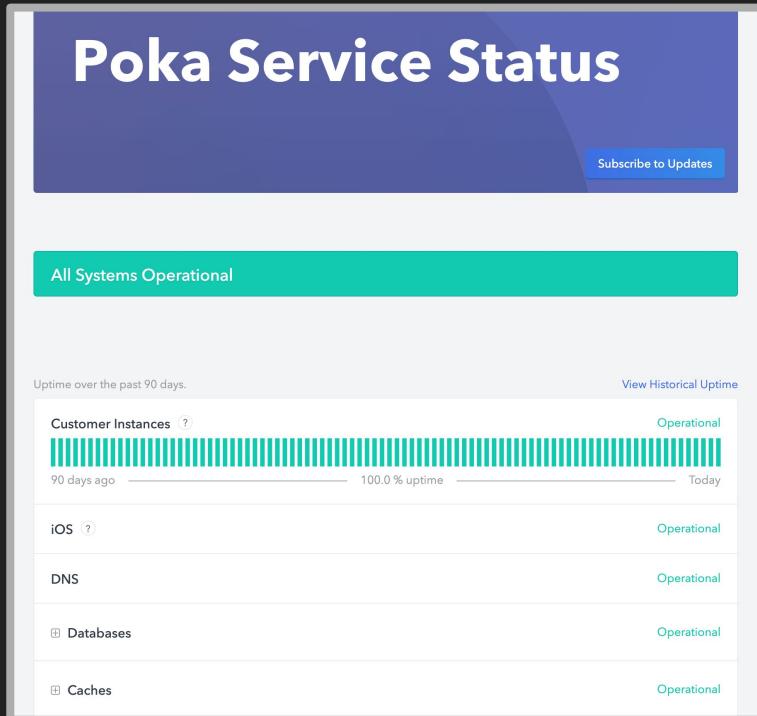


Zero Downtime Migration of Our Distributed Task Queue System



Edmund Lam
Staff Software Developer @ Poka

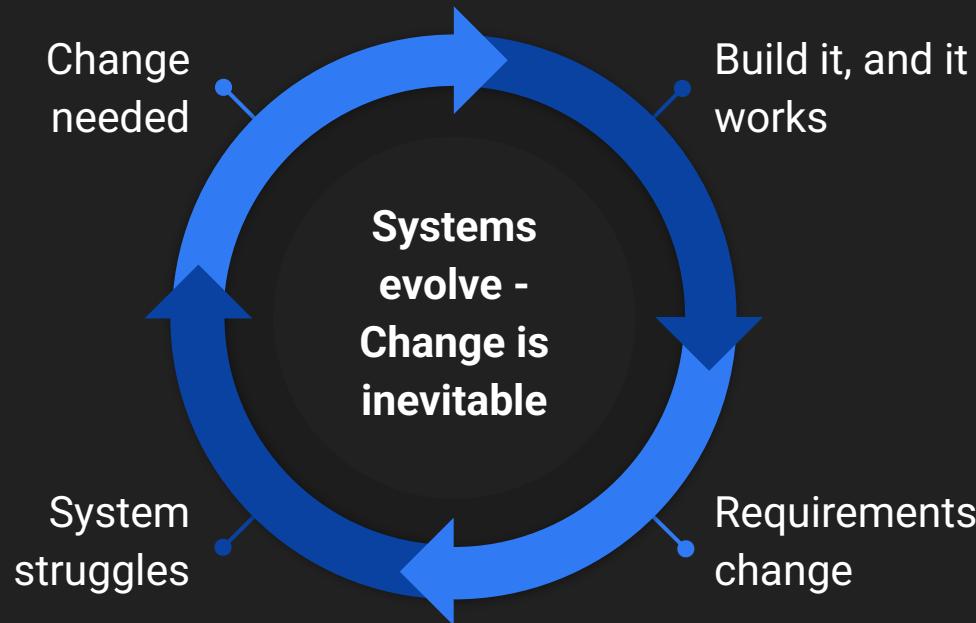
At first...



Now...



Why this talk matters:



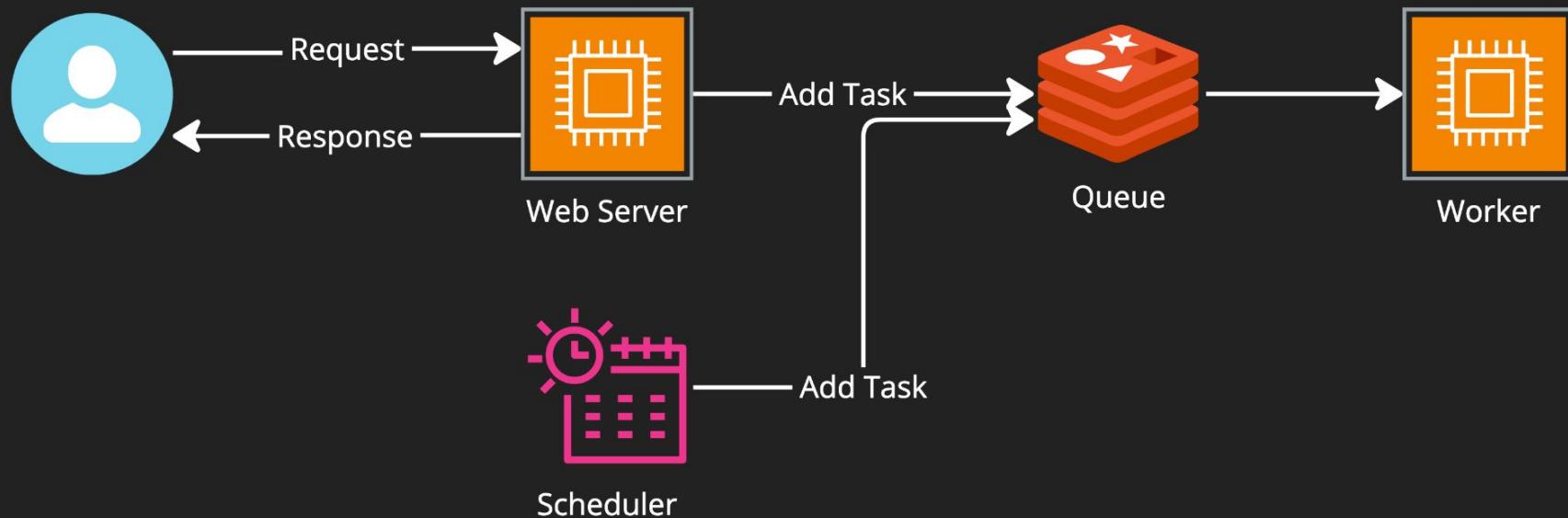
A dramatic painting depicting a naval battle at sea. In the center, a large three-masted sailing ship with its sails partially unfurled is engaged in combat, its hull dark and weathered. Several other ships are visible in the background and foreground, some with their sails fully unfurled and others appearing as smaller figures. The sky is filled with heavy, dark clouds, and the overall atmosphere is one of chaos and conflict.

6 key lessons

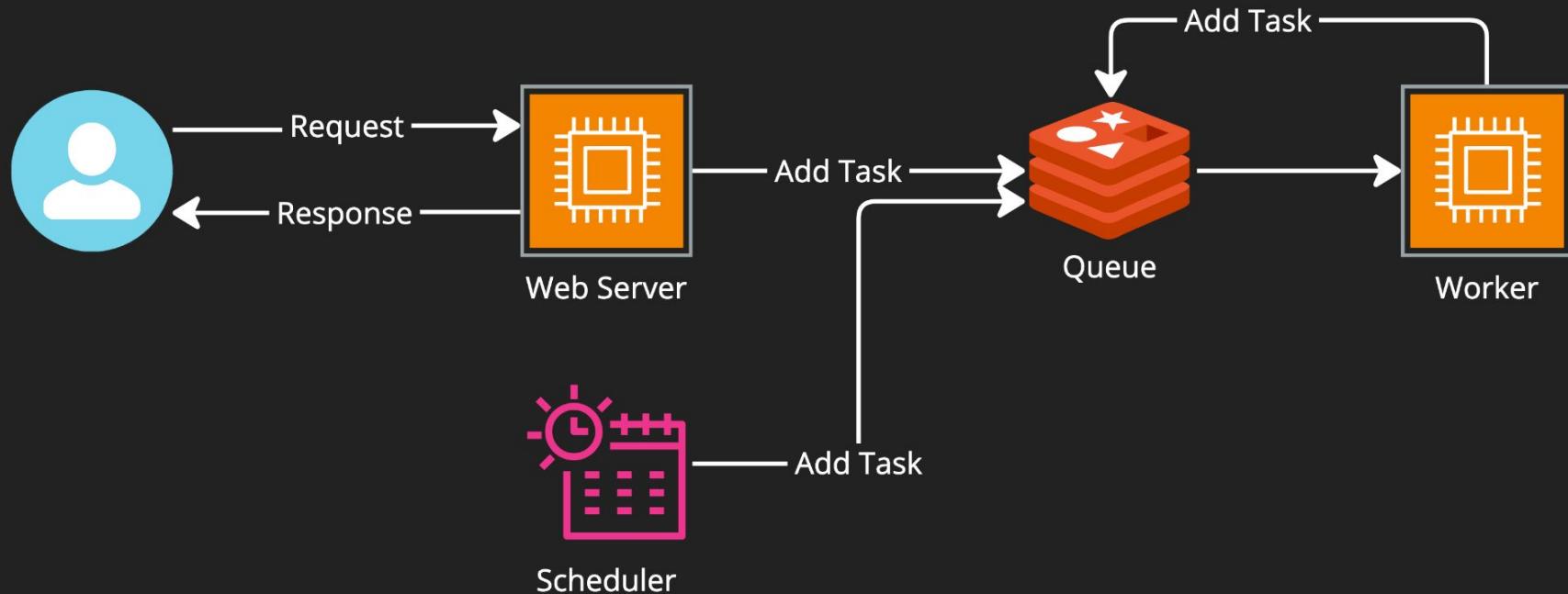
What Does a Distributed Task Queue Do?



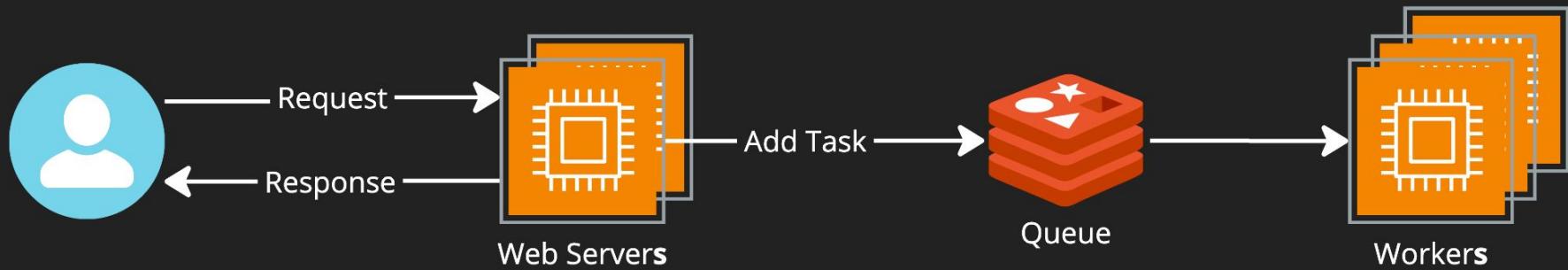
What Does a Distributed Task Queue Do?



