Leveraging Advanced Networking and Load Balancing Services on the GCP

CACHING HTTP(S) LOAD BALANCED CONTENT USING CLOUD CDN

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Content Delivery Network to cache load balanced content close to users

Understand caching and invalidation on CDNs

Activating, de-activating and using Cloud CDN with backend buckets

Setting and modifying cache keys

Prerequisites and Course Outline

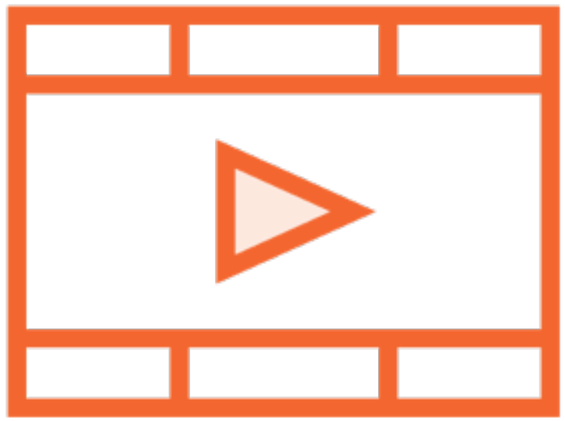
Prerequisites

Basic understanding of cloud computing

Basic understanding of working with Kubernetes clusters

Basic understanding of load balancing

Prerequisites: Basic Cloud Computing



Leveraging Load Balancing Options on the GCP

Deploying Containerized Workloads Using Google Cloud Kubernetes Engine

Course Outline

Caching content using Cloud CDN

- Caching content stored in backend buckets
- Caching and invalidation techniques

Cloud DNS for low-latency Serving

- Global DNS network running on Google infrastructure
- Managing and migrating zones using the web console and command line

Container-native load balancing on Kubernetes

- Configure load balancer to target pods rather than nodes

Scenarios: SpikeySales.com

Hypothetical Online Retailer

- Flash sales of trending products
- Spikes in user traffic

SpikeySales on the GCP

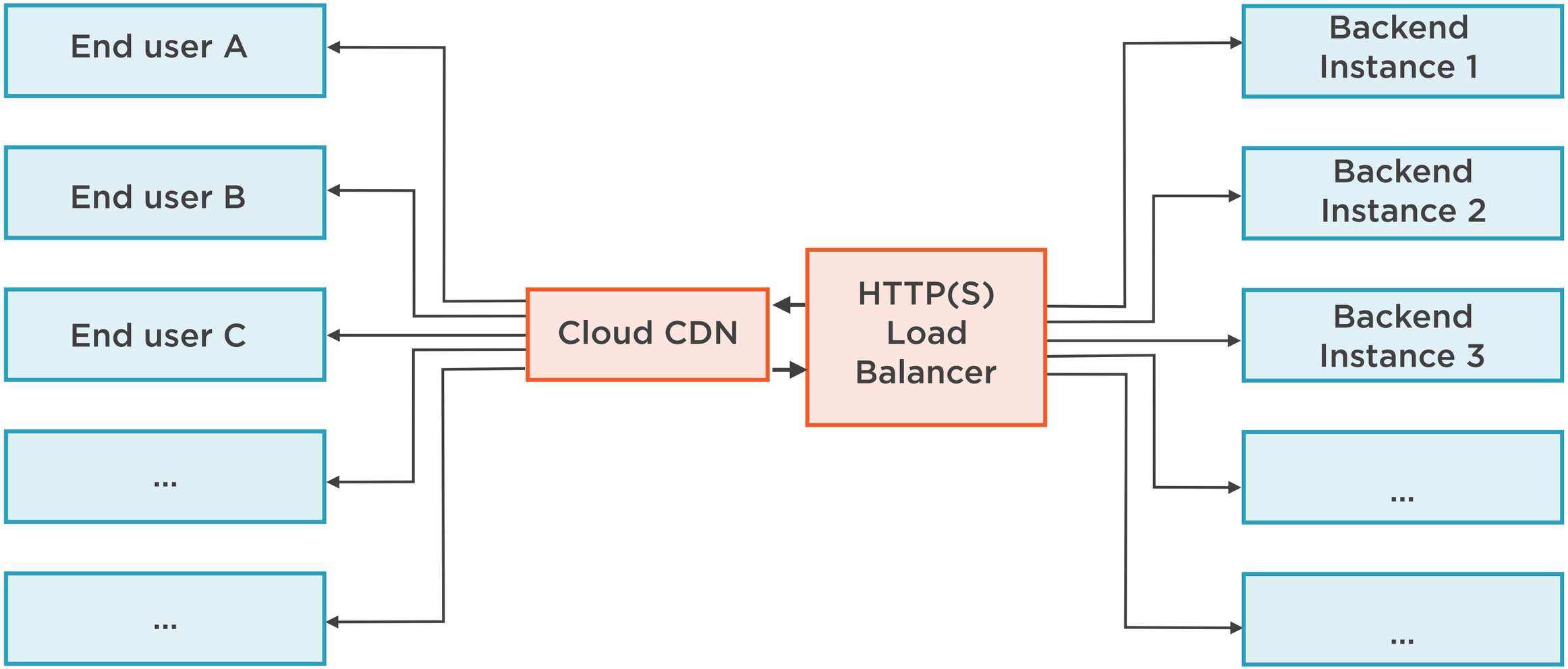
- Cloud computing fits perfectly
- Pay-as-you-go
- No idle capacity during off-sale periods

