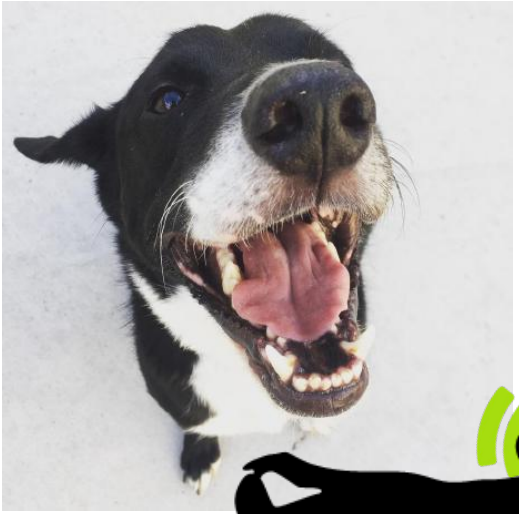
Batch Encoding Audio with Kubernetes

Leigh Capili
Infrastructure Engineer, Beatport

John Slivka
Infrastructure Engineer, Beatport



Leigh Capili

- Infrastructure Engineer
- SIG Cluster-Lifecycle

- @capileigh
- github: stealthybox

John Slivka

- Infrastructure Engineer
- github: jslivka
- @jslivka

GENRES ▾ TRACKS RELEASES CHARTS STEMS SOUNDS SEARCH... My Beatport

NEW ON BEATPORT

MACEO PLEX
MUTANT ROMANCE
MPLX

EXCLUSIVE

MPLX
01

Mutant Romance a
Maceo Plex | MPLX
$4.98

DJ CHARTS

Maceo Plex

Jack Back

Amelie Lens

beatport top10

1 **Hanging Tree** Original Mix
Michael Bibi
Repopulate Mars

2 **Breathe** Original Mix
CamelPhat, Jem Cooke, Cristoph
Pryda Presents

3 **Along Came Polly** Original Mix
Rebuke
Hot Creations

4 **Praise You** Purple Disco Machine Extended Remix
Purple Disco Machine, Fatboy Slim
Defected

5 **Do It Like This** Extended Mix
Biscits
SOLOTOKO

6 **Push Pull** Club Mix

NEW RELEASES

Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph

125 BPM / A min / 7:13

$1.99

https://www.beatport.com/release/mutant-romance/2448894

GENRES ▾ TRACKS RELEASES CHARTS STEMS SOUNDS SEARCH... My Beatport

NEW ON BEATPORT

MACEO PLEX
MUTANT ROMANCE
MPLX

EXCLUSIVE

MPLX
01

Mutant Romance
Maceo Plex | MPLX
$4.98

DJ CHARTS

Amelie Lens

beatport top10

Hanging Tree Original Mix
Michael Bibi
Repopulate Mars

Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph
Pryda Presents

Along Came Polly Original Mix
Rebuke
Hot Creations

Praise You Purple Disco Machine Extended Remix
Purple Disco Machine, Fatboy Slim
Defected

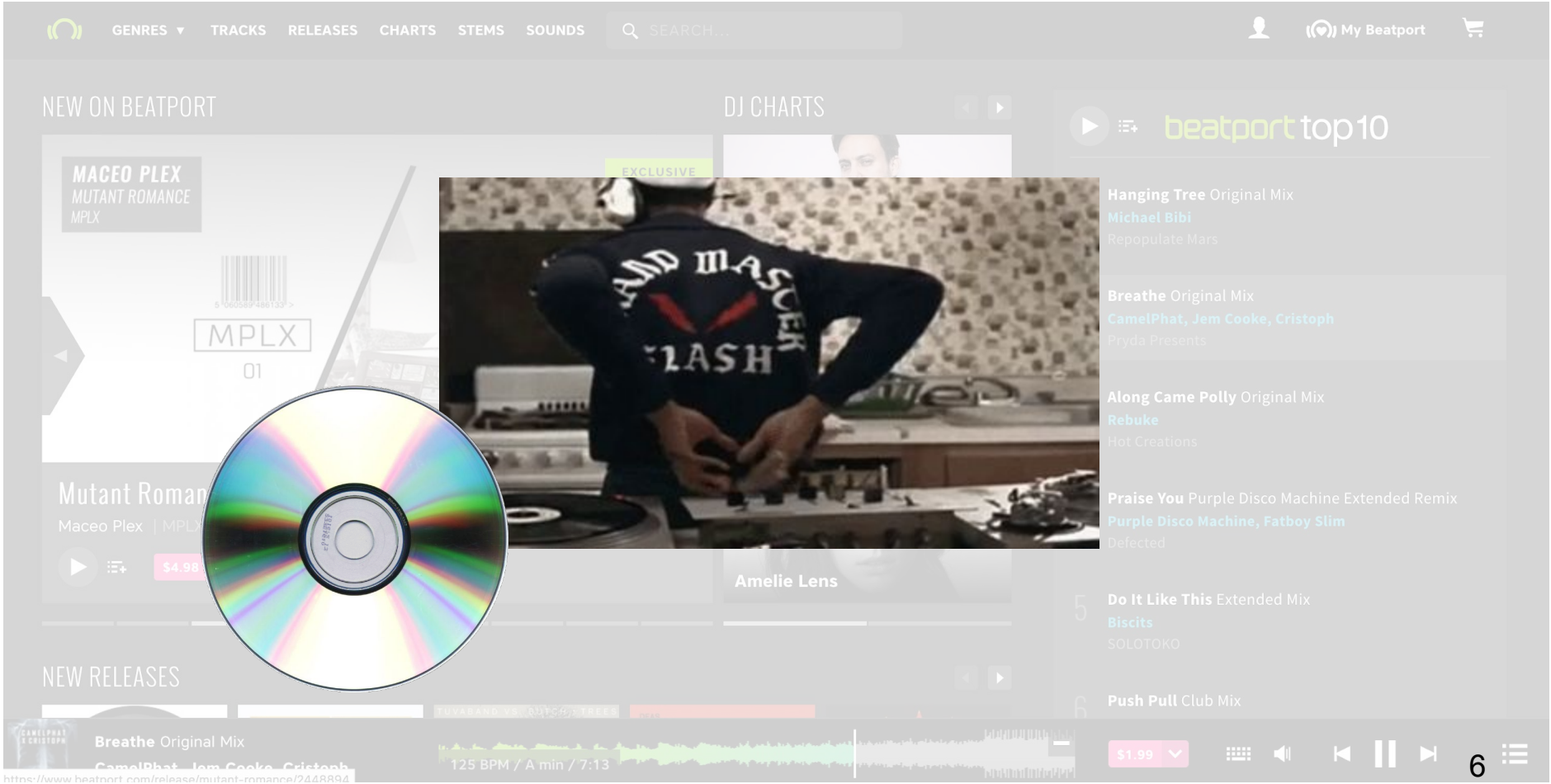
5 **Do It Like This** Extended Mix
Biscits
SOLOTOKO