What Does a Distributed Task Queue Do?



What Does a Distributed Task Queue Do?



Task Throughput

67.7 /s

50.9 /s

0

Total tasks per day

128k

5M

0

1d

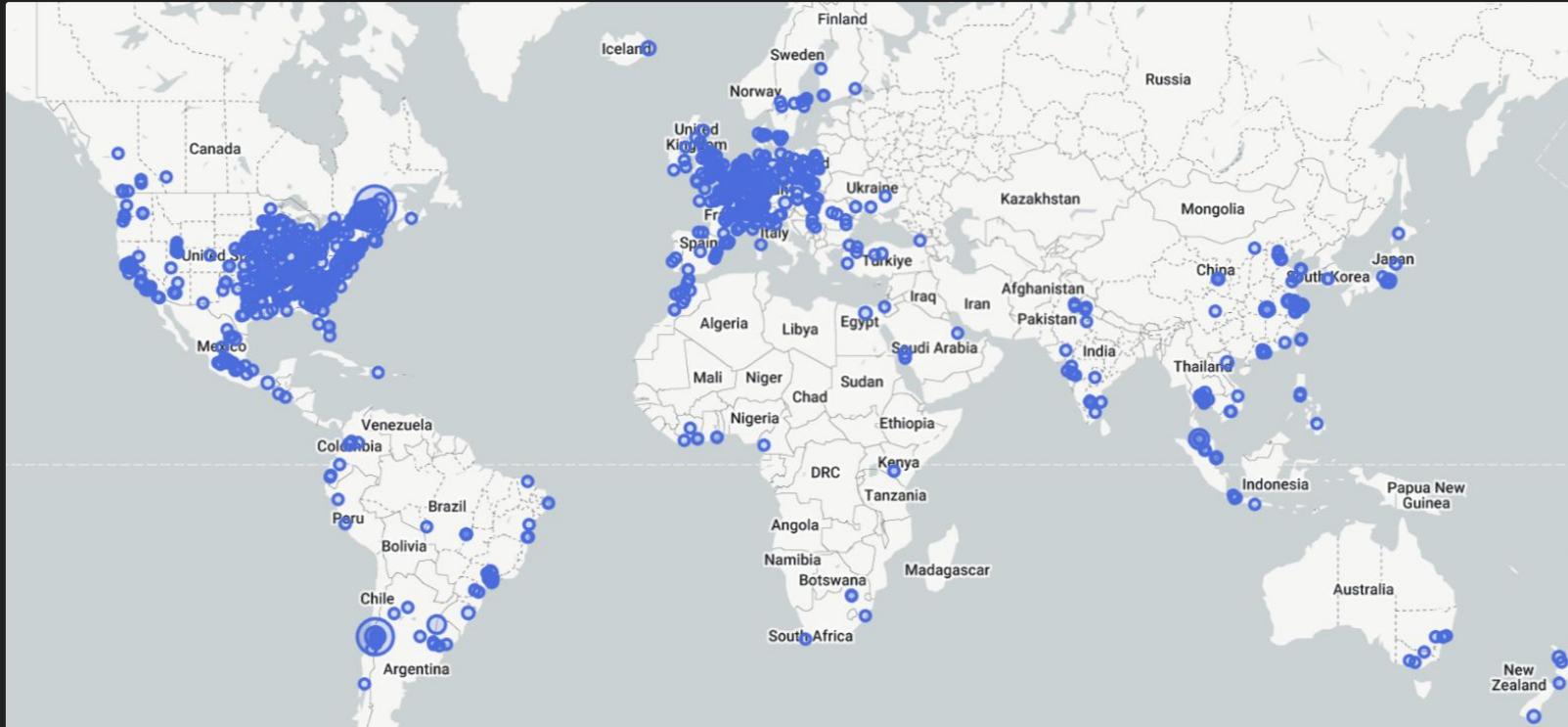
Poka by the Numbers

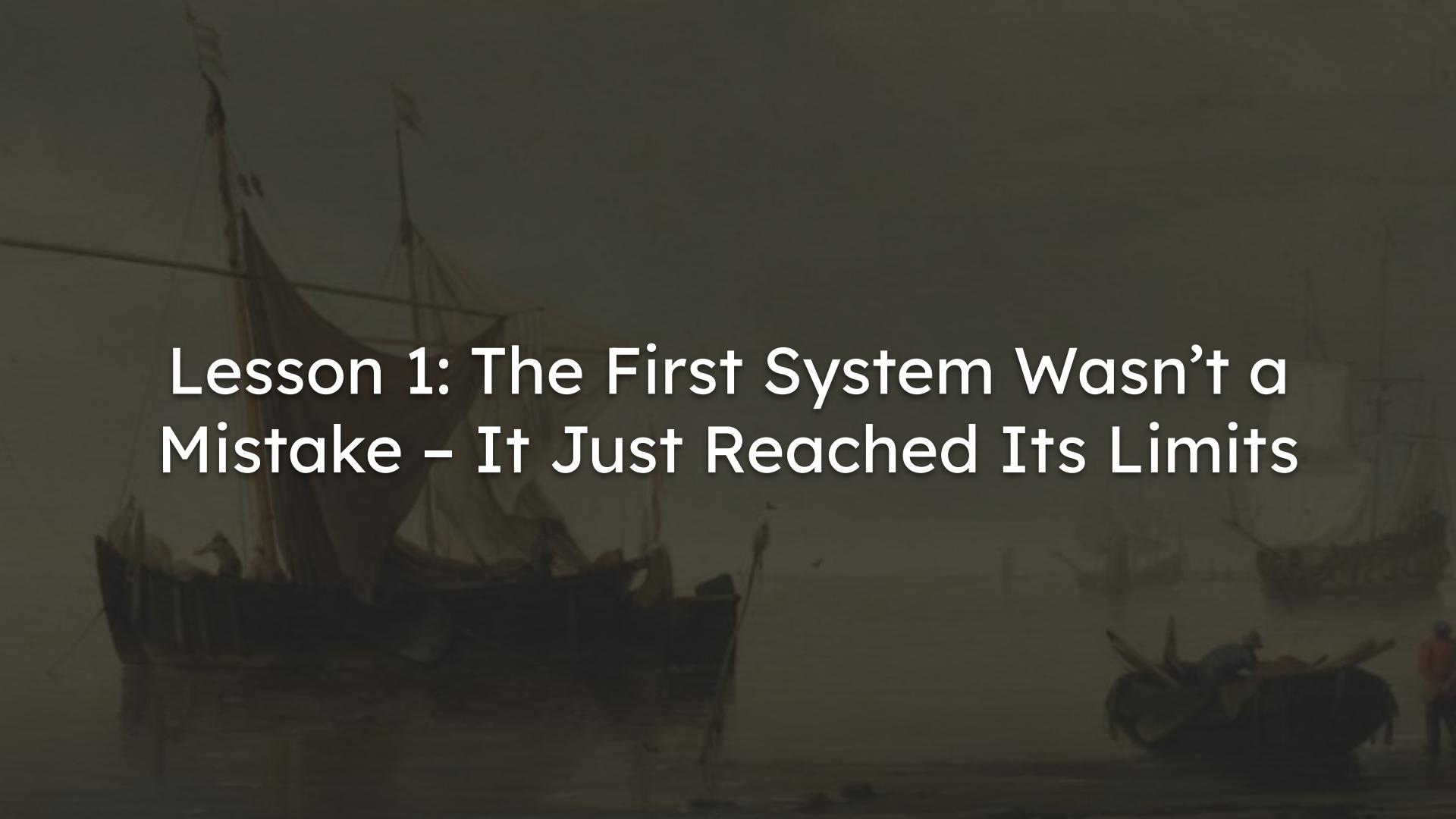
 **475K+**
Users

 **70+**
Countries

 **1,600+**
Factories

 **36**
Supported
Languages



A dark, atmospheric photograph showing several traditional wooden boats with sails on a body of water. The scene is hazy and dimly lit, creating a somber and reflective mood.

