Batch Encoding Audio with Kubernetes

Leigh Capili
Infrastructure Engineer, Beatport

John Slivka
Infrastructure Engineer, Beatport

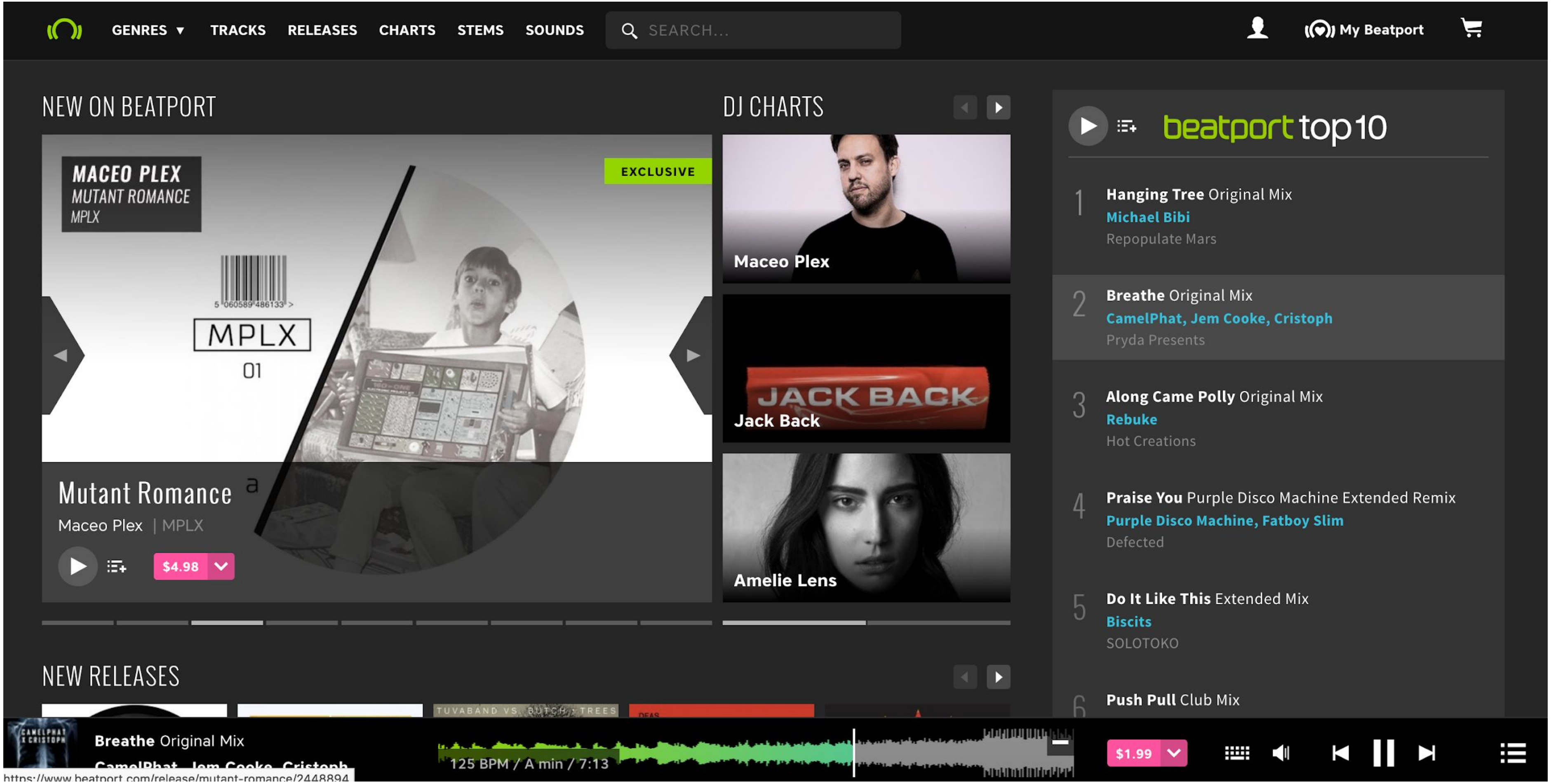
Leigh Capili

- Infrastructure Engineer
- SIG Cluster-Lifecycle

- @capileigh
- github: stealthybox

John Slivka

- Infrastructure Engineer
- https://github.com/jslivka
- @jslivka



NEW ON BEATPORT

Mutant Romance

Maceo Plex | MPLX

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

