



Building Your First Application with Cassandra

Luke Tillman (@LukeTillman)

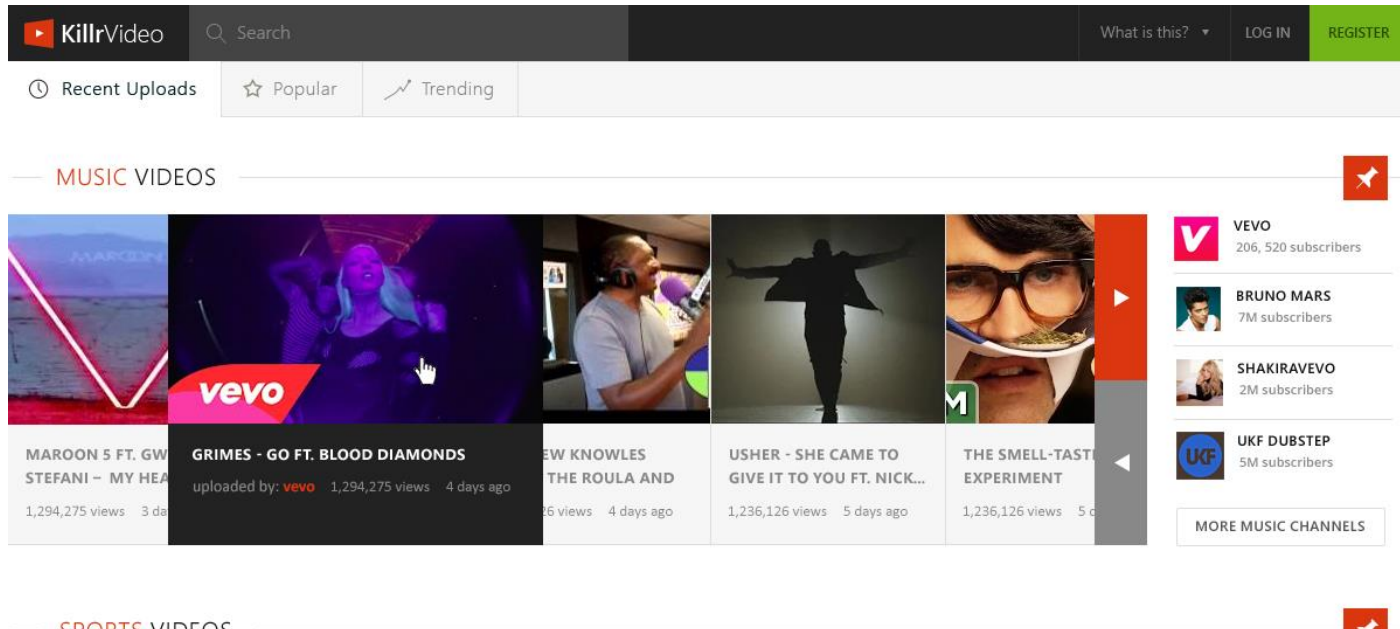
Language Evangelist at DataStax

Who are you?!

- Evangelist with a focus on the .NET Community
- Long-time .NET Developer
- Recently presented at Cassandra Summit 2014 with Microsoft



KillrVideo, a Video Sharing Site



- Think a YouTube competitor
 - Users add videos, rate them, comment on them, etc.
 - Can search for videos by tag

See the Live Demo, Get the Code

- Live demo available at <http://www.killrvideo.com>
 - Written in C#
 - Live Demo running in Azure
 - Open source: <https://github.com/luketillman/killrvideo-csharp>
- Interesting use case because of different data modeling challenges and the scale of something like YouTube
 - More than 1 billion unique users visit YouTube each month
 - 100 hours of video are uploaded to YouTube every minute

- 1** Think Before You Model
- 2** A Data Model for Cat Videos
- 3** Phase 2: Build the Application
- 4** Software Architecture, A Love Story
- 5** The Future

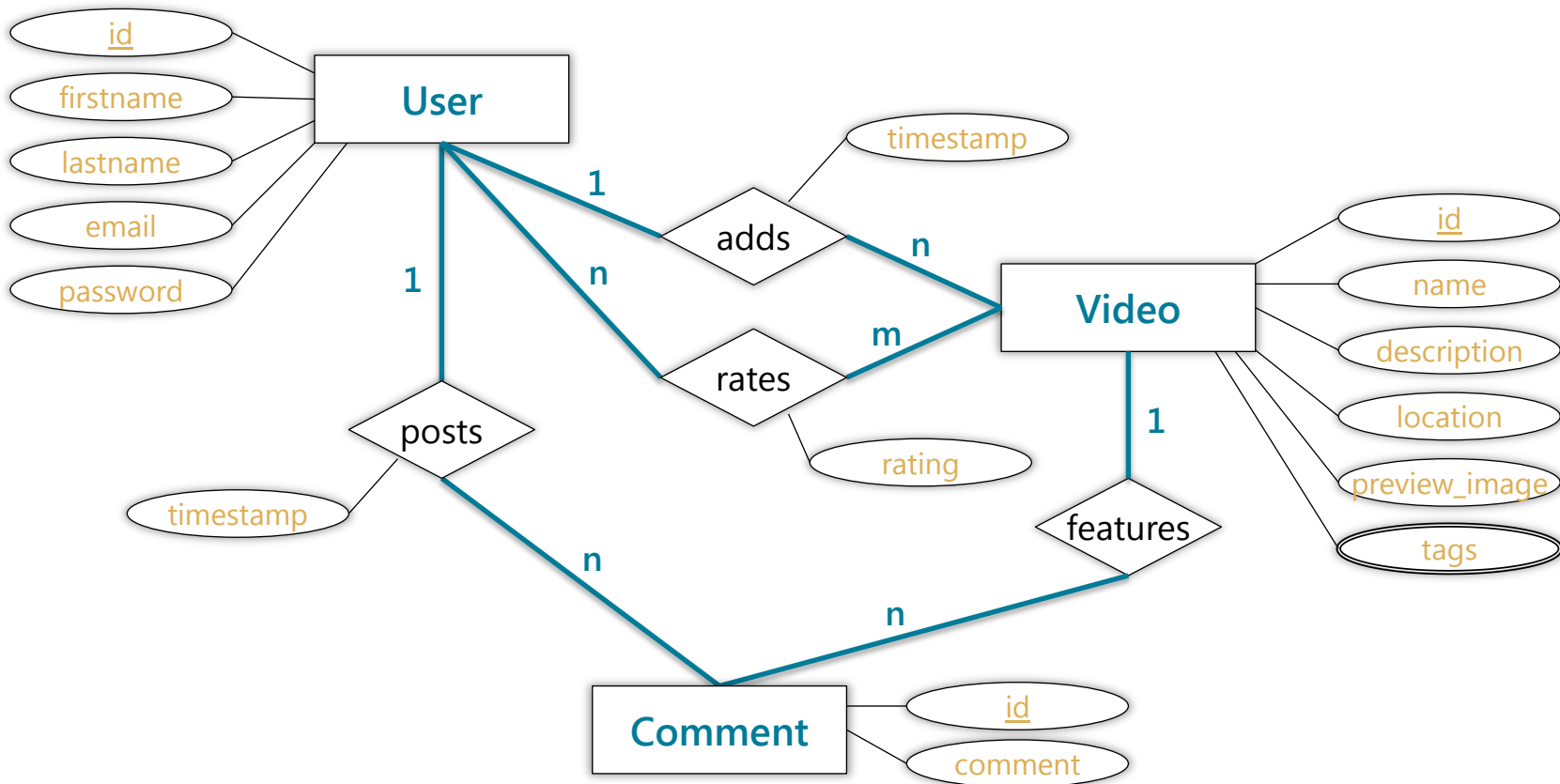
Think Before You Model

Or how to keep doing what you're already doing

Getting to Know Your Data

- What things do I have in the system?
- What are the relationships between them?
- This is your conceptual data model
- You already do this in the RDBMS world

