Real-World Microservice Development with IBM WebSphere Liberty:

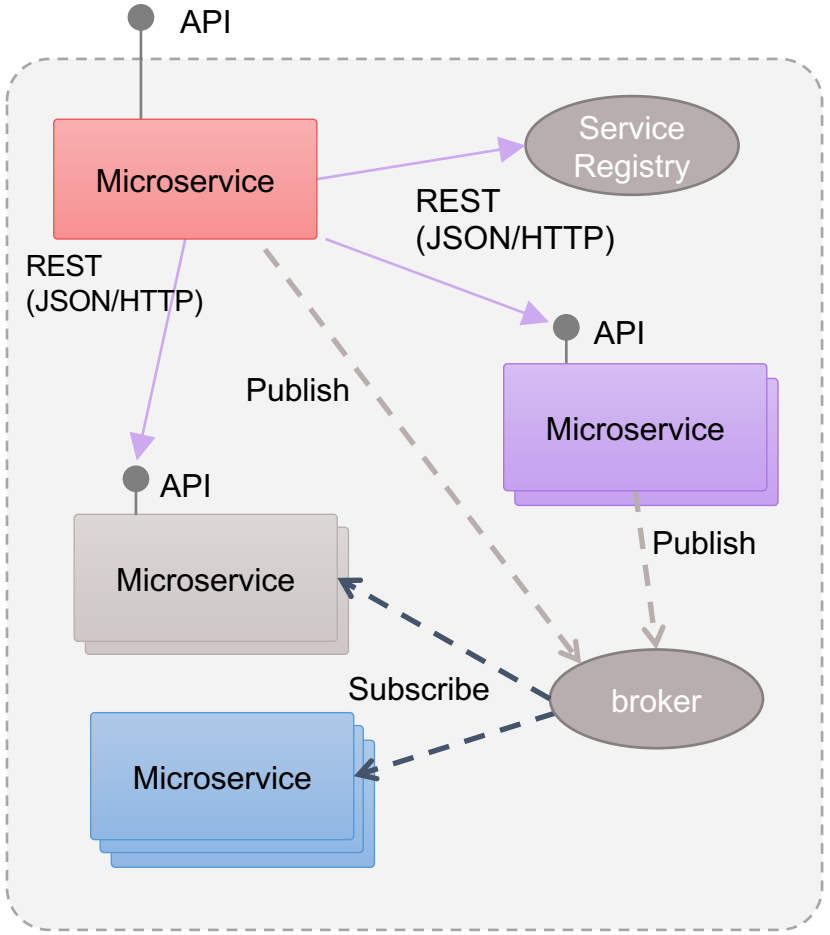
Game On!

Erin Schnabel
@ebullientworks

InterConnect
2017

Microservices are used to…

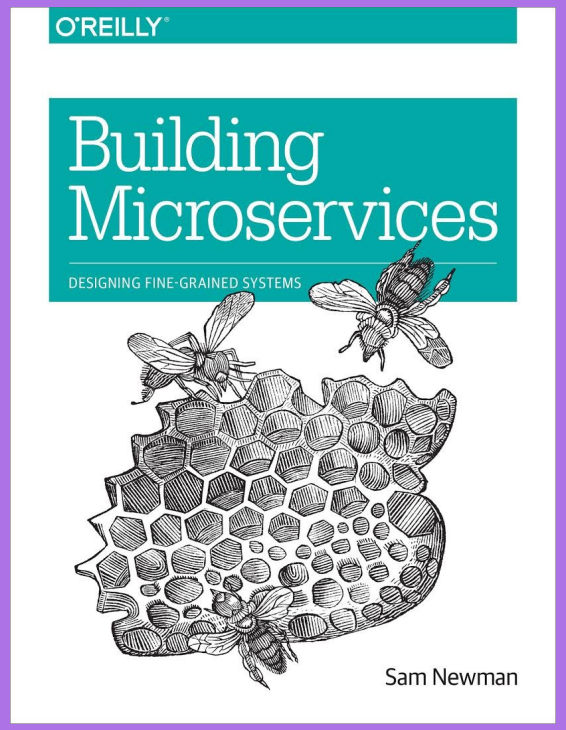
- compose a complex application using
  - “small”
  - independent (autonomous)
  - replaceable
  - processes
- that communicate via
  - language-agnostic APIs

