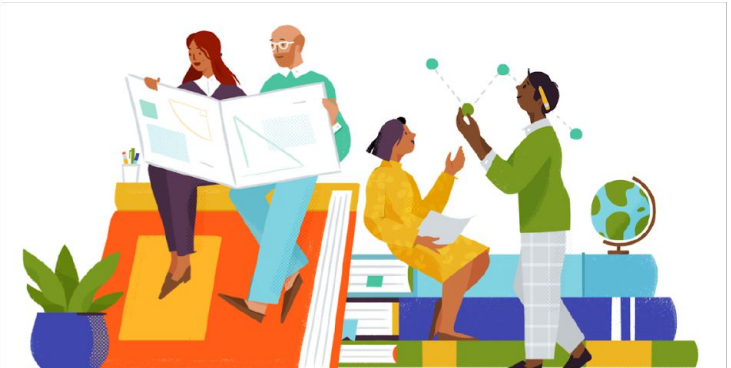
November 6, 2018

Scaling Slack

The Good, The Unexpected, and The Road Ahead

Michael Demmer

mdemmer@slack-corp.com | @mjdemmer



Me

Julia Grace

It's an exciting day for the **Infrastructure Engineering team** I'd like to welcome **Michael Demmer** (@demmer) to Slack He has 2 opinionated kids , plays hockey and used to live in NYC Michael joins us from Jut where he was the Chief Technology Officer and Vice President Engineering. He has held many other engineering leadership roles and did his PhD work in networking at Berkeley Cal. Welcome Michael!

Posted in #yay | Oct 17th, 2016 | View message

(Not) This Talk

1. 2016: ~~Monolith~~
2. 2016-2018: ~~Microservices~~
3. 2016-2018: ~~Best Practices~~
4. 2018: ~~Lessons Learned~~

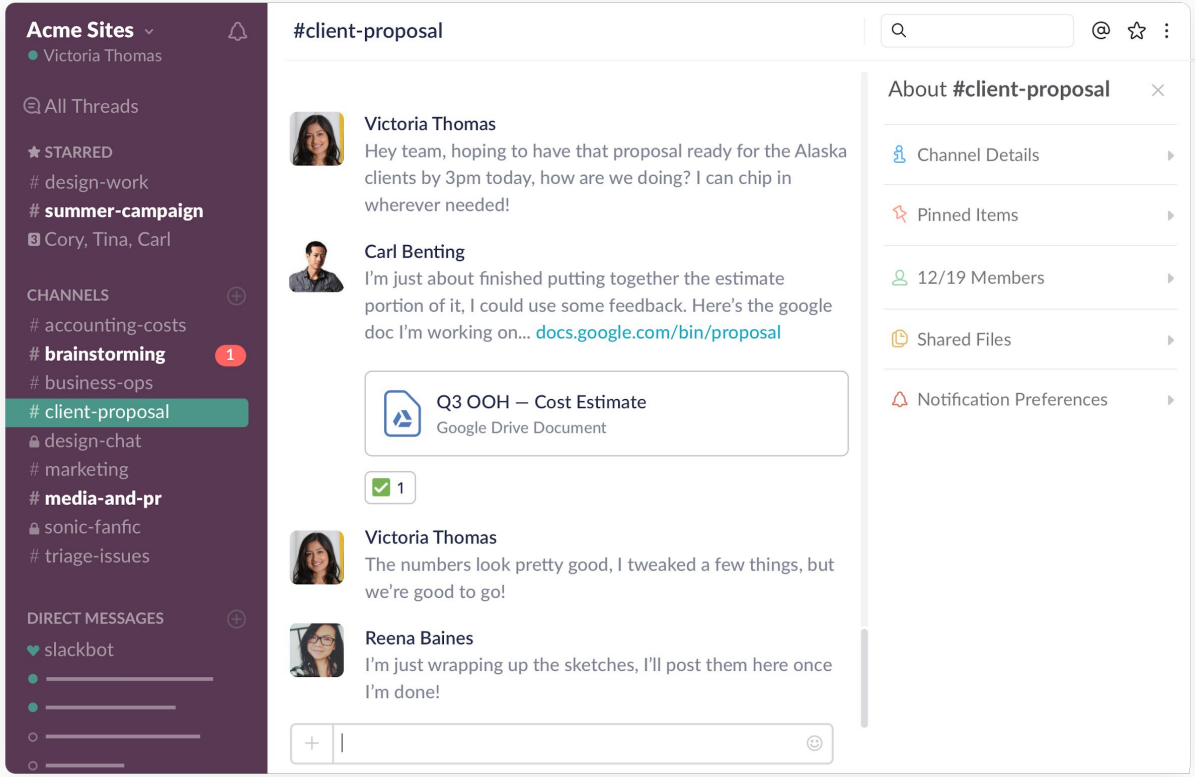
This Talk

1. 2016: How Slack Worked
2. 2016-2018: Things Got More Interesting
3. 2016-2018: What We Did About It
4. 2018+: Themes and Road Ahead

slack

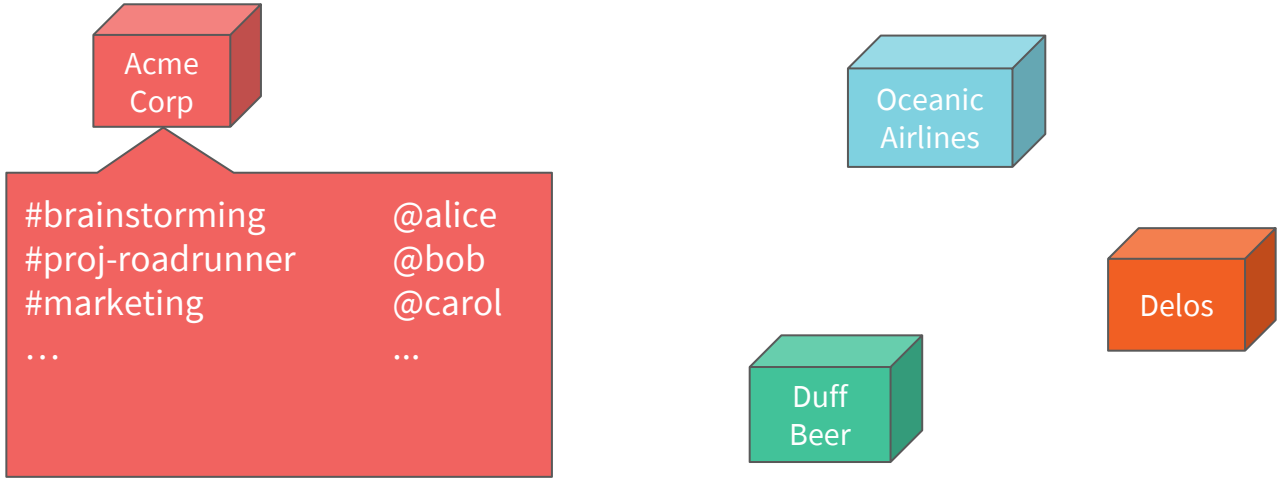
Slack in 2016

Slack


Acme Sites
Victoria Thomas
All Threads
STARRED
design-work
summer-campaign
Cory, Tina, Carl
CHANNELS
accounting-costs
brainstorming 1
business-ops
client-proposal
design-chat
marketing
media-and-pr
sonic-fanfic
triage-issues
DIRECT MESSAGES
slackbot
#client-proposal
Victoria Thomas
Hey team, hoping to have that proposal ready for the Alaska clients by 3pm today, how are we doing? I can chip in wherever needed!
Carl Benting
I'm just about finished putting together the estimate portion of it, I could use some feedback. Here's the google doc I'm working on... docs.google.com/bin/proposal
Q3 OOH — Cost Estimate
Google Drive Document
1
Victoria Thomas
The numbers look pretty good, I tweaked a few things, but we're good to go!
Reena Baines
I'm just wrapping up the sketches, I'll post them here once I'm done!
About #client-proposal
Channel Details
Pinned Items
12/19 Members
Shared Files
Notification Preferences

Workspaces, Channels, Users, and more

A **workspace** logically contains all **channels** and **messages**, as well as **users**, **emoji**, **bots**, and more. All interactions occur within the workspace boundary.



Slack Facts (2016)