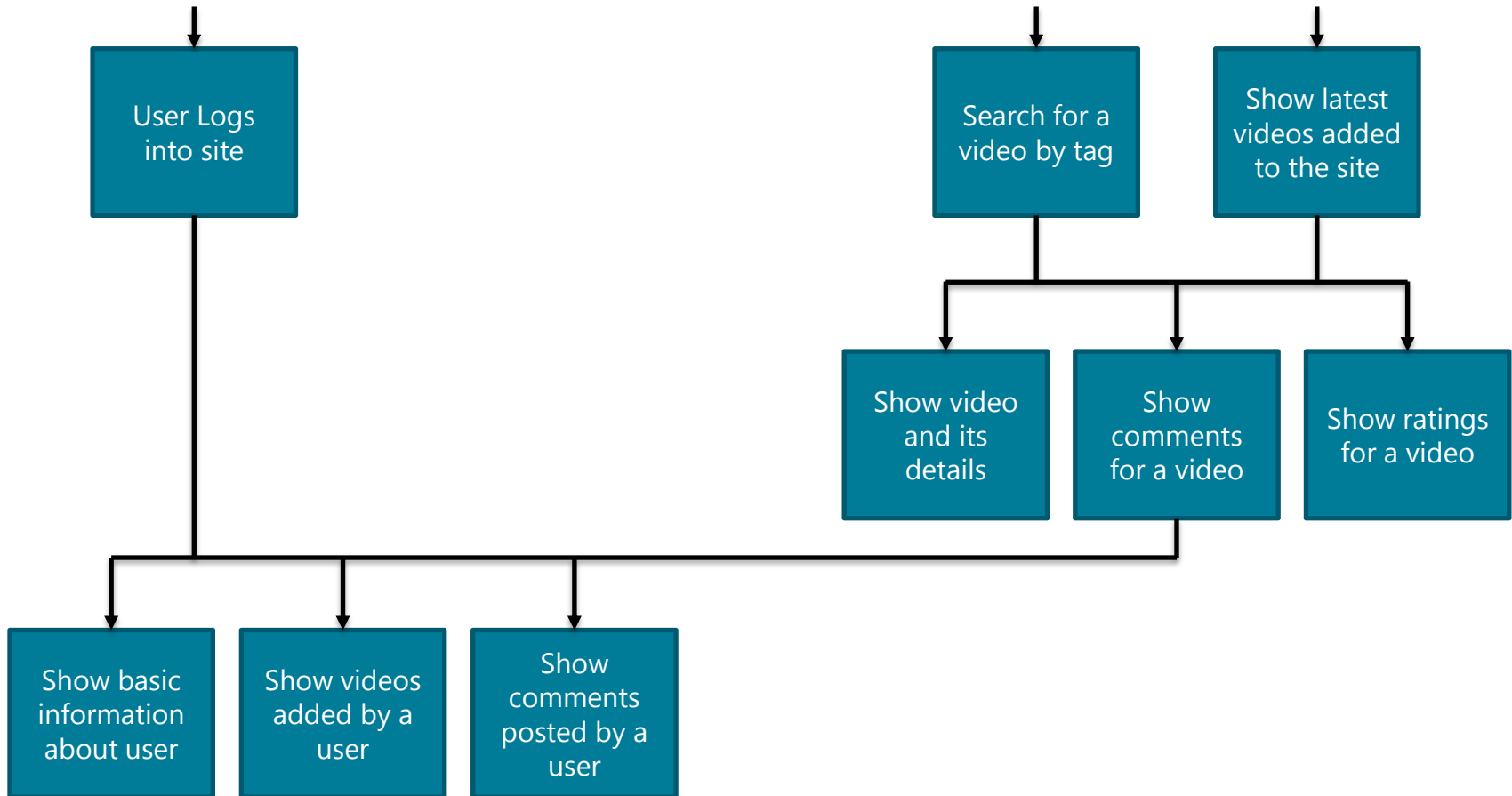
Some of the Entities and Relationships in KillrVideo



Getting to Know Your Queries

- What are your application's workflows?
- How will I access the data?
- Knowing your queries in advance is **NOT** optional
- Different from RDBMS because I can't just JOIN or create a new indexes to support new queries

Some Application Workflows in KillrVideo



Some Queries in KillrVideo to Support Workflows

Users

User Logs
into site

Find user by email
address

Show basic
information
about user

Find user by id

Comments

Show
comments
for a video

Find comments by
video (latest first)

Show
comments
posted by a
user

Find comments by
user (latest first)

Ratings

Show ratings
for a video

Find ratings by video

Some Queries in KillrVideo to Support Workflows

Videos

Search for a
video by tag

Find video by tag

Show latest
videos added
to the site

Find videos by date
(latest first)

Show video
and its
details

Find video by id

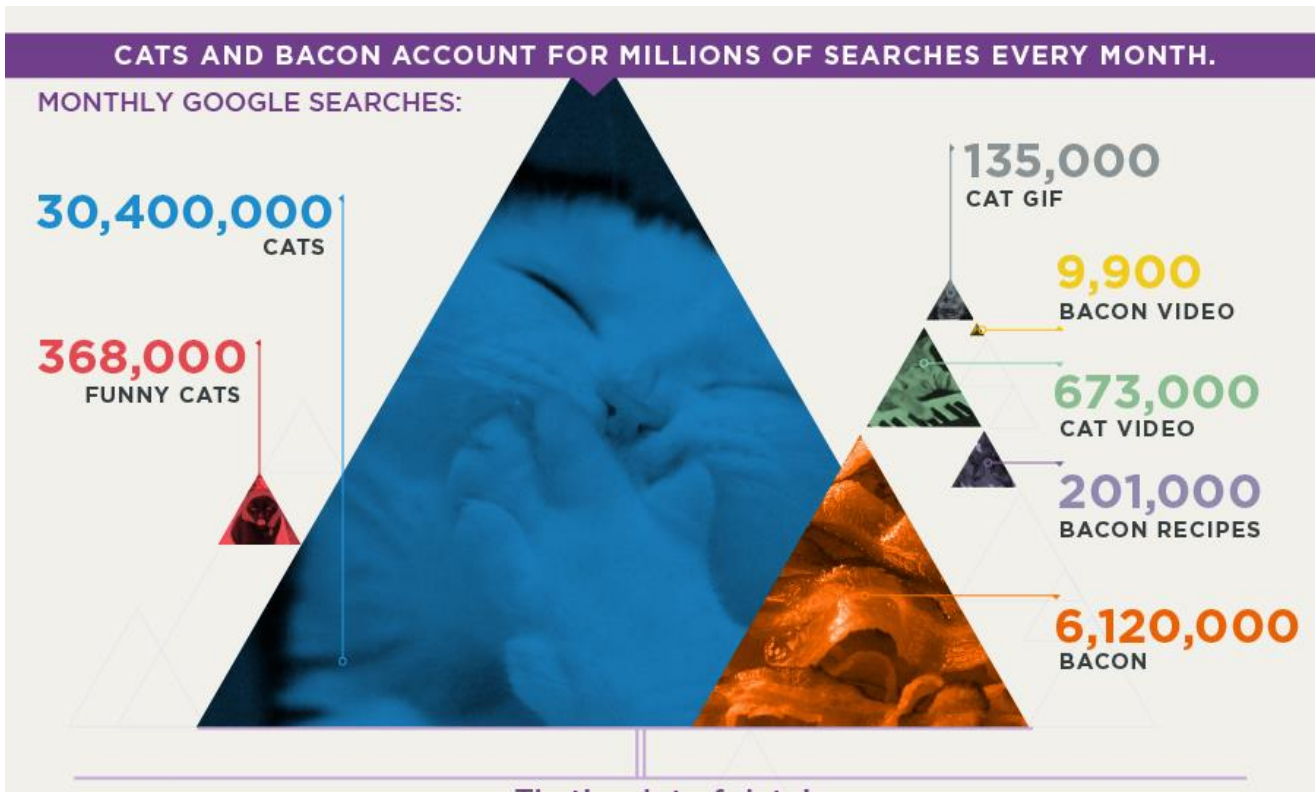
Show videos
added by a
user

Find videos by user
(latest first)

A Data Model for Cat Videos

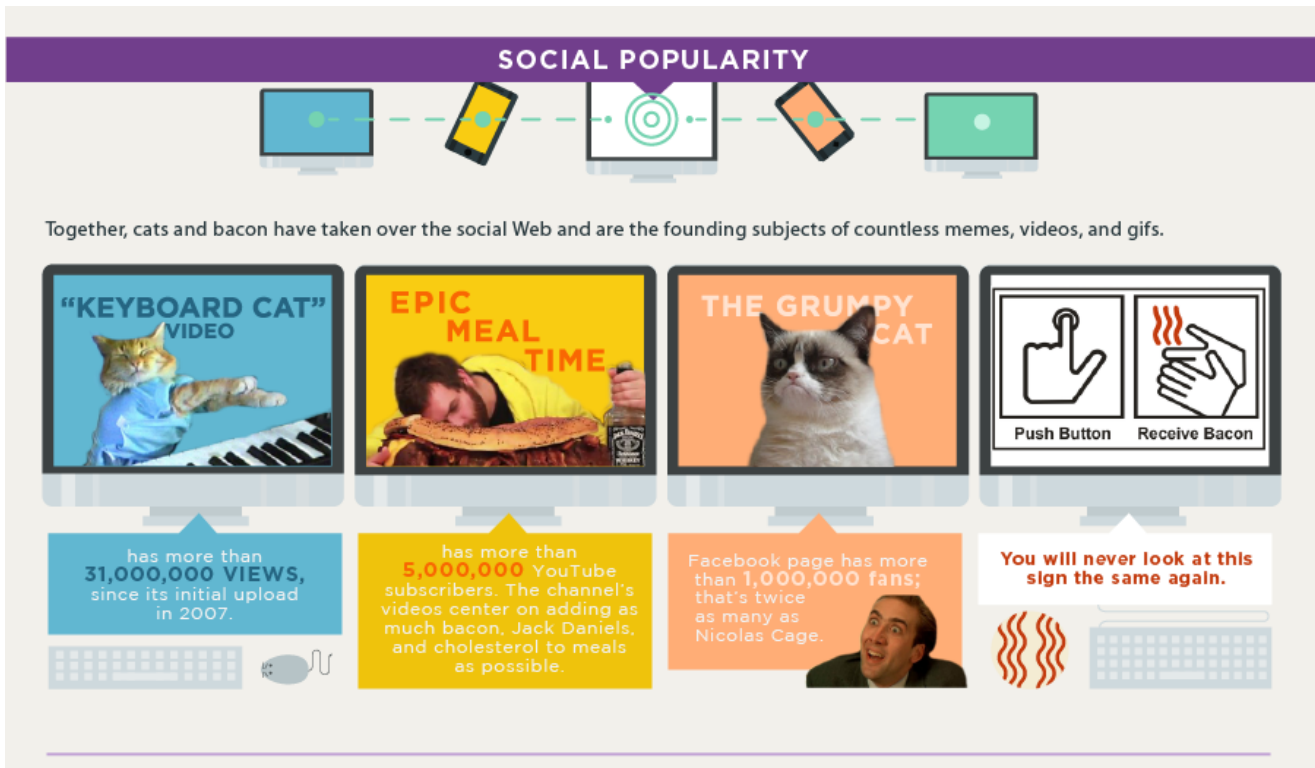
Because the Internet loves 'em some cat videos

Just How Popular are Cats on the Internet?



