influxdata

Why Architecting for Disaster Recovery is Important for Your Time Series Data

AN INFLUXDATA CASE STUDY

Rajeev Tomer

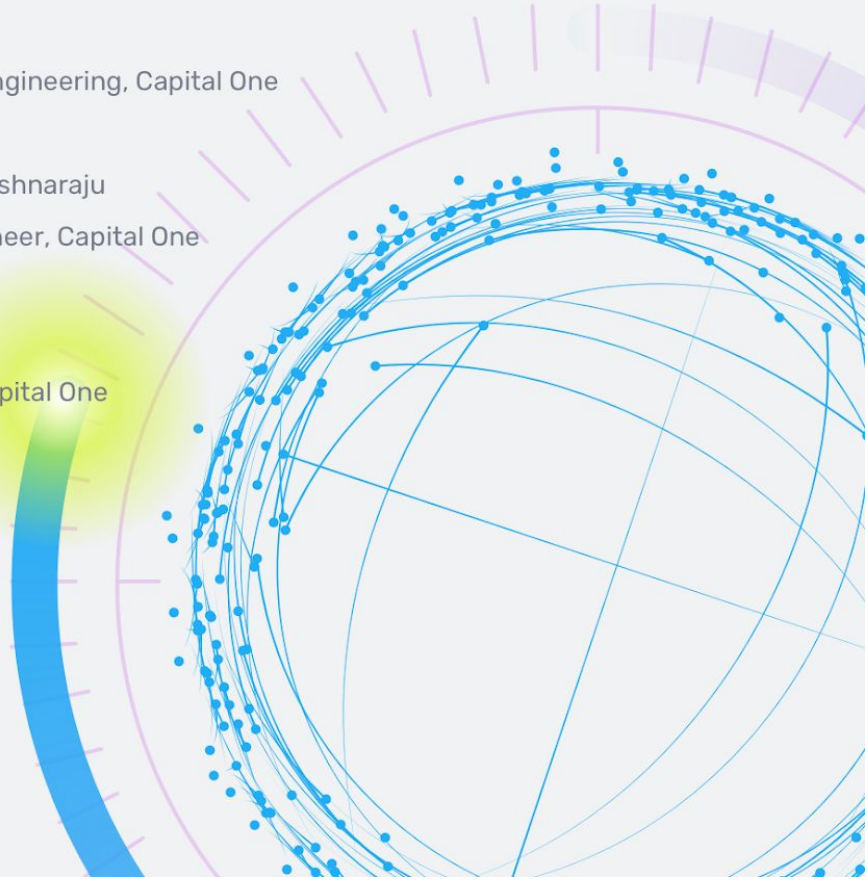
Sr. Manager of Data Engineering, Capital One

Saravanan “Krisha” Krishnaraju

Master Software Engineer, Capital One

Karl Daman

Software Engineer, Capital One



September 2018

Company in brief

Capital One Financial Corporation is a bank holding company specializing in credit cards, auto loans, banking and savings products headquartered in McLean, Virginia.

Capital One is a diversified bank that offers a broad array of financial products and services to consumers, small businesses and commercial clients. A Fortune 500 company, Capital One has one of the most widely recognized brands in America.

As one of the nation's top ten largest banks based on deposits, Capital One serves banking customers through branch locations primarily in New York, New Jersey, Texas, Louisiana, Maryland, Virginia, and the District of Columbia.

Case overview

Time series data at Capital One consists of Infrastructure, Application, and Business Process Metrics. The combination of these metrics are what the internal stakeholders rely on for observability which allows them to deliver better service and uptime for their customers, so protecting this critical data with a proven and tested recovery plan is not a "nice to have" but a "must have."

The IT team at Capital One use InfluxDB to store and visualize all their business, infrastructure, and application metrics that are visualized in Grafana.

They built a fault-tolerant solution with full disaster recovery capabilities, based on InfluxDB Enterprise and AWS, that collects and stores metrics and events. They also use InfluxDB with their machine learning (ML) framework, whereby the collected time series is used to model predictions which are then brought back into InfluxDB for real-time access.

