# How Netlify Migrated to a Multicloud Architecture

And no one noticed

# Who am I?

@ry_boflavin

rybit

ryan@

# Who am I?

- Dog Dad

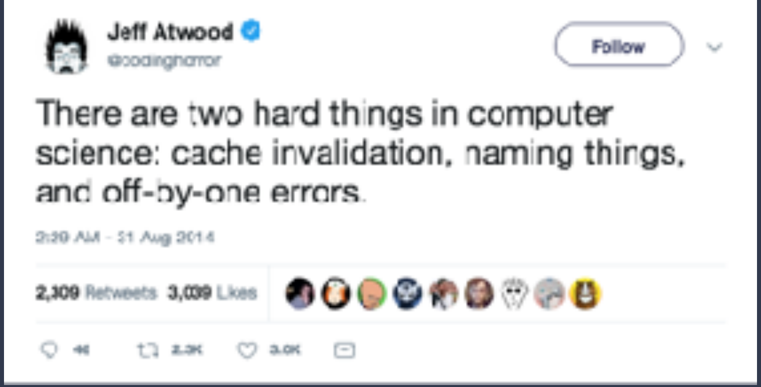
# Who am I?

- Dog Dad
- Engineer

# Who am I?

- Dog Dad
- Engineer
- Fire Spinner

# Engineer of things



Jeff Atwood
@codinghorror

There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors.

2:20 AM - 31 Aug 2014

2,309 Retweets 3,039 Likes

- Tech Passions
  - Distributed Systems
  - Streaming Data System
  - Infrastructure Automation
  - System Design
- Worked
  - Raytheon
  - Palantir Middle East
  - Yelp
  - Netlify

# What is Netlify?

*Netlify is the simplest way to build, deploy, and manage web projects on the JAMstack. We're changing the way the web is built by collapsing the modern front-end development process into a single, simplified workflow.*

- full CI/CD
- prerendering
- content delivery
- lambda deployment
- routing layer
- split testing
- identity provider
- dns provider
- ...

# What is Netlify?

*Over*

- *5 million sites*
- *4,000 requests/sec*
- *1,200 deploys/hour*

- full CI/CD
- prerendering
- content delivery
- lambda deployment
- routing layer
- split testing
- identity provider
- dns provider
- ...

# What is Netlify?

*Over*

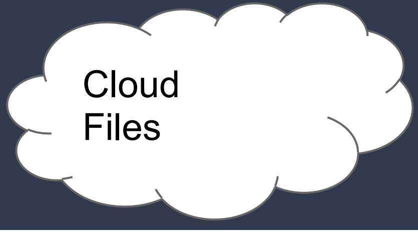
- *5 million sites*
- *4,000 requests/sec*
- *1,200 deploys/hour*

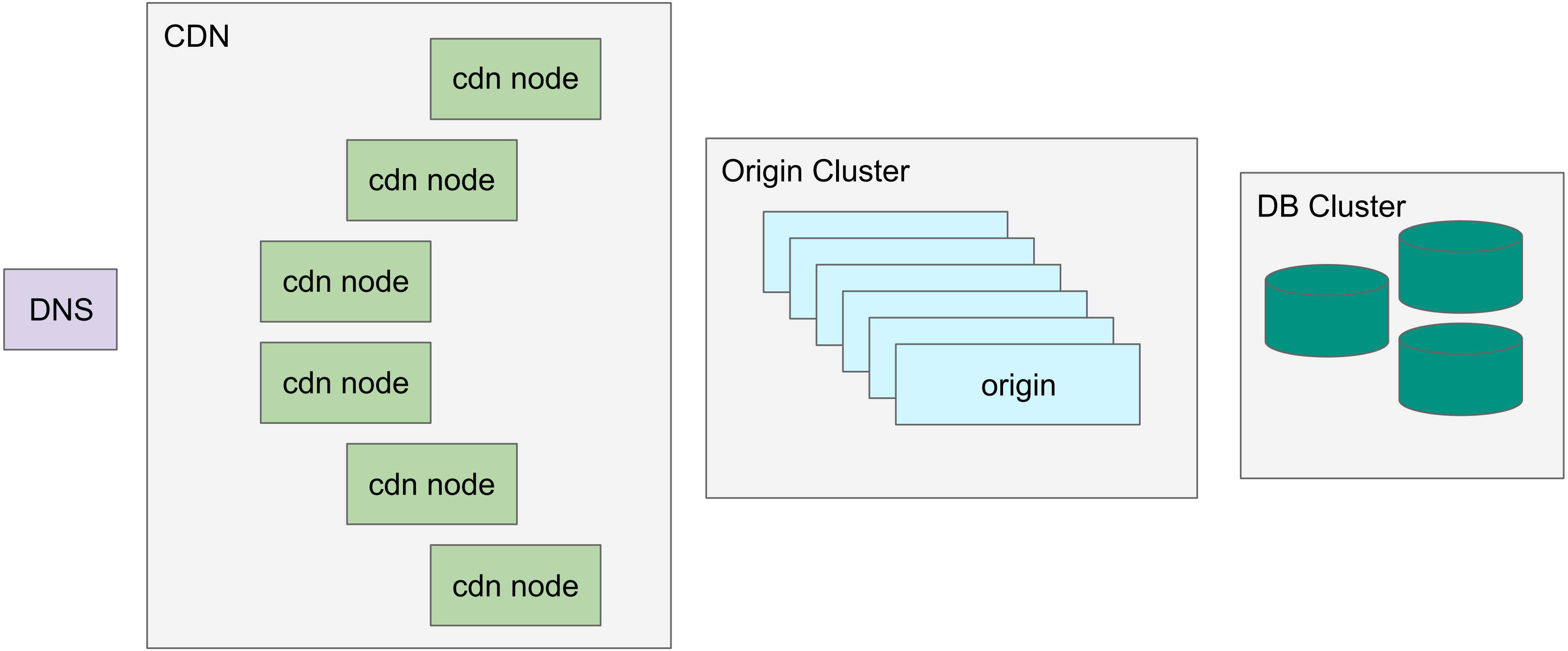
- full CI/CD
- prerendering
- content delivery
- lambda deployment
- routing layer
- split testing
- identity provider
- dns provider
- ...

# What am I going to talk about?

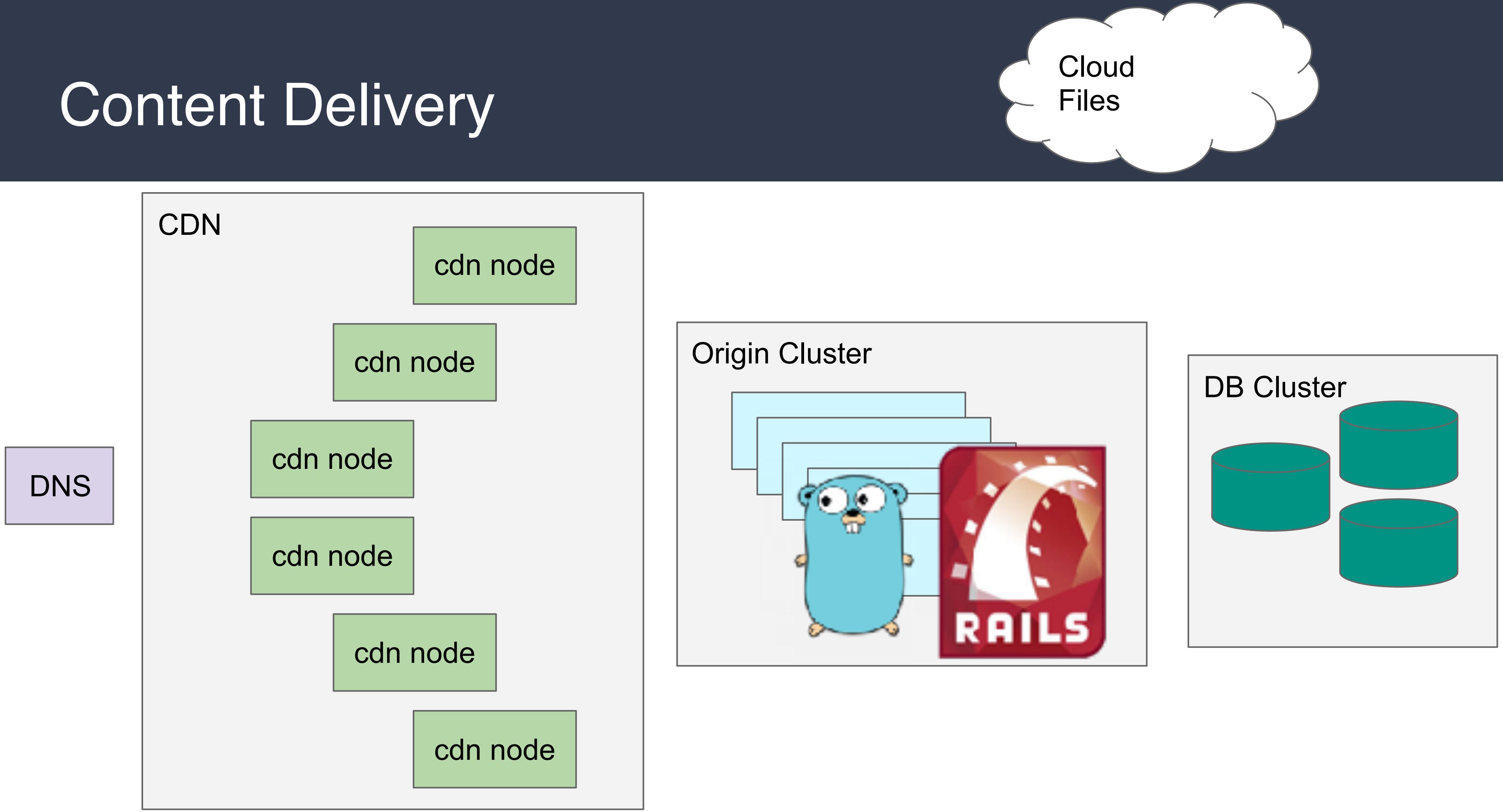
1. Intro to the system
2. Why we did all this work
3. How we accomplished it
4. The actual migration
5. Next steps

# Content Delivery

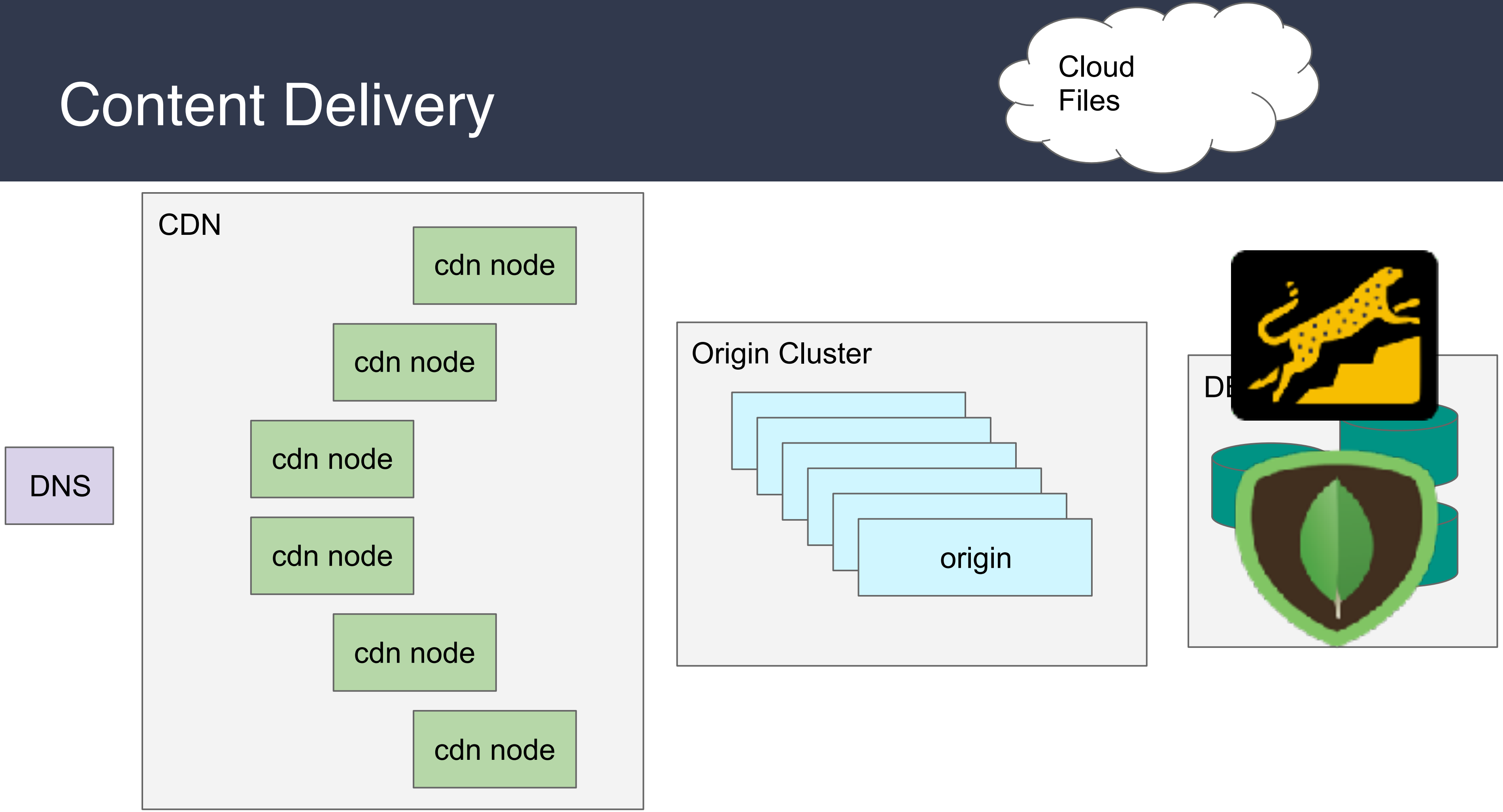




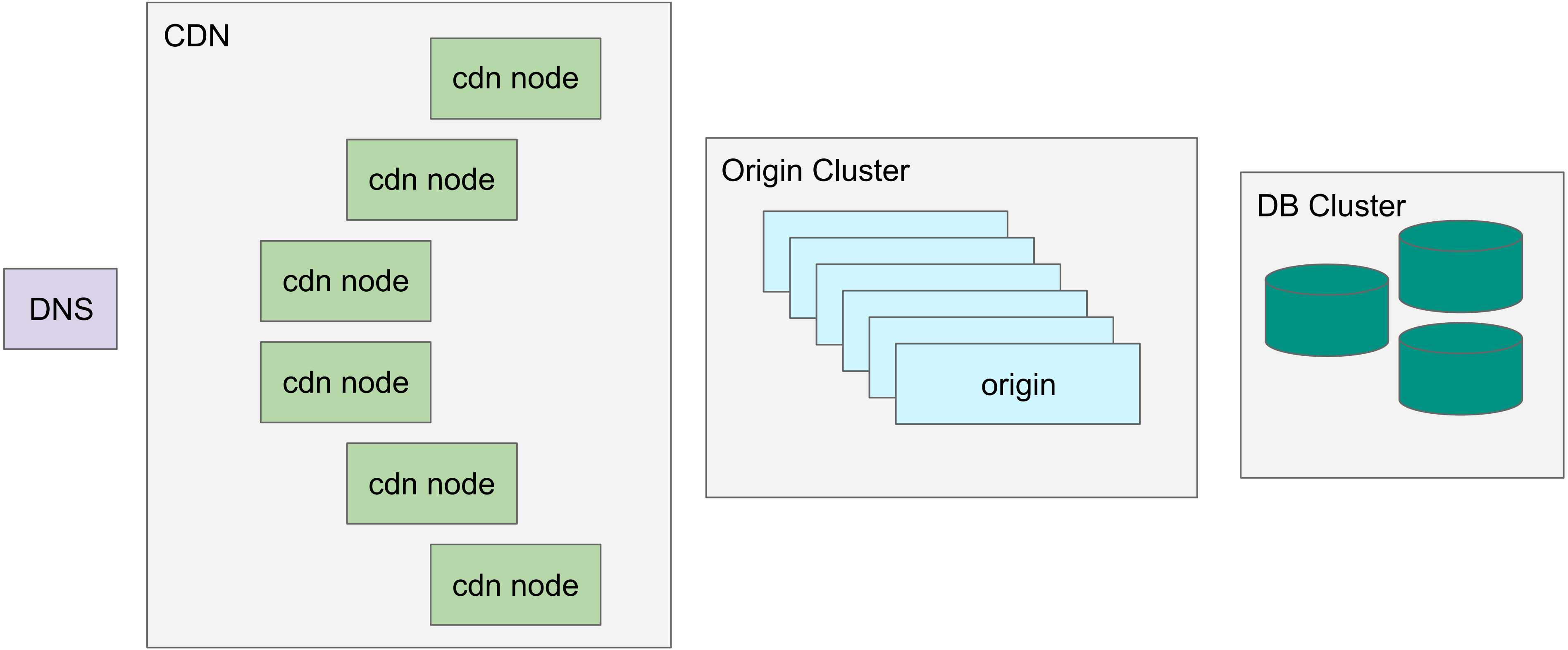
# Content Delivery



# Content Delivery



# Content Delivery



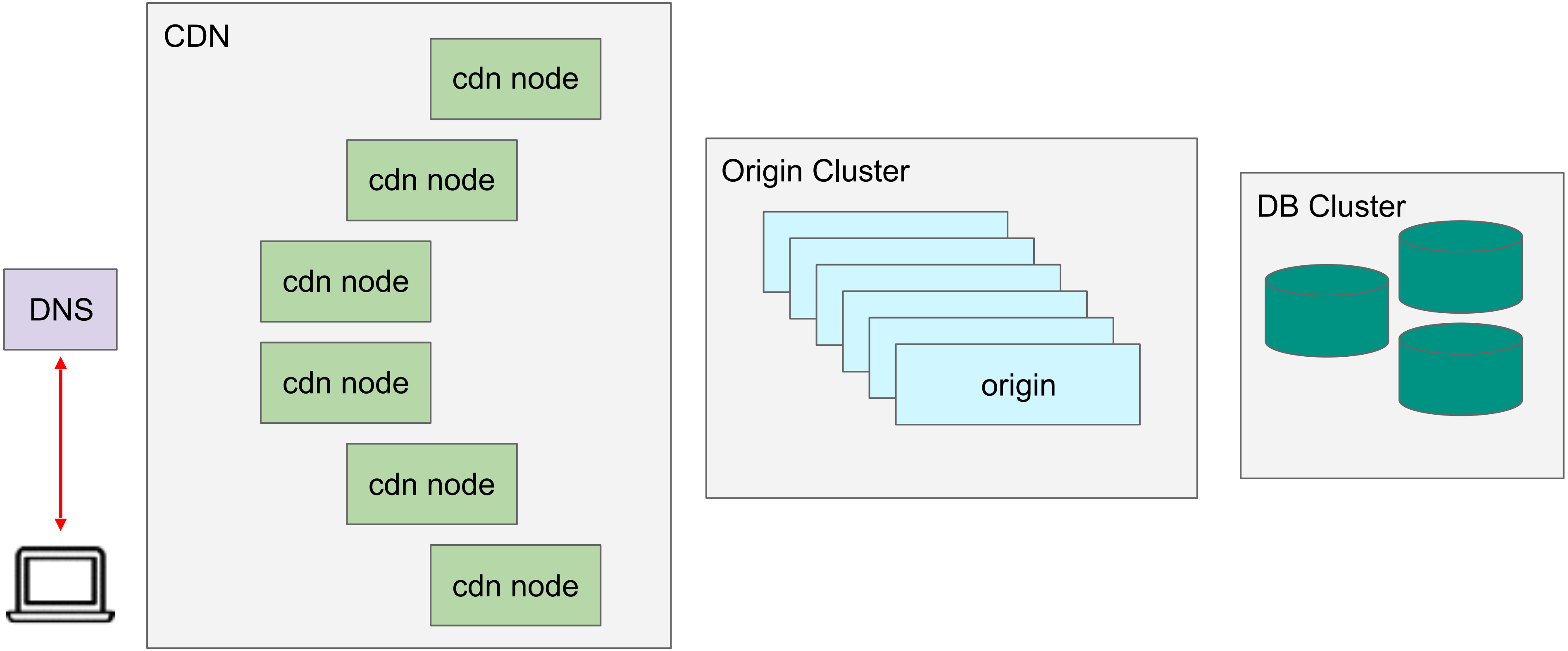
# Plan for failure

- Redundancy is a priority
- Everything is horizontally scalable
- Everything runs in cluster
- Health checking for everything