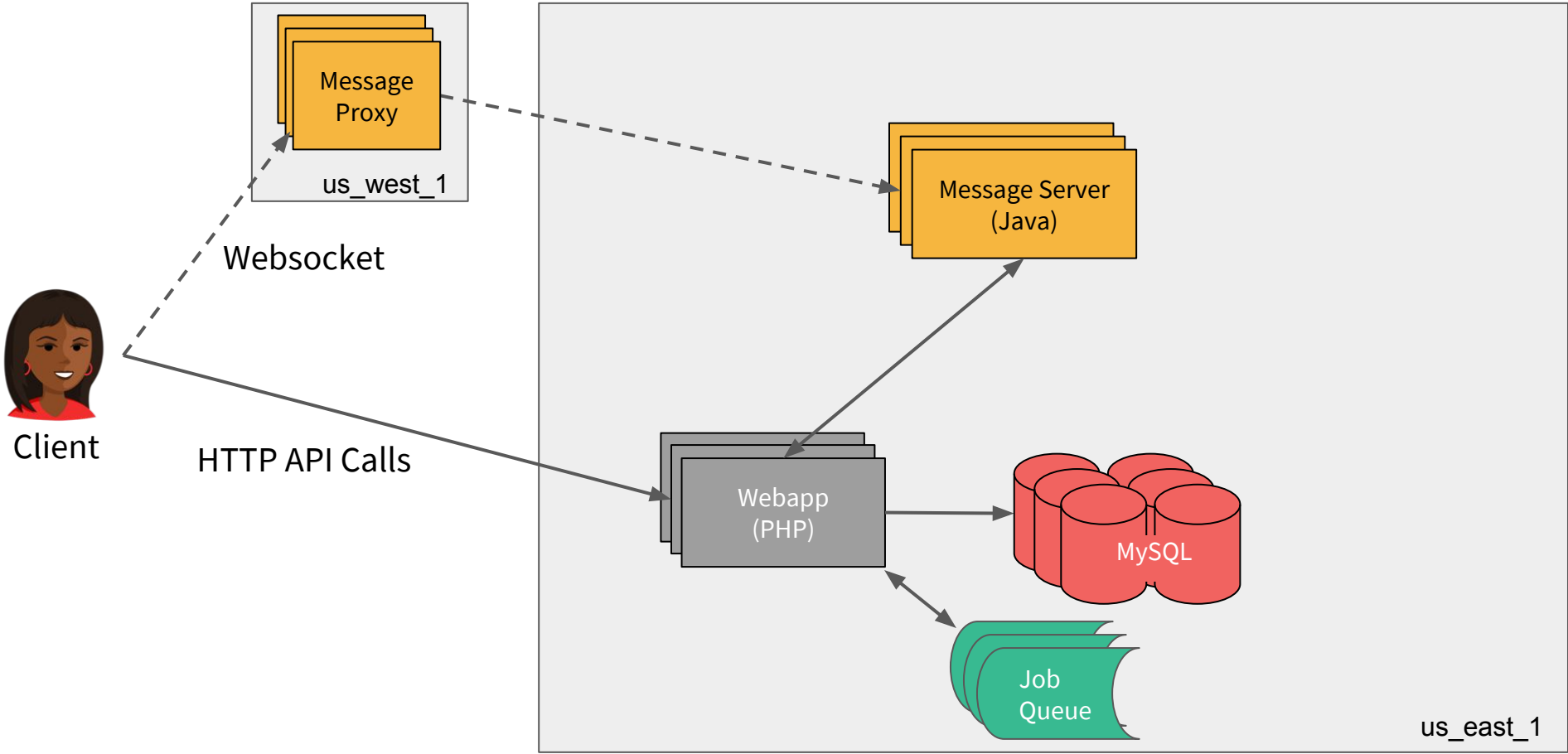
User Base
4M Daily Active Users

Largest Organizations
>10,000 Active Users

Connectivity
2.5M peak simultaneous connected
Avg 10 hrs/day

Engineering Style
Conservative, Pragmatic, Minimal
Most systems > 10 year old technology

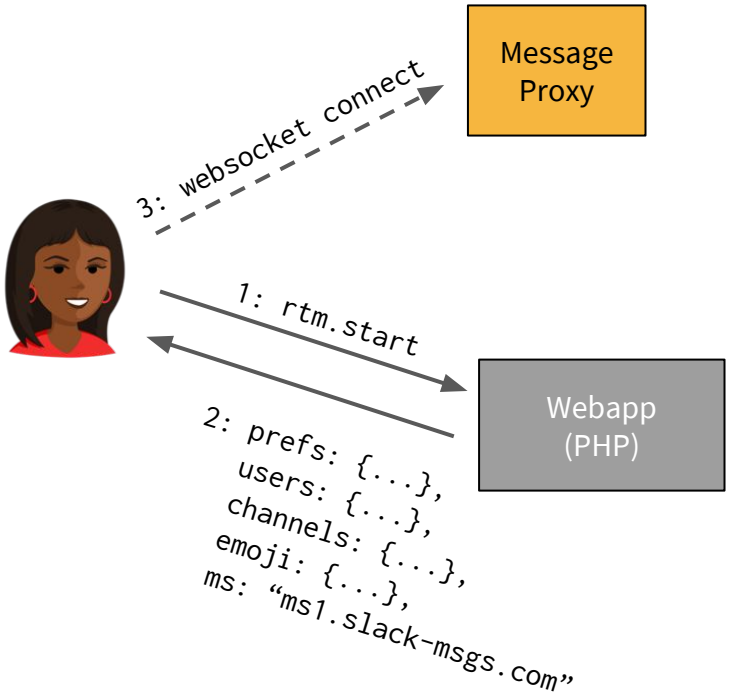
How Slack Works (2016)



Client / Server Flow

Initial login:

- Download **full workspace model** with all channels, users, emoji, etc.
- Establish **real time websocket**



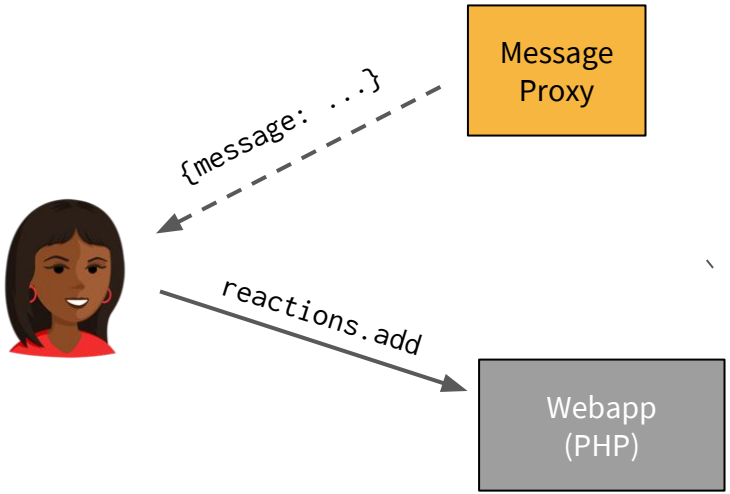
Client / Server Flow

Initial login:

- Download **full workspace model** with all channels, users, emoji, etc.
- Establish **real time websocket**

While connected:

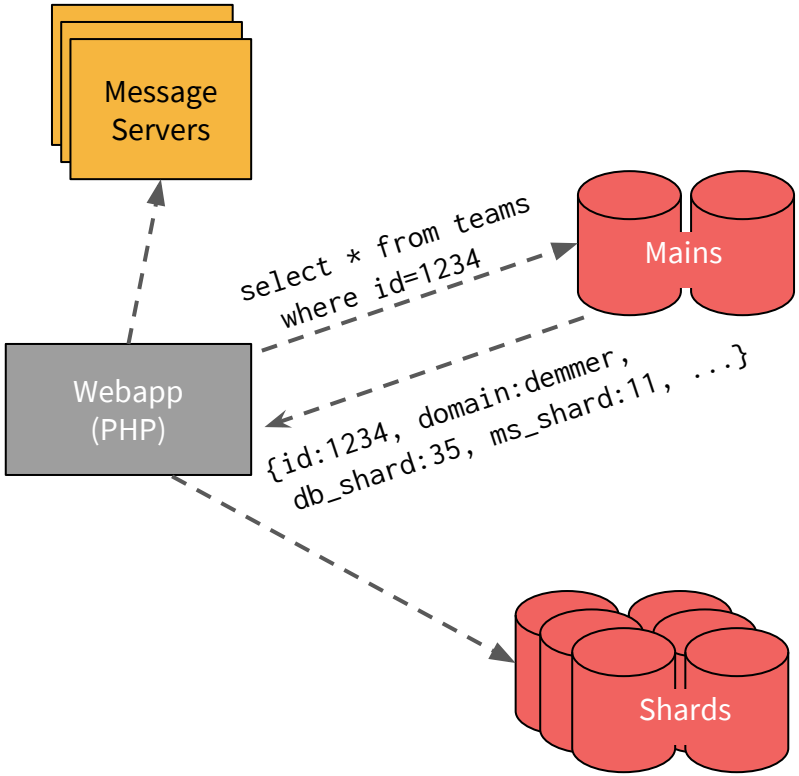
- **Push updates** via websocket
- **API calls** for channel history, message edits, create channels, etc.



Sharding And Routing

Workspace Sharding

- Assign a workspace to a DB and MS shard at creation
- Metadata table lookup for each API request to route



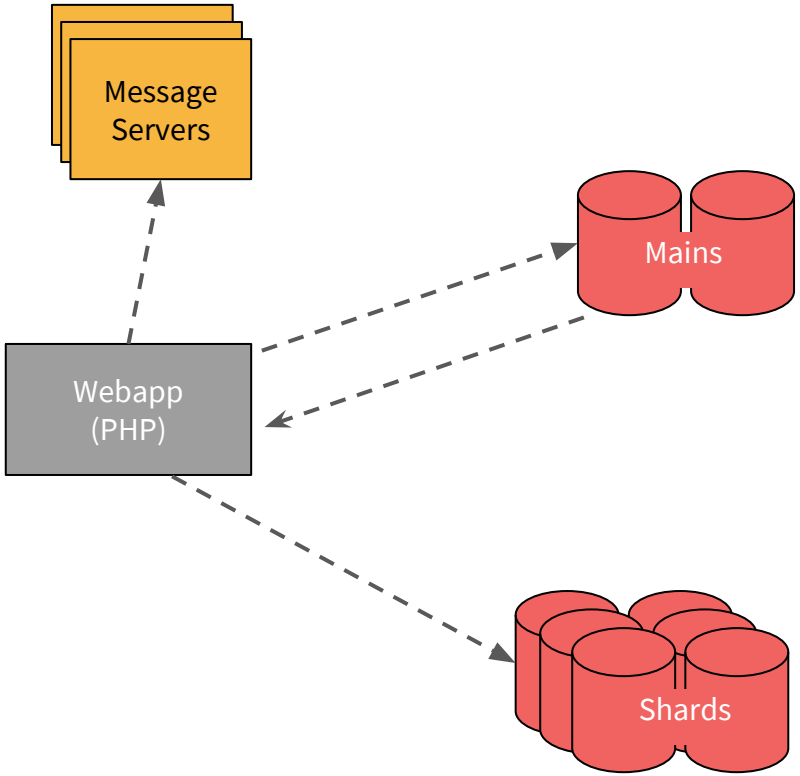
Sharding And Routing

Workspace Sharding

- Assign a workspace to a DB and MS shard at creation
- Metadata table lookup for each API request to route

“Herd of Pets”

- DBs run in active/active pairs with application failover
- Service hosts are addressed in config and manually replaced



Why This Worked

Client Experience

Data model lends itself to a seamless, **rich real-time client experience**.

- Full data model available in memory
- Updates appear instantly
- Everything feels real time

Server Experience

Implementation model is **straightforward**, **easy** to reason about and **debug**.