Lesson 1: The First System Wasn't a
Mistake – It Just Reached Its Limits

The ASDI metric

Alexandre

Sleep

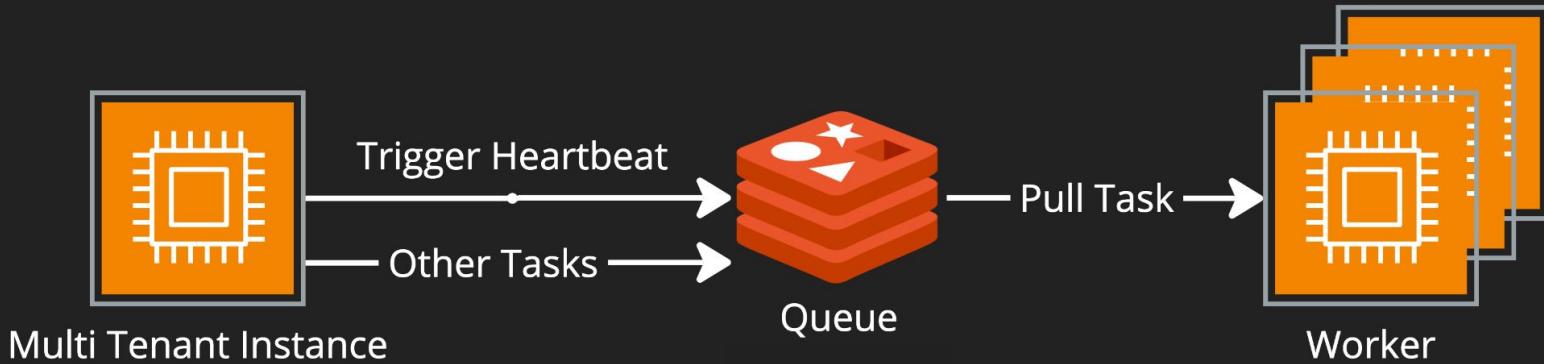
Deprivation

Index

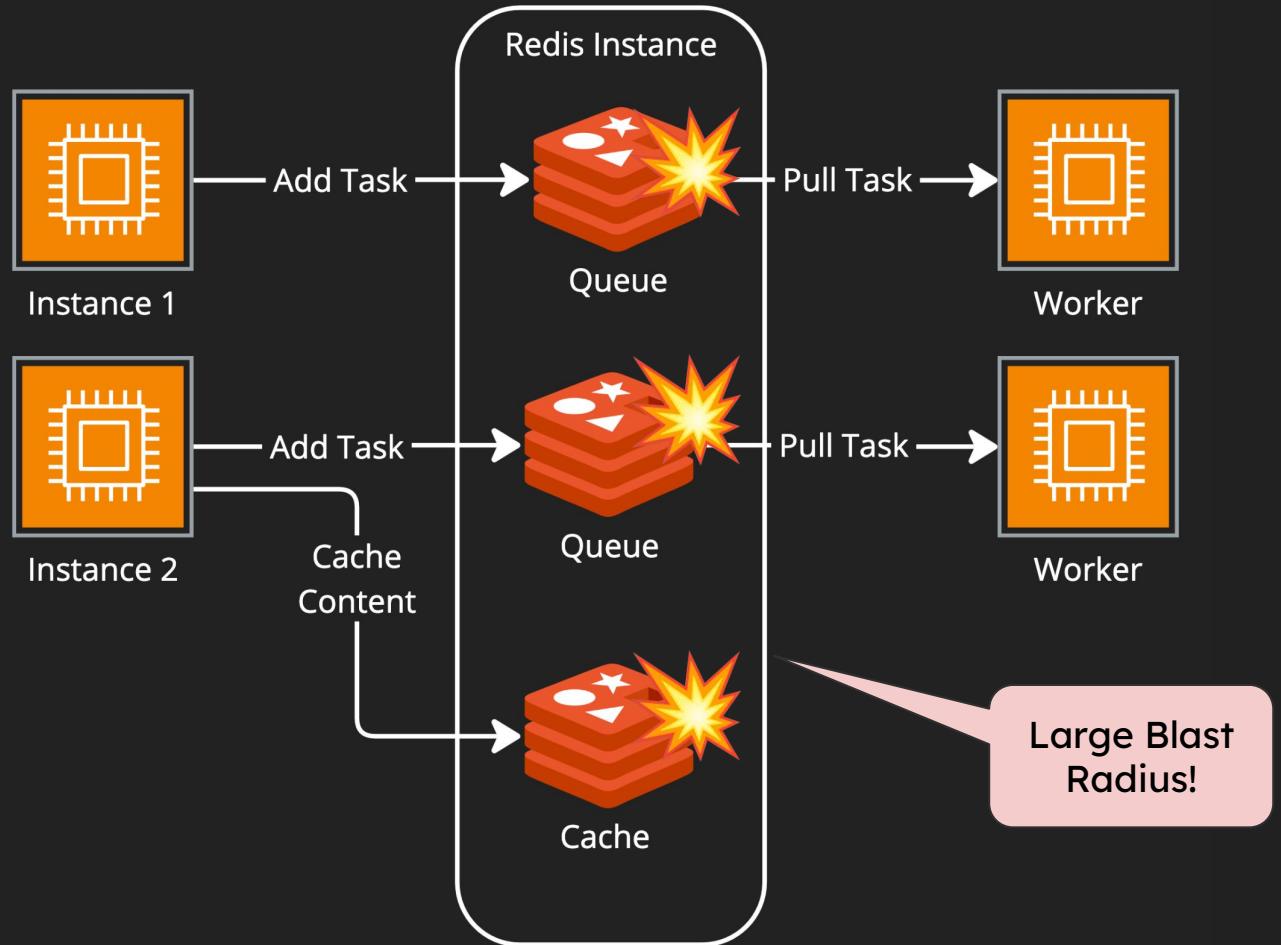
Our on-call team:

- Alexandre
- Adrien
- Arthur
- Aube
- sAm

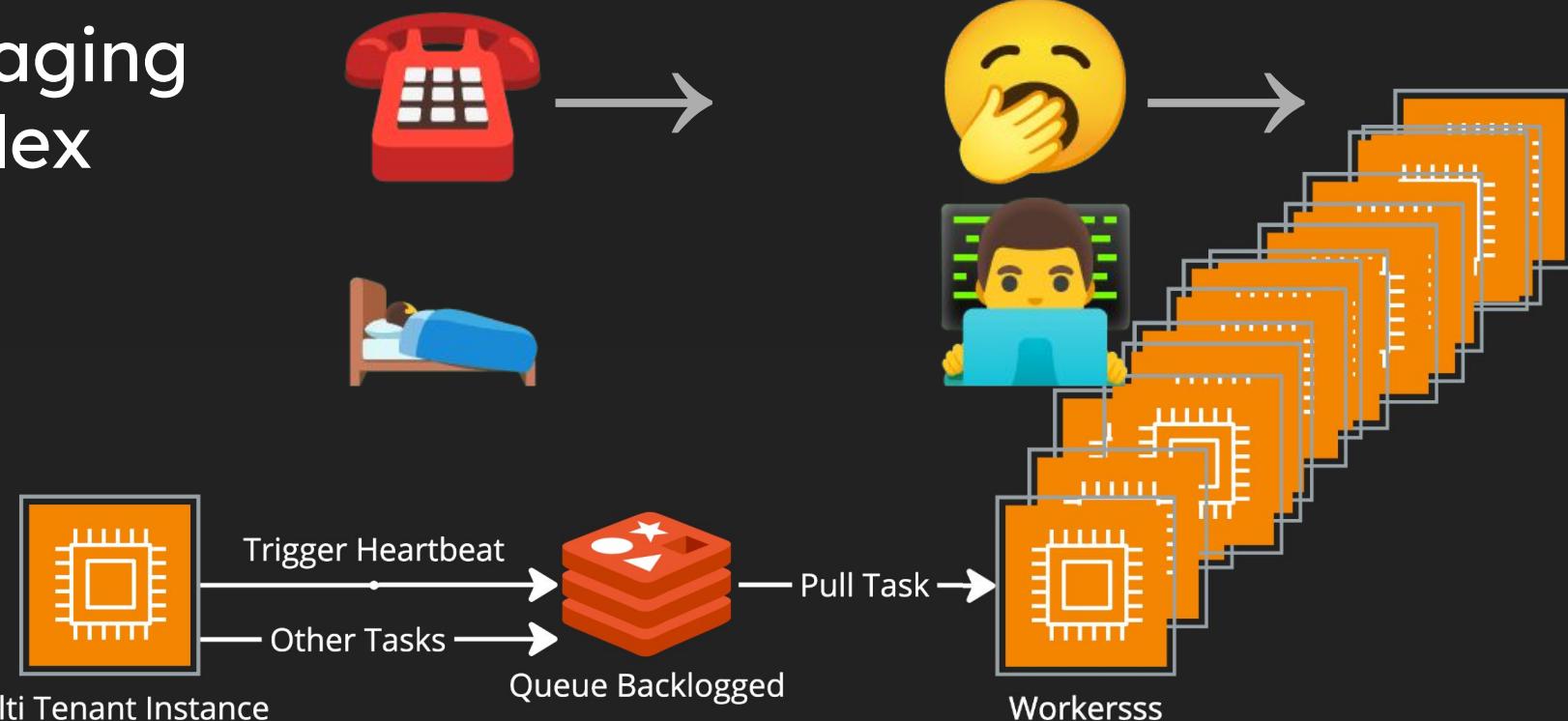
Why it was breaking?



Why it was breaking?



Paging Alex



A dark, atmospheric painting depicting a shipwreck at sea. In the center, a multi-masted sailing ship is partially submerged, its hull and masts silhouetted against a bright, cloudy sky. A small lifeboat with several people is visible in the lower right foreground, and another small boat is nearby. The ocean is depicted with dark, choppy waves.

Lesson 2: Migrations Are Risk Management, Not Just Tech

Setting Non-Functional Requirements

Availability

A backlog of tasks should not bring down the system.

Observability

It should be easy to see the number of tasks in the queue.

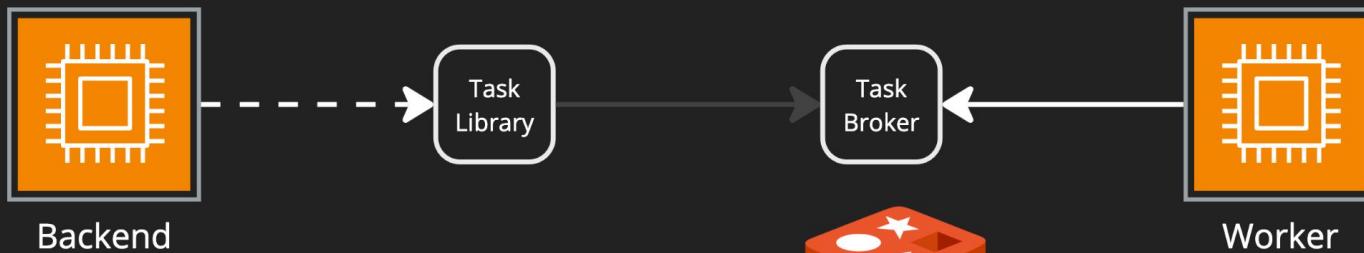
Scalability

The system must scale automatically based on task queue usage.

Maintainability

It should be easy to manage and extend.

Potential Technologies - Which fits our requirements?