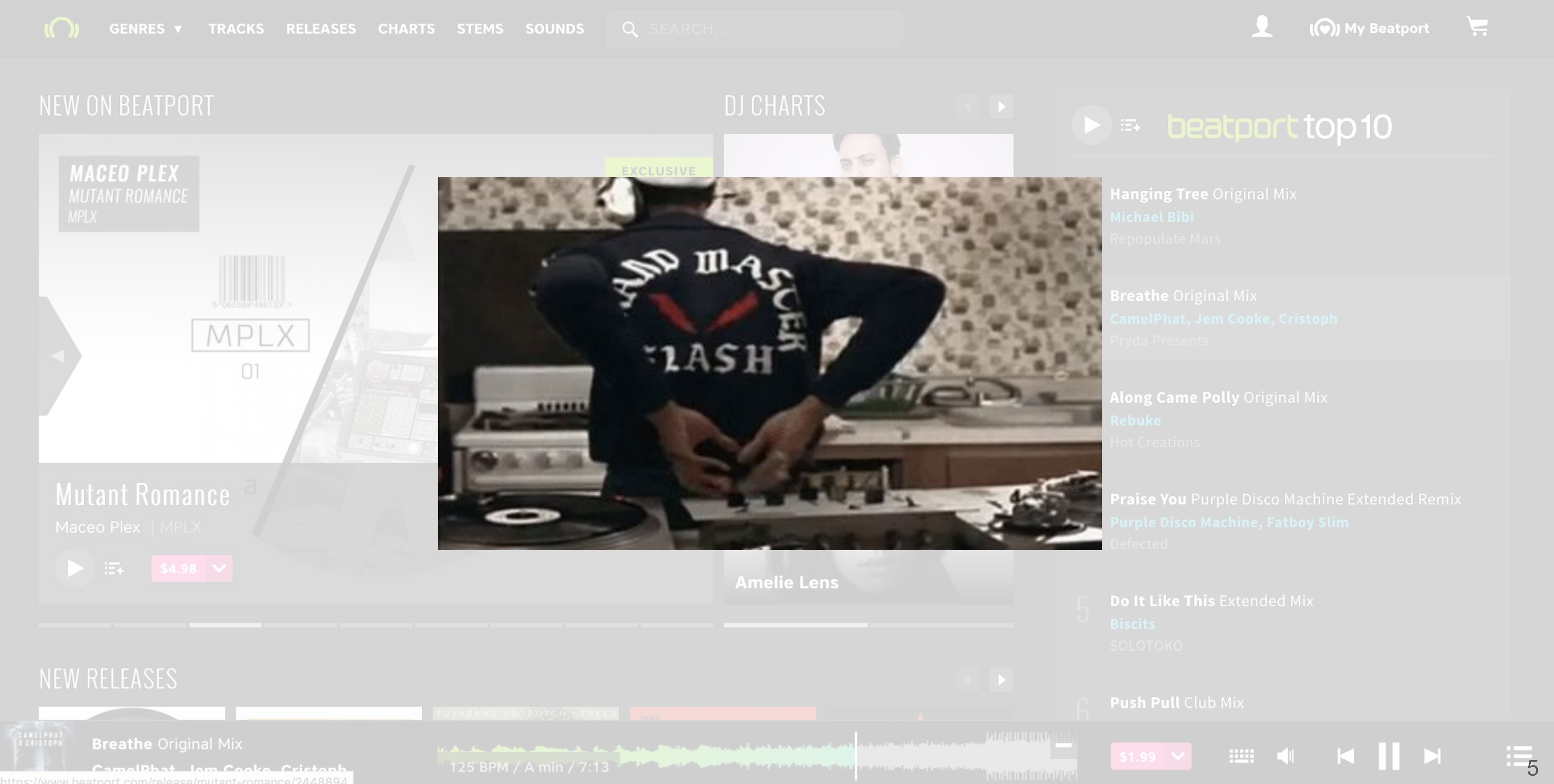
6 **Push Pull** Club Mix

NEW RELEASES

Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph

125 BPM / A min / 7:13

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

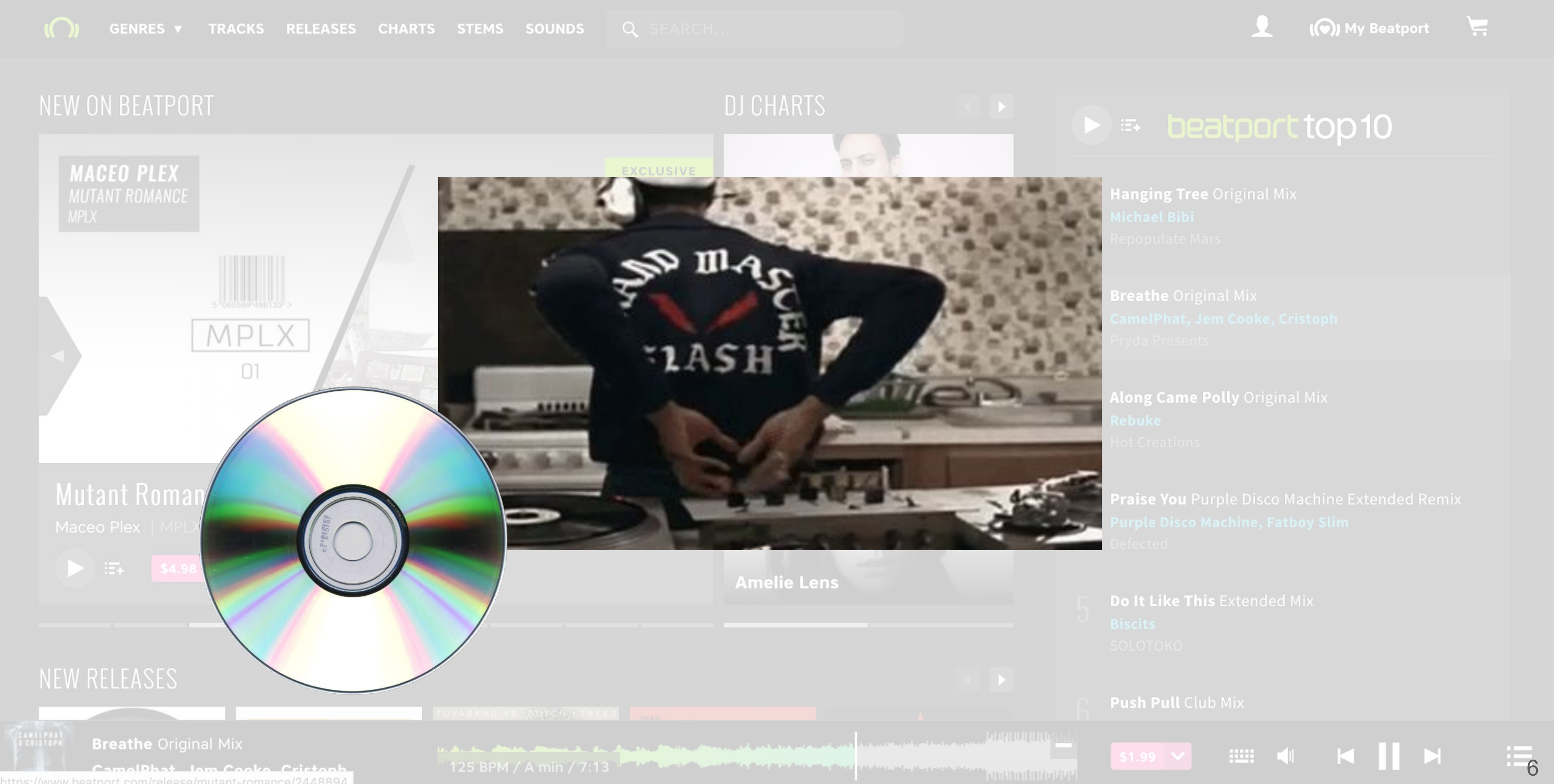
6 **Push Pull** Club Mix

NEW RELEASES

TUVABAND VS. DUTCH TREES

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

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



NEW ON BEATPORT

Mutant Roman