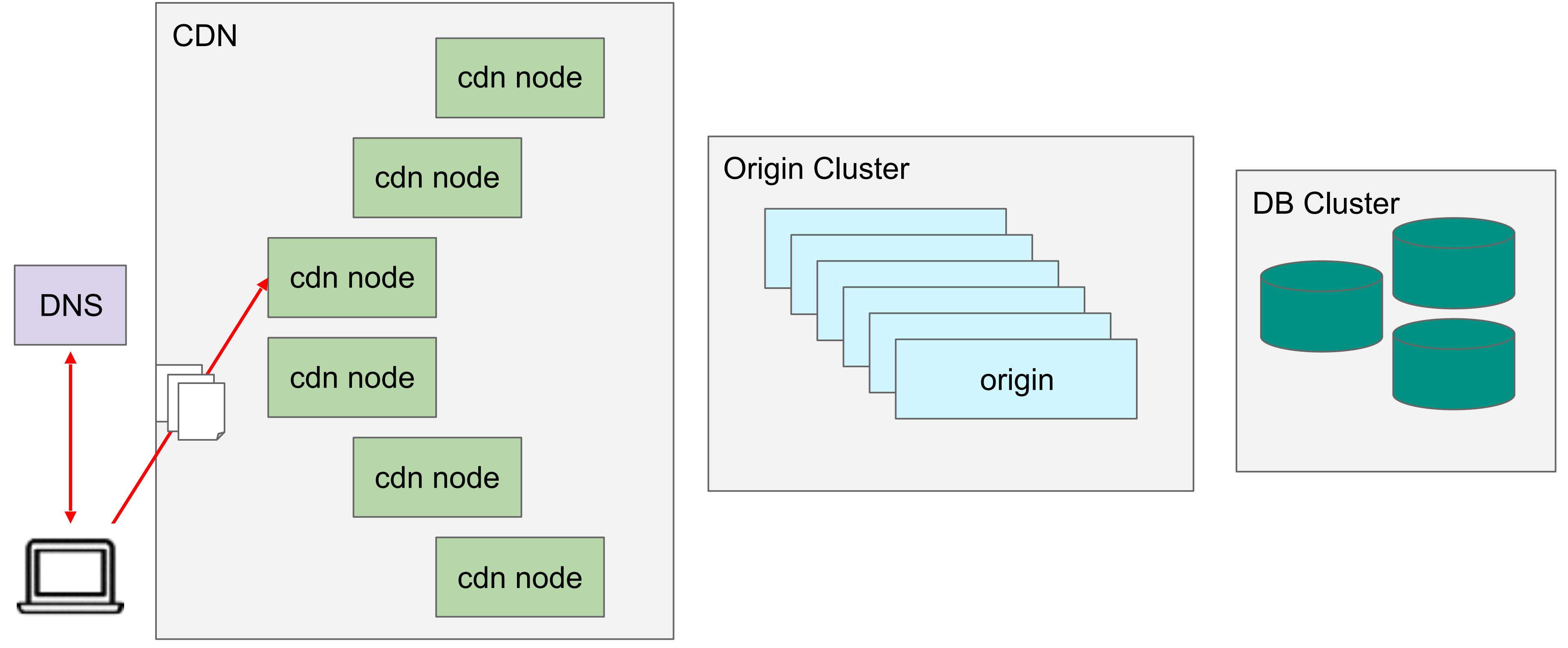
# Getting Data into the system

# Content Delivery

Cloud Files

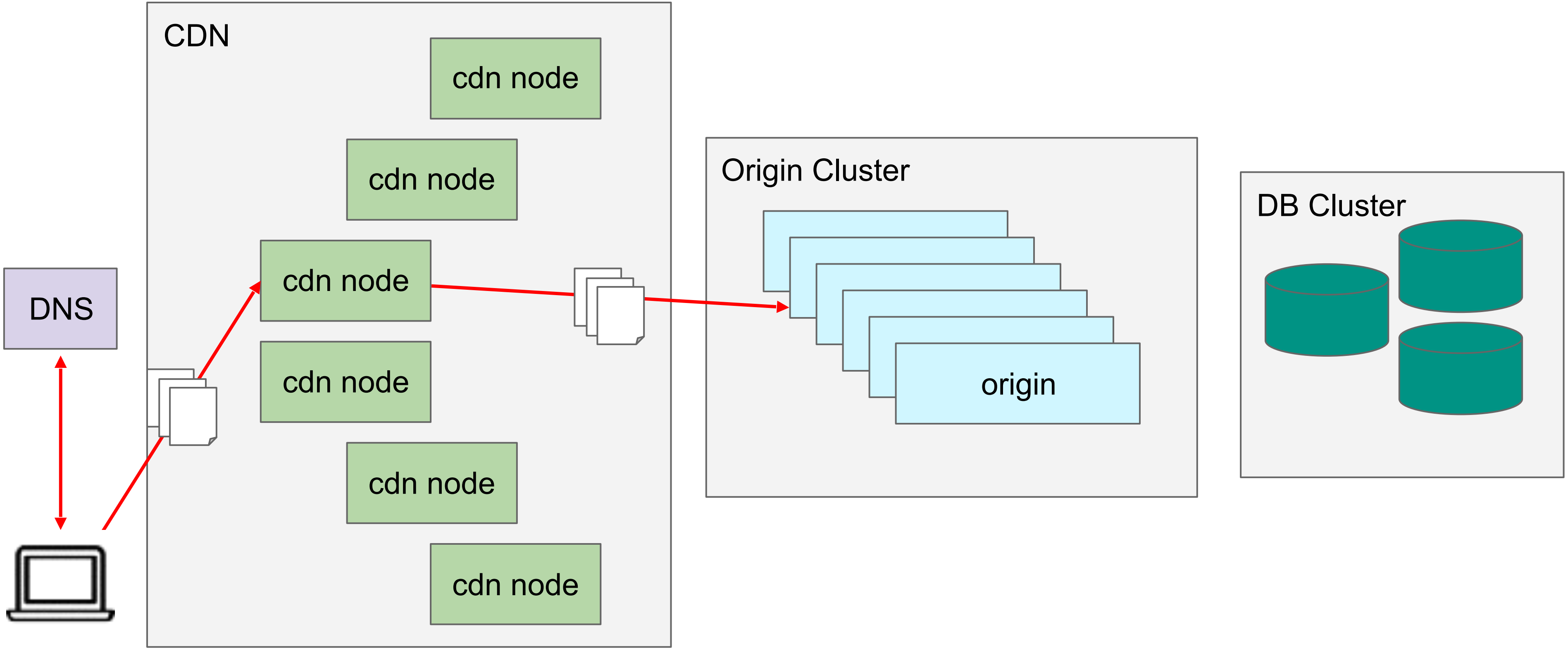


# Content Delivery



# Content Delivery

Cloud Files



DNS

CDN

cdn node

cdn node

cdn node

cdn node

cdn node

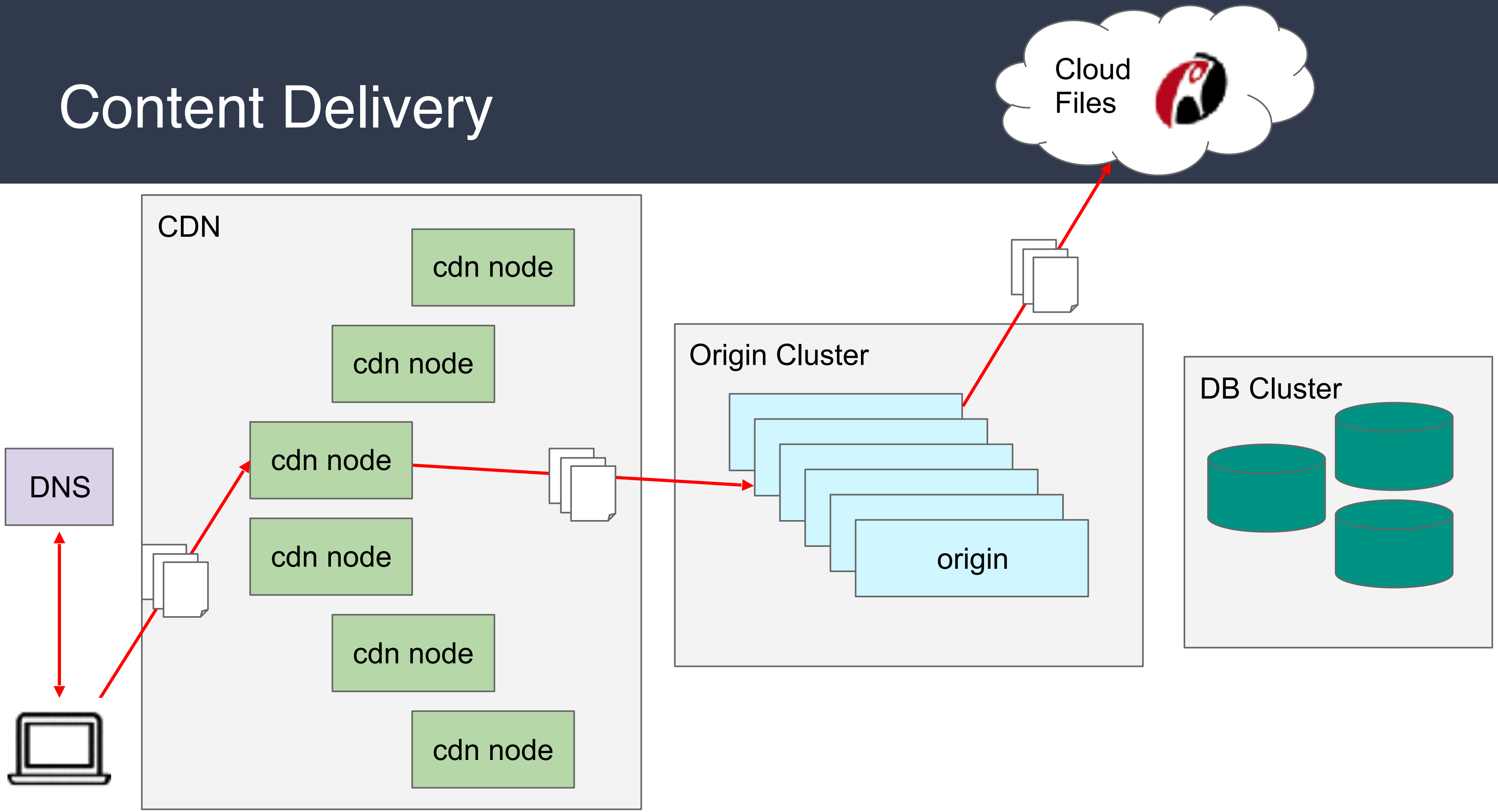
cdn node

Origin Cluster

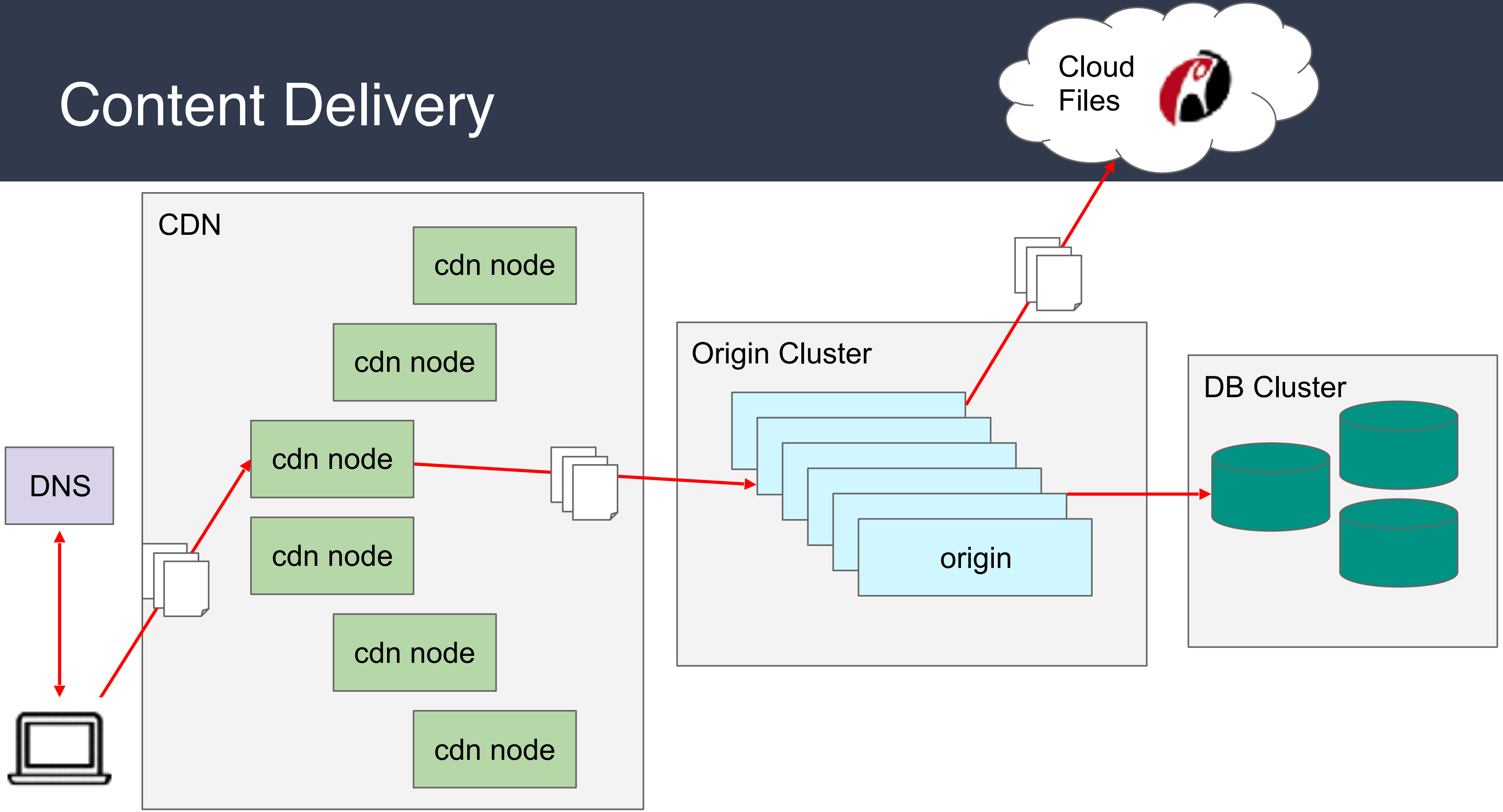
origin

DB Cluster

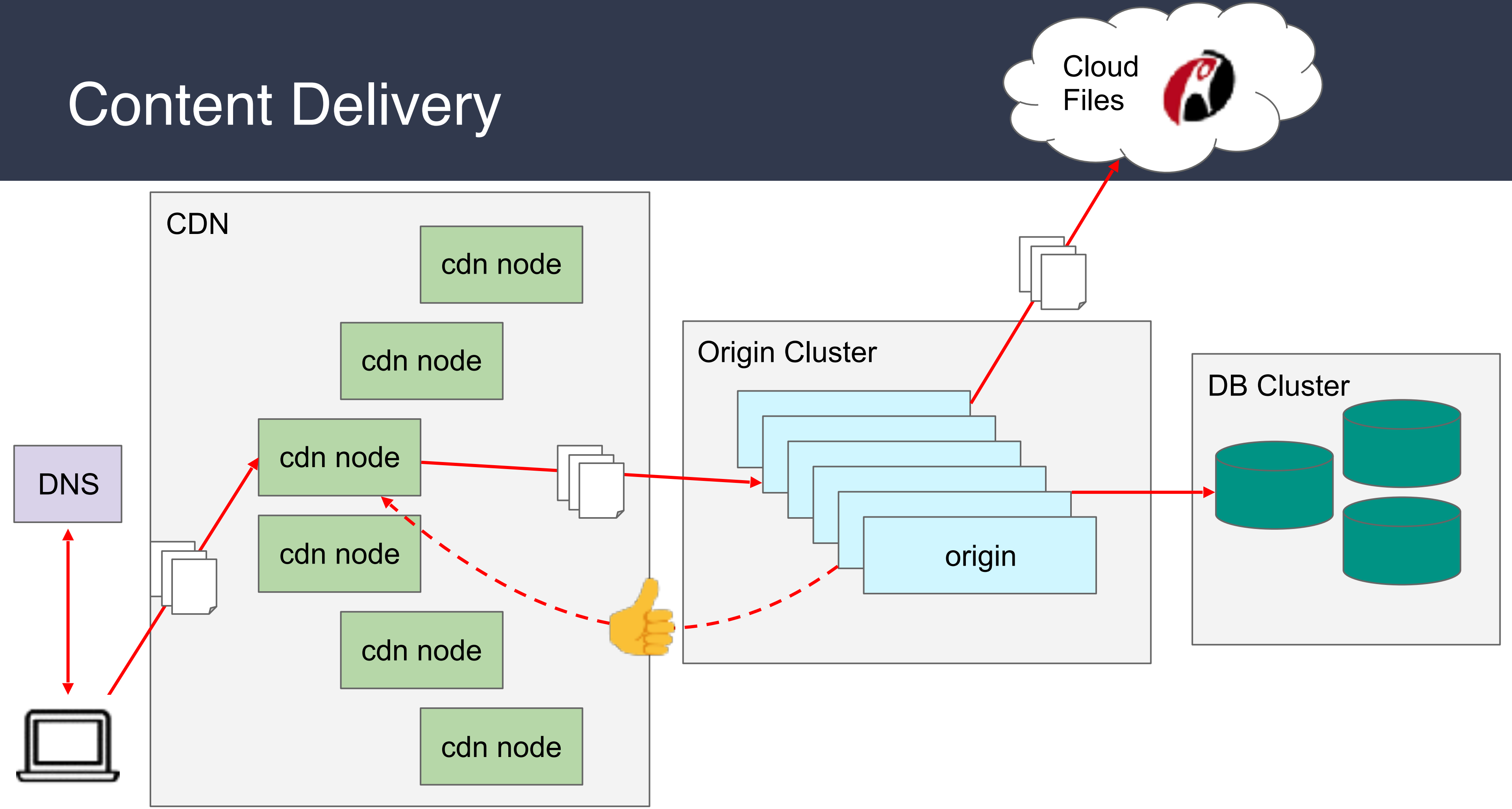
# Content Delivery



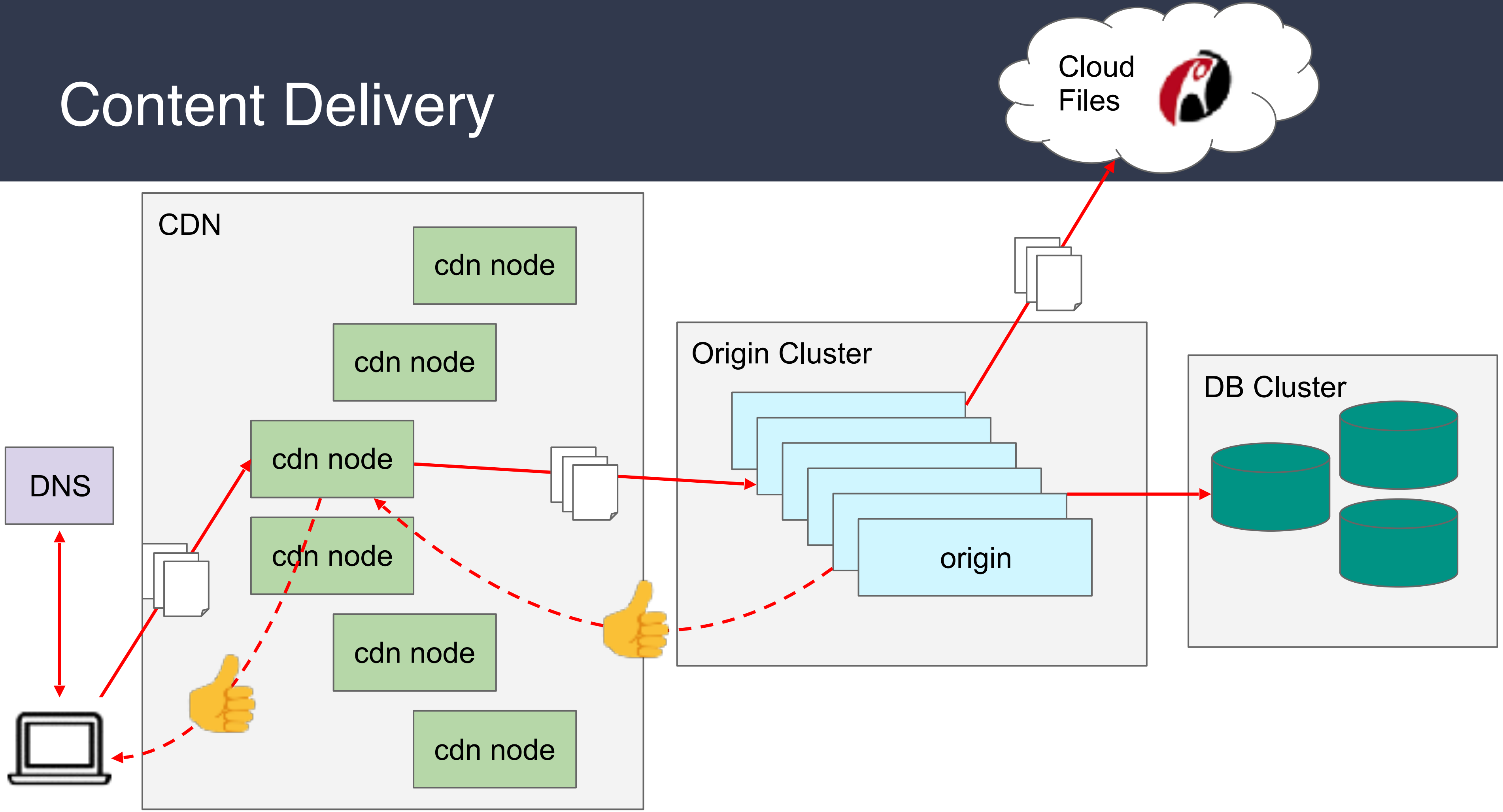
# Content Delivery



# Content Delivery



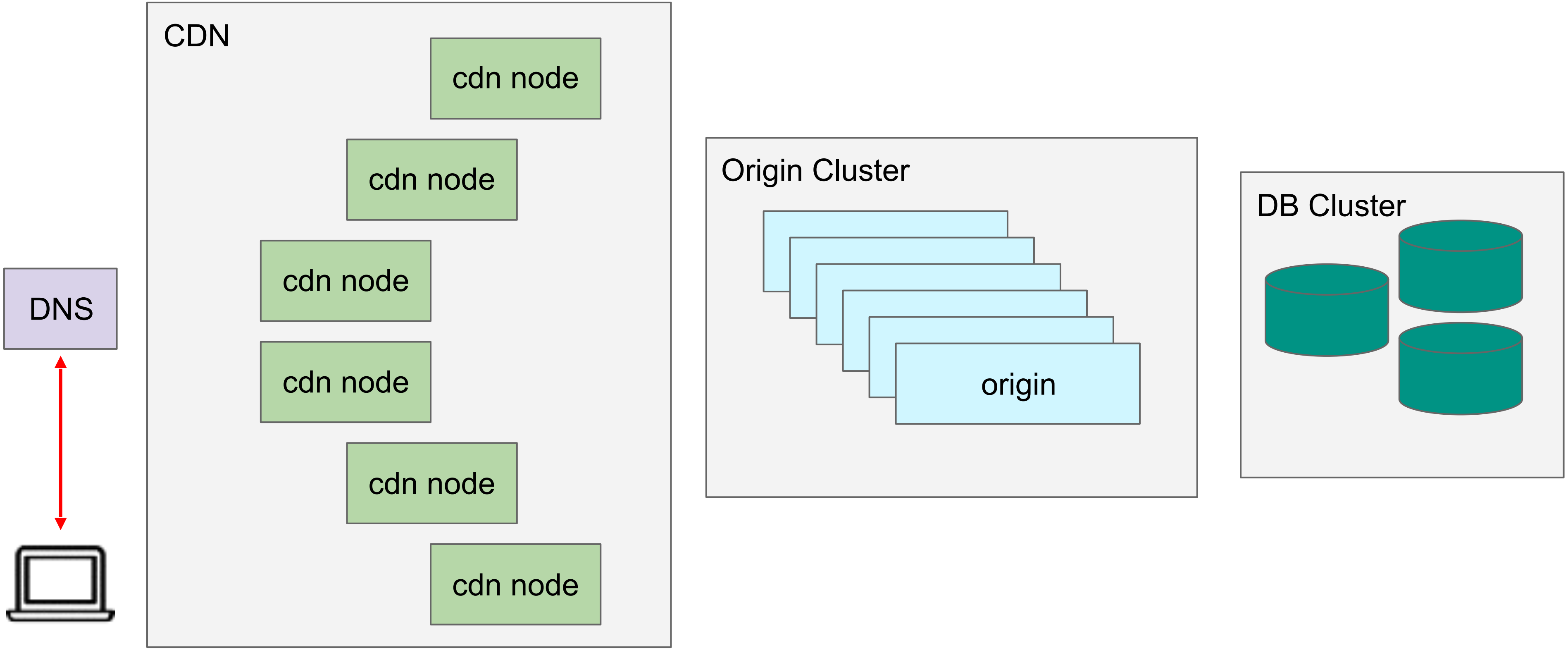
# Content Delivery



# Getting Data out of the system

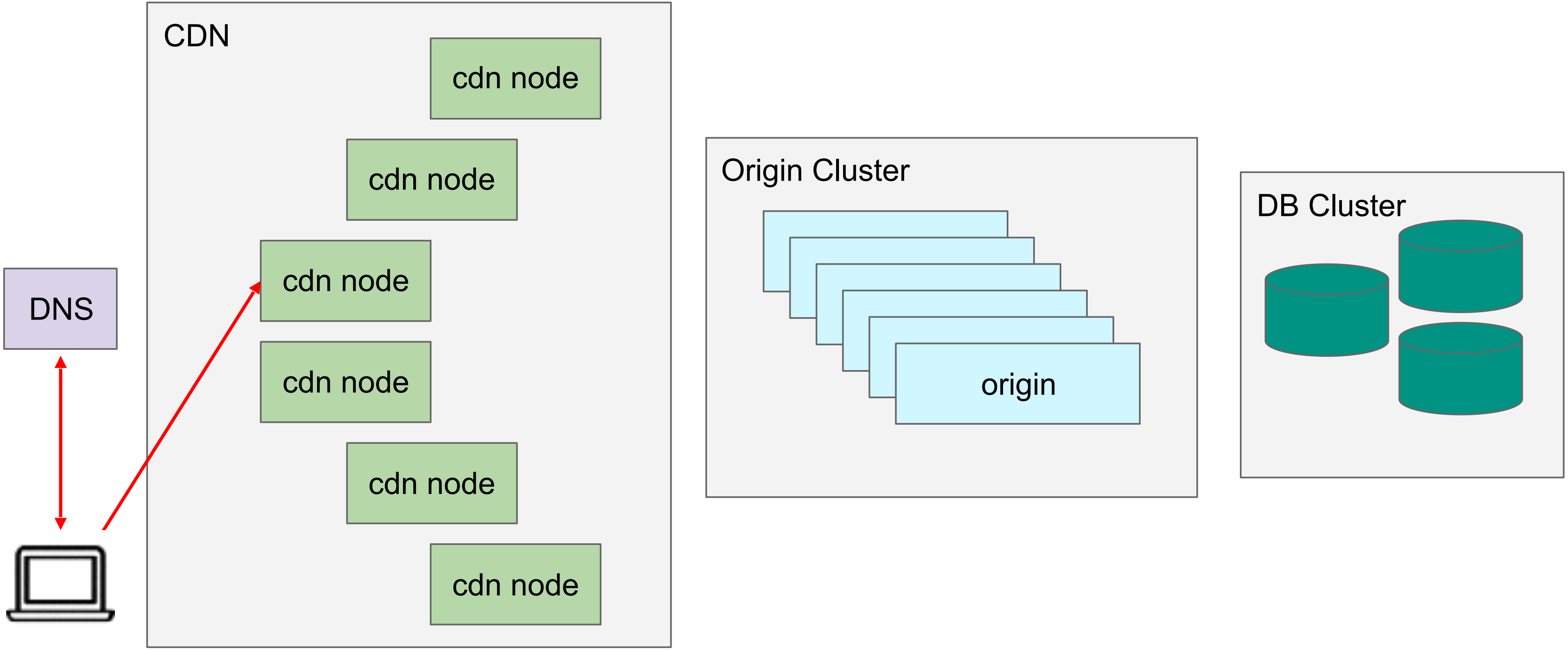
# Content Delivery

Cloud Files