> *"The solution should be for a purpose. If you want to do this kind of thing [manage time series data], InfluxDB is the best for it."*
>
> ***Rajeev Tomer**, Sr. Manager of Data Engineering*

The business problem

Capital One relies on its system metrics to maintain system performance visibility and meet its SLA's in the context of company — and therefore data volume — growth. Since its metrics consist of various types of time series data, Capital One needed to sustainably store, manage, and protect this data across a variety of use cases and regions.

The company has a centralized cluster of InfluxDB for everyone in the organization to store and manage their time series data. At Capital One, InfluxDB was already used for multiple types of metrics:

- Business Transaction Metrics - Used to monitor business processes to understand performance, as well as associated metrics that report and alert on events (such as user volume changes, number of transactions)
- Infrastructure Health metrics - Encompassing typical infrastructure health metrics (such as CPU and memory)
- Application Performance Metrics - Using InfluxDB to assess application performance family (determining whether it's a web, database, or application tier); all application metrics are sent to InfluxDB, and from there, they drive their line chart and alerting for testing.
- Service Adoption Metrics - Monitored to show how well services are being adopted, how much value they are providing to the organization, and how they are performing.

Given that context, Capital One needed to solve the following business challenges:

- Achieve resiliency across multiple regions to protect the various types of time series data that the company's internal stakeholders rely on for observability.
- Architect InfluxDB for high availability since as a bank and credit card company, Capital One services must be highly available to their customers, which in turn means that the Network Operations Center (NOC) team must always have access to their metrics.
- Model existing metrics (time series data) with their machine learning framework to enable forecasts and respond proactively.

The technical problem

The Capital One IT team set out to build an architecture that would meet their business objectives. The technical challenges they faced, such as high data retention and an unstable disaster recovery

solution, are best encapsulated in outlining the technical journey they undertook to build that architecture.

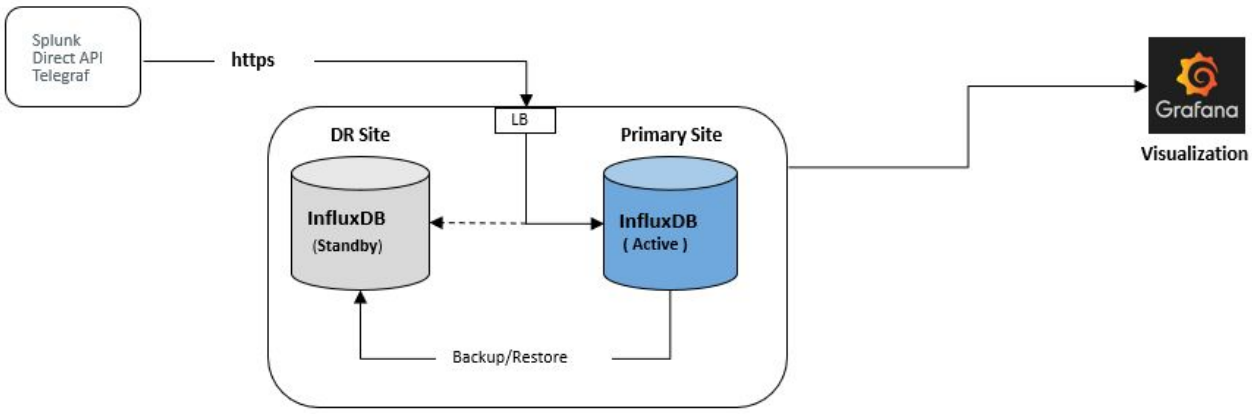
Technical architecture

> *"InfluxDB is high-speed read and write database. So think of it. The data is being written in real-time, you can read in real-time, and when you're reading it, you can apply your machine learning model. So, in real-time, you can forecast, and you can detect anomalies."*
>
> ***Rajeev Tomer**, Sr. Manager of Data Engineering*

Designing and building Capital One's solution comprised a journey involving three versions of the solution leading up to their current version, as well as planning and executing on their disaster recovery plan.

How InfluxDB architecture at Capital One evolved

Architecture - Gen 1



As shown above, the first generation (Gen 1) architecture has a centralized cluster and a DR cluster. The DR cluster used InfluxDB's backup/restore capability (a check, find, data recovery capability for incremental backups, or for full backups to restore data in case of data loss). The backup is shifted to other sites and restored whenever needed.

