Microservices Architecture: Building Scalable (Library) Software Solutions

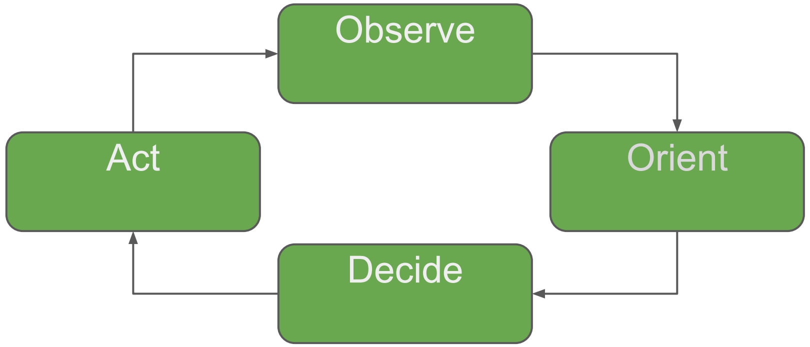
Jason Varghese
(New York Public Library)

Microservices

- Buzzword compliant
- Not a microservices “sales” talk
- Explore microservice principles
- Explore Use Cases/Examples

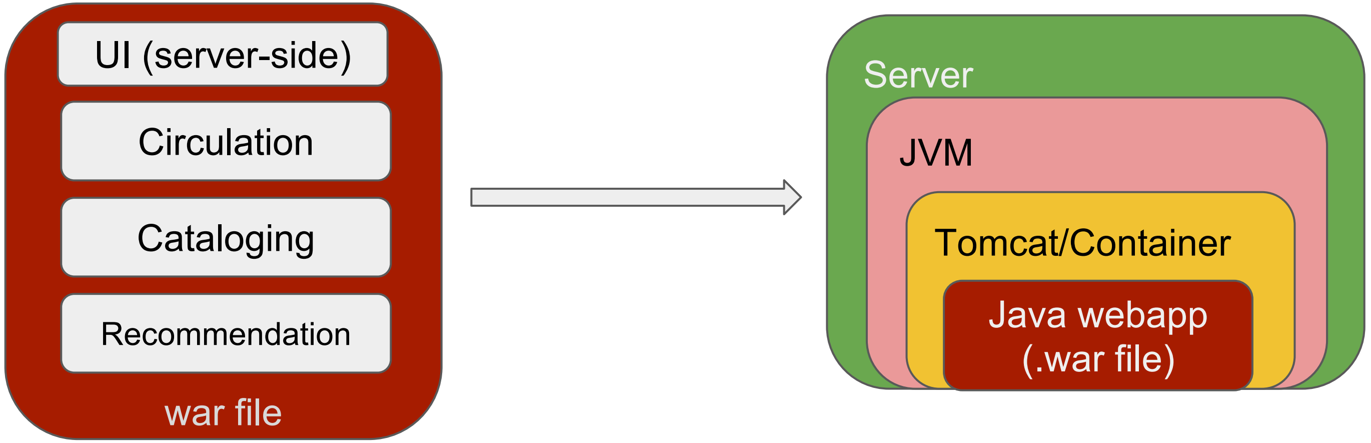
Microservices

- For web scale only?
- Non technical advantages
- Allows for increased pace of innovation

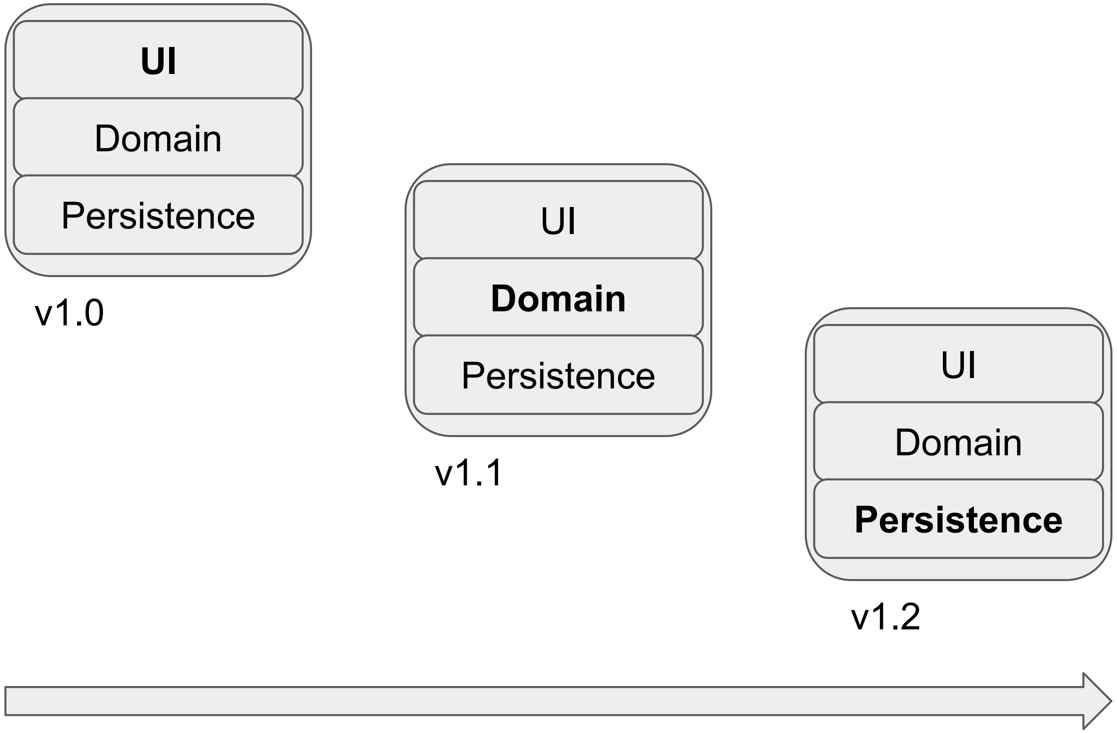


Traditional Web Applications

- Sometimes referred to as “monoliths”
- Can follow best practices in regards to architecture
- Applications packaged and deployed as single artifact or directory hierarchy