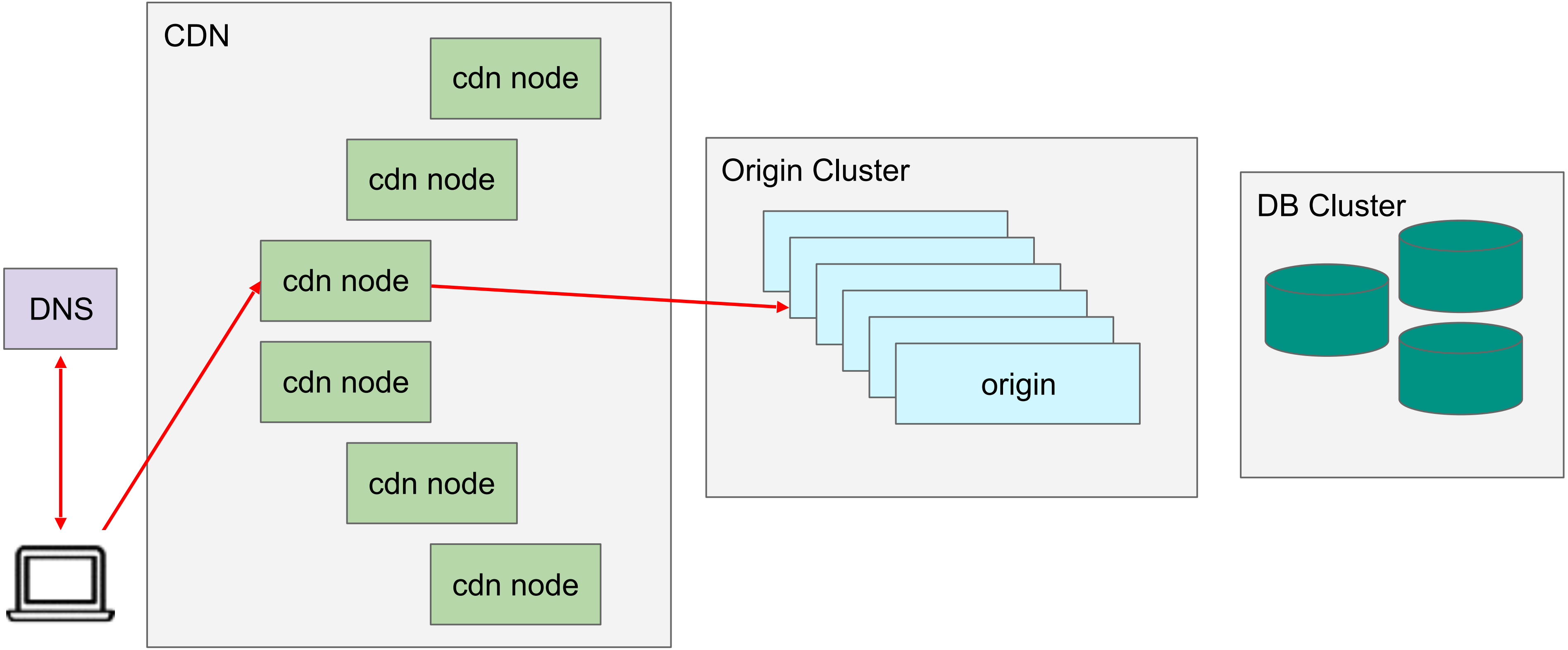
# Content Delivery

Cloud Files



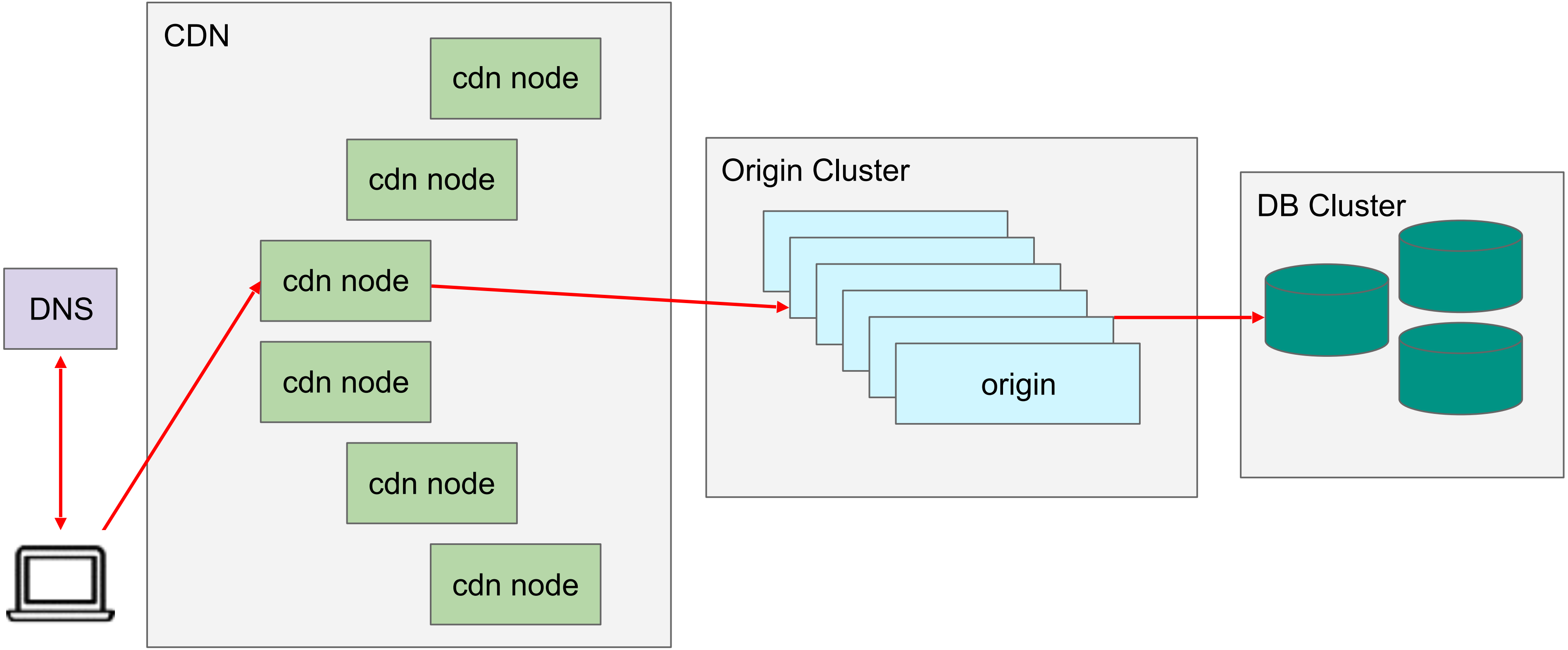
# Content Delivery

Cloud Files



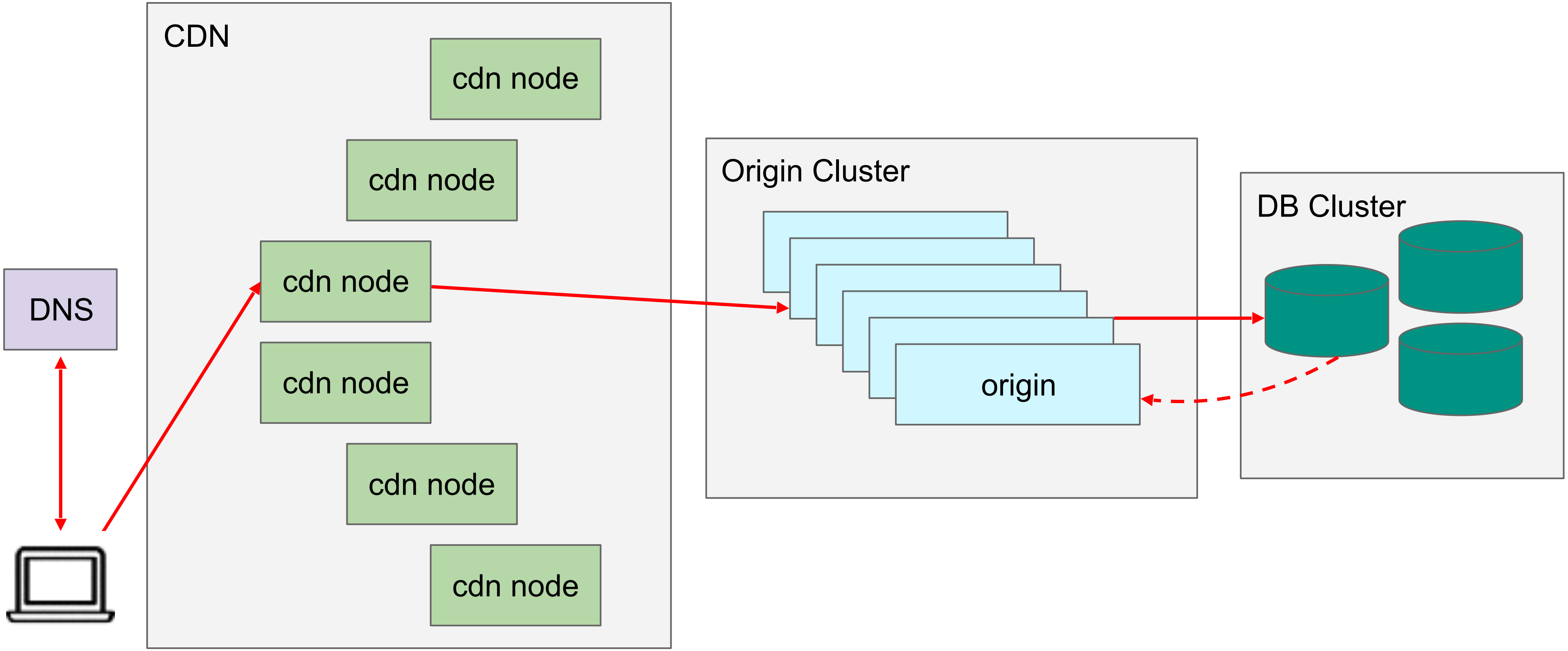
# Content Delivery

Cloud Files

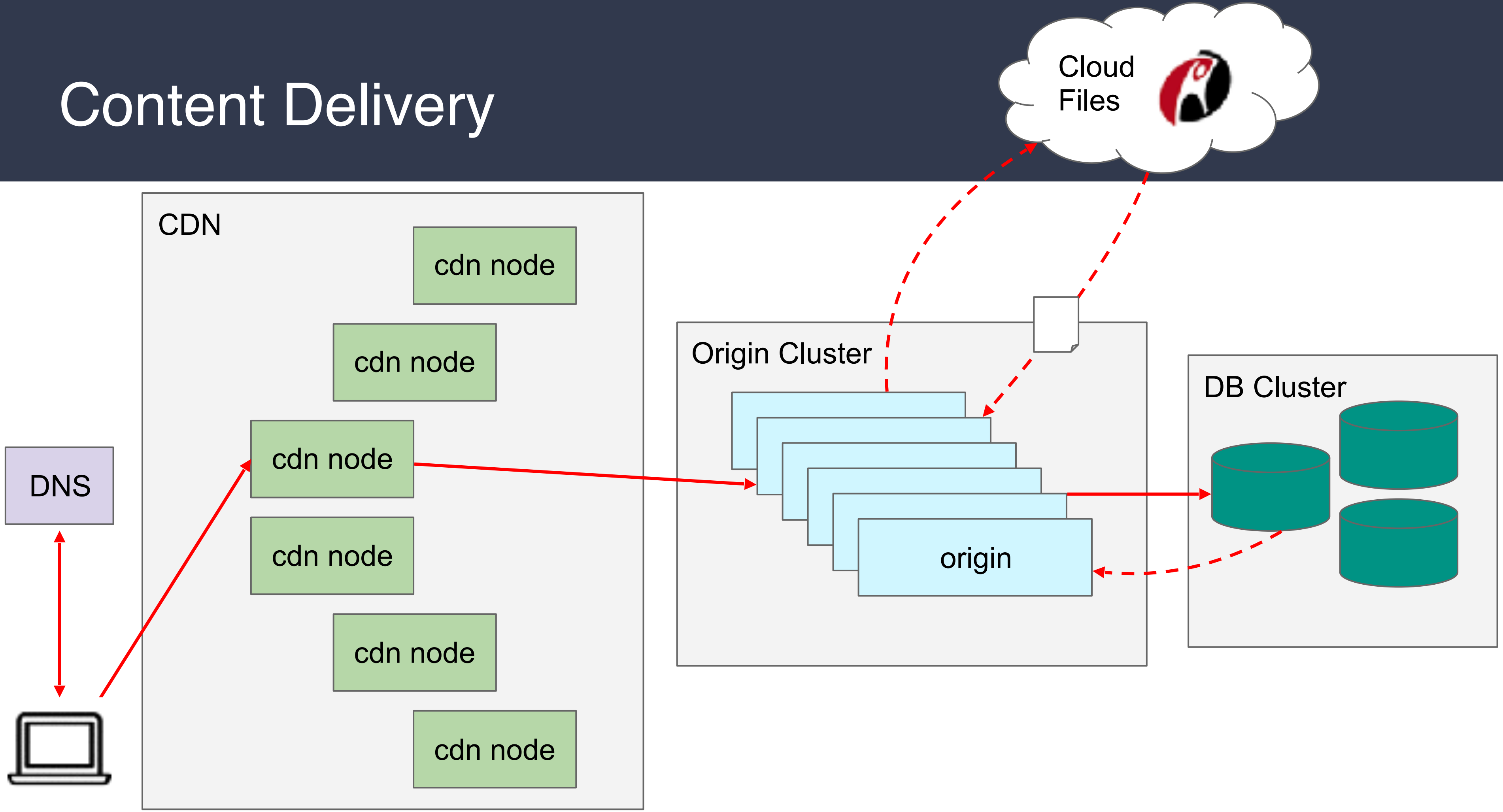


# Content Delivery

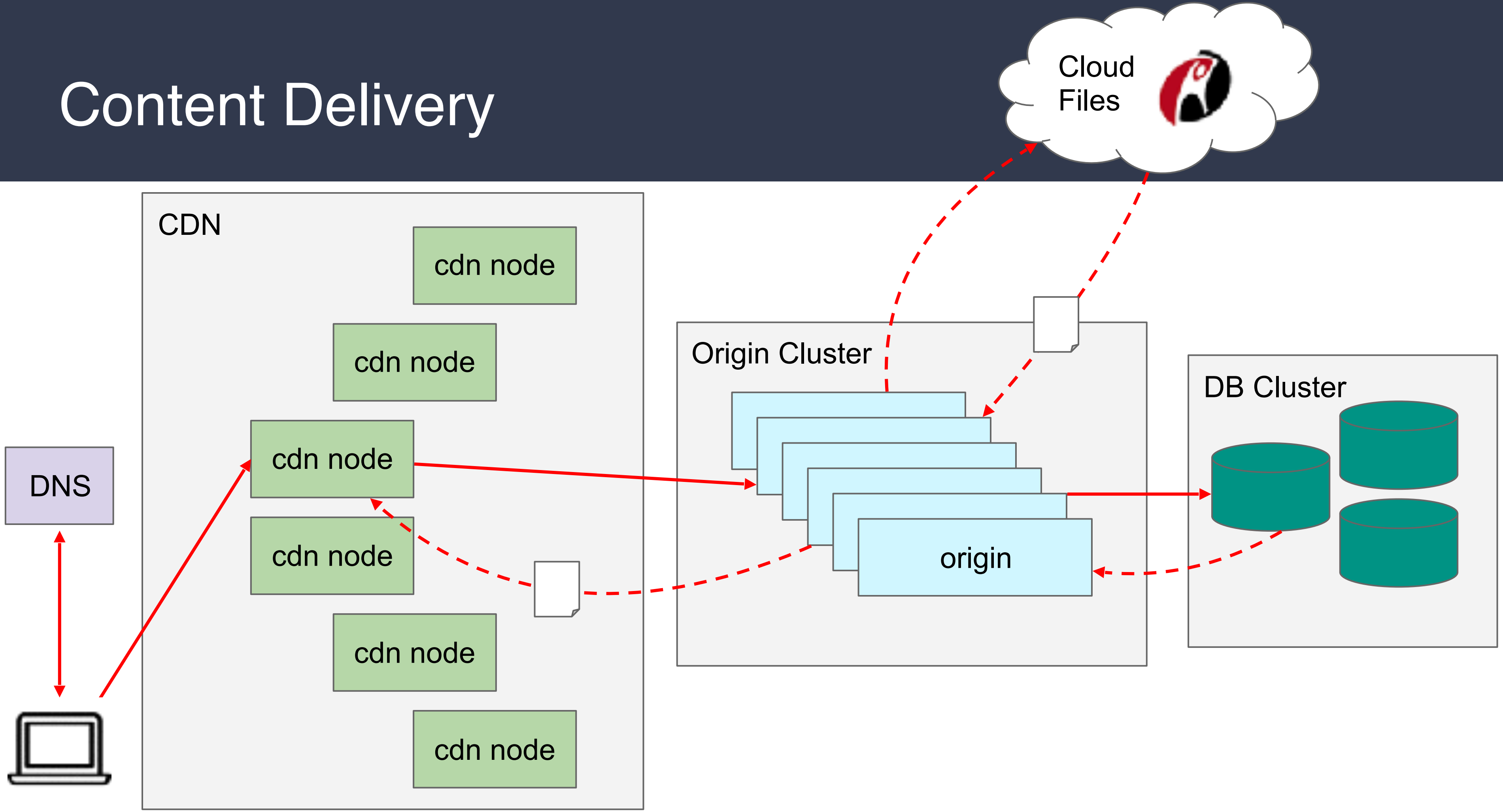
Cloud Files



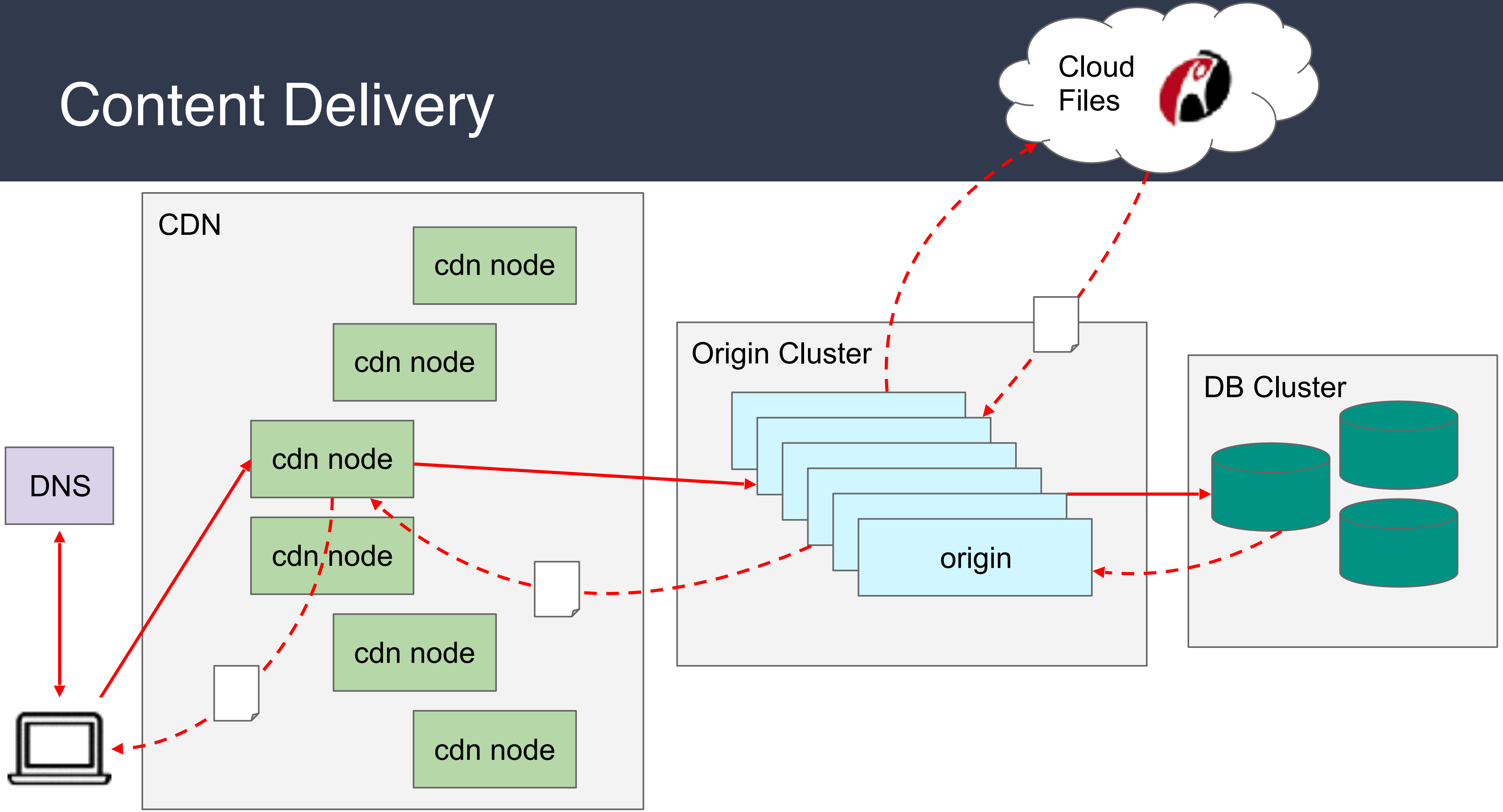
# Content Delivery



# Content Delivery

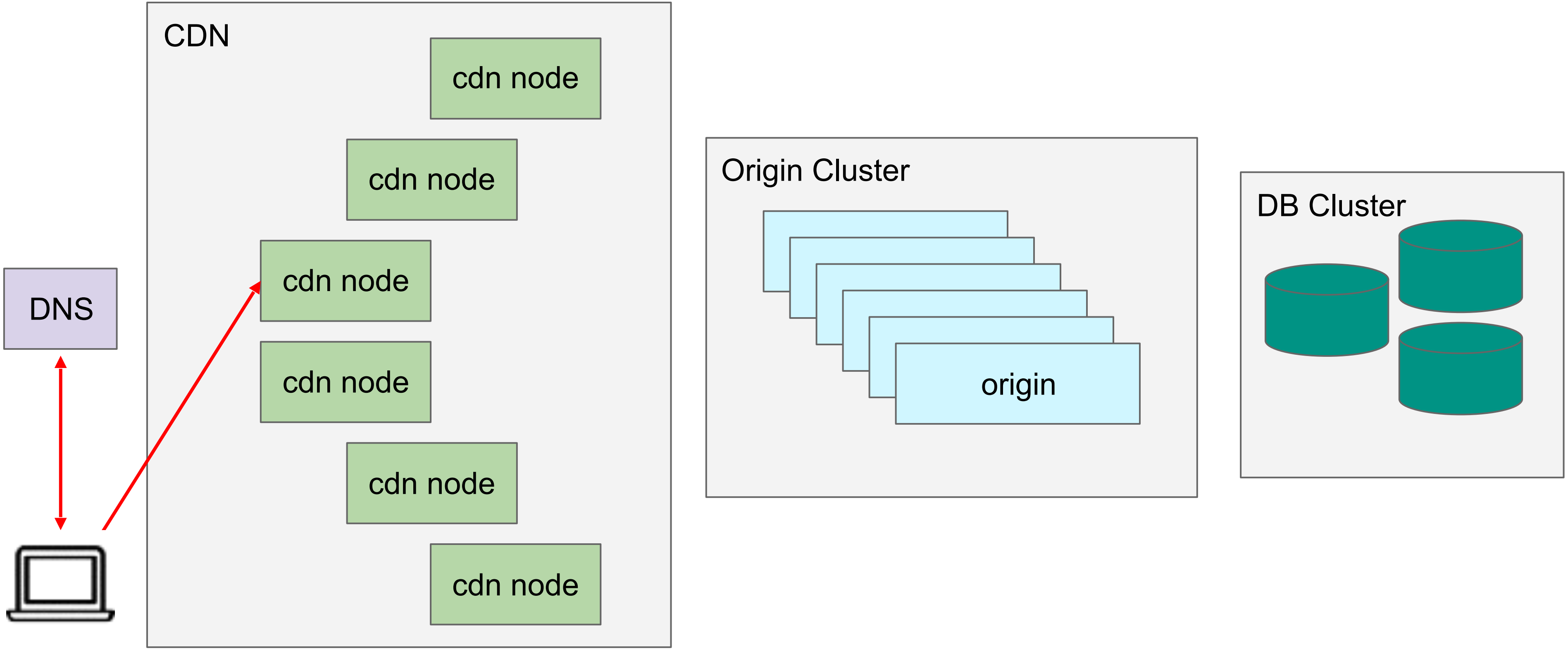


# Content Delivery

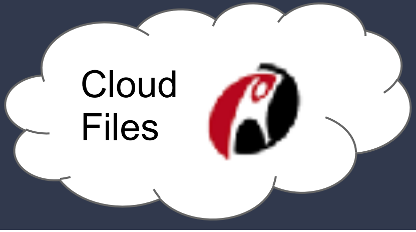


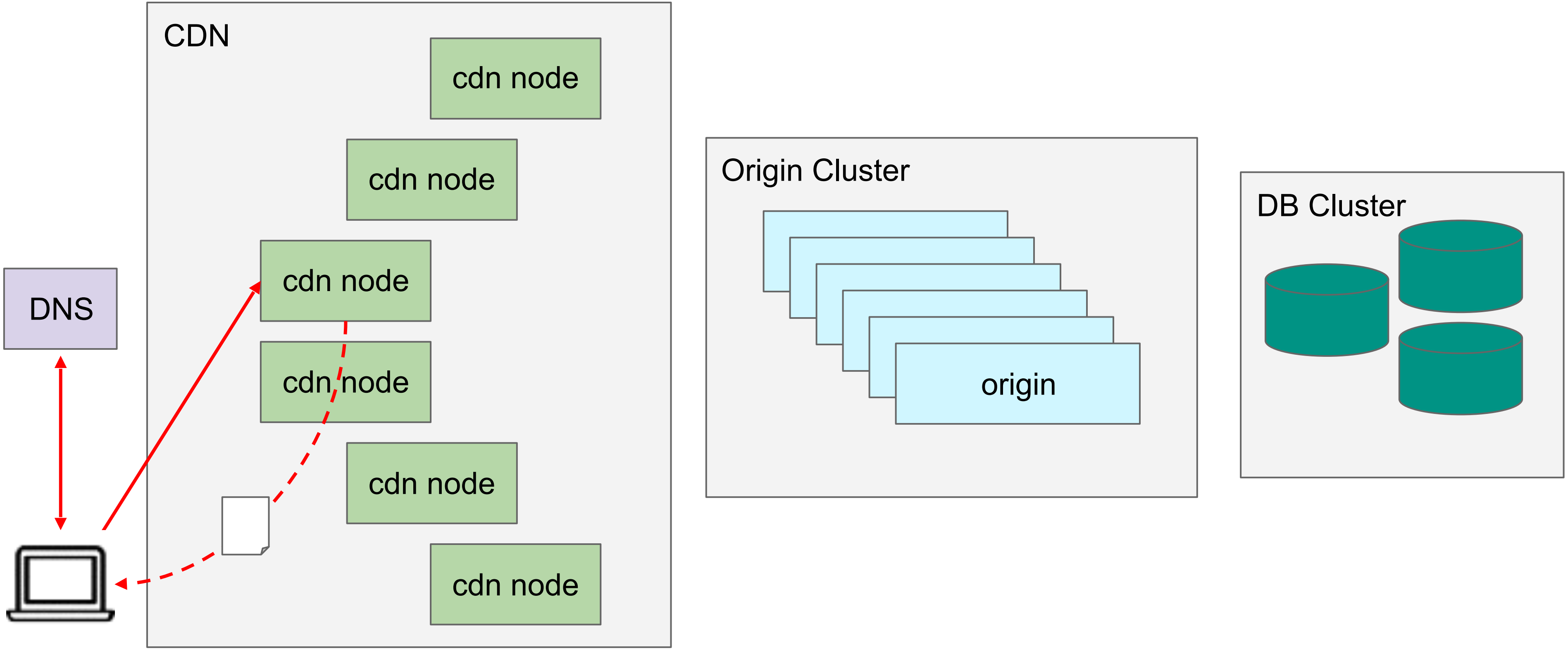
# Content Delivery

Cloud Files



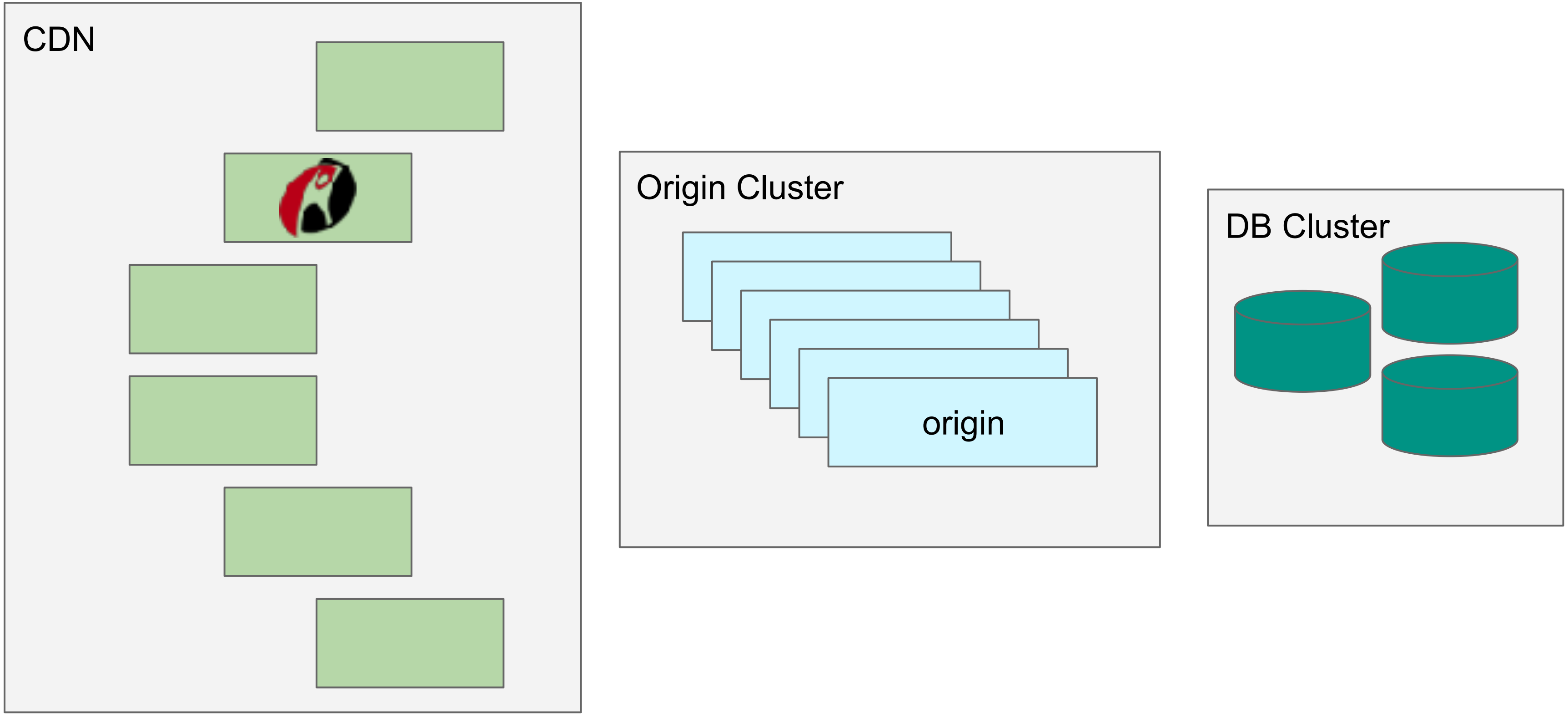
# Content Delivery



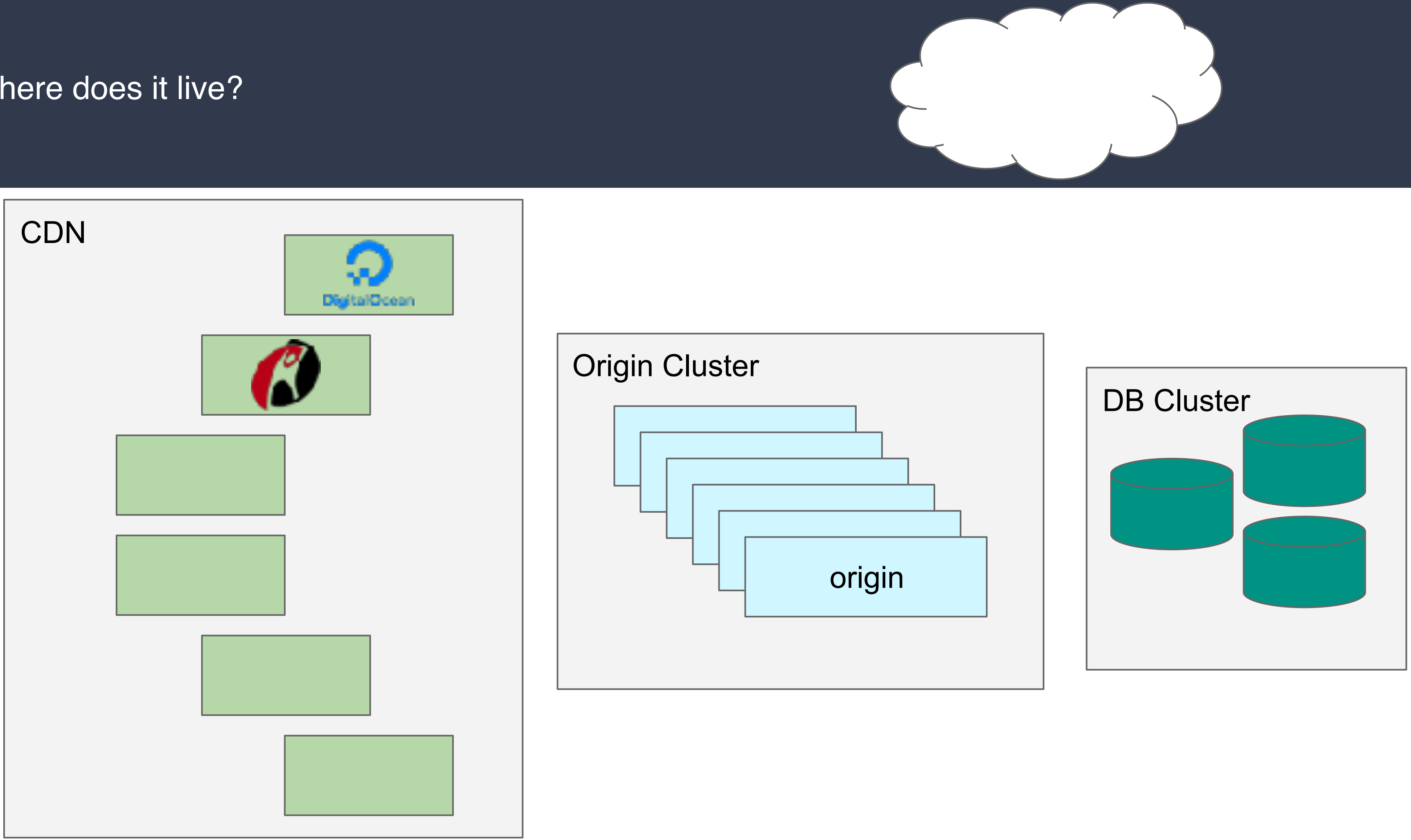