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

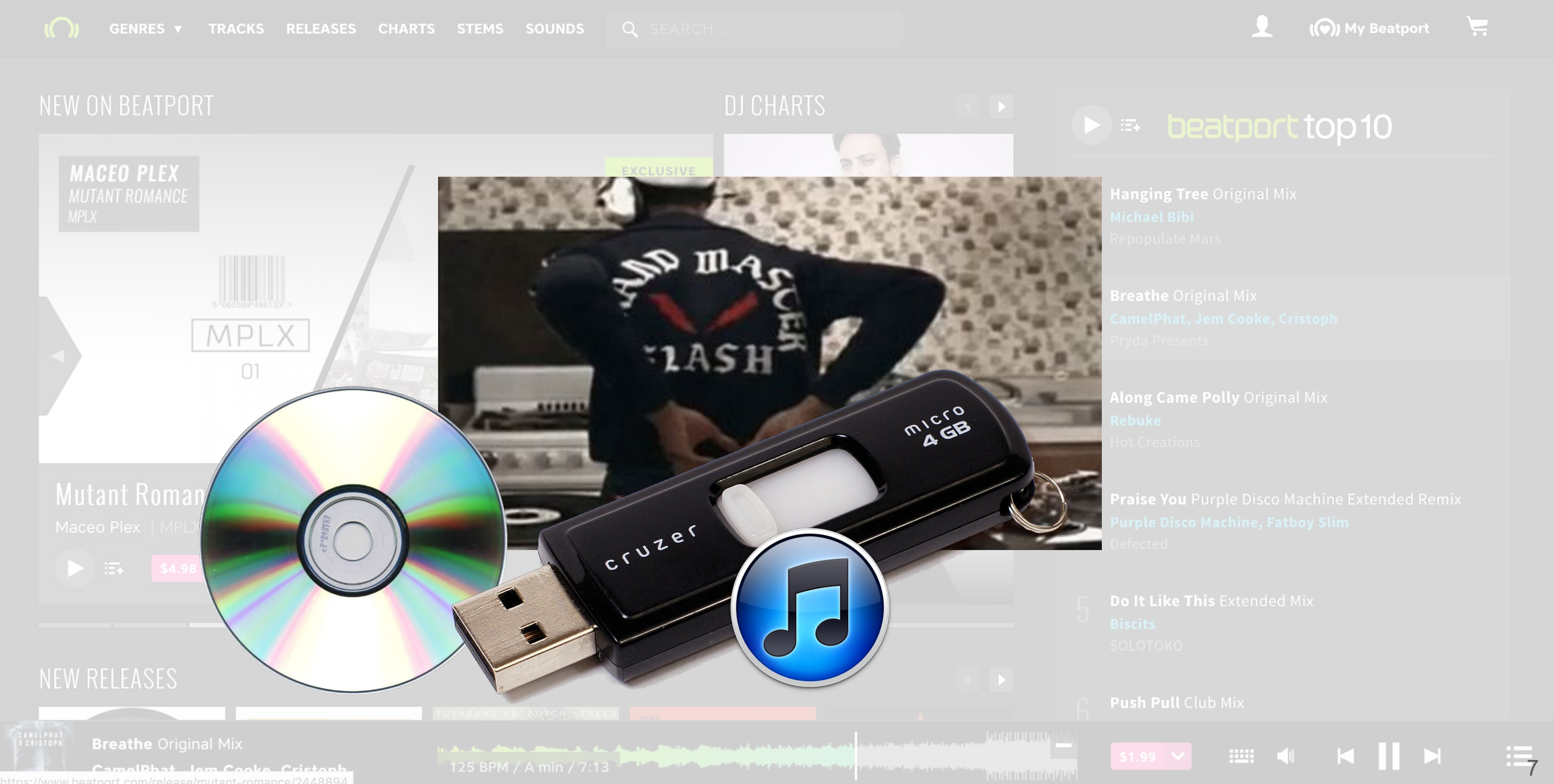
NEW RELEASES

Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph

125 BPM / A min / 7:13

$1.99

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



NEW ON BEATPORT

Mutant Roman

Maceo Plex | MPLX

DJ CHARTS

beatport top 10

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

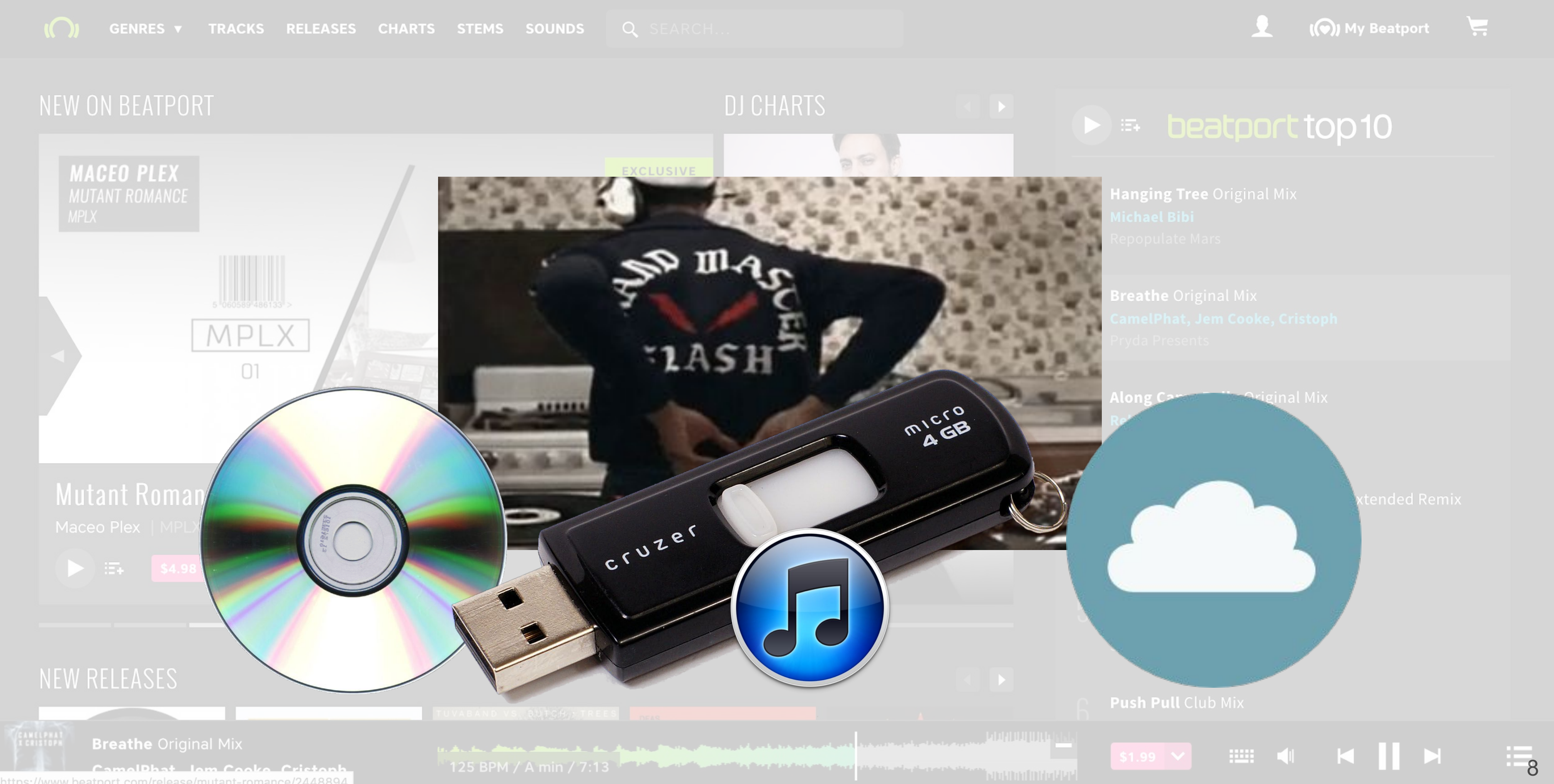
5 Do It Like This Extended Mix
Biscits
SOLOTOKO