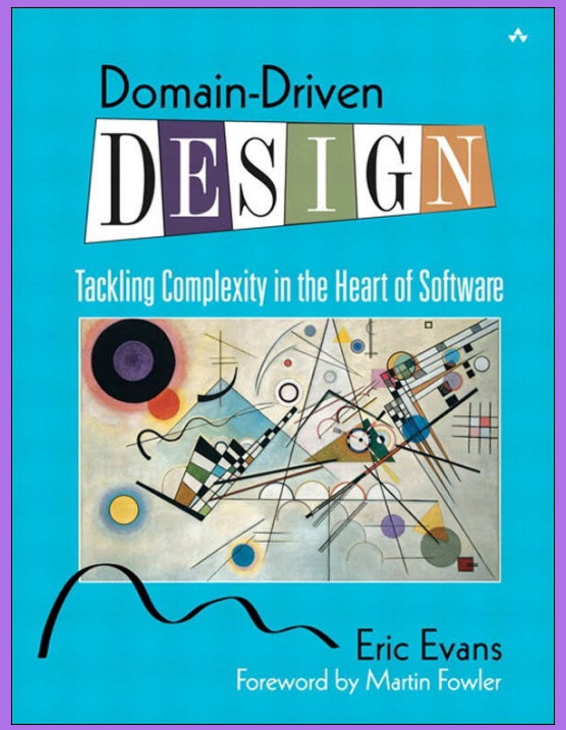


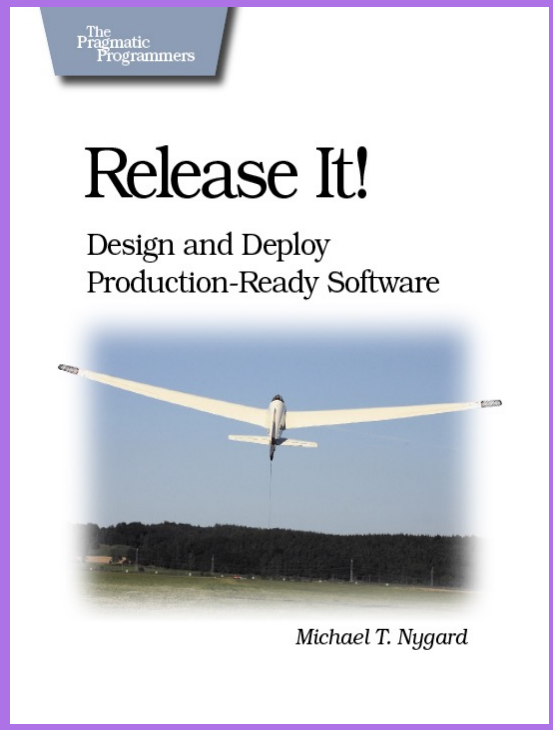
Microservices application

Conway’s law

Bounded Contexts







Eventual consistency

DevOps

Automation

Testing?