# Cool, but *where* are the actual servers?

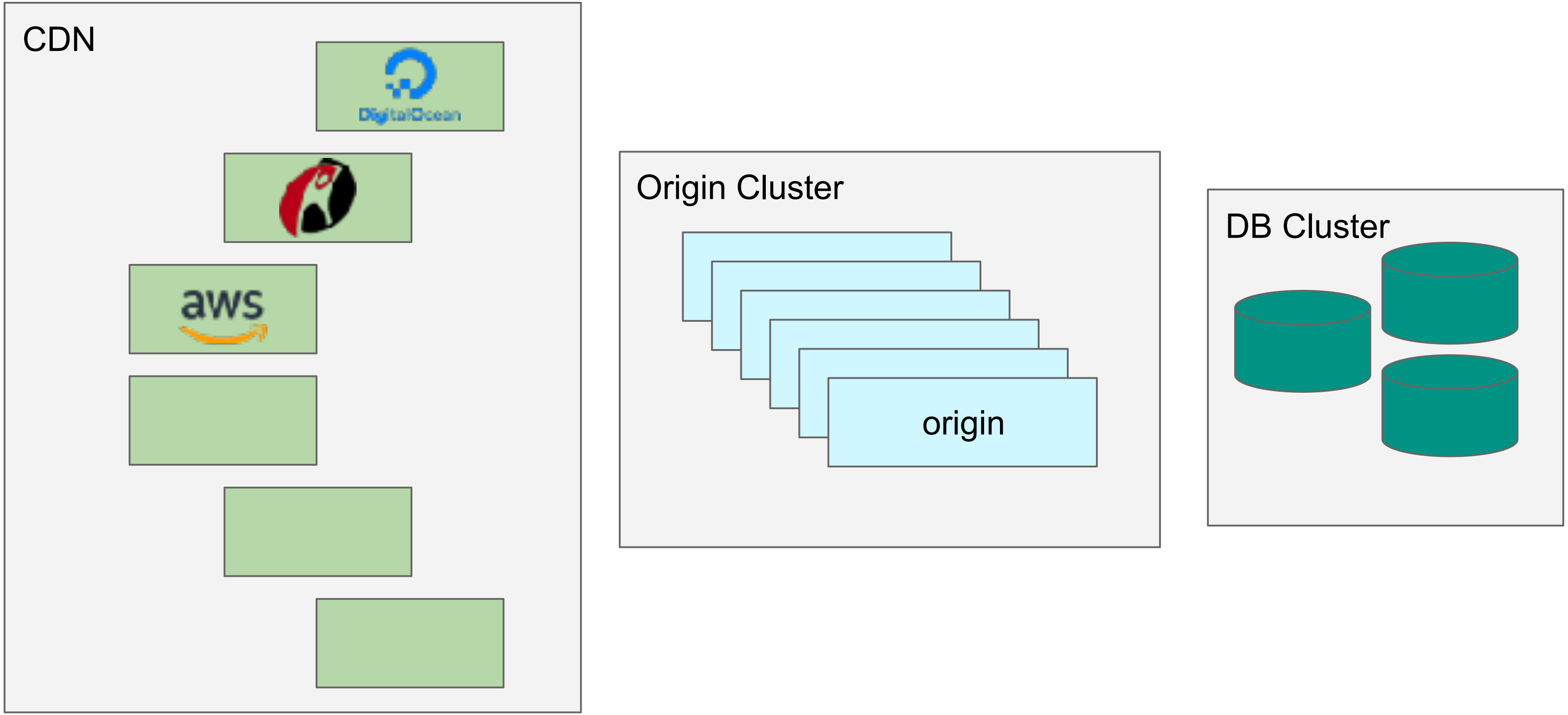
# But where does it live?



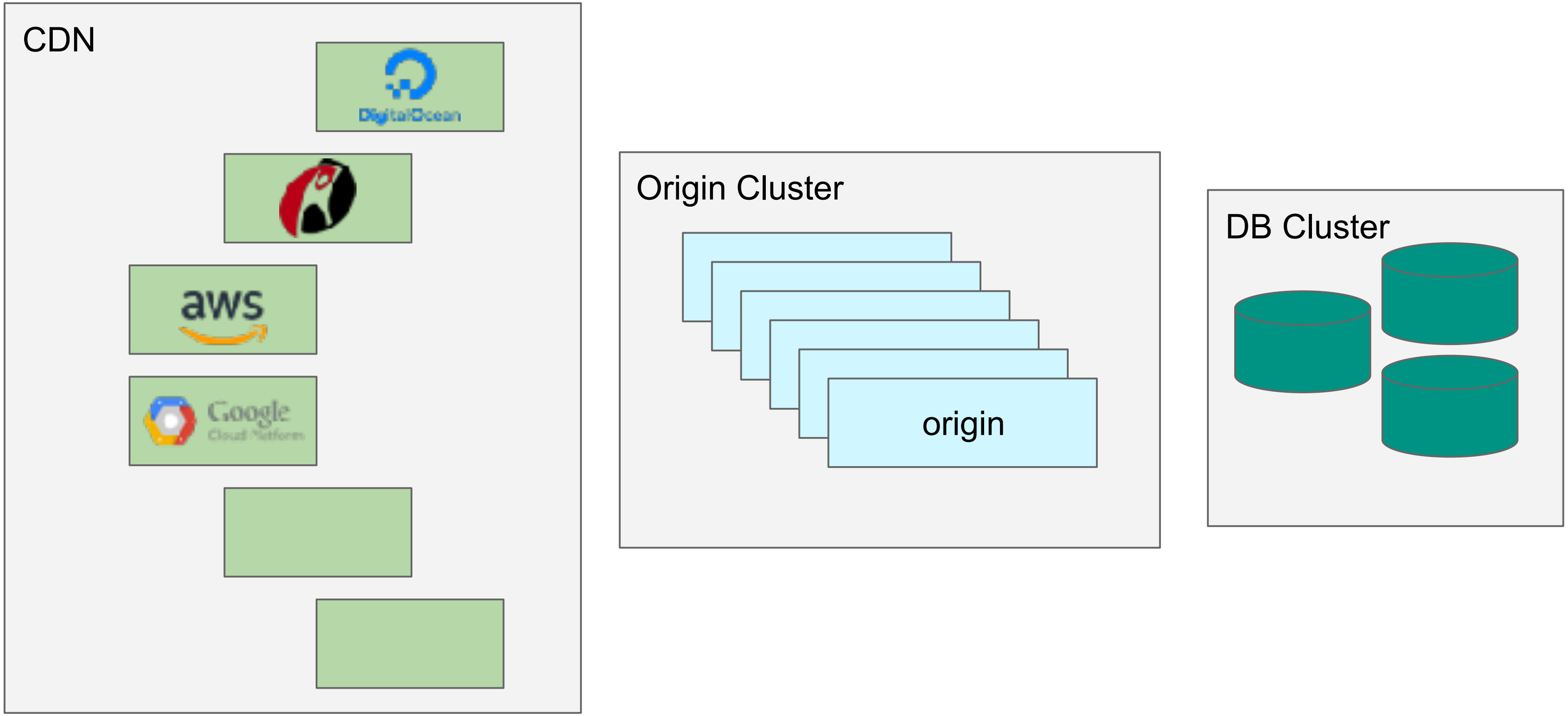
# But where does it live?



# But where does it live?



# But where does it live?



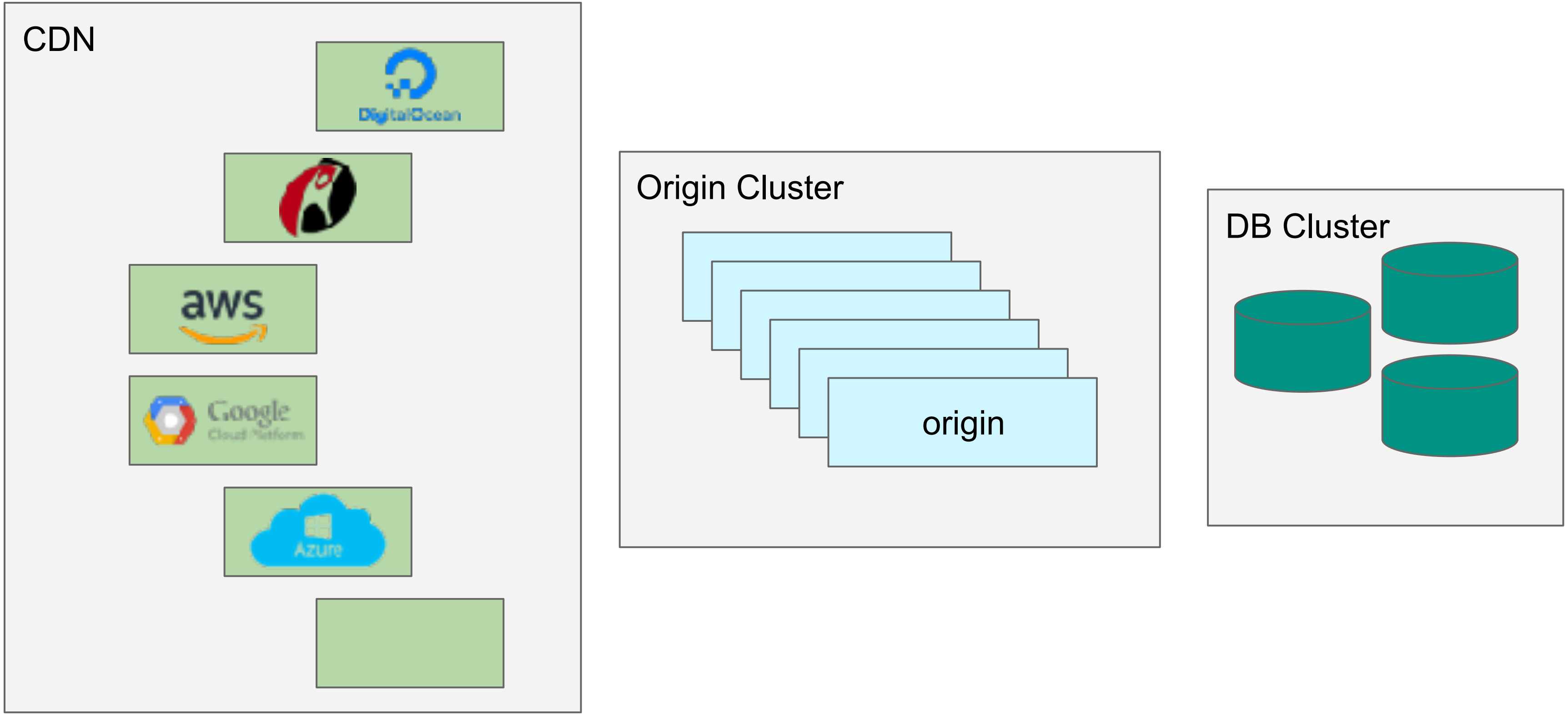
CDN

Origin Cluster

origin

DB Cluster

# But where does it live?



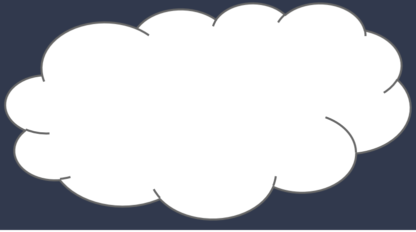
CDN

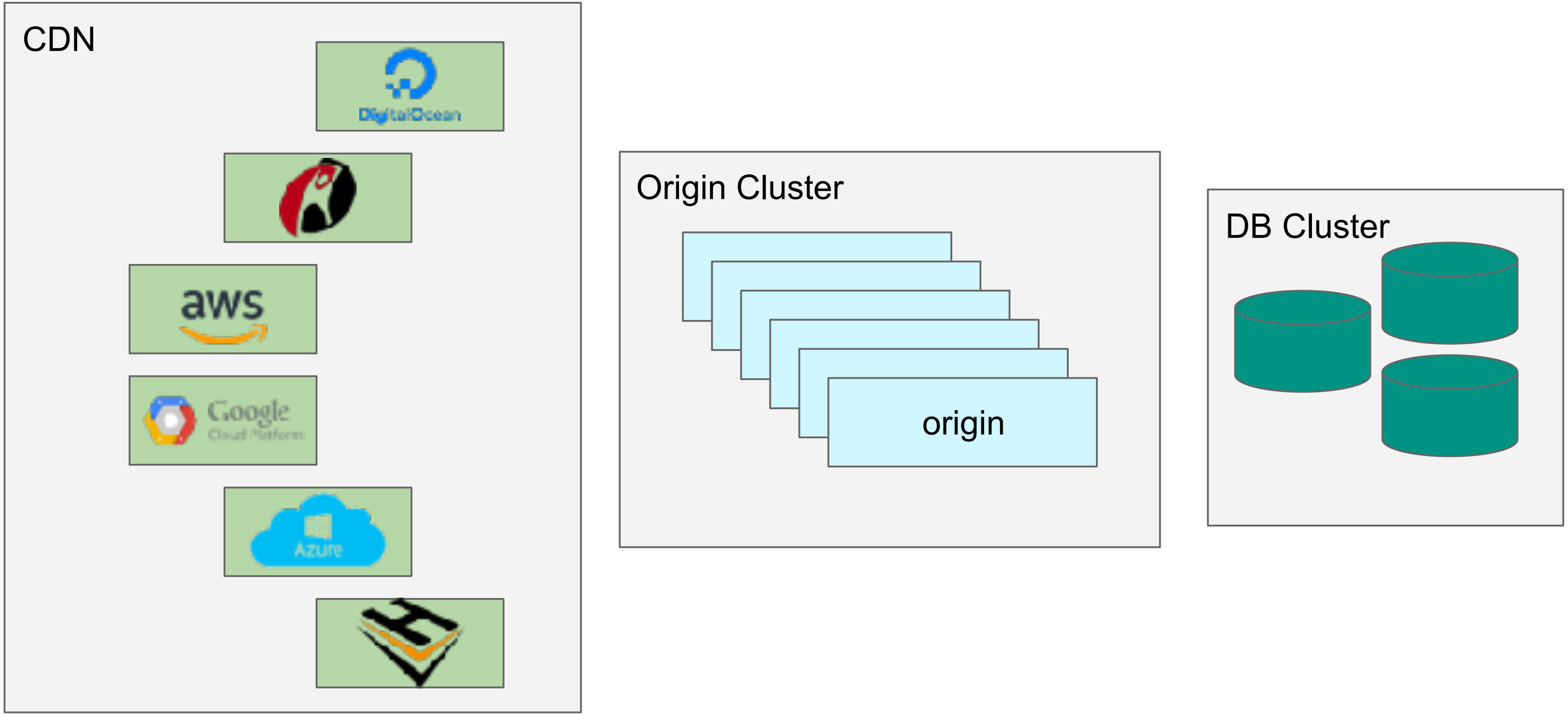
Origin Cluster

origin

DB Cluster

# But where does it live?





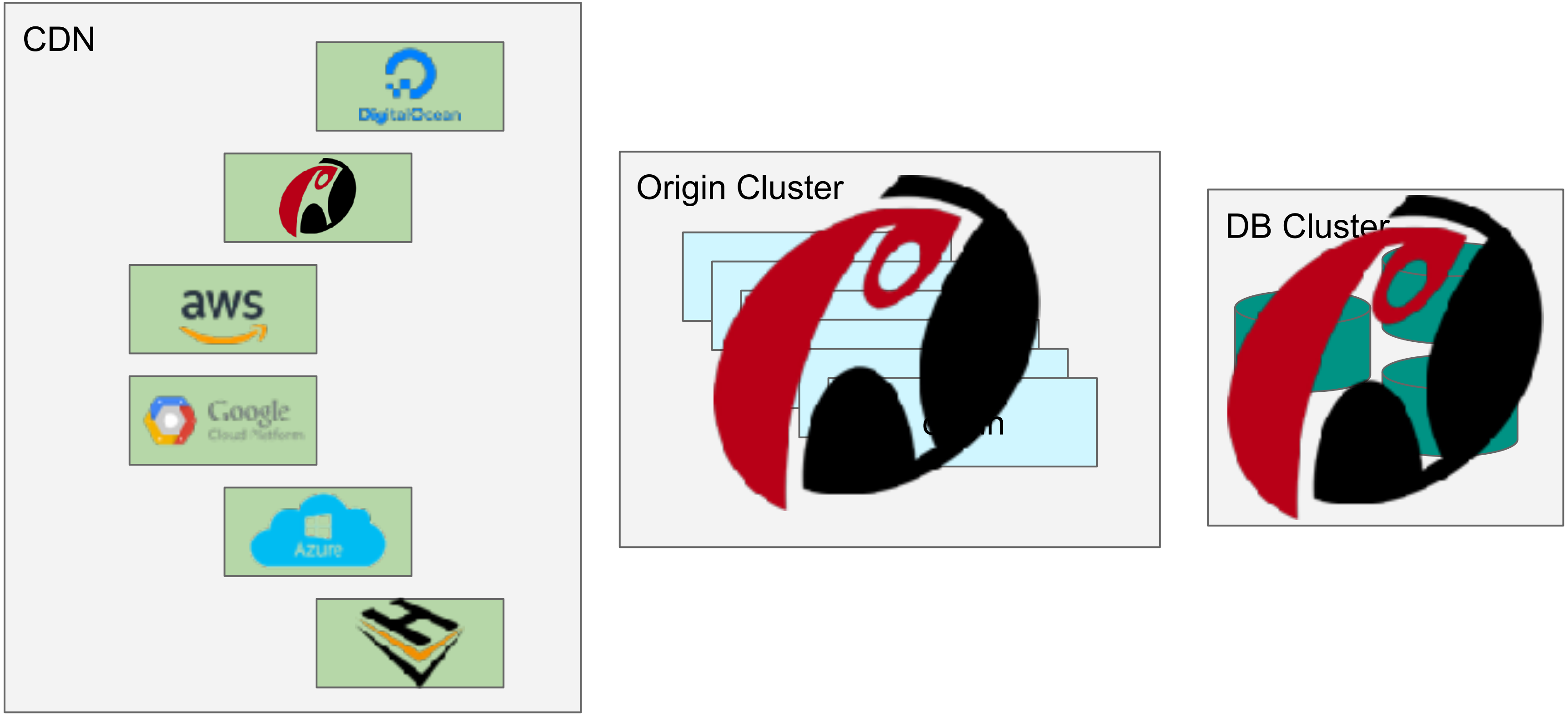
CDN

Origin Cluster

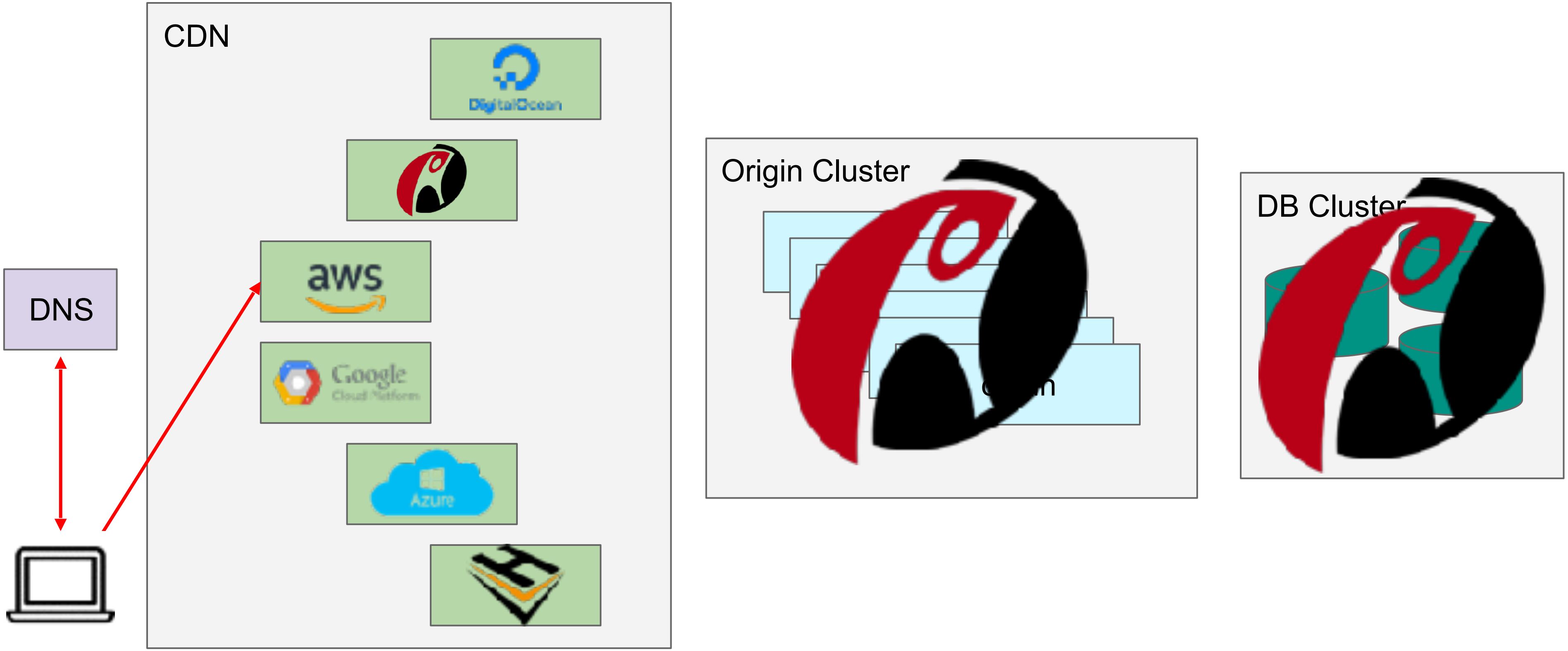
origin

DB Cluster

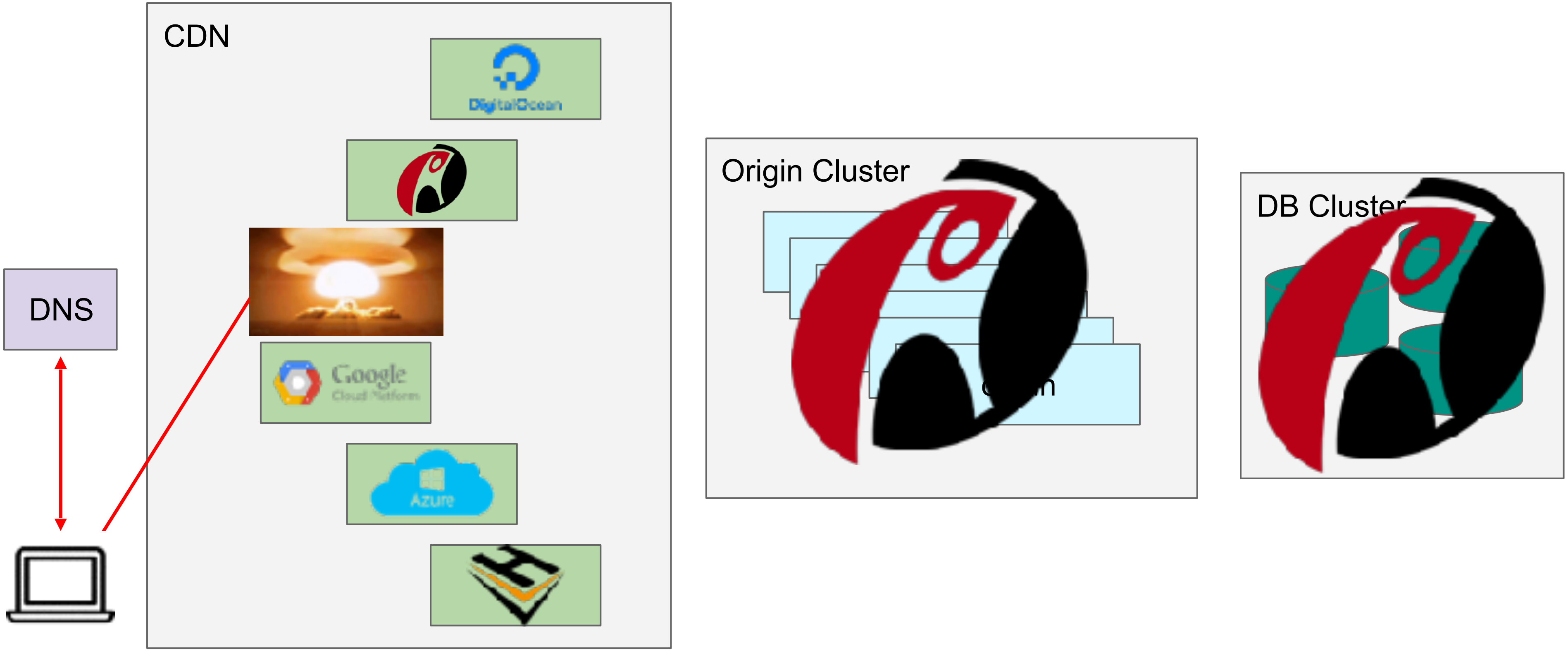
# But where does it live?