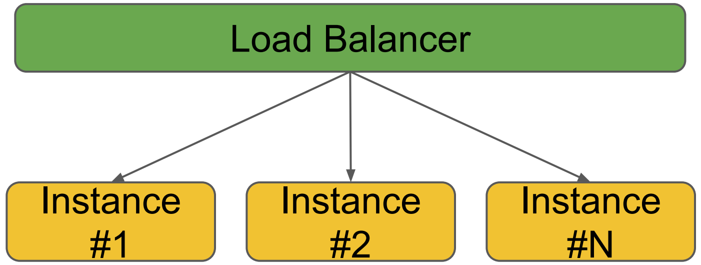


Releasing/versioning monolithic applications



Scaling Traditional Web Applications

- Horizontal Scaling

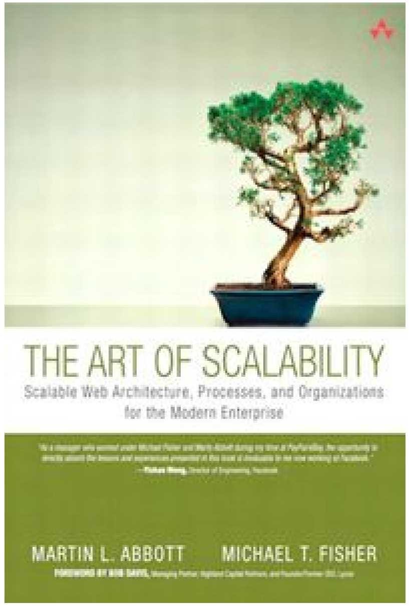


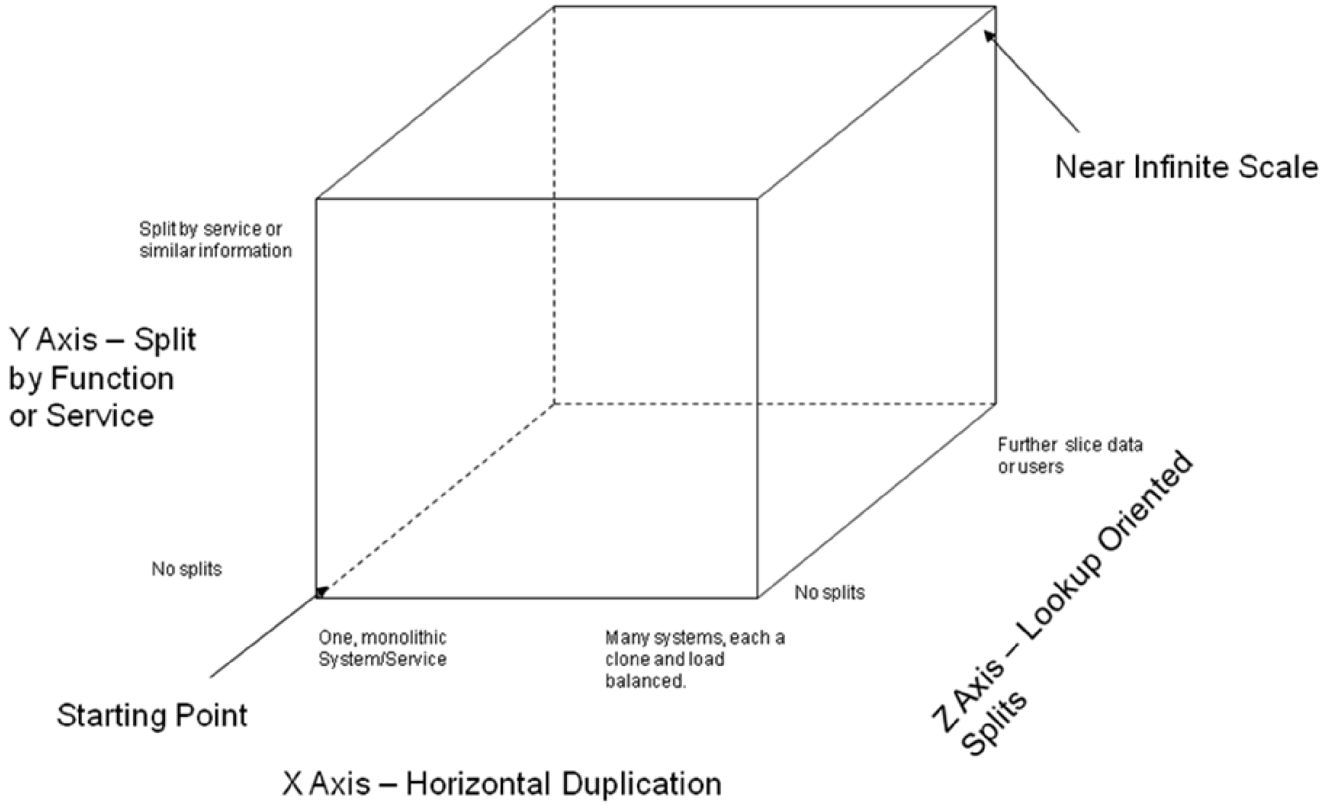
- Vertical Scaling
- Sharding

“Limitations” of monoliths

- Difficult to maintain and release over time
- Dependency management issues
- “Wholesale” scaling of apps.
- Upfront technology stack decision (lock-in).

Scaling by Functional Decomposition


THE ART OF SCALABILITY
Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise
MARTIN L. ABBOTT
MICHAEL T. FISHER

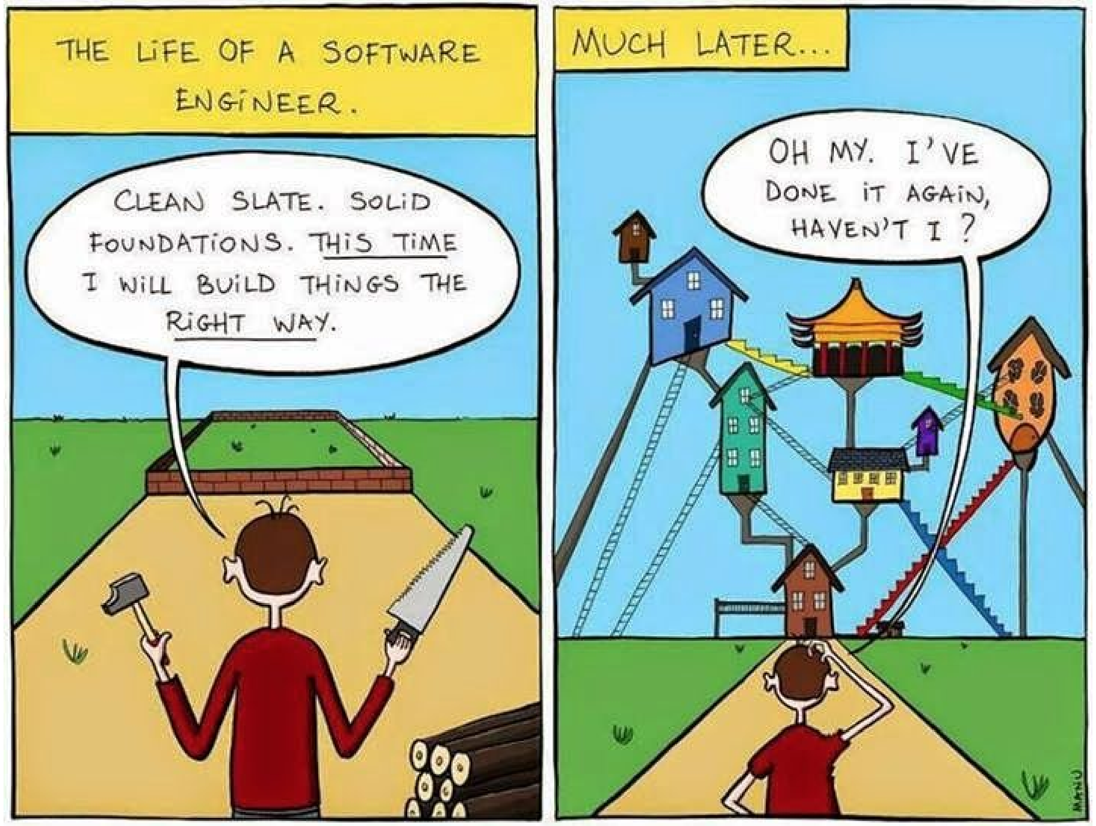

Near Infinite Scale
Split by service or similar information
Y Axis – Split by Function or Service
No splits
Starting Point
One, monolithic System/Service
Many systems, each a clone and load balanced.
No splits
Further slice data or users
Z Axis – Lookup Oriented Splits
X Axis – Horizontal Duplication

http://akfpartners.com/techblog/2008/05/08/splitting-applications-or-services-for-scale/

Microservices doesn’t solve everything

- Microservices are not a silver bullet
- “Microservices - not a free lunch”
- Distributed big balls of mud
  - “If you can't build a monolith, what makes you think microservices are the answer?”
- “Micro” is probably not the right term anyways

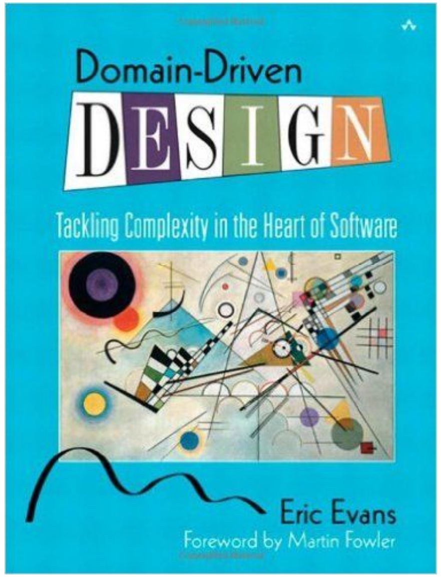
http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html
http://www.codingthearchitecture.com/2014/07/06/distributed_big_balls_of_mud.html


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?
MANU

https://plus.google.com/+ProgrammingCom/posts/LECNUgi4t6R?pid=5851335745881982002&oid=105792929719283190873

Road to microservices

- Domain Driven Design
- Bounded Context

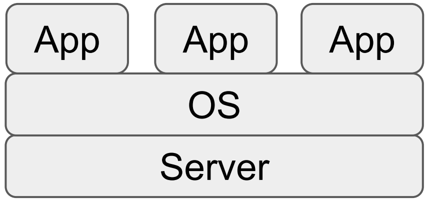

Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Road to microservices

- Virtualization
- Devops Practice
- IaaS
- PaaS
- Containerization

Road to microservices - Pre VM

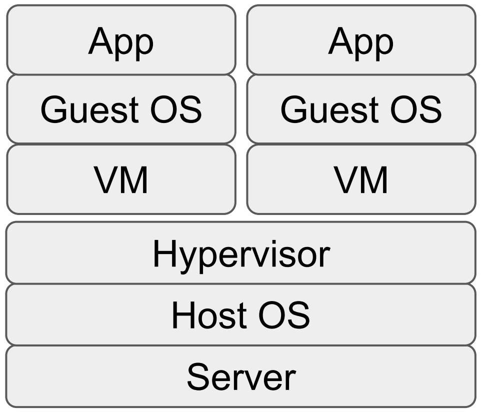
- Long lead time to stand up new server
- Special snowflake servers
  - ie. diglibdev1, diglibqa1, diglibprod1, diglibprod2, diglibprod3
- Inconsistent environments
- Package artifact and throw over wall
- Pack everything onto single machine
- But it works on my laptop!



http://blog.codinghorror.com/the-works-on-my-machine-certification-program/

Road to microservices - Virtualization

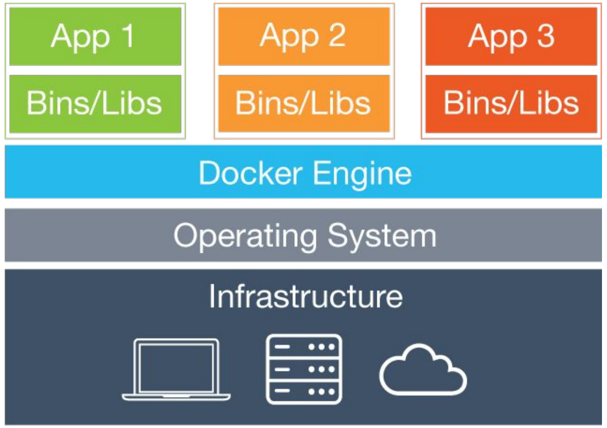
- Virtualization
- Multiple VMs per machine
- VMs still require resources
- IaaS (AWS)
- Cloud based VM’s
- Rapid elasticity