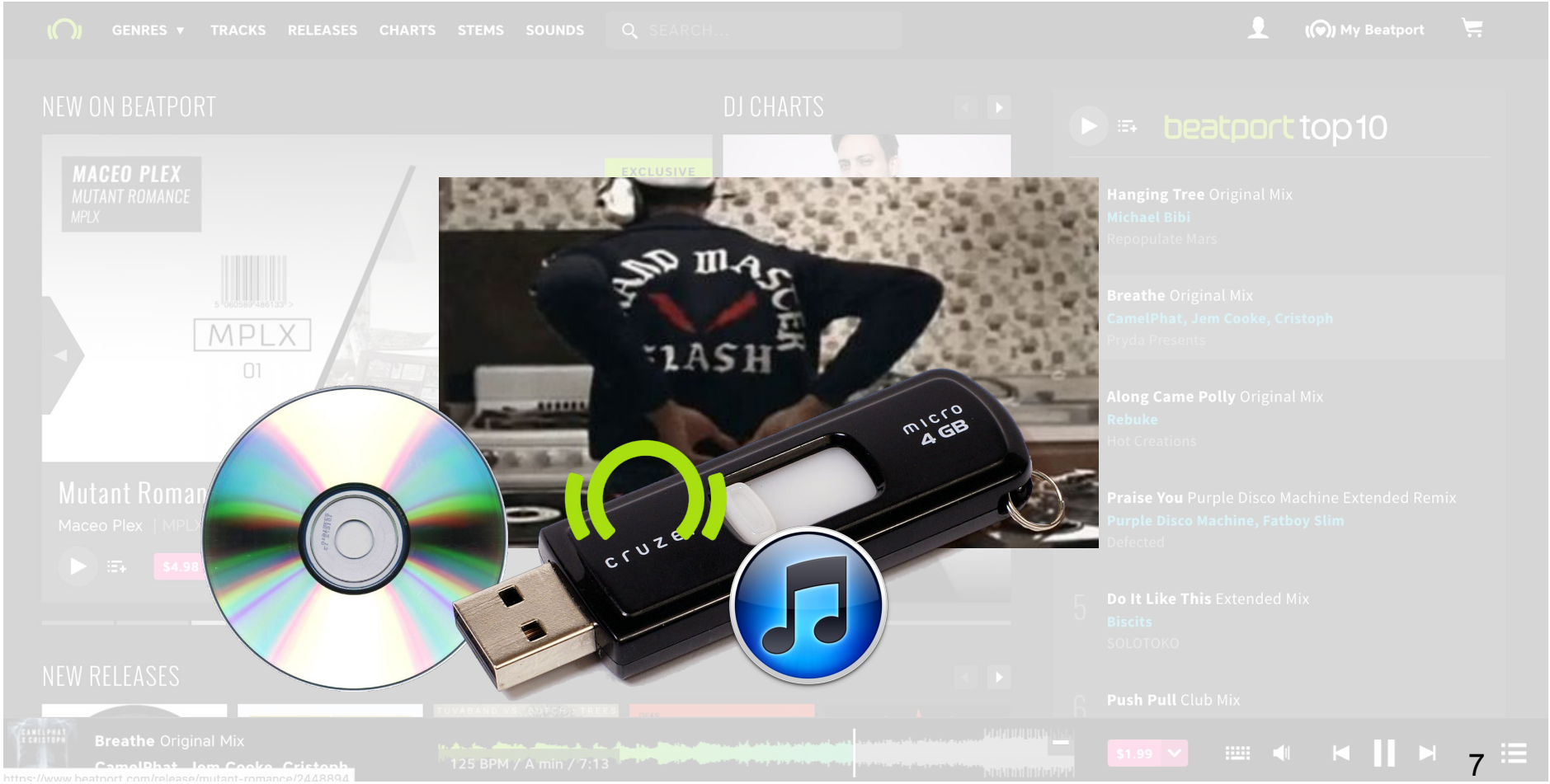
6 **Push Pull** Club Mix

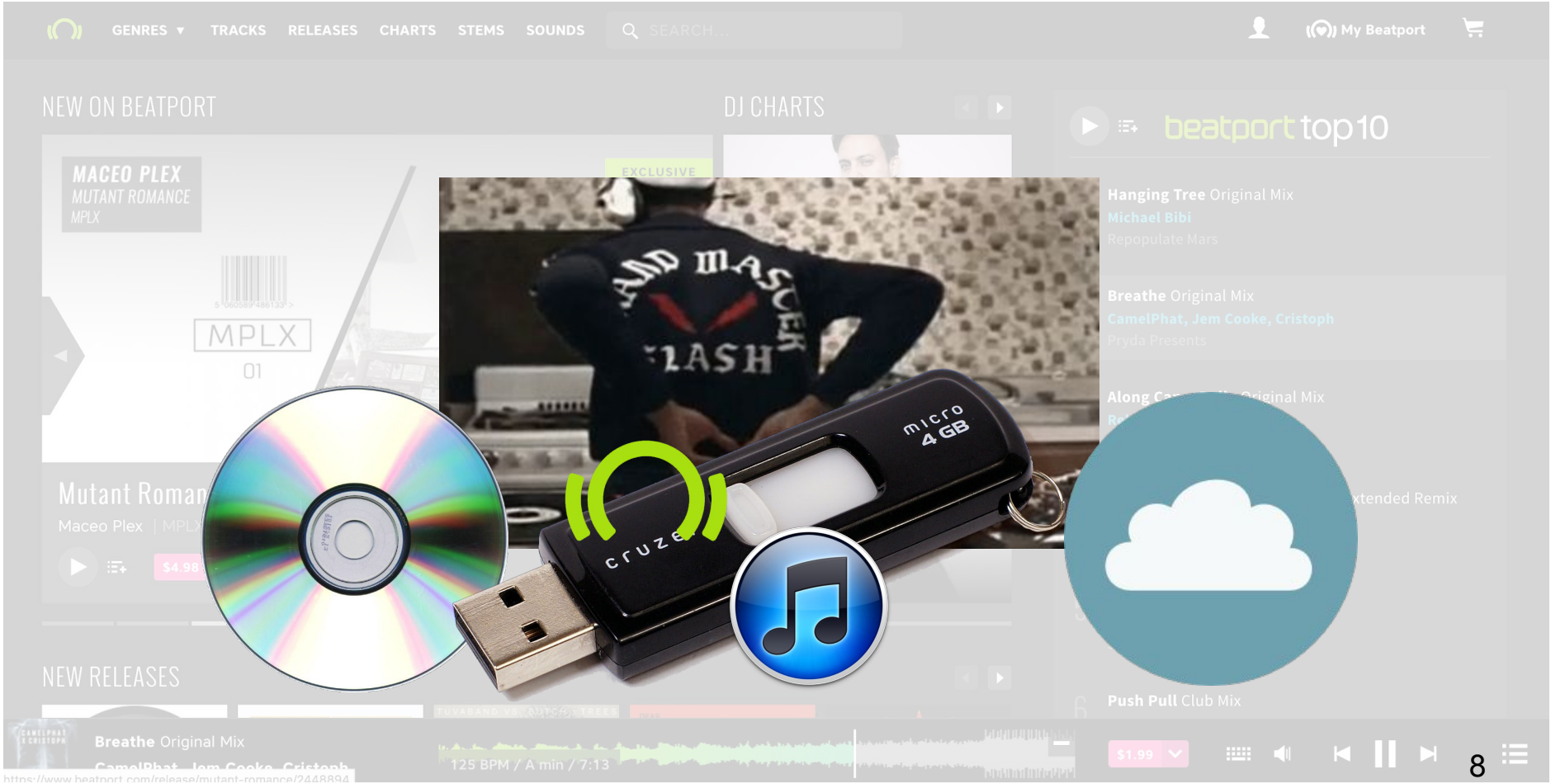
NEW RELEASES

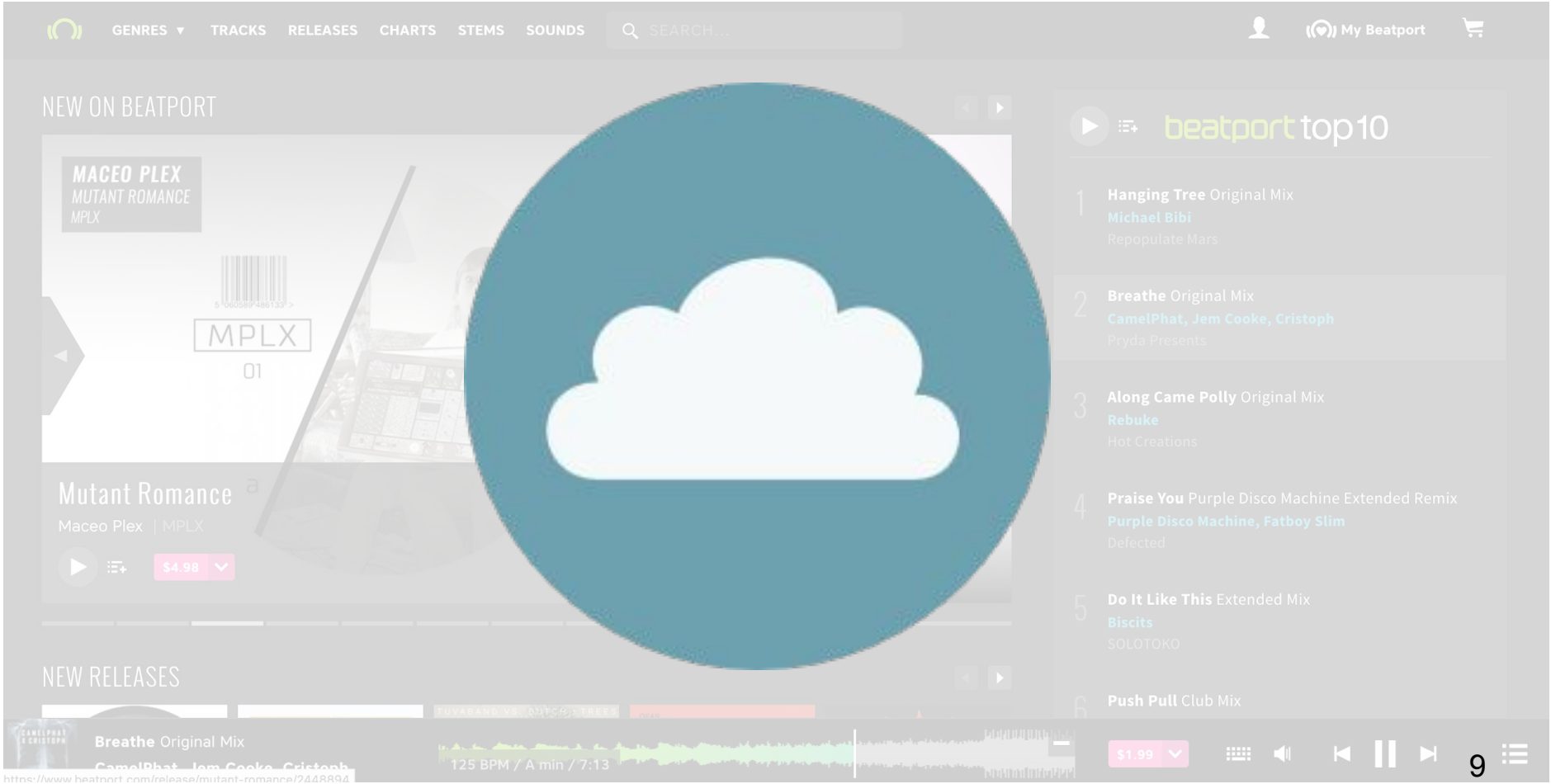
Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph
125 BPM / A min / 7:13
$1.99

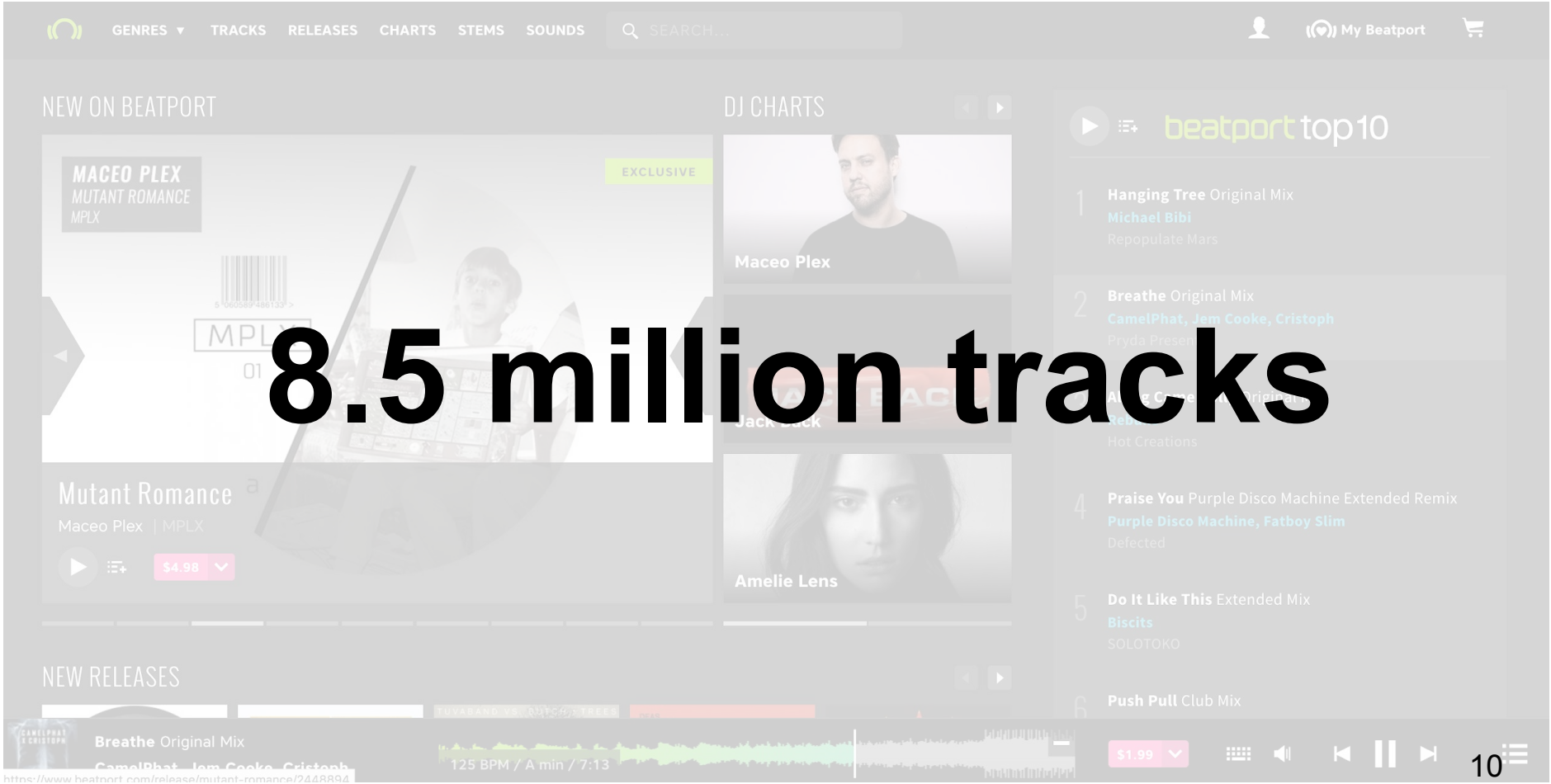
https://www.beatport.com/release/mutant-romance/2448894


GENRES
TRACKS
RELEASES
CHARTS
STEMS
SOUNDS
SEARCH...
My Beatport
NEW ON BEATPORT
DJ CHARTS
MACEO PLEX
MUTANT ROMANCE
MPLX
MPLX
01
EXCLUSIVE
Mutant Roman
Maceo Plex
$4.98
Amelie Lens
beatport top10
Hanging Tree Original Mix
Michael Bibi
Repopulate Mars
Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph
Pryda Presents
Along Came Polly Original Mix
Rebuke
Hot Creations
Praise You Purple Disco Machine Extended Remix
Purple Disco Machine, Fatboy Slim
Defected
5
Do It Like This Extended Mix
Biscits
SOLOTOKO
6
Push Pull Club Mix
NEW RELEASES
Breathe Original Mix
125 BPM / A min / 7:13
$1.99


