east-1a} which found to be poorly configured.
2. EC2 Intances {1,2,3} pointing to the same drberg subdomain: **shop.drberg.com** and EC2#4 is the old drberg site running.
3. EC2#2-New prod-inst-current-production has 0.62 load on RDS#drbeg where EC2#1 has load factor 0.38 on it. Means, the major EC2 player of drberg is 23.20.111.103(EC2#2).
4. From the findings of CloudFlare bypass we got to know that, 2X ip has 99% structural similarities with drberg.com and those two are linked to the following EC2 instances:
  - 23.20.111.103 -> EC#2: New prod-inst-current-production
  - 18.212.199.43 -> EC#2: prod1-MVP-ebc-optimized1.1-23-12-2019
5. EC2#3:NewUX\_ec2\_clone is found to be have 0.47 load factor on RDS#Database-New<sup>11</sup>

<sup>11</sup>ITS THE EVIL DAD... WE WILL FIND SOON...THE REAL DATA SUCKER!!!!

### 1.1.5.3 Findings 03: From CloudWatch Network In/Out Bytes: EC2, RDS, ElastiCache

In Figure 1.5, I had analyzed the data for the last 3 days in the period of 19/4/2020 to 22/4/2020.

1. EC2 Network In: Everyday at around 2.15 UTC, the EC2 instance (i-085e01e0193867cef: new prod int-current production in us-east-1d) reaches to a pick (1.51GB to 1.93GB network inflow) for the last 3 consecutive days (see Figure 1.5a).
2. EC2 Network Out: Similarity, everyday at around 2.15UTC, the same EC2 instance produce more Network Outflow(around 500 MB to 751 MB) (see Figure 1.5b).
3. The ElastiCache cluster has seen to have a maximum hike for the cahce-001 NetworkBytesOut Primary Node, which is 121M on 20/4/2020 at 9.04UTC (see Figure 1.5c).
4. The RDS instances similar to EC2 Network-IN/Out, experienced a hike to 42MB at 2.15UTC (see Figure 1.5d).

**Note** - However these were using up data for sure, but the data spikes had no correlation with the consistent price increase. I did not believe this was the reason for high cost. *You will find this analytical report in CloudWatch -> Dashboards-> Problem\_debug.*

### 1.1.5.4 Findings 04: Did I find any inconsistant Data Flow?

1. In section 1.1.4.2, I didn't find any inconsistent data flow in between 19-22 Apr,2020. The data flow was in the category "EC2-Other", means within the Different AZ data transfer and consistently remain from 1024GB to 1800GB and the cost had gone up from \$0.2/day to \$10-\$17 per day since 25 Dec,2019.

### 1.1.5.5 Findings 05: On VPC Log Analysis using CloudWatch

1. Bidirectional data synchronization found among EC2 instances, RDS, ElastiCache and EFS.
2. Hence, all of these are in **Same Region-Different AZs**- bidirectional data flow charges applies in most of the cases.
3. My inspection of VPC flow log over a period of 3 days, reveals **HUGE...HUGE!!!** data flows from the followings (see Table 1.7):
  - RDS02(172.31.55.52,us-east-1f)->EC#3(172.31.93.238,us-east-1d) around 2.72TB @\$0.010 per GB X2
  - RDS01(172.31.17.213,us-east-1a)->EC#1(172.31.94.191,us-east-1d) around 2.25TB @\$0.010 per GB X2