<http://mashable.com/2013/07/08/cats-bacon-rule-internet/>

Just How Popular are Cats on the Internet?



<http://mashable.com/2013/07/08/cats-bacon-rule-internet/>

- Cassandra limits us to queries that can scale across many nodes
 - Include value for Partition Key and optionally, Clustering Column(s)
- We know our queries, so we build tables to answer them
- Denormalize at write time to do as few reads as possible
- Many times we end up with a “table per query”
 - Similar to materialized views from the RDBMS world

Users – The Relational Way

User Logs
into site

Find user by email
address

Show basic
information
about user

Find user by id

- Single Users table with all user data and an Id Primary Key
- Add an index on email address to allow queries by email

Users – The Cassandra Way

User Logs
into site

Find user by email
address

```
CREATE TABLE user_credentials (  
  email text,  
  password text,  
  userid uuid,  
  PRIMARY KEY (email)  
);
```

Show basic
information
about user

Find user by id

```
CREATE TABLE users (  
  userid uuid,  
  firstname text,  
  lastname text,  
  email text,  
  created_date timestamp,  
  PRIMARY KEY (userid)  
);
```

Videos Everywhere!

Show video
and its
details

Find video by id

```
CREATE TABLE videos (  
  videoid uuid,  
  userid uuid,  
  name text,  
  description text,  
  location text,  
  location_type int,  
  preview_image_location text,  
  tags set<text>,  
  added_date timestamp,  
  PRIMARY KEY (videoid)  
);
```

Show videos
added by a
user

Find videos by user
(latest first)

```
CREATE TABLE user_videos (  
  userid uuid,  
  added_date timestamp,  
  videoid uuid,  
  name text,  
  preview_image_location text,  
  PRIMARY KEY (userid,  
              added_date, videoid)  
)  
WITH CLUSTERING ORDER BY (  
  added_date DESC,  
  videoid ASC);
```

Videos Everywhere!

Search for a
video by tag

Find video by tag

Show latest
videos added
to the site

Find videos by date
(latest first)

Considerations When Duplicating Data

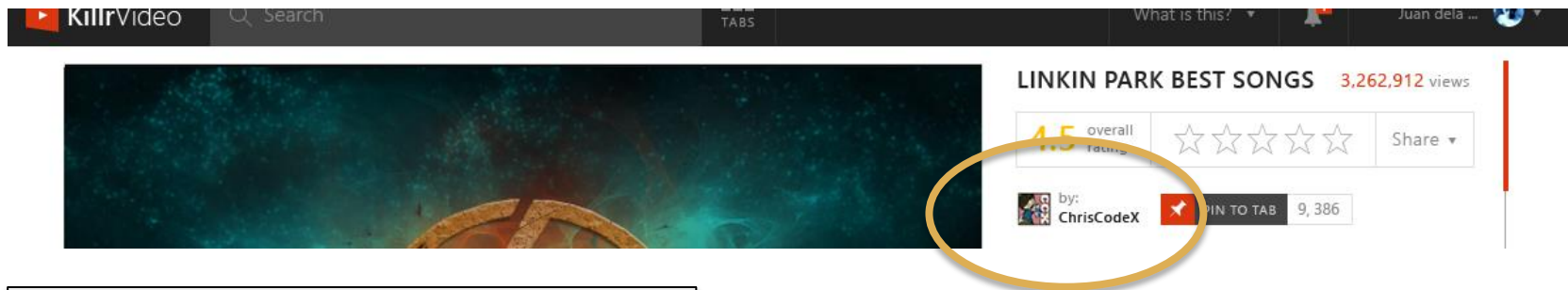
- Can the data change?
- How likely is it to change or how frequently will it change?
- Do I have all the information I need to update duplicates and maintain consistency?

Modeling Relationships – Collection Types

- Cassandra doesn't support JOINS, but your data will still have relationships (and you can still model that in Cassandra)
- One tool available is CQL collection types

```
CREATE TABLE videos (  
    videoid uuid,  
    userid uuid,  
    name text,  
    description text,  
    location text,  
    location_type int,  
    preview_image_location text,  
    tags set<text>,  
    added_date timestamp,  
    PRIMARY KEY (videoid)  
);
```

Modeling Relationships – Client Side Joins

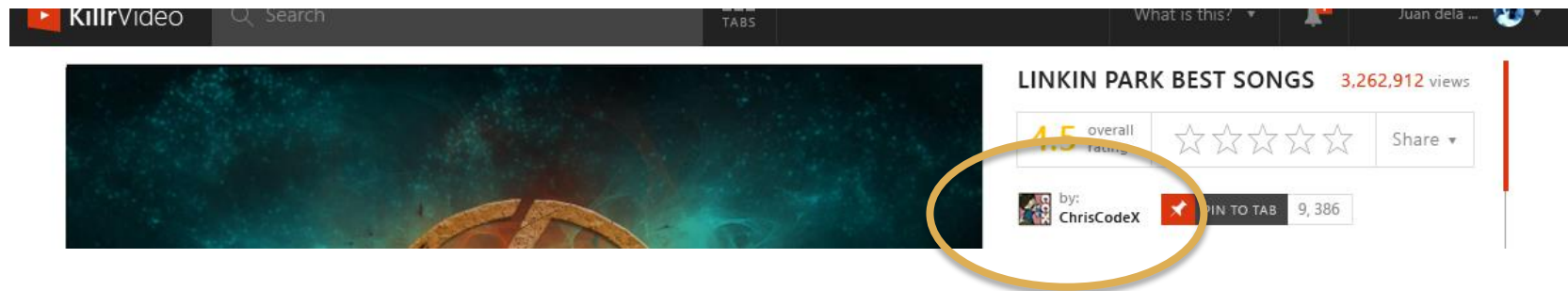


```
CREATE TABLE videos (  
  videoid uuid,  
  userid uuid,  
  name text,  
  description text,  
  location text,  
  location_type int,  
  preview_image_location text,  
  tags set<text>,  
  added_date timestamp,  
  PRIMARY KEY (videoid)  
);
```

Currently requires query for video,
followed by query for user by id based
on results of first query

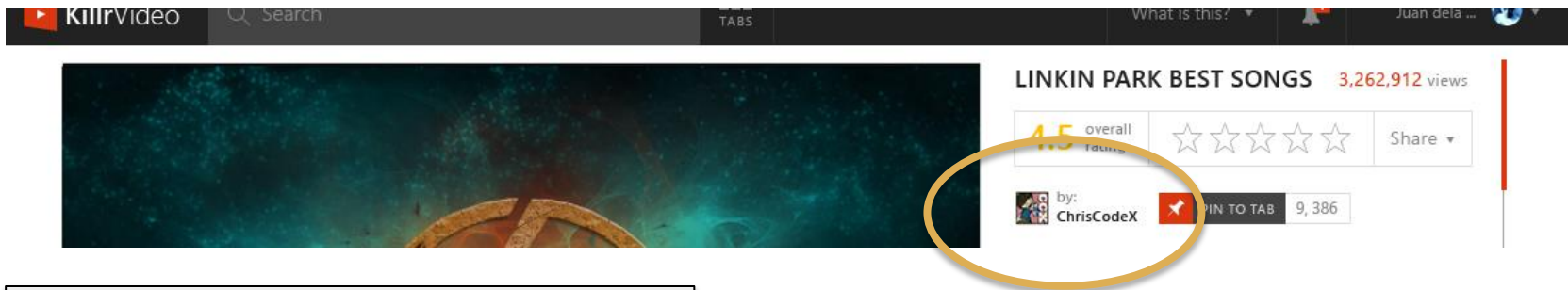
```
CREATE TABLE users (  
  userid uuid,  
  firstname text,  
  lastname text,  
  email text,  
  created_date timestamp,  
  PRIMARY KEY (userid)  
);
```

Modeling Relationships – Client Side Joins



- What is the cost? Might be OK in small situations
- Do **NOT** scale
- Avoid when possible