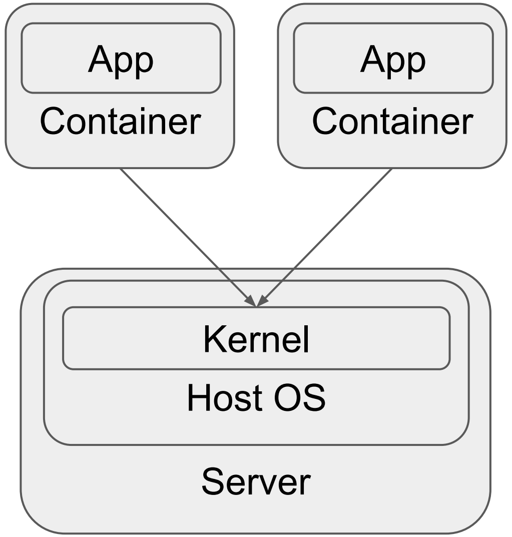


Cloud VM
(EC2)

Road to microservices - Containers

- More lightweight than VMs
- No guest OS
- Docker
- Docker Hub



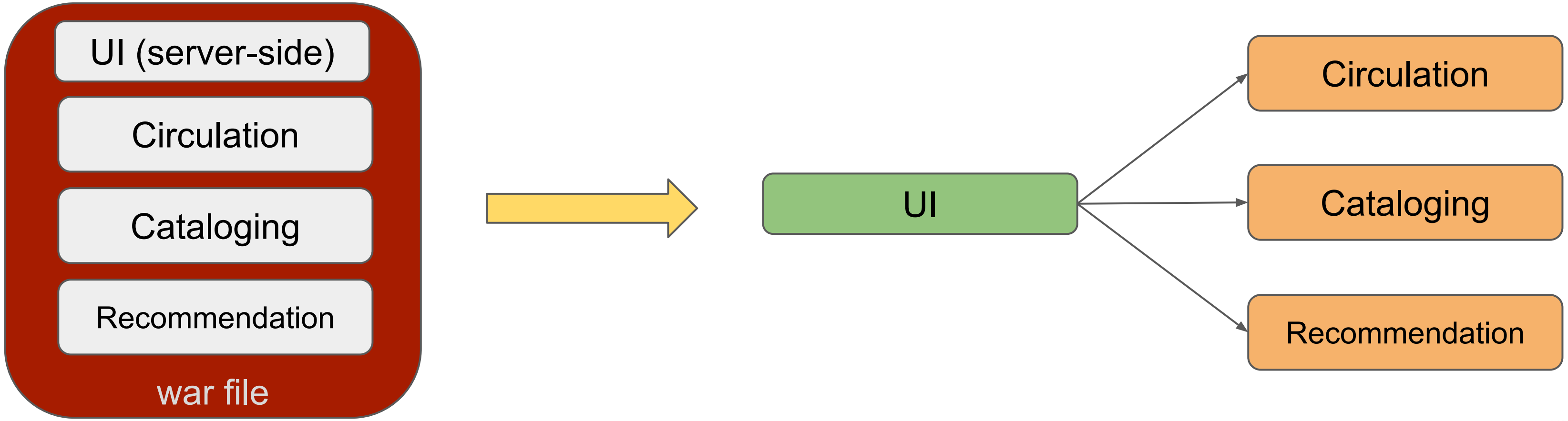


Road to microservice - Devops

- Devops is about organization culture
- Devops != {‘Chef’ || ‘Puppet’ || ‘Ansible’}
- Automation
- Getting things done
- Turn around time is minutes/days not weeks/months
- PaaS

Enter Microservices

- “An application architectural style for creating a set of loosely coupled services each with a bounded context”



Advantages of microservices

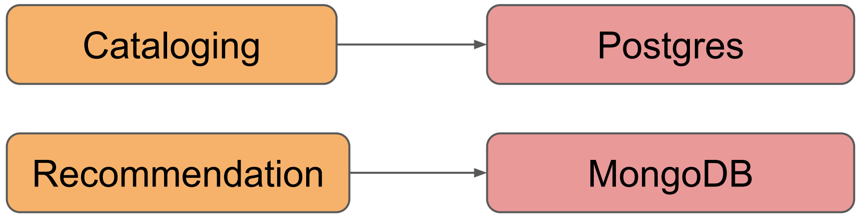
- Allows organizations to evaluate and experiment with new technologies
- Fine grain scaling resulting in more efficient use of resources
- Faster/less risky incremental releases
- Break down complex application and problem spaces

Amazon/Bezos API mandate

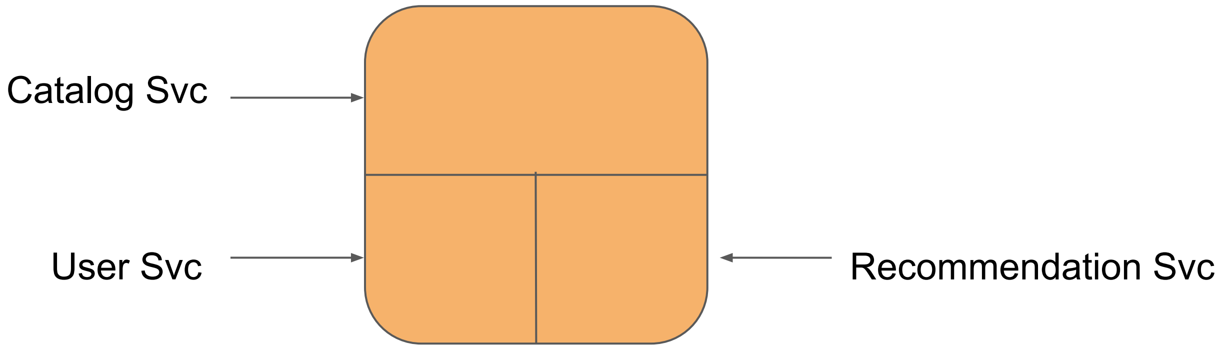
- All teams will henceforth expose their data and functionality through service interfaces.
- Teams must communicate with each other through these interfaces.
- There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
- It doesn't matter what technology they use.
- All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

Properties of Microservices

- Bounded context
- Encapsulate implementation details
- One datastore per microservice/no integration database across services
- Loosely coupled
- Clear external interfaces