Cloud CDN and HTTP(S) Load Balancing

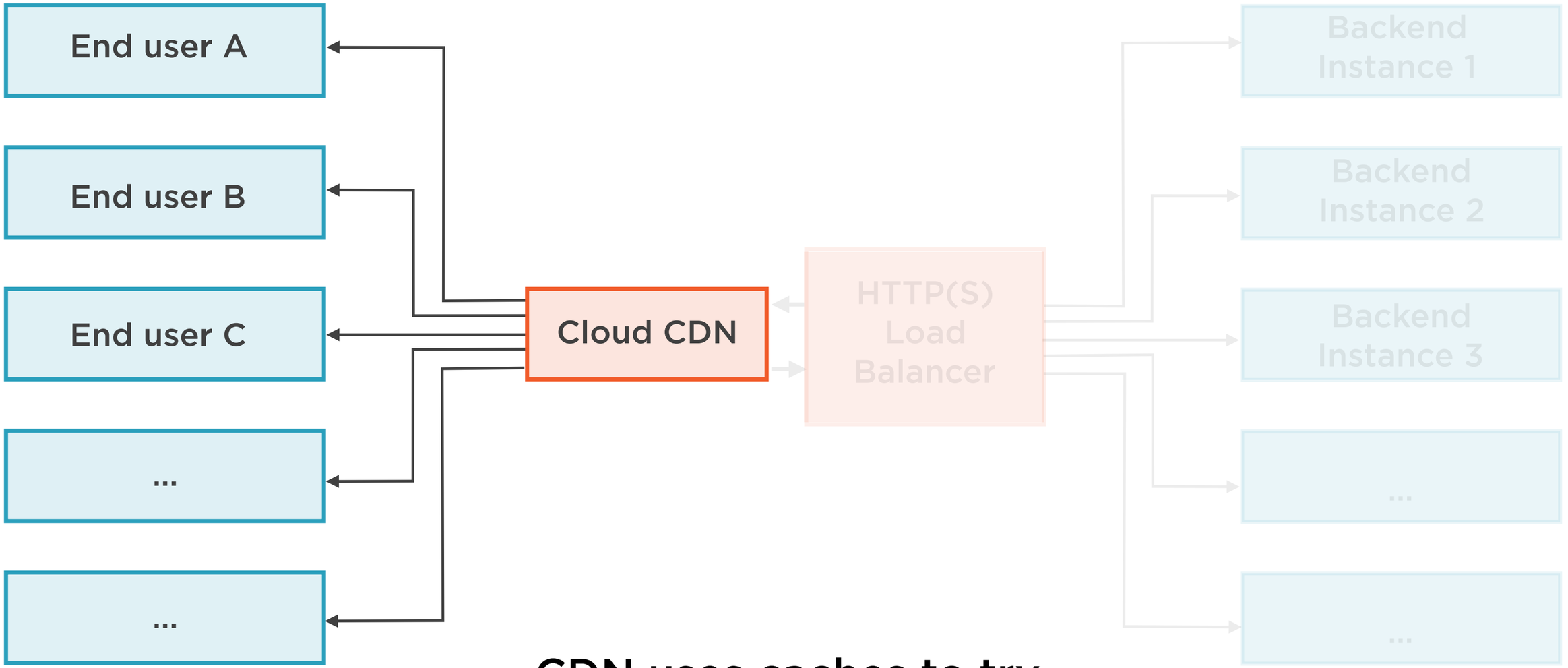
Cloud CDN

Works with HTTP(S) load balancing to deliver content to users from numerous worldwide caches located close to users and at the edge of Google's network

