Cloud Networking in Practice

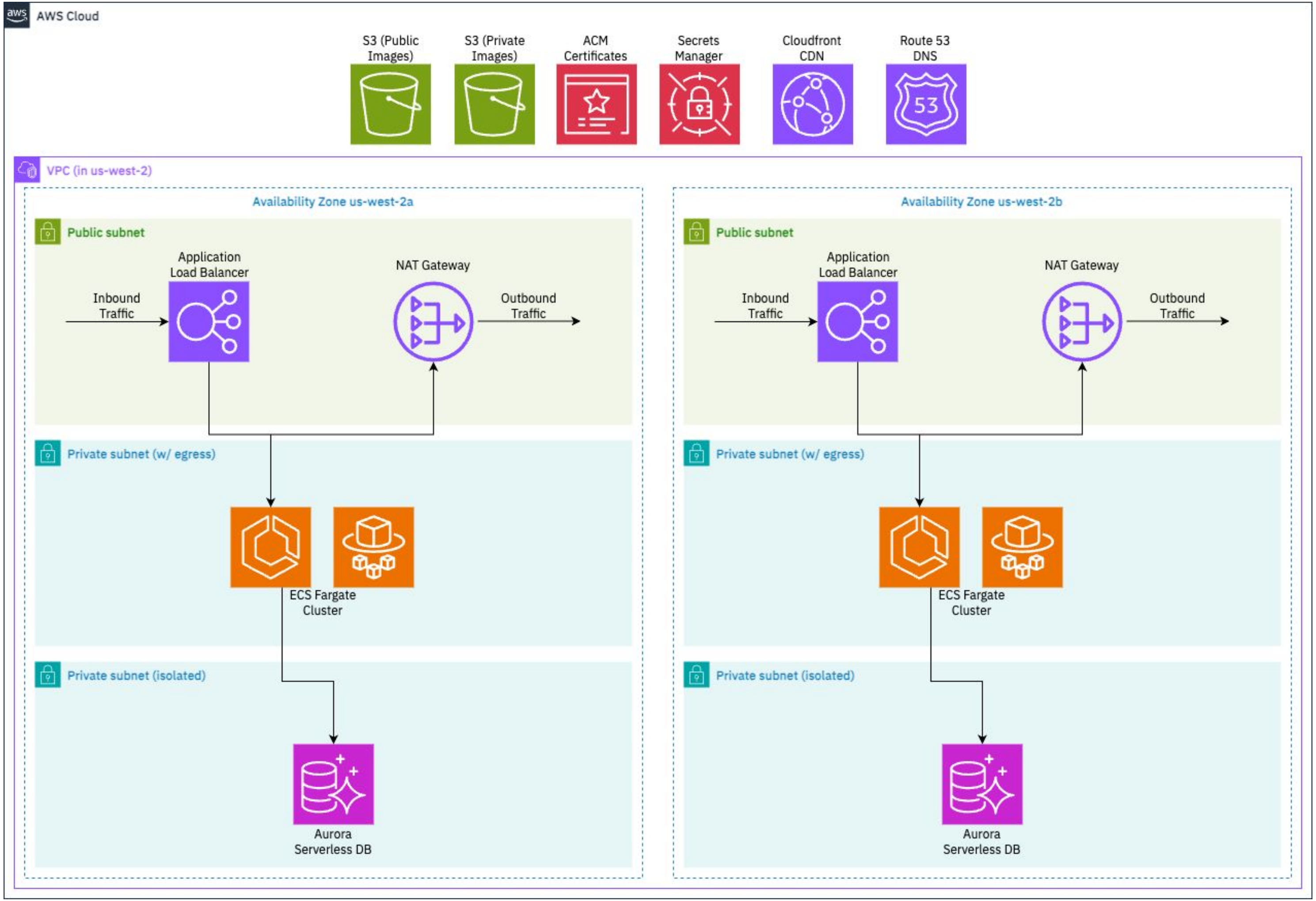
CS 40 | January 17, 2024

Agenda

1. (Very) Brief Networking Review
2. Virtual Private Clouds
3. 5 Minute Break
4. Load Balancing
5. CDNs