Redis



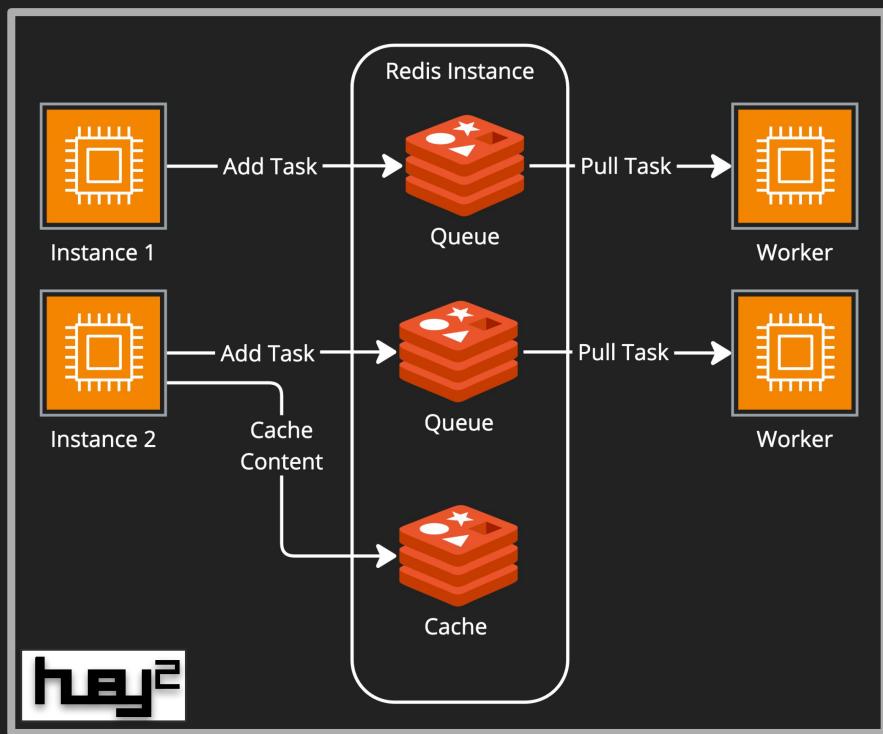
Celery



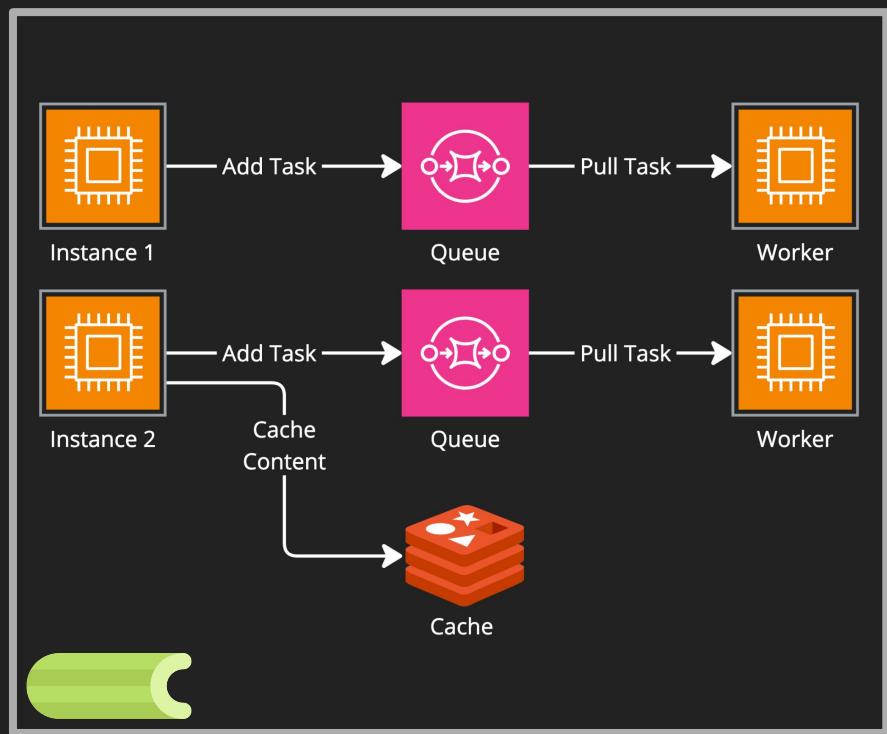
Amazon SQS



Before



After



A dramatic painting depicting a shipwreck at sea. A multi-masted sailing vessel is shown capsizing, with its hull and masts submerged in dark, churning waves. A small boat with a single occupant is visible in the lower right foreground. A flag with red, white, and blue horizontal stripes flies from the stern of the sinking ship. The sky is filled with heavy, dark clouds, suggesting a storm or sunset.

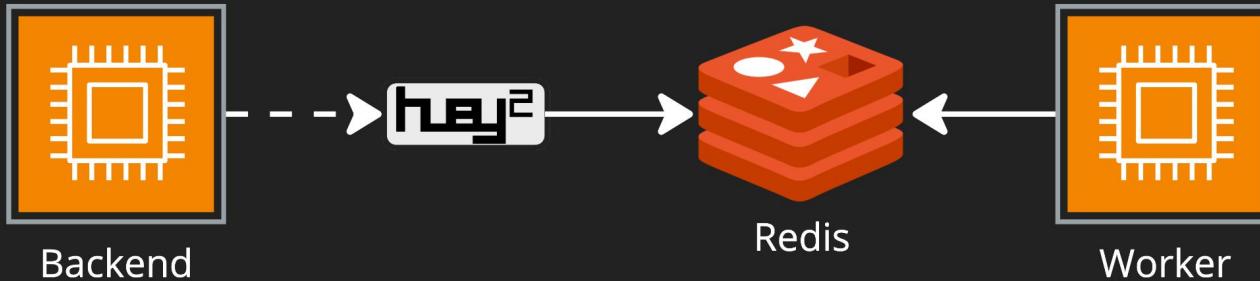
Lesson 2 (continued): Migrations Are Risk Management, Not Just Tech

A dramatic painting depicting a scene of maritime disaster. In the foreground, a large three-masted sailing ship is listing heavily to its side, its hull partially submerged in dark, churning waves. The ship's rigging and sails are visible against a backdrop of a dark, stormy sky. Behind it, another ship is seen capsizing, with its masts and debris floating in the water. The overall atmosphere is one of chaos, danger, and the power of nature.

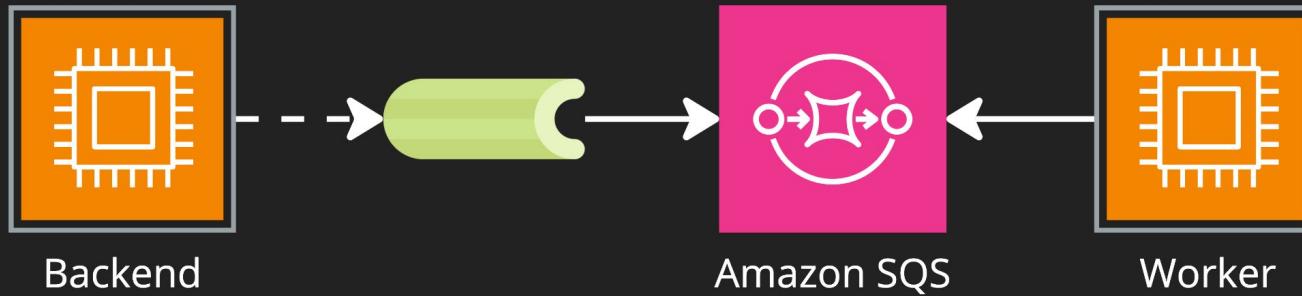
The Migration itself is a risk

How can we ensure zero downtime?

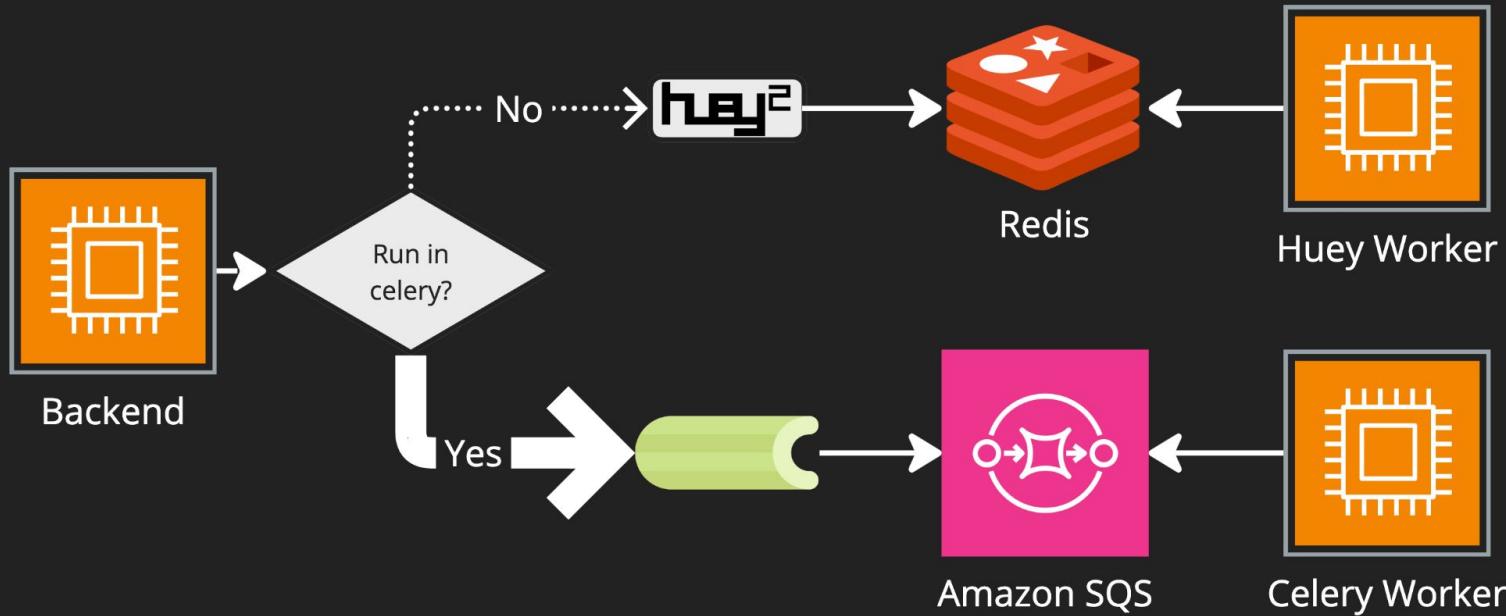
Risk Management - Big Bang Migration



Flip a switch, config, update... then



Risk Management - Gradual Rollout



A dark, atmospheric painting depicting a naval battle at sea. Numerous ships of various sizes are visible, their masts and rigging silhouetted against a bright sky. Many flags and banners are flying from the ships, though they are mostly illegible due to the low light. The water is dark and choppy, reflecting the light from the sky and the ships. The overall mood is one of chaos and intensity.

Lesson 3: Feature Flags Give You Control Over Change.

Our Flag Structure

Feature flags

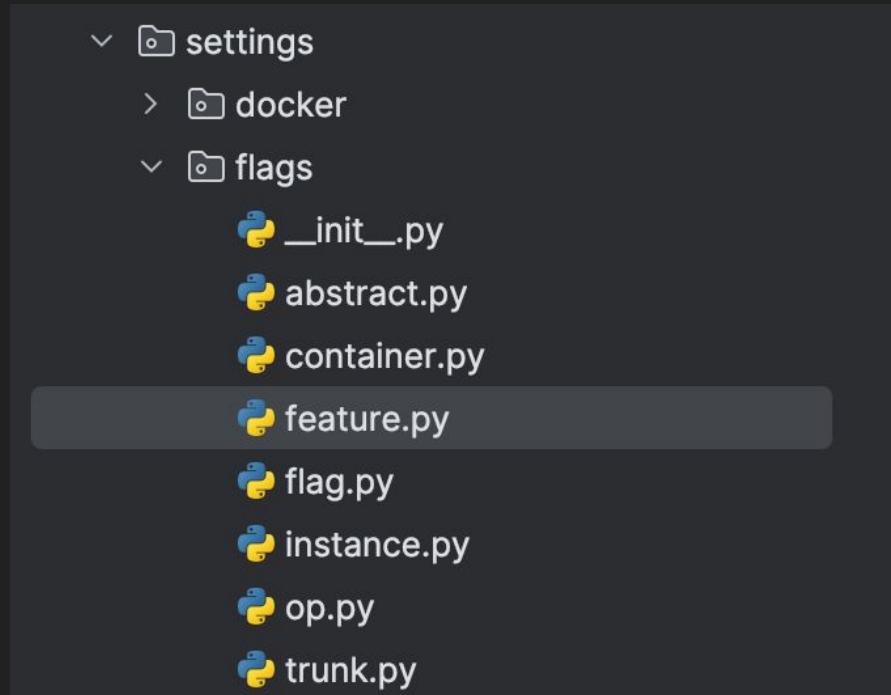
- Permanent
- Turn parts of the app on and off

