AWS Webinar

Exploiting Multi-Region Data Locality with Lambda@Edge

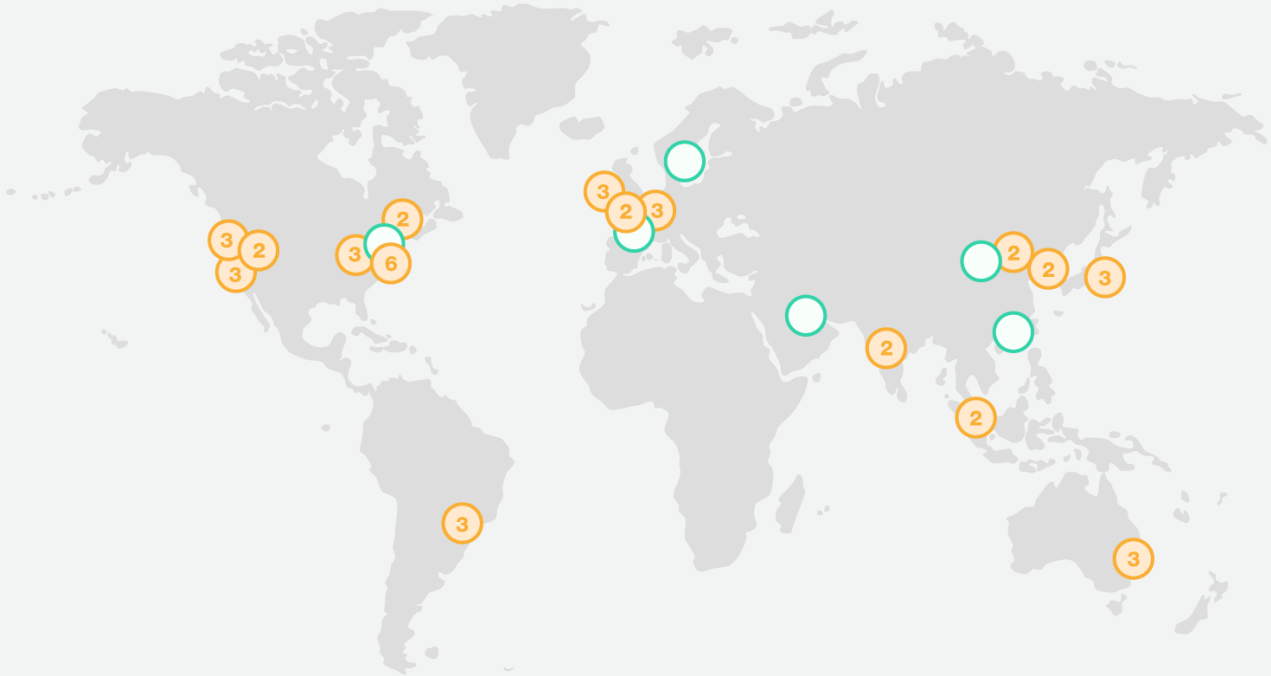
Vijay Potharla, Senior Product Manager, AWS
Seth Fitzsimmons, Technical Fellow, Radiant Earth Foundation

17 June 2019

What is covered in this session

- Lambda@Edge overview
    - What is Global Serverless Computing?
    - How Lambda@Edge works?
    - What can you do with Lambda@Edge?
- How Radiant Earth implemented multi-region data locality with Lambda@Edge ?

AWS Locations: Where is Your Customer?



180 Points of Presence

69 cities

30 countries

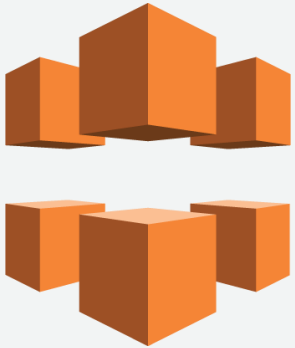
21 AWS Regions

AWS Lambda: Why Serverless?

Build and run applications without managing servers

- No servers to manage
- Run at scale
- Respond quickly to events
- Only pay for compute time that you use
- Developer productivity

Amazon CloudFront: Global Content Delivery



- Global, growing content delivery network
- High performance—speed up distribution of your static and dynamic web content to your users across the world.
- DDoS resiliency, scale, and cost effective
- Flexible and highly “programmable”, with AWS Lambda@Edge