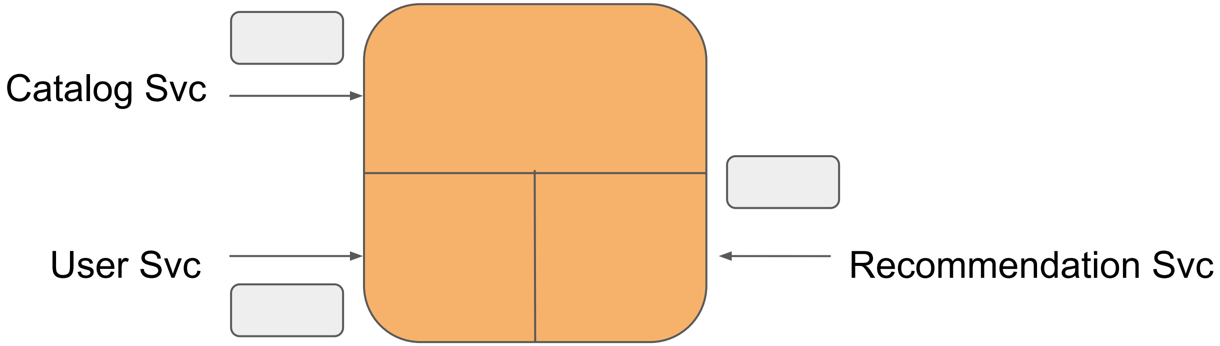


UI Strategies



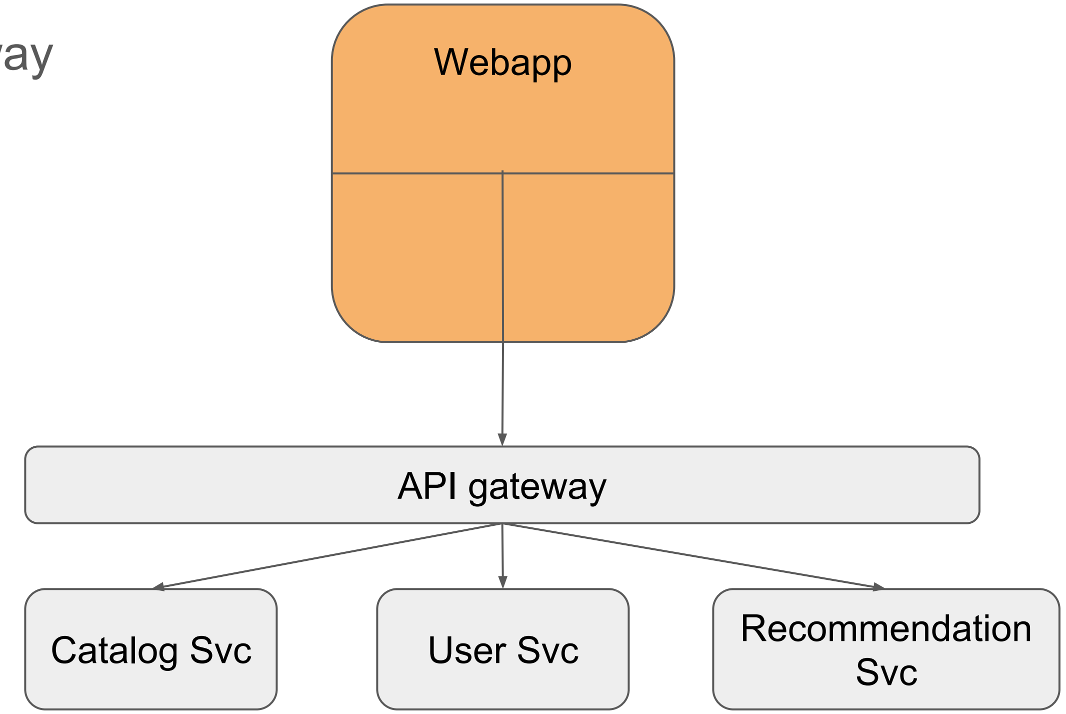
UI Strategies

- UI Fragment Composition



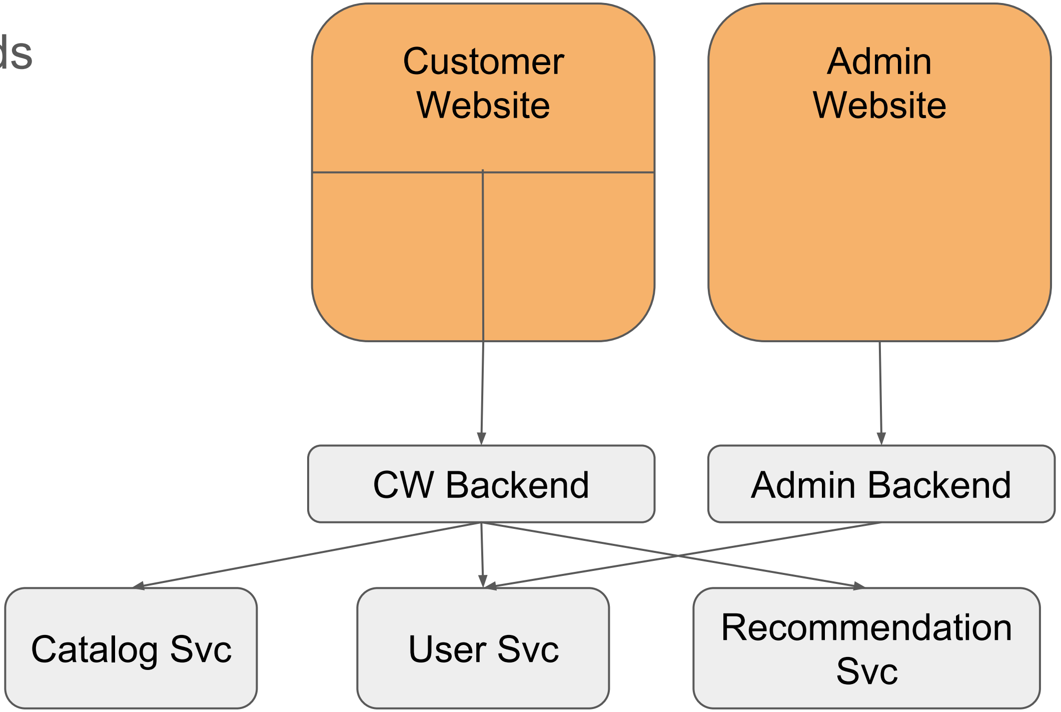
UI Strategies

- API gateway



UI Strategies

- Backends for Frontends



Complexities

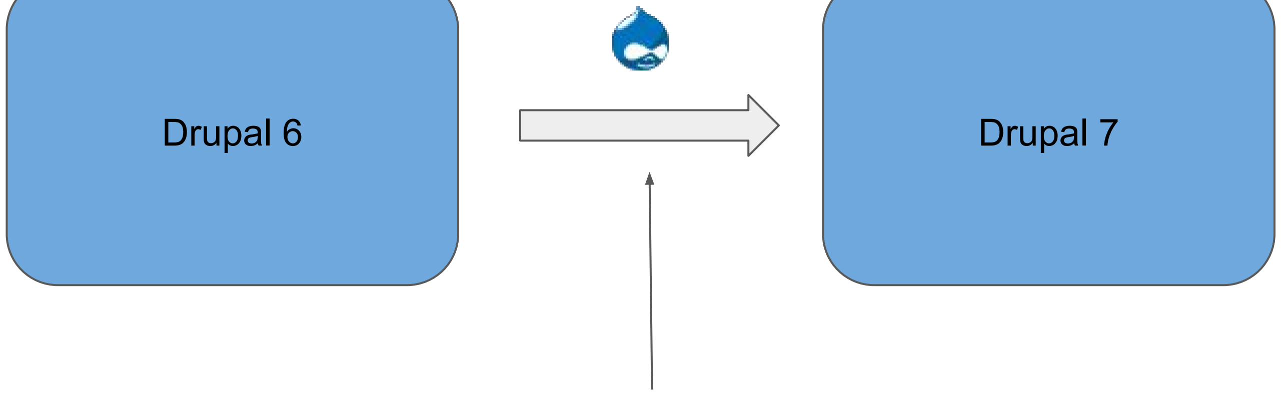
- Distributed Systems
- CAP Theorem
  - Eventually Consistent Systems

But what about SOA?

- Best of SOA Principles
- SOA 2.0
- Less emphasis on middleware (ESB, etc)
- Smart endpoints, dump pipes
- Beware of microservice-enabled middleware, “micro-service product”

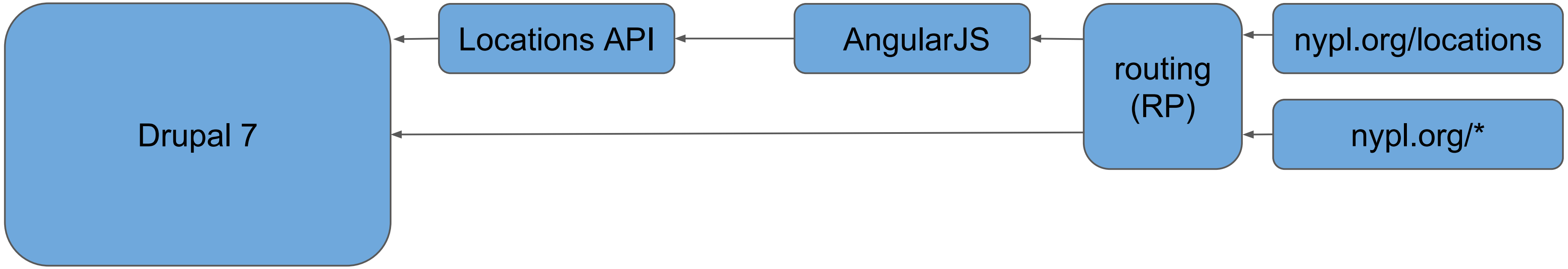
Refactoring/Integrating with existing apps

- Strategies for dealing with legacy/3rd party applications
- Case study



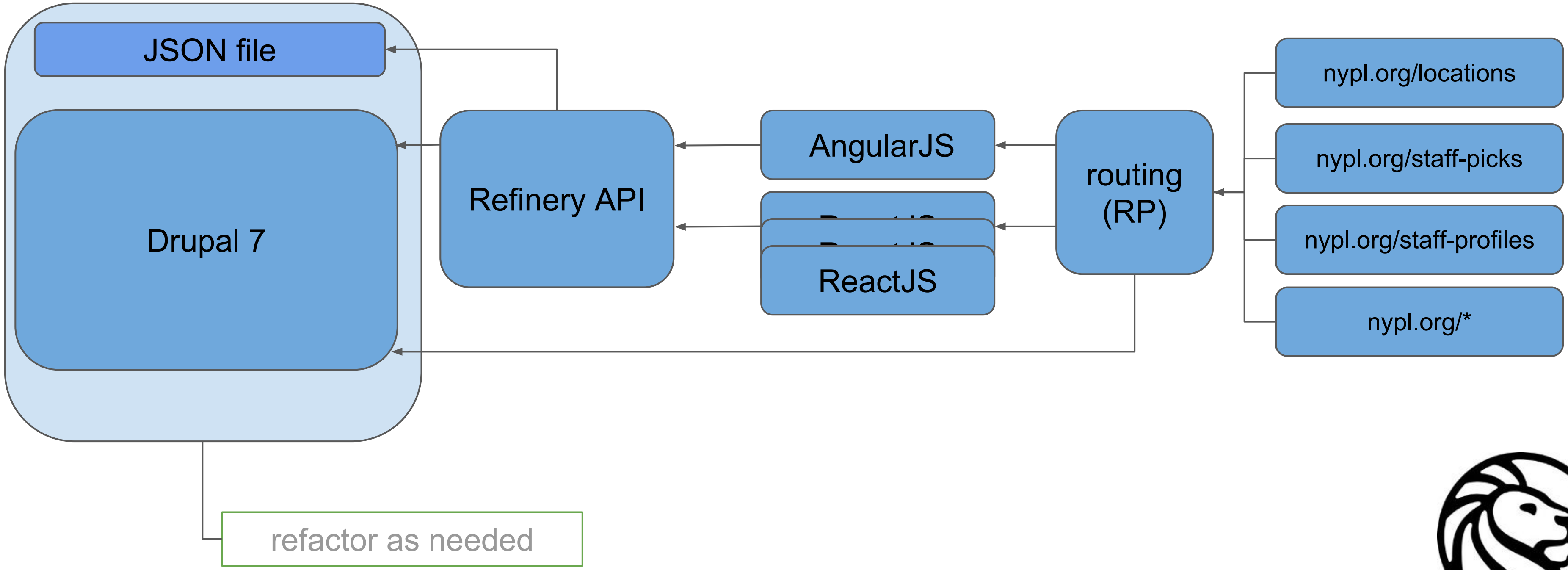
Refactoring/Integrating with existing apps