Microservices Sample Apps…

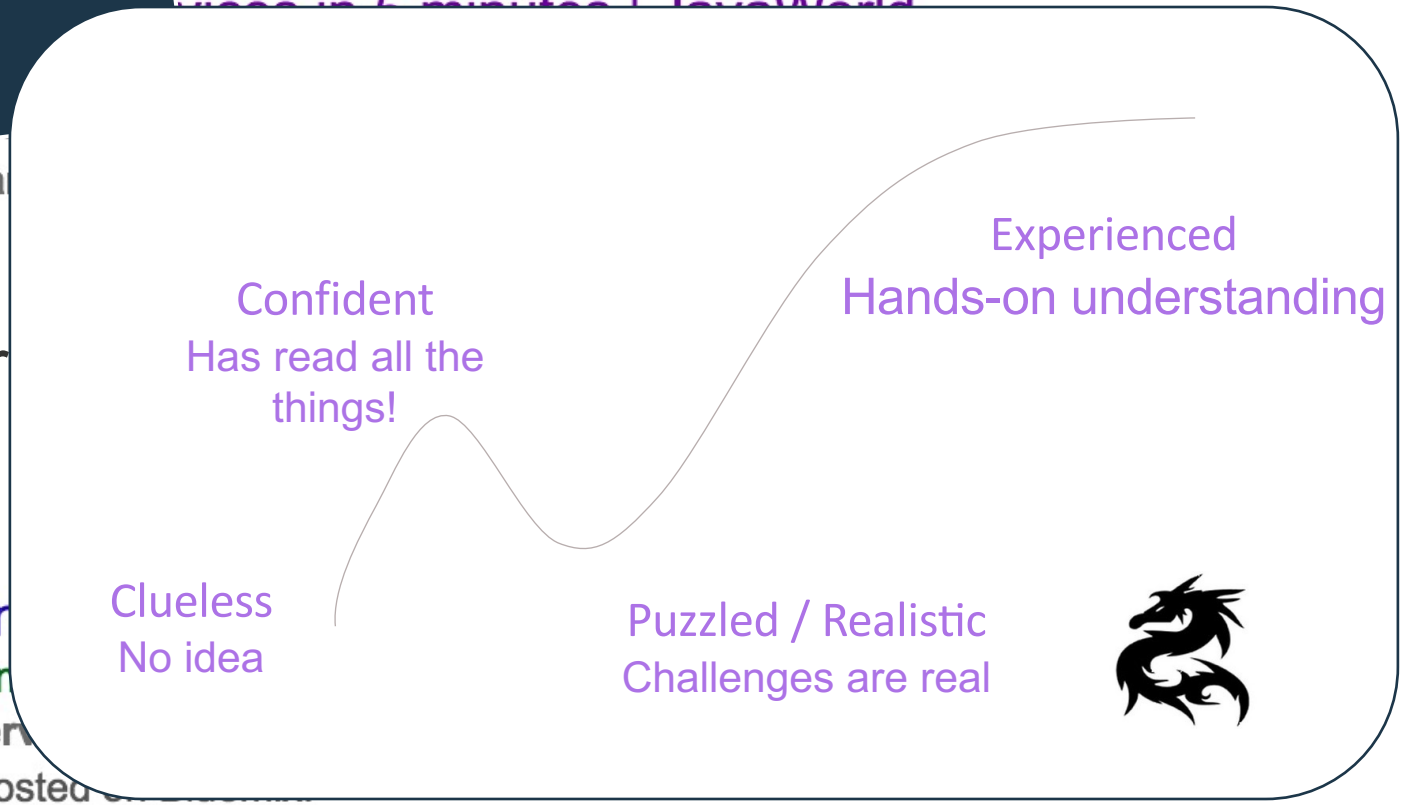
- Create a single servic

Microservices are so easy!