- All operations are workspace scoped
- Horizontally scale by adding servers
- Few components or dependencies

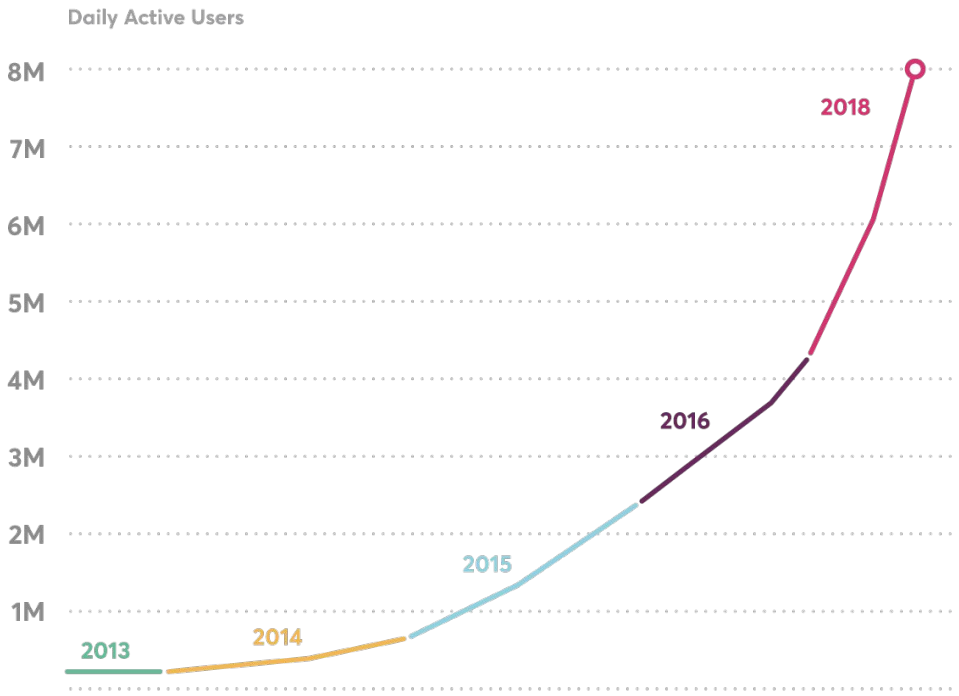
Things Get More Interesting...

Things Get More Interesting

Size and Scale

Product Model

Slack Growth



8M+ daily active users

4M+ of users outside of US

500K+ organizations use Slack in more than 100 countries

Slack Facts (2018)

User Base
>8M Daily Active Users

Largest Organizations
>125,000 Active Users

Connectivity
>7M peak simultaneous connected
Avg 10 hrs/day

Engineering Style
Still pragmatic, but embrace complexity where needed to solve hardest problems

Slack Facts (2018)

2x

User Base
>8M Daily Active Users

10x !

Largest Organizations
>125,000 Active Users

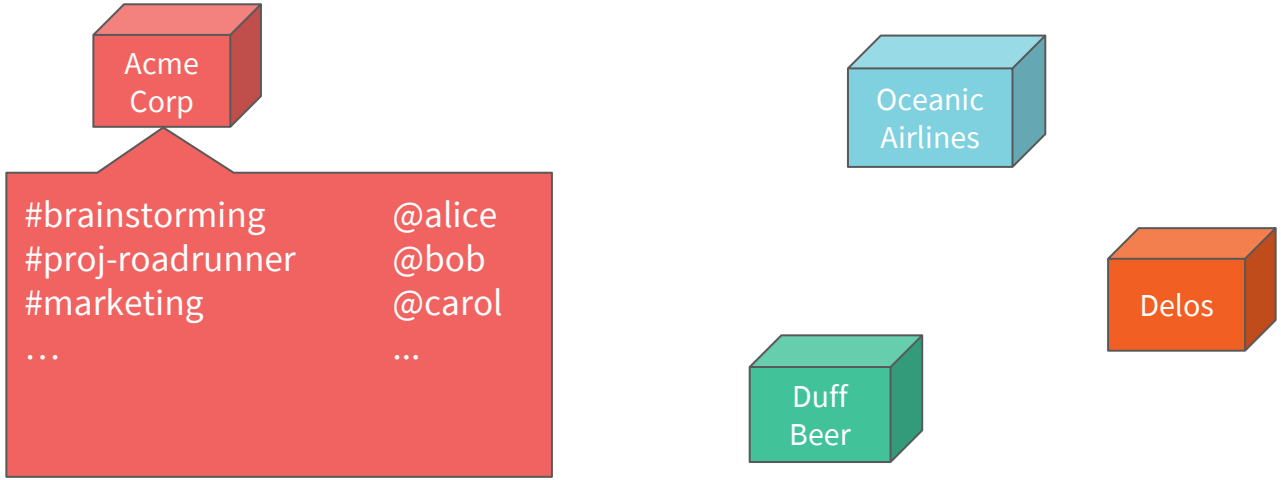
3x

Connectivity
>7M peak simultaneous connected
Avg 10 hrs/day

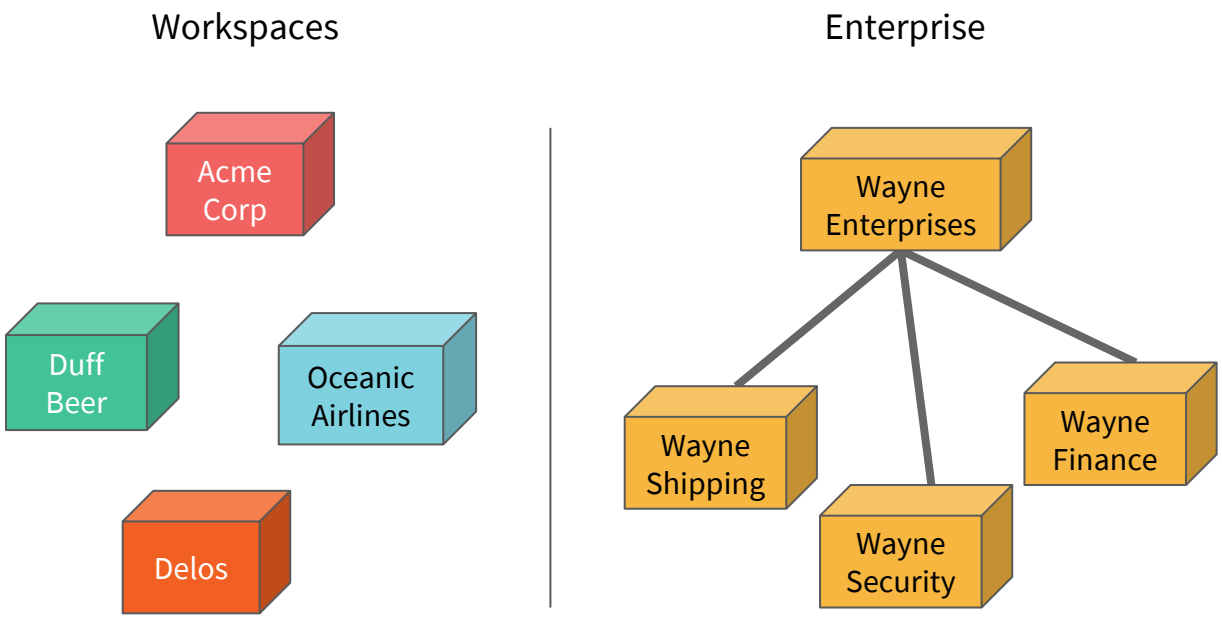
Engineering Style
Still pragmatic, but embrace complexity where needed to solve hardest problems

Change the Model

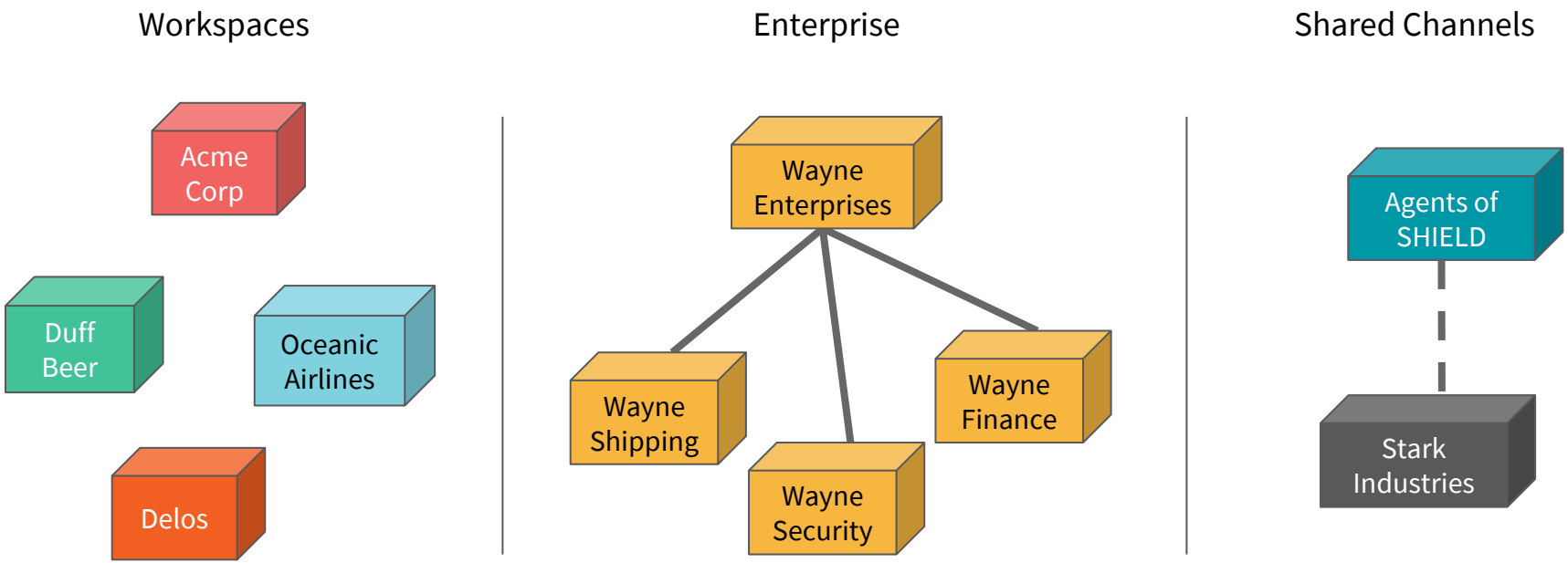
A **workspace** logically contains all **channels** and **messages**, as well as **users**, **emoji**, **bots**, and more. All interactions occur within the workspace boundary.



Change the Model



Change the Model



Challenges

Recurring Issues

- **Large organizations**: Boot metadata download is slow and expensive
- **Thundering Herd**: Load to connect >> Load in steady state
- **Hot spots**: Overwhelm database hosts (mains and shards) and other systems
- **Herd of Pets**: Manual operation to replace specific servers
- **Cross Workspace Channels:** Need to change assumptions about partitioning

So What Did We Do?