The Gen 1 primary topology was as follows:

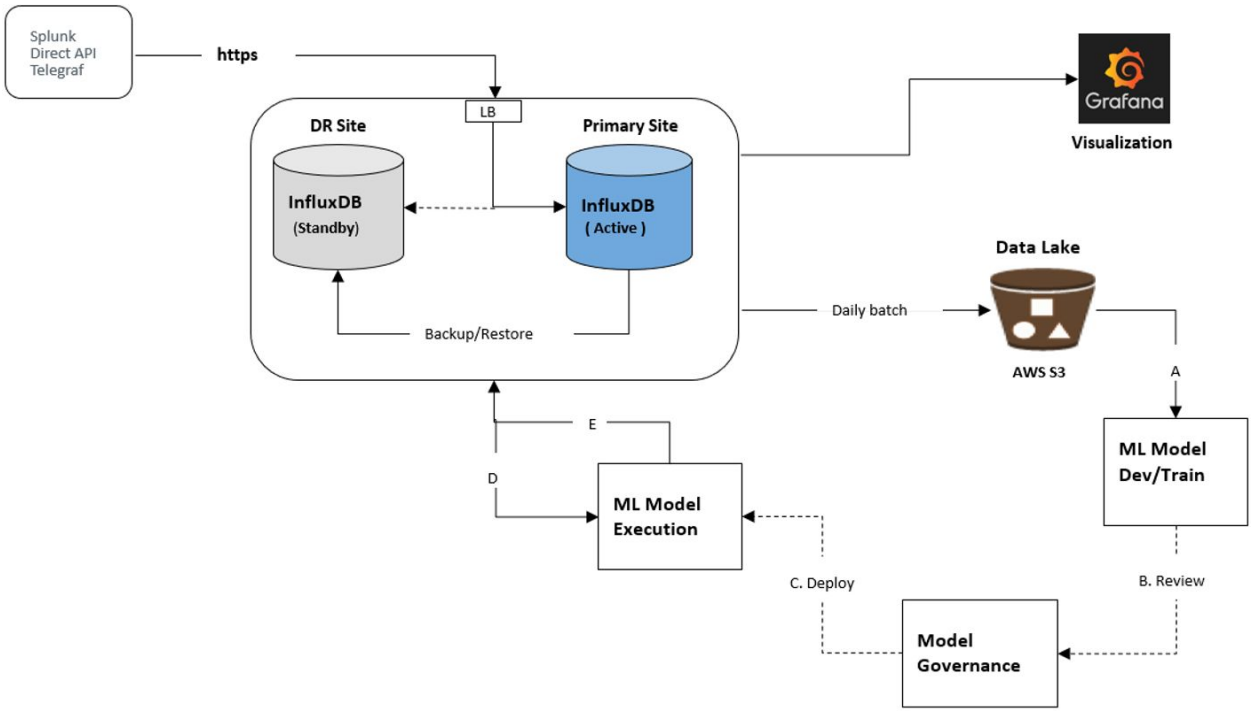
- The primary input came from Splunk, and Grafana dashboards visualized the metrics stored in InfluxDB.
- Splunk sent high data volume, and reporting was done using Splunk's line chart ability (to track value trends over time). That included InfluxDB to support the line dashboard.
- InfluxData's metrics collection agent Telegraf sent a direct API-based data load to InfluxDB.

Gen 1 of the architecture presented two challenges:

1. **Unsustainable high data retention (> 400 days):** Grafana was primarily used for visualization, full testing, anomaly detection, and forecasting. Data retention became challenging as the company grew. The 80/20 rule applied: they observed that 20% of recent re-inserted data was used 80% of the time, and the other 80% was hardly used.

2. **The challenge of Backup/Restore in the DR solution:** The backup/restore worked well if they had to use it for the primary site only, but when it comes to the primary site and then recovering the DR site, backup/restore was not proving sustainable. As the data size grew, it presented new challenges, as discussed below.

Architecture - Gen 2



For the generation 2 architecture (Gen 2), they continued using Grafana for visualization. They started thinking not only about data retention but also about the ecosystem around InfluxDB since databases and applications need to be integrated with other components. They considered how the data would be explored for the variety of time series use cases it is used for.