ve not imported the sa

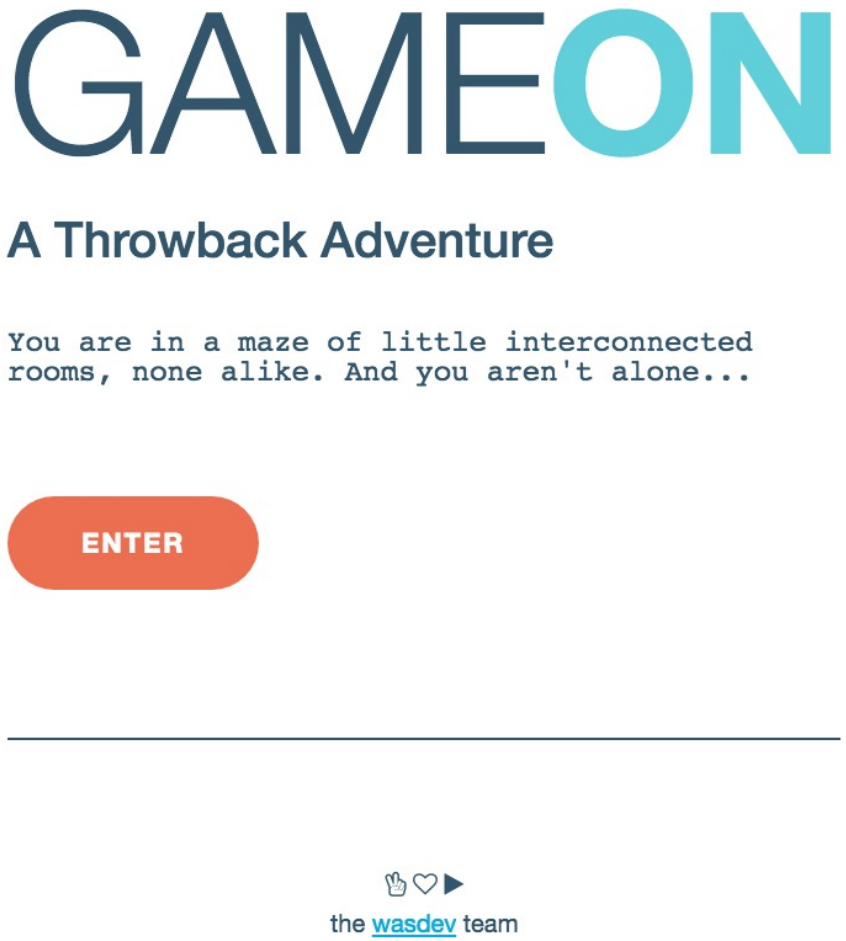
- Rebuild a pre-baked micr

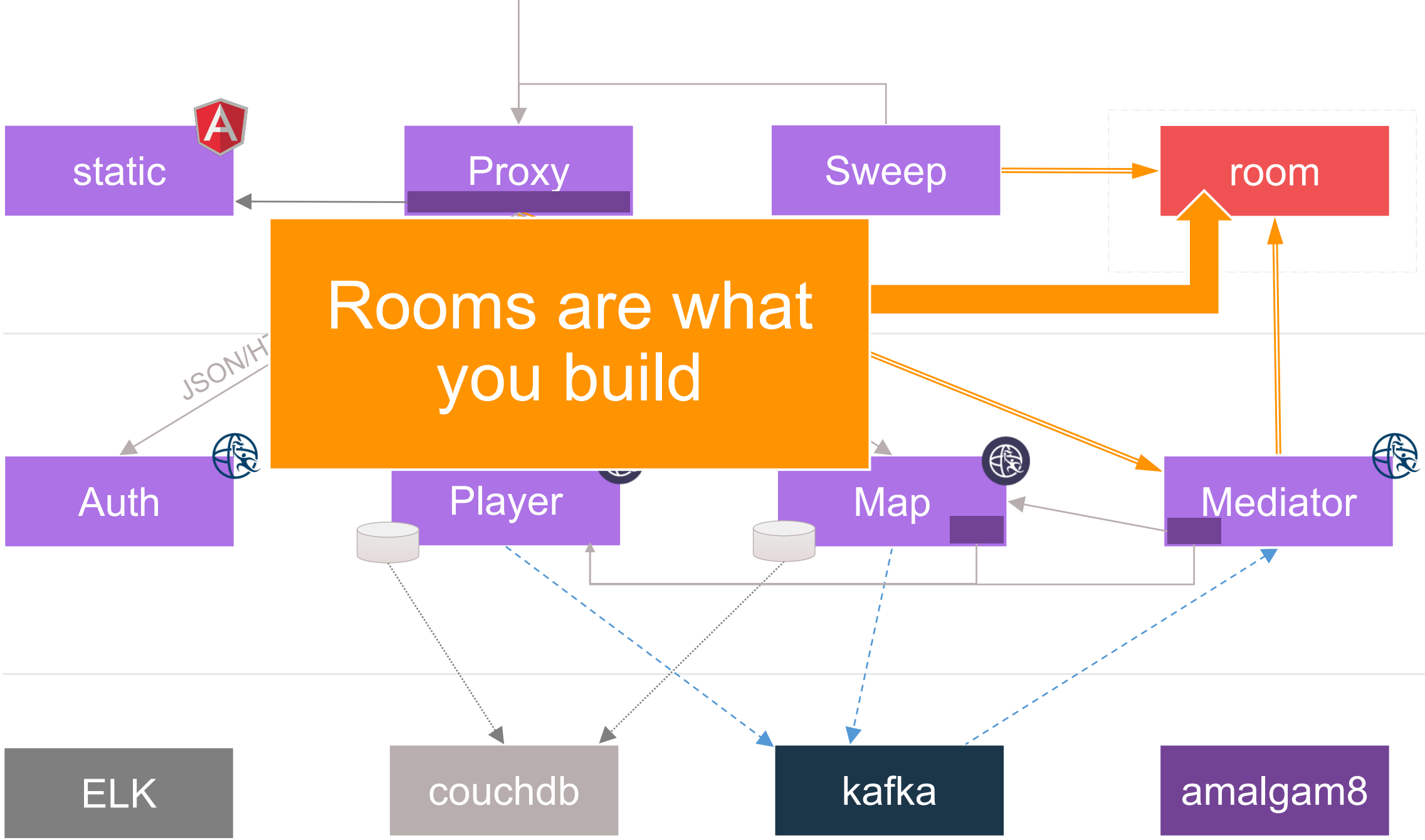
Microservices Onlin
https://developer.ibm.cor
Mar 16, 2015 - A microser
Java JAX-RS, PHP and hosted