6 Push Pull Club Mix

NEW RELEASES

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

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



NEW ON BEATPORT

Mutant Roman

Maceo Plex | MPLX

$4.98

DJ CHARTS

Hanging Tree Original Mix
Michael Bibi
Repopulate Mars

Breathe Original Mix
CamelPhat, Jem Cooke, Cristoph
Pryda Presents

Along Ca… Original Mix

…xtended Remix

Push Pull Club Mix

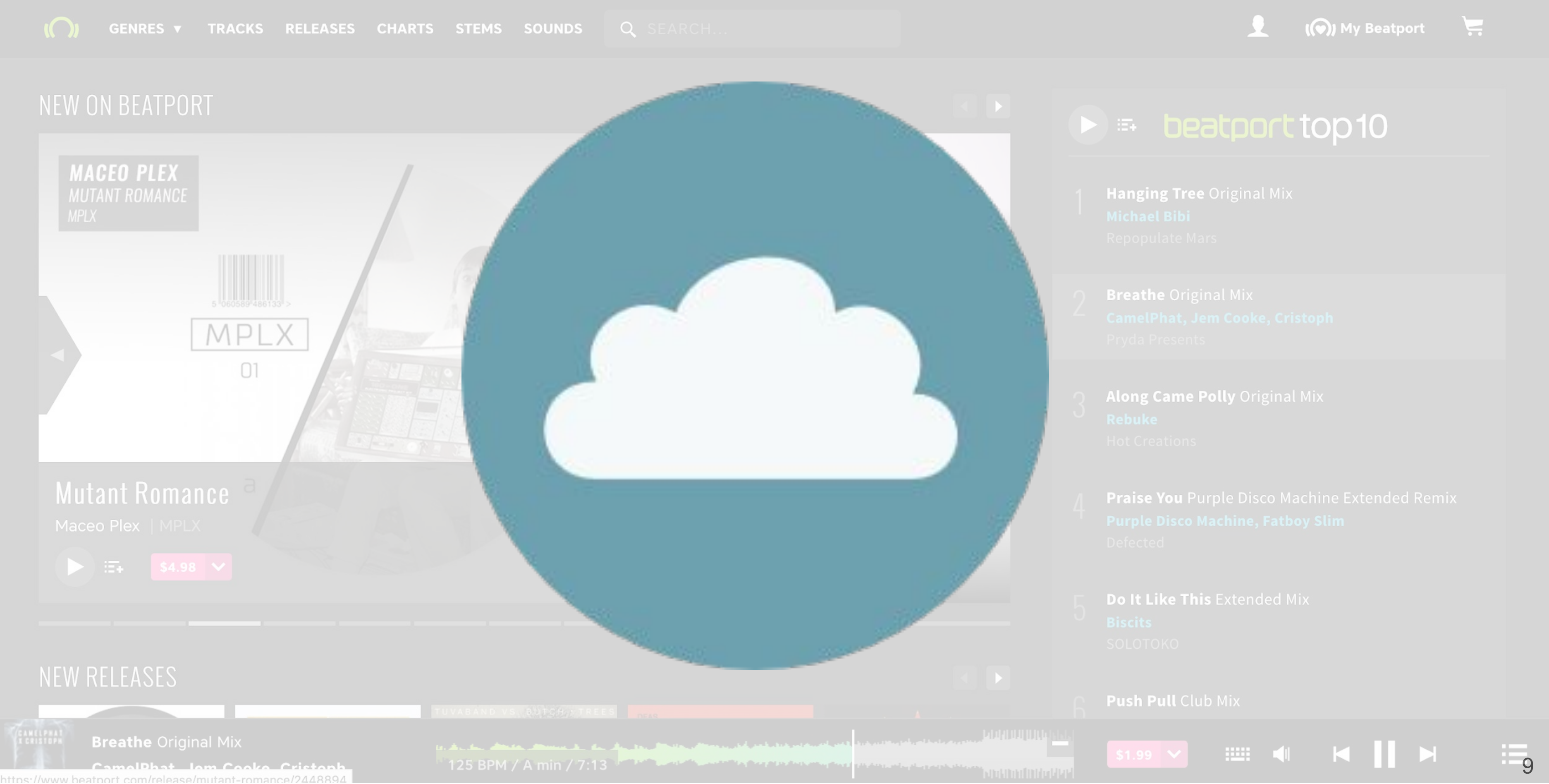
NEW RELEASES

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

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

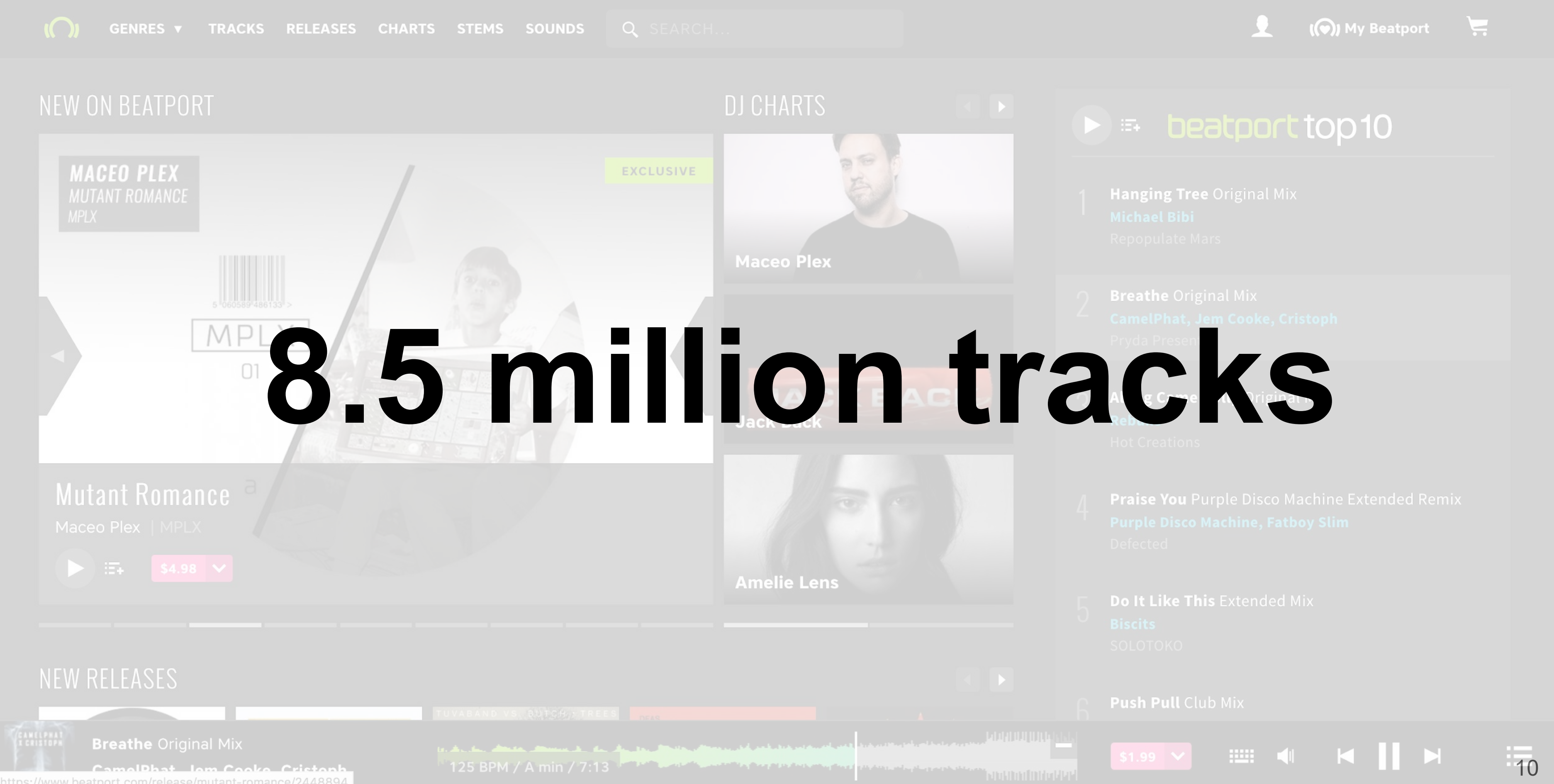
5 **Do It Like This** Extended Mix
Biscits
SOLOTOKO

6 **Push Pull** Club Mix

NEW RELEASES

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

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



8.5 million tracks

Agenda

- Timeline
- Business Goal
- Problem Definition
- Approach
- Prototype
- Performance
- Scheduling
- Operations
- Conclusion
- Q&A

Business Goal

- Batch process our entire back catalog to-date (8-9 million tracks) to render the following derivative assets
    - Recalculate BPM (more accurate algorithm)
    - Recalculate Key (substantially more accurate algorithm)
    - 128K AAC for full-length preview and download
    - Capability to plug-in additional derivative assets into the system easily with minimal/no downtime

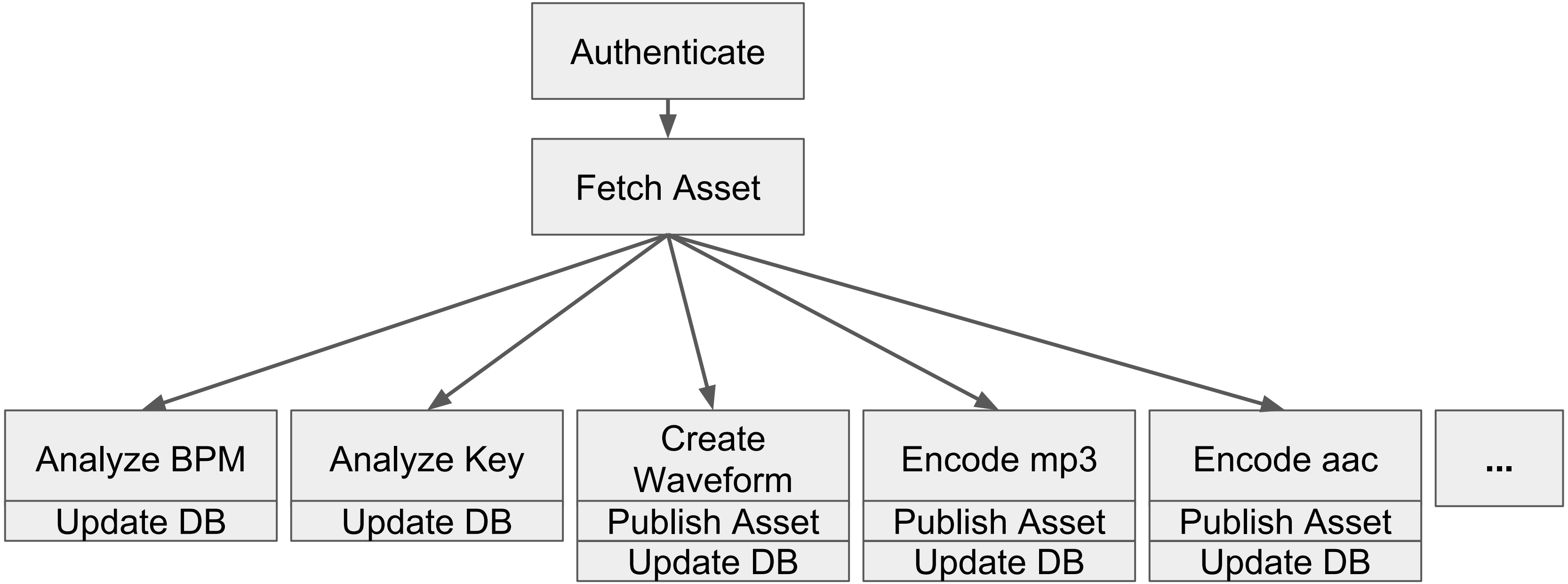
Terms

- **Master**: authoritative source of audio provided by our suppliers to be sold in the store. 16-bit stereo 44.1K 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)