Modeling Relationships – Client Side Joins

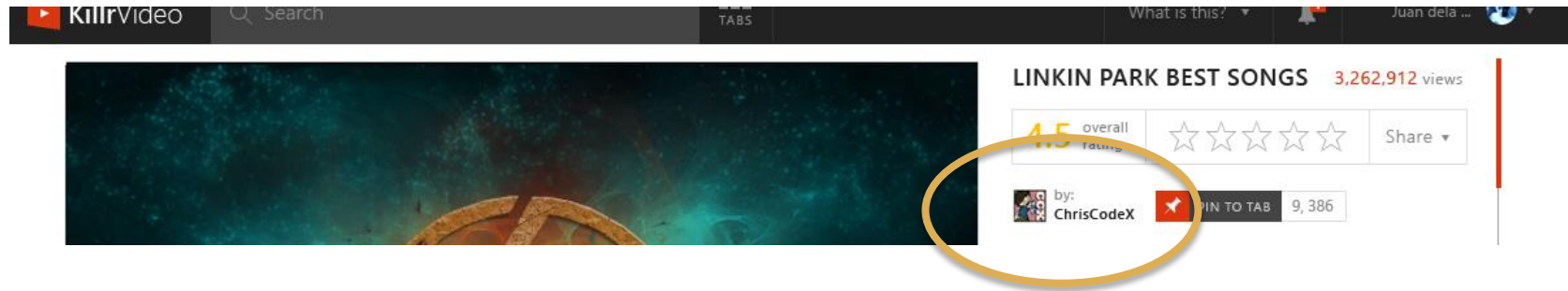


```
CREATE TABLE videos (  
  videoid uuid,  
  userid uuid,  
  name text,  
  description text,  
  ...  
  user_firstname text,  
  user_lastname text,  
  user_email text,  
  PRIMARY KEY (videoid)  
);
```

or

```
CREATE TABLE users_by_video (  
  videoid uuid,  
  userid uuid,  
  firstname text,  
  lastname text,  
  email text,  
  PRIMARY KEY (videoid)  
);
```


Modeling Relationships – Client Side Joins


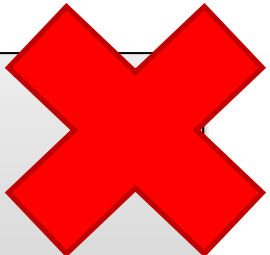


- Remember the considerations when you duplicate data
- What happens if a user changes their name or email address?
- Can I update the duplicated data?

Cassandra Rules Can Impact Your Design

- Video Ratings – use counters to track sum of all ratings and count of ratings

```
CREATE TABLE videos (  
  videoid uuid,  
  userid uuid,  
  name text,  
  description text,  
  ...  
  rating_counter counter,  
  rating_total counter,  
  PRIMARY KEY (videoid)  
);
```



```
CREATE TABLE video_ratings (  
  videoid uuid,  
  rating_counter counter,  
  rating_total counter,  
  PRIMARY KEY (videoid)  
);
```

- Counters are a good example of something with special rules

Single Nodes Have Limits Too

Show latest
videos added
to the site

Find videos by date
(latest first)

```
CREATE TABLE latest_videos (  
  yyyyymmdd text,  
  added_date timestamp,  
  videoid uuid,  
  name text,  
  preview_image_location text,  
  PRIMARY KEY (yyyyymmdd,  
               added_date, videoid)  
) WITH CLUSTERING ORDER BY (  
  added_date DESC,  
  videoid ASC  
);
```

- Latest videos are bucketed by day
- Means all reads/writes to latest videos are going to same partition (and thus the same nodes)
- Could create a hotspot

Single Nodes Have Limits Too

Show latest videos added to the site

Find videos by date (latest first)

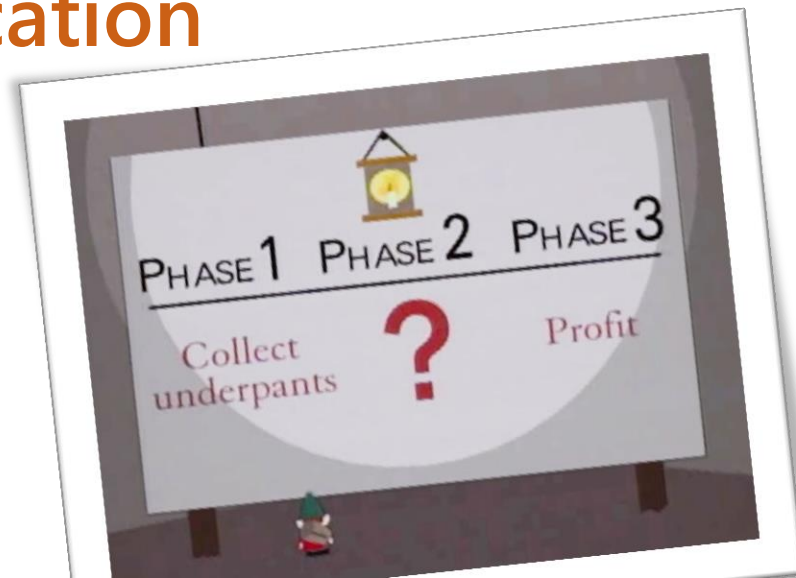
```
CREATE TABLE latest_videos (  
  yyyyymmdd text,  
  bucket_number int,  
  added_date timestamp,  
  videoid uuid,  
  name text,  
  preview_image_location text,  
  PRIMARY KEY (  
    (yyyyymmdd, bucket_number)  
    added_date, videoid)  
  ) ...
```

- Mitigate by adding data to the Partition Key to spread load
- Data that's already naturally a part of the domain
 - Latest videos by category?
- Arbitrary data, like a bucket number
 - Round robin at the app level

Phase 1: Data Model

Phase 2: Build the Application

Phase 3: Profit



The DataStax Drivers for Cassandra

- Open source, Apache 2 licensed, available on GitHub
 - <https://github.com/datastax/>
- Currently Available
 - C# (.NET)
 - Python
 - Java
 - NodeJS
 - Ruby
 - C++
- Will Probably Happen
 - PHP
 - Scala
 - JDBC
- Early Discussions
 - Go
 - Rust

The DataStax Drivers for Cassandra

Language	Bootstrapping Code
C#	<pre>Cluster cluster = Cluster.Builder().AddContactPoint("127.0.0.1").Build(); ISession session = cluster.Connect("killrvideo");</pre>
Python	<pre>from cassandra.cluster import Cluster cluster = Cluster(contact_points=['127.0.0.1']) session = cluster.connect('killrvideo')</pre>
Java	<pre>Cluster cluster = Cluster.builder().addContactPoint("127.0.0.1").build(); Session session = cluster.connect("killrvideo");</pre>
NodeJS	<pre>var cassandra = require('cassandra-driver'); var client = new cassandra.Client({ contactPoints: ['127.0.0.1'], keyspace: 'killrvideo' });</pre>

Use Prepared Statements

- Performance optimization for queries you run repeatedly
- Pay the cost of preparing once (causes roundtrip to Cassandra)
- KillrVideo: looking a user's credentials up by email address

```
PreparedStatement prepared = session.Prepare(  
    "SELECT * FROM user_credentials WHERE email = ?");
```

- Save and reuse the PreparedStatement instance after preparing

Use Prepared Statements

- Bind variable values when ready to execute

```
BoundStatement bound = prepared.Bind("luke.tillman@datastax.com");  
RowSet rows = await _session.ExecuteAsync(bound);
```

- Execution only has to send variable values over the wire
- Cassandra doesn't have to reparse the CQL string each time
- Remember: Prepare **once**, bind and execute **many**

Batch Statements: Use and Misuse

- You can mix and match Simple/Bound statements in a batch
- Batches are Logged (atomic) by default
- Use when you want a group of mutations (statements) to all succeed or all fail (denormalizing at write time)
- Large batches are an anti-pattern (Cassandra will warn you)
- **Not** a performance optimization for bulk-loading data

KillrVideo: Update a Video's Name with a Batch

```
public class VideoCatalogDataAccess
{
    public VideoCatalogDataAccess(ISession session)
    {
        _session = session;
        _prepared = _session.Prepare(
            "UPDATE user_videos SET name = ? WHERE userid = ? AND videoid = ?");
    }

    public async Task UpdateVideoName(UpdateVideoDto video)
    {
        BoundStatement bound = _prepared.Bind(video.Name, video.UserId, video.VideoId);
        var simple = new SimpleStatement("UPDATE videos SET name = ? WHERE videoid = ?",
            video.Name, video.VideoId);

        // Use an atomic batch to send over all the mutations
        var batchStatement = new BatchStatement();
        batchStatement.Add(bound);
        batchStatement.Add(simple);
        RowSet rows = await _session.ExecuteAsync(batch);
    }
}
```

Lightweight Transactions when you need them

- Use when you don't want writes to step on each other
 - Sometimes called Linearizable Consistency
 - Similar to Serial Isolation Level from RDBMS
- Essentially a Check and Set (CAS) operation using Paxos
- **Read the fine print:** has a latency cost associated with it
- The canonical example: unique user accounts