Goal: Understand the architecture of modern cloud applications

yoctogram


AWS Cloud
S3 (Public Images)
S3 (Private Images)
ACM Certificates
Secrets Manager
Cloudfront CDN
Route 53 DNS
53
VPC (in us-west-2)
Availability Zone us-west-2a
Public subnet
Application Load Balancer
Inbound Traffic
NAT Gateway
Outbound Traffic
Private subnet (w/ egress)
ECS Fargate Cluster
Private subnet (isolated)
Aurora Serverless DB
Availability Zone us-west-2b
Public subnet
Application Load Balancer
Inbound Traffic
NAT Gateway
Outbound Traffic
Private subnet (w/ egress)
ECS Fargate Cluster
Private subnet (isolated)
Aurora Serverless DB

Networking Review

- **Routing**: Rules on where a network request should be sent

- **Subnet**: A range of IP addresses behind a router

- **NAT**: A way to give machines with only private IP addresses outbound internet access by assigning them the same public IP address

- **DNS**: Used to resolve domain names to IP addresses or other domain names

- **TLS**: Provides connection encryption using X.509 certificates

Virtual Private Clouds

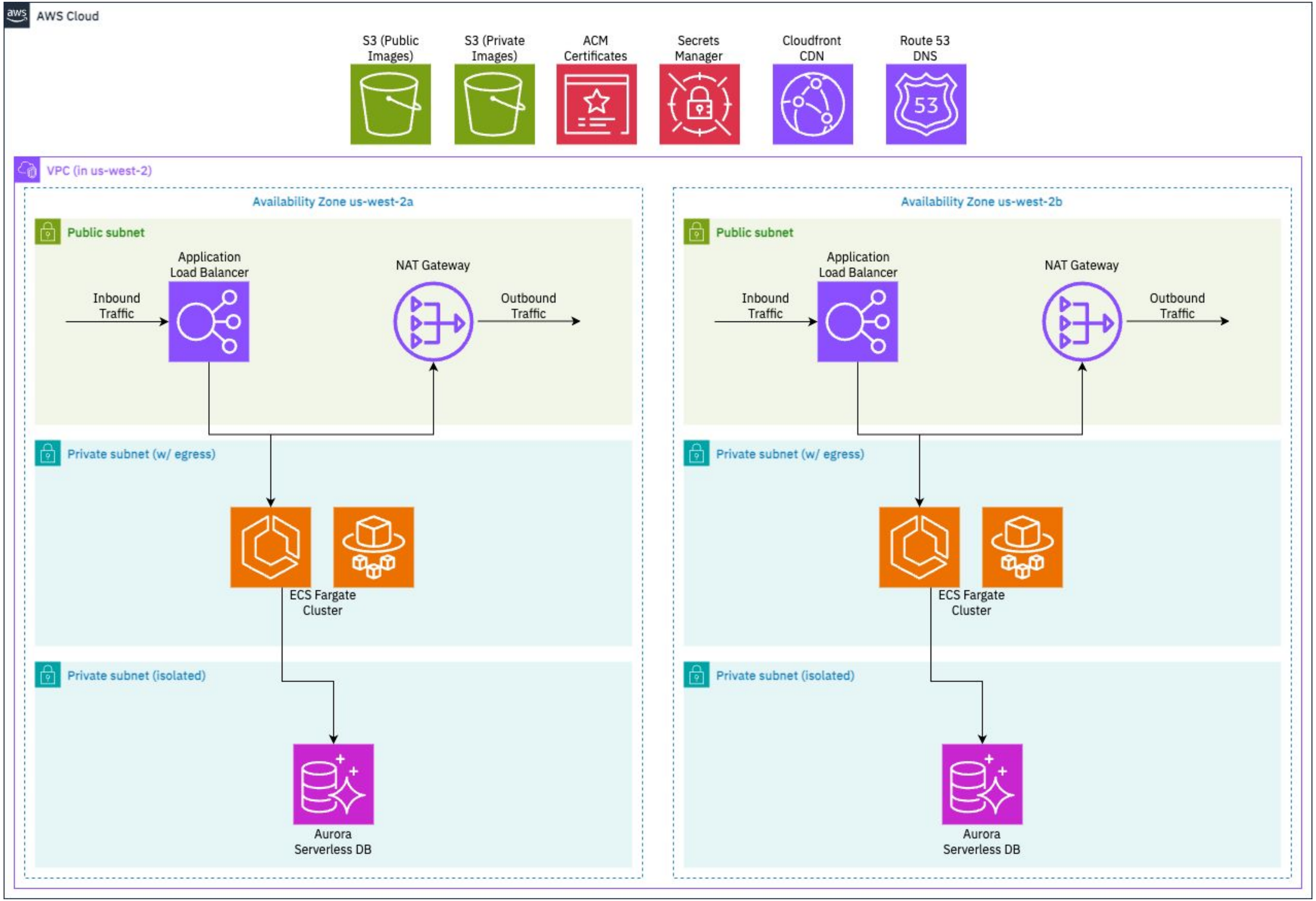
Virtual Private Clouds

- An entire network in the cloud
  - Allows complete control of your backend infrastructure’s networking configuration
- VPCs only span a single region
- Components:
  - An entrypoint (almost always a load balancer)
  - Public/Private subnets