Solving the high data retention problem

Considering the ecosystem around InfluxDB helped solve the data retention problem:

- Raw data was exported daily to a number of data lakes and stored there.
- Capital One data lakes are an AWS S3 dish and give all their users the capability to perform analysis on structured as well as unstructured data.
- These data lakes have become an online storage for InfluxDB, whereby 20% of the data is stored in InfluxDB, and the rest is stored in the Data Lake, where it can be used for a variety of other uses, such as machine learning.

Modeling machine learning predictions using InfluxDB and data lake

Capital One uses the data lake and InfluxDB to build machine learning models. The model works as follows:

- Since the data is copied to the data lake in the same format as InfluxDB, their analyst data scientist can double up a machine-learning model (to enable forecasts) using their Data Lake data.
- When the model is ready, by using the infinite history available in Data Lake, it goes to the Model Governance.
- Then the algorithm is ready to execute and will apply in real time on InfluxDB.
- That machine learning model is deployed using their means, not on top of real-time InfluxData.

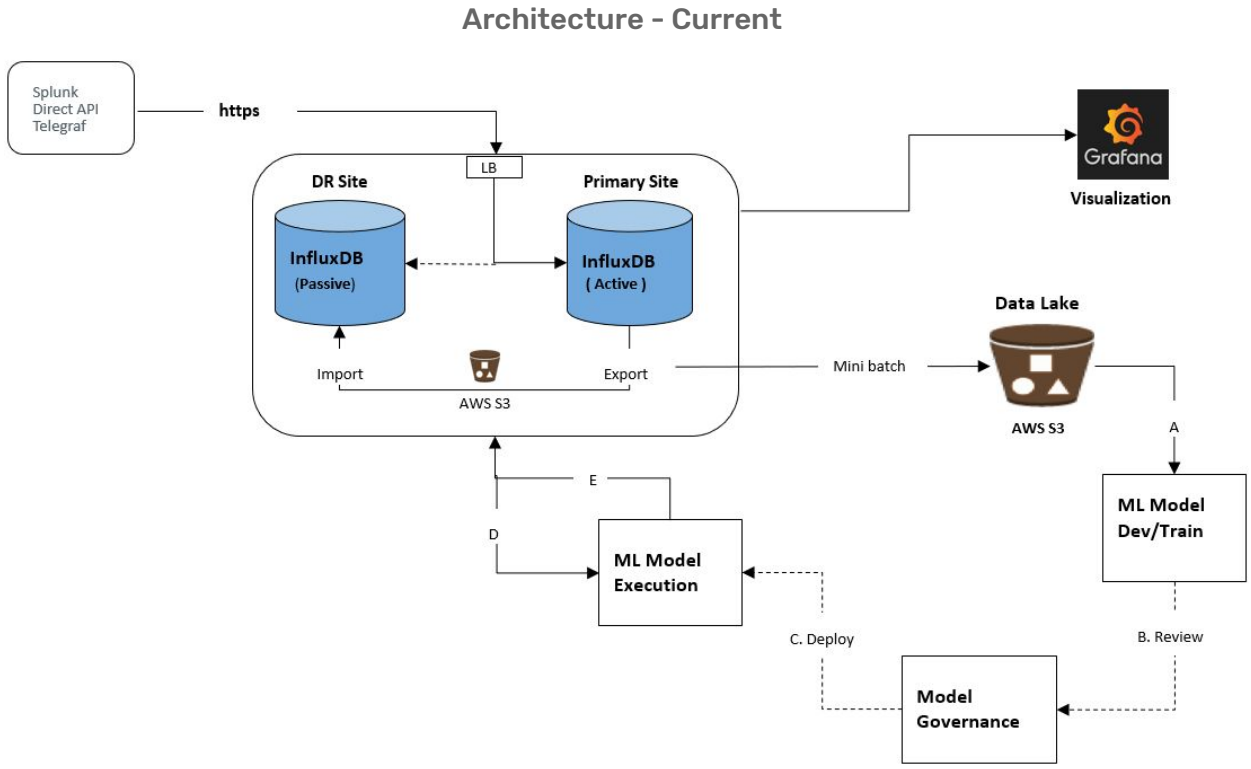
Since InfluxDB is a high-speed read and write database, the machine learning model could be applied in real time to forecast and detect anomalies in real time.