GENRES
TRACKS
RELEASES
CHARTS
STEMS
SOUNDS
SEARCH...
My Beatport
NEW ON BEATPORT
DJ CHARTS
MACEO PLEX
MUTANT ROMANCE
MPLX
EXCLUSIVE
MPLX
01
Mutant Roman
Maceo Plex | MPLX
$4.98
beatport top10
Hanging Tree Original Mix
Michael Bibi
Repopulate Mars
Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph
Pryda Presents
Along Came Polly Original Mix
Rebuke
Hot Creations
Praise You Purple Disco Machine Extended Remix
Purple Disco Machine, Fatboy Slim
Defected
5
Do It Like This Extended Mix
Biscits
SOLOTOKO
6
Push Pull Club Mix
NEW RELEASES
micro
4 GB
cruzer
Breathe Original Mix
125 BPM / A min / 7:13
$1.99


GENRES
TRACKS
RELEASES
CHARTS
STEMS
SOUNDS
SEARCH...
My Beatport
NEW ON BEATPORT
DJ CHARTS
beatport top10
MACEO PLEX
MUTANT ROMANCE
MPLX
EXCLUSIVE
MPLX
01
Mutant Roman
Maceo Plex
$4.98
Hanging Tree Original Mix
Michael Bibi
Repopulate Mars
Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph
Pryda Presents
micro
4 GB
cruzer
NEW RELEASES
Push Pull Club Mix
Breathe Original Mix
125 BPM / A min / 7:13
$1.99


GENRES
TRACKS
RELEASES
CHARTS
STEMS
SOUNDS
SEARCH...
My Beatport
NEW ON BEATPORT
MACEO PLEX
MUTANT ROMANCE
MPLX
MPLX
01
Mutant Romance
Maceo Plex | MPLX
$4.98
beatport top10
1 Hanging Tree Original Mix
Michael Bibi
Repopulate Mars
2 Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph
Pryda Presents
3 Along Came Polly Original Mix
Rebuke
Hot Creations
4 Praise You Purple Disco Machine Extended Remix
Purple Disco Machine, Fatboy Slim
Defected
5 Do It Like This Extended Mix
Biscits
SOLOTOKO
6 Push Pull Club Mix
NEW RELEASES
Breathe Original Mix
125 BPM / A min / 7:13
$1.99



8.5 million tracks

Business Goal

- Batch process our entire back catalog to-date (8.5 million tracks) to render the following derivative assets:
  - Recalculate **BPM**
    (more accurate algorithm)
  - Recalculate **Key**
    (substantially more accurate algorithm)
  - **128kb AAC** for full-length preview and download
  - Capability to plug-in additional derivative assets into the system easily with minimal/no downtime

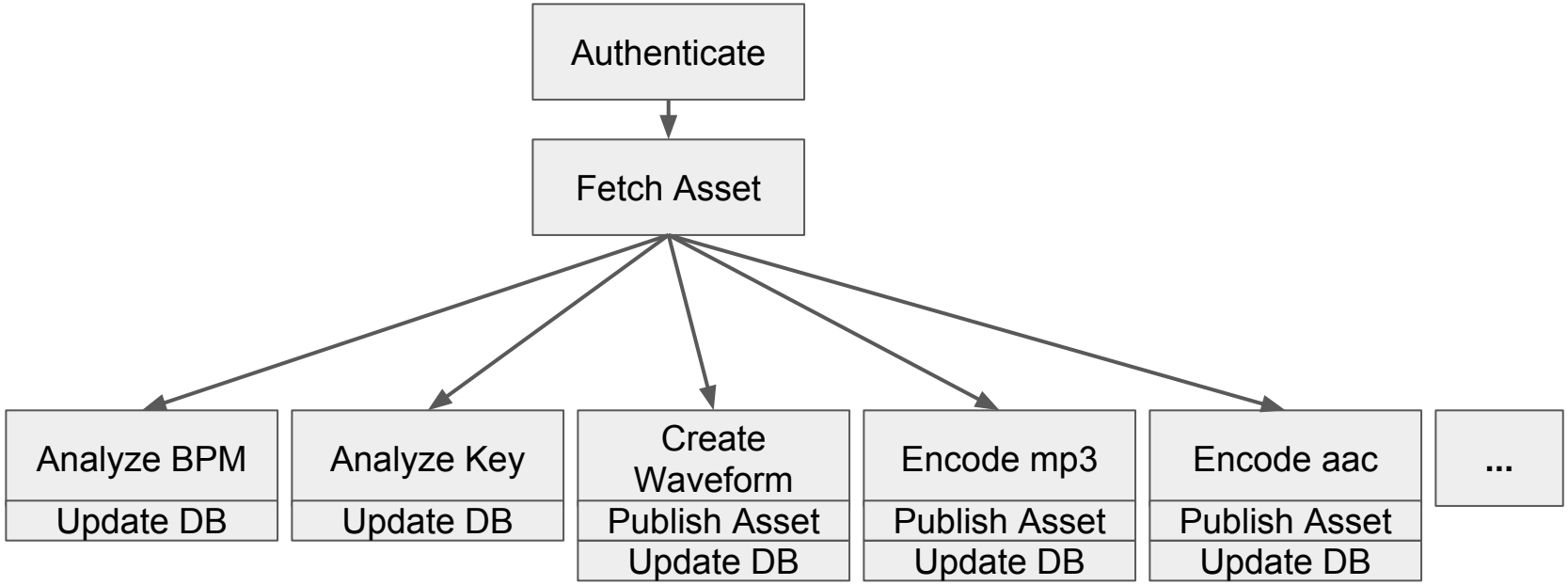
Terms

- **Master**: authoritative source of audio provided by our suppliers to be sold in the store. 16-bit stereo 44.1khz PCM data stored as WAV
- **Derivative Asset:** Any information that is *derived* from an audio master (lossless and lossy compressions, waveforms, metadata like BPM and Key)
- **Back Catalog:** Collection of masters we keep that are currently available in our store for purchase
- **Release:** Single, EP or LP. Collection of content including tracks. Also: album artwork and additional metadata
- **Task:** collection of work to produce derivative assets for a single track (ultimately they share a single pod)