- Migrated D6--->D7 instance
- “Locations” experiment



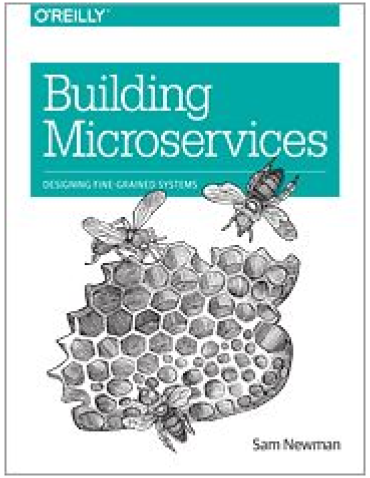
Refactoring/Integrating with existing apps

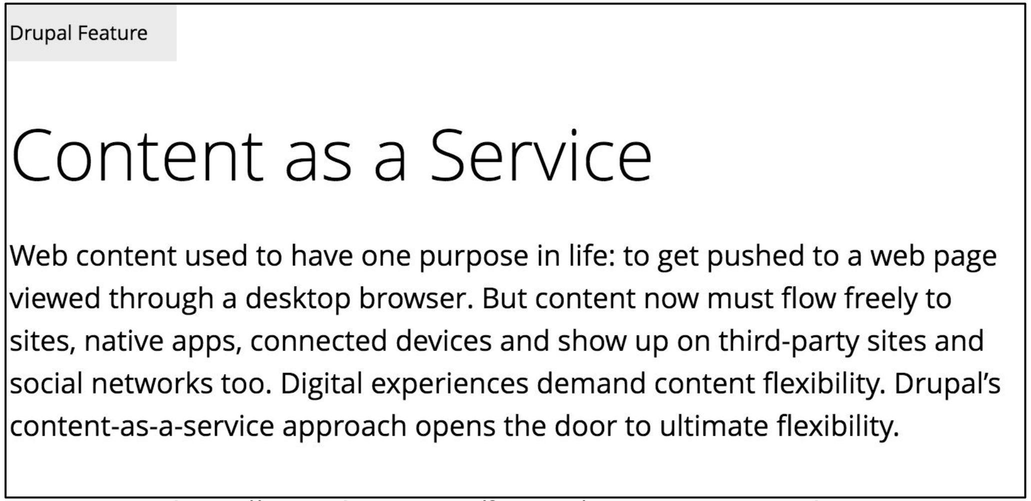
- Iterate and expand on architecture



Best practices/Emerging Trends

- Providing CMS content as service is appearing to be a best practice
- Building Microservice book describes “CMS as a Service”
- Drupal 8 describes “Content as a Service”



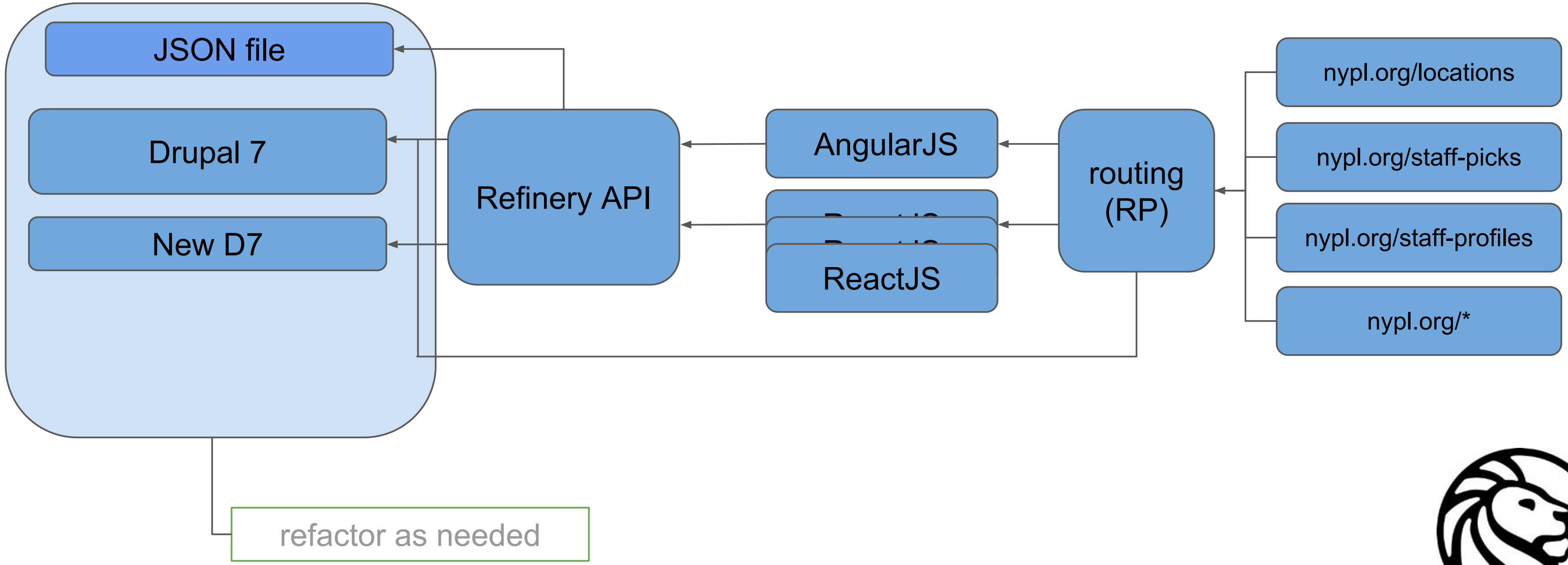

Drupal Feature

Content as a Service

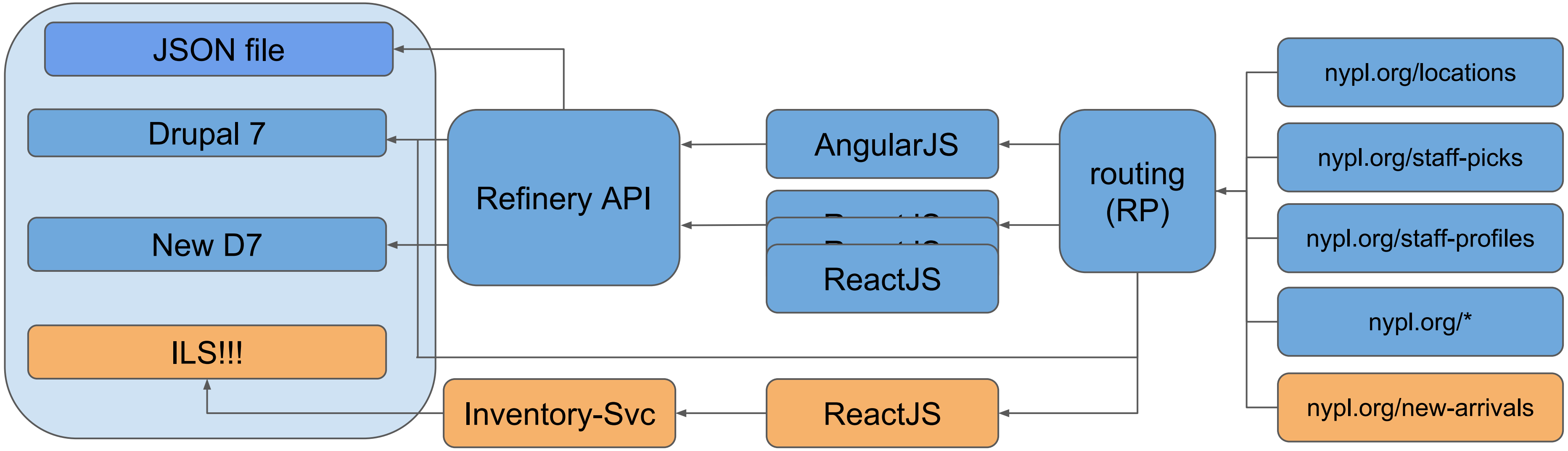
Web content used to have one purpose in life: to get pushed to a web page viewed through a desktop browser. But content now must flow freely to sites, native apps, connected devices and show up on third-party sites and social networks too. Digital experiences demand content flexibility. Drupal’s content-as-a-service approach opens the door to ultimate flexibility.