KillrVideo: LWT to create user accounts

- Returns a column called **[applied]** indicating success/failure

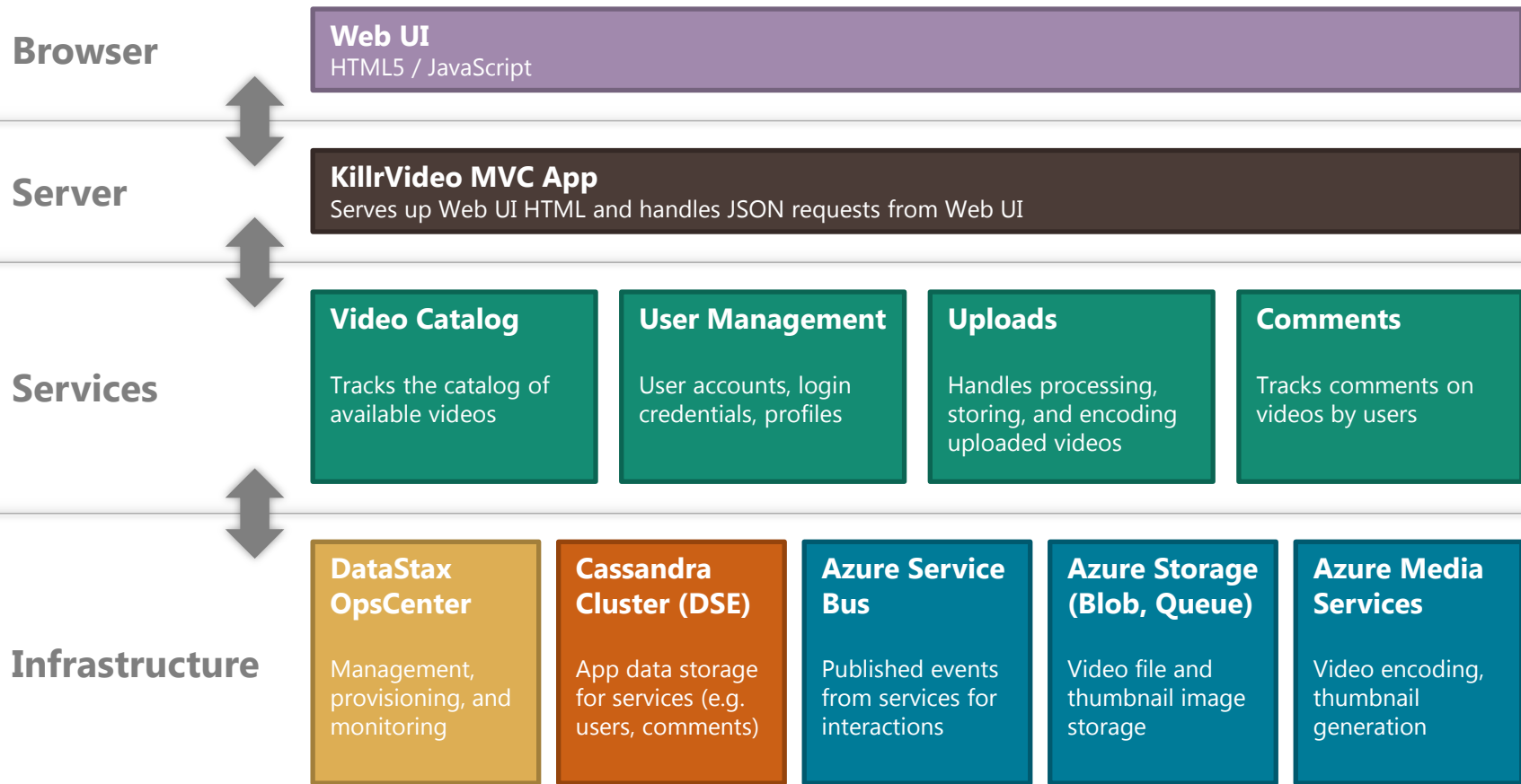
```
string cql = "INSERT INTO user_credentials (email, password, userid)" +  
            "VALUES (?, ?, ?) IF NOT EXISTS";  
  
var statement = new SimpleStatement(cql, user.Email, hashedPassword, user.UserId);  
  
RowSet rows = await _session.ExecuteAsync(statement);  
var userInserted = rows.Single().GetValue<bool>("[applied]");
```

- Different from relational world where you might expect an Exception (i.e. PrimaryKeyViolationException or similar)

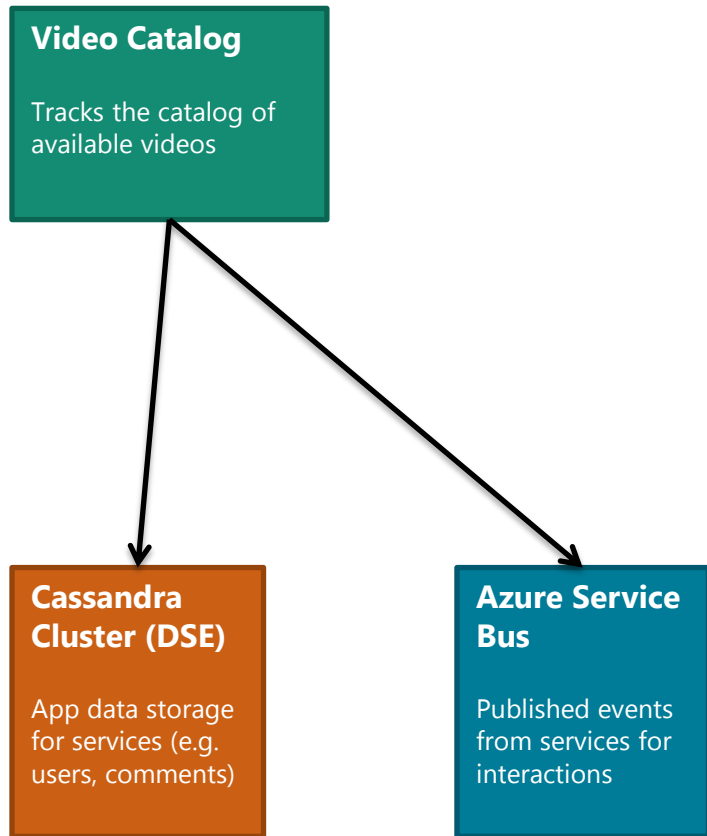
Software Architecture, A Love Story

Disclaimer: I am not paid to be a software architect

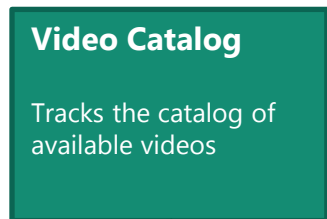
KillrVideo Logical Architecture



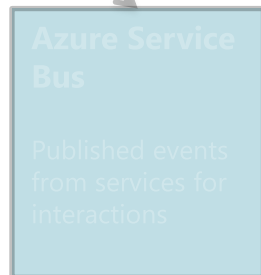
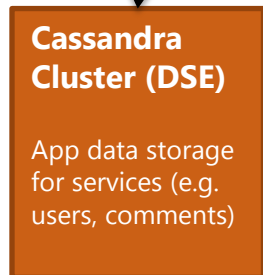
Inside a Simple Service: Video Catalog



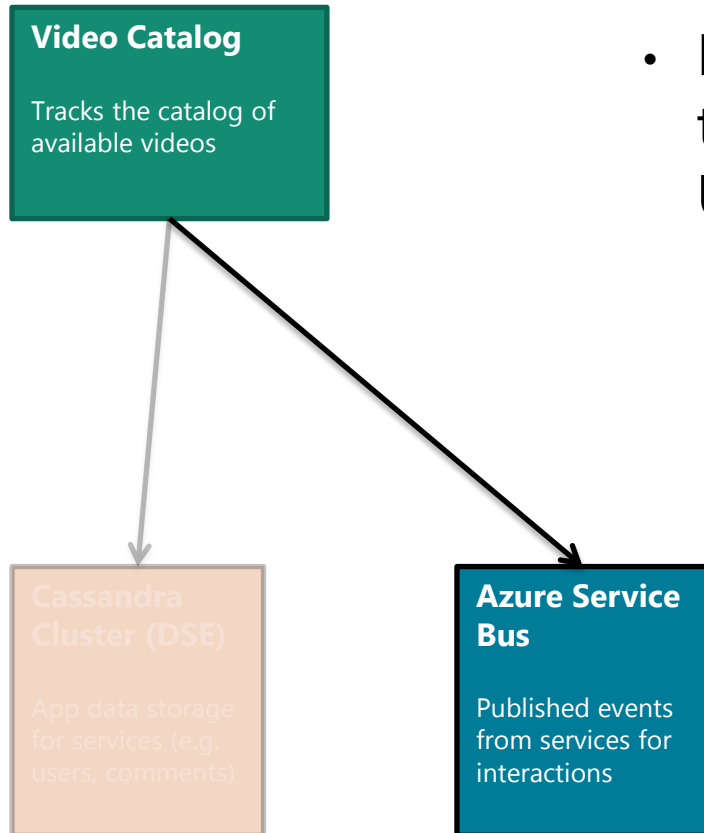
Inside a Simple Service: Video Catalog



- Stores metadata about videos in Cassandra (e.g. name, description, location, thumbnail location, etc.)