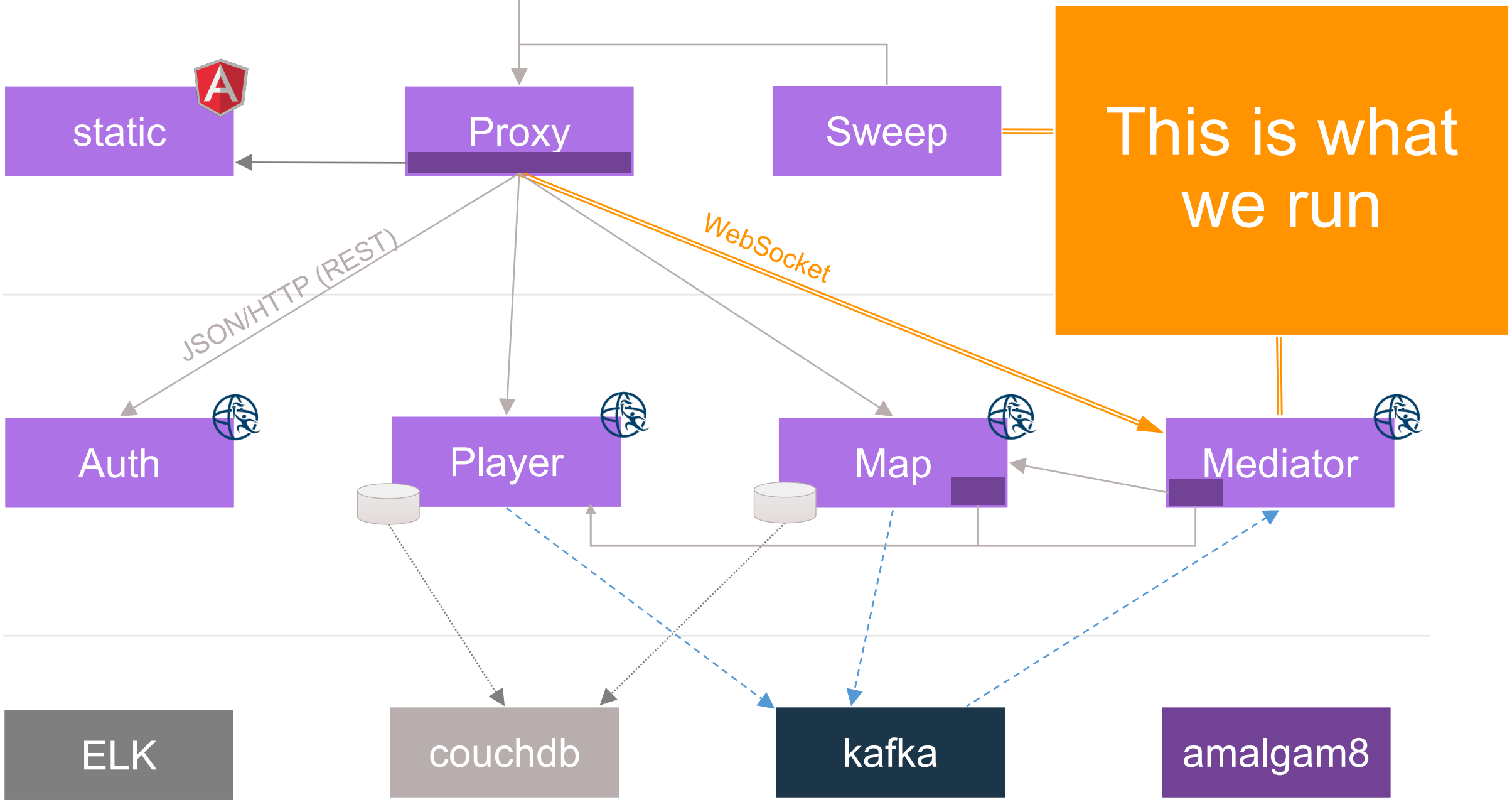


The premise …

- Hands on with microservices
- Stick with ’Hello World’ simplicity
- Choose your own adventure
- Fast path to the hard stuff
- Build something cool (to you!)
- Learn as you go




static
Proxy
Sweep
room
Rooms are what you build
JSON/H
Auth
Player
Map
Mediator
ELK
couchdb
kafka
amalgam8


static
Proxy
Sweep
This is what we run
JSON/HTTP (REST)
WebSocket
Auth
Player
Map
Mediator
ELK
couchdb
kafka
amalgam8

Game On! in action

connected: validating JWT
enter The First Room

Welcome to The First Room

The First Room

You've entered a vaguely squarish room, with walls of an indeterminate color.

TL;DR README (The extended edition is here):

- Commands start with '/'.
- Use `/help` to list all available commands. The list
- Use `/exits` to list all available exits.
- Use `/sos` to return to First Room if you're stuck.
- Rooms might try to fool you, but these three commands will always work.

Retro, text-only interface

Simple text commands

</> /go N

/go N

You head North

So long, and thanks for all the fish.

exit The First Room
enter Look out what can you see

Status updates

Connecting to Look out what can you see. Please hold.

The room is strangely warm, expressing the malaise that comes with a fever as well as a room can. (1)

New microservice serving content

Look out what can you see

A hasty message has been taped to the wall, `Not feeling well, I've gone to lie down -- Look out what can you see`

You notice:

- Monitor
- Chart

How odd. The room has a stretched tense feeling, like it is desperately trying not to sneeze. (2)