Cloud CDN

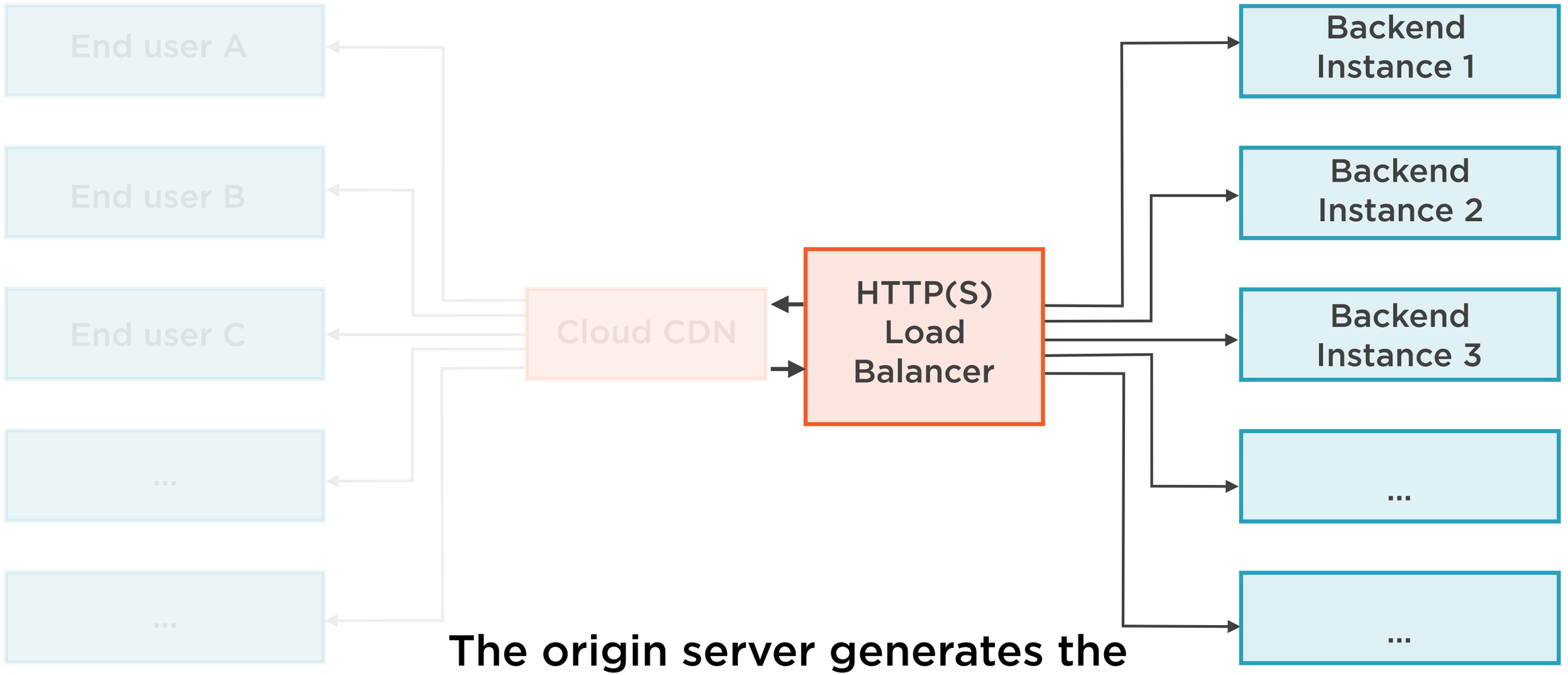

End user A
End user B
End user C
...
...
Cloud CDN
HTTP(S)
Load
Balancer
Backend
Instance 1
Backend
Instance 2
Backend
Instance 3
...
...