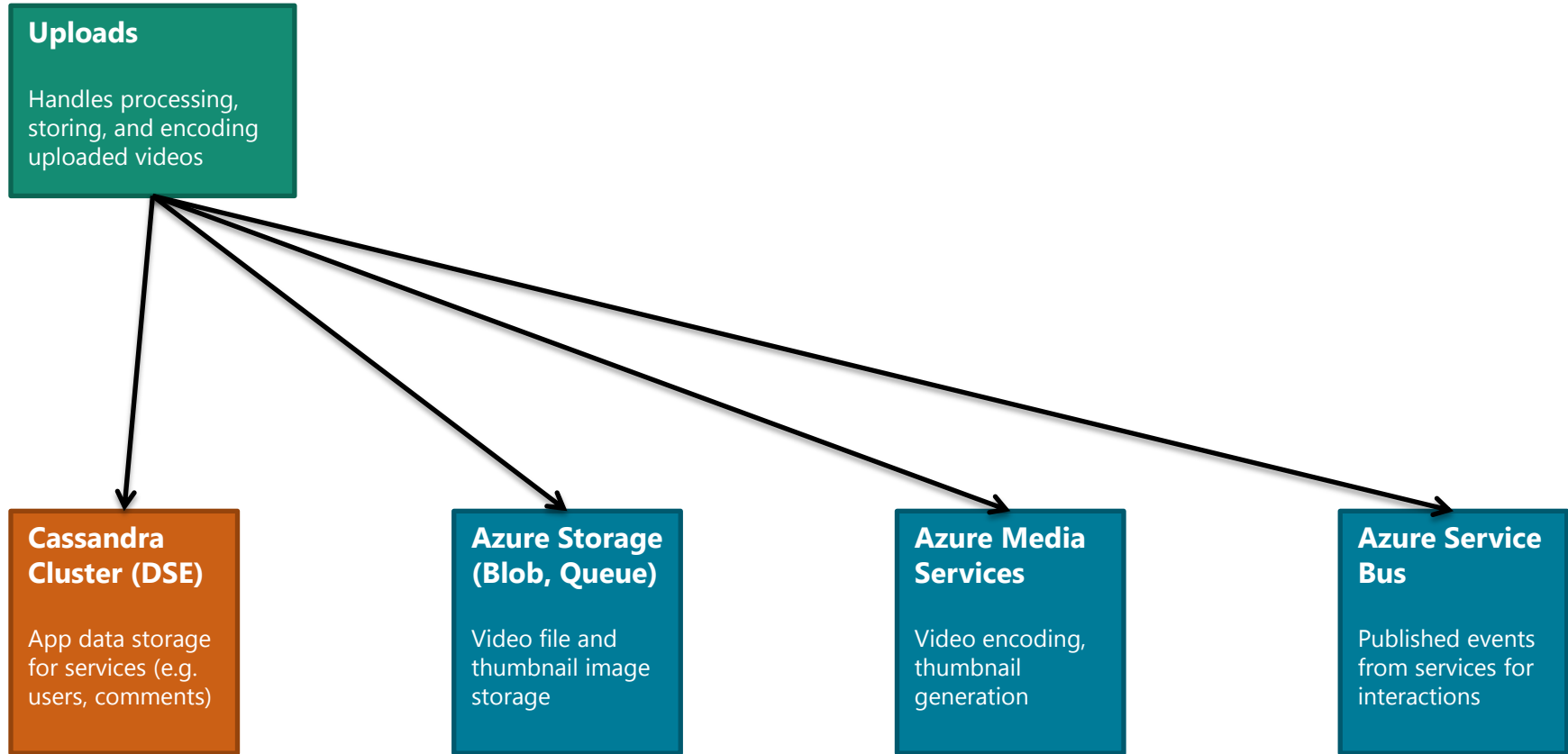


Inside a Simple Service: Video Catalog



- Publishes events about interesting things that happen (e.g. YouTubeVideoAdded, UploadedVideoAccepted, etc.)

Inside a More Complicated Service: Uploads

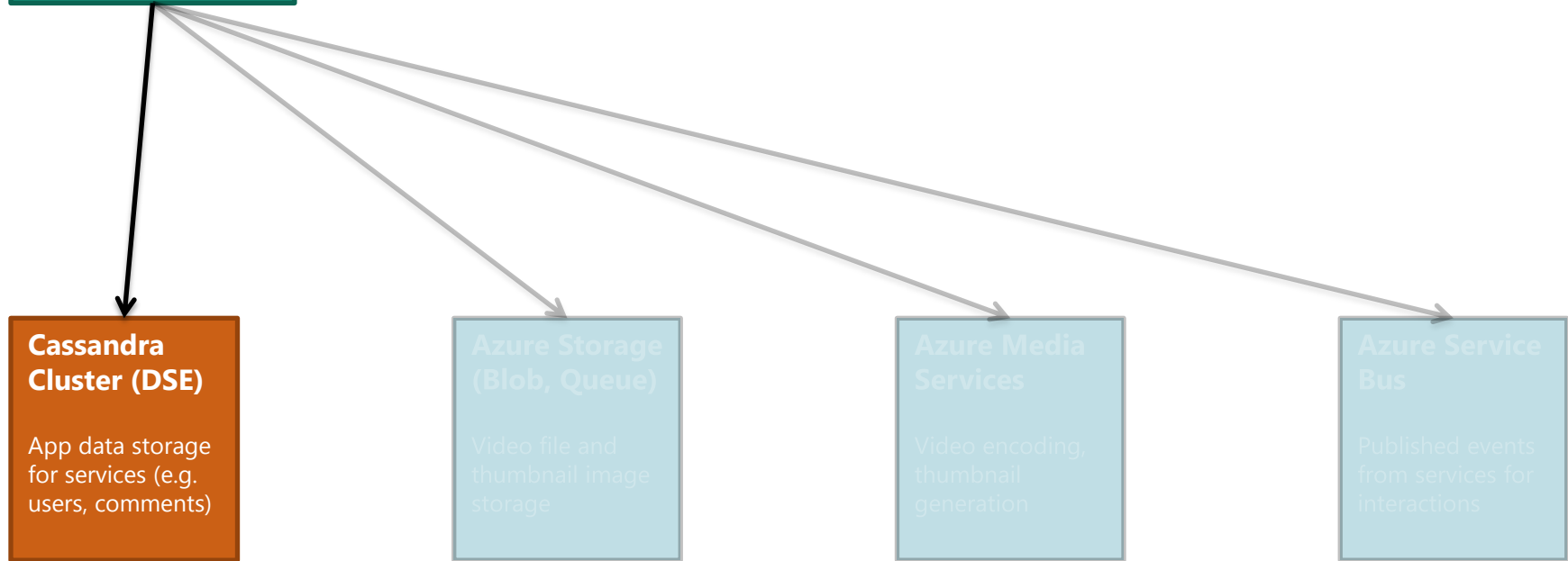


Inside a More Complicated Service: Uploads

Uploads

Handles processing, storing, and encoding uploaded videos

- Stores data about uploaded video file locations, encoding jobs, job status, etc.

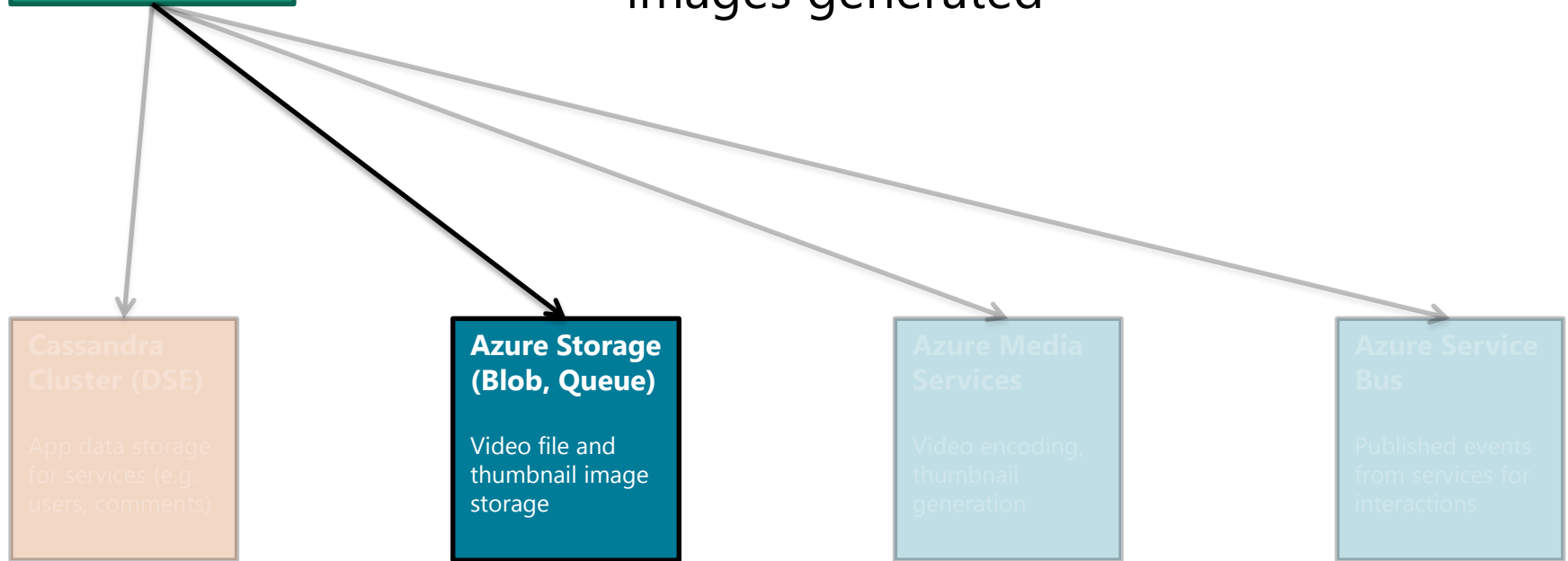


Inside a More Complicated Service: Uploads

Uploads

Handles processing, storing, and encoding uploaded videos

- Stores original and re-encoded video file assets, as well as thumbnail preview images generated