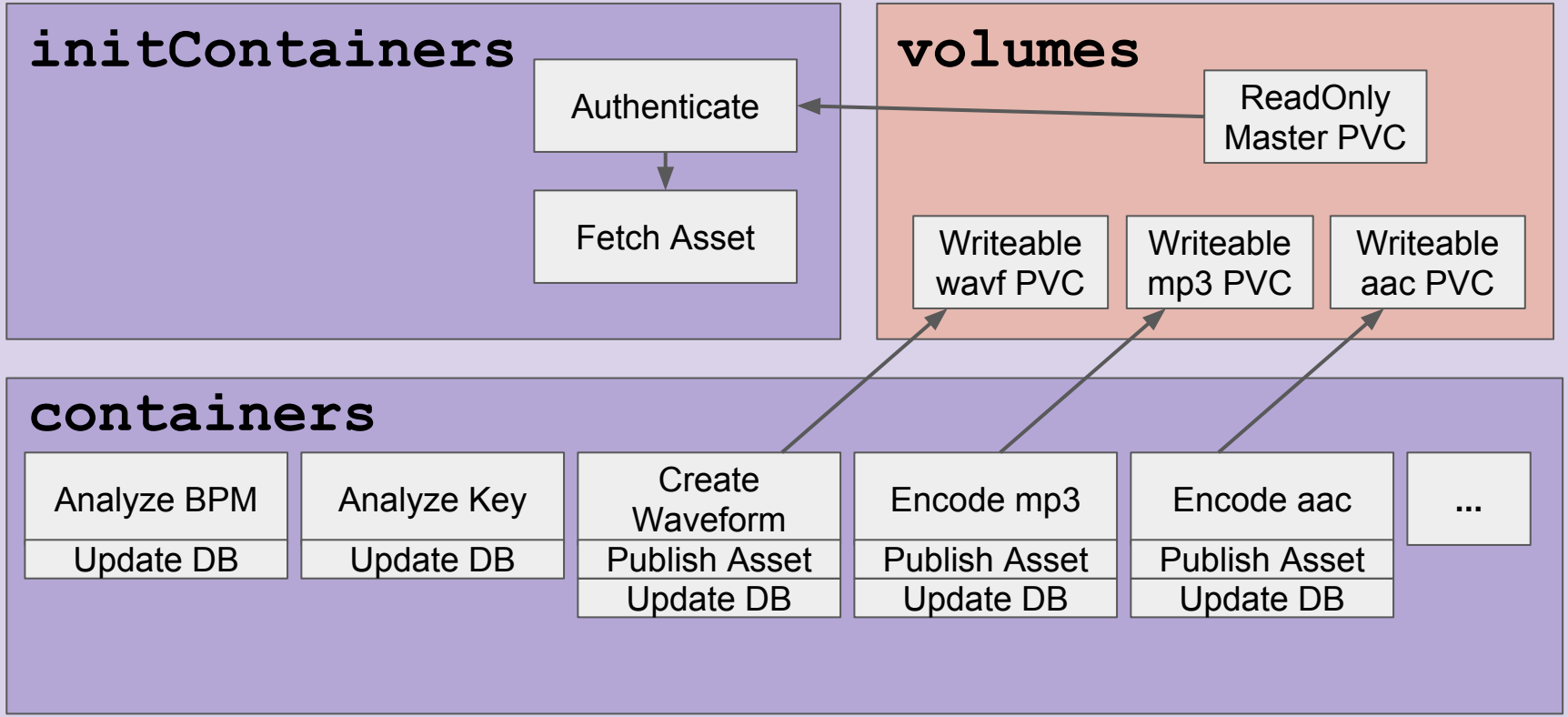
Constraints / Goals

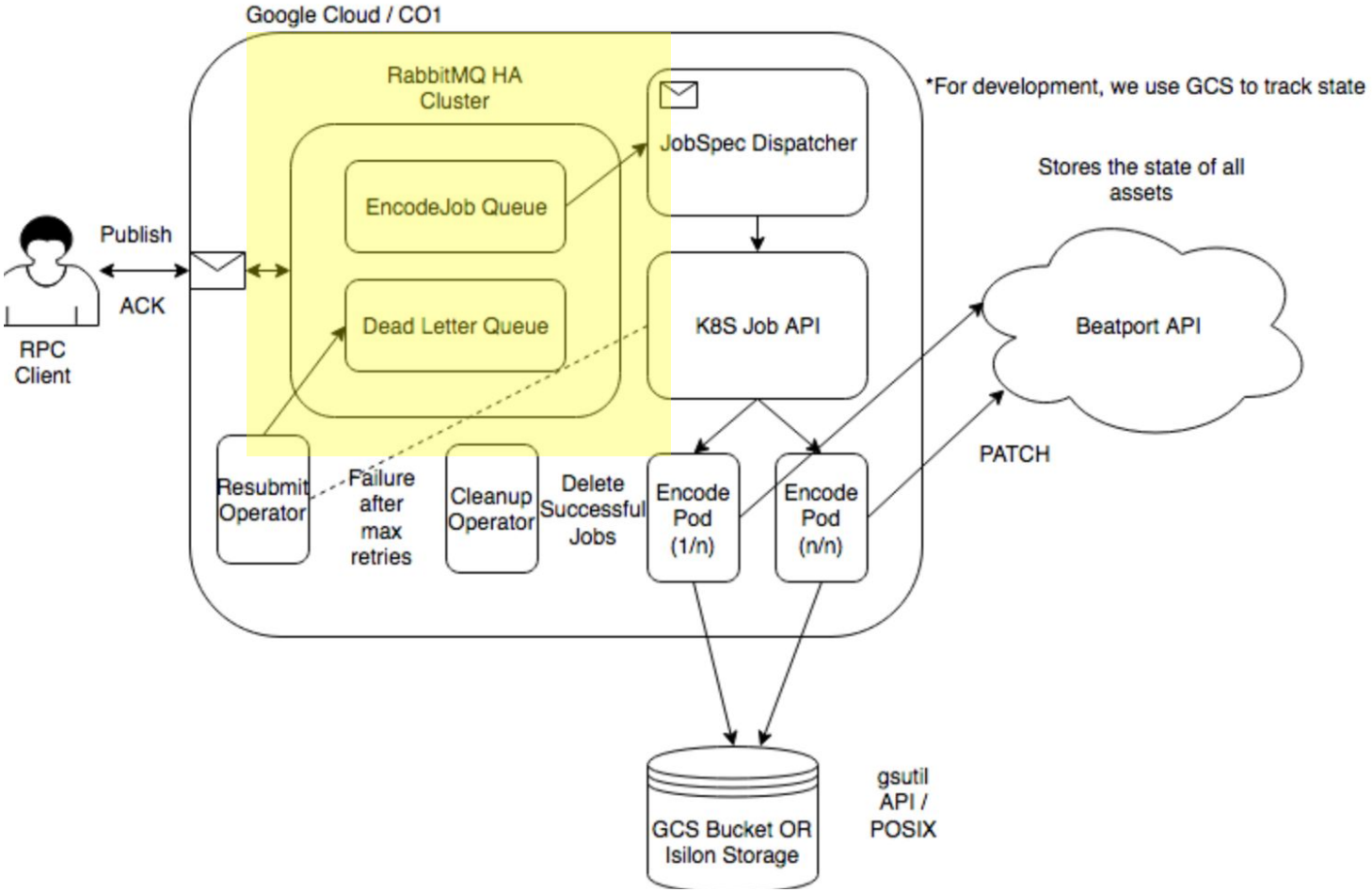
- Newer products running on Google Kubernetes Engine (GKE)
- Old enterprise hardware in our datacenter
- Portable
- Cost-effective
- Storage Models: NFS, Google Cloud Storage Buckets
- Cross-WAN connection from datacenter to GKE, Minimize round-trips
- Observability more challenging for ephemeral workloads
- Handle concurrency, maximizing completion rate, minimizing errors