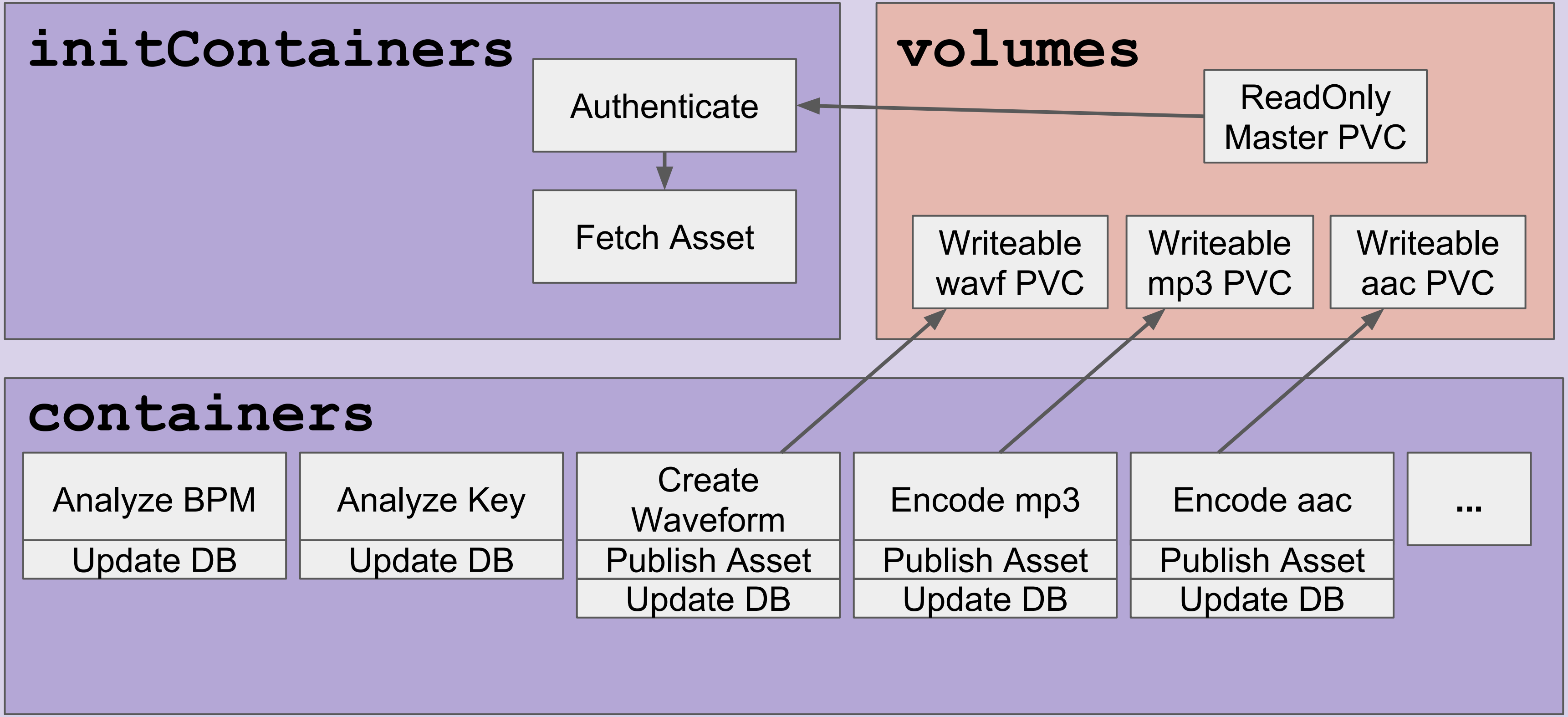
Constraints / Goals

- Newer products running on Google Kubernetes Engine (GKE)
- Old enterprise hardware in our datacenter
- Portability
- Storage Models: NFS, Google Cloud Storage Buckets
- Cross-WAN connection from datacenter to GKE, Minimize round-trips
- Observability more challenging for ephemeral workloads
- Handling concurrency and maximizing completion rate is tricky