Trunk flags

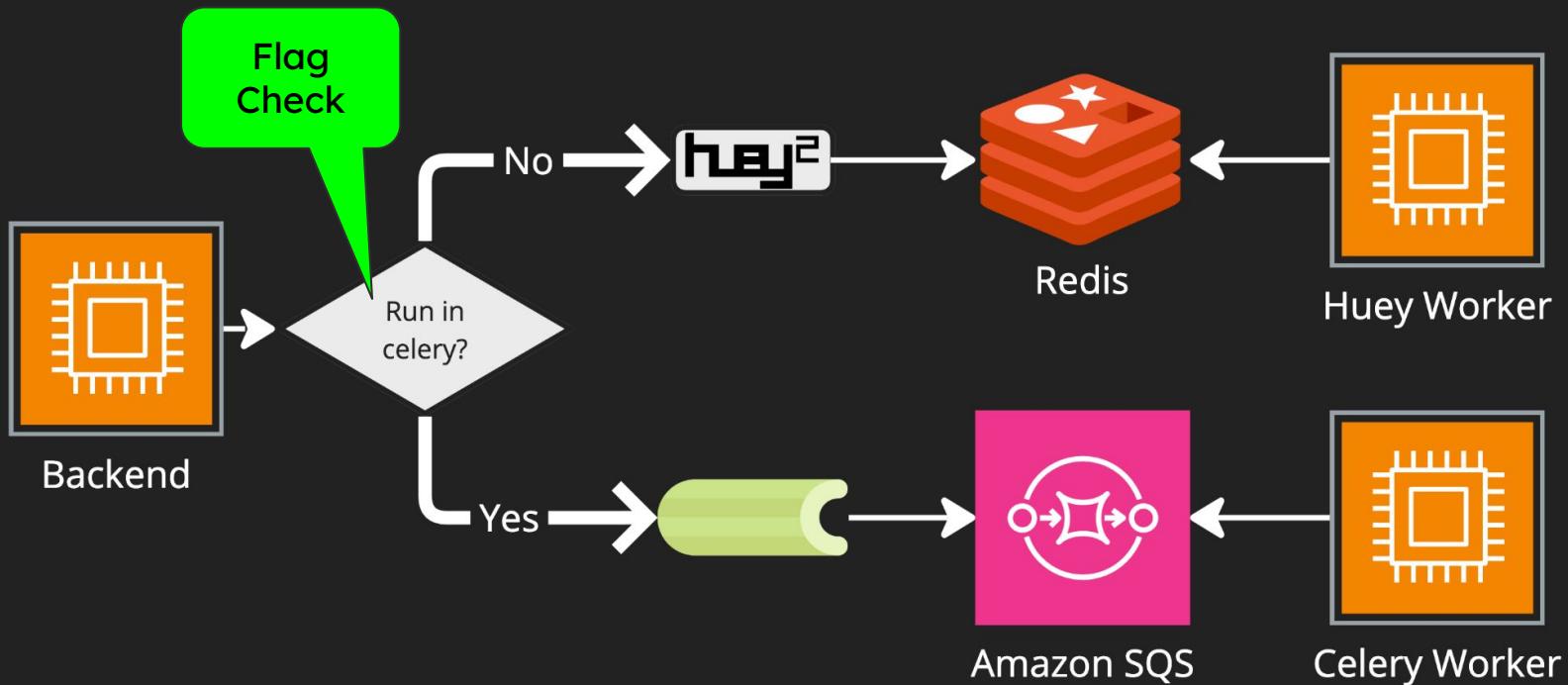
- Transient
- Deploy changes/fixes

Ops flags

- Long lived
- Configurations



Dynamic Task Routing using an Ops Flag



```
31     def should_run_in_celery(task_name):  
32         """This method checks an ops flag to determine if a task should run in celery or huey."""  
33         return OperationFlags.SHOULD_TASK_RUN_IN_CELERY.get_value_with_extra_context(  
34             task_name=task_name  
35         )
```

Flag check based
on task name

Flags / Ops - Should Task Run in Celery

Run specific tasks in Celery

+ Add rollout ↵

IF
Context kind
user

Attribute
extra.task_name

Operator
is one of

Values
task.audit.consume_event X
task.audit.handle_exported_event X
task.notify_comment X