Requests to Cloud CDN Close to Users



CDN uses caches to try and fill request

Backend is the Origin Server



The origin server generates the response which fills caches

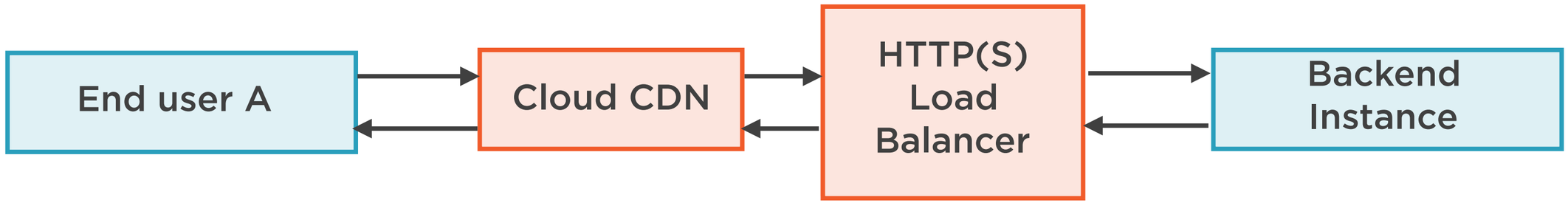
CDN

Load balancing configuration specifies frontend IP address and port

CDN can cache content from two types of backend services

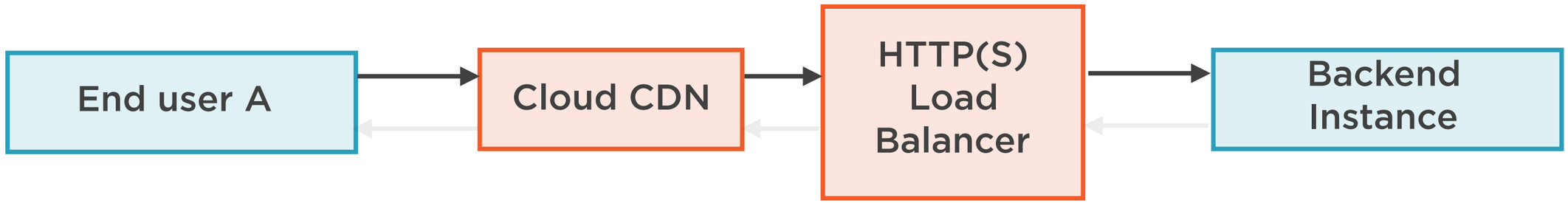
- VM instance groups
- GCS buckets

User Makes Request to Cache



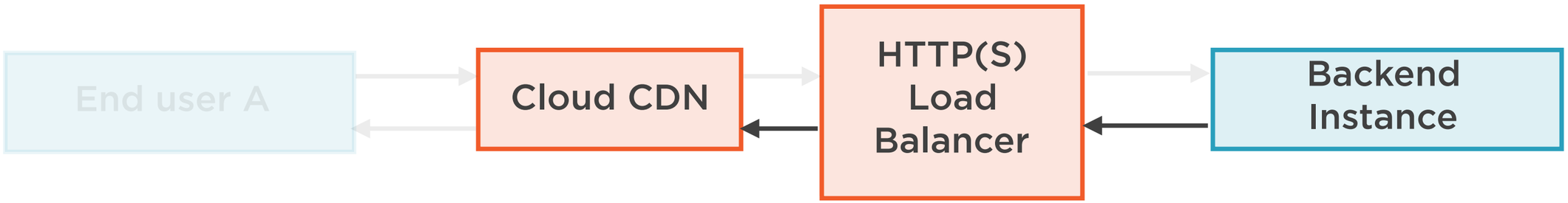