What Did We Do

Thin Client Model

Flannel Cache

Vitess

Fine-Grained DB Sharding

Message Services

Service Decomposition

What Did We Do

Flannel Cache

Challenge: Boot Model Explosion

```
boot_payload_size ~=
   (num_users * user_profile_bytes) +
   (num_channels * (channel_info_size +
                    (num_users_in_channel * user_id bytes)))
```

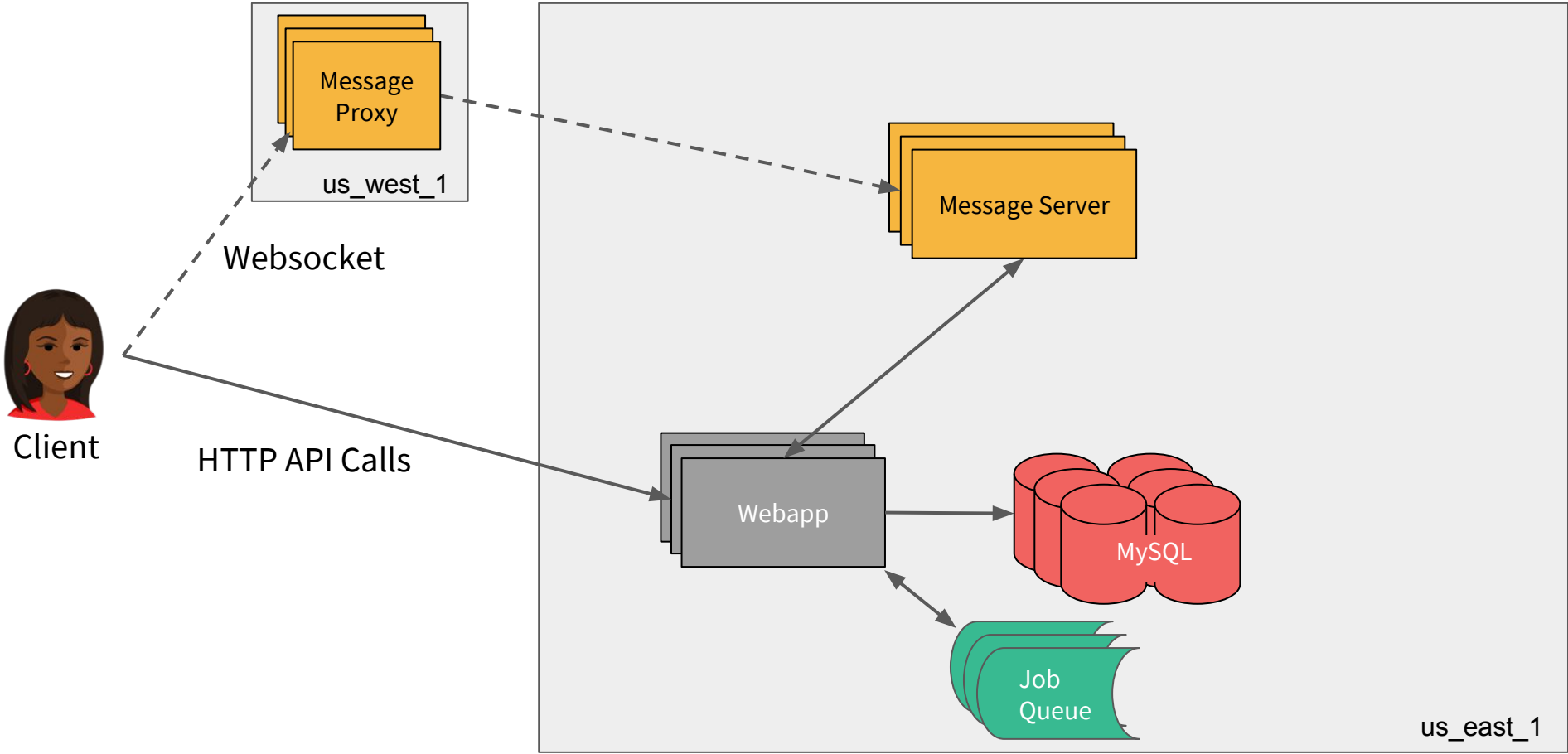
Users	Profiles	Channels	Total
12	6 KB	1 KB	**7 KB**
530	140 KB	28 KB	**168 KB**
4,008	5 MB	2 MB	**7 MB**

Challenge: Boot Model Explosion

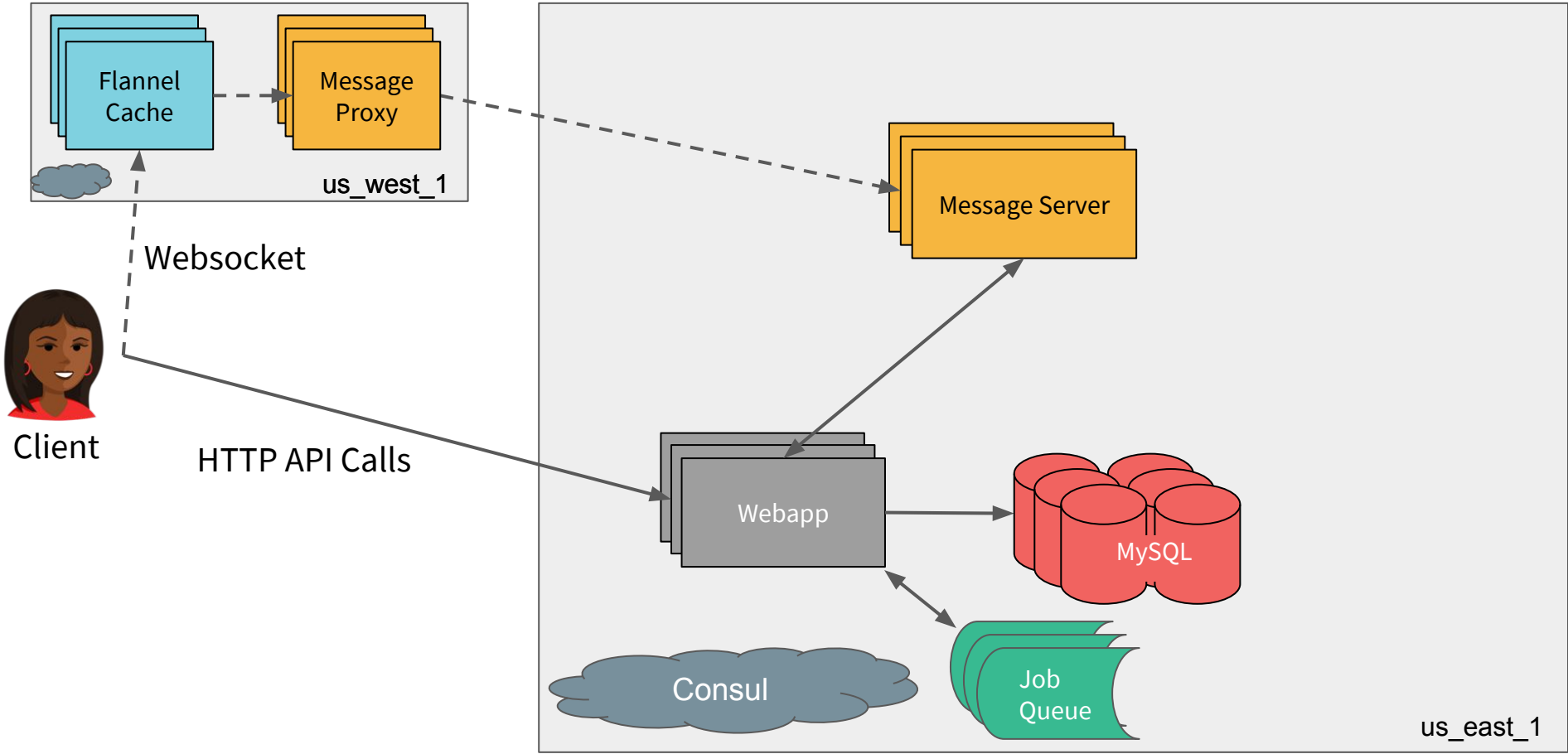
```
boot_payload_size ~=
   (num_users * user_profile_bytes) +
   (num_channels * (channel_info_size +
                    (num_users_in_channel * user_id bytes)))
```

Users	Profiles	Channels	Total
12	6 KB	1 KB	**7 KB**
530	140 KB	28 KB	**168 KB**
4,008	5 MB	2 MB	**7 MB**
44,030	36 MB	25 MB	**59 MB**
148,170	78 MB	40 MB	**118 MB**

Thin Client Model



Thin Client Model



Thin Client Model

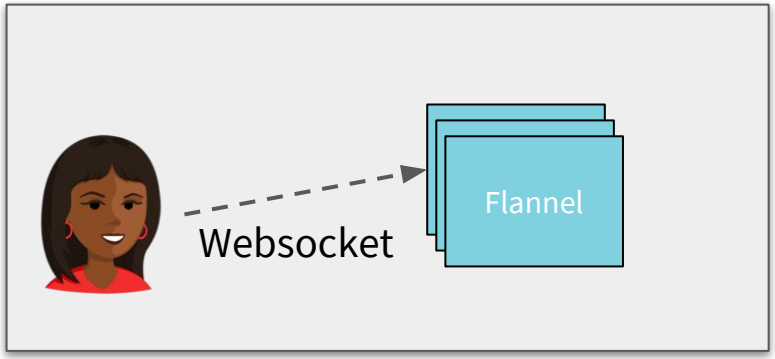
Flannel Service
Globally distributed **edge cache**

Routing
Workspace affinity for cache locality

Cache Updates
Proxy **subscription messages** to clients

Minimize Workspace Model
Much smaller **boot payload**

Query API
Fetch **unknown objects** from cache



Thin Client Model

Unblock Large Organizations

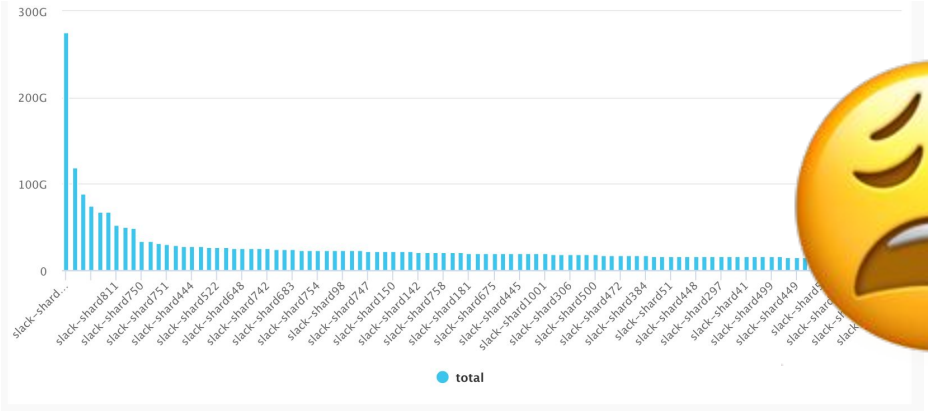
Adapting clients to a lazy load model was a critical change to enable Slack for large organizations.