Global Serverless:
Run Lambda Functions Across AWS Locations

Benefits of AWS Lambda@Edge

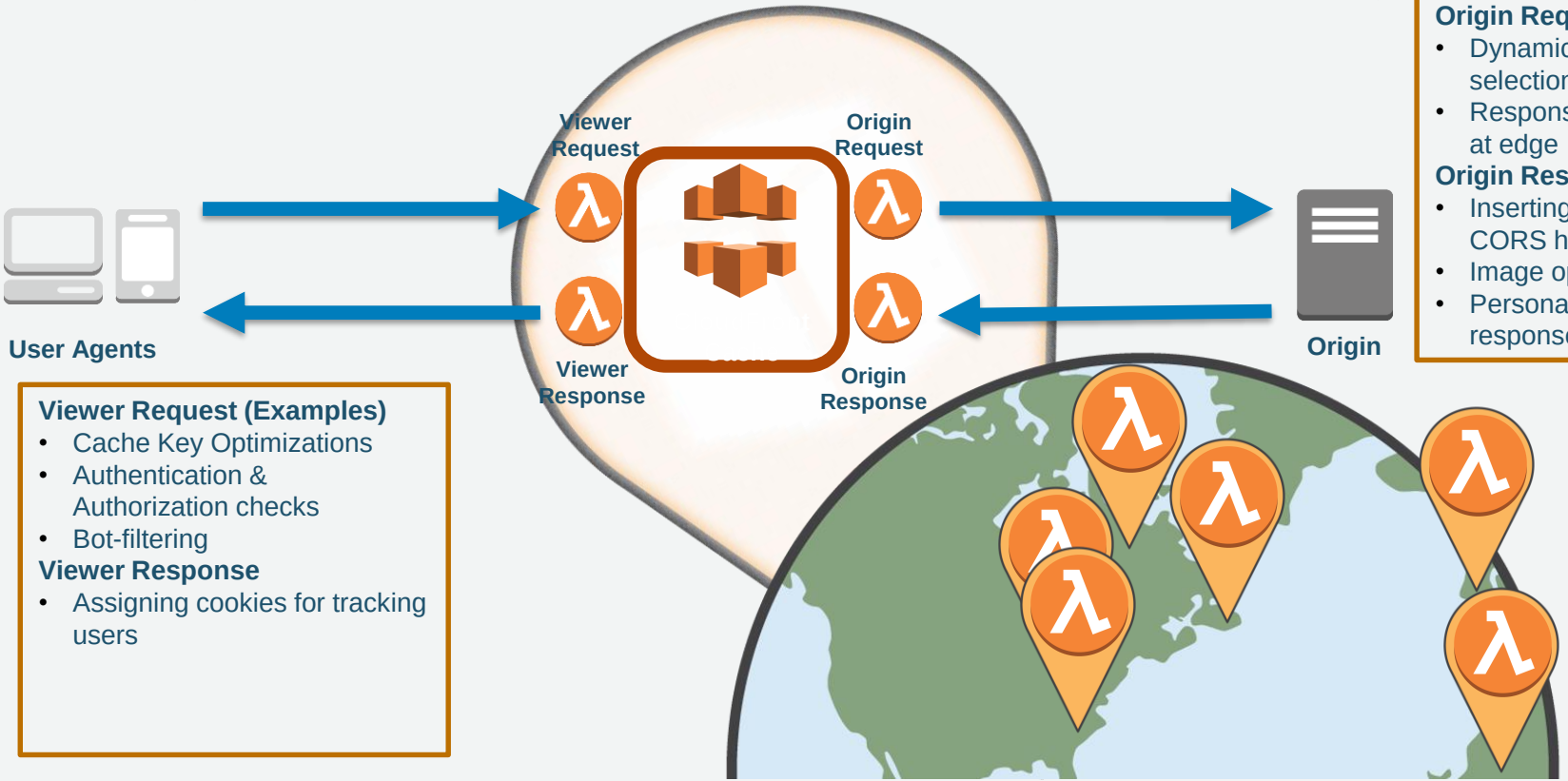
No servers to manage

Continuous scaling

Never pay for idle

Global presence

Programmable CloudFront; Lambda@Edge



Origin Request
- Dynamic origin selection
- Response generation at edge

Origin Response
- Inserting security or CORS headers
- Image optimizations
- Personalizing error responses

Viewer Request (Examples)
- Cache Key Optimizations
- Authentication & Authorization checks
- Bot-filtering

Viewer Response
- Assigning cookies for tracking users

Exploiting Multi-Region Data Locality with Lambda@Edge

Seth Fitzsimmons for Radiant Earth Foundation

17 June 2019

Seth Fitzsimmons

Seth is an independent consultant who architects software solutions for raster imagery, data pipelines, and humanitarian needs.

Amongst other things, he is a Technical Fellow at Radiant Earth, focusing on tooling and evangelization of standards and specifications like COG (cogeo.org) and STAC (stacspec.org).

COG
CLOUD OPTIMIZED
GEOTIFF

SpatioTemporal
Asset Catalog

seth@pacatlas.com

@mojodna

Radiant Earth Foundation

EARTH IMAGERY FOR IMPACT

About

- Non-profit, founded in 2016
- Funded by Omidyar Network, Gates Foundation, Schmidt Futures & McGovern Foundation

Vision

- Machine Learning and EO for Positive Impact

Expected Impact

- Democratize ML data & models
- High quality data for better SDG monitoring & decision-making
- Increase use of ML on EO

Services

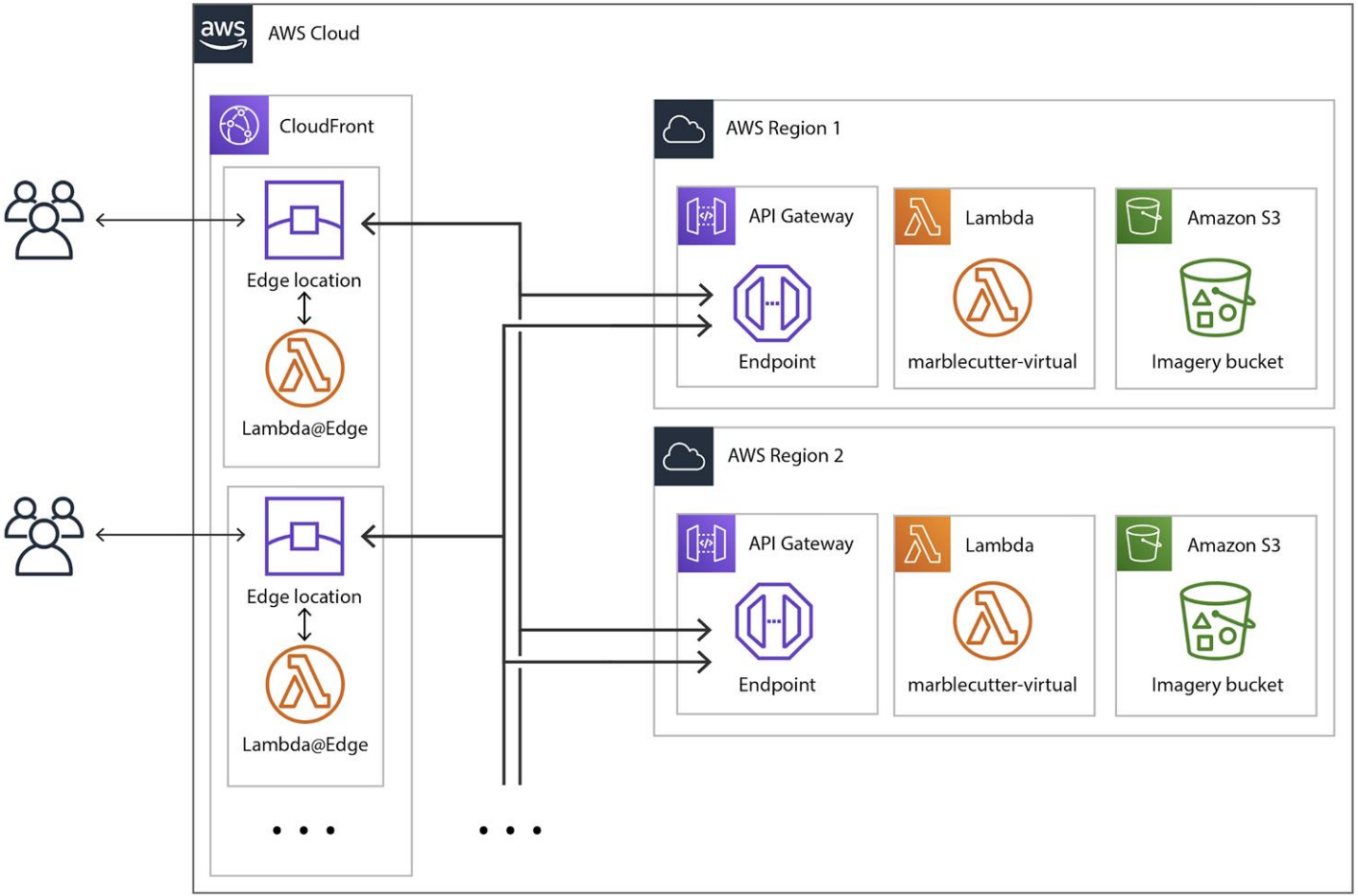
Library: Training Datasets and Models

Community: Best Practices, Standards (STAC), Fellowships, Events, Publications

Raise awareness on Data, Policy, Ethics & EO Market

Why we're here today:

https://tiles.rdnt.io


AWS Cloud
CloudFront
Edge location
Lambda@Edge
Edge location
Lambda@Edge
AWS Region 1
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket
AWS Region 2
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket

How it works

1

Rural Oregon

Request for tile is made to nearest CloudFront edge location.

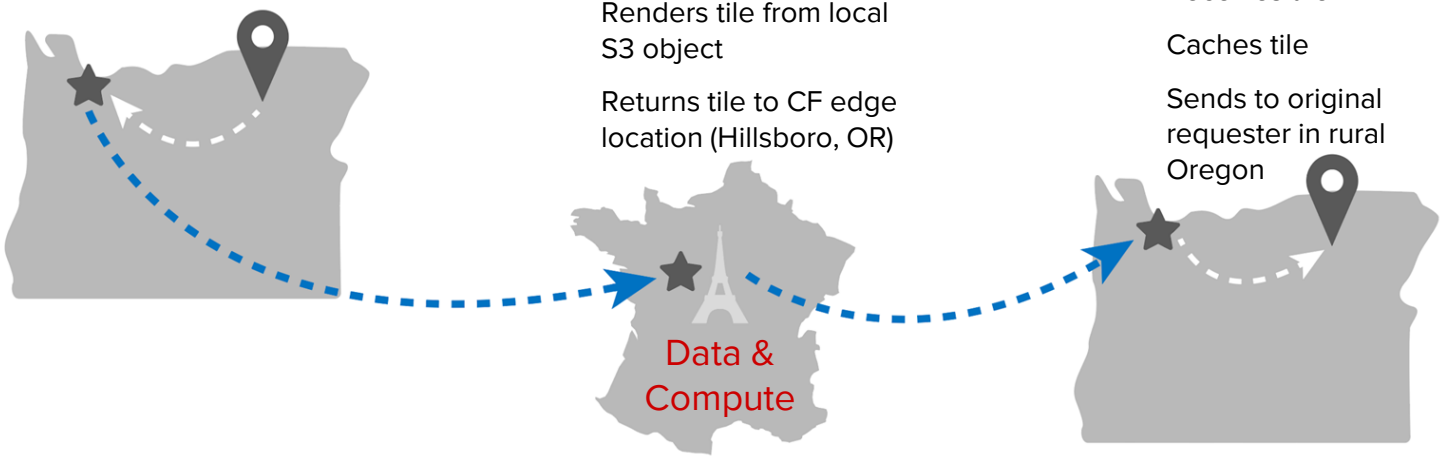
2

Hillsboro, Oregon

Lambda@Edge receives request

Determines the source file's region (`eu-west-3` (Paris))

Sends request to API Gateway / Lambda function in `eu-west-3`



3

Paris, France

Receives request

Renders tile from local S3 object

Returns tile to CF edge location (Hillsboro, OR)

4

Hillsboro, Oregon

Receives tile

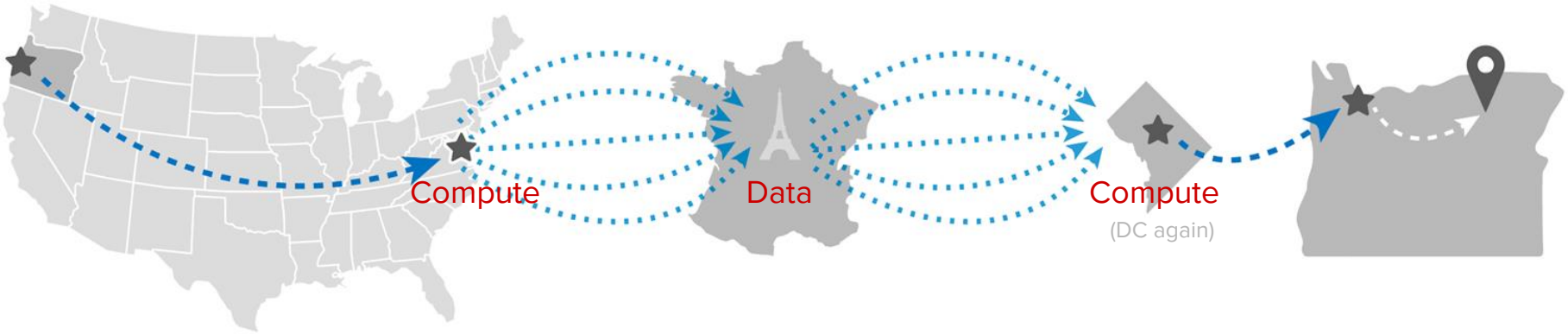
Caches tile

Sends to original requester in rural Oregon

Example of how it works

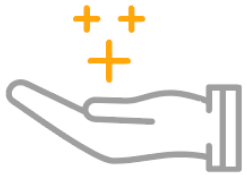
- Initially computed further from **users**, but closer to **data**
- Will be cached **close to users**

Before Lambda@Edge



- Data and compute functions are in different regions
- Multiple requests must be made between regions. This incurs latency and data transfer costs.

Benefits of Lambda@Edge + tiles.rdnt.io



- Tangible benefit for adopting a standardized format [Cloud-Optimized GeoTIFF (COG)]
- Able to browse multi-petabyte archives (HUGE) easily, interactively
- Affordable: free or cheap to access
  Affordable: costs nothing when not in use
  Affordable: minimize data transfer costs
- Access to imagery and data from AWS Public Dataset program

AWS Public Dataset Program

https://aws.amazon.com/opendata/public-datasets/

aws

Contact Sales Support English My Account Sign In to the Console

Products Solutions Pricing Documentation Learn Partner Network AWS Marketplace Explore More

AWS Public Dataset Program

The AWS Public Dataset Program covers the cost of storage for publicly available high-value cloud-optimized datasets. We work with data providers who seek to:

1. Democratize access to data by making it available for analysis on AWS.
2. Develop new cloud-native techniques, formats, and tools that lower the cost of working with data.
3. Encourage the development of communities that benefit from access to shared datasets.