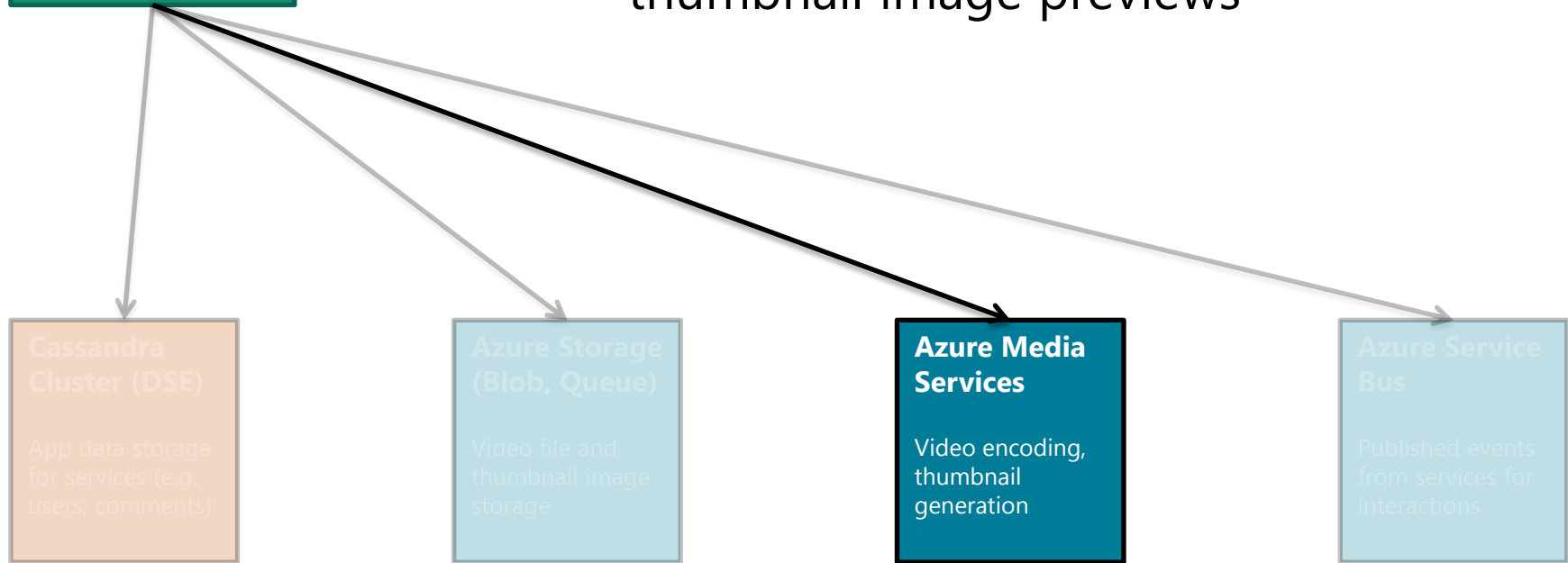


Inside a More Complicated Service: Uploads

Uploads

Handles processing, storing, and encoding uploaded videos

- Re-encodes uploaded videos to format suitable for the web, generates thumbnail image previews

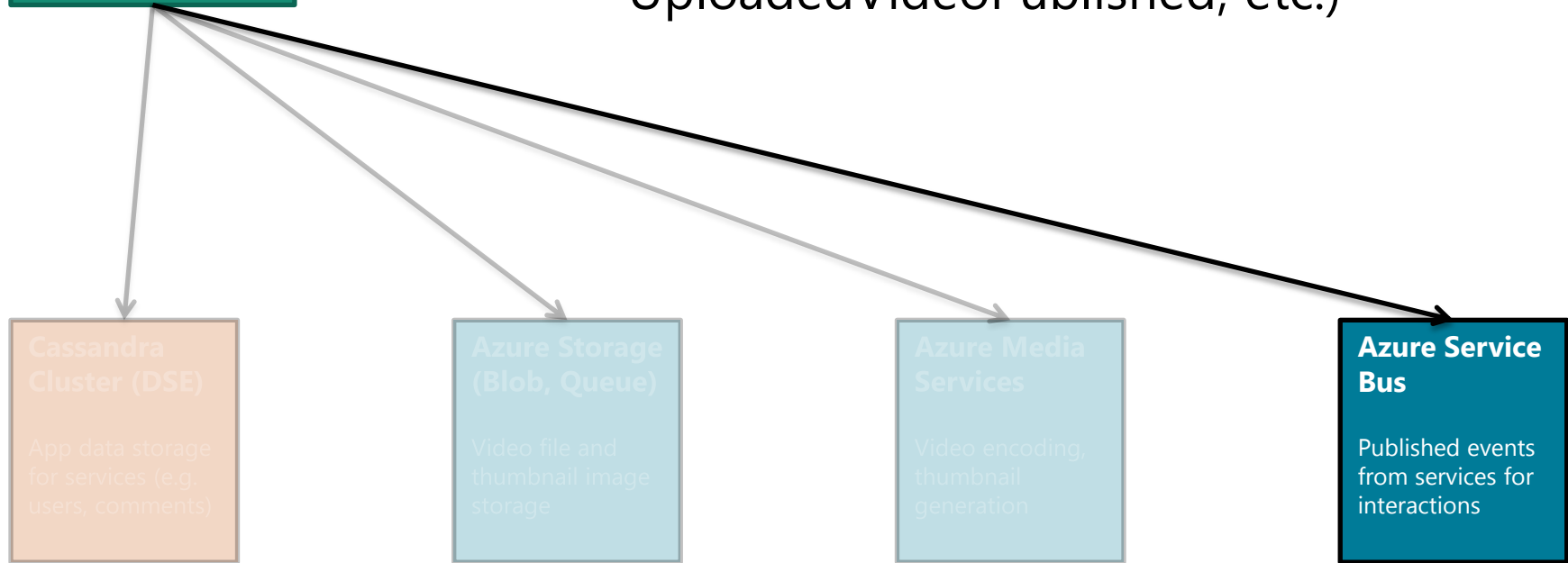


Inside a More Complicated Service: Uploads

Uploads

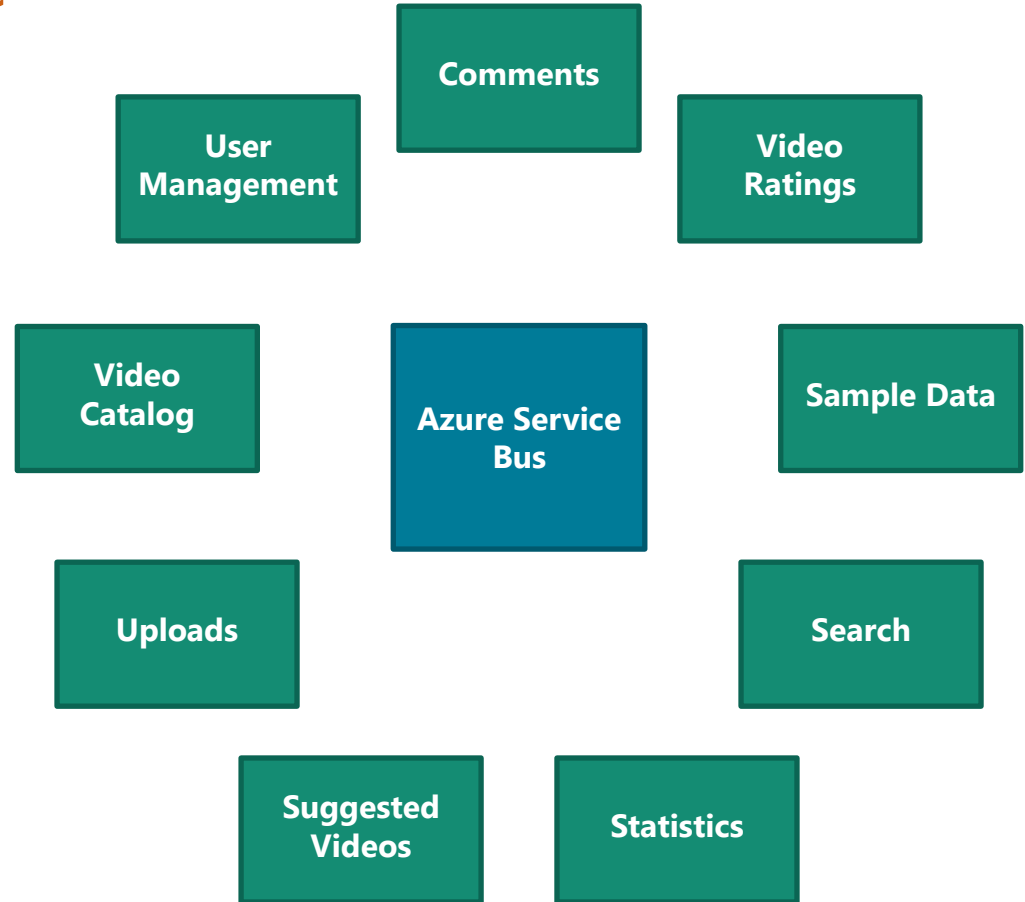
Handles processing, storing, and encoding uploaded videos

- Publishes events about interesting things that happen (e.g. `UploadedVideoPublished`, etc.)