https://www.drupal.com/feature/content-as-a-service

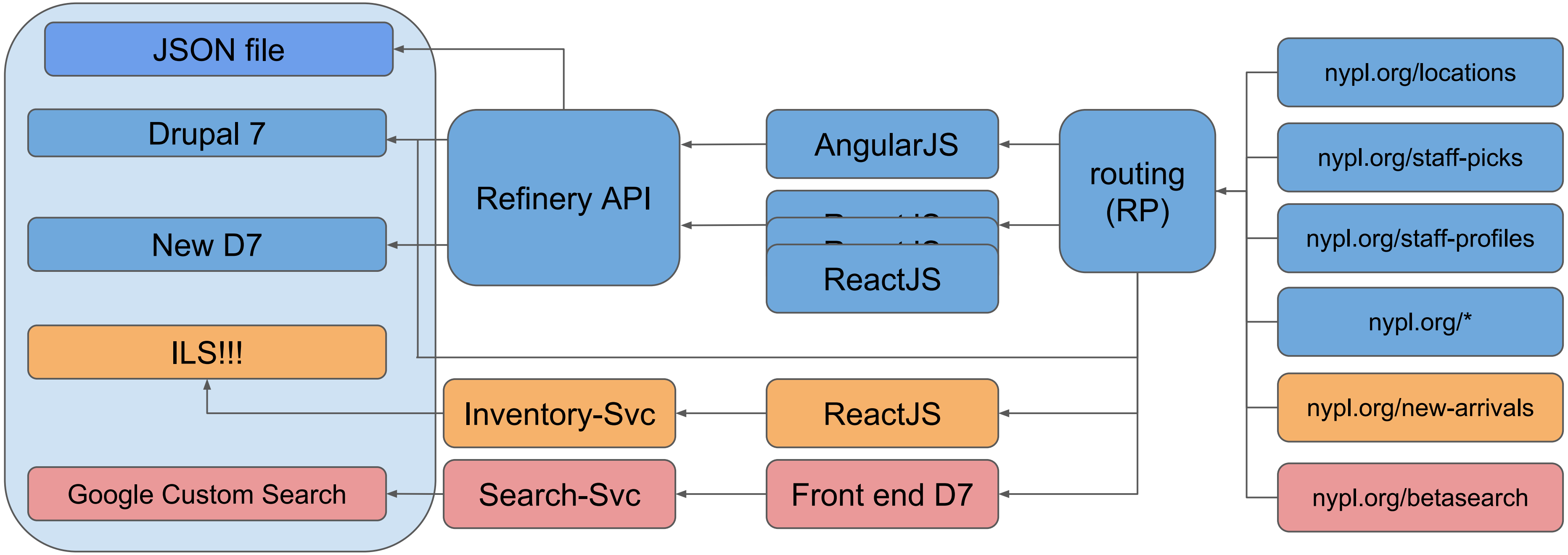
Refactoring/Integrating with existing apps



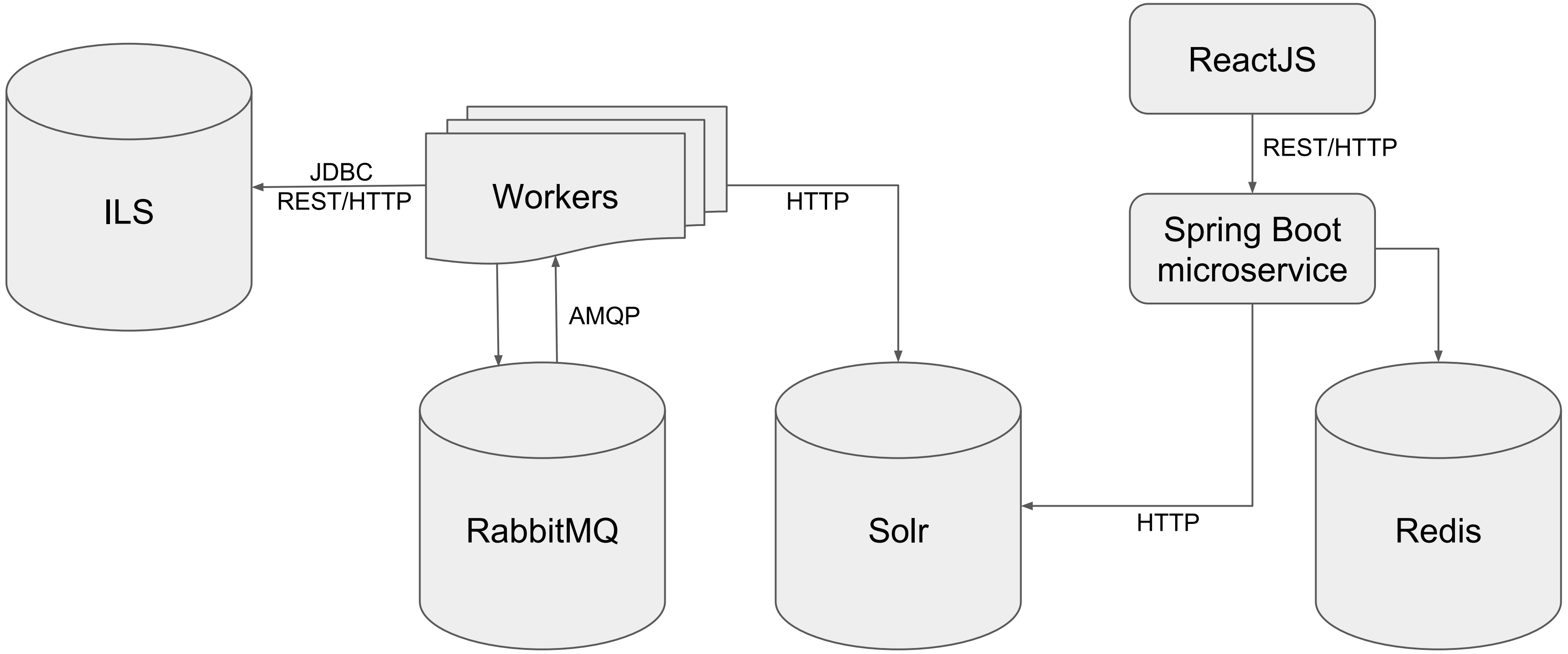
Refactoring/Integrating with existing apps



Refactoring/Integrating with existing apps



Inventory Service - closer look

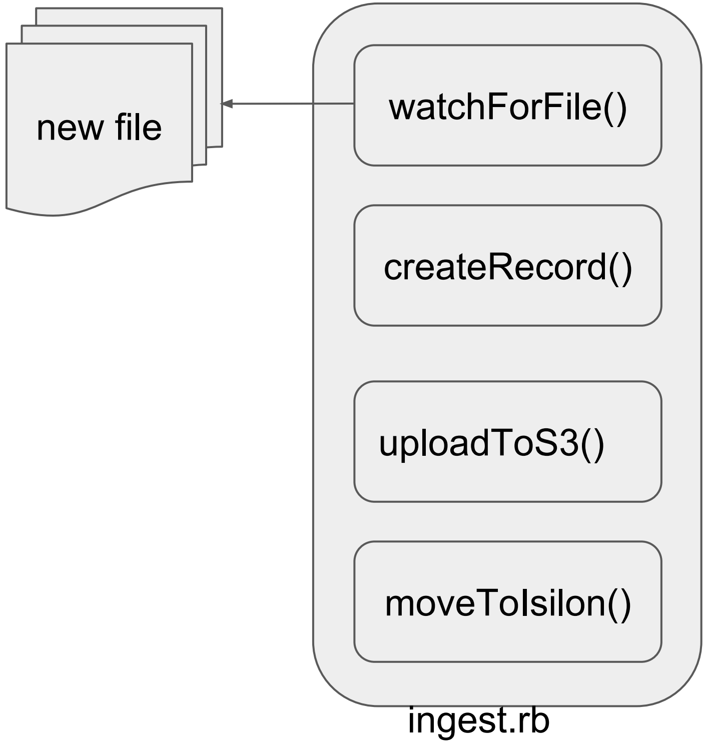


Microservices for complex workflow

- Orchestration
- Choreography
- NYPL A/V ingest use case
  - Have an idea of end state
  - current state and workflows unclear
  - Spreadsheets
  - “Parking lot” storage (backlog > 600+TB)
  - **Bag created as result of current process**
- Initial known steps
  - Watch for file/bag
  - Create entry in metadata system
  - Transcode the video
  - Place preservation master in Isilon storage