# And when it fails?



# And when it fails?



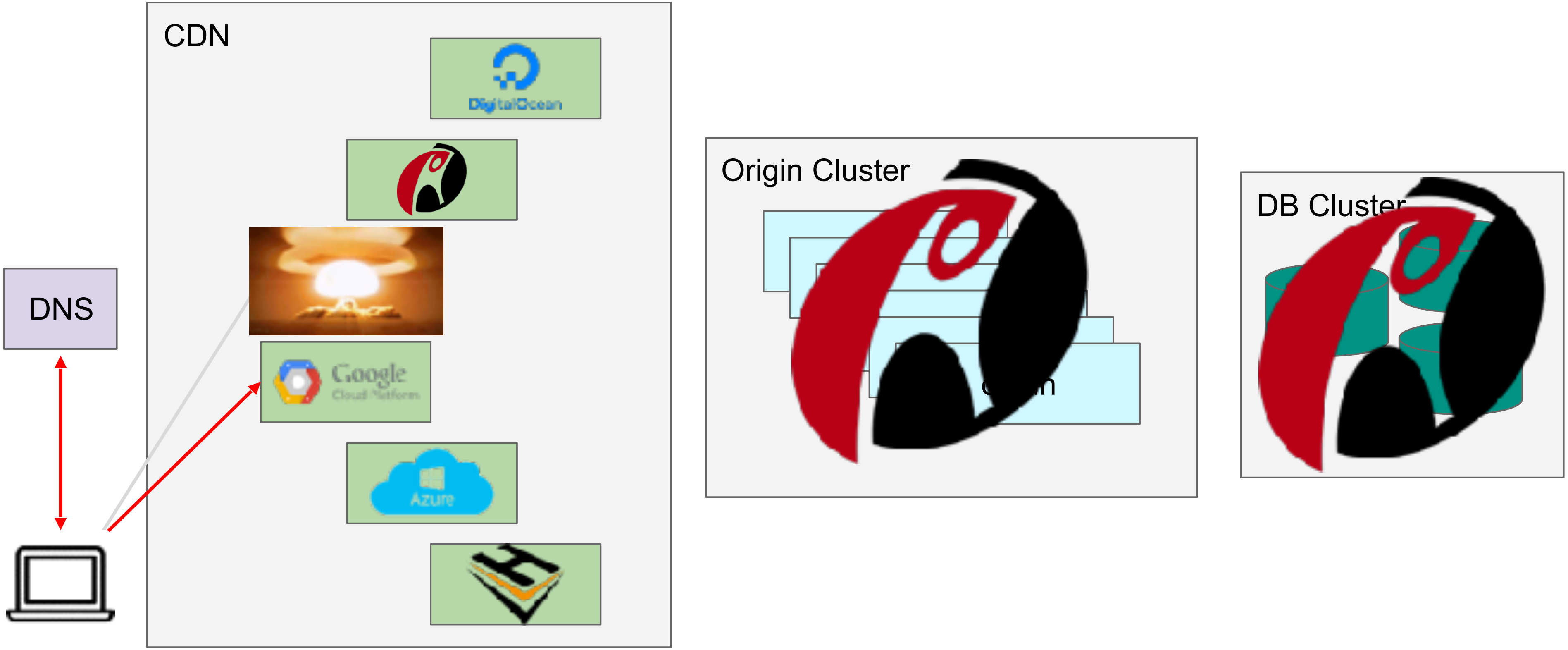
CDN

DNS

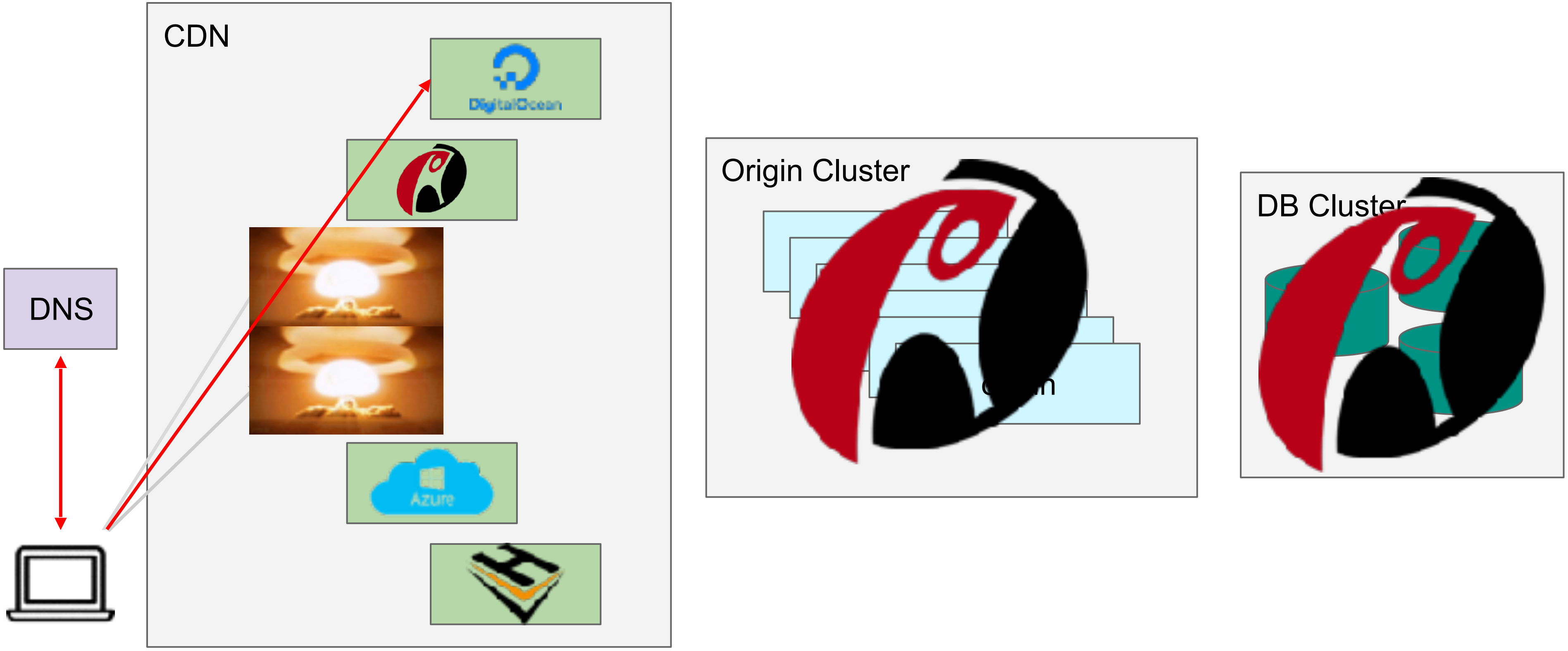
Origin Cluster

DB Cluster

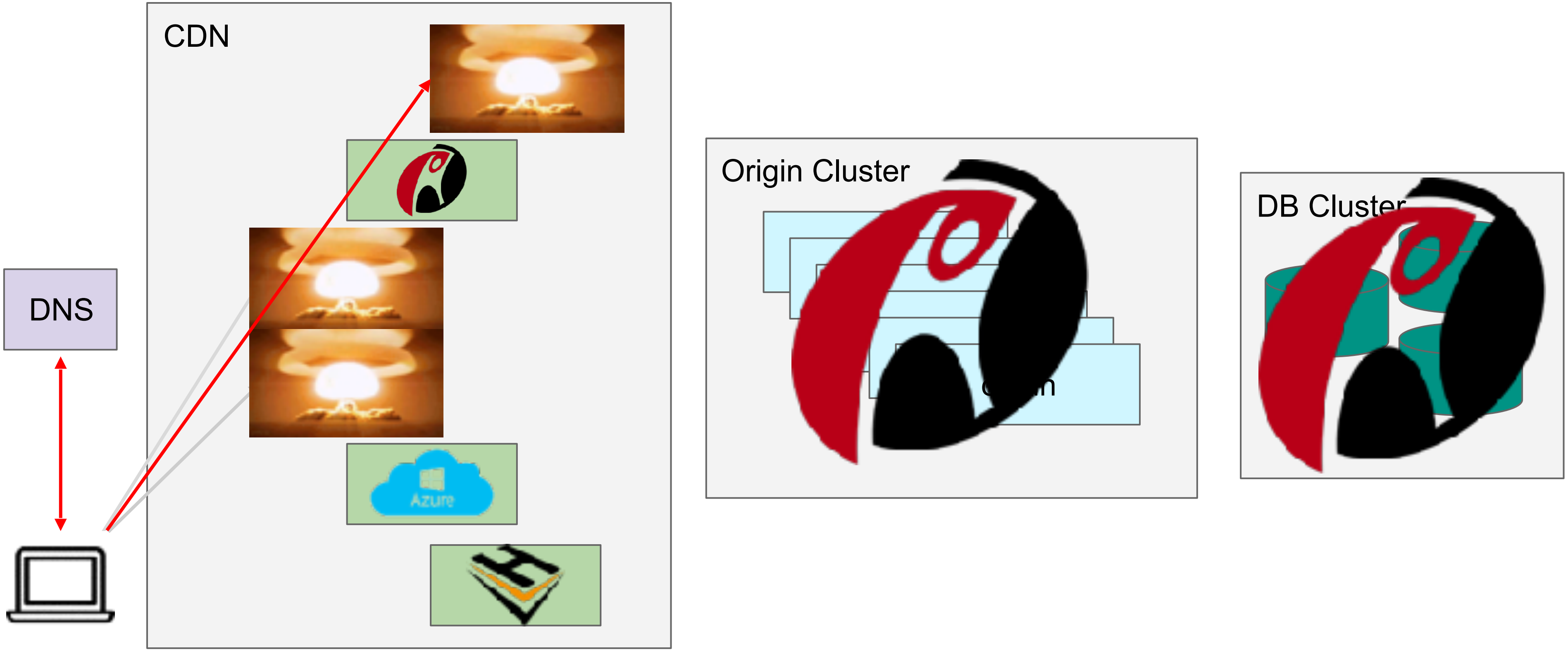
# And when it fails?



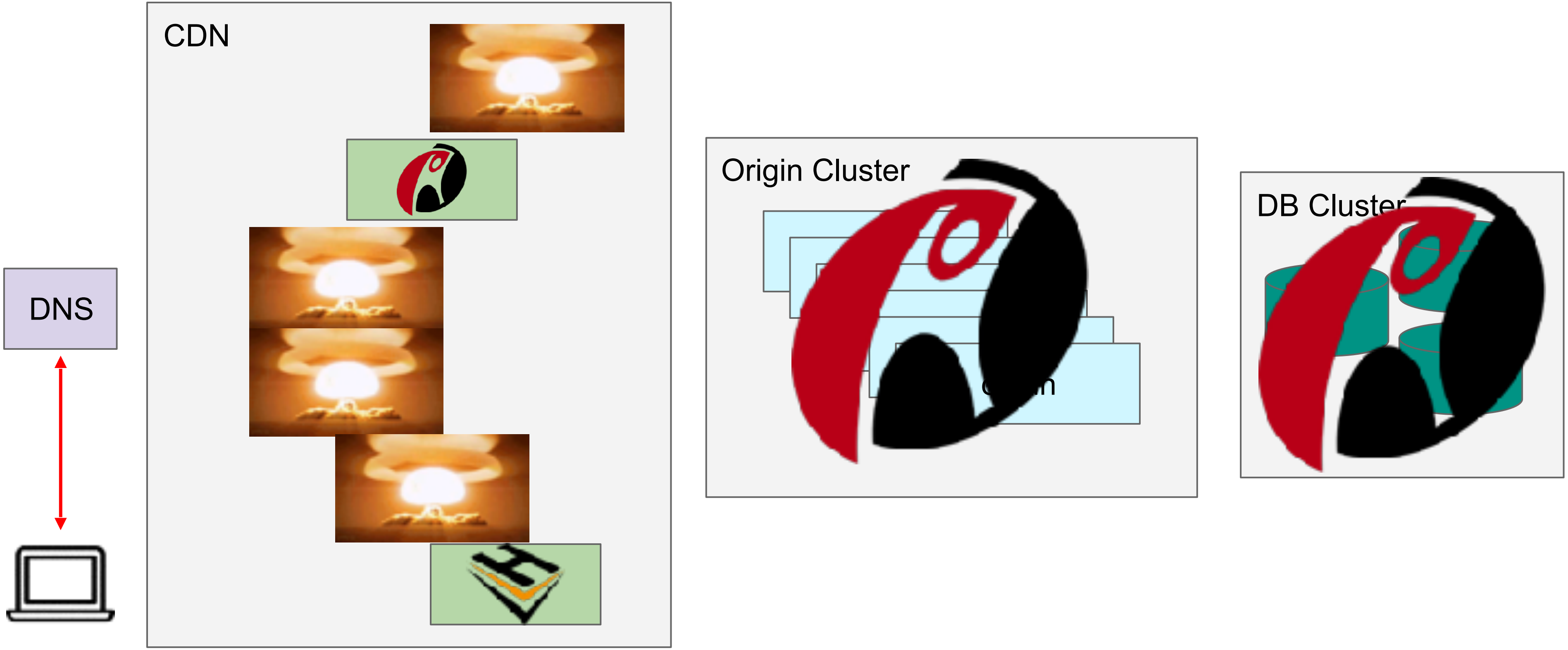
# And when it fails?



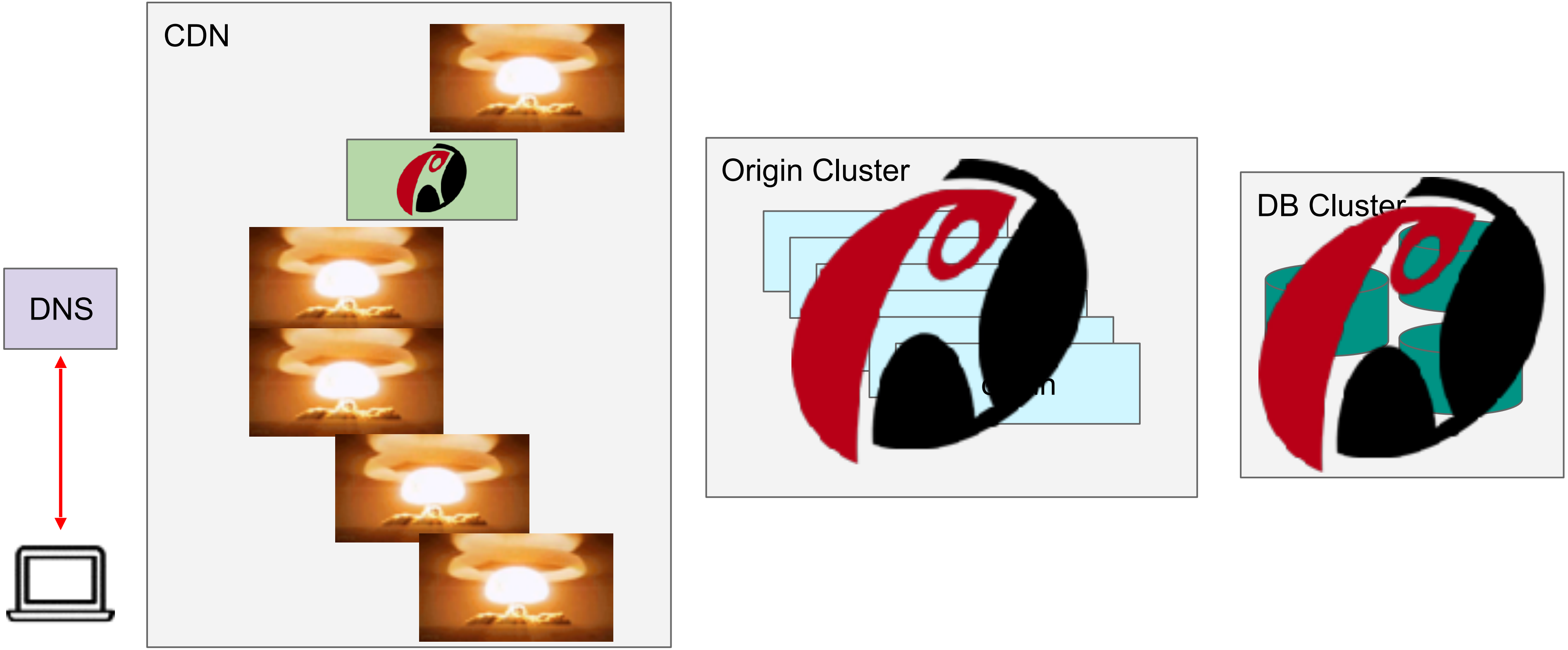
# And when it fails?



# And when it fails?



# And when it fails?



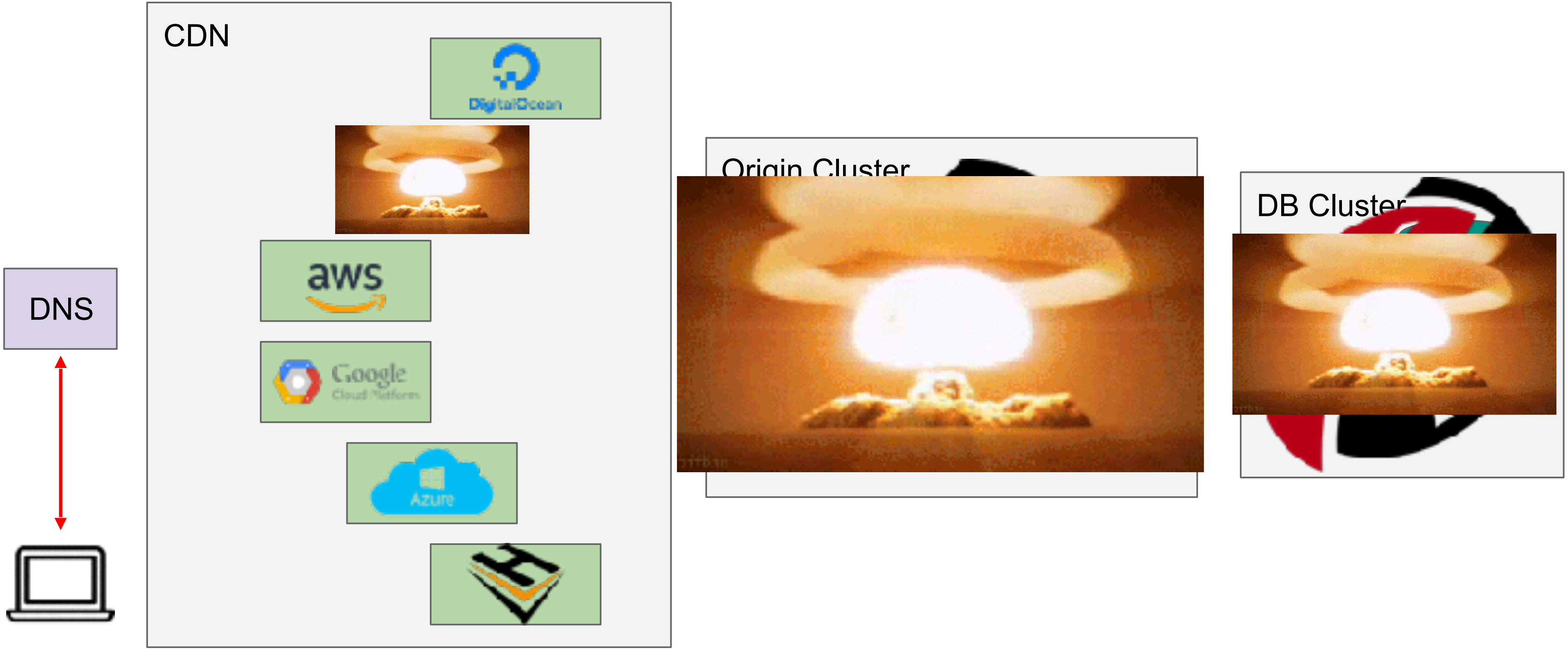
CDN

DNS

Origin Cluster

DB Cluster

# And when it fails?



# What happens when things go wrong?

# What happens when things go wrong?

- CDN stays up
- Keep serving cached content
- Higher traffic sites are going to be happier

# What happens when things go wrong?

- CDN stays up
- Keep serving cached content
- Higher traffic sites are going to be happier

# What happens when things go wrong?

- CDN stays up
- Keep serving cached content
- Higher traffic sites are going to be happier

# Multicloud Setup

- Providers fail with no notice
- Degraded perf > outage
- Same cloud is fastest

# Multicloud Setup

- Providers fail with no notice
- Degraded perf > outage
- Same cloud is fastest

- RAX
  - Cloud Servers
  - Cloud Files

- AWS
  - Elastic Compute Cloud
  - S3

- GCP
  - Compute Engine
  - Cloud Storage

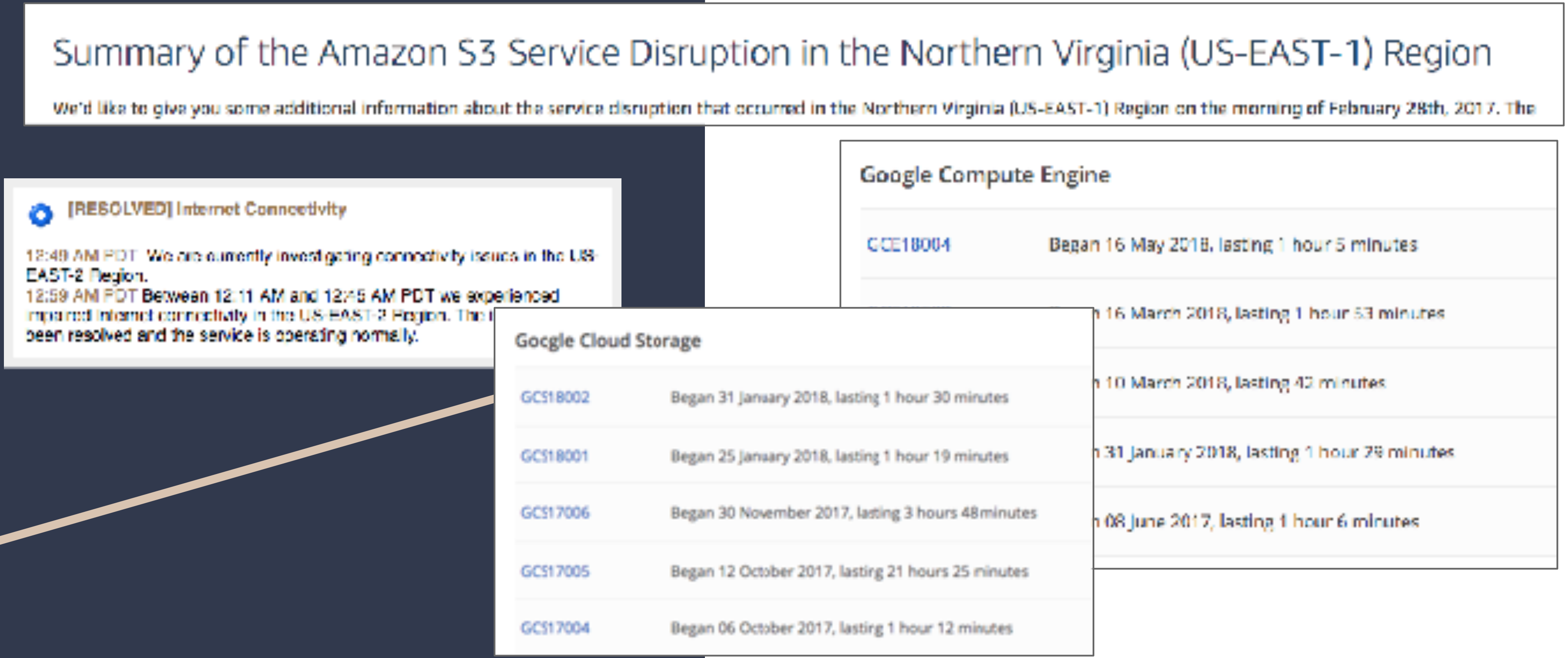
# Multicloud Setup

- Providers fail with no notice
- Degraded perf > outage
- Same cloud is fastest

- ~~RA~~
  - ~~ud Servers~~
  - ~~Cloud Files~~
- AWS
  - Elastic Compute Cloud
  - S3
- GCP
  - Compute Engine
  - Cloud Storage

# Why do all of this?

## Because clouds fail



### Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region

We'd like to give you some additional information about the service disruption that occurred in the Northern Virginia (US-EAST-1) Region on the morning of February 28th, 2017. The

**[RESOLVED] Internet Connectivity**

12:49 AM PDT We are currently investigating connectivity issues in the US-EAST-2 Region.
12:59 AM PDT Between 12:11 AM and 12:45 AM PDT we experienced impaired internet connectivity in the US-EAST-2 Region. The i
been resolved and the service is operating normally.

**Google Compute Engine**

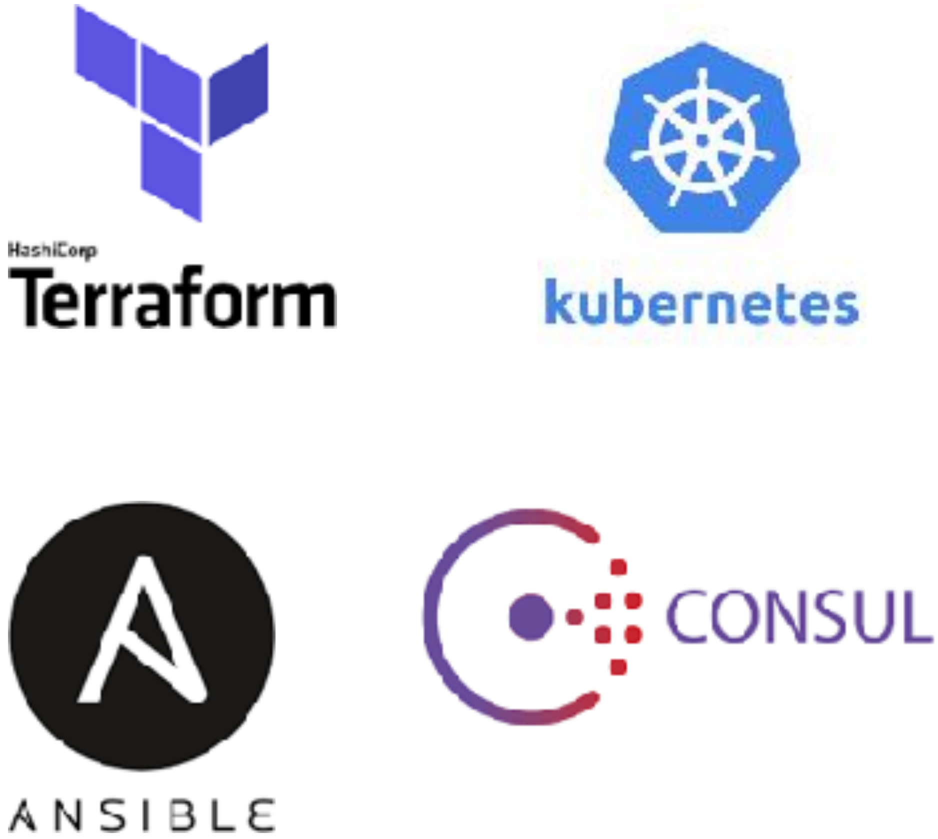
| | |
|---|---|
| GCE18004 | Began 16 May 2018, lasting 1 hour 5 minutes |
| | n 16 March 2018, lasting 1 hour 53 minutes |
| | n 10 March 2018, lasting 42 minutes |
| | n 31 January 2018, lasting 1 hour 28 minutes |
| | n 08 June 2017, lasting 1 hour 6 minutes |

**Google Cloud Storage**

| | |
|---|---|
| GCS18002 | Began 31 January 2018, lasting 1 hour 30 minutes |
| GCS18001 | Began 25 January 2018, lasting 1 hour 19 minutes |
| GCS17006 | Began 30 November 2017, lasting 3 hours 48minutes |
| GCS17005 | Began 12 October 2017, lasting 21 hours 25 minutes |
| GCS17004 | Began 06 October 2017, lasting 1 hour 12 minutes |

# But how do we build around that?

But how do we build around that?