+

Serve

True - Run in Celery

Run in Celery for
these tasks

Default rule

+ Add rollout ↵

Serve

False - Run in Huey

Default
Rule

Rollouts based on region

ca-central-1 Celery Rollout

⋮ + Add rollout ↺

Context kind Attribute Operator Values

IF user extra.task_name is one of task.audit.consume_event X -

task.audit.handle_exported... X ↻

task.notify_comment X

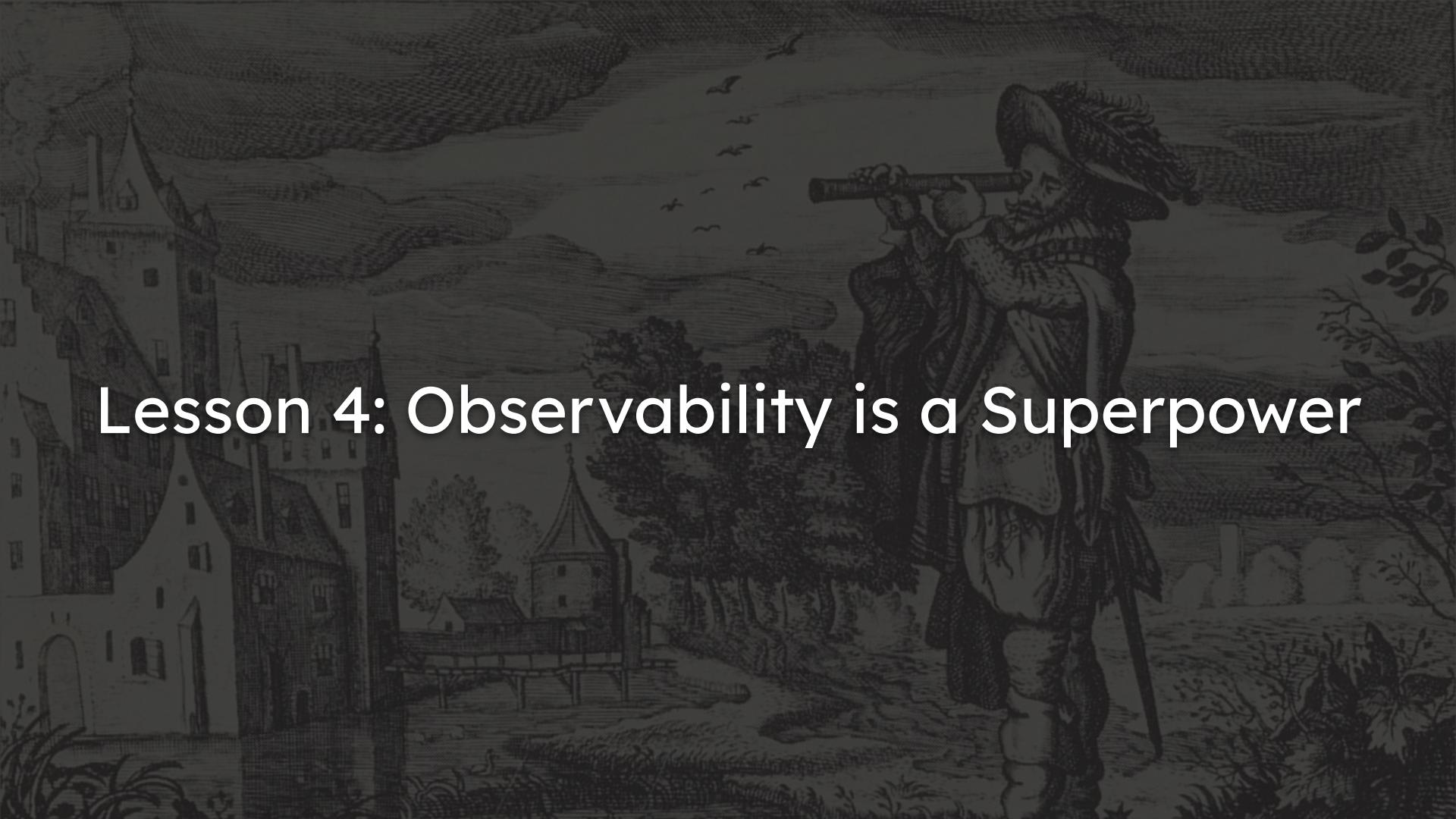
Context kind Attribute Operator Values

AND user instance_region is one of ca-central-1 X - +

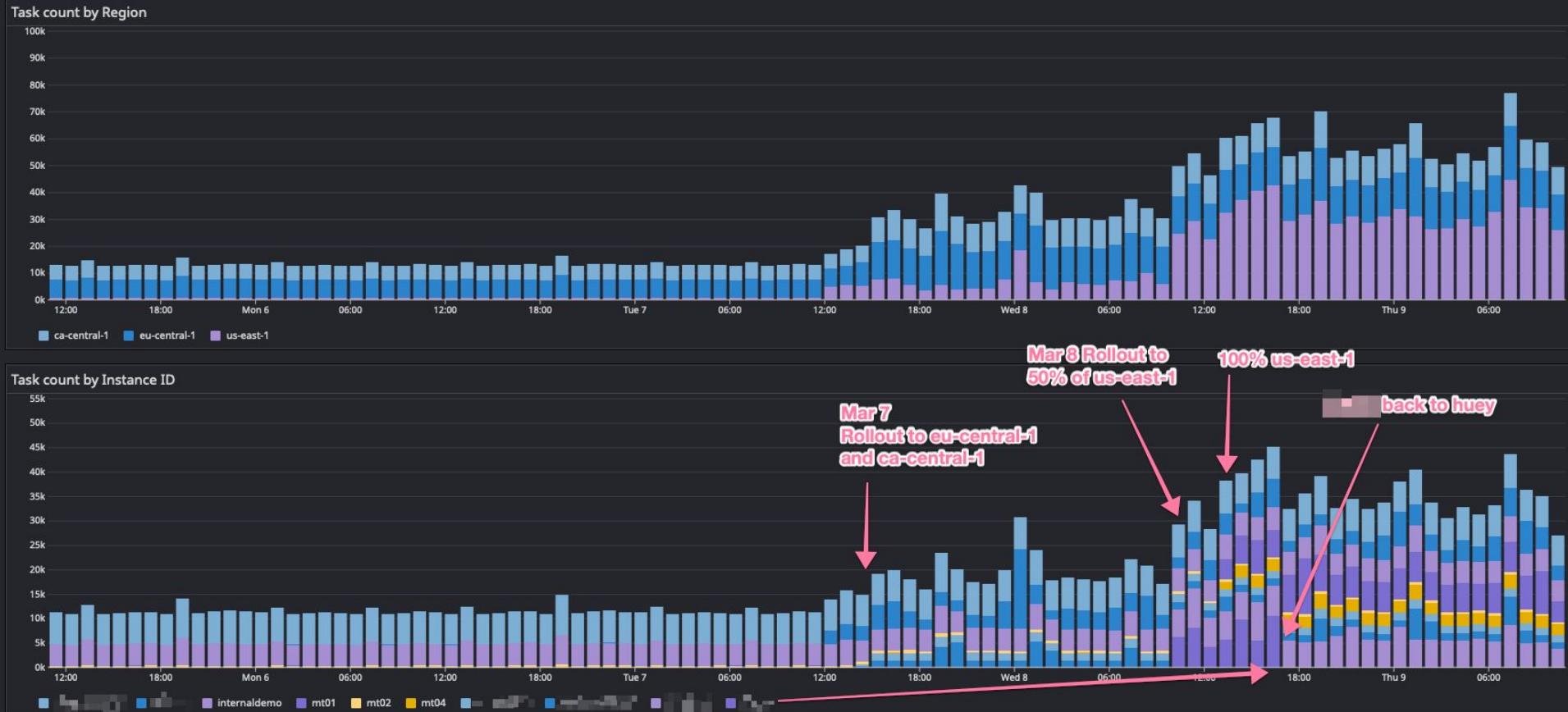
Serve True - Run in Celery

Only in this region

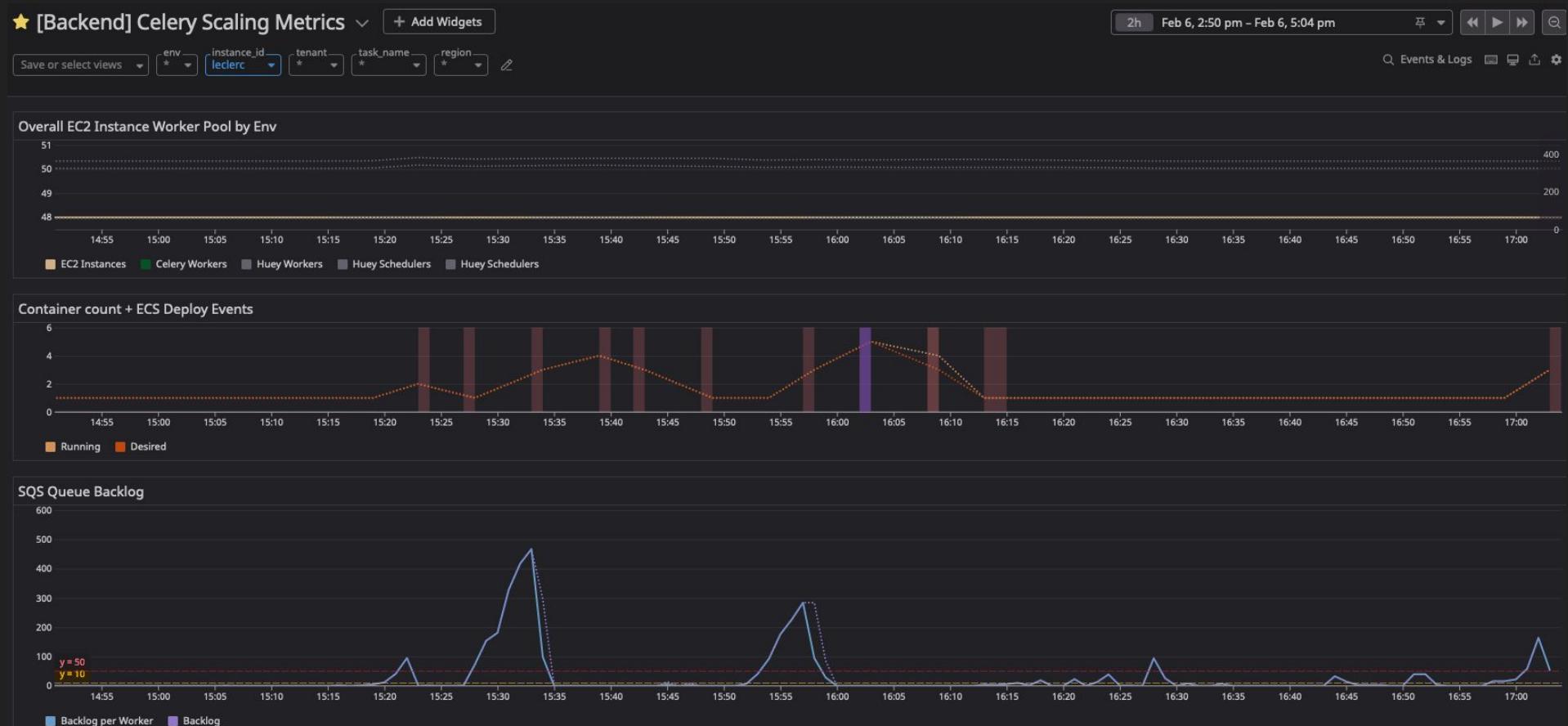
Lesson 4: Observability is a Superpower



Track Region Based Celery Task Rollout



Track Queue Backlog and Scaling Metrics



Problematic Tasks

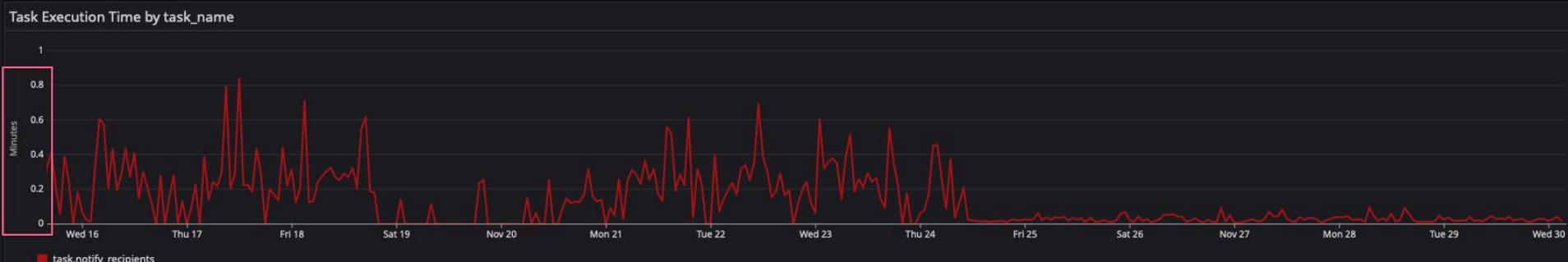
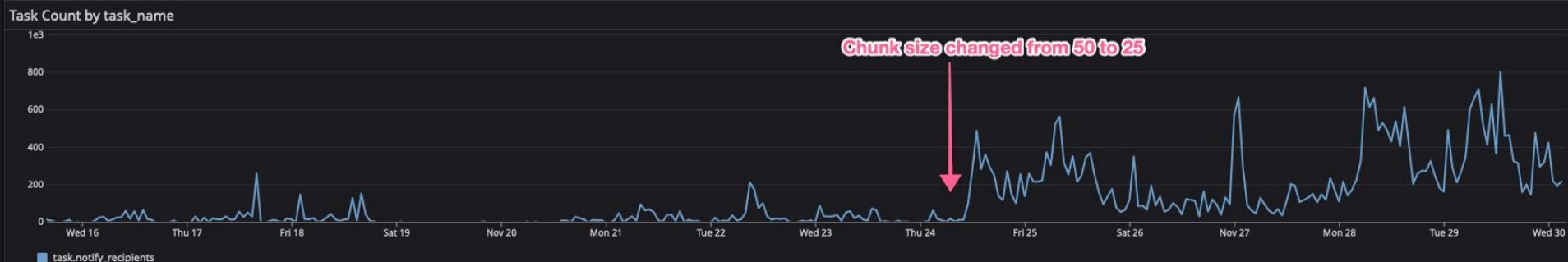
The new system brought in new constraints:

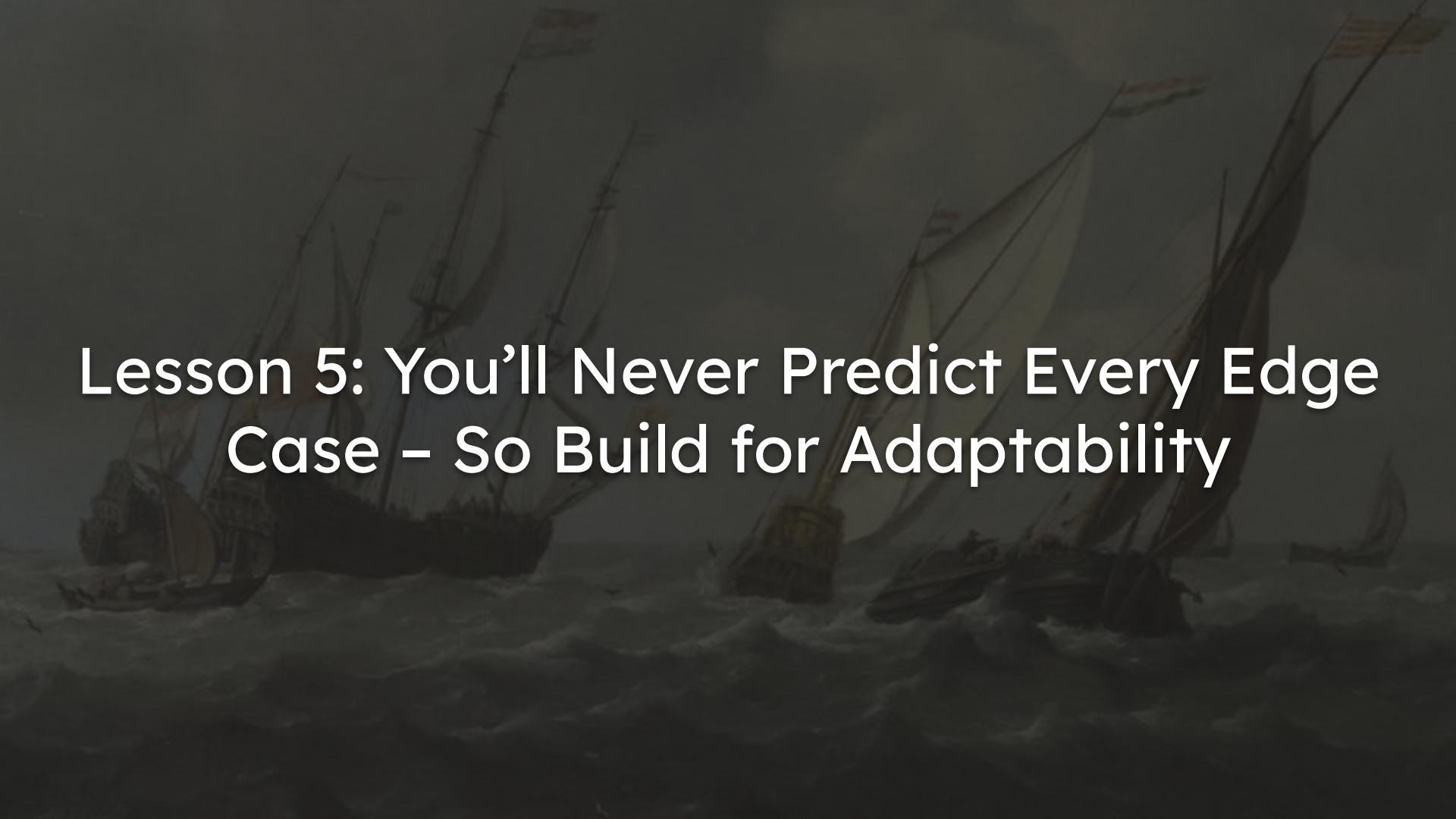
1. No time limit --> 3 minute time limit
2. Undefined max payload size --> 256 KiB max payload size