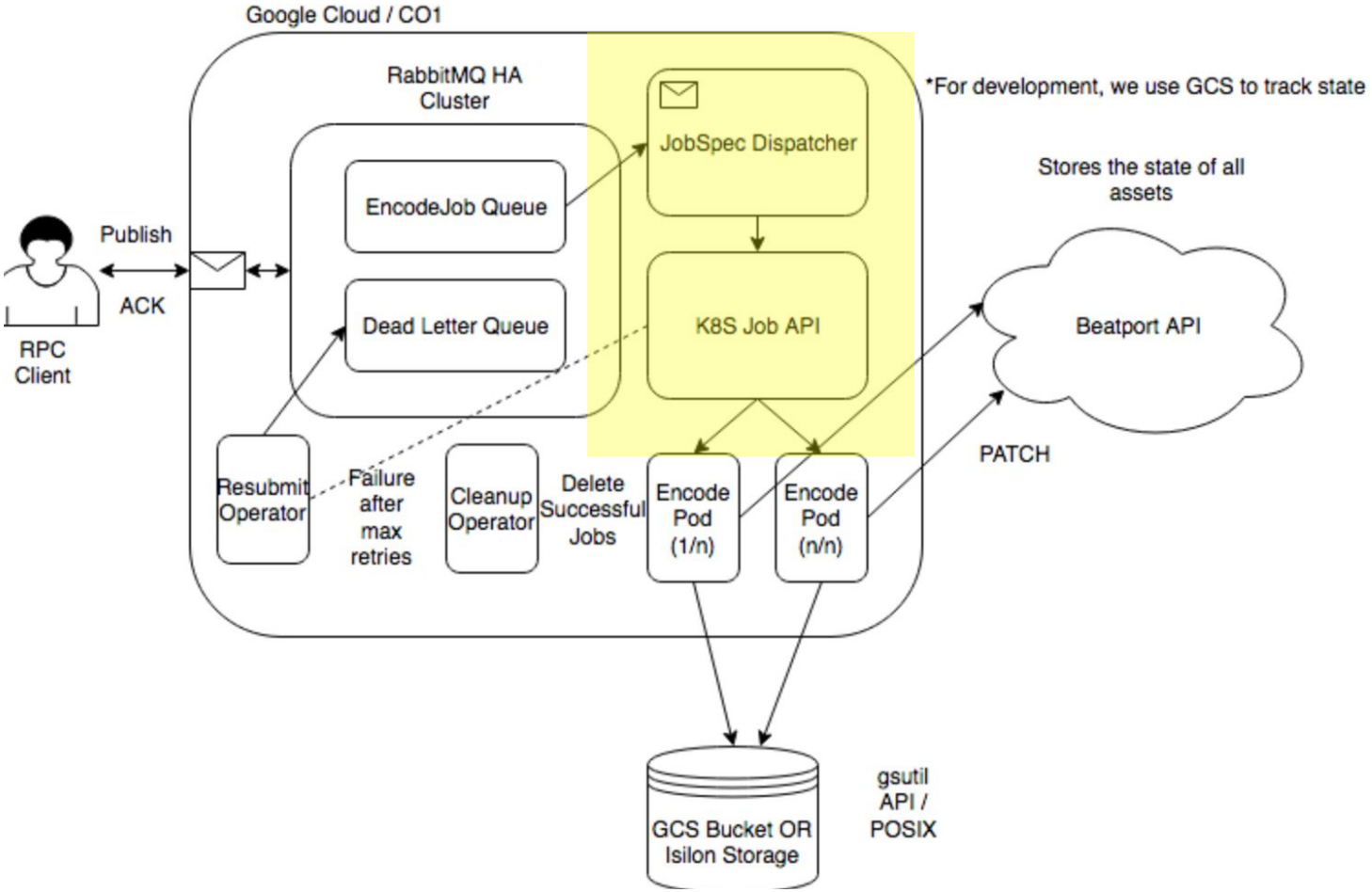
Business Logic

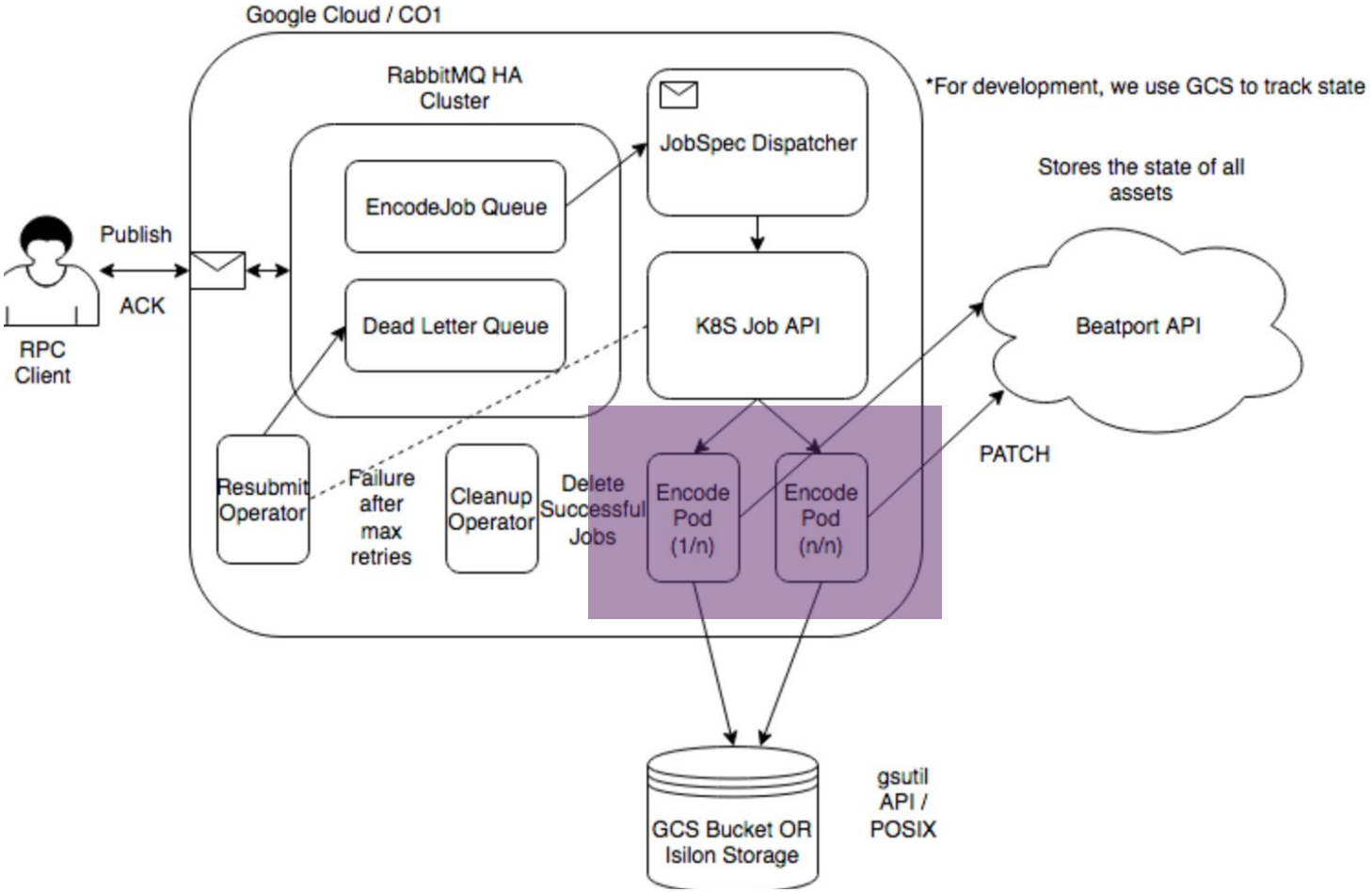


Pod Representation




Google Cloud / CO1
RabbitMQ HA Cluster
EncodeJob Queue
Dead Letter Queue
JobSpec Dispatcher
K8S Job API
*For development, we use GCS to track state
Stores the state of all assets
Publish
ACK
RPC Client
Beatport API
PATCH
Resubmit Operator
Failure after max retries
Cleanup Operator
Delete Successful Jobs
Encode Pod (1/n)
Encode Pod (n/n)
GCS Bucket OR Isilon Storage
gsutil API / POSIX


Google Cloud / CO1
RabbitMQ HA Cluster
*For development, we use GCS to track state
JobSpec Dispatcher
Stores the state of all assets
EncodeJob Queue
Publish
ACK
RPC Client
Dead Letter Queue
K8S Job API
Beatport API
PATCH
Resubmit Operator
Failure after max retries
Cleanup Operator
Delete Successful Jobs
Encode Pod (1/n)
Encode Pod (n/n)
GCS Bucket OR Isilon Storage
gsutil API / POSIX


Google Cloud / CO1
RabbitMQ HA Cluster
*For development, we use GCS to track state
JobSpec Dispatcher
Stores the state of all assets
EncodeJob Queue
Publish
ACK
RPC Client
Dead Letter Queue
K8S Job API
Beatport API
PATCH
Resubmit Operator
Failure after max retries
Cleanup Operator
Delete Successful Jobs
Encode Pod (1/n)
Encode Pod (n/n)
GCS Bucket OR Isilon Storage
gsutil API / POSIX

What information does the system need?