# Steps to Multicloud

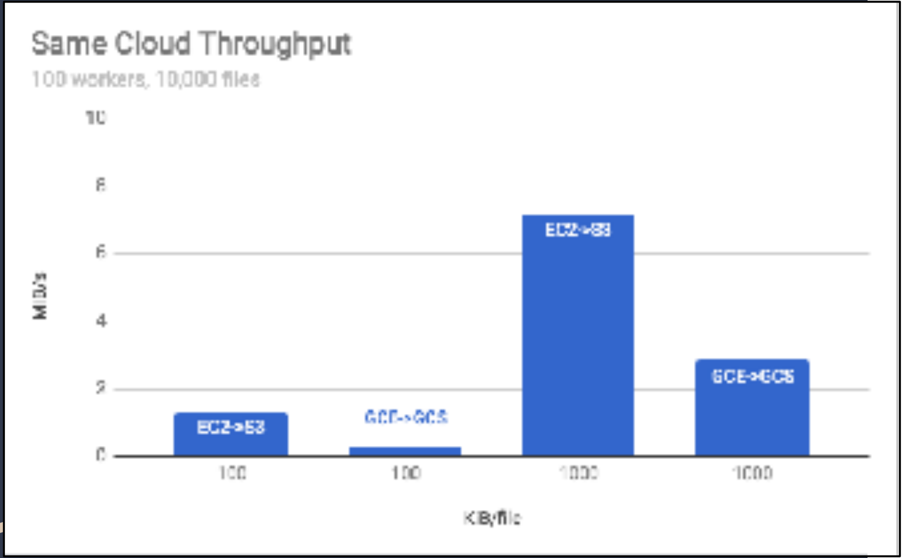
1. Double check assumptions
2. Replicate all the objects
3. Prepare the database
4. Make the origin services cloud agnostic
5. Test everything
6. Do the actual cutover

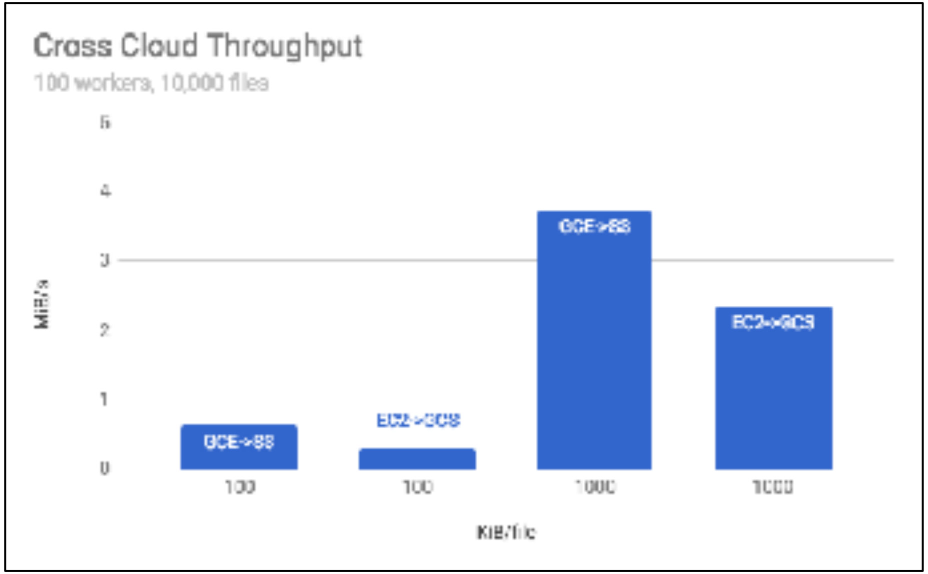
# Steps to Multicloud

1. Double check assumptions
2. Replicate all the objects
3. Prepare the database
4. Make the origin services cloud agnostic
5. Test everything
6. Do the actual cutover

# Assumption Checking

https://github.com/rybit/cloud-bench



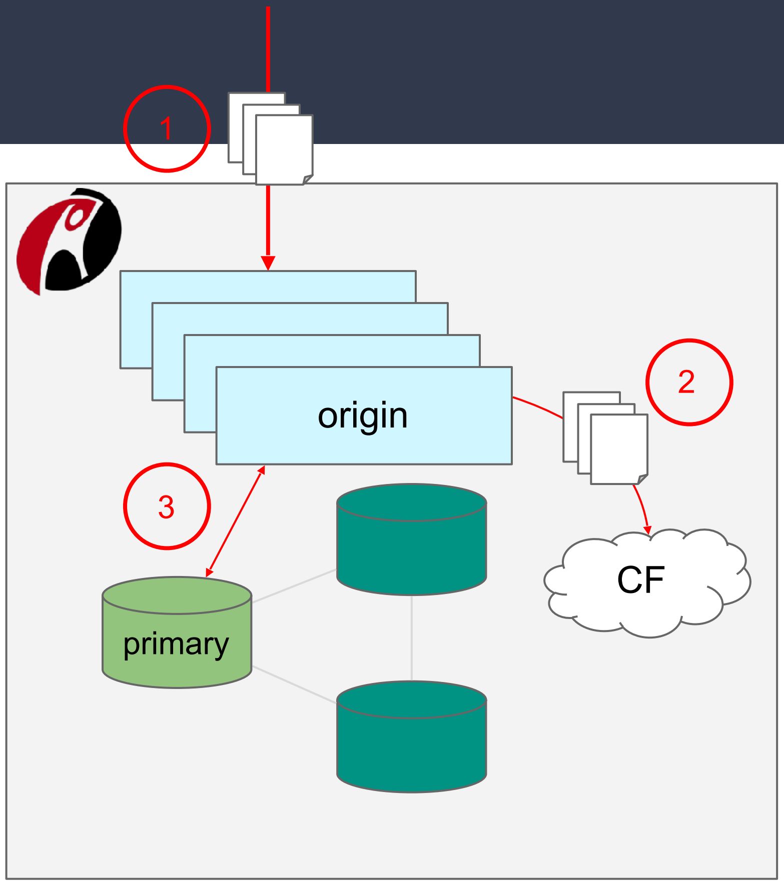


# Steps to Multicloud

1. Double check assumptions
2. Replicate all the objects
3. Prepare the database
4. Make the origin services cloud agnostic
5. Test everything
6. Do the actual cutover

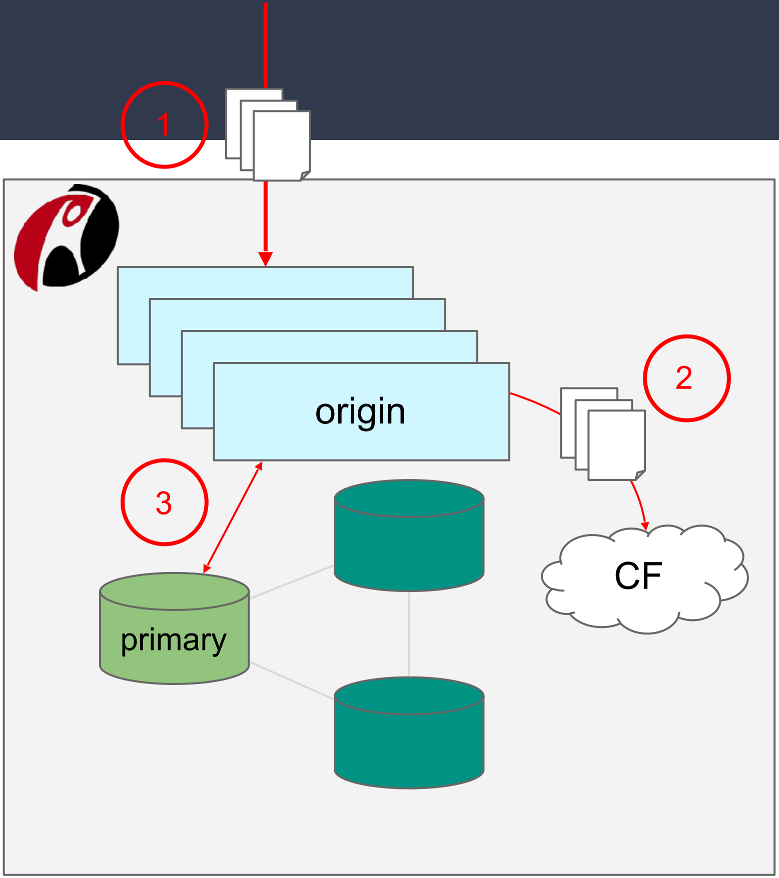
# Replicate it all

```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
}
```



# Replicate it all

```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
}
```



# Replicate it all

```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
  “m”: 1
}
```

Upload mask

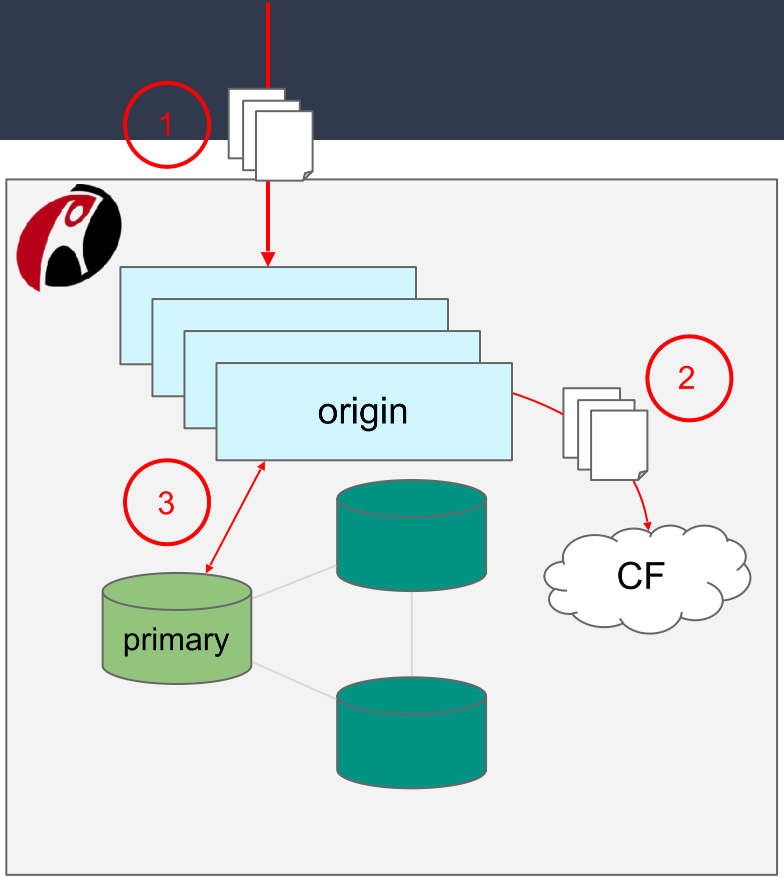
RAX  = 1
AWS = 2
GCP = 4

Example:

m = 6 → AWS & GCP

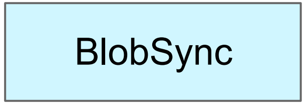
m = 3 → AWS & RAX

m = 1 → RAX only



# BlobSync

- Done out of band from the request cycle
- Constantly queries for unreplicated blobs
- Pulls object down, pushes to the other clouds
- Records progress and errors

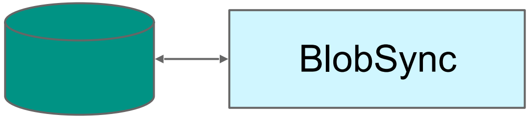


BlobSync

# BlobSync

```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
   "m": 1
}
```

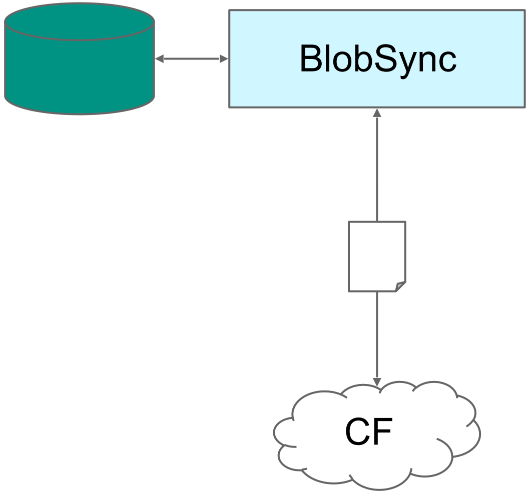
- Done out of band from the request cycle
- Constantly queries for unreplicated blobs
- Pulls object down, pushes to the other clouds
- Records progress and errors



# BlobSync

```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
    “m”: 1
}
```

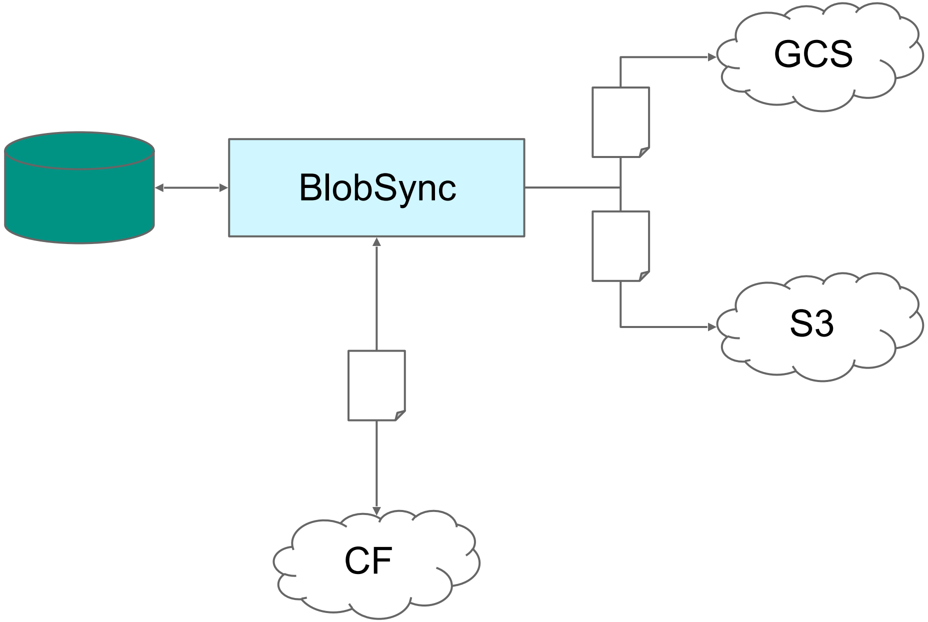
- Done out of band from the request cycle
- Constantly queries for unreplicated blobs
- Pulls object down, pushes to the other clouds
- Records progress and errors



# BlobSync

```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
    “m”: 1
}
```

- Done out of band from the request cycle
- Constantly queries for unreplicated blobs
- Pulls object down, pushes to the other clouds
- Records progress and errors

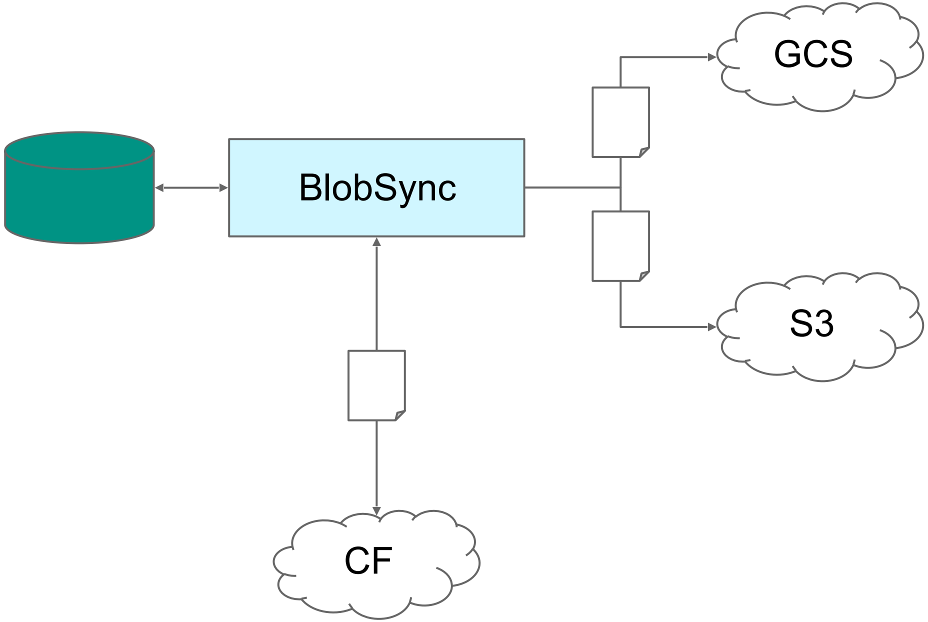


# BlobSync

```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
    "m": 1
}
```

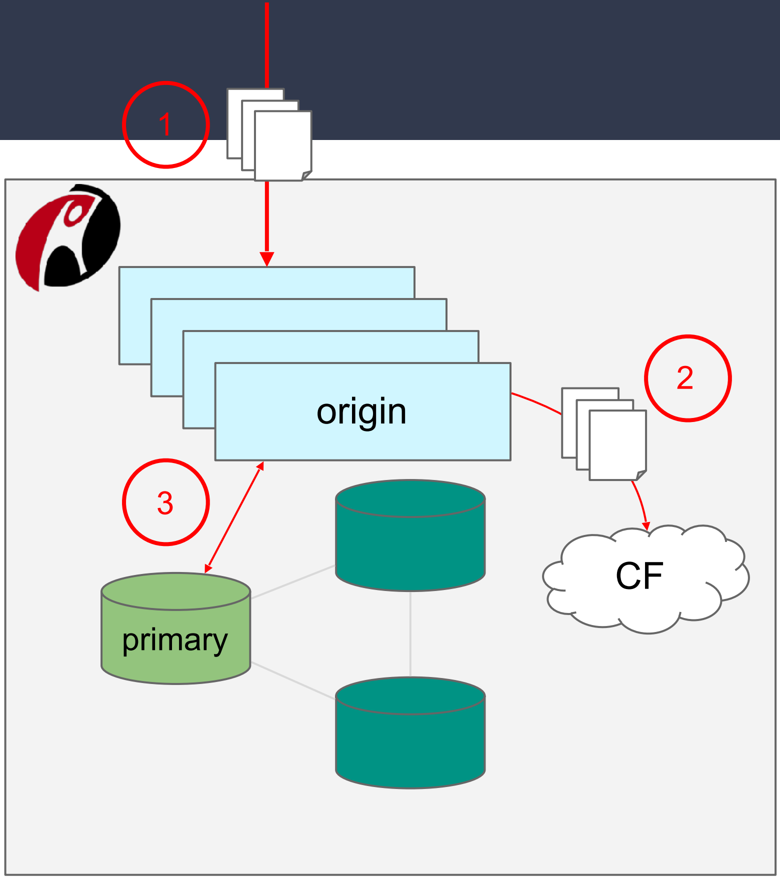
```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
    "m": 7
}
```

- Done out of band from the request cycle
- Constantly queries for unreplicated blobs
- Pulls object down, pushes to the other clouds
- Records progress and errors



# Replicate it all

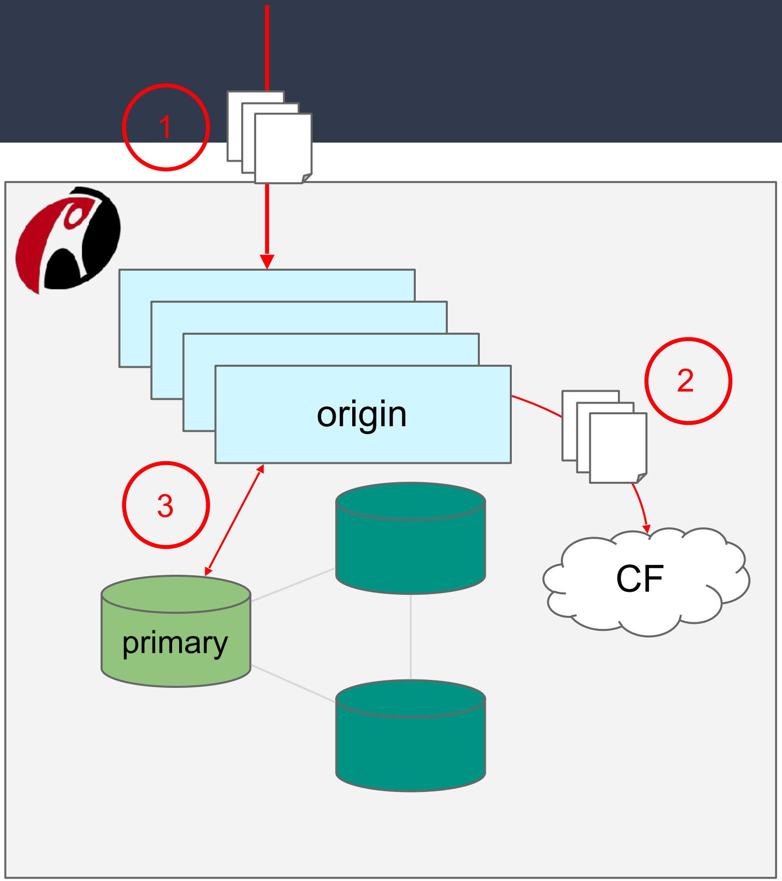
```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
  "m": 1
}
```



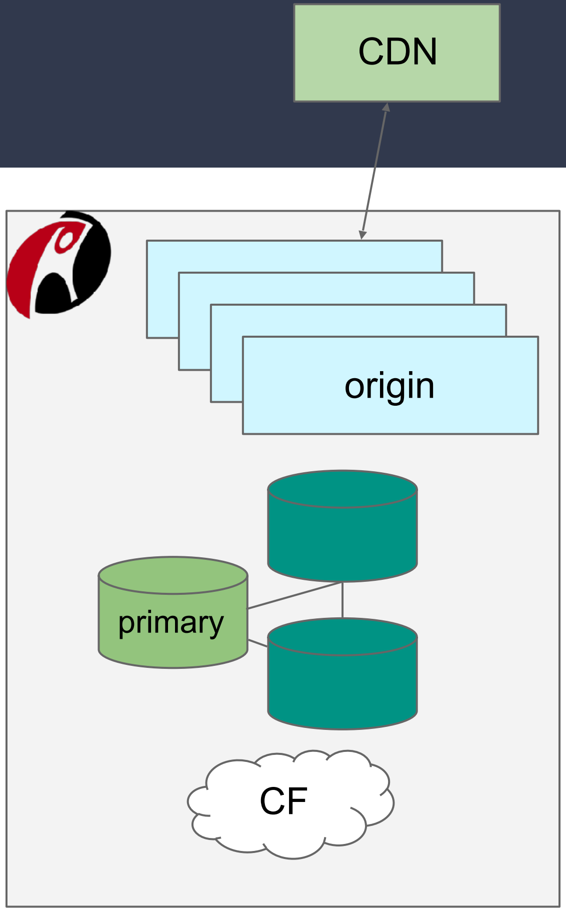
# Replicate it all

```
{
  "_id" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "size" : 9935,
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  "created_at" : ISODate("2018-06-07T21:02:29.240Z"),
  “m”: 1,
  “r”: true
}
```

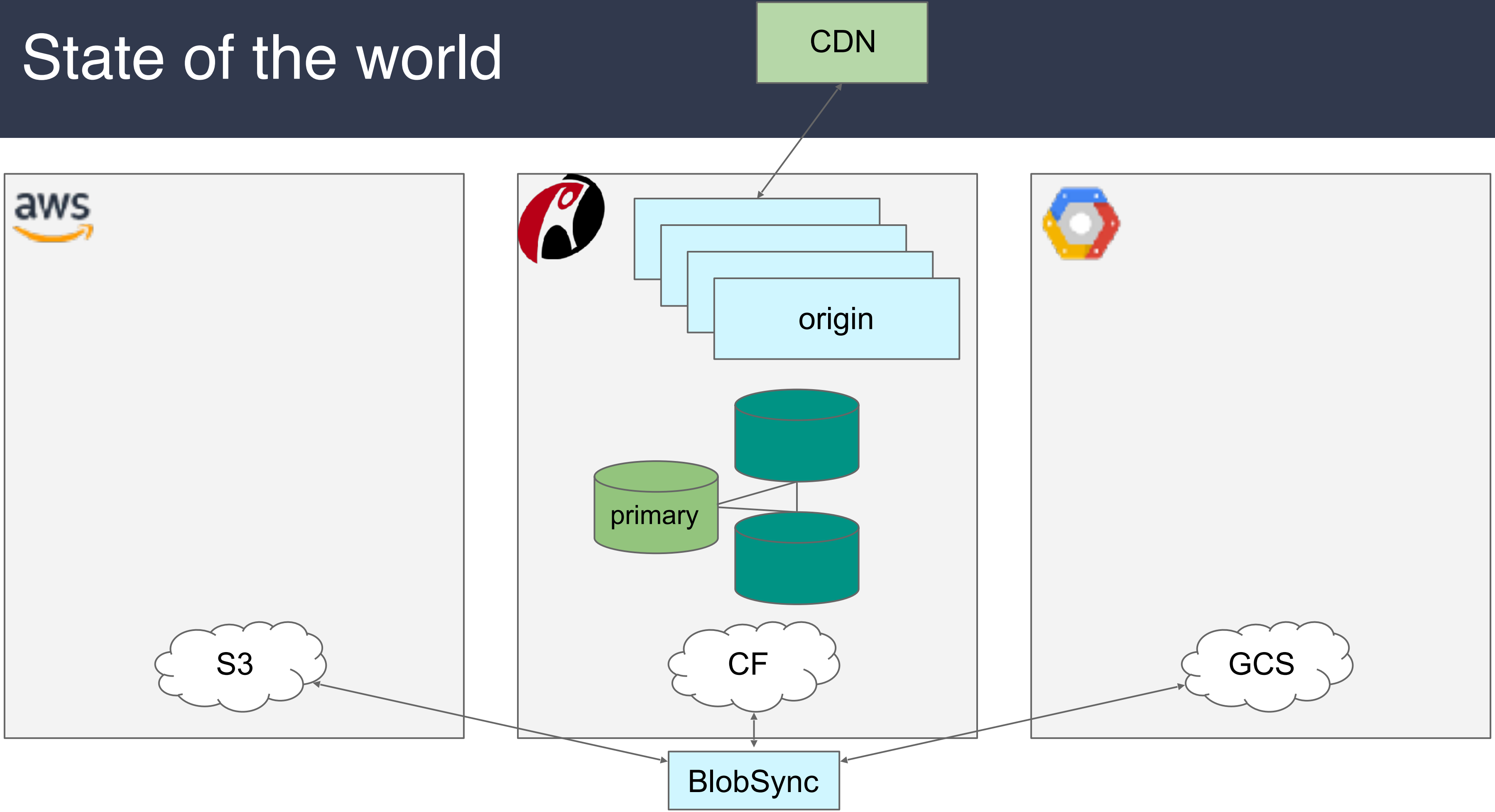
Replication Flag   Spares index in mongo