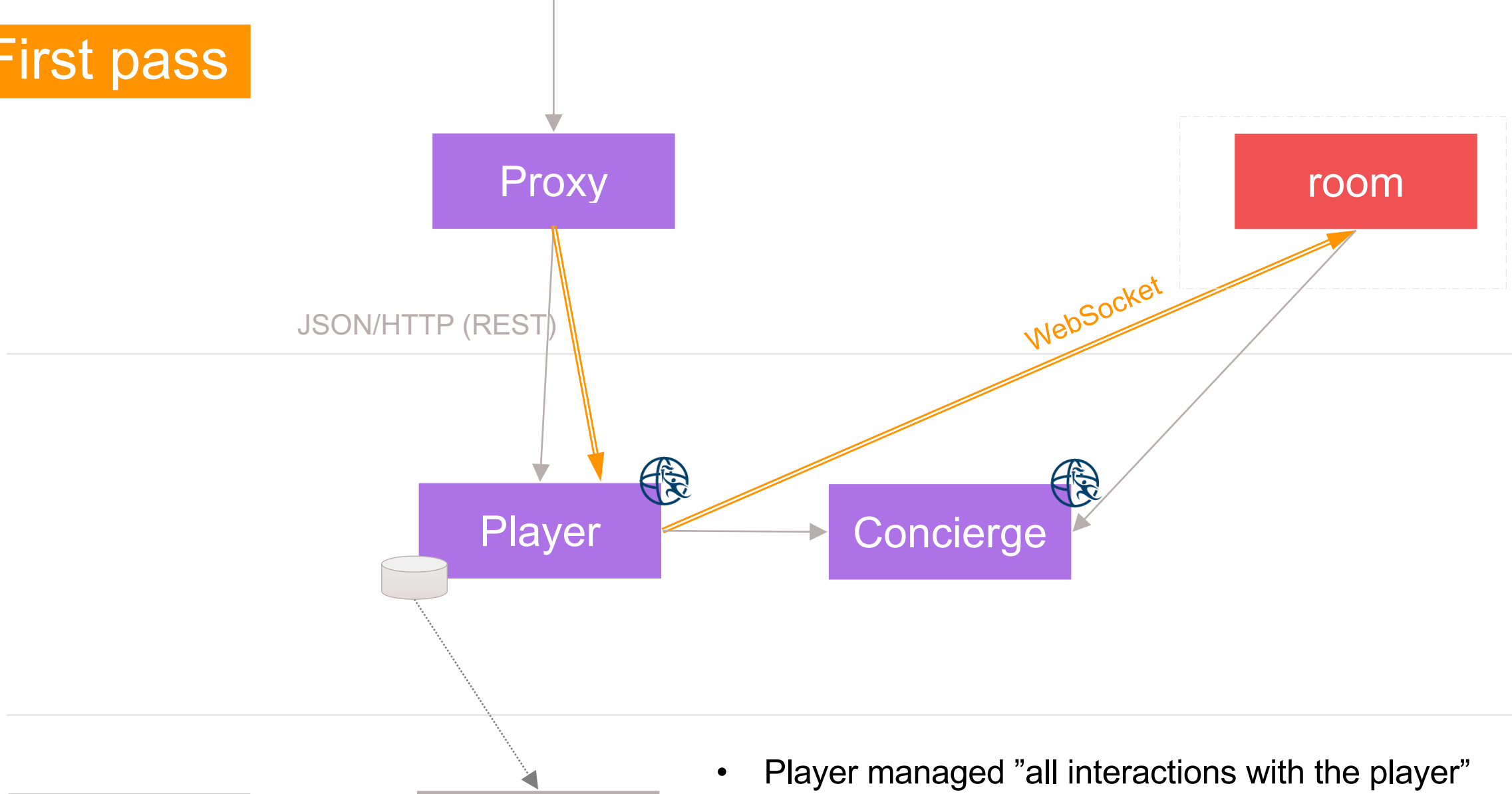
The room emits a low and rhythmic rumble, like a congested chest. Is it breathing? (3)

How odd. The room has a stretched tense feeling, like it is desperately trying not to sneeze. (4)