You can see examples of datasets supported by the AWS Public Dataset Program on the Registry of Open Data on AWS.

Requirements

To share your dataset through the AWS Public Dataset Program, you must agree to the AWS Public Dataset Program Terms and Conditions, which are available at: https://aws.amazon.com/public-datasets/terms/

Application process

AWS evaluates applications to the AWS Public Dataset Program every three months. The quarterly deadlines for submitting AWS Public Dataset Program applications are: March 31, June 30, September 30, and December 31 (or the first business day after those dates). Application decisions are typically communicated within one month following the applicable quarterly deadline.

If we bring your dataset into the AWS Public Dataset Program, we will cover the costs of storage and data transfer for a period of two years, in accordance with the AWS Public Dataset Program Terms and Conditions.

To apply, simply fill out the secure form at https://application.opendata.aws

Registry of Open Data on AWS

https://registry.opendata.aws

Registry of Open Data on AWS

About

This registry exists to help people discover and share datasets that are available via AWS resources. Learn more about sharing data on AWS.

See all usage examples for datasets listed in this registry.

Search datasets (currently 110 matching datasets)

Search datasets

Add to this registry

If you want to add a dataset or example of how to use a dataset to this registry, please follow the instructions on the Registry of Open Data on AWS GitHub repository.

Unless specifically stated in the applicable dataset documentation, datasets available through the Registry of Open Data on AWS are not provided and maintained by AWS. Datasets are provided and maintained by a variety of third parties under a variety of licenses. Please check dataset licenses and related documentation to determine if a dataset may be used for your application.

Sentinel-2

earth observation | satellite imagery | geospatial | natural resource | sustainability | disaster response

The Sentinel-2 mission is a land monitoring constellation of two satellites that provide high resolution optical imagery and provide continuity for the current SPOT and Landsat missions. The mission provides a global coverage of the Earth's land surface every 5 days, making the data of great use in on-going studies. L1C data are available from June 2015 globally. L2A data are available from April 2017 over wider Europe region and globally since December 2018.

Details →

Usage examples

- Using Vector tiles and AWS Lambda, we can build a really simple API to get Landsat and Sentinel images by Remote Pixel
- FME Landsat-8/Sentinel-2 File Selector by Safe Software
- Sentinel Playground by Sinergise
- Exploring the Chile wildfires with Landsat and Sentinel-2 imagery by Timothy Whitehead
- EO Browser by Sinergise

See 16 usage examples →

Landsat 8

earth observation | satellite imagery | geospatial | natural resource | sustainability | disaster response

An ongoing collection of satellite imagery of all land on Earth produced by the Landsat 8 satellite.

Details →

Usage examples

- EOS Land Viewer by Earth Observing System
- Development Seed Geolambda by Matthew Hanson
- Using Vector tiles and AWS Lambda, we can build a really simple API to get Landsat and Sentinel images by Remote Pixel

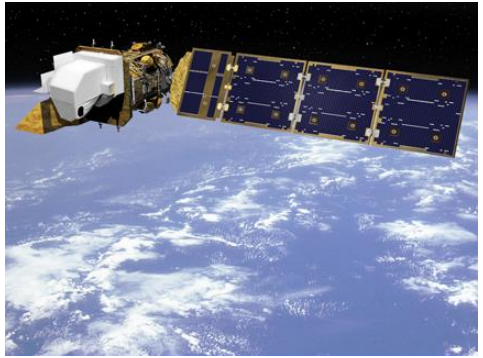
Amazon Sustainability Data Initiative

blog.aboutamazon.com/sustainability/unlocking-sustainability-insights-around-the-world

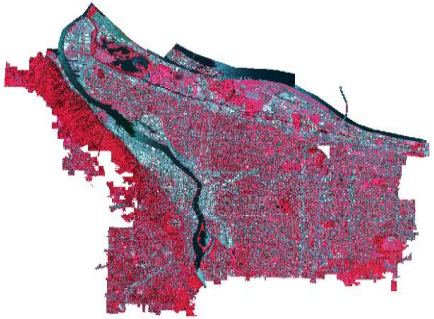
aws.amazon.com/blogs/publicsector/geo-diverse-open-training-data-as-a-global-public-good/

