# If the Queue Fits

### A Novel Task

# Let me guess: You need to make HTTP requests to some flaky provider to record numerous points of data for many different customers.



#### All at Once

#### Node supports concurrency, so why not?

await Promise.all(allTheStuff);

#### "aLI aT oNcE"

#### Rate limits, resource contention, errors, chaos.



## One at a Time

#### Okay then, so we'll go slower.

```
for (const thingToDo of thingsToDo) {
   await thingToDo();
}
```

#### "oNe aT A tImE"

Slow as molasses.



# What Do We Want?

- Concurrency / parallelism
- Crash resilience
- Error handling and retries

# RabbitMQ

- Networked message-broker
- Easily solves concurrency and crash concerns
- Number of consumers ~ resource usage



# RestOfTheRabbitMQ

- Great for generic messaging
- Many useful features
- Job queue requires more work



## **RI** RetailLink Services

- Used amplib directly
- Inconsistent error handling
- Hard-to-follow retry logic



# Queueball Begins

- RabbitMQ = good, lots of boilerplate = bad
- Consistency can help avoid previous issues
- Stephen vendored it into analytics-scheduler

### Example

```
const config = {
  rabbitmqUrl: 'amqp://guest:guest@localhost:5672',
  name: 'queueball-test',
};
const producer = new Producer(config);
await producer.start();
const consumer = new Consumer(config);
consumer.processJob = async ({ payload }) => {
  console.log(payload.message);
};
await consumer.start();
producer.enqueue({ message: 'Hello world!' });
```

### Punched in the Face

- Queued jobs are a happy path
- Things rarely go as planned
- Additional techniques can help

Lt Create metric



#### Resource Locks

- Lock resource when job starts
- Per user, or even whole job type
- Re-enqueue conflicting jobs
- Redis or Postgres can work

```
await this.sessionsRepo.manager.transaction(async (manager) => {
    const lockKey = this.generateLockKey(`supplier-session-${supplierId}`);
    await manager.query('SELECT pg_advisory_xact_lock($1)', [lockKey]);
    // ...
```

# Job Striping

Space resource contenders apart
AAABBBCCC -> ABCABCABC

```
const jobPayloadGroups: JobPayload[][] = [];
```

```
for (const supplier of suppliers) {
   const jobPayloads: JobPayload[] = [];
   jobPayloads.push(...someSupplierJobPayloads);
   jobPayloads.push(...moreSupplierJobPayloads);
   jobPayloadGroups.push(jobPayloads);
}
```

```
return zipArrays(...jobPayloadGroups);
```

# Structure Tips

- Job generation can be a job
- Prefer smaller, simpler jobs
- Have contextual error messages

# Tooling Tips

- Use Kube for scheduling
- Add job params to Sentry errors
- Use trace ID when searching logs

Job		(3) 6:35:00 AM - 6:43:00 AM Q Search all fields
		1 jsonPayload.span_id="2a38f1dd1341ae9a"
deductionId	6723	
		😑 🛛 Log fields 🛛 🥪 Histogram
id	5ceec178-2ee5-4c0f-99a7-bb69aedcddd9	Histogram
kind scrape-		
	scrape-purchase-order-pdf	6:35 Feb 15, 6:35:00 AM 6:36 AM 6:37 AM 6:38 AM 6:39 AM 6:4
		Query results 7 log entries
priority	low	SEVERITY TIMESTAMP 1 CST - SUMMARY 2 EDIT
		(i) To view older entries: Extend time by: 1 minute V Edit time
purchaseOrderNumber	4PX46K3C	> i 2023-02-15 06:36:58.254 CST [Job processing: 08460683-c329-489a-a3ab-999b262aa859 ("_p":"F", "hostname":"dn-amazon-api-work
		> i 2023-02-15 06:36:58.256 CST Getting session lock {"_p":"F", "hostname":"dn-amazon-api-worker-6f55cc9cbd-dt87d", "kubernete
		> i 2023-02-15 06:36:58.257 CST Got session lock {"_p":"F", "hostname":"dn-amazon-api-worker-6f55cc9cbd-dt07d", "kubernetes":{
supplierId	22	> i 2023-02-15 06:36:58.258 CST Preparing browser {"_p":"F", "hostname":"dn-amazon-api-worker-6f55cc9cbd-dt87d", "jobId":"0846
		> i 2023-02-15 06:36:58.831 CST Setting user agent {"_p":"F", "hostname":"dn-amazon-api-worker-6f55cc9cbd-dt87d", "jobId":"084
		> 😃 2023-02-15 06:37:00.754 CST 🛛 Job failed: 08460603-c329-489a-a3ab-999b262aa859 Redirected to account selection page 🔞 (
		> i 2023-02-15 06:37:00.755 CST Job processed: 08460683-c329-489a-a3ab-999b262aa859 {"_p":"F", "hostname":"dn-amazon-api-worke

# My Ask

If you use Queueball, let me know how I can make it better.