- RDS01(172.31.17.213,us-east-1a)->EC#2(172.31.81.143,us-east-1d) around 1.79TB @\$0.010 per GB X2
  - Redis-Cache-001(172.31.43.114,us-east-1b)->EC#1(172.31.94.191, us-east-1d) around 428.49GB @\$0.010 per GB X2
  - EC#2(172.31.81.143,us-east-1d) -> RDS01(172.31.17.213,us-east-1a) around 78.3GB @\$0.010 per GB X2
  - EC#3(172.31.93.238,us-east-1d) -> RDS02(172.31.55.52,us-east-1f) around 21.4GB @\$0.010 per GB X2
  - EC#1(172.31.94.191, us-east-1d) -> RDS01(172.31.17.213, us-east-1a) around 13.9GB @\$0.010 per GBX2
  - EC#1(172.31.94.191, us-east-1d) -> Redis-Cache-001(172.31.43.114, us-east-1b) around 7.40GB @\$0.010 per GB X2
  - EC#2(172.31.81.143, us-east-1d) -> **External IP(18.233.43.190)** around 7.04GB
  - ip:{104.26.9.179, CloudFlare Reverse Proxy} -> EC#3(172.31.93.238,us-east-1d) around 6.79GB
  - ip:{104.26.8.179} -> EC#3(172.31.93.238,us-east-1d) around 6.55GB
4. The major data transfer is found among the EC2#{1,2,3} instances, RDS and ElastiCache. But the interesting fact is the data transferred between RDS#2-Database-New and the EC2#3-NewUX\_ec2\_clone.
  5. An unknown IP **18.233.43.190** found to receive around 7.04GB data over the past 03 days from EC2#3(172.31.93.238- NewUX\_ec2\_clone). This IP found to have Nginx server running as inspected with tool **whatweb**.

#### 1.1.5.6 Findings 06- Major misconfiguration found in the drberg aws architecture

1. The different AZs for EC2, RDS and ElastiCache is the major cause of this huge data transfer cost. Though the data transfer rate found to be in TeraBytes, but with the same AZs configuration, these transfer may not incur any additional BW cost.
2. The SecurityGroup (SG) for EC2 and RDS has NFS:2049 protocol access. I also have found the data transferring from RDS to EFS, which is not a standard design. The purpose of EFS is completely different and RDS access to it is not necessary.
3. The SG of EC2 and RDS has a non VPC public ip access: 198.255.52.130/32 which I thought to be a Client Application sucking the data. But from my cloudwatch observation and VPC flow log analysis, I didnt find any data transfer record of it.

### 1.1.5.7 Recommendations

1. Magento2 of drberg is not the root cause of this huge data transfer flow, rather the data flows as in Table 1.7 are the key players.
2. Due to my ssh access limit, I would request to check the cron for the following two EC2 instances: EC2#2(new prod inst- current production) and EC#3(NewUX-ec2-clone) under both the webserver user and the root user.
3. If possible, try to install IPTraf and iftop tool in all the 4X EC2 instances to trace the activity of any API call for this continuous **HUGE!!!** data transfer.
4. May be the source code of drberg has been managed improperly, which might results in such calls and syncs. I request the drberg team to think about the scale when writing the code (frontend and backend).Think about what might happen when you are processing a million transactions each second. Prepare your code to handle the worst cases. The best cases would be handled automatically.
5. Furthermore, astonishingly the drberg team is not using the AWS ElasticSearch, while they are using the AWS ElastiCache and RDS services. One question remains.... Do they really using ElasticSearch at all? Probably .. **NOT!!!**

### 1.1.5.8 Proposed Cost to take over the drberg site

Feature	Use it (Yes/No)	Cost(\$)
AWS EC2	Yes	575
AWS RDS	No	0
AWS ElastiCache	No	0
AWS ElasticSearch	No... drberg site is still not using it and may be they dont have any internal ElasticSearch setup too for the site	
Probable AWS Data Transfer Cost	Hence, we will not be using the AWS RDS and AWS ElastiCache as those are used in current drberg architecture and thereby no hazard and dilemma of multi AZ data transfer, the probable data transfer cost will fall down to around 30 USD, because the overall data transfer cost of HWW is around 65 USD, which is hosting more than 50 sites.	<30
Backup/Monitoring/Others		100
Total		705 USD

### 1.1.5.9 Learning

1. Have different tools at your disposal which can help you out when the need arises. Every data point in situations like these is helpful. I don't use CloudFail and CloudFlair during my everyday tasks. Eg: drberg didn't have VPC flow logs running on their VPC from before and that ate up some time during debugging. If I didn't have it, it would have taken much more time to get to the root cause of the issue.

**1.1.5.10 Tools used, in chronological order**

1. AWS Cost Explorer
2. AWS VPC Flow logs
3. AWS CloudWatch
4. dig
5. whatweb
6. traceroute
7. whois
8. CloudFail
9. CloudFlair
10. censys.io api
11. dnsdumpster