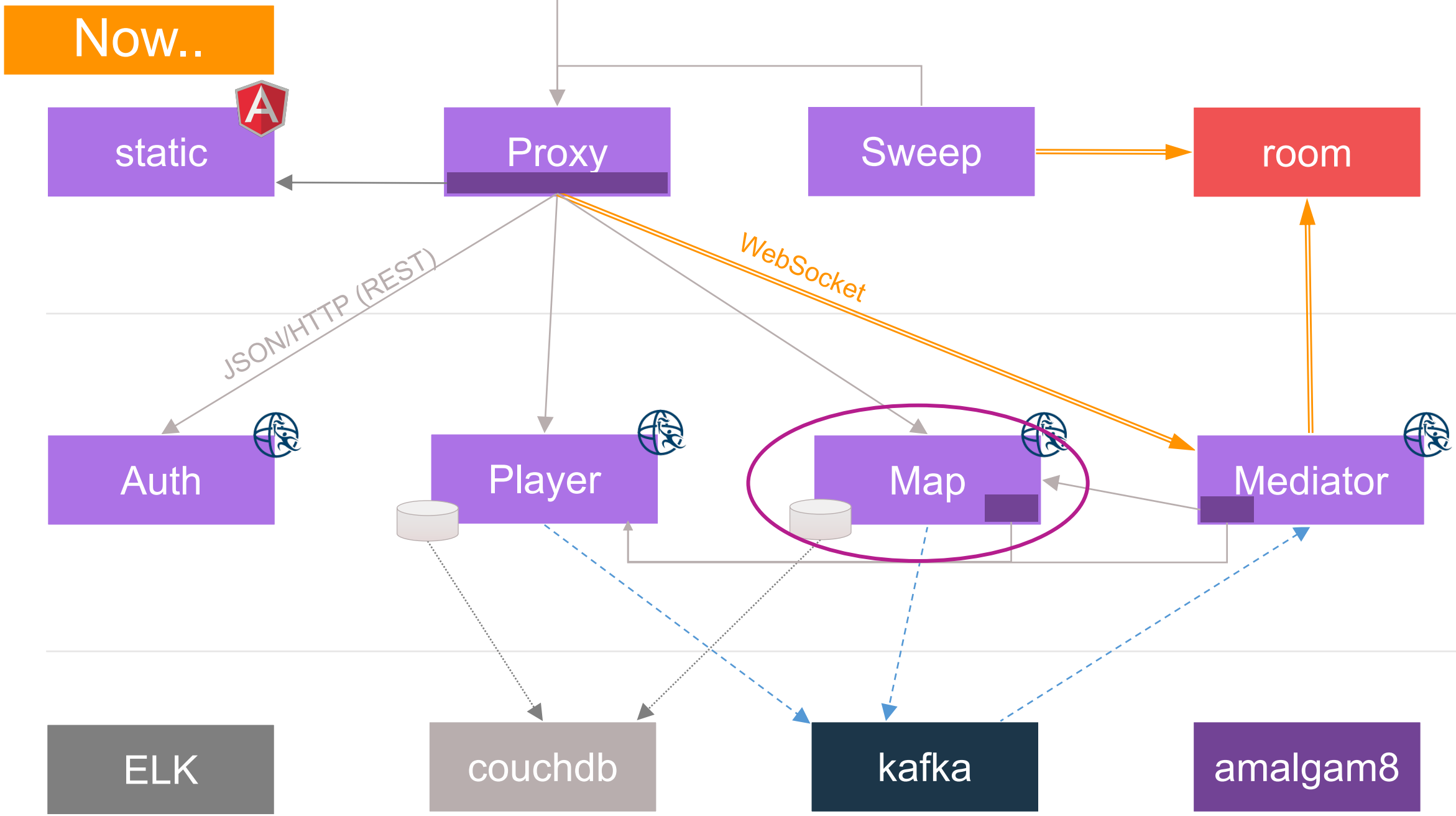
</>

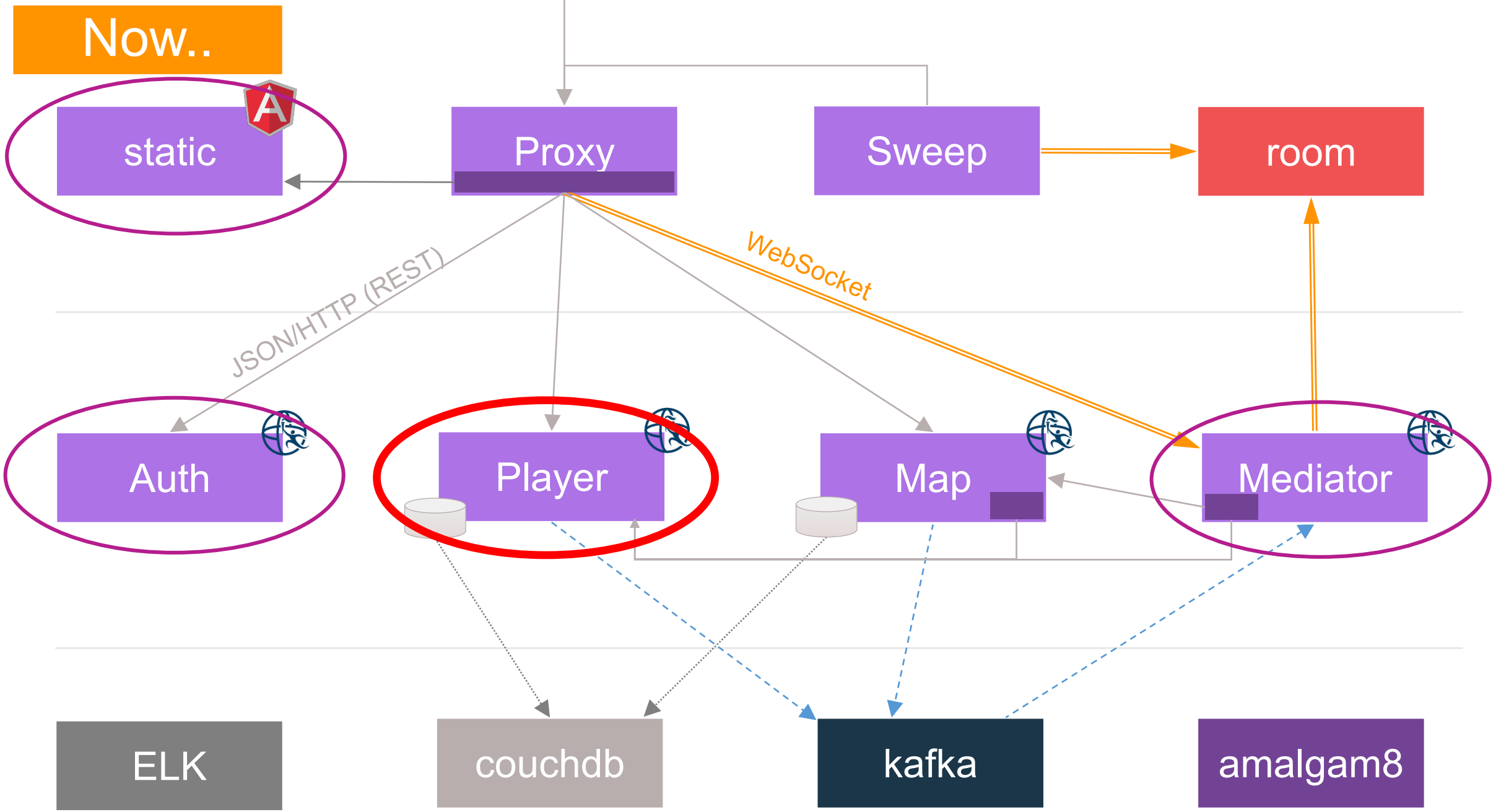