- Huge reduction in payload times on initial connect
- Flannel efficiently responds to > 1+ million queries per second
- Adds challenges of cache coherency and reconciling business logic

What Did We Do

Fine-Grained
DB Sharding

Challenge: Hot Spots & Manual Repair



62 lines (35 sloc) 2.98 KB

Raw Blame History

2017-08-25 shard218 thread exhaustion for one hour

2017-09-07 Shards 150, 284, and 644 overwhelmed with

81 lines (66 sloc) 7.39 KB

Raw Blame History

2017-06-28 Channel Highlights Introduced New Load on the [redacted] Shard

16.3 KB

Raw Blame History

-07-13 [redacted] Flannel and Shard are Hot

Timeline

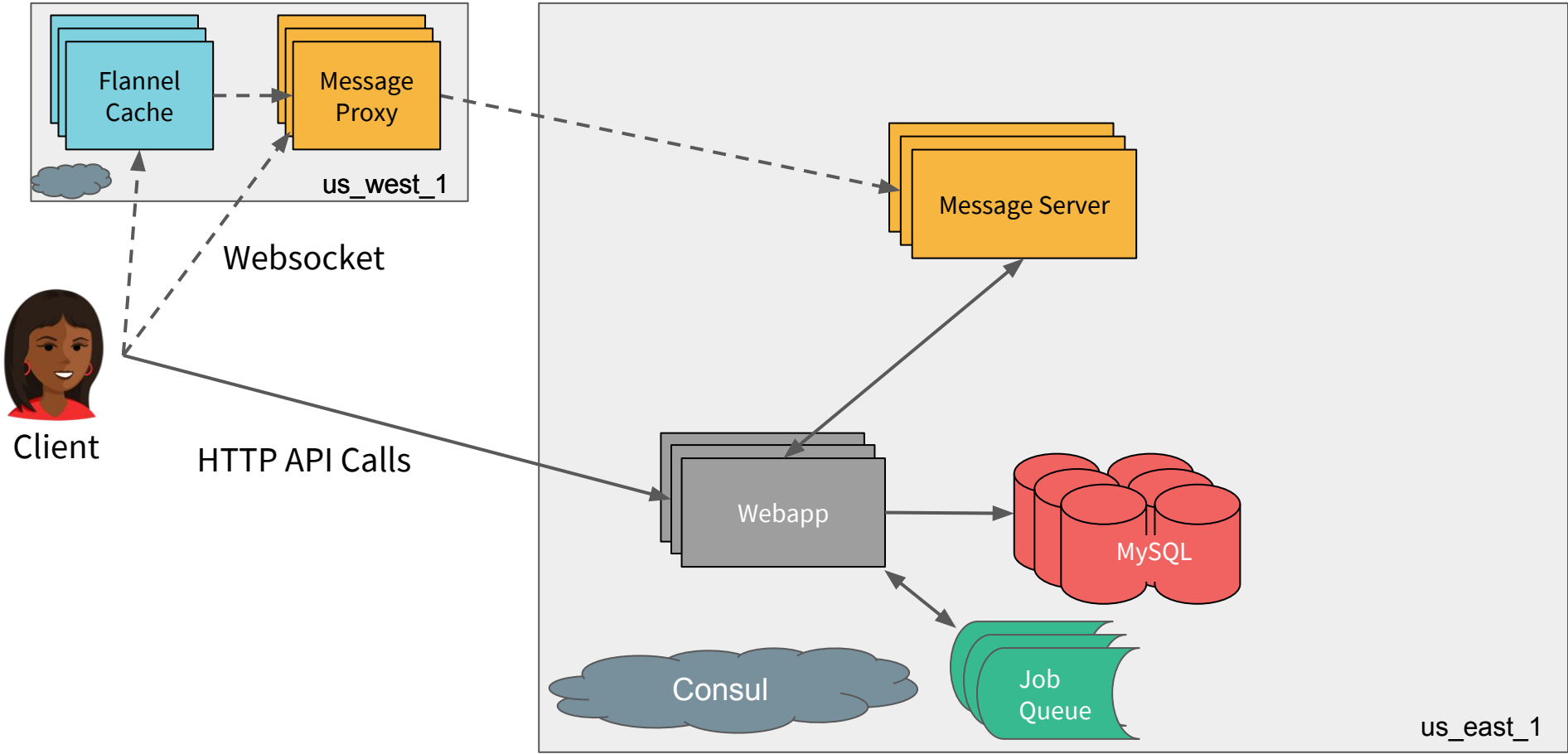
nagios APP 7:56 AM

Host '[redacted]' is DOWN:
CRITICAL - Socket timeout after 10 seconds

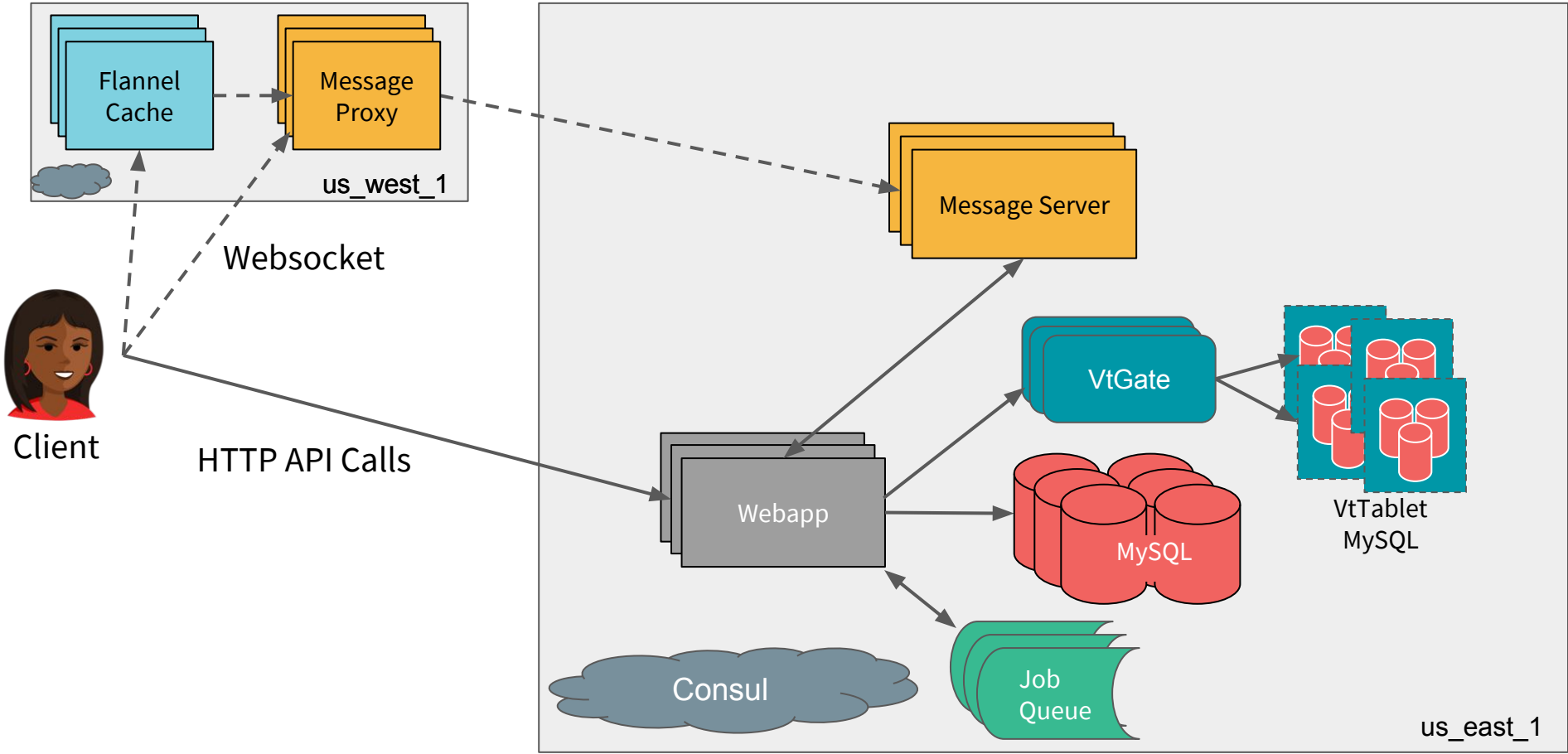
@slack-sha[redacted] APP 8:04 PM

skipped SQL statements in `/mnt/mysql_dupes/ff-dupe-pk-2017-09-18T20:04:02-K9VMsj.log` need to be inspected

Vitess


Flannel Cache
Message Proxy
us_west_1
Message Server
Websocket
Client
HTTP API Calls
Webapp
MySQL
Consul
Job Queue
us_east_1

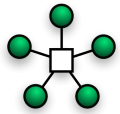
Vitess


Flannel Cache
Message Proxy
us_west_1
Message Server
Websocket
Client
HTTP API Calls
VtGate
VtTablet MySQL
Webapp
MySQL
Consul
Job Queue
us_east_1