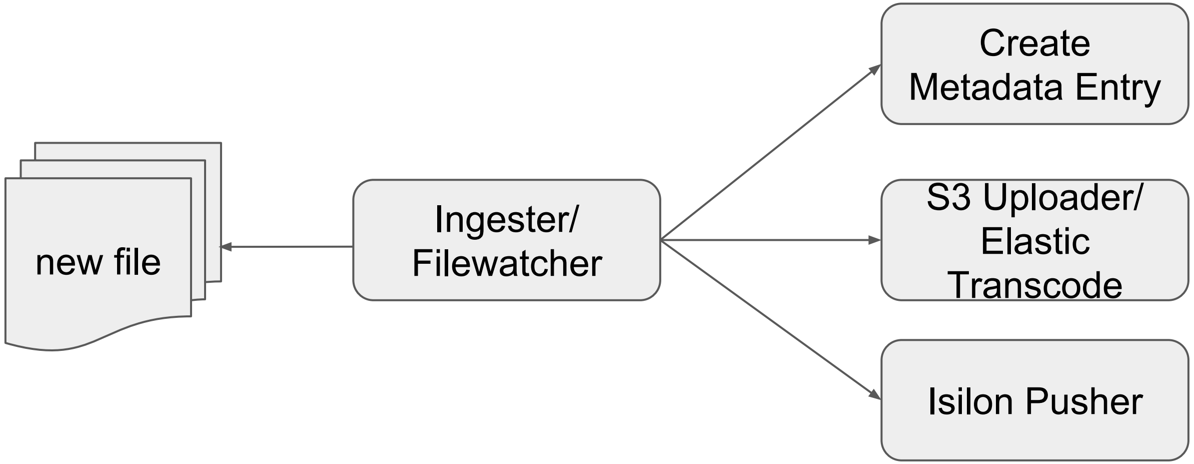
Audio/Video Ingest

- Not service based
- Ingest Script



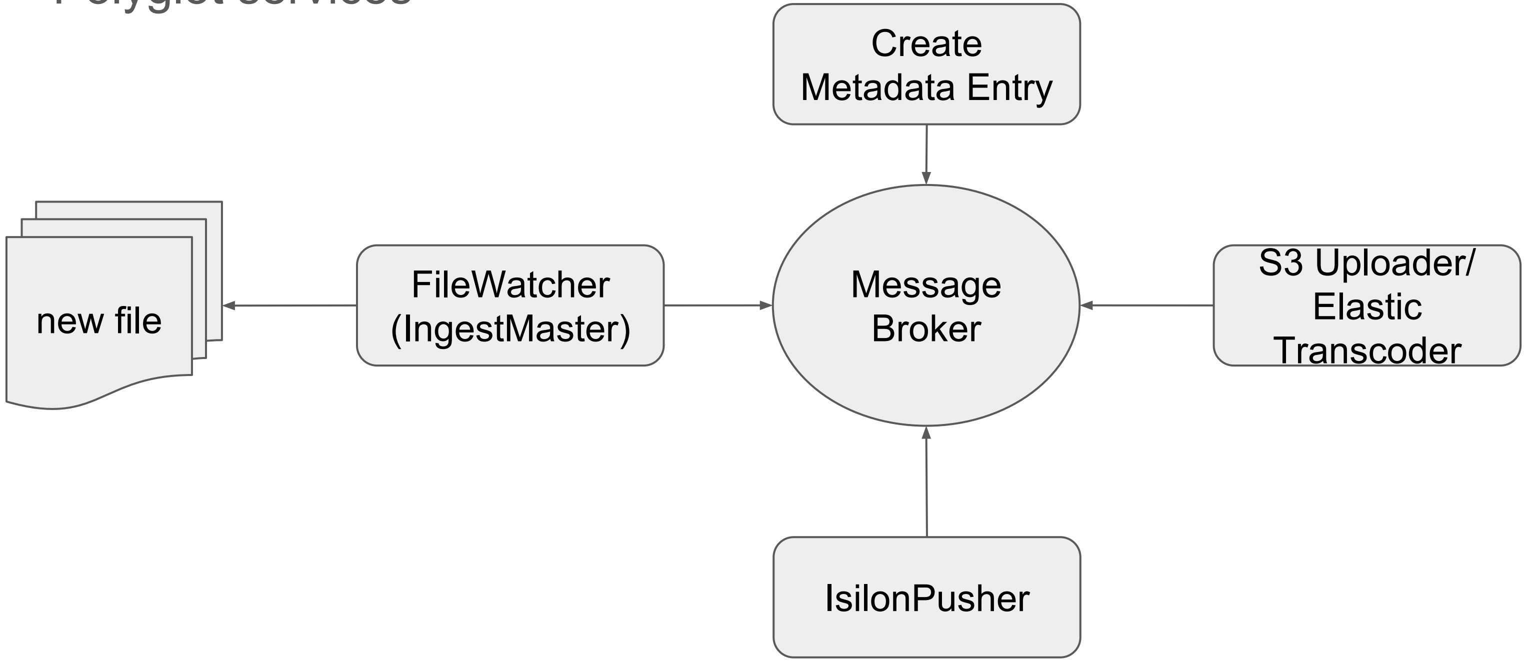
Audio/Video Ingest

- Make it service based
- Orchestrated Approach Example



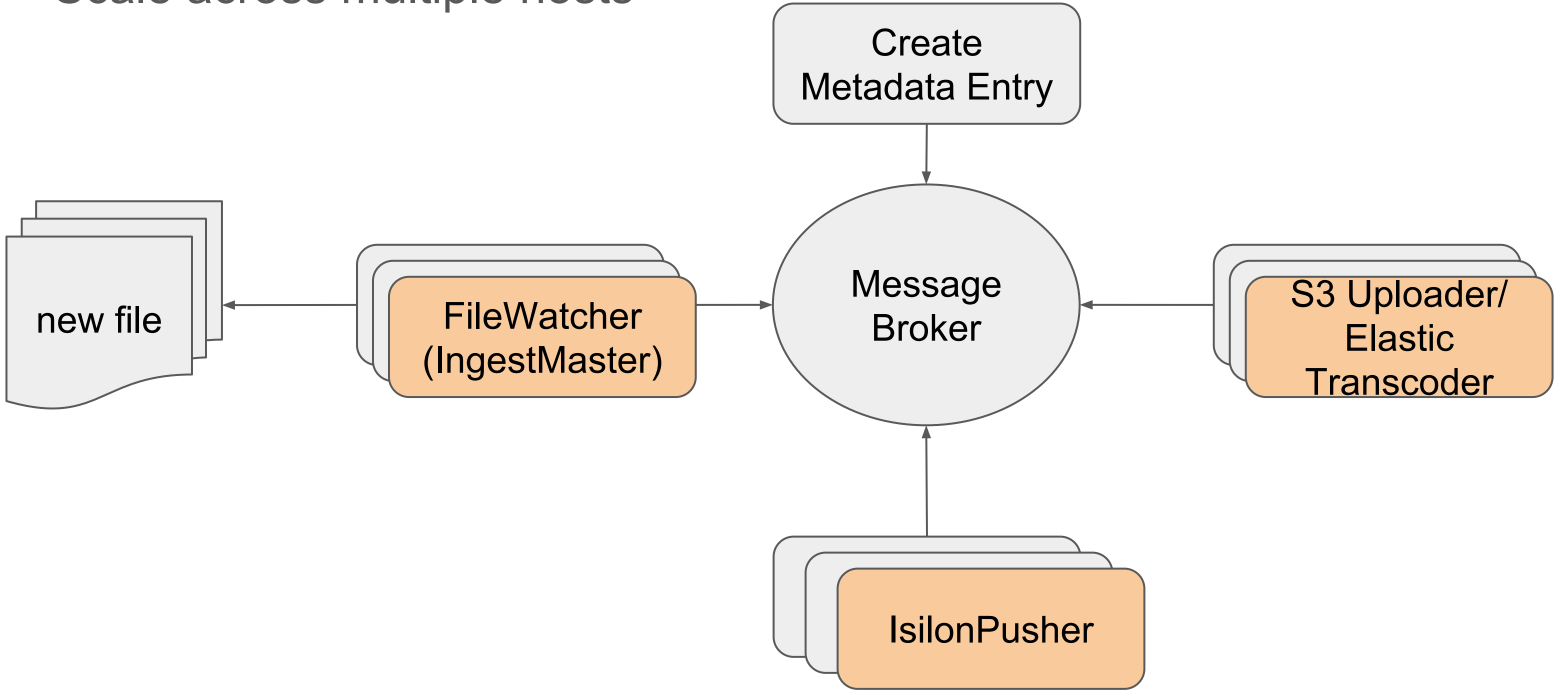
Audio/Video Ingest

- Choreographed Approach Example
- Polyglot services



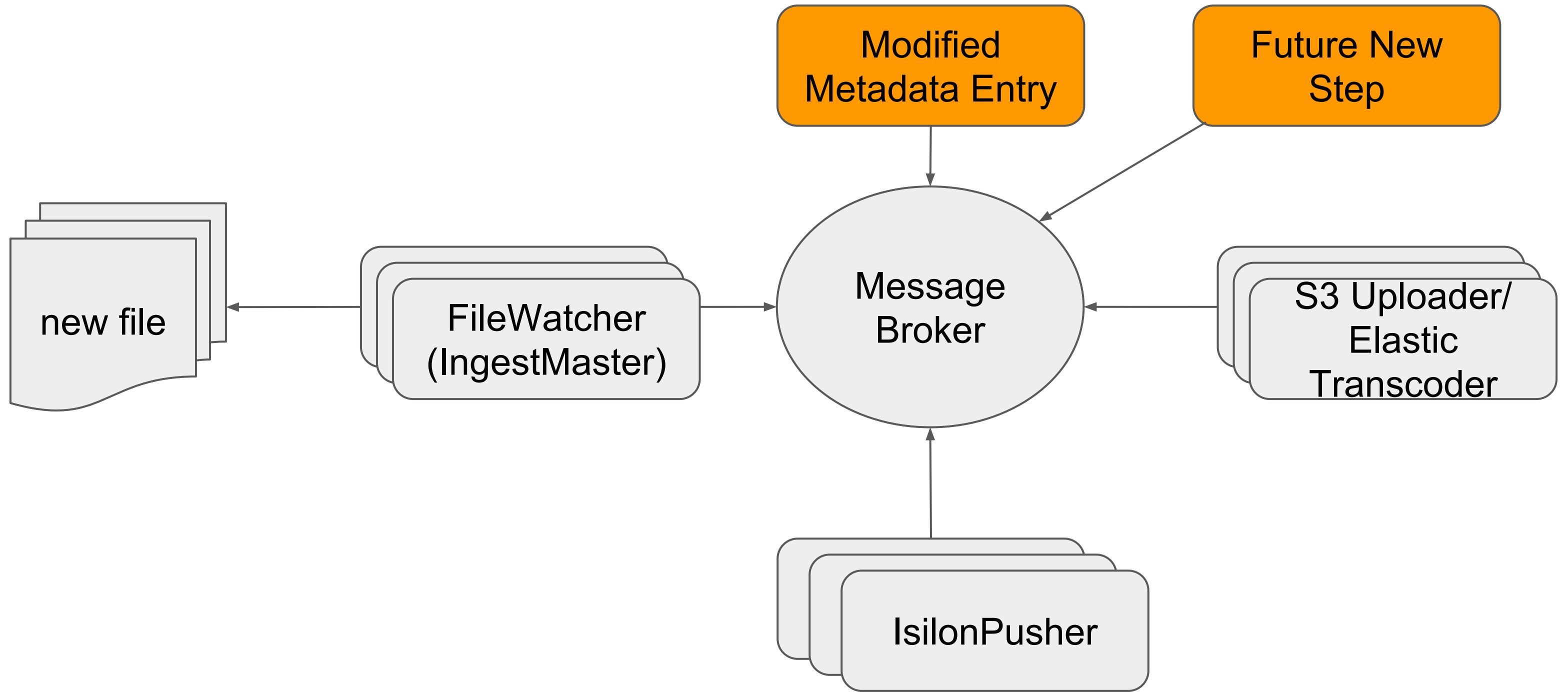
Audio/Video Ingest

- Scale out services as needed
- Scale across multiple hosts



Audio/Video Ingest

- Modify and extend

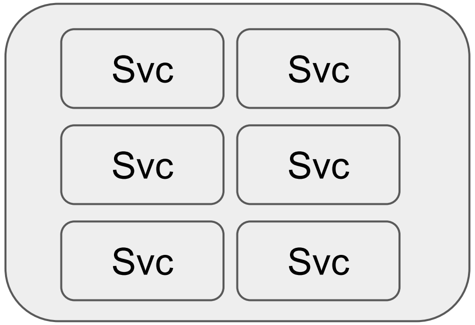


A/V ingest workflow

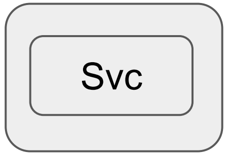
- Tracking status
- Use Correlation ids (uuid-based)
- View status in log aggregation tool

Deployment Strategies

- Multiple services per host



- Definition of host
- Service per host



- Docker/Kubernetes

Microservice Requirements

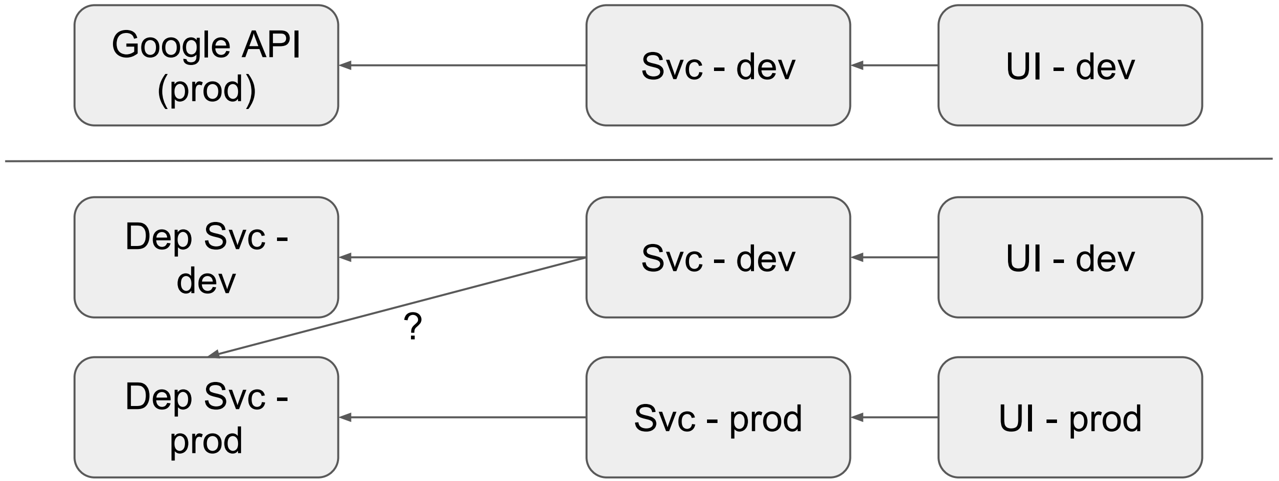
- Automation
  - Provisioning
  - Automated testing
  - CI/CD
  - Rollback
- Monitoring
  - APM
  - Log aggregation
  - Checkpoints (queue-in, queue-out)
  - Correlation IDs
- Service Discovery
- Design for Failure
  - Any node can go down
  - Network failure
  - Degrade functionality without taking down system

Security

- OAuth2.0
- Parties
  - Resource Owner
  - Resource Server
  - Client
  - Authorization Server (can be same as resource server sometimes)
- Authorization Grant Types
  - Authorization Code
  - Client Credentials
  - Implicit Resource Owner
  - Password Credentials
- Open ID Connect
  - Open ID providers
  - Relying Party
- JSON Web Tokens (JWT)

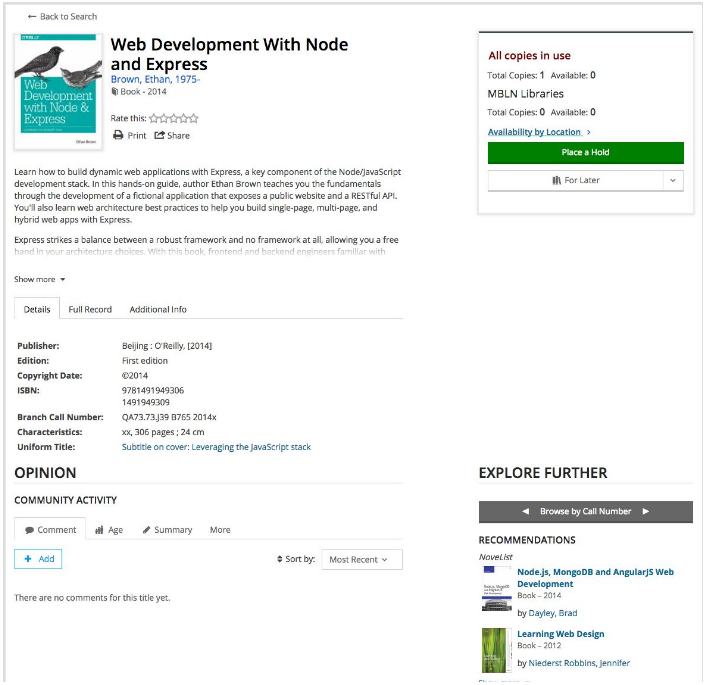
Testing

- Integration Environments
- End to End tests?



Discussion/Exercise