Sentinel-2

Landsat-8

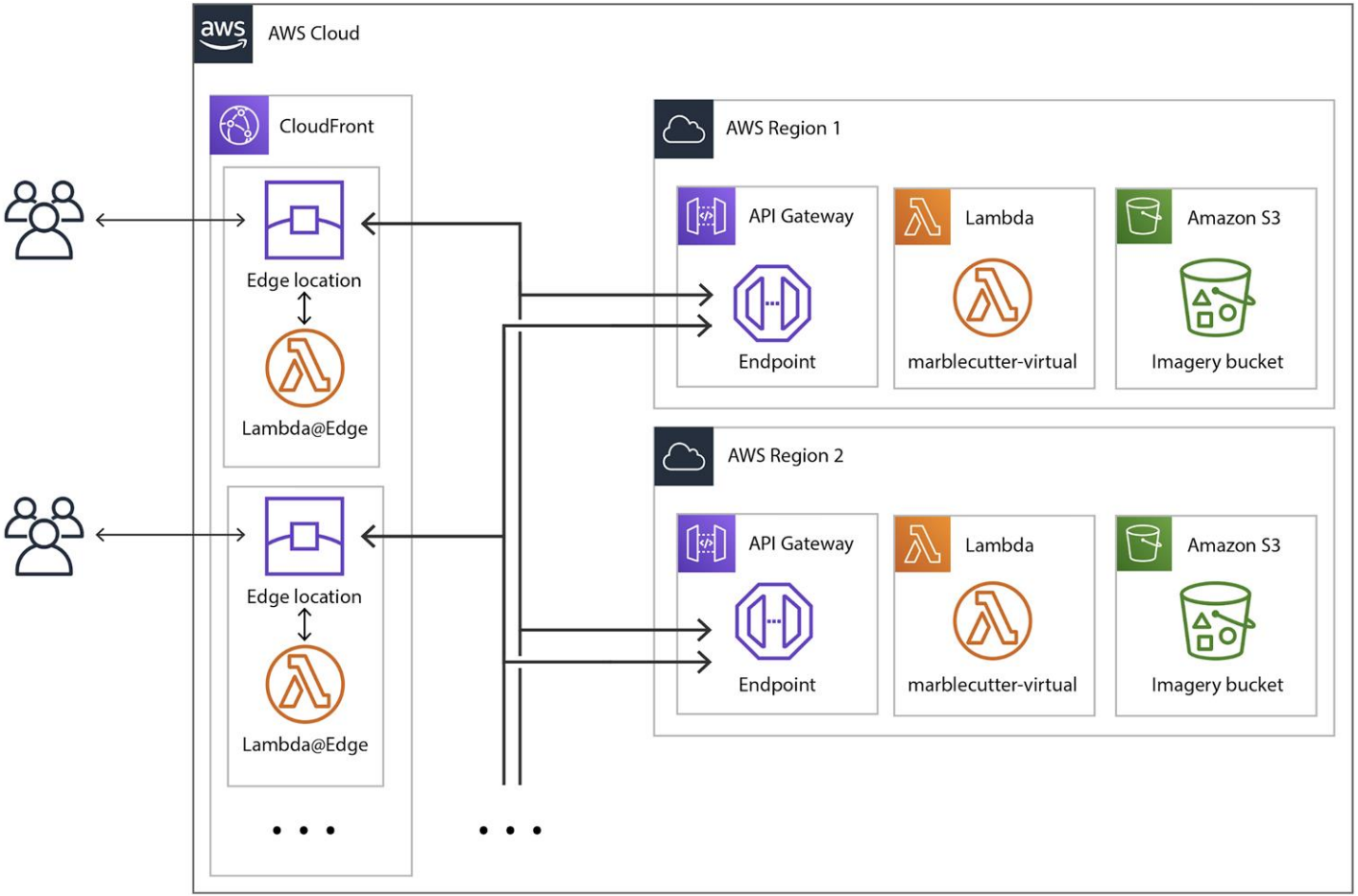


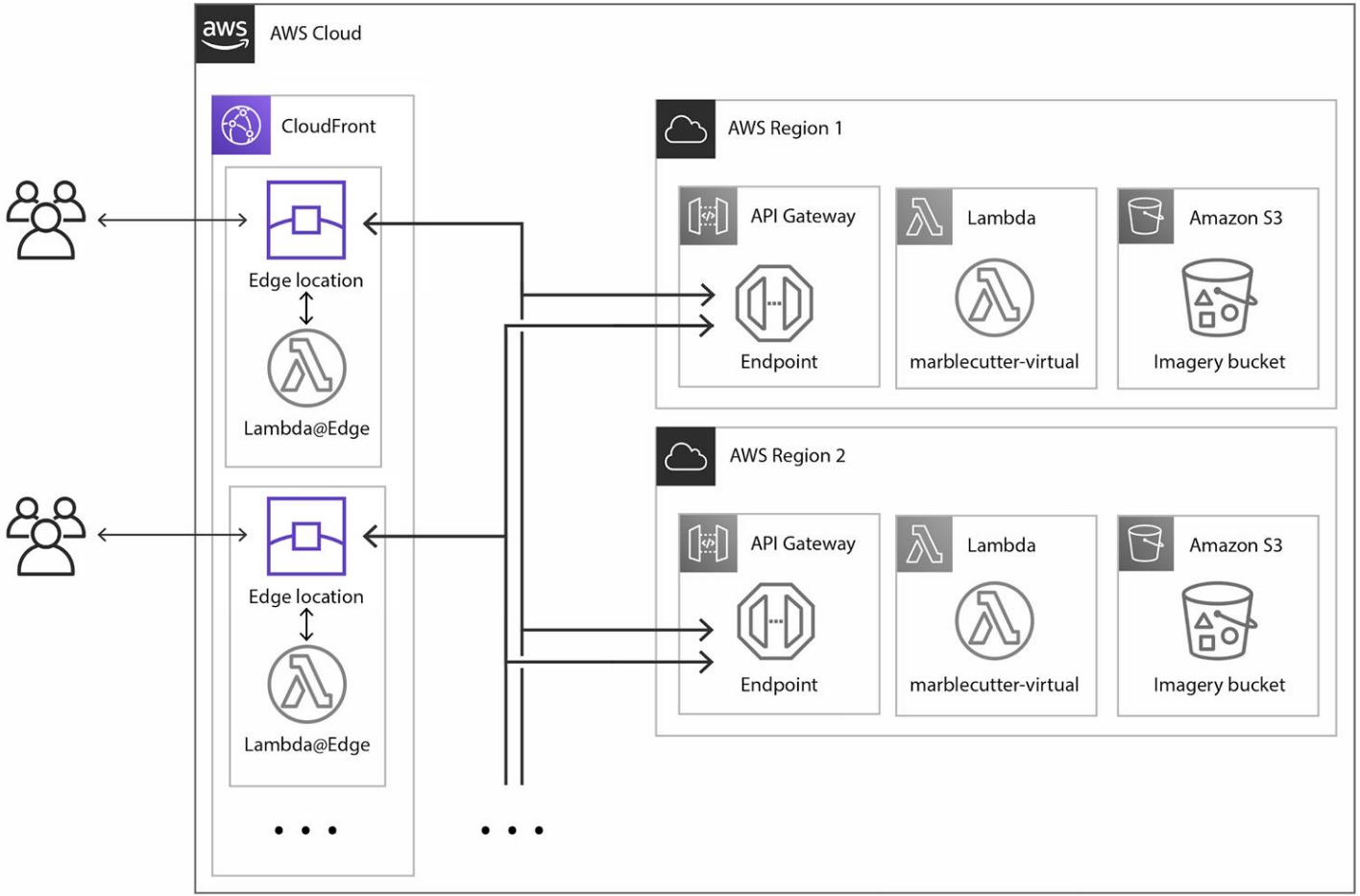
NAIP



Architecture

https://tiles.rdnt.io


AWS Cloud
CloudFront
Edge location
Lambda@Edge
Edge location
Lambda@Edge
AWS Region 1
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket
AWS Region 2
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket

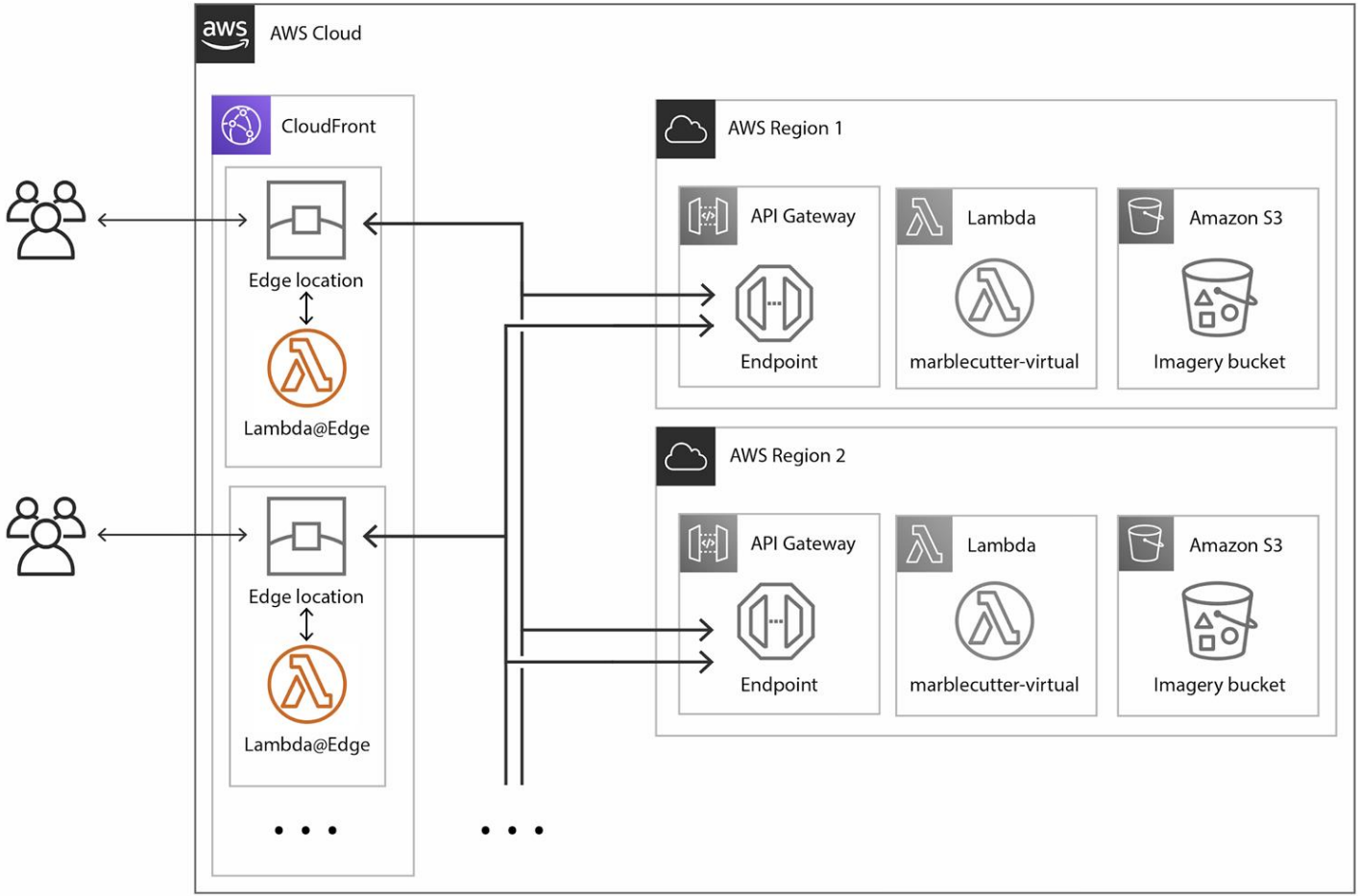

AWS Cloud
CloudFront
Edge location
Lambda@Edge
Edge location
Lambda@Edge
AWS Region 1
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket
AWS Region 2
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket

Anatomy of a request

CF ENDPOINT | ZOOM | TILE X | TILE Y | SCALE | TARGET URL

https://tiles.rdnt.io/tiles/16/10679/23825@2x?url=s3://naip-visualization/or/2016/100cm/rgb/44121/m_4412162_sw_10_1_20160621.tif

=


AWS Cloud
CloudFront
Edge location
Lambda@Edge
Edge location
Lambda@Edge
AWS Region 1
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket
AWS Region 2
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket

Lambda@Edge Function

```
exports.handler = async (event, context, callback) => {
  const request = event.Records[0].cf.request;

  const qs = querystring.parse(request.querystring);

  if (qs.url == null) {
    return callback(null, request);
  }

  const uri = url.parse(qs.url);

  const region = await getRegion(uri);
  let origin = MAPPING[region];

  if (origin == null) {
    region = DEFAULT_REGION;
    origin = MAPPING[region];
  }

  request.origin.custom.domainName = `${origin}.execute-api.${region}.amazonaws.com`;
  request.origin.custom.path = "/Prod";

  request.headers["host"] = [
    { key: "host", value: `${origin}.execute-api.${region}.amazonaws.com` }
  ];

  return callback(null, request);
};
```

Get the request object

Parse the query string

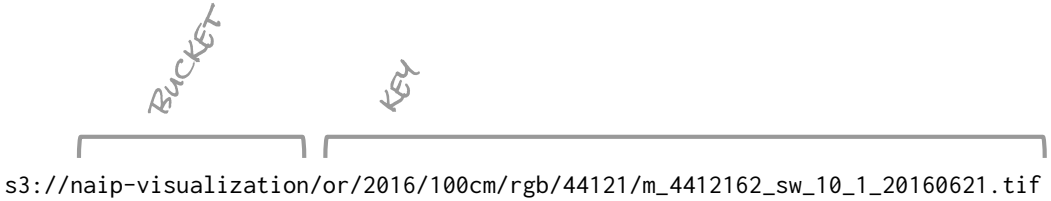
Extract the target URL

Find the target URL's region

Lookup up the API Gateway ID

Rewrite the origin request

1. Extract the bucket



2. Determine the bucket’s region

```
$ host naip-visualization.s3.amazonaws.com

naip-visualization.s3.amazonaws.com is an alias for s3-us-west-2-w.amazonaws.com.
s3-us-west-2-w.amazonaws.com has address 52.218.232.227
```

```
const getRegion = async ({ hostname }) => {
  try {
    if (uri.protocol === "s3:") {
      hostname += ".s3.amazonaws.com";
    }
```

Convert to a hostname + resolve

```
    let matches;
    if (hostname.endsWith(".s3.amazonaws.com")) {
      const addresses = await resolveCname(hostname);

      if (addresses.length > 0) {
        if (
          (matches = addresses[0].match(
            /s3-([\w-]+)-w.amazonaws\.com/
          ))
        ) {
          if (matches[1] === "1") {
            return "us-east-1";
          }
```