Yet the backup/restore issue was still a problem, and as such the DR solution was still unstable. Though on the surface, their backup/restore solution seemed to be working well, it gave rise to the following challenges:

- One challenge, besides longer backup/restore duration (due to data volume growth), was the lack of an incremental backup. Since there is no such thing as an incremental cluster, they needed to build an empty cluster and do a full restore. Depending on the data they had, this required a variable amount of time.

- Another challenge was the specific proportion. If they were using v1.5.2 and below, they faced some issues. When the backup/restore would fail due to anti-entropy, they had to clone out the anti-entropy, make a good backup, and restore.

They did overcome these operational challenges initially, did a dry run and everything seemed to be working fine. But the DR solution was still an unstable DR solution. They were betting on the backup working properly, evaluating their DR solution every three to six months, and having to wait until the next exercise to ensure both sites are restored. Ultimately, they decided to design a process solution to address this problem (as shown in the current architecture diagram below).



Building a stable DR solution

In their current architecture, which has proven to be very robust, they were able to solve the unstable DR Solution:

- Raw data is exported every 30 minutes to the Data Lake and is also available for ML.
- They replaced the backup/restore with an InfluxDB export/import script and leveraged AWS S3 solution to move the exported data files from the primary site to the DR site.

- The DR site is not standby (not an empty cluster) anymore. It is now a Passive cluster that is survivable for importing the data from the primary site.
- The team controls who can read and write via a load balancer. All reads/writes directly go to the primary site (as the dotted line to the DR site designates). This enabled them to ensure that their DR side of the cluster is available and that they can monitor, on a real-time basis, to ensure their DR site remains functional.
- They can switch sites at any time, which wasn't possible before with the standby cluster. Now they export all the database, transporting data via AWS S3 to the DR site and importing it back, thereby eliminating DR solution instability.

Achieving resiliency across multiple regions

Capital One needed complete protection against an entire region failure and leverages the following AWS resources to achieve resiliency in its architecture:

- Route53, which acts like a DNS switch
- Elastic Load Balancer, which automatically distributes incoming application traffic across multiple targets
- AWS S3, an object-based storage service where any of the AWS services can be accessed to store and retrieve dedicated data