First time content is requested, cache can not fulfill the request

Initial Cache Miss



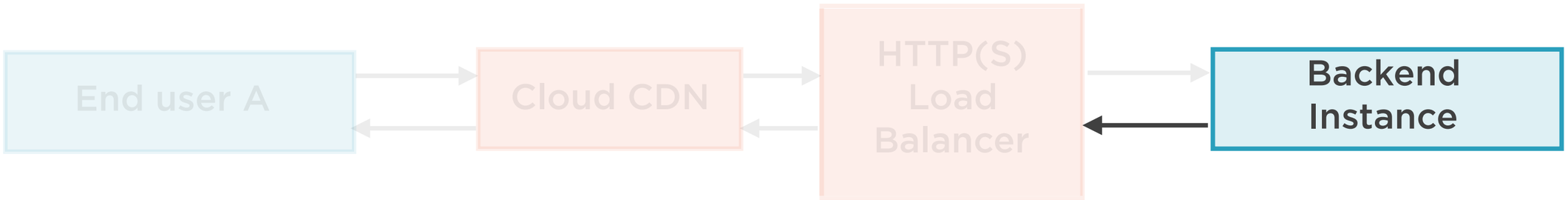
First time content is requested, cache can not fulfill the request

Initial Cache Miss



Data transfer from backend to cache is known as cache fill

Initial Cache Miss



The backend originating the response is called the Origin Server

Cache Miss

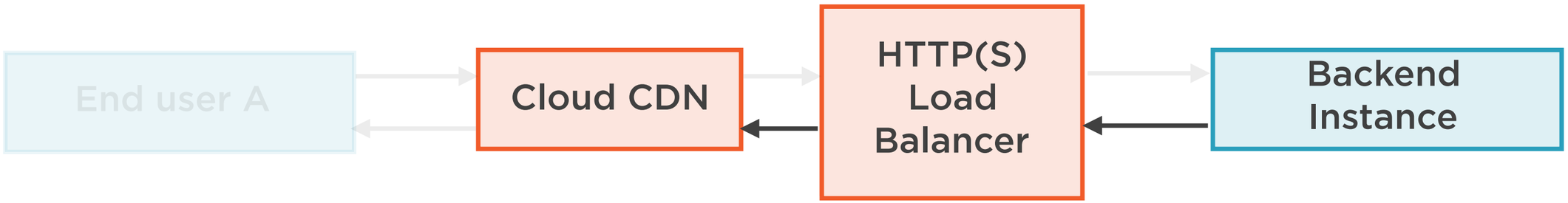
First time content is requested, cache miss occurs