Extract the region

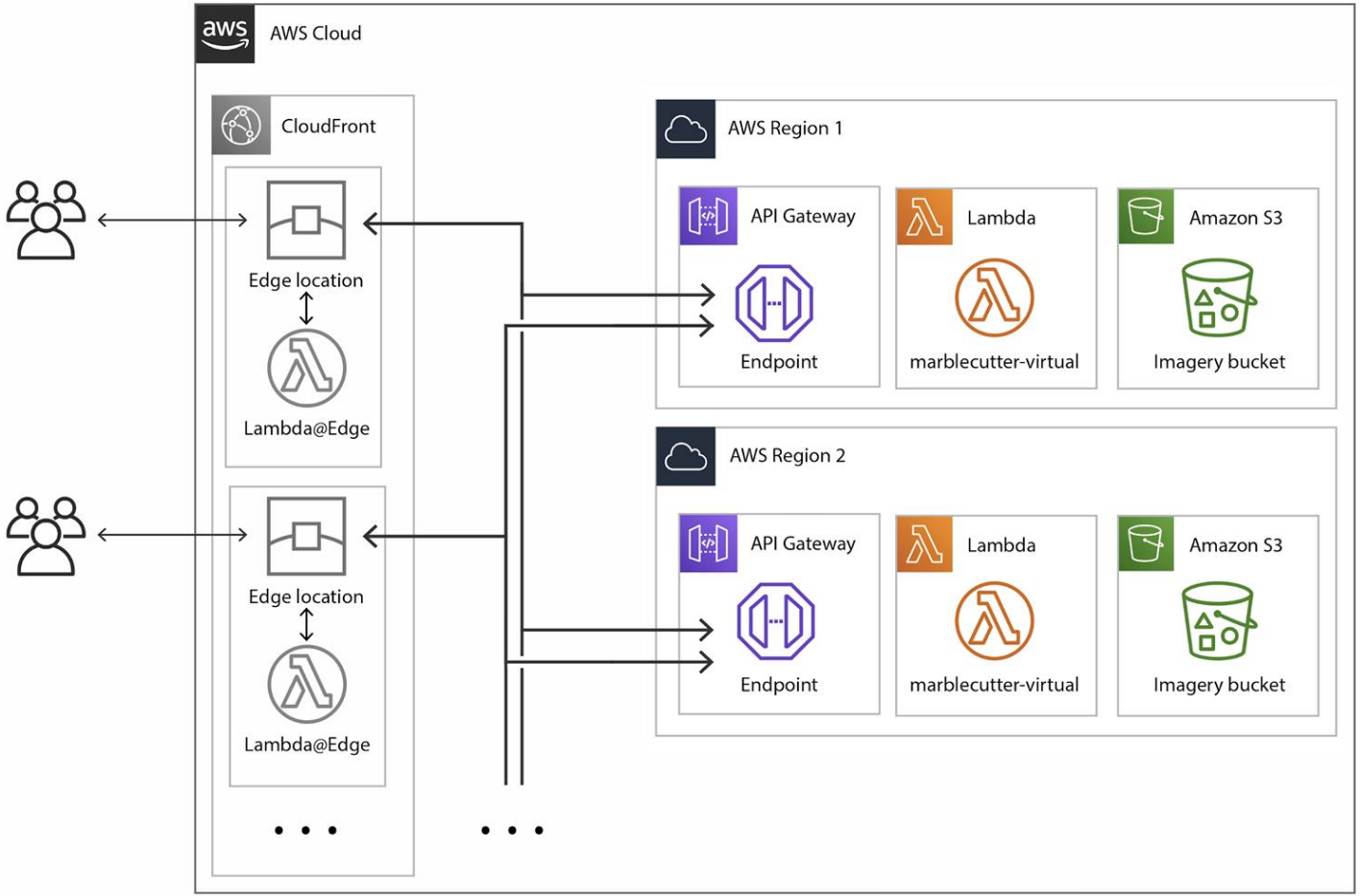
Special cases

```
          if (matches[1] === "3") {
            return "eu-west-1";
          }

          return matches[1];
        } else if (
          (matches = addresses[0].match(
            /s3(-w(ebsite)?)?(\.dualstack)?[.-]([\w-]+)\.amazonaws\.com/
          ))
        ) {
          return matches[4];
        }
      }
    } else if (
      (matches = uri.hostname.match(
        /s3(-w(ebsite)?)?(\.dualstack)?[.-]([\w-]+)\.amazonaws\.com/
      ))
    ) {
      return matches[4];
    }
  } finally {
    return DEFAULT_REGION;
  }
};
```

3. Set the origin to an API Gateway in that region

```
{
  "us-east-1": "gkh62xc7he",
  "us-east-2": "pmc4xepmhj",
  "us-west-1": "9z8wrtc0y2",
  "us-west-2": "p6q1upto2a",
  "ap-south-1": "3ozyebs7ic",
  "ap-northeast-1": "09ydnoc09g",
  "ap-northeast-2": "1oj45tquq3",
  "ap-southeast-1": "18lsnrvvyf",
  "ap-southeast-2": "1xgmgjnwxf",
  "ca-central-1": "813ilrpqtg",
  "eu-central-1": "1nk7vx4ye4",
  "eu-west-1": "b6jx1o8lbi",
  "eu-west-2": "aexc3bj0bi",
  "eu-west-3": "bz2jkis2x3",
  "eu-north-1": "96pywtwas9",
  "sa-east-1": "b3bz87h9n7"
}
```