# State of the world



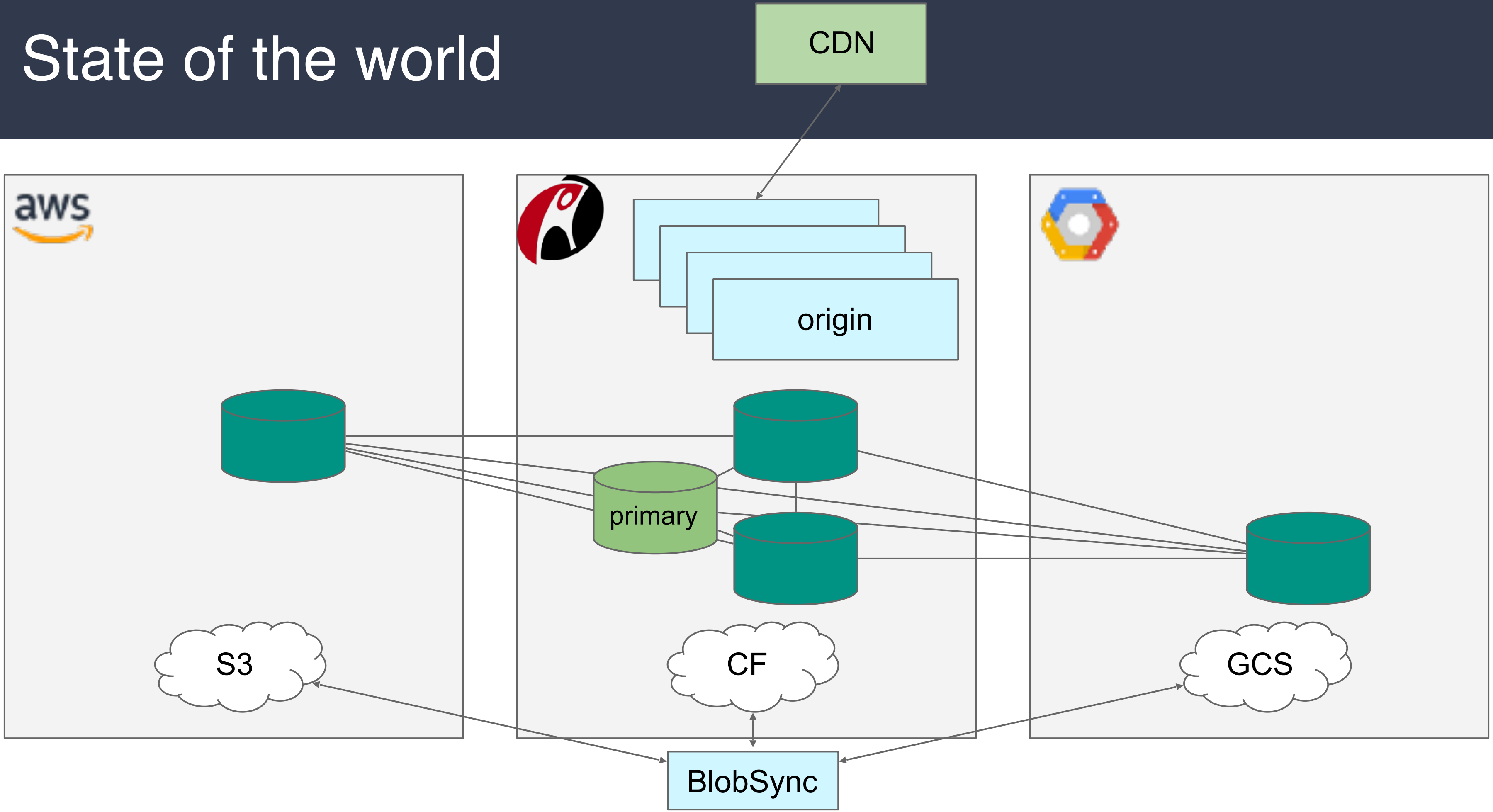
# State of the world



# Steps to Multicloud

1. Double check assumptions
2. Replicate all the objects
3. Prepare the database
4. Make the origin services cloud agnostic
5. Test everything
6. Do the actual cutover

# State of the world



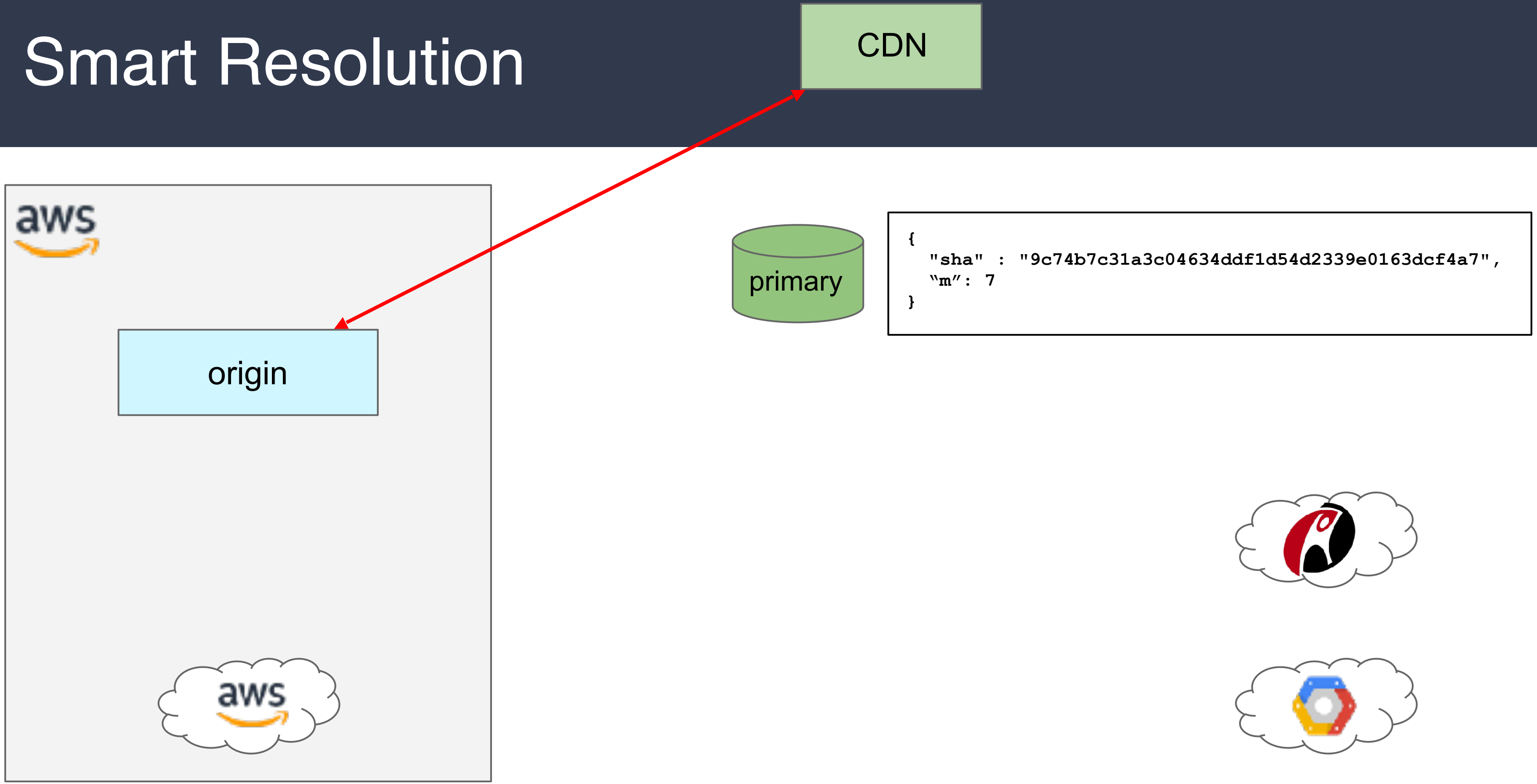
# Steps to Multicloud

1. Double check assumptions
2. Replicate all the objects
3. Prepare the database
4. Make the origin services cloud agnostic
5. Test everything
6. Do the actual cutover

# Cloud Agnostic Origin Services

- Generic cloud storage interface
- Automatic failover
- Prefer staying in cloud
- Forceable overrides

# Smart Resolution



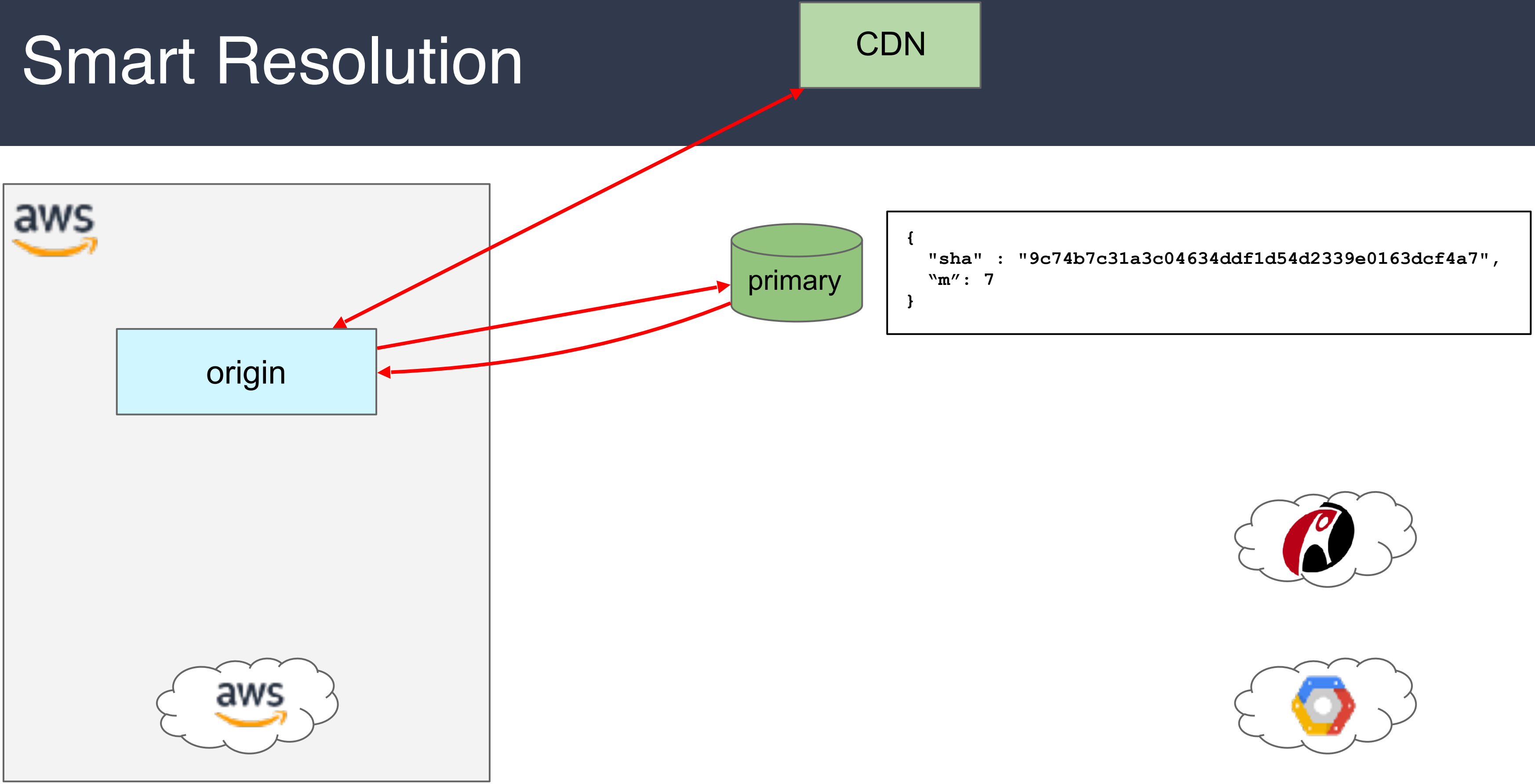
CDN

origin

primary

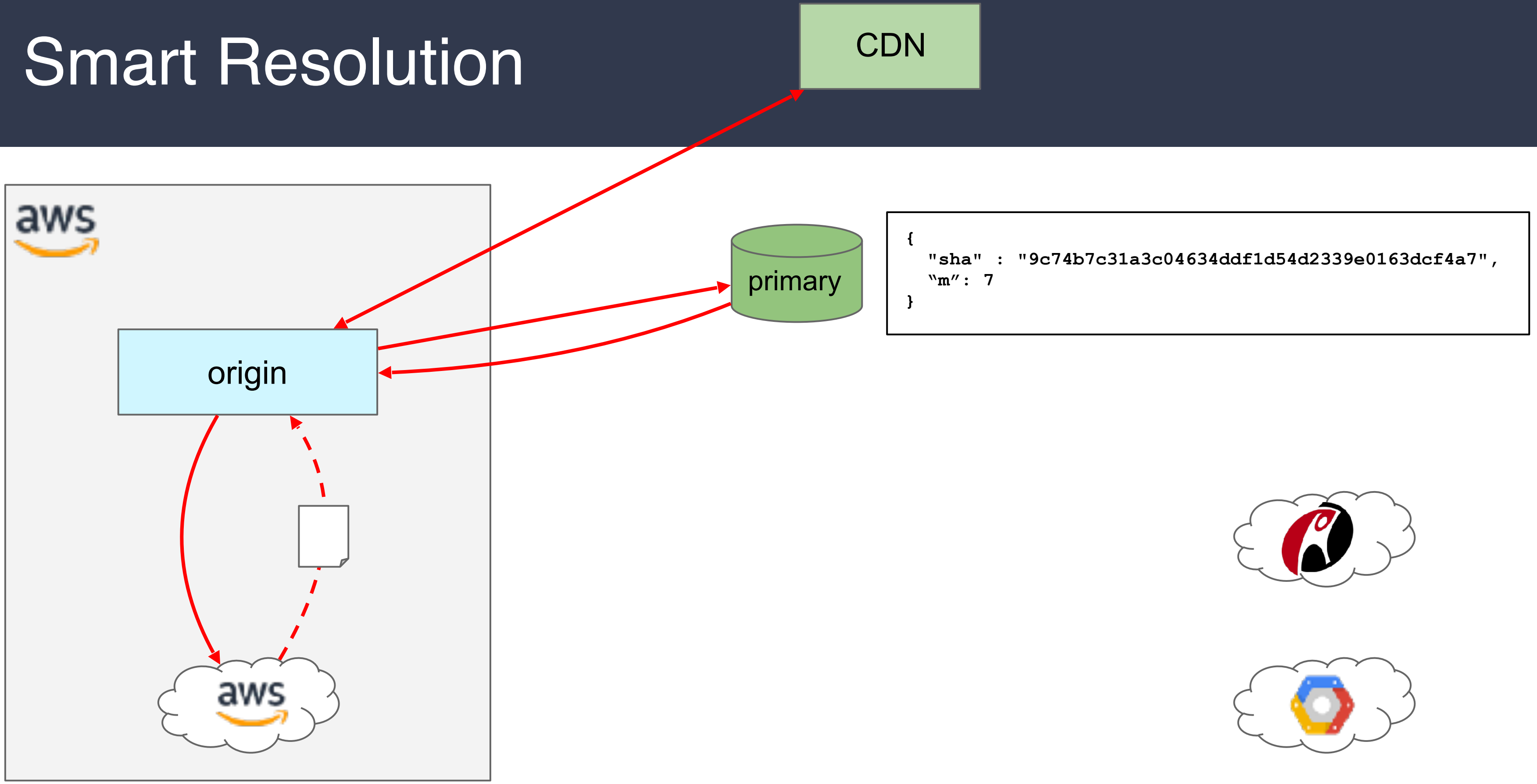
```
{
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  “m”: 7
}
```

# Smart Resolution



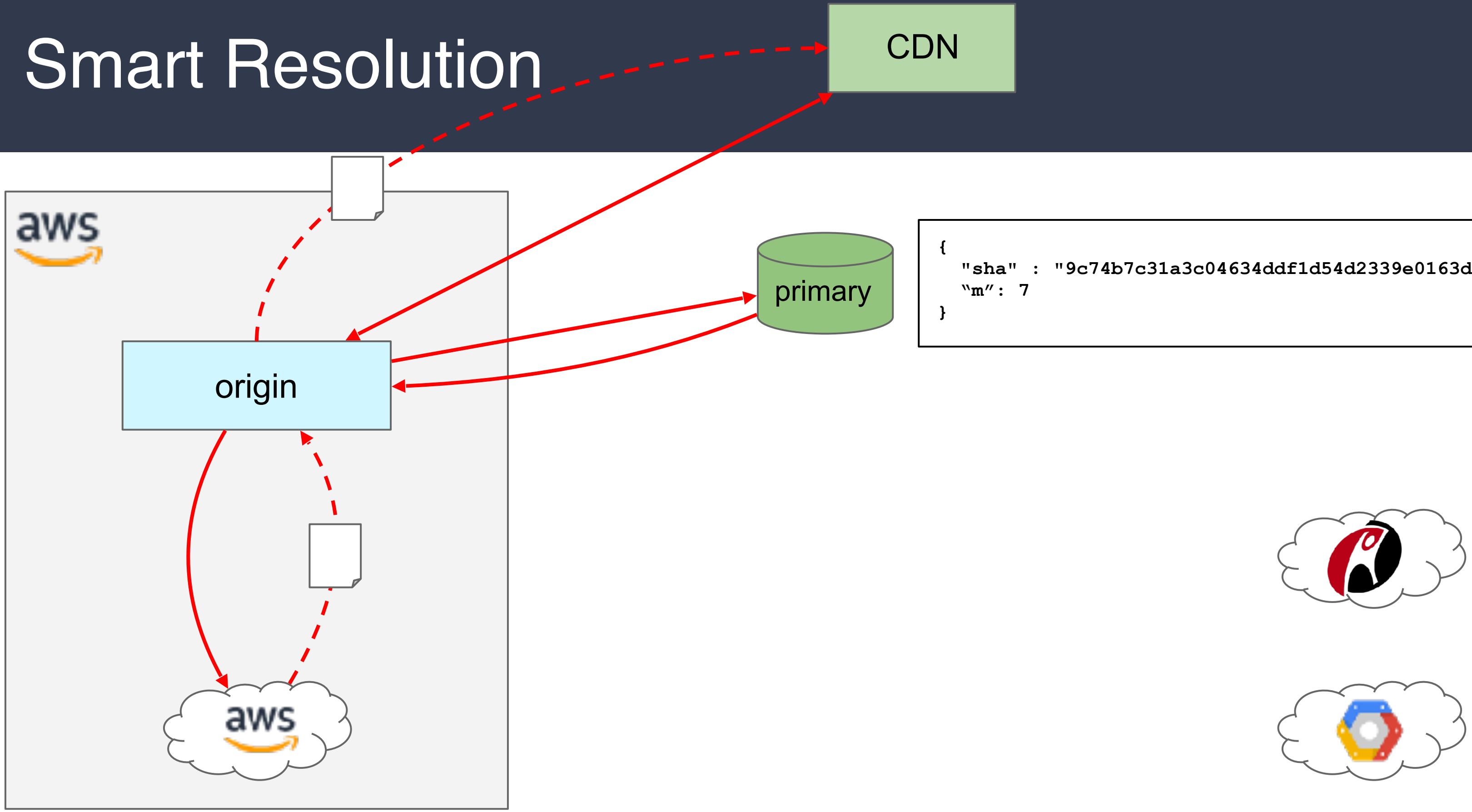
```
{
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  “m”: 7
}
```

# Smart Resolution



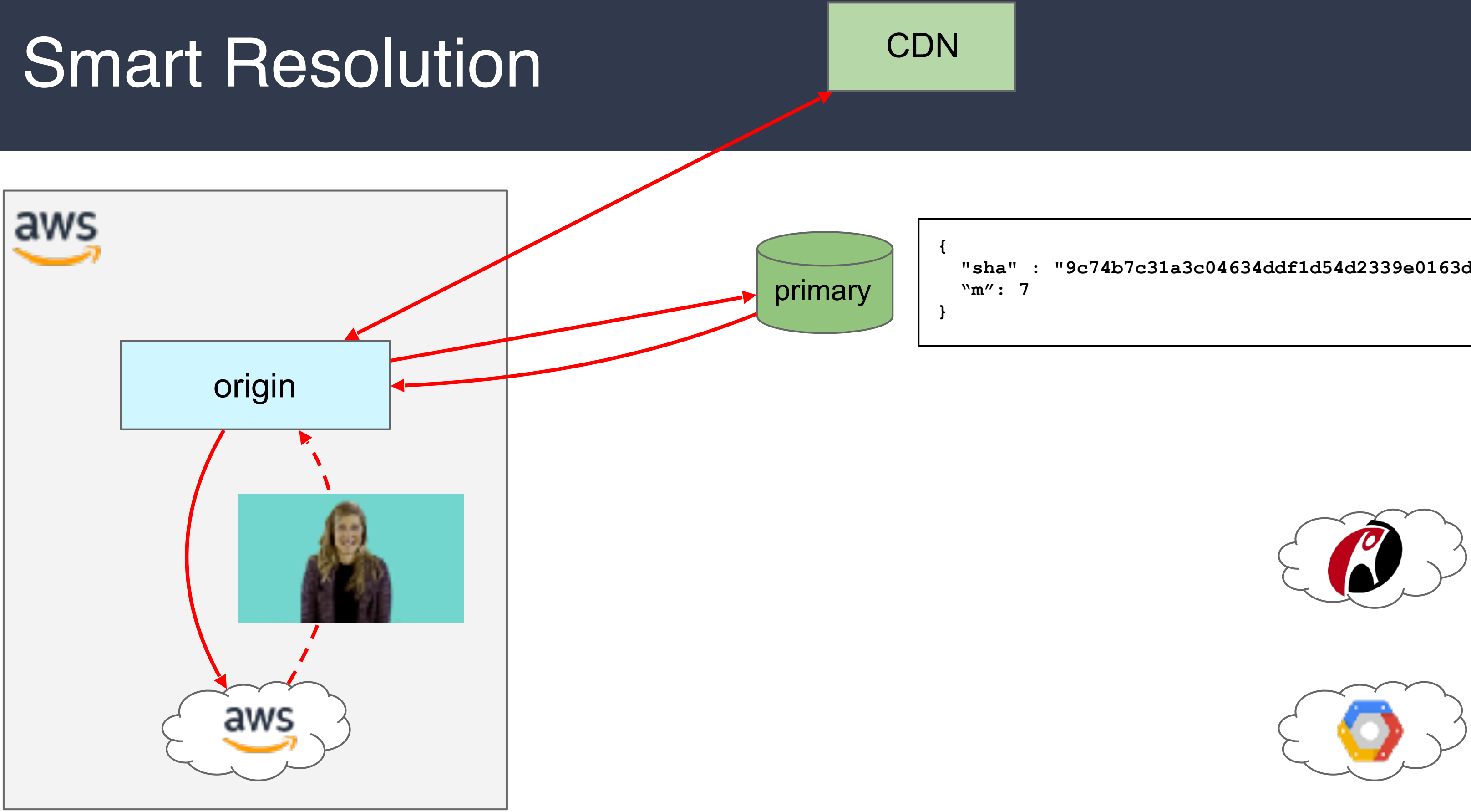
```
{
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  “m”: 7
}
```

# Smart Resolution



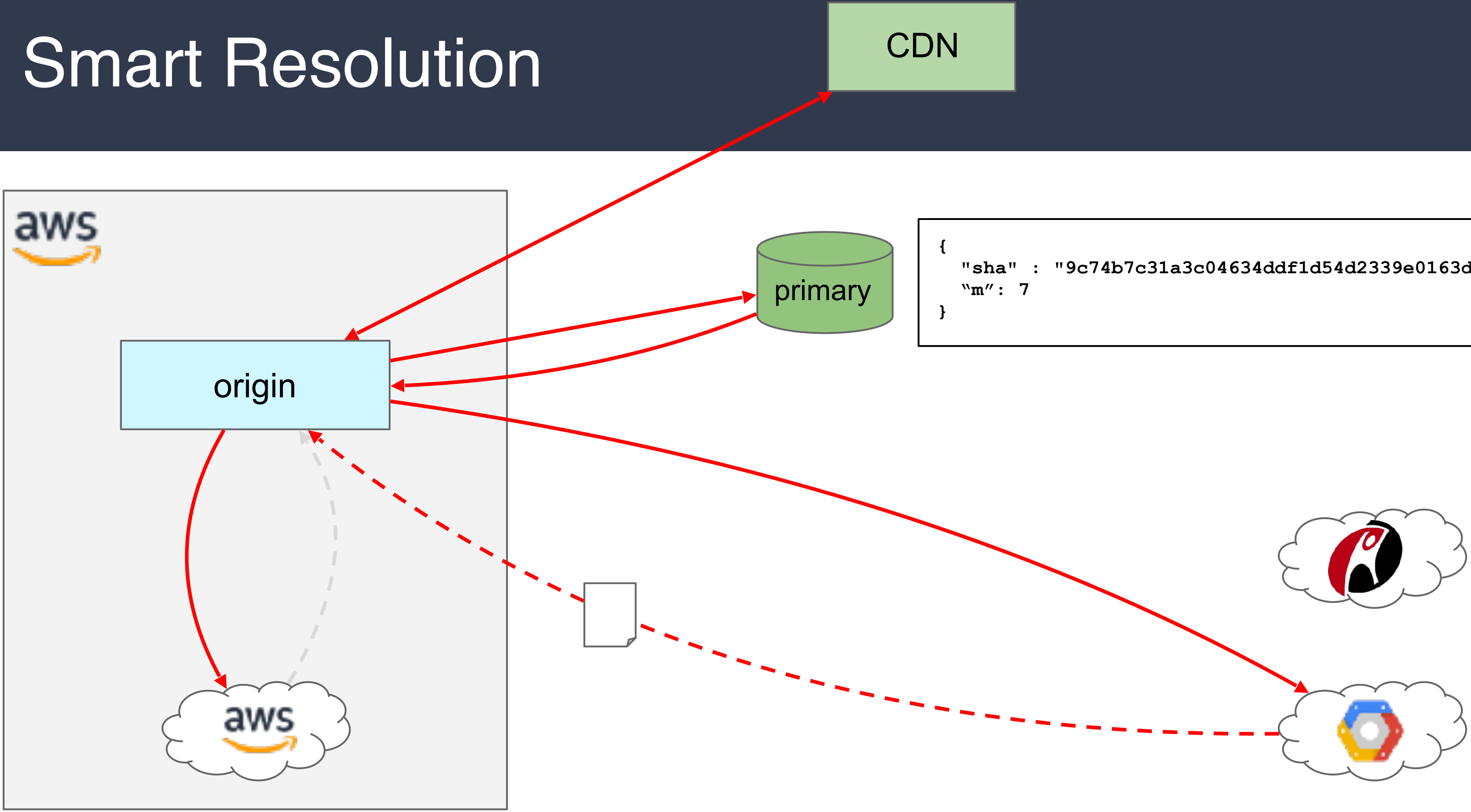
```
{
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  “m”: 7
}
```