Cache might ask nearby cache

If not present, request goes back to load balancer

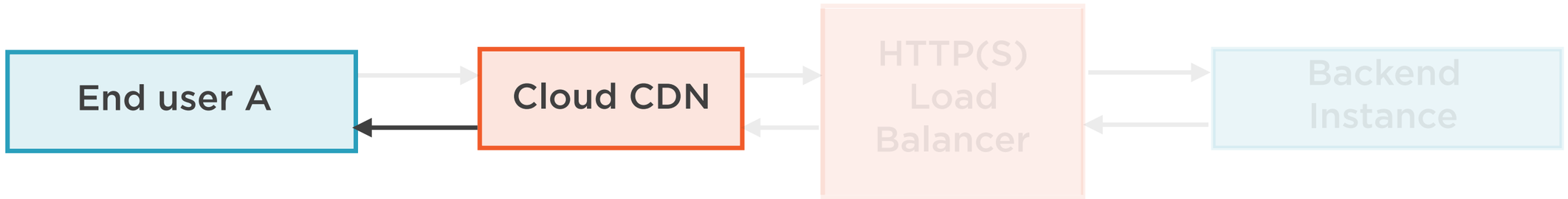
Load balancer in turn sends request to origin server

Initial Cache Miss



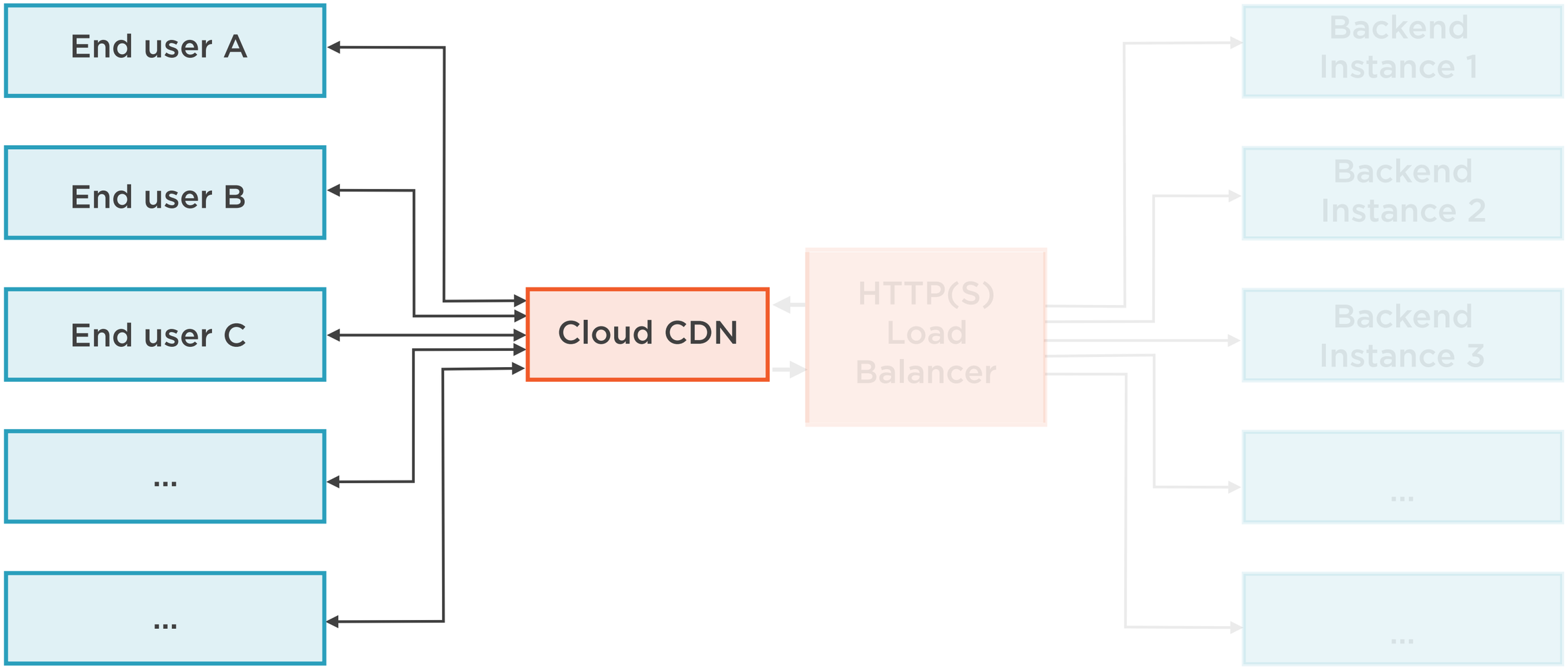
Cache fill could originate in backend, or in nearby cache