Business Logic



Pod Representation



`initContainers`

Authenticate

Fetch Asset

`volumes`

ReadOnly Master PVC

Writeable wavf PVC

Writeable mp3 PVC

Writeable aac PVC

`containers`

Analyze BPM
Update DB

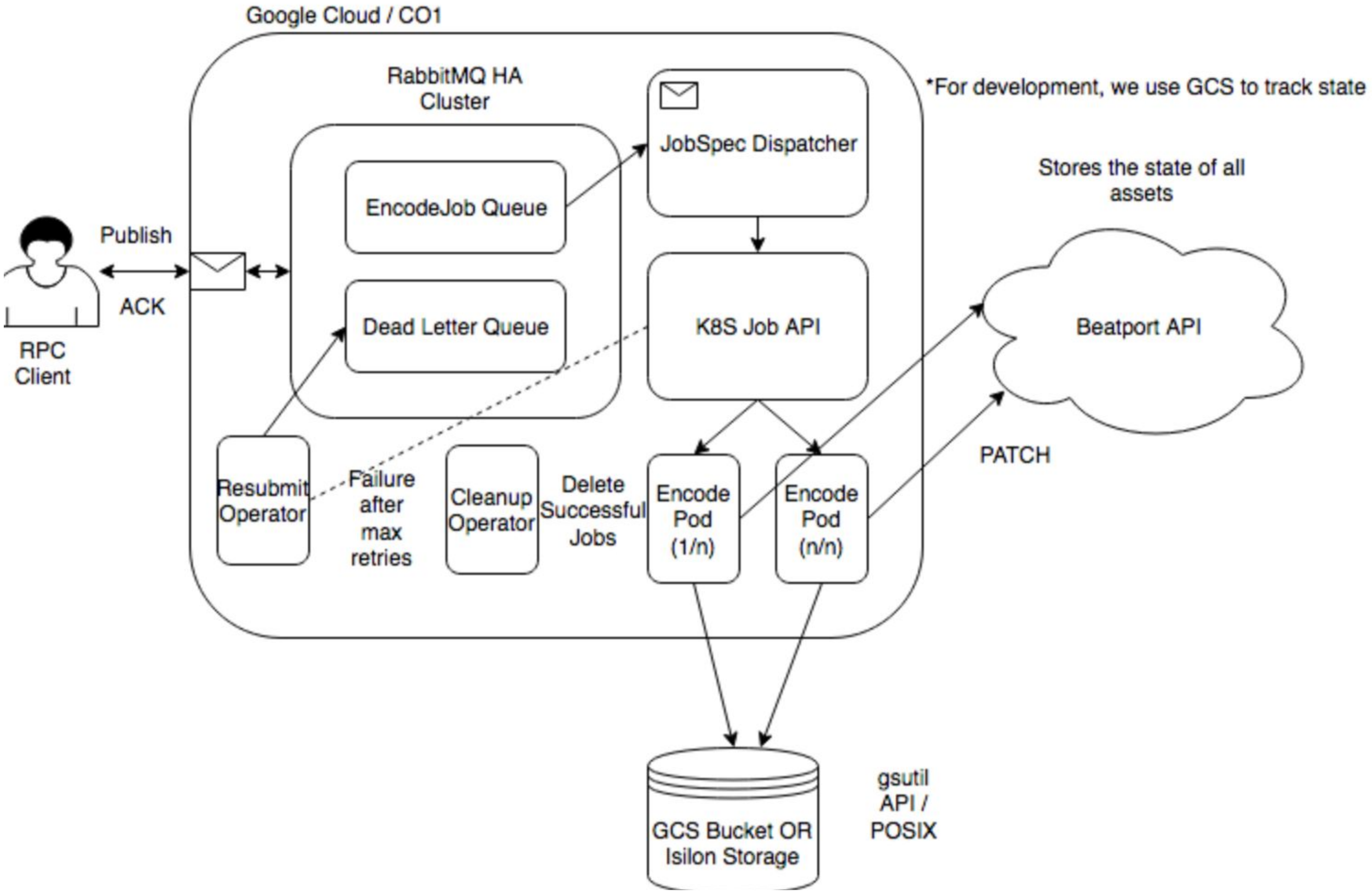
Analyze Key
Update DB

Create Waveform
Publish Asset
Update DB

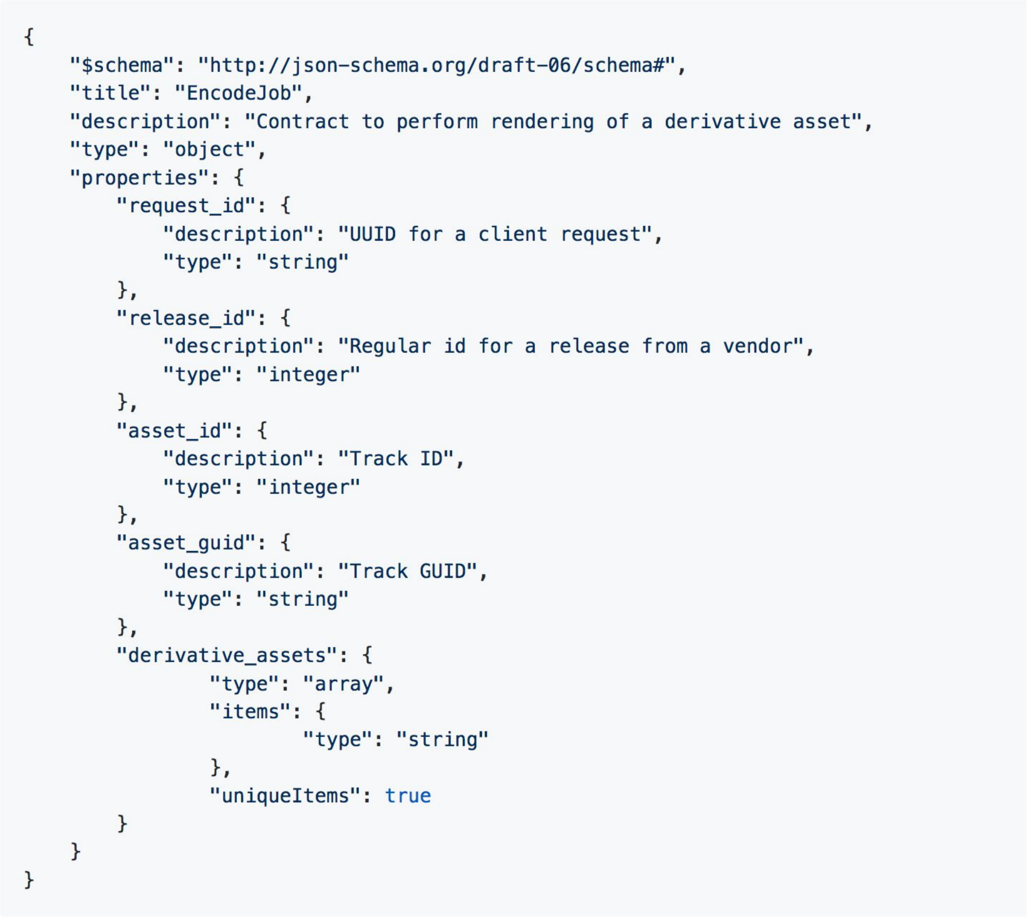
Encode mp3
Publish Asset
Update DB

Encode aac
Publish Asset
Update DB

...

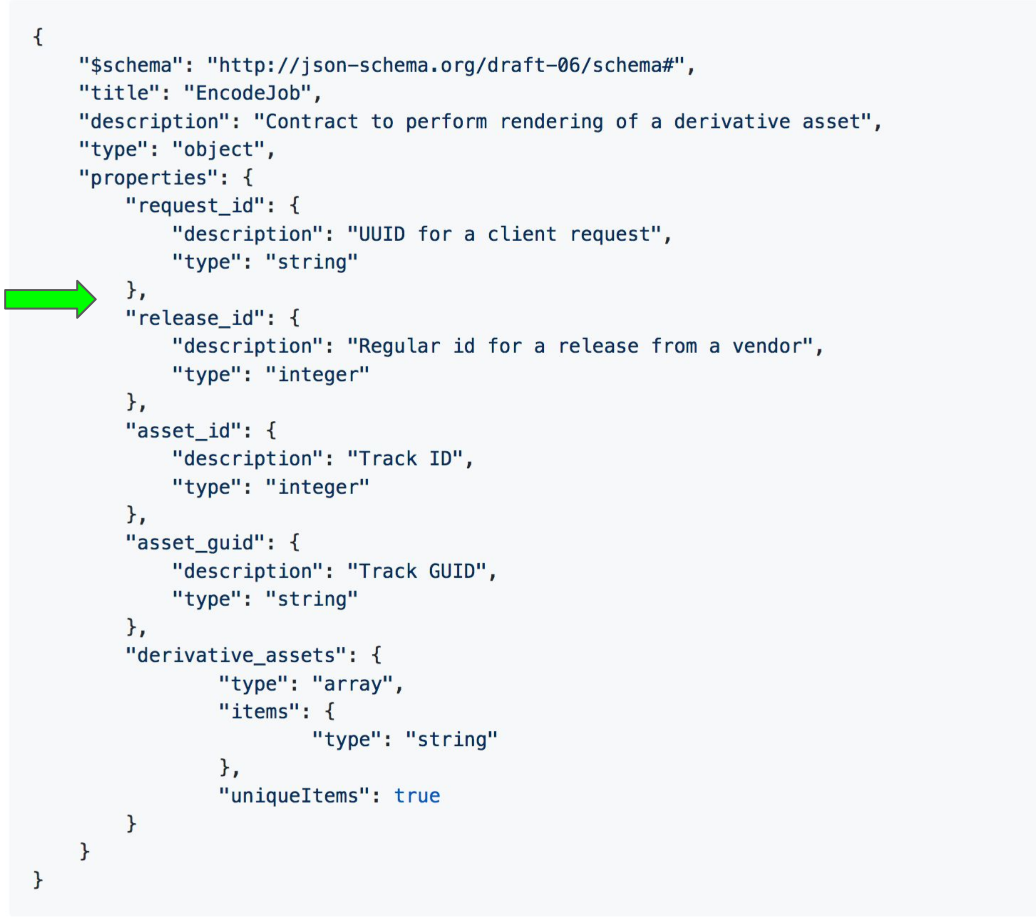

Google Cloud / CO1
RabbitMQ HA Cluster
EncodeJob Queue
Dead Letter Queue
JobSpec Dispatcher
K8S Job API
*For development, we use GCS to track state
Stores the state of all assets
Beatport API
Publish
ACK
RPC Client
Resubmit Operator
Failure after max retries
Cleanup Operator
Delete Successful Jobs
Encode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

What information does the system need?