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "EncodeJob",
    "description": "Contract to perform rendering of a derivative asset",
    "type": "object",
    "properties": {
        "request_id": {
            "description": "UUID for a client request",
            "type": "string"
        },
        "release_id": {
            "description": "Regular id for a release from a vendor",
            "type": "integer"
        },
        "asset_id": {
            "description": "Track ID",
            "type": "integer"
        },
        "asset_guid": {
            "description": "Track GUID",
            "type": "string"
        },
        "derivative_assets": {
                "type": "array",
                "items": {
                        "type": "string"
                },
                "uniqueItems": true
        }
    }
}
```

This will mostly be used for processing release artwork (image resizing)

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "EncodeJob",
    "description": "Contract to perform rendering of a derivative asset",
    "type": "object",
    "properties": {
        "request_id": {
            "description": "UUID for a client request",
            "type": "string"
        },
        "release_id": {
            "description": "Regular id for a release from a vendor",
            "type": "integer"
        },
        "asset_id": {
            "description": "Track ID",
            "type": "integer"
        },
        "asset_guid": {
            "description": "Track GUID",
            "type": "string"
        },
        "derivative_assets": {
                "type": "array",
                "items": {
                        "type": "string"
                },
                "uniqueItems": true
        }
    }
}
```

Integer ID we use to reference asset in catalog API

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "EncodeJob",
    "description": "Contract to perform rendering of a derivative asset",
    "type": "object",
    "properties": {
        "request_id": {
            "description": "UUID for a client request",
            "type": "string"
        },
        "release_id": {
            "description": "Regular id for a release from a vendor",
            "type": "integer"
        },
        "asset_id": {
            "description": "Track ID",
            "type": "integer"
        },
        "asset_guid": {
            "description": "Track GUID",
            "type": "string"
        },
        "derivative_assets": {
                "type": "array",
                "items": {
                        "type": "string"
                },
                "uniqueItems": true
        }
    }
}
```

Used for storage of an asset. Introduced recently so we must include this as well

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "EncodeJob",
    "description": "Contract to perform rendering of a derivative asset",
    "type": "object",
    "properties": {
        "request_id": {
            "description": "UUID for a client request",
            "type": "string"
        },
        "release_id": {
            "description": "Regular id for a release from a vendor",
            "type": "integer"
        },
        "asset_id": {
            "description": "Track ID",
            "type": "integer"
        },
        "asset_guid": {
            "description": "Track GUID",
            "type": "string"
        },
        "derivative_assets": {
                "type": "array",
                "items": {
                        "type": "string"
                },
                "uniqueItems": true
        }
    }
}
```