Initial Cache Miss



Data transfer from cache to client is known as cache egress

Subsequent Cache Hits



Cache Hits

Cache organizes content by cache key

Subsequent requests looked up by cache key

If found, returned immediately

Cache Hits

For cache hits, still incur egress bandwidth

For cache misses, also incur cache fill bandwidth

CDN Features and Pricing

Cacheable Content

When response is received, CDN checks whether content is cacheable

Some responses never cached

Other responses cached only if certain conditions are met

Cacheable Content

Might decline to cache if more popular content needs to be evicted

Might also decline to cache large content on first request

Inserting Content into Cache

Always reactive - no way to store object in cache other than requesting it

Cache fill only happens in response to client request

Object stored in one cache does not replicate to another cache

Serving Content from Cache

If CDN enabled, caching is automatic for all cacheable content

Origin server uses HTTP headers to indicate which content should be cached

Serving Content from Cache

Impossible to predict which content will be served from cache

Can tell whether content was from CDN by examining logs

Content only will be served from cache if not evicted and not expired

Eviction and Expiration

Eviction occurs when content is being removed to make way for new content

Expiration occurs when expiry timeout has been configured, and has elapsed

Cache Keys

Each CDN entry is identified by cache key

When request comes in, CDN converts URI to cache key and checks cache

Cache Keys

By default, complete URI used as cache key

Can customize cache key by omitting protocol, host or query string

Can configure whitelists and blacklists

Cache Invalidation

Cached object usually stays in cache until evicted or expired

Cache invalidation forces removal of objects from cache

Cache Invalidation

Intended for exceptional cases

Large-scale invalidation can cause traffic spike to your application

Cache Invalidation

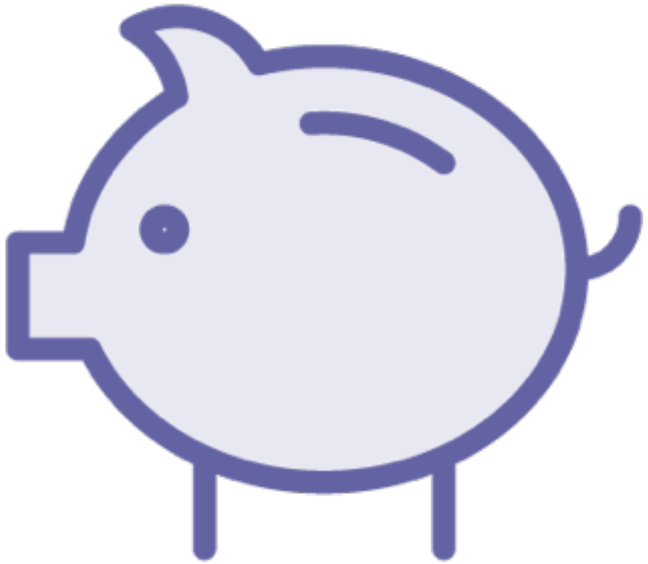
Do not invalidate if backend is serving wrong response

- Wrong response will be re-cached

Will not affect

- Browser cached copies
- Third-party ISP cache copies

Pricing



Cache egress: $0.02 - $0.20 per GB

Cache fill: $0.04 - $0.15 per GB

HTTP(S) cache lookup requests: $0.0075 per 10,000 requests

Cache invalidation: $0.005 per invalidation

https://cloud.google.com/cdn/pricing

Demo

Enabling CDN for caching content served from backend buckets

Summary

Content Delivery Network to cache load balanced content close to users

Understand caching and invalidation on CDNs

Activating, de-activating and using Cloud CDN with backend buckets

Setting and modifying cache keys