# Smart Resolution



```
{
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  “m”: 7
}
```

# Smart Resolution



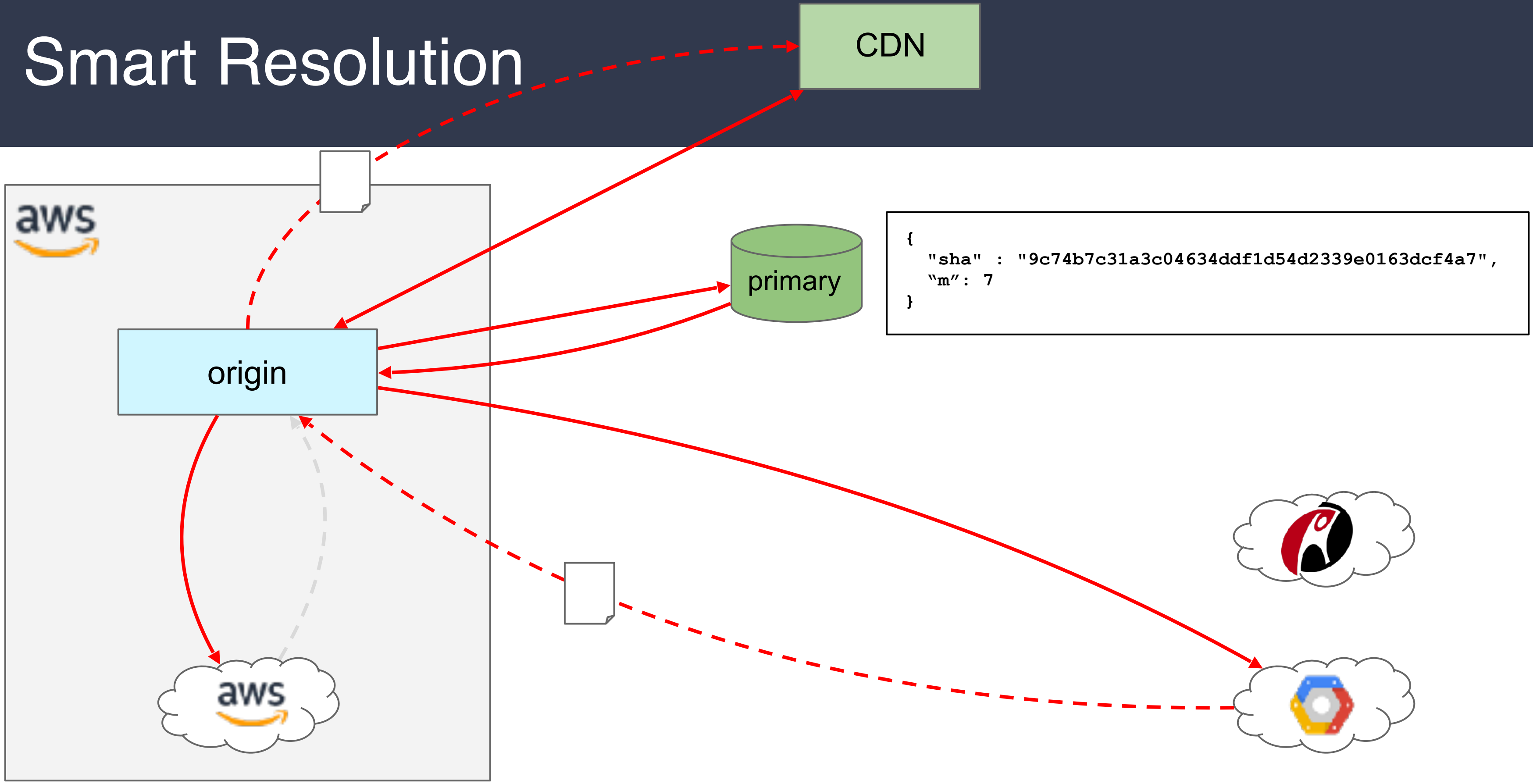
CDN

primary

origin

```
{
  "sha" : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  “m”: 7
}
```

# Smart Resolution

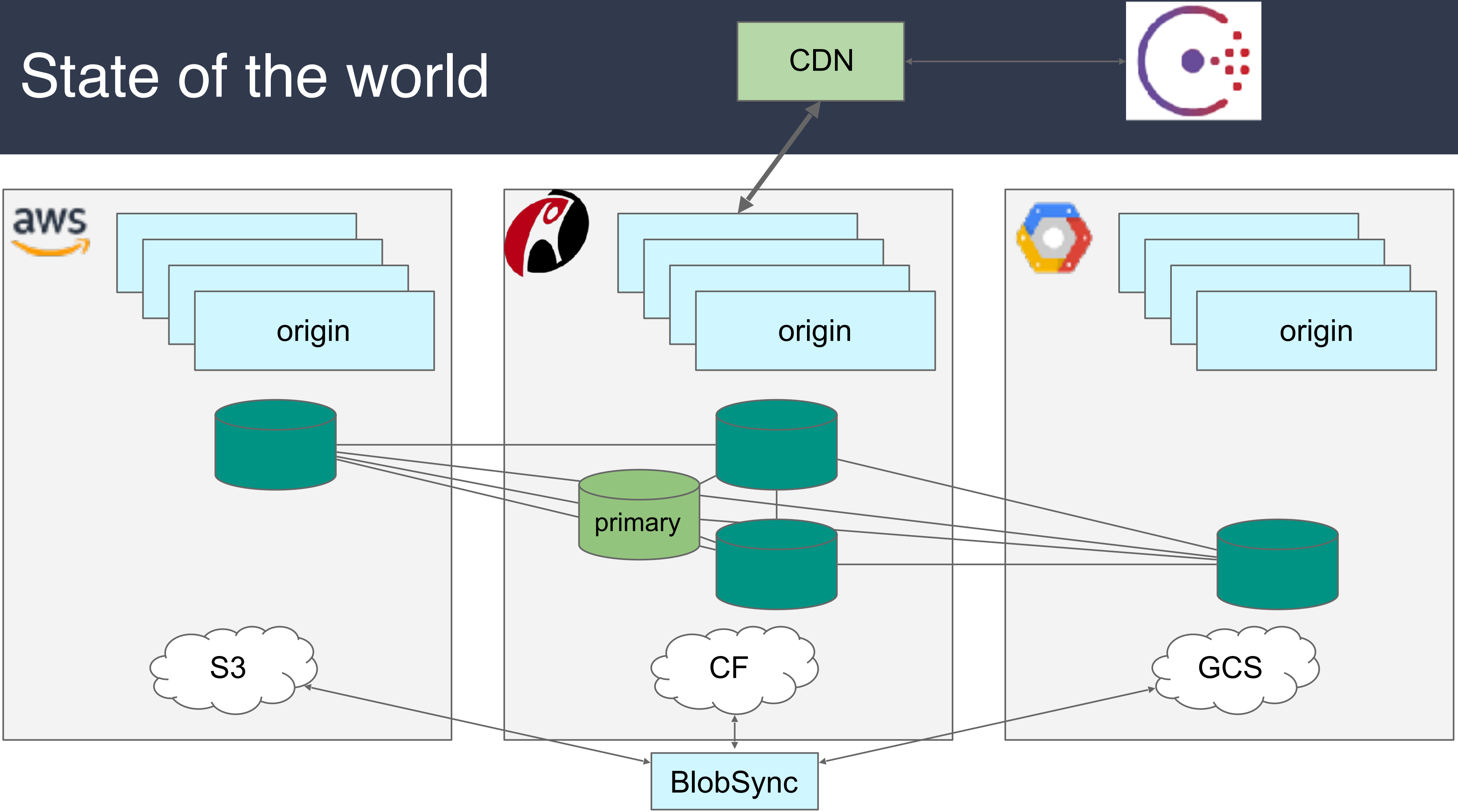


```
{
  "sha"  : "9c74b7c31a3c04634ddf1d54d2339e0163dcf4a7",
  “m”: 7
}
```

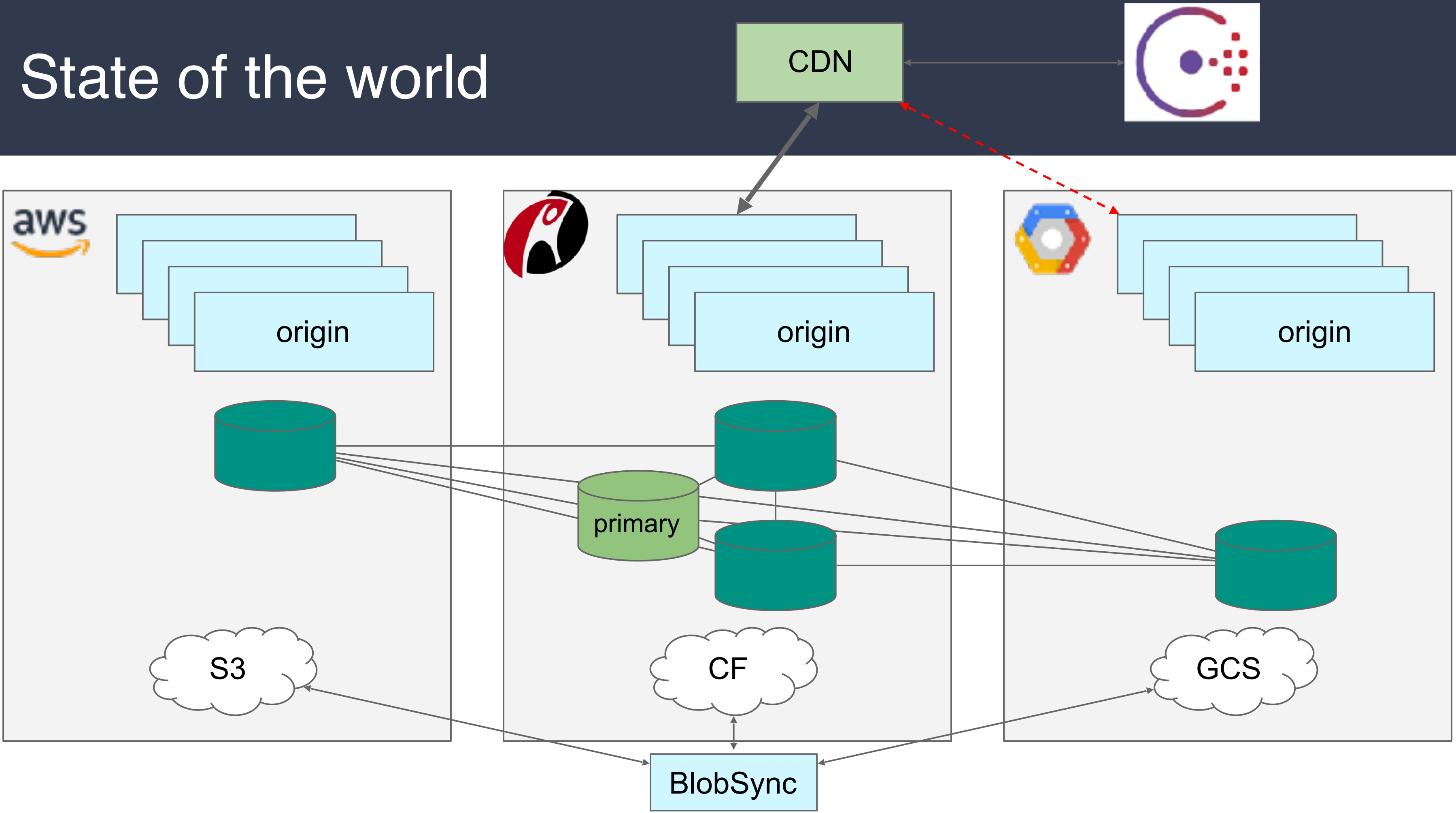
# Steps to Multicloud

1. Double check assumptions
2. Replicate all the objects
3. Prepare the database
4. Make the origin services cloud agnostic
5. Test everything
6. Do the actual cutover

# State of the world



# State of the world



# Steps to Multicloud

1. Double check assumptions
2. Replicate all the objects
3. Prepare the database
4. Make the origin services cloud agnostic
5. Test everything
6. Do the actual cutover

# Steps to Multicloud

1. Double check assumptions

the objects

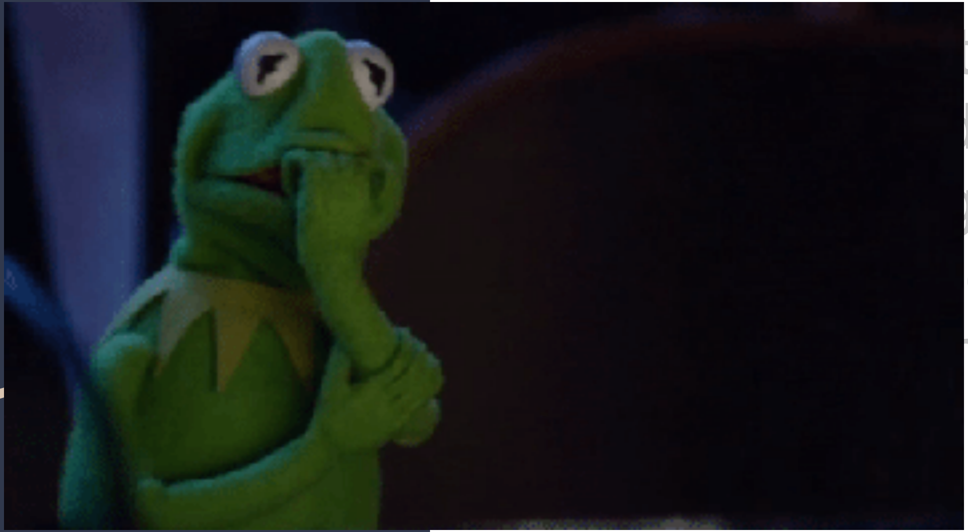
database

gin services

c

ng

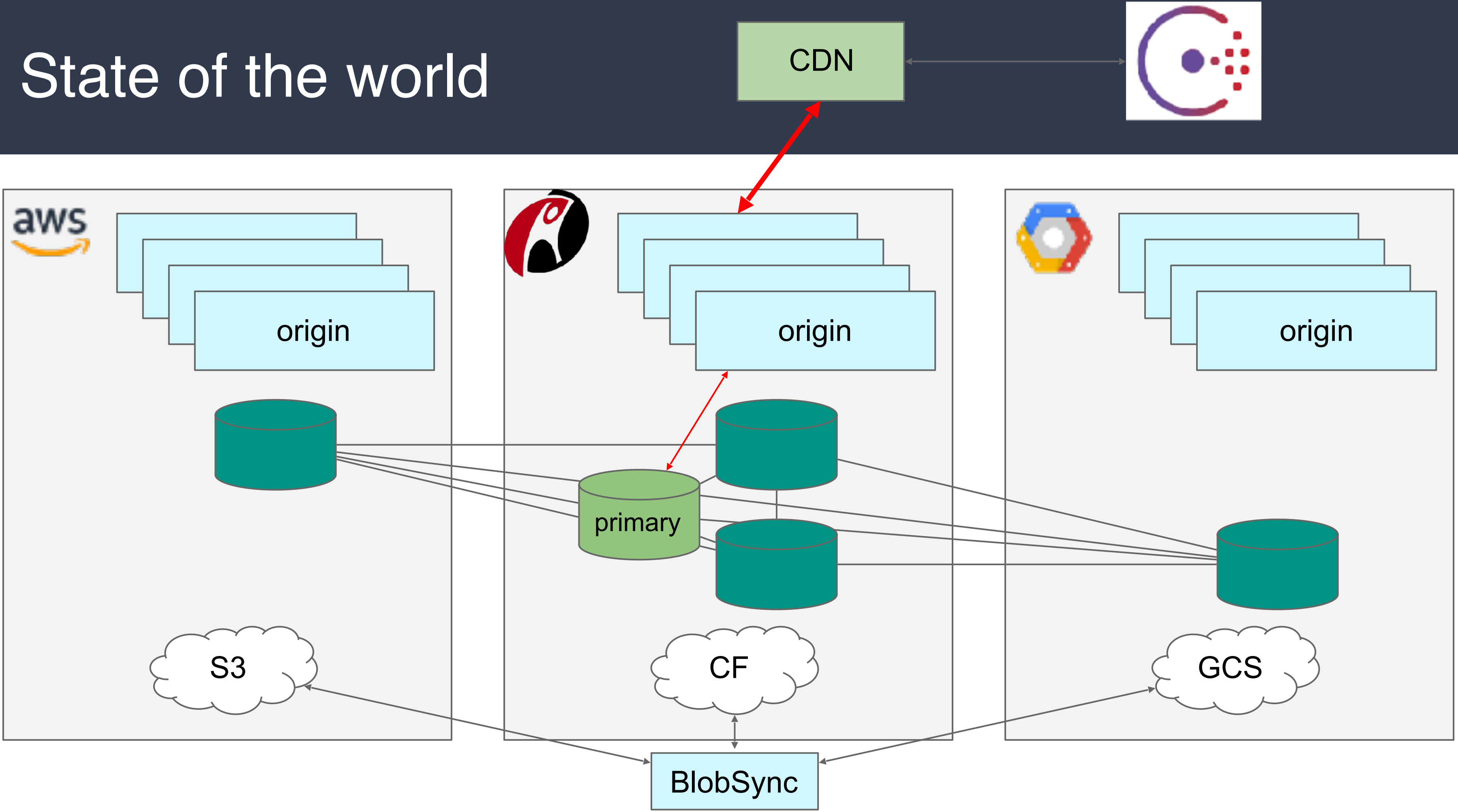
cutover



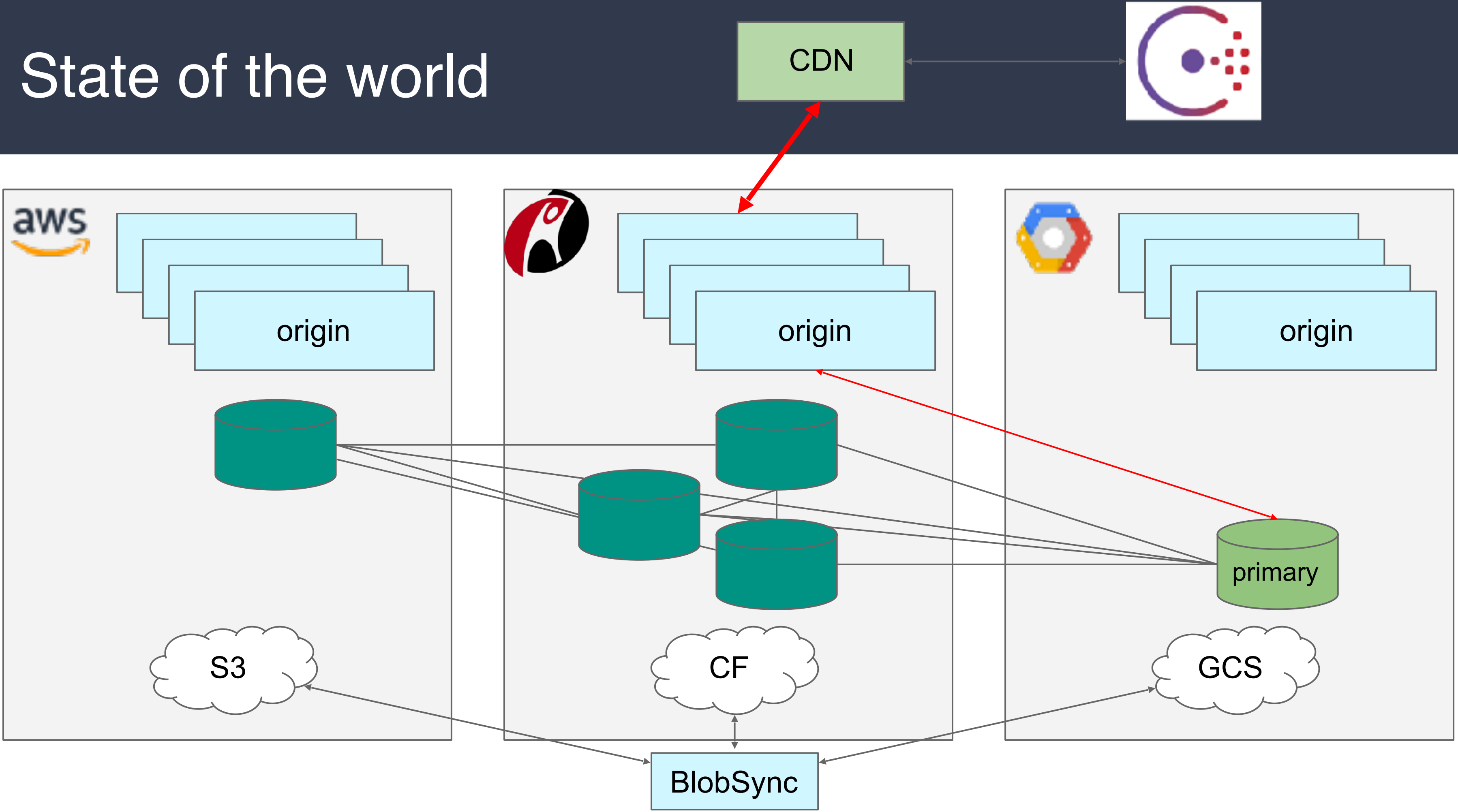
# Pulling the trigger

1. Spin up enough origin services
2. Fail over the DB
3. Update the consul entry
4. Aggressively stare at monitors

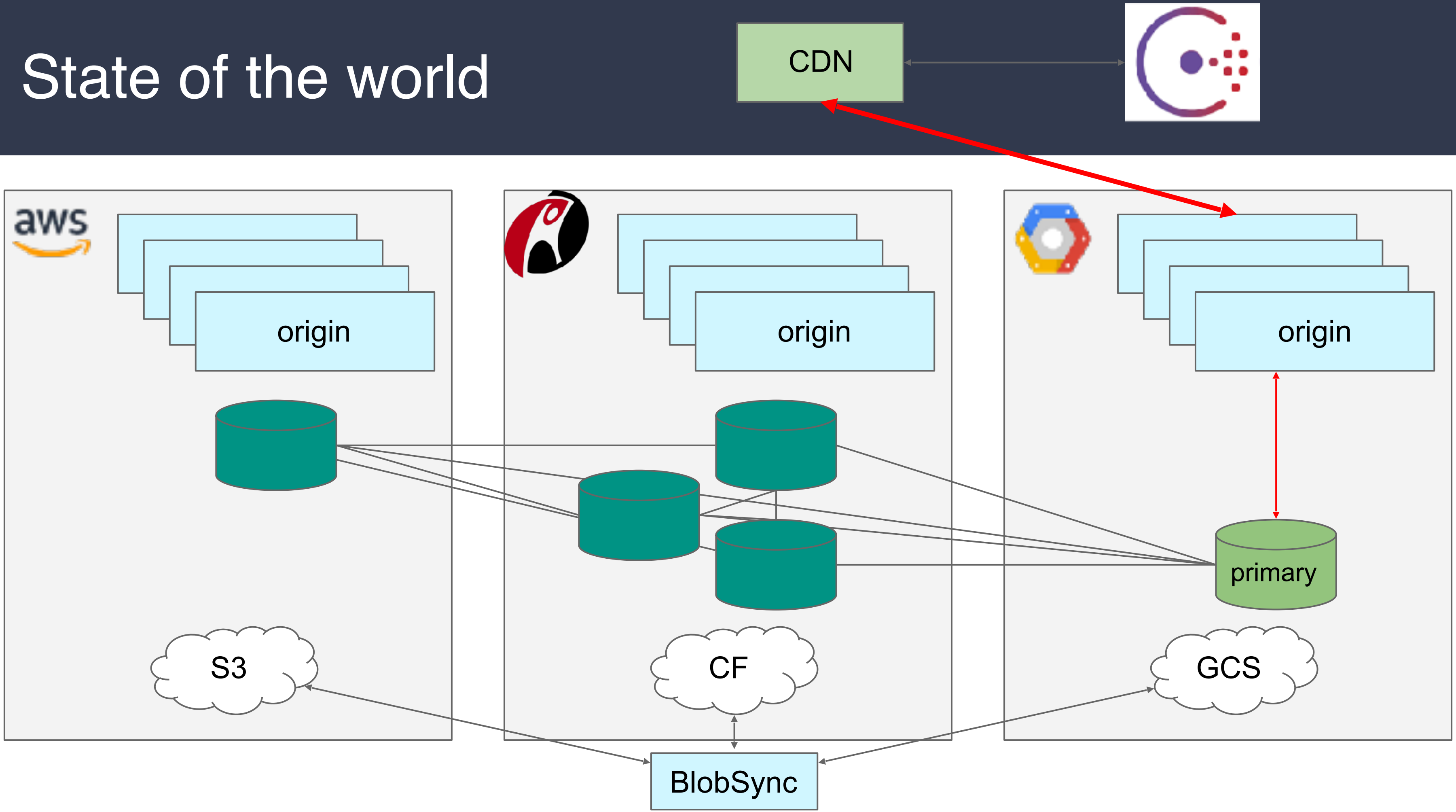
# State of the world

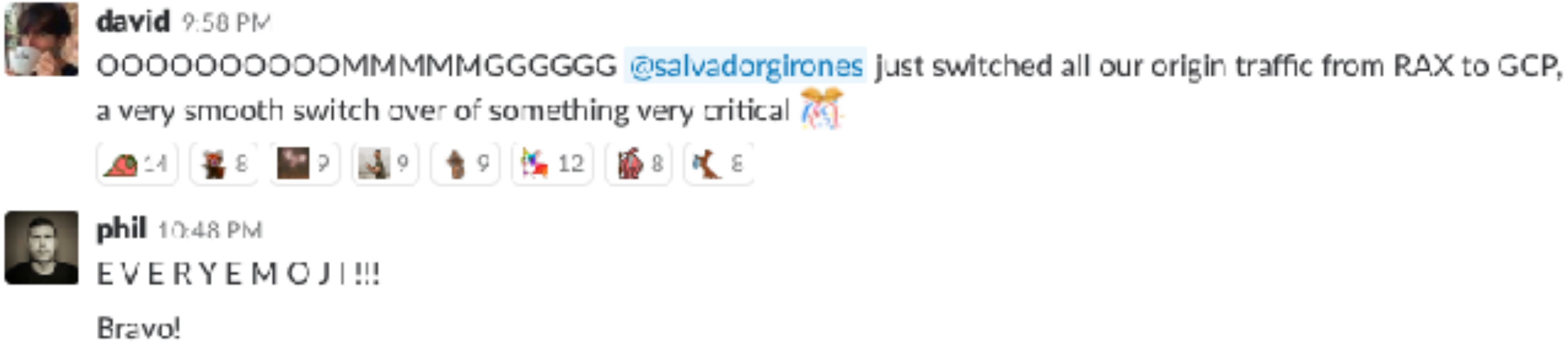


# State of the world



# State of the world





**david** 9:58 PM

OOOOOOOOOOMMMMMGGGGGG @salvadorgirones just switched all our origin traffic from RAX to GCP, a very smooth switch over of something very critical

**phil** 10:48 PM

EVERYEMOJI!!!

Bravo!

# Summary

- Redundant everything
- Cloud agnostic origin and CDN
- Programmable infrastructure
- Out of band replication
- Smart routing
- Automated failover

# So now what?

- Setup trickle of traffic to live standby
- Automate the traffic switch
- Speedup network scale up
- More monitoring

# WE
# ARE
# HIRING

# Find me to talk!

@ry_boflavin

rybit

ryan@