Vitess

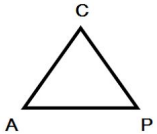
Flexible Sharding

Vitess manages **per-table sharding** policy



Topology Management

Database servers **self-register**

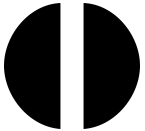


Single Master

Using **GTID and semi-sync** replication

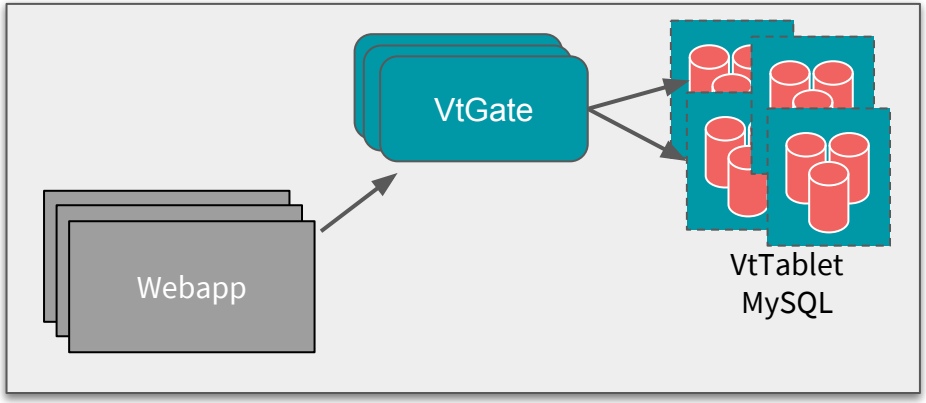
Failover

Orchestrator promotes a replica on failover



Resharding Workflows

Automatically **expand the cluster**



Vitess

Fine-Grained Sharding

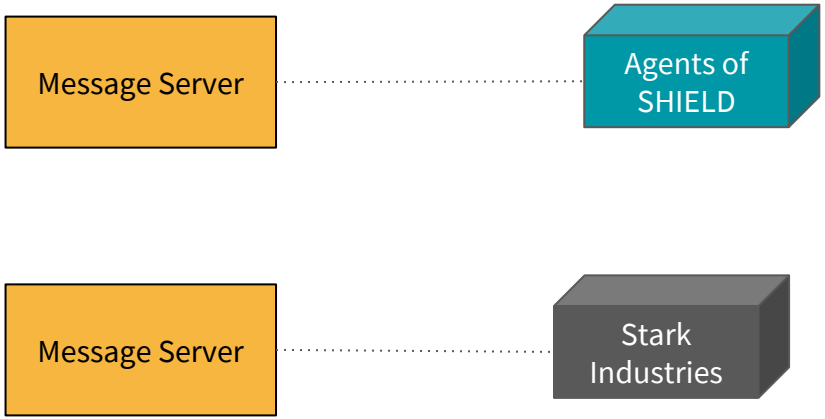
Migrating to a channel-sharded / user-sharded data model helps mitigate hot spots for large teams and thundering herds.

- Retains MySQL at the core for developer / operations continuity
- More mature topology management and cluster expansion systems
- Data migrations that change the sharding model take a long time

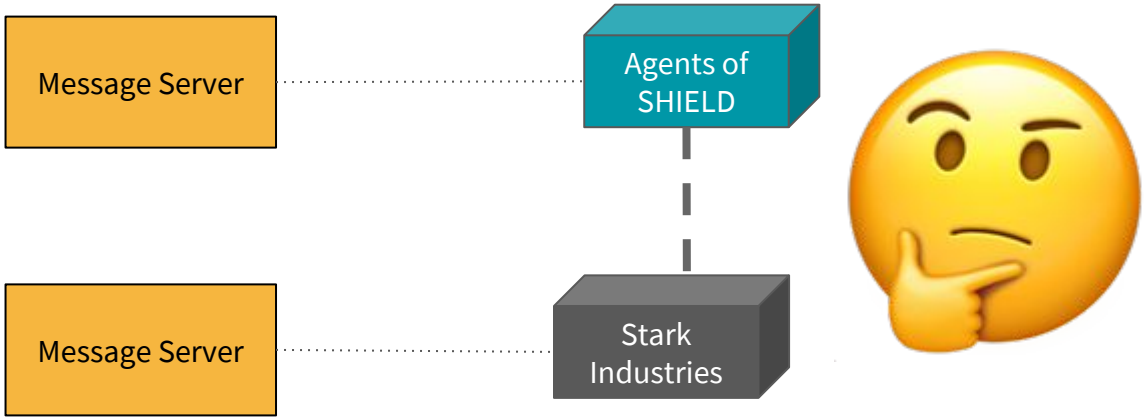
What Did We Do

Service
Decomposition

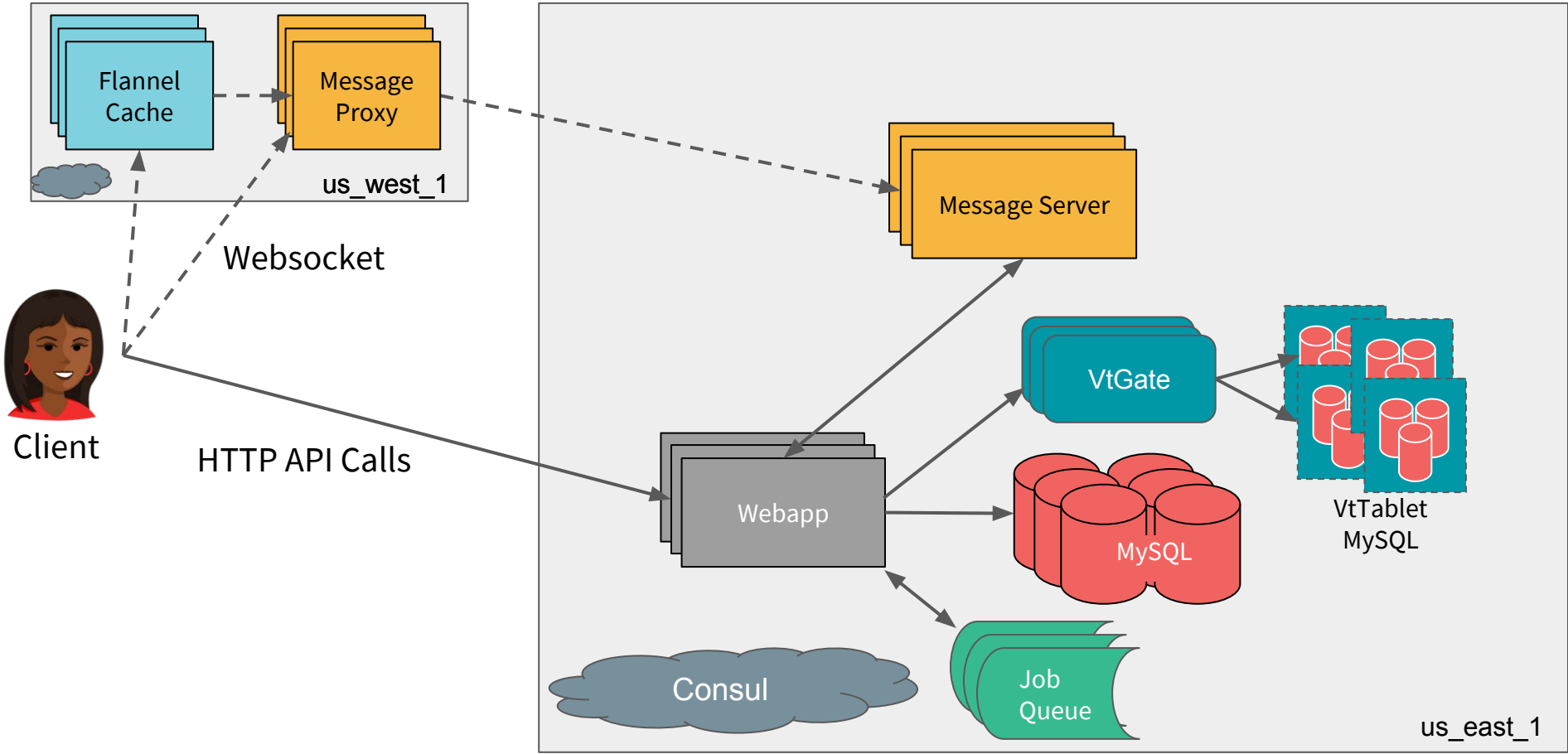
Challenge: Shared Channels?



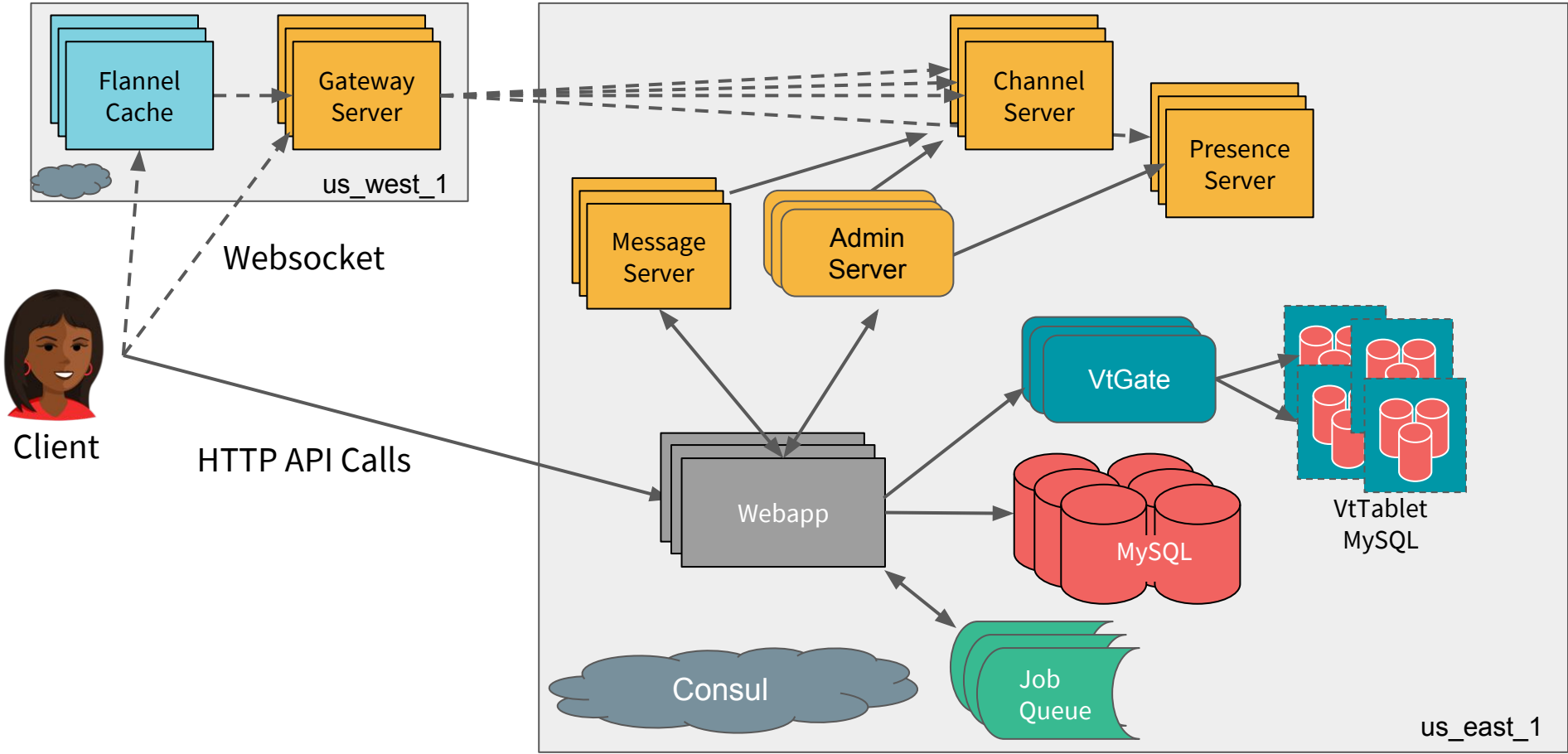
Challenge: Shared Channels?