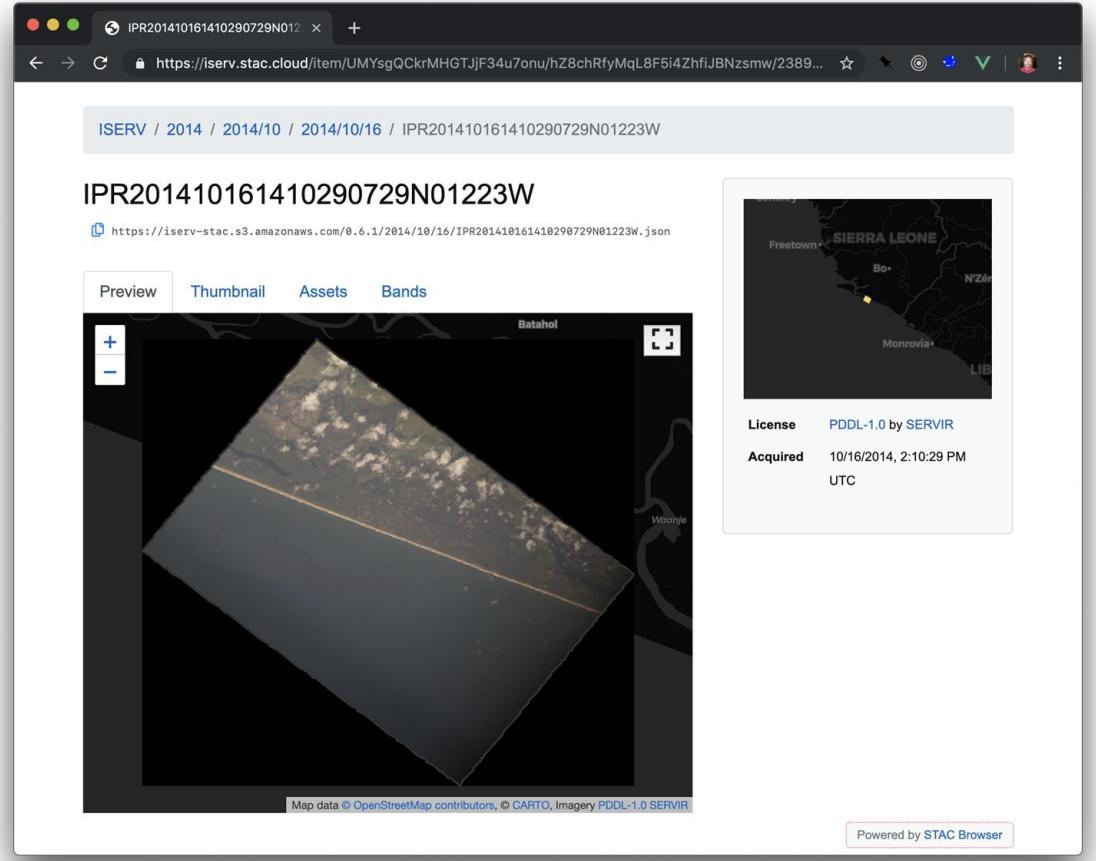

AWS Cloud
CloudFront
Edge location
Lambda@Edge
Edge location
Lambda@Edge
AWS Region 1
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket
AWS Region 2
API Gateway
Endpoint
Lambda
marblecutter-virtual
Amazon S3
Imagery bucket

Co-locate **interactive** compute with data

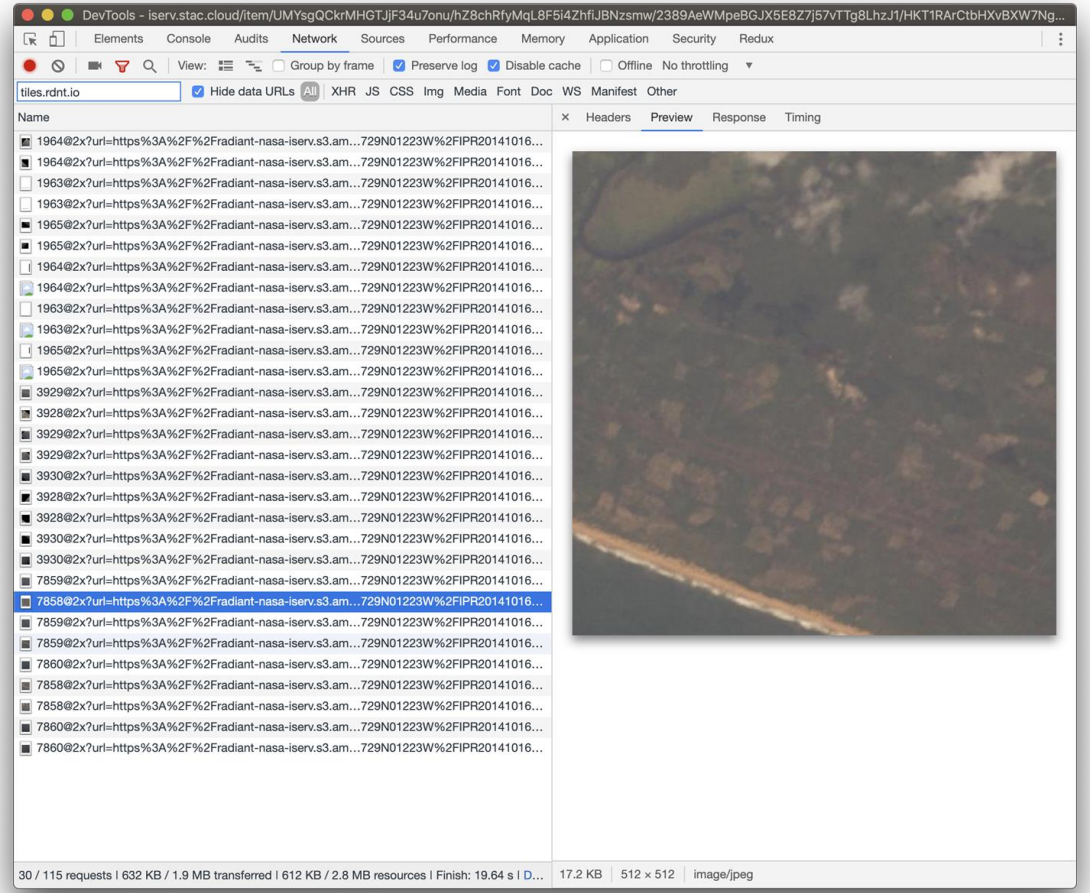
Examples

https://tiles.rdnt.io enables other applications

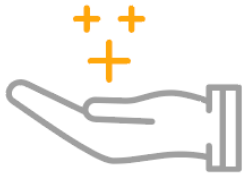
ISERV STAC browser



ISERV STAC browser



Customer benefits



- Region-agnostic
- Lower bandwidth costs
- Lower S3 costs
  - No need for multi-region replication
  - No cross-region S3 reads
- Lower latency

Cloud-Optimized Files / Formats

Principles of Cloud-Optimized Files / Formats

- Predictable filenames / S3 keys
- A few large objects are better than many small objects
- Specify well-known offsets to avoid random reads to derived locations
- HTTP Range Request-friendly
- Related data is contiguous / local
- Block-level compression
- Block-level summaries
- Multi-resolution

Benefits of Cloud-Optimized Formats

- Doesn’t read everything
- Lower costs
- Customers don’t need copies of data
- Reduced data transfer when accessed interactively
- Easier object management
- No need for external indices

Examples of Cloud-Optimized Formats

- Web video
- ORC
- Parquet
- Cloud-Optimized GeoTIFF (COG)
- ZIP

Questions?