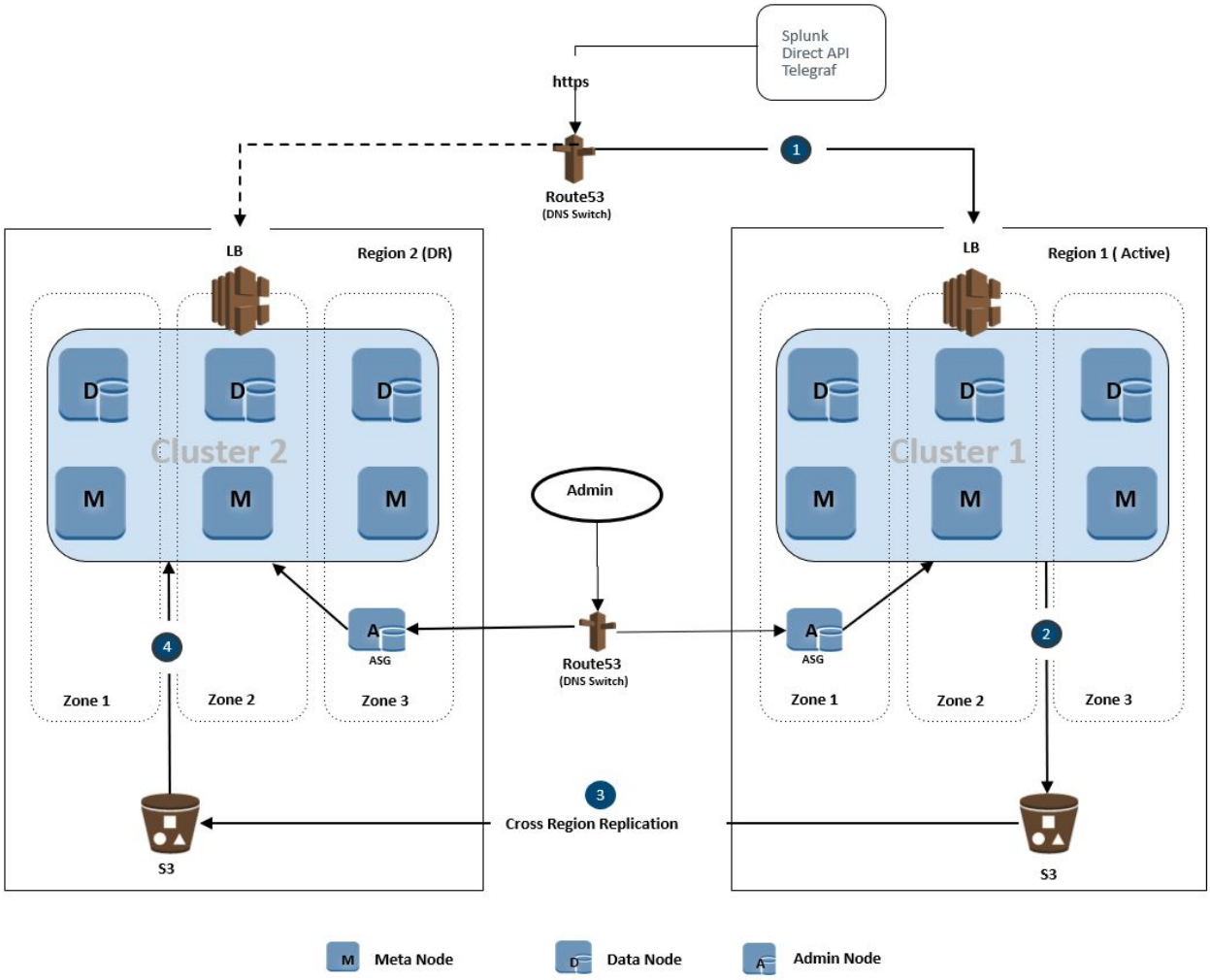
Shown below is the Region 1 and Region 2 architecture (with AWS, Capital One has data centers across the continental United States and the globe):

- The region refers to a geographical location where Capital One has multiple zones.
- Zones refers to physical data centers (such as physical buildings), where zones are separated by buildings, power, and other infrastructure components.
- The zones are connected with low-latency lines, resulting in very high response times.

In Region 1 architecture, the data flow is as follows:

- All traffic is routed to Region 1 .
- InfluxDB Export Script runs every 15 minutes.
- Data is replicated to Region 2.
- InfluxDB Import Script runs every 15 minutes.