  - Routing table
  - Access Control Lists (ACLs)
  - (optional) VPN entrypoint

Availability Zones

- Each AWS zone (us-west-2, us-east-1, etc) has multiple availability zones
  - Physically separate datacenter for each zone, for fault tolerance
  - Named using letters, ie, us-west-2a, eu-west-1c

- AWS zones contain subnets in the VPC
  - Example: overall VPC range is `192.168.0.0/16` in us-west-2
    - Availability zone 1 in us-west-2a, subnets `192.168.1.0/25` and `192.168.1.128/25`
    - Availability zone 2 in us-west-2b, subnets `192.168.2.0/25` and `192.168.2.128/25`


AWS Cloud
S3 (Public Images)
S3 (Private Images)
ACM Certificates
Secrets Manager
Cloudfront CDN
Route 53 DNS
VPC (in us-west-2)
Availability Zone us-west-2a
Public subnet
Application Load Balancer
NAT Gateway
Inbound Traffic
Outbound Traffic
Private subnet (w/ egress)
ECS Fargate Cluster
Private subnet (isolated)
Aurora Serverless DB
Availability Zone us-west-2b
Public subnet
Application Load Balancer
NAT Gateway
Inbound Traffic
Outbound Traffic
Private subnet (w/ egress)
ECS Fargate Cluster
Private subnet (isolated)
Aurora Serverless DB

VPC Subnets

- Public subnet
  - Entrypoint (load balancer)
  - NAT Gateway (allows egress)

- Private subnets
  - Application backend servers
  - Databases
  - Can optionally have egress via the public subnet’s NAT gateway

Why separate public and private subnets?