Message Server to Services



Message Server to Services



Message Server to Services

Gateway Server

Websocket termination and subscriptions

Channel Server

Pub/Sub **fanout** with 5 minute buffering

Presence Server

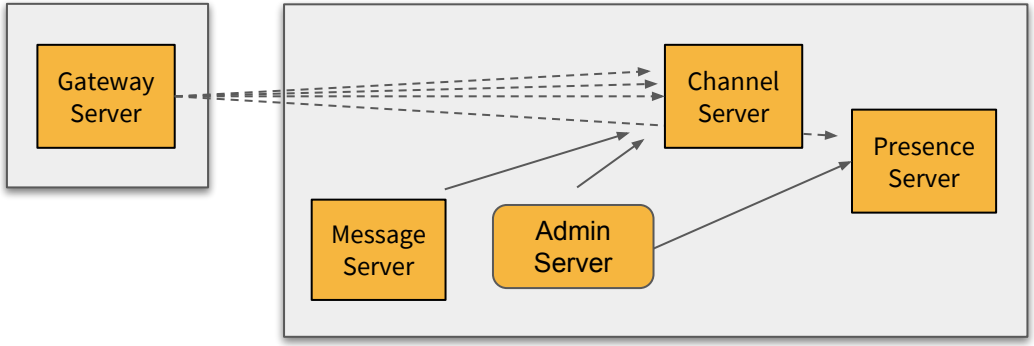
Store and distribute **presence state**

Admin Server

Cluster management and routing

(Legacy) Message Server

Used for reminders, Google Calendar integration



Message Server to Services

Generic Messaging Services

Everything is a pub/sub “channel”, including message channels as well as workspace / user metadata channels.

- Clients / Flannel subscribes to updates for all relevant objects
- Each Message Service has dedicated clear roles and responsibilities
- Self-healing cluster orchestration to maintain availability
- Each user session now depends on many more servers being available

What Did We Do

Flannel Cache

Fine-Grained
DB Sharding

Service
Decomposition

slack

Some Themes...

Herd of Pets to Service Mesh

Topology Management

For each of these projects (and more), architecture evolved from hand-configured server hostnames to a discovery mesh.

- Enables self-registration and automatic cluster repair
- Adds reliance on service discovery infrastructure (consul)
- Led to changes in service ownership and on-call rotation

Scatter May Be Harmful

Fine-Grained Sharding

Migrating from a workspace-scope to channel or user scoped spreads out the load but adds a requirement to sometimes scatter/gather.

- Removes artificial couplings on back end systems
- Teams are less isolated, so need extra protections from noisy neighbors
- When scattering, clients should tolerate partial results and retry
- Tail latencies can dominate performance when fetching from many

Deploying Is Only The Beginning

Deprecation Challenges

As hard as it is to add new services into production under load, it’s proven as hard if not harder to remove old ones.

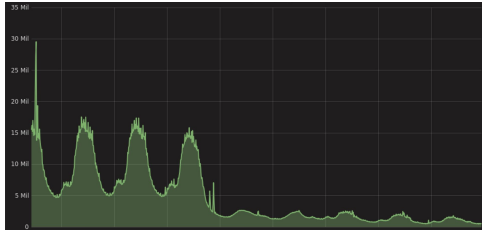
- With few exceptions, all 2016 services still in production
- Need to support legacy clients and integrations
- Data migrations need application changes takes time

Grinding It Out

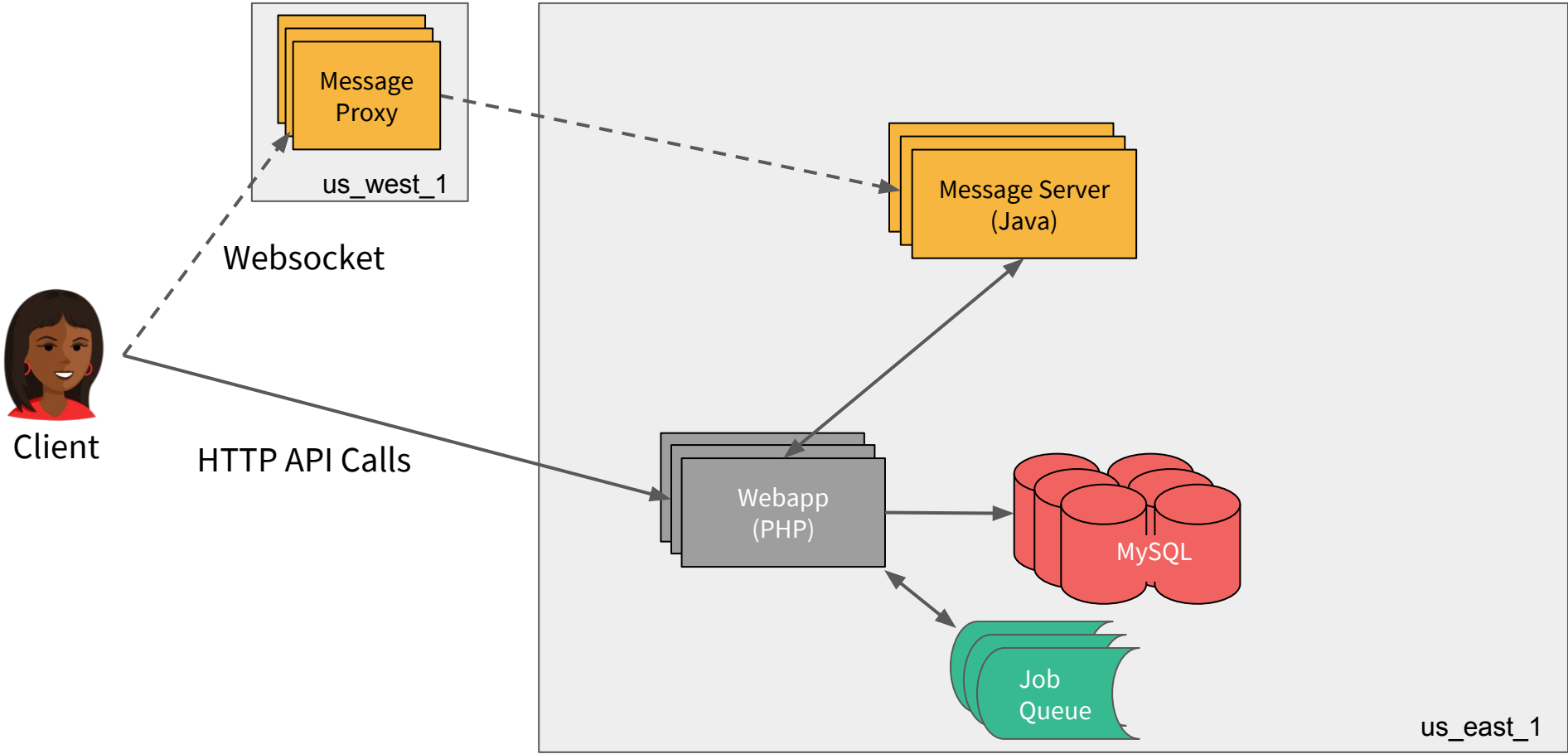
Performance Short Game

Architectural rework is necessary, but less glamorous performance optimizations pay huge dividends

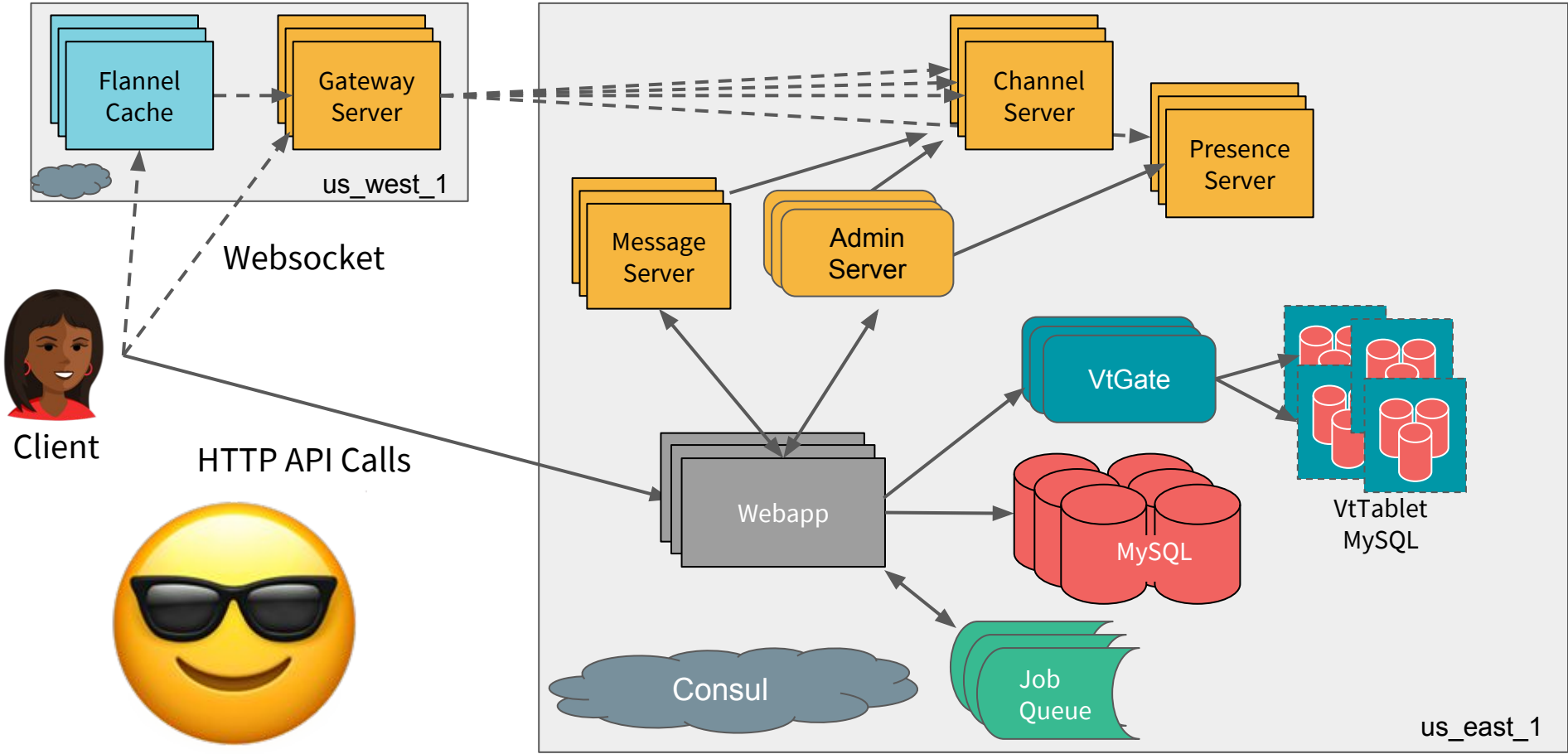
- Simple approaches to caching or refactoring
- Client-side jitter to spread out load
- Eliminate unnecessary methods / queries