Use observability to detect issues beforehand



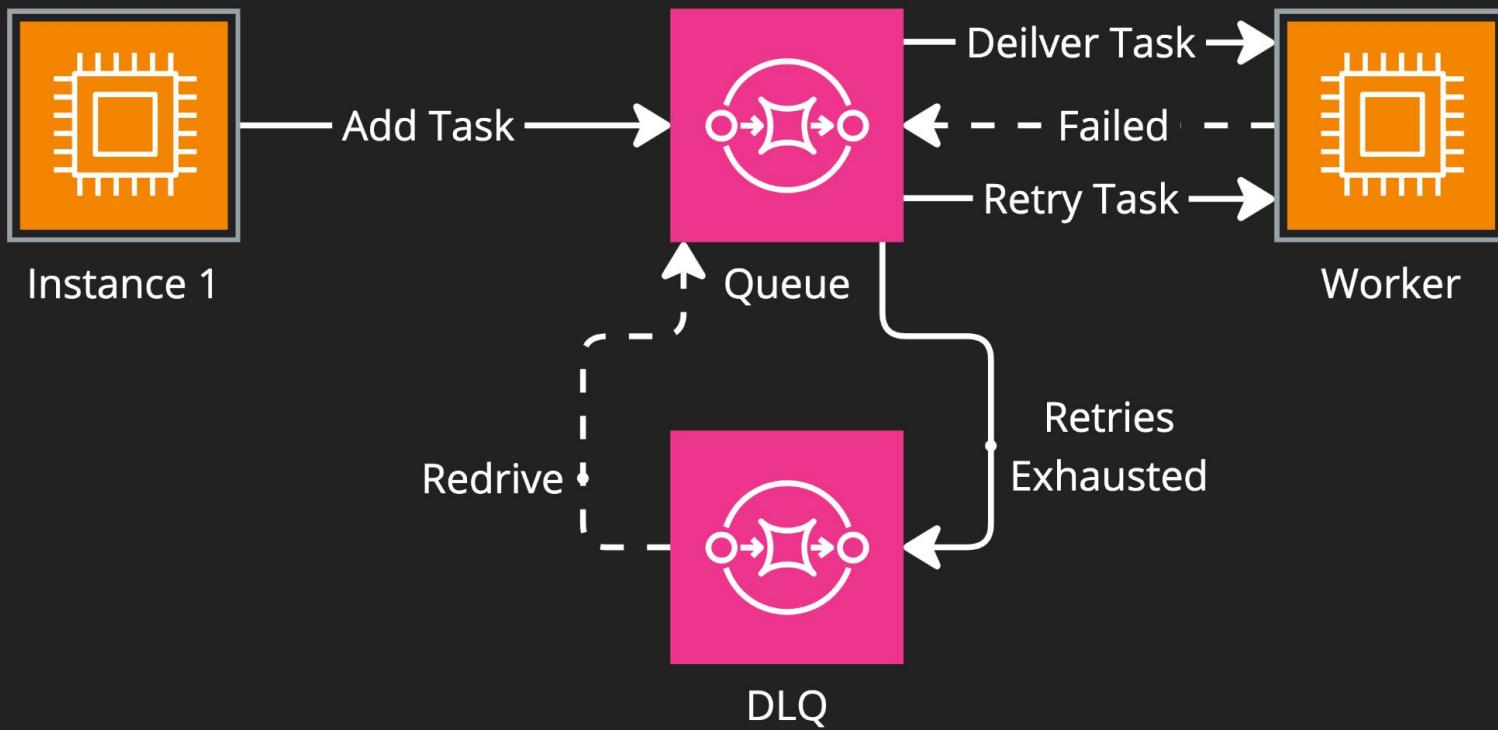
Confirming fixes to problematic tasks



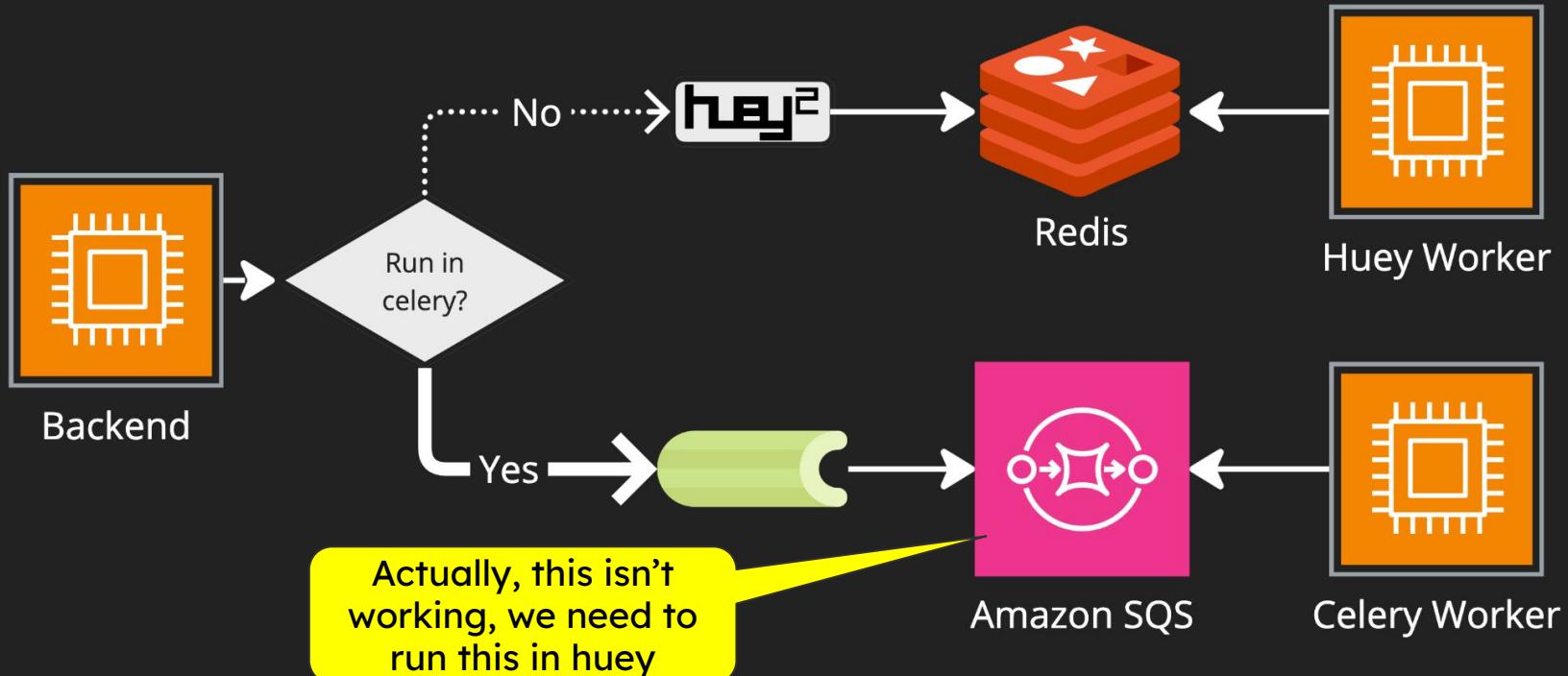
A dark, atmospheric painting of several sailboats on choppy water under a cloudy sky.

Lesson 5: You'll Never Predict Every Edge Case – So Build for Adaptability

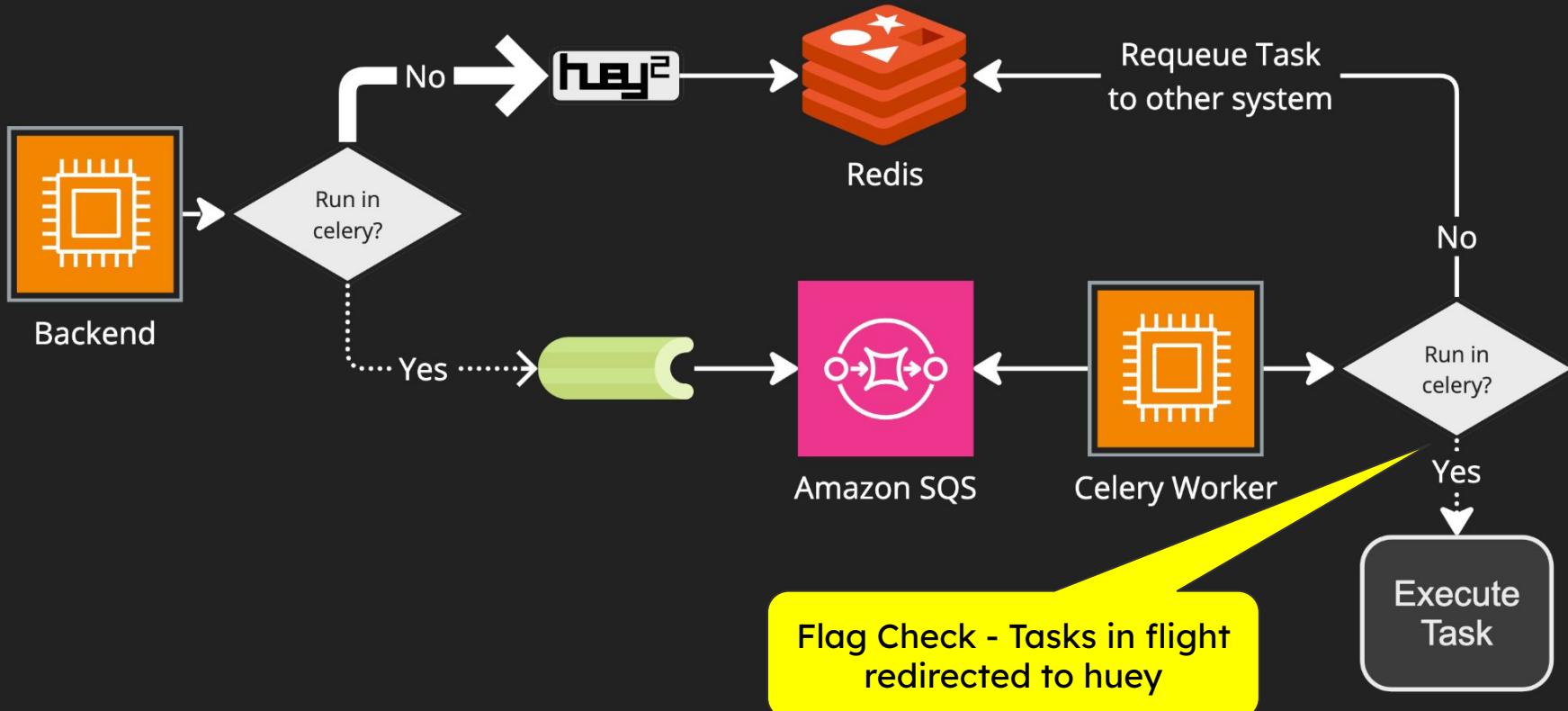
Resilient Systems



What if we need to switch back?