Public/Private Subnet Split

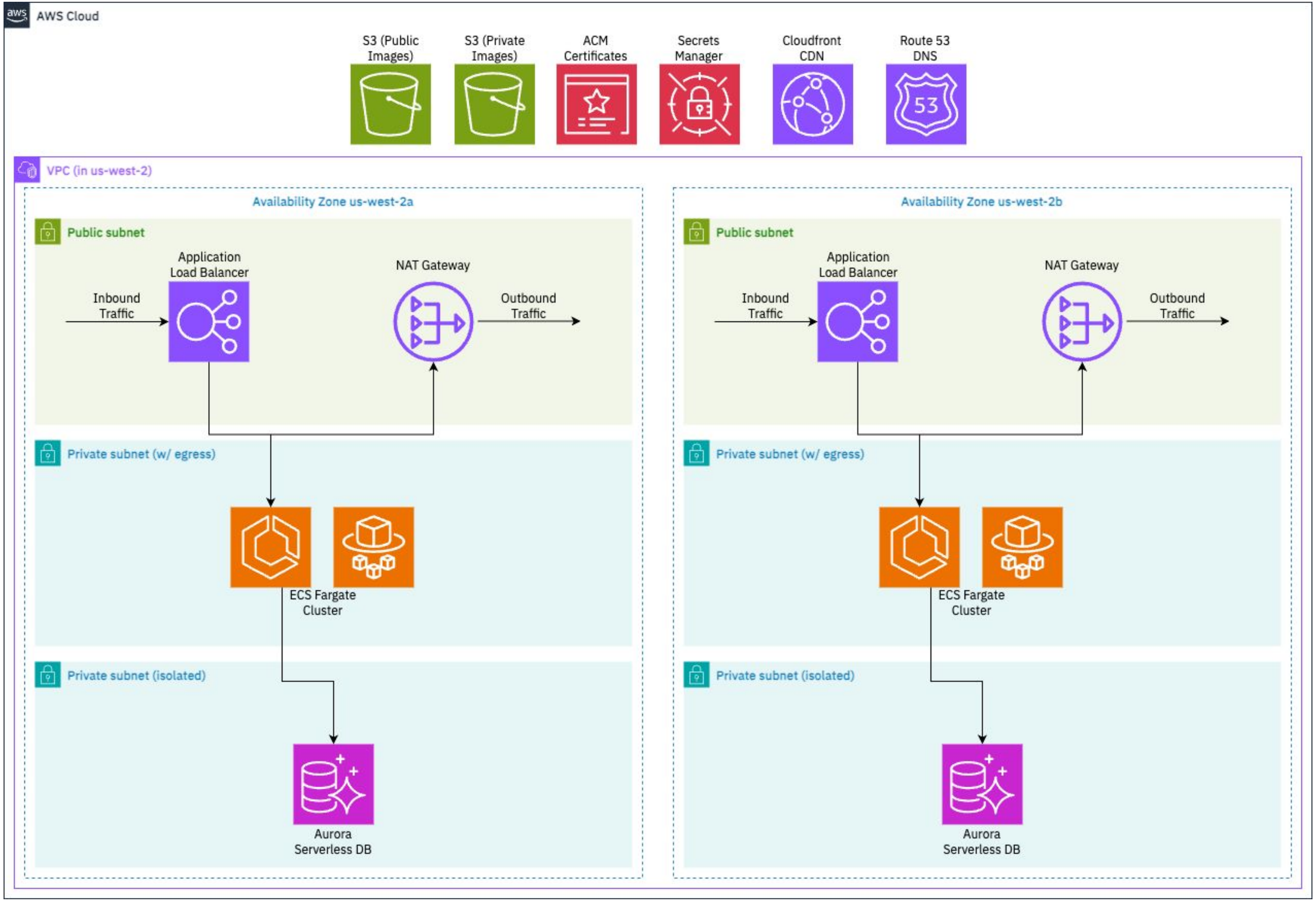
- Save IPv4 addresses
- Control inflows and outflows
- Logical isolation of different components
- Abstraction of internal details

VPC Access Control Lists

- Essentially a network-level firewall
  - Like a security group, but operates at a subnet level

- Regulates traffic inflows and outflows to an entire subnet

- *Stateful*: don't need to worry about raw packets
  - Operates at incoming and outgoing connection level