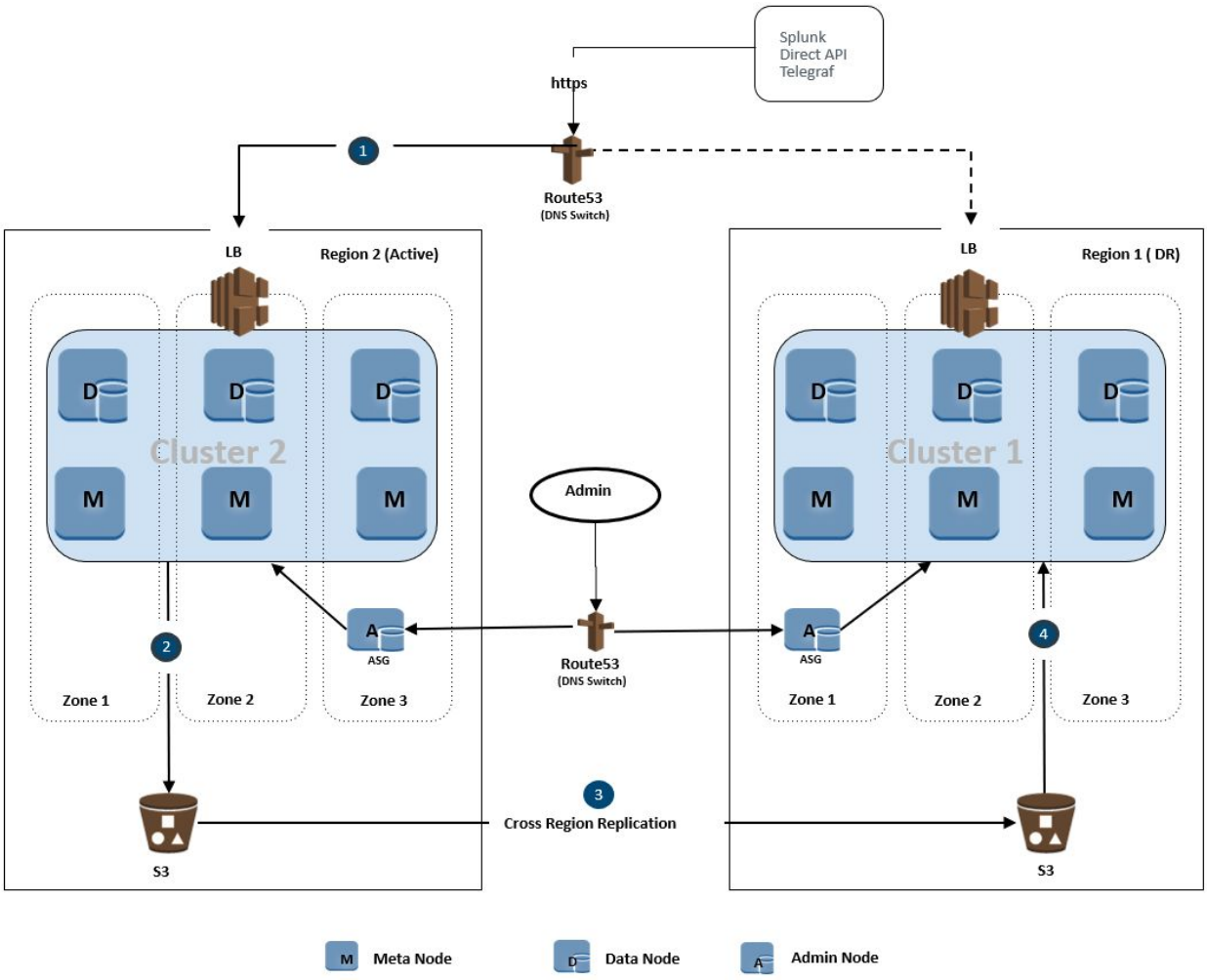
Resiliency - Region 1 Active



In the Region 2 architecture, the data flow is as follows:

- All traffic is routed to Region 2.
- InfluxDB Export Script runs every 15 minutes.
- Data is replicated to Region 1.
- InfluxDB Import Script runs every 15 minutes.

Resiliency - Region 2 Active



How Capital One architectured InfluxDB for high availability

InfluxDB Enterprise 1.6.2 is used on all the nodes (the latest version at the time), and Capital One plans to keep using newer versions as they are released.

In the resiliency architecture shown above:

- Three data nodes are accompanied by three meta nodes.
- The three data nodes are put in different availability zones leveraging AWS high-end infrastructure, so that even if two AWS nodes are lost, Capital One customers will still be able to retrieve read/write.
- A home-grown admin tool is used.
- They have the same setup on DR side regions: three data nodes and three meta nodes.

Data flow and data import/export

They have built in some logic so at any given time, all the traffic is automatically routed to the load balancer that's under Region 1. For Region 2, they have written an export code that leverages InfluxDB, aims the command to export data from all databases, and puts it to a S3 bucket. They leverage one of AWS current features — cross-region replication — so that the data is totally off their site and is taken care of by AWS.

They have another code which runs on the DR site, reads all the files, and imports that data into this structure. The cycle runs every 15 minutes. At any given time, the cluster in the DR region is 15 minutes behind the primer, which is well within Capital One's established SLA and very well within all regulatory times.