Decompose the Biblicommons items page into services
https://bpl.bibliocommons.com/item/show/4652502075_web_development_with_node_and_express


← Back to Search

Web Development With Node and Express
Brown, Ethan, 1975-
Book - 2014

Rate this: ☆☆☆☆☆
Print Share

Learn how to build dynamic web applications with Express, a key component of the Node/JavaScript development stack. In this hands-on guide, author Ethan Brown teaches you the fundamentals through the development of a fictional application that exposes a public website and a RESTful API. You'll also learn web architecture best practices to help you build single-page, multi-page, and hybrid web apps with Express.

Express strikes a balance between a robust framework and no framework at all, allowing you a free hand in your architecture choices. With this book, frontend and backend engineers familiar with

Show more ▾

Details | Full Record | Additional Info

Publisher:	Beijing : O'Reilly, [2014]
Edition:	First edition
Copyright Date:	©2014
ISBN:	9781491949306 1491949309
Branch Call Number:	QA73.73.J39 B765 2014x
Characteristics:	xx, 306 pages ; 24 cm
Uniform Title:	Subtitle on cover: Leveraging the JavaScript stack

OPINION

COMMUNITY ACTIVITY

Comment | Age | Summary | More

+ Add   Sort by: Most Recent

There are no comments for this title yet.

All copies in use
Total Copies: 1 Available: 0
MBLN Libraries
Total Copies: 0 Available: 0
Availability by Location ›
Place a Hold
For Later

EXPLORE FURTHER

◀ Browse by Call Number ▶

RECOMMENDATIONS
NoveList

Node.js, MongoDB and AngularJS Web Development
Book – 2014
by Dayley, Brad

Learning Web Design
Book – 2012
by Niederst Robbins, Jennifer

Thank You