AWS Cloud
S3 (Public Images)
S3 (Private Images)
ACM Certificates
Secrets Manager
Cloudfront CDN
Route 53 DNS
53
VPC (in us-west-2)
Availability Zone us-west-2a
Public subnet
Application Load Balancer
Inbound Traffic
NAT Gateway
Outbound Traffic
Private subnet (w/ egress)
ECS Fargate Cluster
Private subnet (isolated)
Aurora Serverless DB
Availability Zone us-west-2b
Public subnet
Application Load Balancer
Inbound Traffic
NAT Gateway
Outbound Traffic
Private subnet (w/ egress)
ECS Fargate Cluster
Private subnet (isolated)
Aurora Serverless DB

NAT Gateways

- Applications and operating systems sometimes need to be able to reach external resources (egress)
  - Updates, telemetry, 3rd party APIs, etc
  - Databases should *not* have egress

- Allows private subnets to have egress
  - This is NAT: all internal resources have the same IP when going outbound

- Regulate and monitor outbound flows

- Charged based on the amount of data you send outbound (expensive)

NAT Gateways are Expensive

- Base price (for no data transferred): **$0.045/hr**
  - A single NAT gateway (for a single subnet) costs **$32.40/mo**, for sitting around doing nothing
  - A two-AZ setup has two NAT gateways, so the entry cost is **$64.80/mo**

- Data-dependent additional pricing: **$0.045/GB**
  - Transferring 1 TB of data per month costs **$45**

Takeaway: *Often, NAT gateways can dominate cloud costs – for both small and large organizations! Consider if you really need them (does your application make connections to external services?)*

Ways around NAT Gateway costs

- VPC Endpoints (AWS PrivateLink)
  - Allow outbound access to other AWS services (e.g. S3, CloudWatch, Secrets Manager) without a NAT gateway
  - Helps you save on the data-dependent pricing, if most of the data you need is in other AWS services anyways
  - Pricing: **$0.01/PB**, basically free

- Alternative NAT instances
  - Run a NAT gateway on an EC2 instance using firewall rules, e.g. AlterNAT
  - Pricing: just the EC2 instance cost: as cheap as **~$5/mo** if not much capacity is needed
  - **Note**: bandwidth capped at **5 GB/s**, so only useful for low data volume scenarios

Scott Piper liked

Corey Quinn
@QuinnyPig

Todays fun fact: if it's through an @awscloud Managed NAT Gateway it would cost $714.

Wear a condom, kids.

> **Quite Interesting** @qikipedia · 7/14/12
> A sperm has 37.5 MB of DNA info. One ejaculation transfers 15,875 GB of data, equivalent to that held on 7,500 laptops.