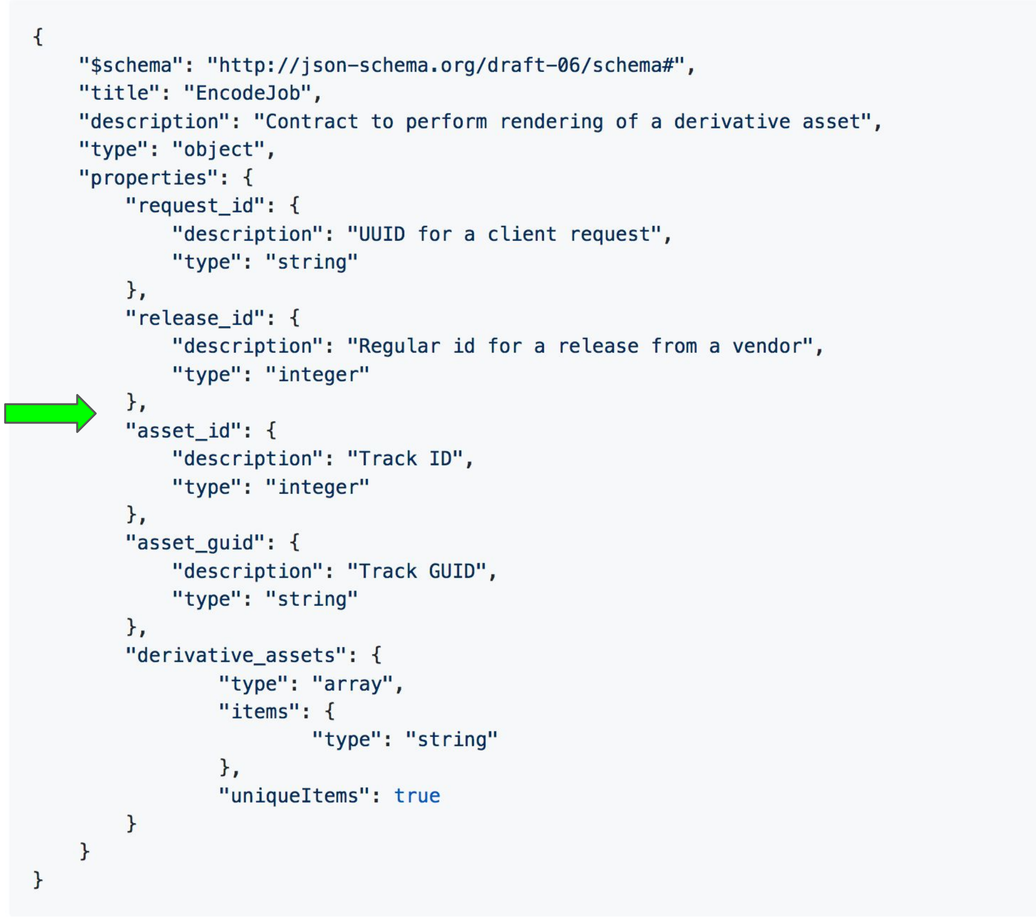
```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "EncodeJob",
    "description": "Contract to perform rendering of a derivative asset",
    "type": "object",
    "properties": {
        "request_id": {
            "description": "UUID for a client request",
            "type": "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)



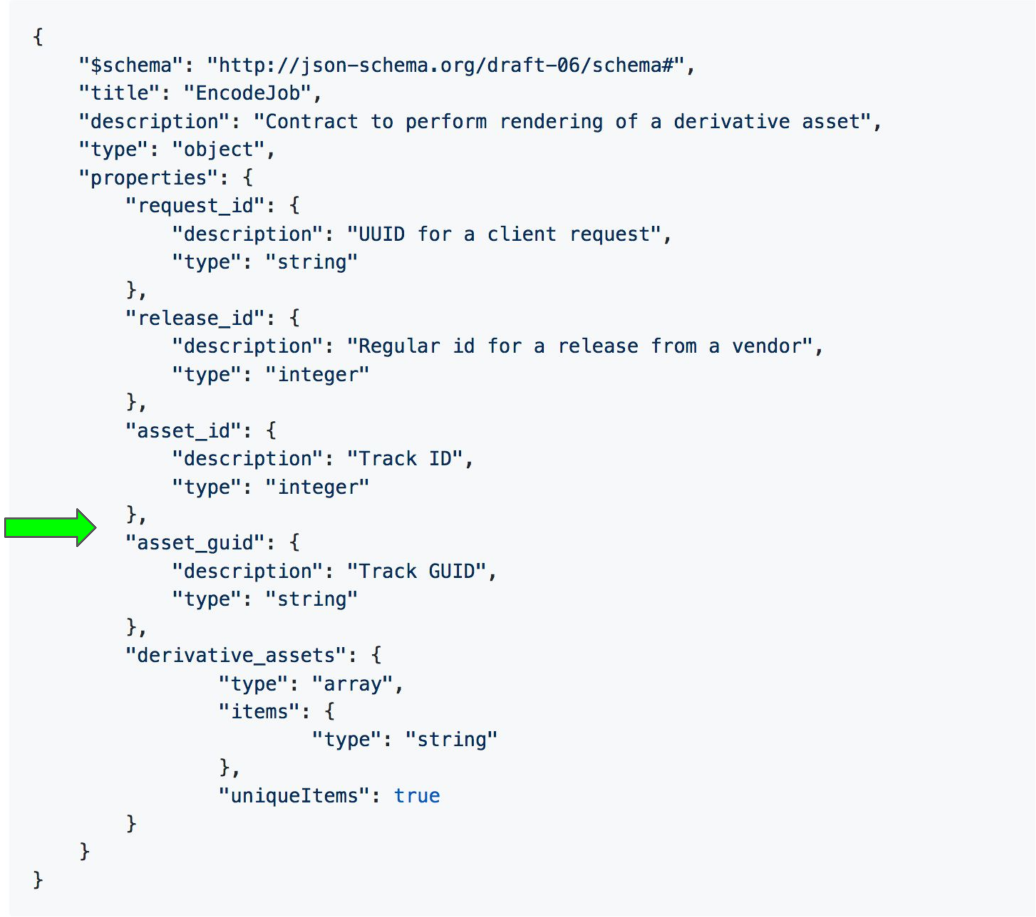
```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "EncodeJob",
    "description": "Contract to perform rendering of a derivative asset",
    "type": "object",
    "properties": {
        "request_id": {
            "description": "UUID for a client request",
            "type": "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