Export*

```
# Declare Variables
DURATION=1800 # In secs

# Set the start and end time ...
STARTTIME=LASTENDTIME
ENDTIME=STARTTIME + DURATION

# Get the db list
DBLIST=$(influx -execute 'show databases')

#  Export the DB
for DB in $DBLIST; do
  influx_inspect export -compress -database $DB -start $STARTTME -end $ENDTIME -out $DB-$ENDTIME.gz
  aws s3 move $DB-$ENDTIME.gz s3://<YOURBUCKET>/export/
done

# Update the time for the next run
ENDTIME=LASTENDTIME
```

Import*

```
#!/bin/bash

# Download the files from S3
aws s3 copy s3://<YOURBUCKET>/export/ $LOCALDIR  --recursive

# Get the list of files
FILES_TO_IMPORT=$(aws s3 list s3://<YOURBUCKET>/export/)

for FILENAME in $FILES_TO_IMPORT; do
  influx -import -compressed -path=$LOCALDIR/$FILENAME
  aws s3 delete s3://<YOURBUCKET>/export/$FILENAME
done
```

* High level pseudo code , Not the actual code

An export script first runs on the Region 1 data node. As long as all the data is replicated, then they can use one of the nodes. So they use one node and set the `DURATION` (how often the script will run):

- Every time the script runs, it sets the `STARTTIME` equals the last `ENDTIME`, so that it always has a block that it operated on. `ENDTIME` will be `STARTTIME` plus `DURATION`.
- The first operation that the script performs is `'show databases'` so that files can be collected from each database individually instead of one big file.
- A `for` loop goes through each of the databases in the list and runs `influx_inspect` (an Influx tool that allows getting the data out in raw text file format).
- They extract each database and use the `-compress -database` feature to reduce file size. Then they just put `STARTTIME`, `ENDTIME` and choose file location.
- Next, the scripts puts the alpha file into S3 and does that for all the databases. Then it updates the `ENDTIME` with the last `ENDTIME`.
- This file goes into an S3 bucket that is using the AWS bucket replication, so that whatever files are put in it show up in the same named bucket or a different bucket in another region.

In Region 2, they have the DR cluster running, which receives the data from the above script. On one of the data nodes in Region 2, they have an input script running repeatedly that puts the data in the cluster:

- The script first copies all the files from that bucket in Region 2 and puts them in a local directory.
- Then they get the list of files and go through a `for` loop, and the script runs the `influx -import` which allows inserting data directly into the database.
- The script deletes the file in S3, and that is the command that terminates the state of the backup there.

Results

> *"If there is a disaster, then we can switch our clients to the other region, region two, and then it would be 30 minutes behind or whatever the timeline would be set."*
>
> ***Karl Daman**, Software Engineer*

By solving the high data retention and DR solution instability problems, Capital One has achieved a much more manageable ecosystem and maintained the visibility needed to solve its business challenges.

Performance Metrics Collected Using Telegraf

In the sample Chronograf dashboard above, Capital One captures Cardinality, Write HTTP Requests, Query Requests, Client Failures, Query Performance, and Write Points (all per minute). This works very well for monitoring performance metrics. Telegraf can be installed on many client servers to collect this data.

Lessons learned in their technical journey include:

- It's critical to keep an open source version of InfluxDB simply for monitoring the cluster that you have the Enterprise version on. They have Chronograf visualizing the metrics in InfluxDB, but going to a different InfluxDB so they can still monitor InfluxDB Enterprise even if it has an issue.
- Surrounding technology capabilities can be the key to choosing the right architecture: being in AWS, they could support InfluxDB export/import data by using S3. The whole technology solution was more available to InfluxDB export/import for DR versus backup/restore.

Powered by InfluxDB Enterprise and AWS, Capital One's solution is providing reliable disaster recovery while delivering better service and uptime for their customers.

About InfluxData

InfluxData is the creator of InfluxDB, the open source time series database. Our technology is purpose-built to handle the massive volumes of time-stamped data produced by IoT devices, applications, networks, containers and computers. We are on a mission to help developers and organizations, such as Cisco, IBM, PayPal, and Tesla, store and analyze real-time data, empowering them to build transformative monitoring, analytics, and IoT applications quicker and to scale. InfluxData is headquartered in San Francisco with a workforce distributed throughout the U.S. and across Europe.

Learn more.

InfluxDB documentation, downloads & guides

Download InfluxDB
Get documentation
Additional case studies
Join the InfluxDB community

799 Market Street
San Francisco, CA 94103
(415) 295-1901
www.InfluxData.com
Twitter: @InfluxDB
Facebook: @InfluxDB