11:30 AM · 1/5/24 from Earth · **2K** Views

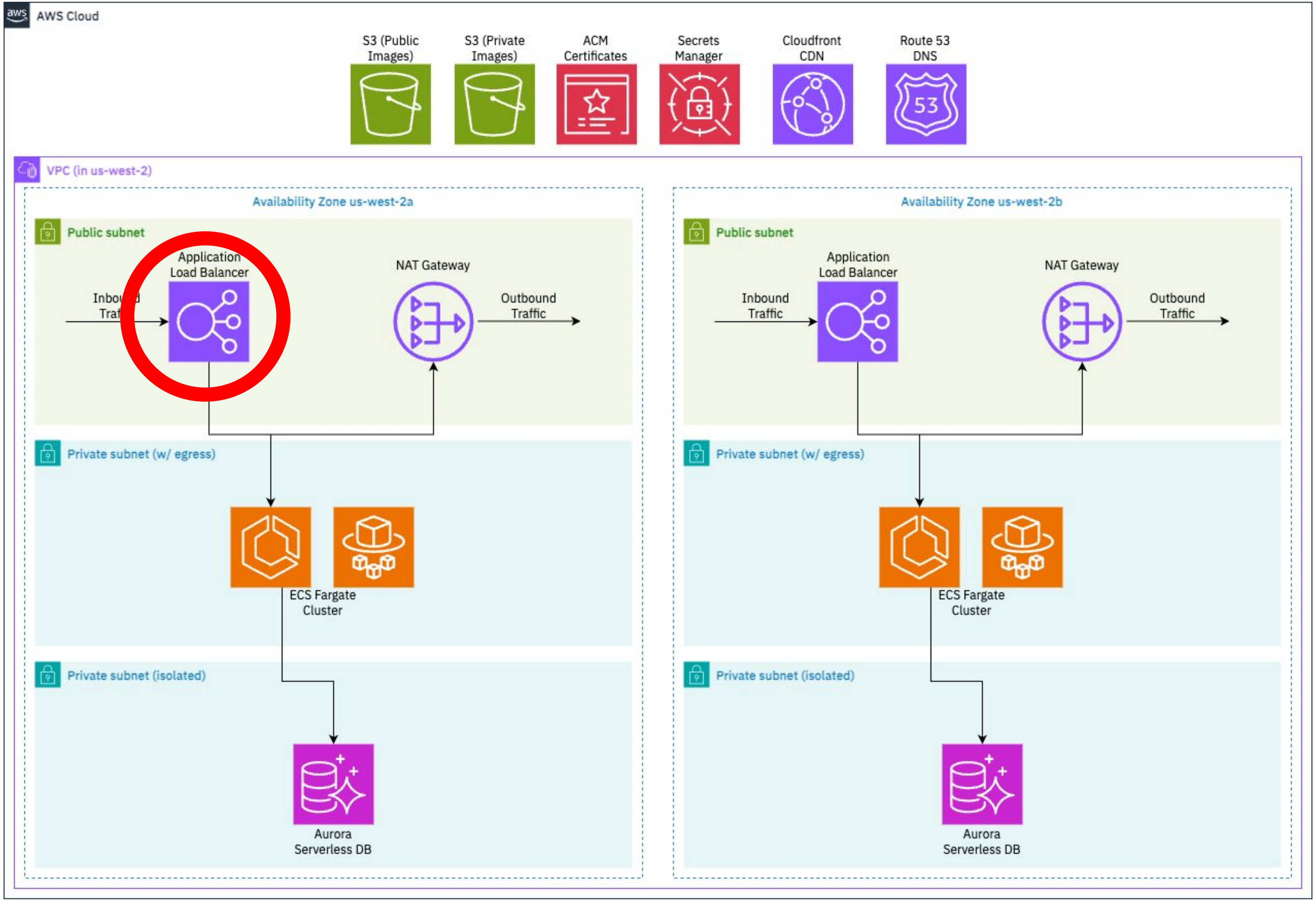
9 Reposts **1** Quote **37** Likes **2** Bookmarks

VPC Tour

KISS: Keep It Simple Stupid

5 Minute Break

Load Balancing


AWS Cloud
S3 (Public Images)
S3 (Private Images)
ACM Certificates
Secrets Manager
Cloudfront CDN
Route 53 DNS
VPC (in us-west-2)
Availability Zone us-west-2a
Public subnet
Application Load Balancer
NAT Gateway
Outbound Traffic
Private subnet (w/ egress)
ECS Fargate Cluster
Private subnet (isolated)
Aurora Serverless DB
Availability Zone us-west-2b
Public subnet
Application Load Balancer
NAT Gateway
Inbound Traffic
Outbound Traffic
Private subnet (w/ egress)
ECS Fargate Cluster
Private subnet (isolated)
Aurora Serverless DB

Load Balancing

- Idea: given multiple backend servers and need to make sure the load is distributed evenly across them

- All inbound requests hit the load balancer first
  - This allows you to keep the actual backend server in a private subnet
  - Load balancer forwards requests to the correct backend server (chosen based on some algorithm)
  - Allows early termination of malicious requests (for DDoS protection and/or WAF – web application firewall)

- **AWS solution**: *Elastic Load Balancer (ELB)*

Static Load Balancing Algorithms

Static load balancing algorithms don't consider the runtime state of the server when assigning servers requests.