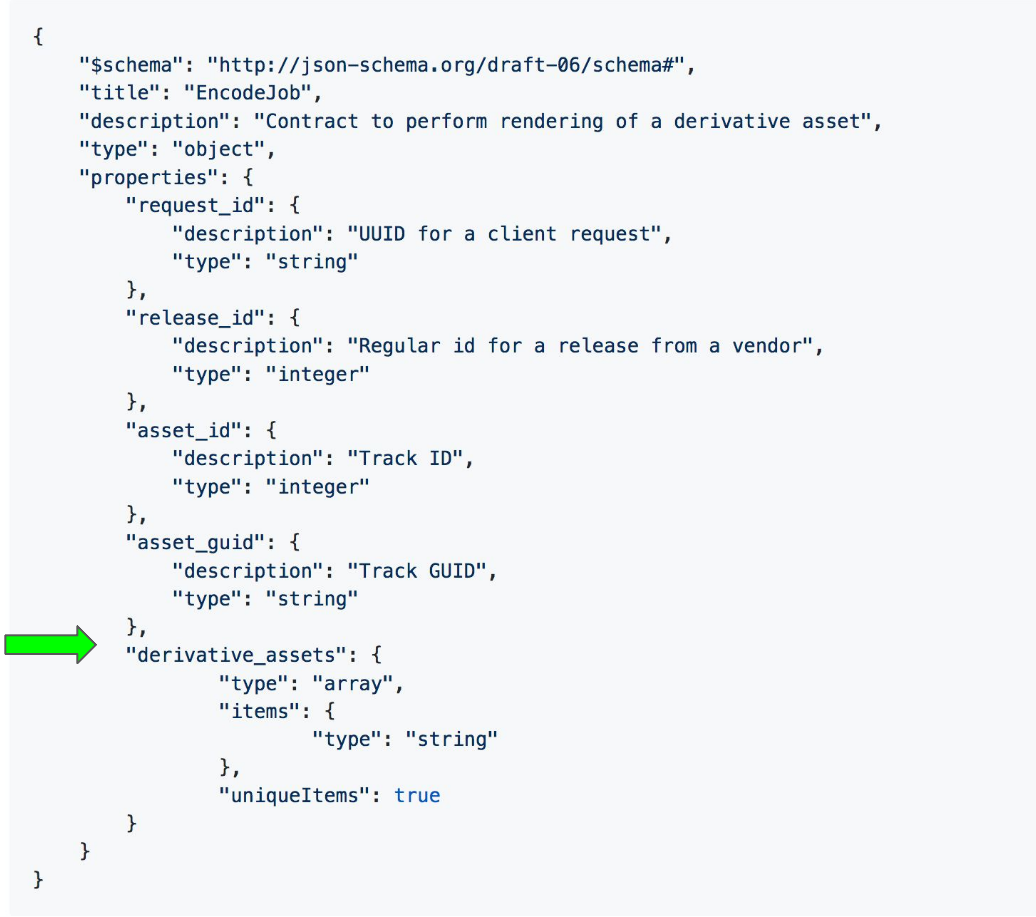
```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "title": "EncodeJob",
    "description": "Contract to perform rendering of a derivative asset",
    "type": "object",
    "properties": {
        "request_id": {
            "description": "UUID for a client request",
            "type": "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

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 :)

Hardware Assets

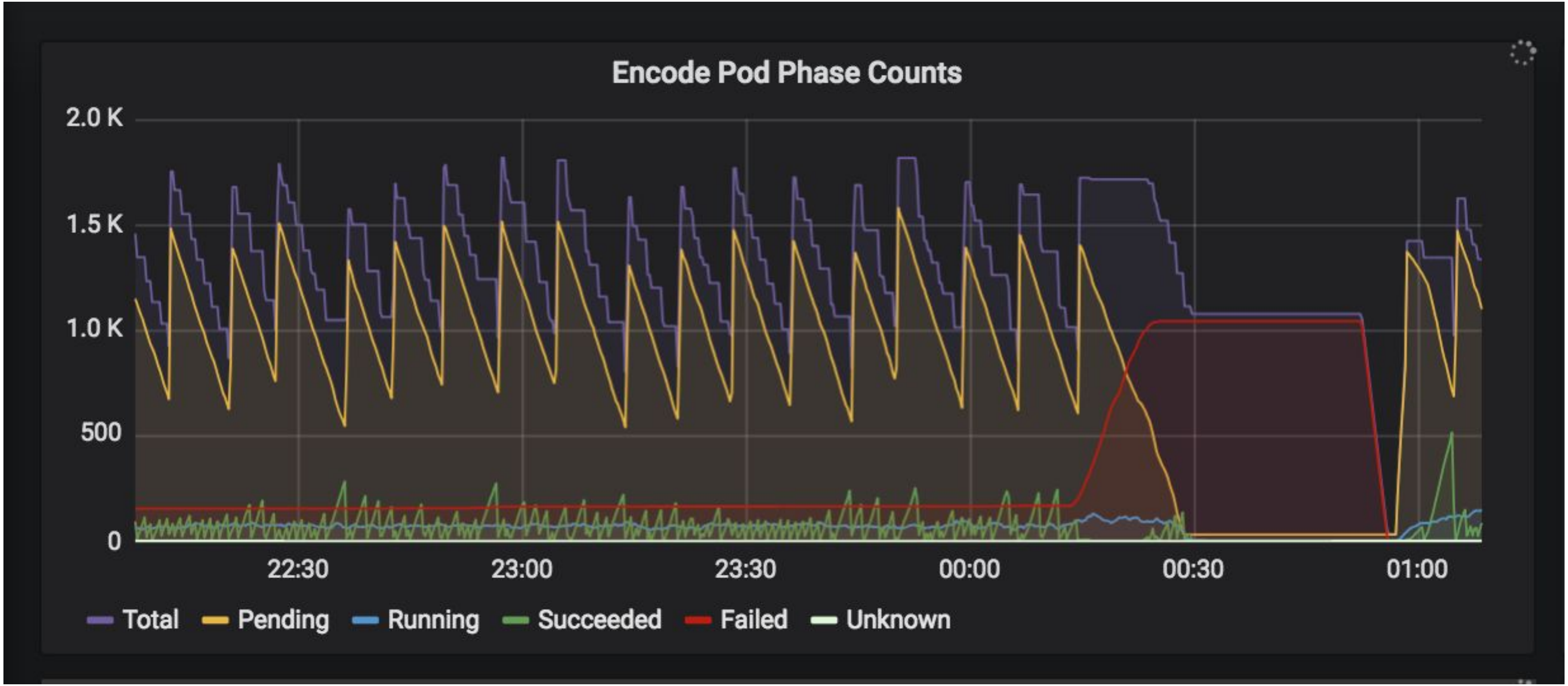
- 1 Rack
- 12 Dell R520 (2012)
- 1 Dell R410 (2009)
- 1 Dell R620 (2012)
- 4 Dell PE2950 (2006-2008)

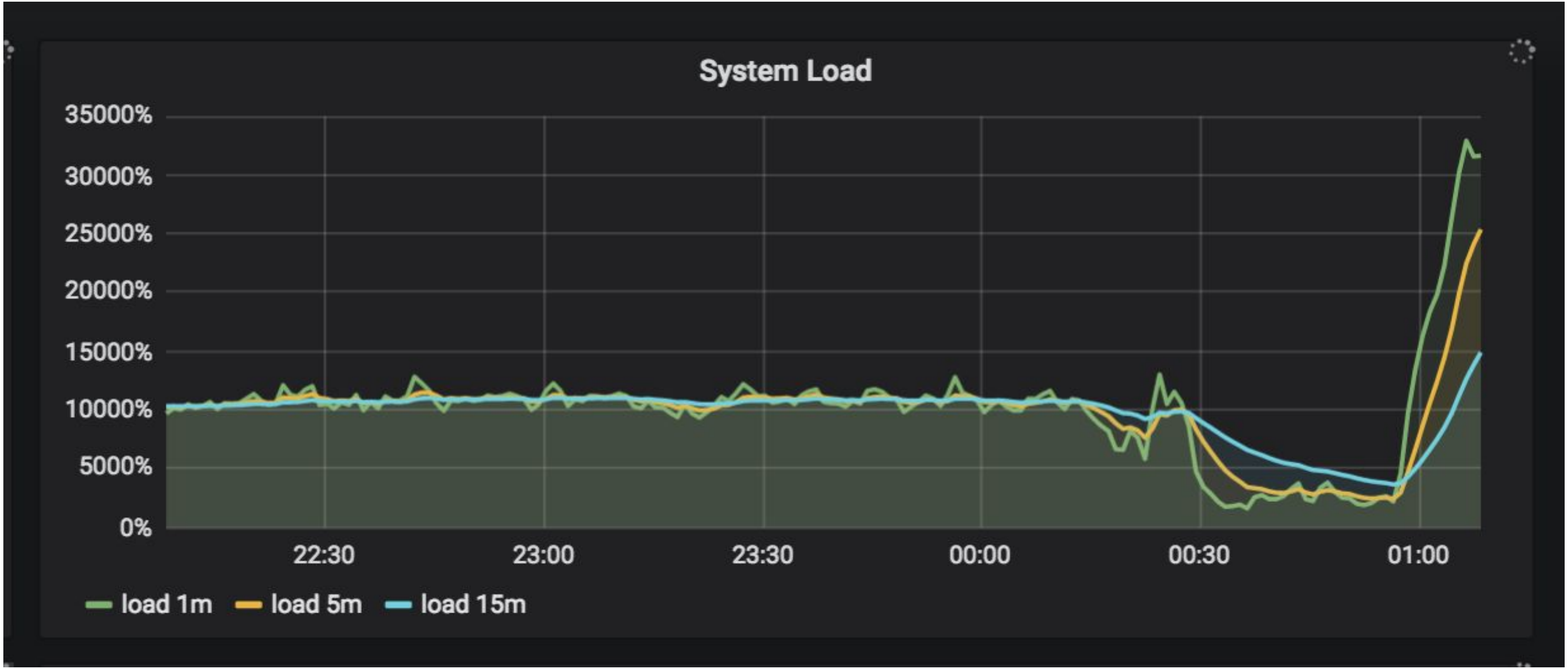
Kubernetes v1.11.0
Docker 18.03.1-ce

CentOS 7
Community maintained Linux Kernel (4.16)

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

- CPU Pinning
- Temporal Slicing
- Requests
- Limits
- https://hackernoon.com/job-concurrency-in-kubernetes-lxd-and-cpu-pinning-to-the-rescue-b9fb7b44f99d

A Problem With Jobs

- Kubernetes API had a saturation limit
- Past a certain threshold, CRUD operations became impossibly slow over time (particularly batch deletes)
- Bottlenecked not by CPU, but now by the API

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

- CronJob for garbage collection (Go K8S client)
- CronJob for dead-lettering failed tasks (Go K8S client)

Pods

- Free retries: already have a dead-lettering mechanism built into the queue, these can be triaged shoveled easily
- Acceptable performance
- Still a threshold for objects that should be in the API at our scale
- Change limits on max pods allowed in API

<Placeholder for Graphs>

Finalizing reporting

Conclusion

Questions