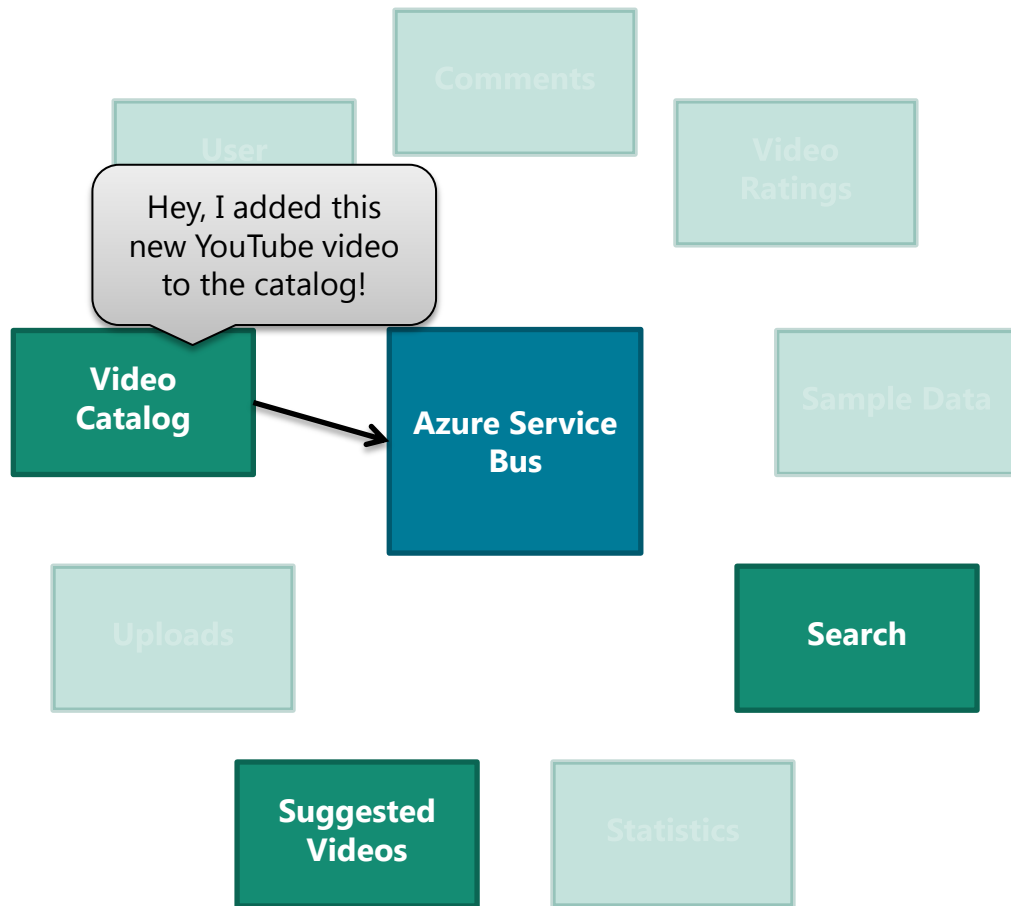
Event Driven Architecture

- Only the application(s) give commands
- Decoupled: Pub-sub messaging to tell other parts of the system something interesting happened
- Services could be deployed, scaled, and versioned independently (AKA microservices)



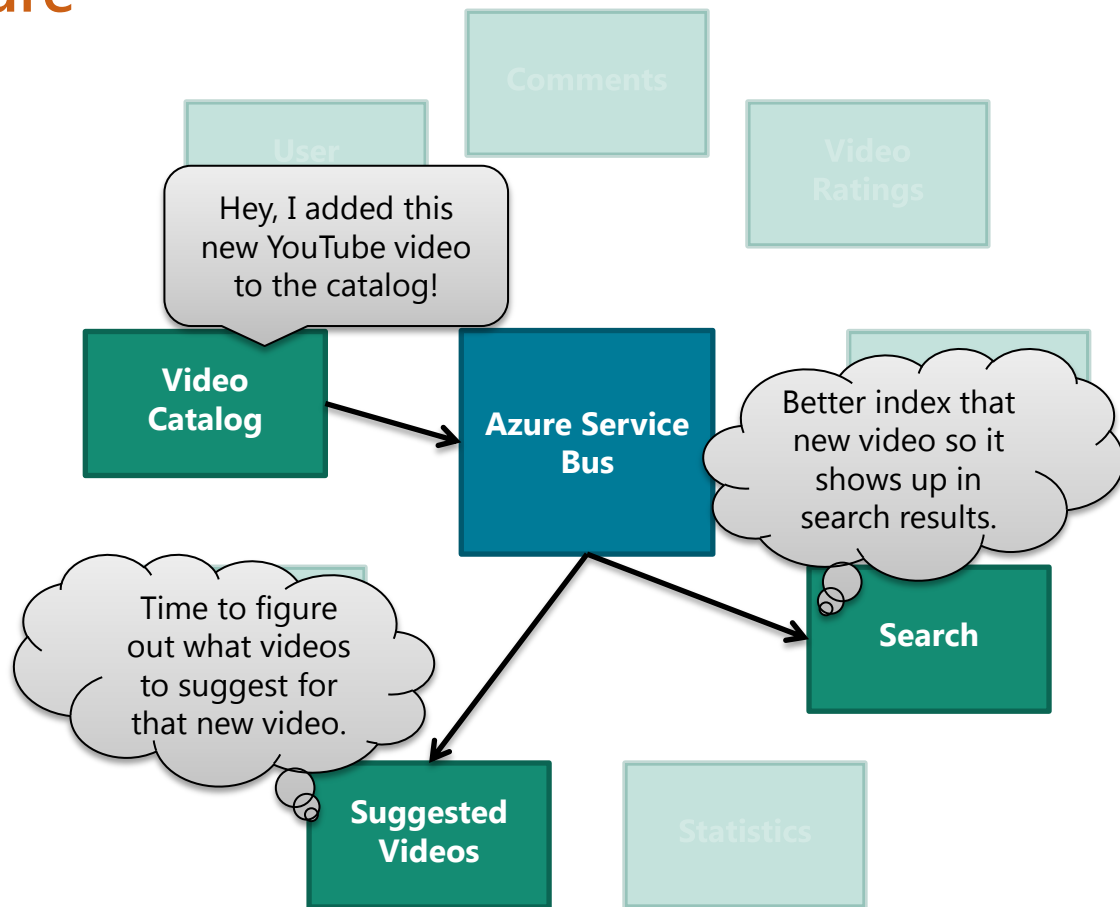
Event Driven Architecture

- Only the application(s) give commands
- Decoupled: Pub-sub messaging to tell other parts of the system something interesting happened
- Services could be deployed, scaled, and versioned independently (AKA microservices)



Event Driven Architecture

- Only the application(s) give commands
- Decoupled: Pub-sub messaging to tell other parts of the system something interesting happened
- Services could be deployed, scaled, and versioned independently (AKA microservices)



The Future

In the year 3,000...

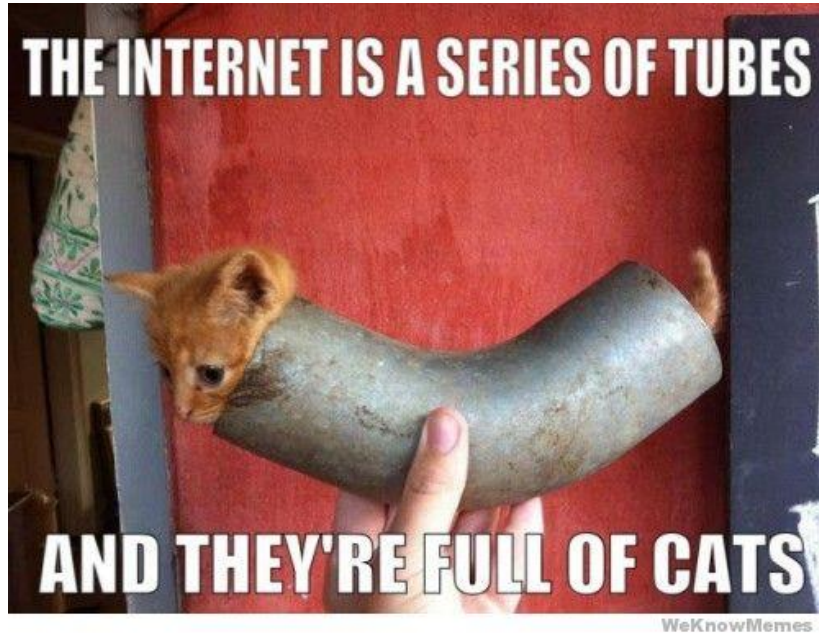
The Future, Conan?



Where do we go with KillrVideo from here?

- Spark or AzureML for video suggestions
- Video search via Solr
- Actors that store state in C* (Akka.NET or Orleans)
- Storing file data (thumbnails, profile pics) in C* using pithos

Questions?



Follow me on Twitter for updates or to ask questions later: @LukeTillman