Types of algorithms:

- **Round robin**: number each server, increment number every time a request is received
  - Assumes every server and request is the same
- **Weighted round robin**: weight every server based on capacity
  - Assumes every request is the same
- **Hashing**: hash every request, then assign to server based on hash
  - Hash: pseudorandom number based on input
  - Effectively, randomly assign requests to backend server

Dynamic Load Balancing

Dynamic load balancing algorithms take into consideration runtime information about the servers

Types of algorithms:

- **Health check**: have an API endpoint the load balancer can query to determine server health
  - Don’t assign servers requests if they’re unhealthy
- **Least connection**: send the connection to the machine with the least number of active connections
  - **Weighted least connection**: weight servers based on capacity
  - Assumes every request is the same
- **Resource-based**: send requests to servers with the least (current) CPU/RAM usage

Types of Load Balancing (AWS)

- **Application**: distribute traffic based on HTTP-level metadata
  - Metadata e.g. request type, headers, cookies, etc.
  - ALB typically has a TLS certificate attached; terminates TLS connection and passes unencrypted HTTP internally to backend servers

- **Network**: distribute traffic based on transport-level network metadata
  - Metadata e.g. IP addresses, ports
  - TLS session persisted through (fully encrypted) to backend servers, but now backend servers need to maintain some certificate infrastructure

- **Takeaway**: Both cost about the same, NLB is somewhat faster, but requires more management overhead

Tour of Elastic Load Balancer

CDN: Content Delivery Network

Content Delivery Networks

- **Idea**: optimize performance of data transfer by pre-caching content close to users

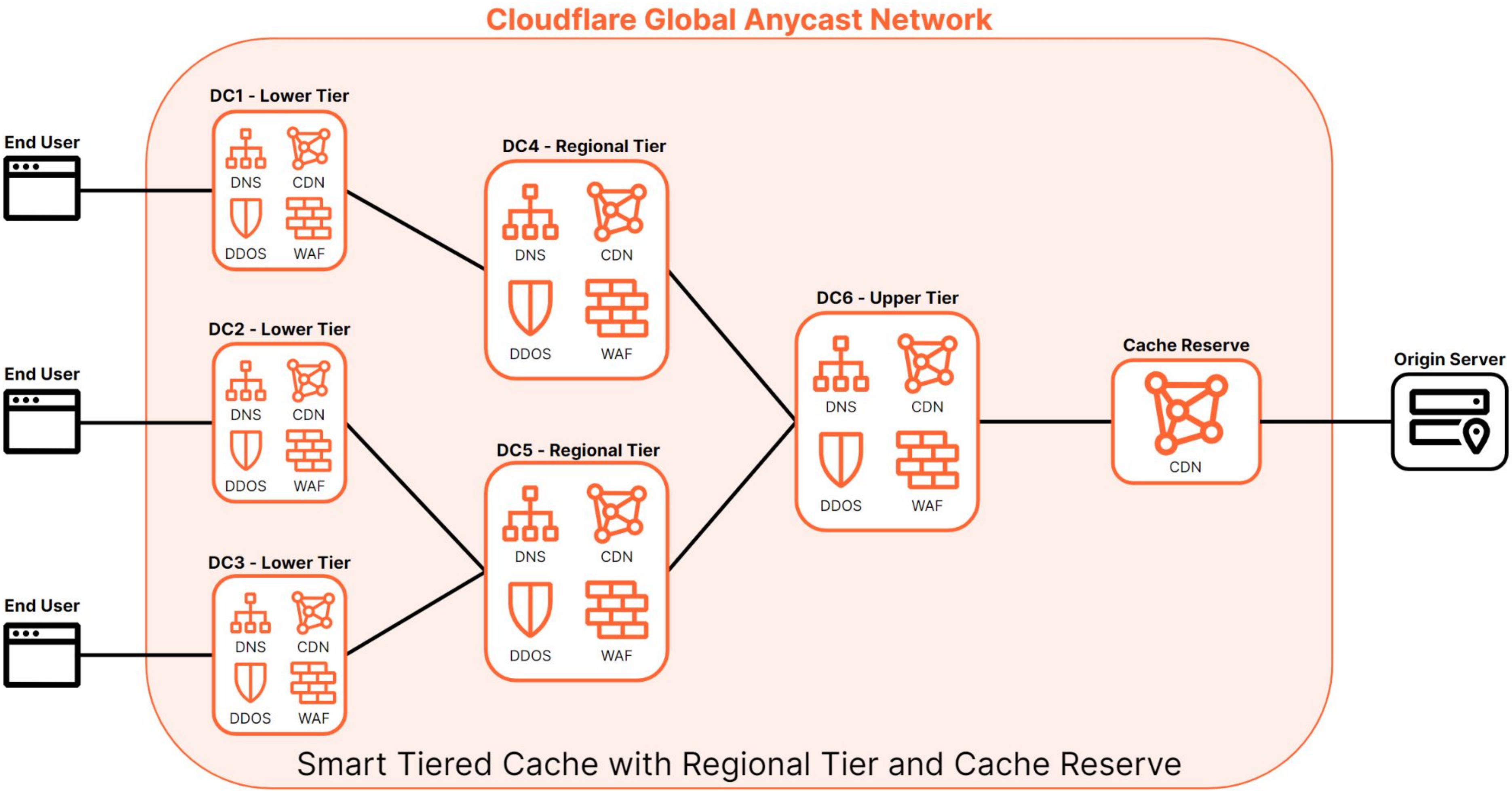
- Providers:
  - AWS Solution: Cloudfront and Lambda Edge
  - Cloudflare
  - Akamai (first CDN – 1998)
  - Fastly
  - etc.