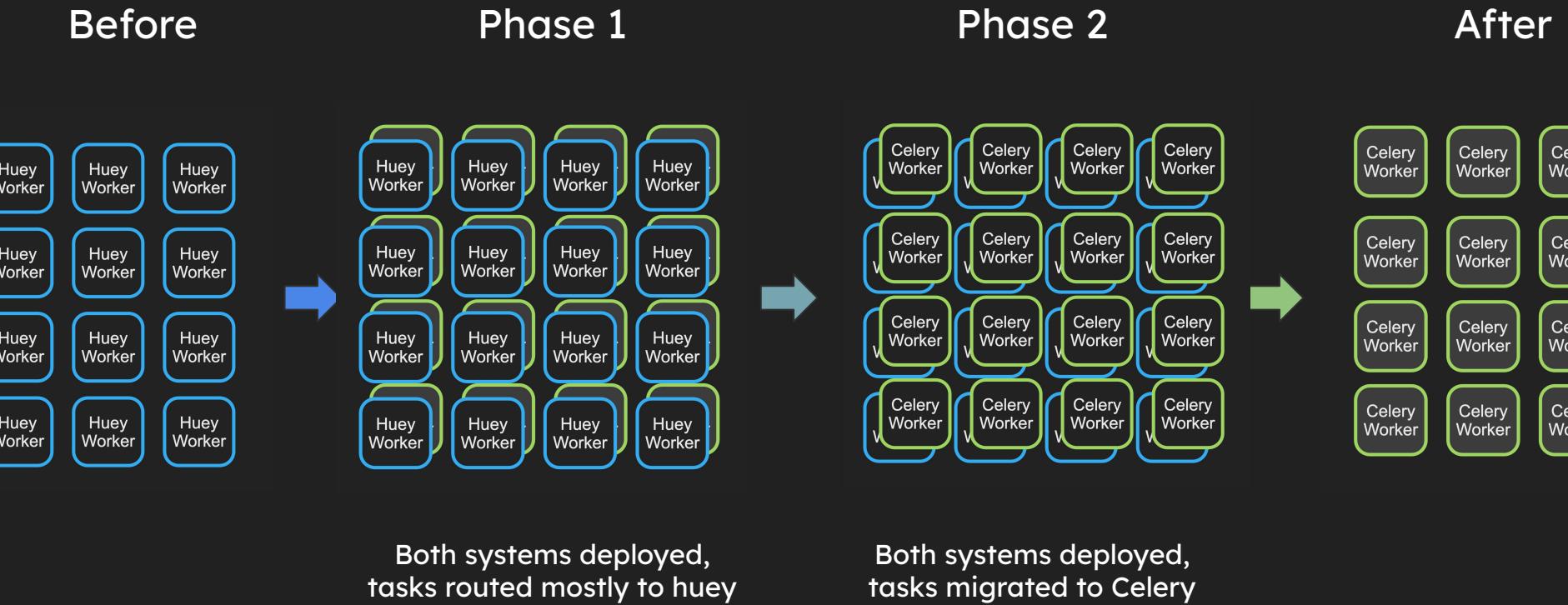
Check the flag and requeue if needed



A dark, atmospheric painting of several sailboats on choppy water under a cloudy sky.

5. Build for Adaptability: Changes to our transition strategy

Horizontal Rollout

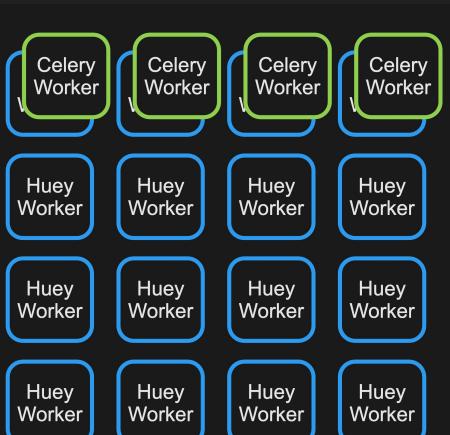


Phased Rollout

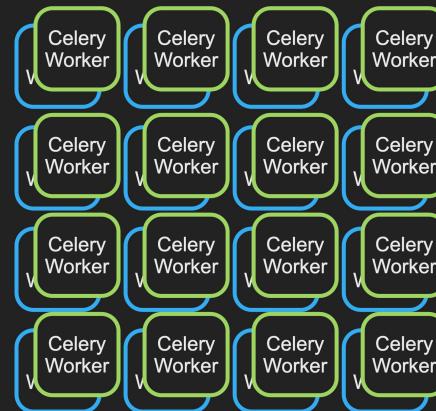
Before



Phase 1



Phase 2



After



Vertical Rollout to a subset
of large instances

Horizontal Rollout to
Remaining instances

A dark, atmospheric painting depicting a harbor or port scene at night or in low light. Numerous sailboats and small boats are scattered across the water, their masts and sails silhouetted against a hazy sky. The overall mood is mysterious and somewhat somber.

Lesson 6: Documentation Helps You Make (and Defend) Decisions

RFC / Tech Designs

Technical design - Task Queue Replacement



Table of contents

Introduction:

Definitions:

Current Huey + Redis Task Queue Architecture

Problems and Top 4 Significant Non-Functional Requirements:

Possible Solutions and Choice:

Choice: Celery + SQS:

Celery + SQS Task Queue Architecture

List of Risks

Requirements

Infrastructure

Cost Considerations

Migration Plan

Task Queue Execution Flow Chart

Sequence Diagrams

Celery Tasks and Schedule Tasks Sequence Diagram

Celery Periodic Task Sequence Diagram

Huey Task Queue Sequence Diagram

Questions to answer

Introduction:

The backend of the Poka application uses as Task Queue System to process jobs asynchronously that exist outside the normal client/request lifecycle.

This document describes the plans to replace the current Task Queue System that uses Huey and Redis with a Task Queue System using Celery and SQS.

For a more detailed dive into the research done to create this design doc, refer to the following document: [Task Queue Replacement Research Doc](#)

Definitions:

For the purposes of this document and clear communication, we use the terms listed below. The terms included in parentheses are alternative names, but for this document we aim to use the main names.

Task Queue System: Describes the whole entity of a task queue, including the Task Queue Library, the Broker and the Result Store.

Task Queue Library: The library or framework used to orchestrate the task queue system (e.g. Huey and Celery)

Broker (Message Transport): Where we store tasks that are queued, and where the workers pull tasks that have been queued. (e.g. Redis and SQS)

Result Store (Backend): Where we store the state of tasks, as well as the return values of the tasks.

Worker (Consumer): Describes the workers that consume tasks from the task queue.

ADRs: Architectural Decision Records

1. Capture the context of a decision
2. Provide a decision snapshot
3. Help teams move forward, even without full consensus
4. Save time in the long run
5. Help prepare talks

Accepted ADRs

Number	Title	Author
ADR-000	Write Architecture Decision Records	Marc-Antoine Aubé
ADR-005	Celery-SQS retry configuration	Alexandre Beaudoin
ADR-006	Web container auto-scaling based on Little's Law	Alain Metivier
ADR-007	Worker container scaling metrics	Edmund Lam
ADR-008	Run all celery tasks in standard SQS queues	Arthur Sylvestre, Edmund Lam
ADR-009	Strategy for handling Celery tasks that exceed time limit	Arthur Sylvestre, Edmund Lam
ADR-062	Exclude soft-deleted models when querying through tables	Marc-Antoine Aubé
ADR-063	Ensure that Unique Fields Have Enough Room For the Anti-Collision Suffix	Benoit Bernard
ADR-064	Use max score fusion for PostgreSQL hybrid search strategy	Edmund Lam
ADR-065	Increase SCIM User Groups members soft limit	Maria-Luisa Gonçalves

A dramatic painting depicting a naval battle at sea. In the center, a large three-masted sailing ship with its sails partially unfurled is engaged in combat. Its hull is dark wood, and it features a prominent stern castle. Several crew members are visible on deck. The ship is surrounded by smoke and fire, with a trail of smoke extending from its stern. To the left, another ship with yellow sails is partially visible. To the right, a larger, more ornate ship with multiple decks and a golden-yellow hull is also involved in the fight. The background is filled with dark, billowing smoke and fire, creating a sense of chaos and destruction. The overall composition is dynamic, capturing a moment of intense conflict.

6 key lessons

Wrap up our journey

The First System wasn't a Mistake - it reached its limits

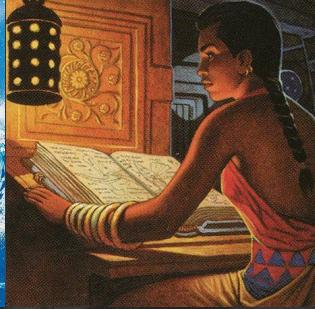
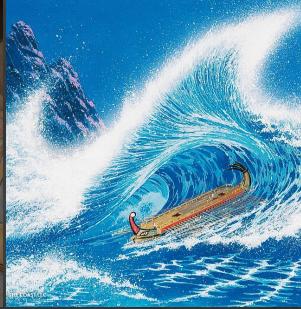
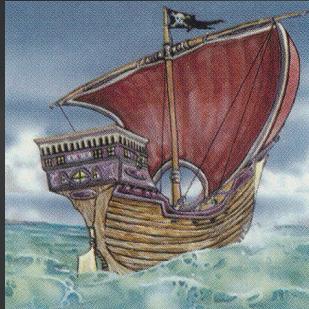
Migrations are Risk Management, not just a Tech upgrade

Feature Flags Give you Control over Change

Observability is a Superpower

You Can't Predict every Edge Case - So build for Adaptability

Documentation helps you make (and defend) decisions



The old ship served us well

Navigating choppy waters takes careful planning

Adjust the sails to steer the winds

A captain's best friend is a good map and compass

Prepare for rogue waves

A captain's log keeps the course accountable

Thank You!

Zero Downtime Migration of Our Distributed Task Queue System

Edmund Lam
Confoo 2025