How Slack Works (2016)



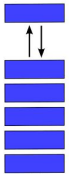
How Slack Works (2018)



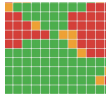
We're Not Done Yet

Storage POPs
Geographically distributed back end

Services Services Services
Decompose the monolith and improve service mesh.



Job Queue
Revamp the asynchronous task queue



Resiliency
Degraded functionality when subsystems are unavailable

Eventual Consistency
Change API expectations

Network Scale
Stay ahead of the growth curve

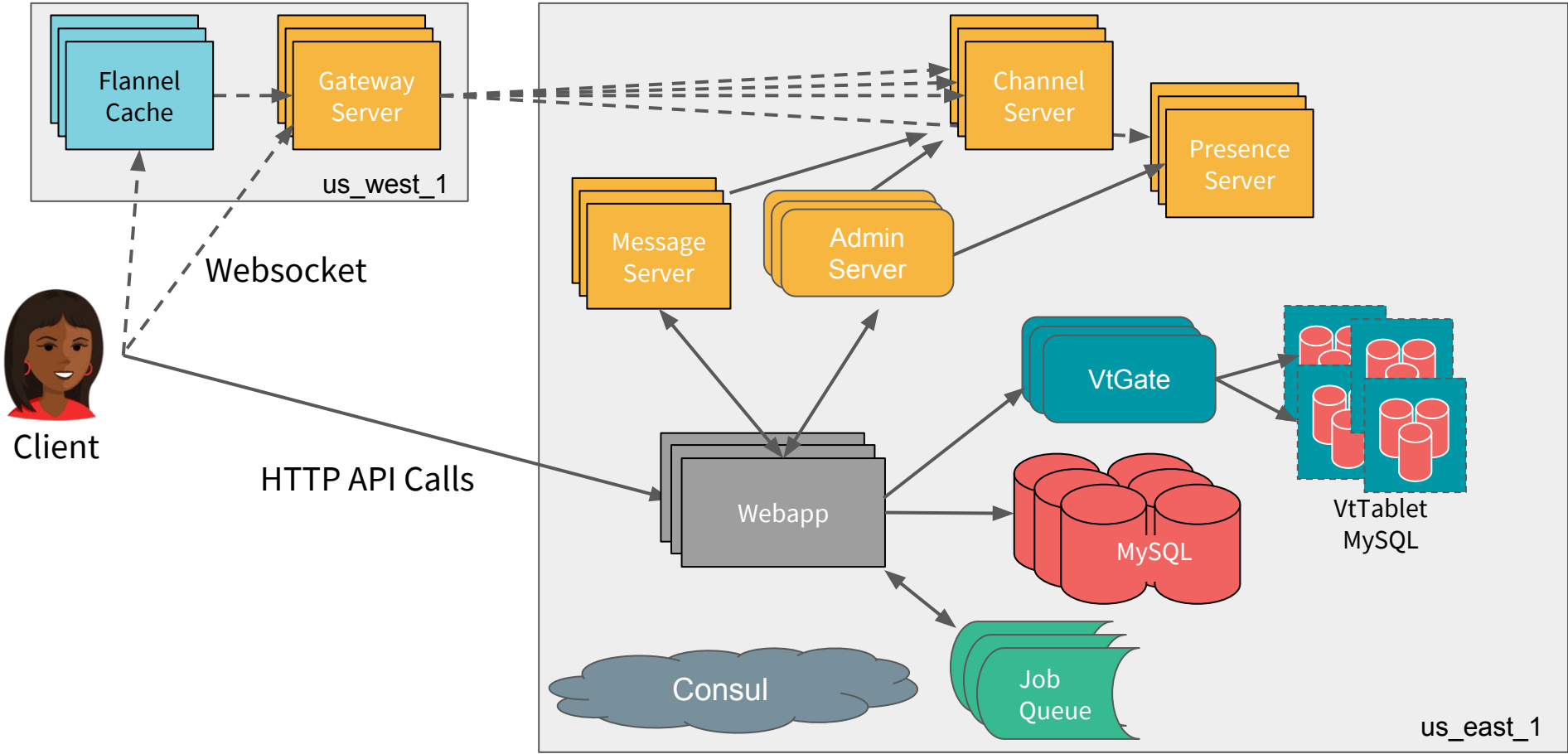
slack

Thank You!

slack

BACKUP

How Slack Works (c 2018)



Message Server

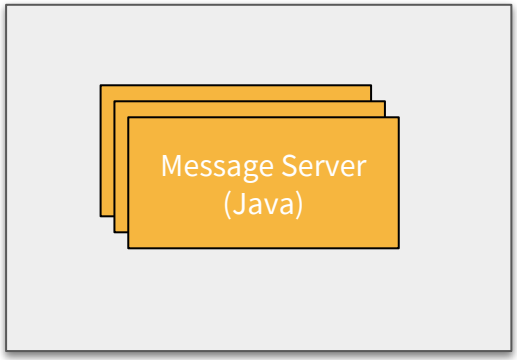
Client Connections
Websocket termination, user / connection state and subscriptions

Webapp Actions
Communication/routing from Webapp → Message Server for channel messages

Presence Indications
User presence state, updates & presence subscriptions - that little green indicator

Subscriptions and Fanout
Last 5 minutes of history, as well as initial subscription and fanout of messages

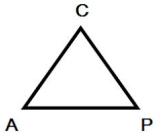
Scheduled Messages
Used for reminders, Google Calendar integration



Team Sharded MySQL

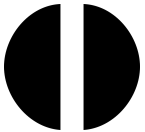
Team Sharding
Application-defined sharding policy routes all queries to the team shard

Manual Topology Management
Operator-managed host configuration is injected into application code

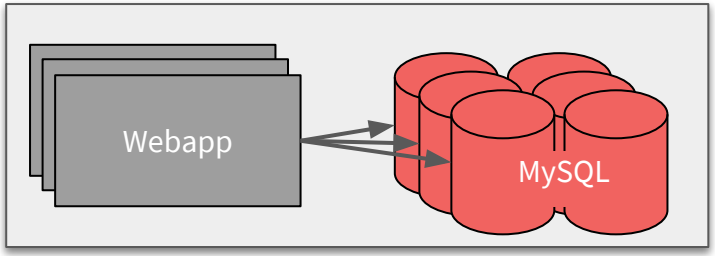


Active Master / Master
Both sides are writable masters, biases for availability with best-effort consistency

Application Retry Failover
If preferred side is unavailable, connect to the backup side and try again



Split Shards
Manually orchestrated switchover to divide some teams to new host.



QCon 2016

You are viewing an OLD QCon year's website. Visit QCon SF 2018 for this year's event.

QCon

SAVE THE DATE FOR QCON SF 2017

CONFERENCE: Nov 13-15, 2017

WORKSHOPS: Nov 16-17, 2017

SCHEDULE SPEAKERS LOG IN

Sign up to be informed when registration opens for QCon SF 2017 (Nov 13-17)

Talk: How Slack Works

Track: Architectures You've Always Wondered About

Location: **Ballroom A**

Day of week: **Monday**

Duration: **10:35am - 11:25am**

Slack is a persistent group messaging app for teams. Slack's 3.4 million active users expect high levels of reliability, low latency, and extraordinarily rich client experiences across a wide variety of devices and network conditions. In this talk, we'll take a tour of Slack's infrastructure, from native and web clients, through the edge, into the Slack datacenter, and around the various services that provide real-time messaging, search, voice calls, and custom emoji.

QCon 2017

CELEBRATING 11 YEARS

are viewing an OLD QCon year's website. Visit QCon SF 2018 for this year's event.

QCon SAN FRANCISCO

SAVE THE DATE FOR QCON SF 2018

CONFERENCE: Nov 5-7, 2018

WORKSHOPS: Nov 8–9, 2018

SCHEDULE SPEAKERS WORKSHOPS

LOG IN

Sign up to be informed when registration for QCon SF 2018 (Nov 5-9) opens!

Presentation: Scaling Slack

Track: Architectures You've Always Wondered About

Location: **Ballroom A**

Duration: **2:55pm - 3:45pm**

Day of week: **Monday**

Level: **Intermediate**

Persona: **Architect, Technical Engineering Manager**

More talks on:

Abstract

Slack is a communication and collaboration platform for teams. Our millions of users spend 10+ hrs connected to the service on a typical working day. They expect reliability, low latency, and extraordinarily rich client experiences across a wide variety of devices and network conditions. In the talk, we'll examine the limitations that Slack's backend ran into and how we overcame them to scale from supporting small teams to serving gigantic organizations of hundreds and thousands of users. We'll hear stories about the edge cache service, real-time messaging system and how they evolved for major product efforts including Grid and Shared Channels.