CDN Architecture

- **Idea**: decompose hosting architecture into multiple smaller servers with a single point of truth

- **Origin server**: original web server, point of truth for all edge servers

- **Edge servers** (Points of Presence, PoPs): many smaller, distributed web servers that connect to origin server
    - Anycast routing: many machines share the same IP address, route request to the closest machine
    - Can be flat (one layer of edge servers) or have a hierarchy of edge servers with progressive caching


Cloudflare Global Anycast Network
End User
End User
End User
DC1 - Lower Tier
DNS
CDN
DDOS
WAF
DC2 - Lower Tier
DNS
CDN
DDOS
WAF
DC3 - Lower Tier
DNS
CDN
DDOS
WAF
DC4 - Regional Tier
DNS
CDN
DDOS
WAF
DC5 - Regional Tier
DNS
CDN
DDOS
WAF
DC6 - Upper Tier
DNS
CDN
DDOS
WAF
Cache Reserve
CDN
Origin Server
Smart Tiered Cache with Regional Tier and Cache Reserve

CDN Optimizations

- **Caching**
  - Store responses to API requests on edge servers
  - Store static assets (frontend HTML/CSS/JavaScript, media files – images and videos)
  - No need to recompute previous requests, retrieve saved assets, etc

- **Proximity**
  - Store responses closer to the geographical recipient, lower latency on connections

- **Reliability**
  - Can lose multiple edge servers without issue, requests routed to available servers

CDN Challenges

- **Cache coherency**
  - Responses to API requests may change over time
  - Individual edge servers may have outdated values

- **TLS termination**
  - TLS is usually handled on the edge servers
  - If a request needs to be forwarded to the origin server, it needs to be re-encrypted

- In general, CDNs can add complexity
  - Managed services mitigate this somewhat

What actually happens when you visit a website

1. DNS resolution, hostname to the IP of a load balancer in a close availability zone
2. Load balancer forwards request to closest CDN edge server
3. Edge server decrypts your request (TLS termination)
4. Edge server checks cache for your request, sends response if found and still valid
5. Otherwise, edge server forwards request to origin server (or another edge server in a hierarchy) over a new encrypted connection
6. Origin server receives request, sends response to edge server which sends to client

Next Lecture: Cloud Storage (1/22)