Give me a list of things to produce

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "EncodeJob",
    "description": "Contract to perform rendering of a derivative asset",
    "type": "object",
    "properties": {
        "request_id": {
            "description": "UUID for a client request",
            "type": "string"
        },
        "release_id": {
            "description": "Regular id for a release from a vendor",
            "type": "integer"
        },
        "asset_id": {
            "description": "Track ID",
            "type": "integer"
        },
        "asset_guid": {
            "description": "Track GUID",
            "type": "string"
        },
        "derivative_assets": {
                "type": "array",
                "items": {
                        "type": "string"
                },
                "uniqueItems": true
        }
    }
}
```

Prototype

- Need to regulate flow of job requests to k8s API: use a queue
- Decided against single-purpose daemons so we only fetch an asset once, don't need complex scheduling logic
- Declare a message spec
- Send to queue
- A "dispatcher" will interpret the sent message and mint a jobspec (or podspec) to send to the kubernetes API

Hardware Assets

- 1 Rack
- 12 Dell R520 (2012)
- 1 Dell R410 (2009)
- 1 Dell R620 (2012)
- 4 Dell PE2950 (2006-2008)

Hardware Assets

- 1 Rack
- 12 Dell R520 (2012)
- 1 Dell R410 (2009)
- 1 Dell R620 (2012)
- 4 Dell PE2950 (2006-2008)

Hardware Assets

- 1 Rack
- 12 Dell R520 (2012)
- 1 Dell R410 (2009)
- 1 Dell R620 (2012)
- 4 Dell PE2950 (2006-2008)

Kubernetes v1.11.0

Docker 18.03.1-ce

CentOS 7

Community maintained Linux Kernel (4.16)

Prototype

Good:

- Quick turn-around time to validating idea. (1 week!)
- Pre-populated content samples into a GCS bucket
- GKE allowed us to prototype the software's behavior at scale easily
- Observed the system was behaving with **hjacobs/kube-ops-view**

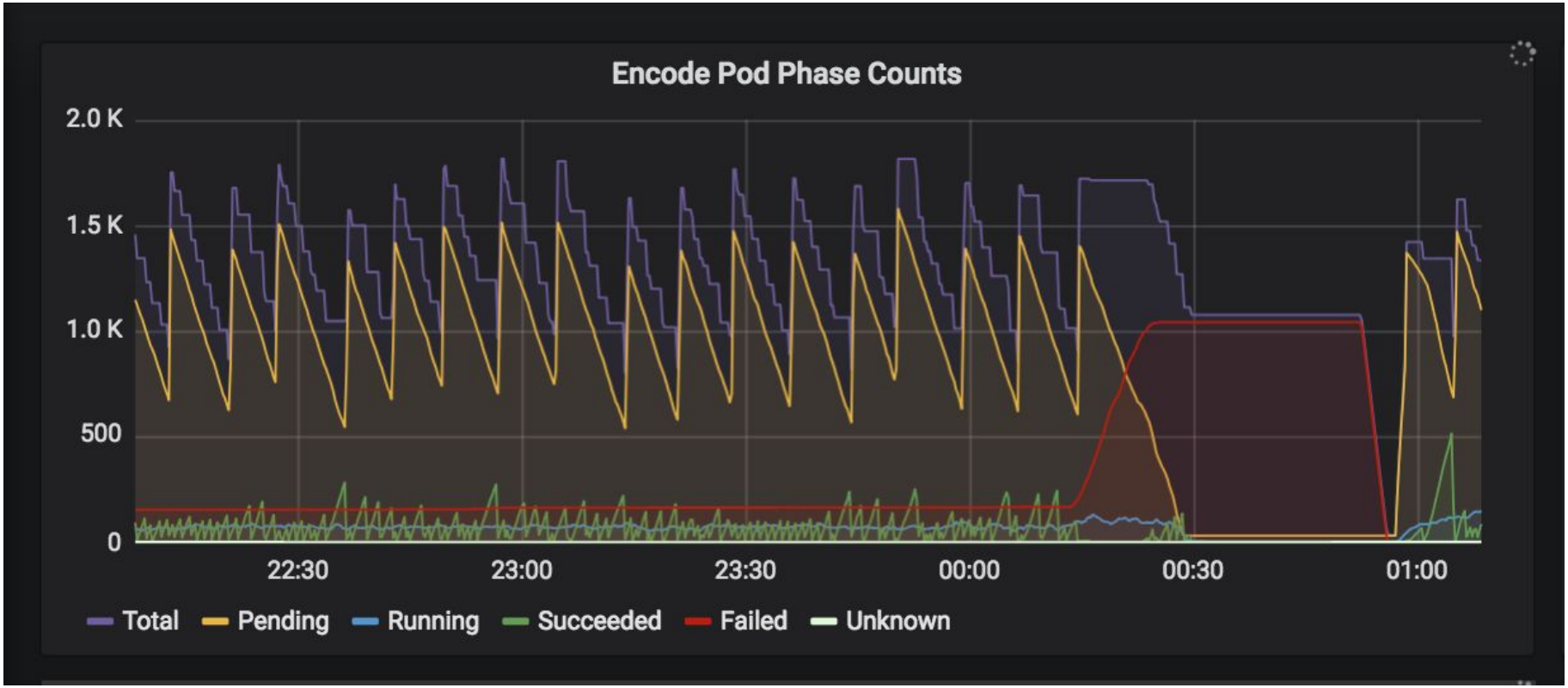
Frustrating:

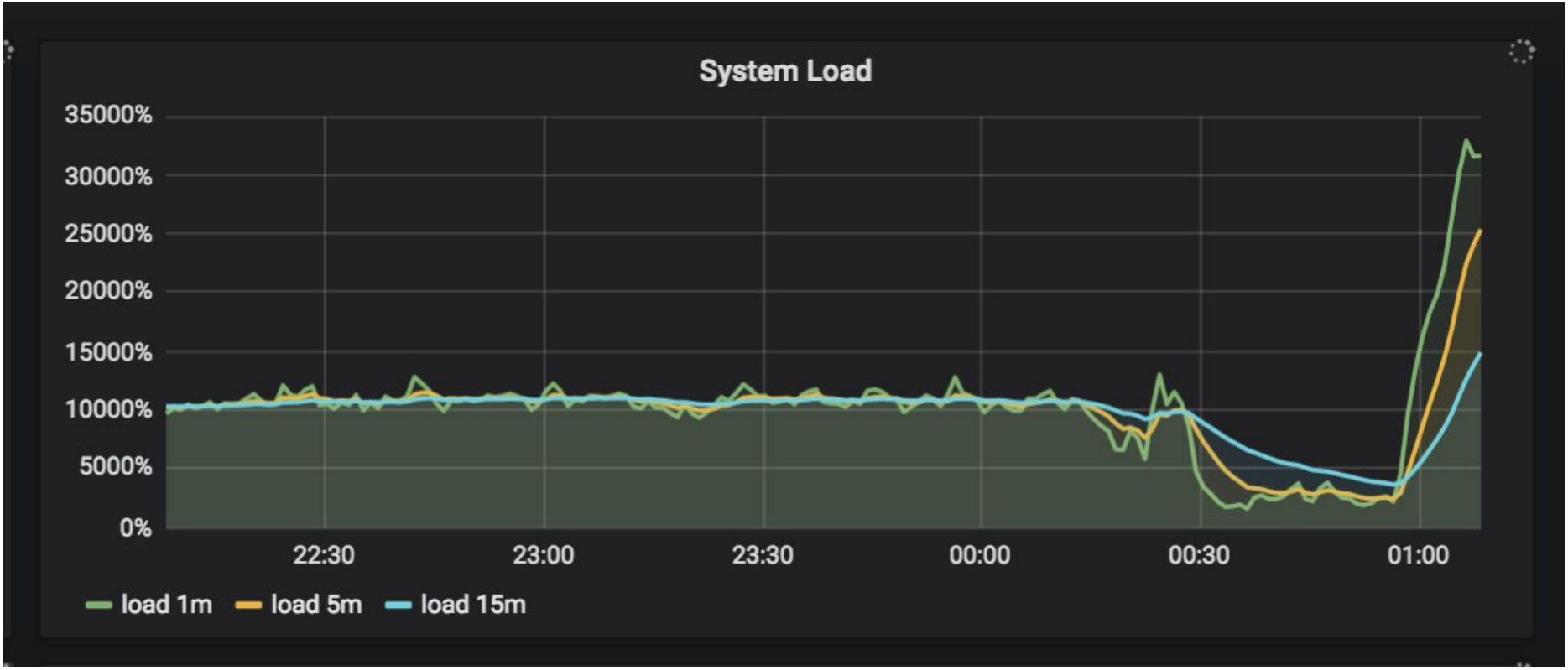
- Python kubernetes client was easy to iterate with, but it was difficult to work with type mismatch errors.

Demo :)

Performance / KPIs

- Failure Rate
- Success Rate
- Completions per minute
- Concurrency
- Garbage collection of Tasks
- Requeuing and recording of failed Tasks


Encode Pod Phase Counts
2.0 K
1.5 K
1.0 K
500
0
22:30
23:00
23:30
00:00
00:30
01:00
Total
Pending
Running
Succeeded
Failed
Unknown


System Load
35000%
30000%
25000%
20000%
15000%
10000%
5000%
0%
22:30
23:00
23:30
00:00
00:30
01:00
load 1m
load 5m
load 15m

Scheduling

- Requests
- Limits

  ??????????

Information on CPU Pinning / Temporal Slicing:

- https://hackernoon.com/job-concurrency-in-kubernetes-lxd-and-cpu-pinning-to-the-rescue-b9fb7b44f99d
- https://stgraber.org/2016/03/26/lxd-2-0-resource-control-412/

Bugs :)

kube-state-metrics CrashLoops

KSM defaults have conservative **resourceRequirements** which are dynamically calculated based on node size. (kubernetes/autoscaler: /addon-resizer)

Our cluster has many more objects per-node than most small clusters.
We needed to scale this because the KSM shared informer was OOM'ing on our # Jobs & Pods.

This was preventing us from viewing needed metrics.

Solution: **Scale KSM CPU and memory limits**

kube-controller-manager /metrics time out

The KCM seems to return metrics about every single object that has ever been in the cluster

When you have 140,000 pods/jobs a day, it takes a very long time to return this list of metrics
It's unclear what the garbage collection time period for this

This causes the prometheus scraper to time out

Workaround: **Restart the KCM when you want metrics**
(This is a static pod -- delete the KCM docker container on the master)

Job Controller Saturation

- Past a certain threshold, CRUD operations became impossibly slow over time (particularly batch deletes)
- Bottlenecked not by CPU, but now by Kubernetes
- Since we use the number of Jobs or Pods in the cluster as a back-pressure mechanism this was effectively thwarting the throughput of the system
- We think this may be related to how the Job Controller waits for child pods to be deleted before removing the parent job object
- Lots of variables in our environment that could affect Job controller performance such as lists, creates, and updates from pod scheduling

Workaround: **Give up the benefits of the Job Controller -- use Bare Pods**

corev1/BearPod

Jobs

- Parallelism
- Retries
- Completions
- “ttlSecondsAfterFinished” automatic GC as of 1.12 alpha

Jobs

- ~~Parallelism~~
- Retries
- ~~Completions~~
- “ttlSecondsAfterFinished” automatic GC as of 1.12 alpha

Pods Implementation

How do you clean up?

- CronJob for garbage collection
  (Golang K8S client)
- CronJob for dead-lettering failed tasks
  (Golang K8S client + AMQP)

Pods Implementation

- Free retries:
  - Already have a dead-lettering mechanism built into the queue
  - These can be triaged shoveled easily
- Acceptable performance

- Limit max pods allowed in API via dispatcher

  Threshold provides backpressure for objects
  Most of our state is in RabbitMQ, not in k8s

Pods API vs Jobs API quality

- Jobs API has some typos in the hidden field selectors
  - Need to fix
- Pods API has more field selectors than Jobs
  - Useful for `kubectl | jq`
- Performance surpasses our business needs

Completed the catalog of 8.5 million tracks:

80 days /w **63** days of runtime

Encoding 250,000 Songs a Day with batchV1/Jobs

Encoding **140,000** Songs a Day with **coreV1/Pods**

@capileigh
@jslivka

Questions