Evolution

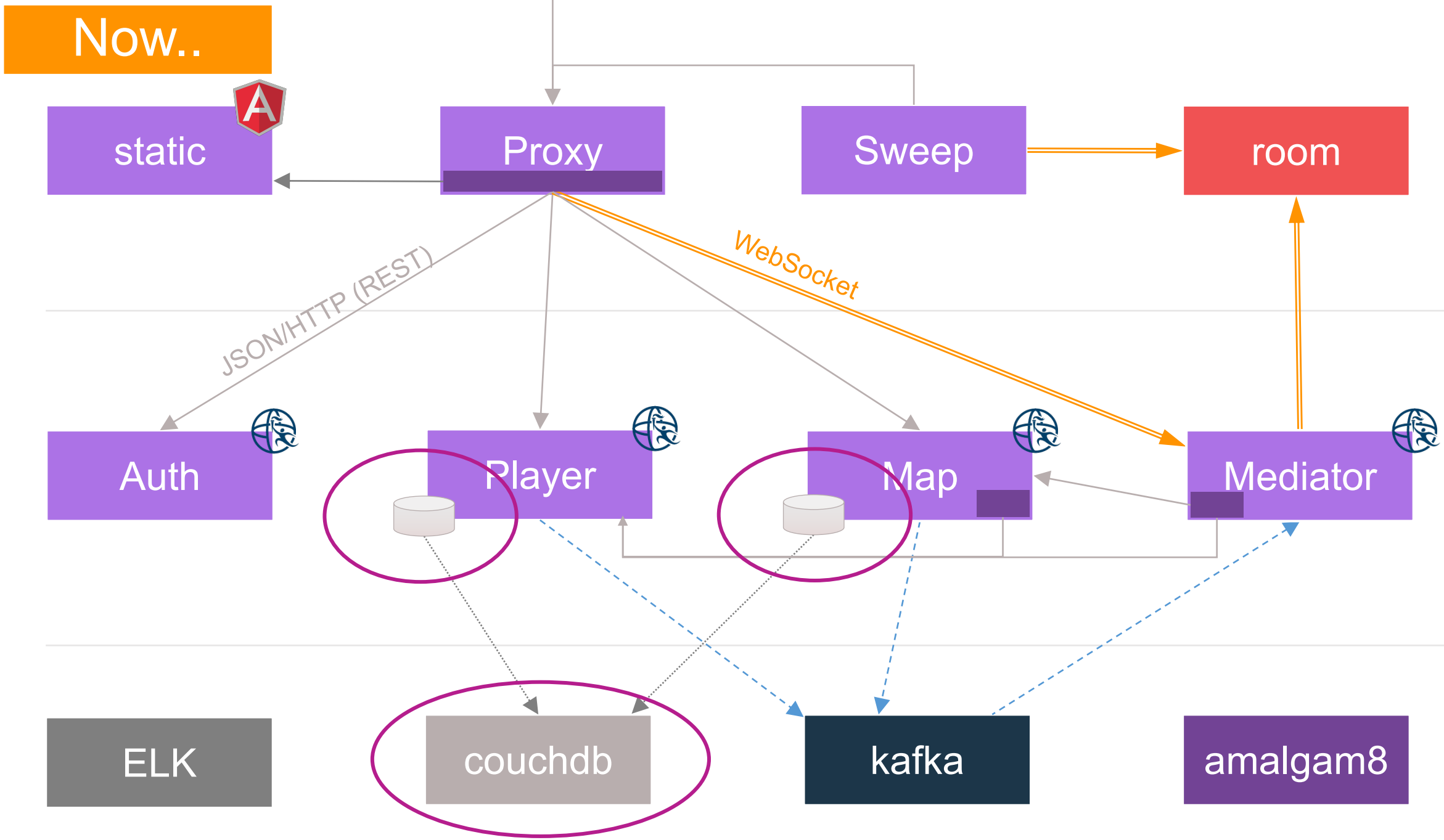
First pass

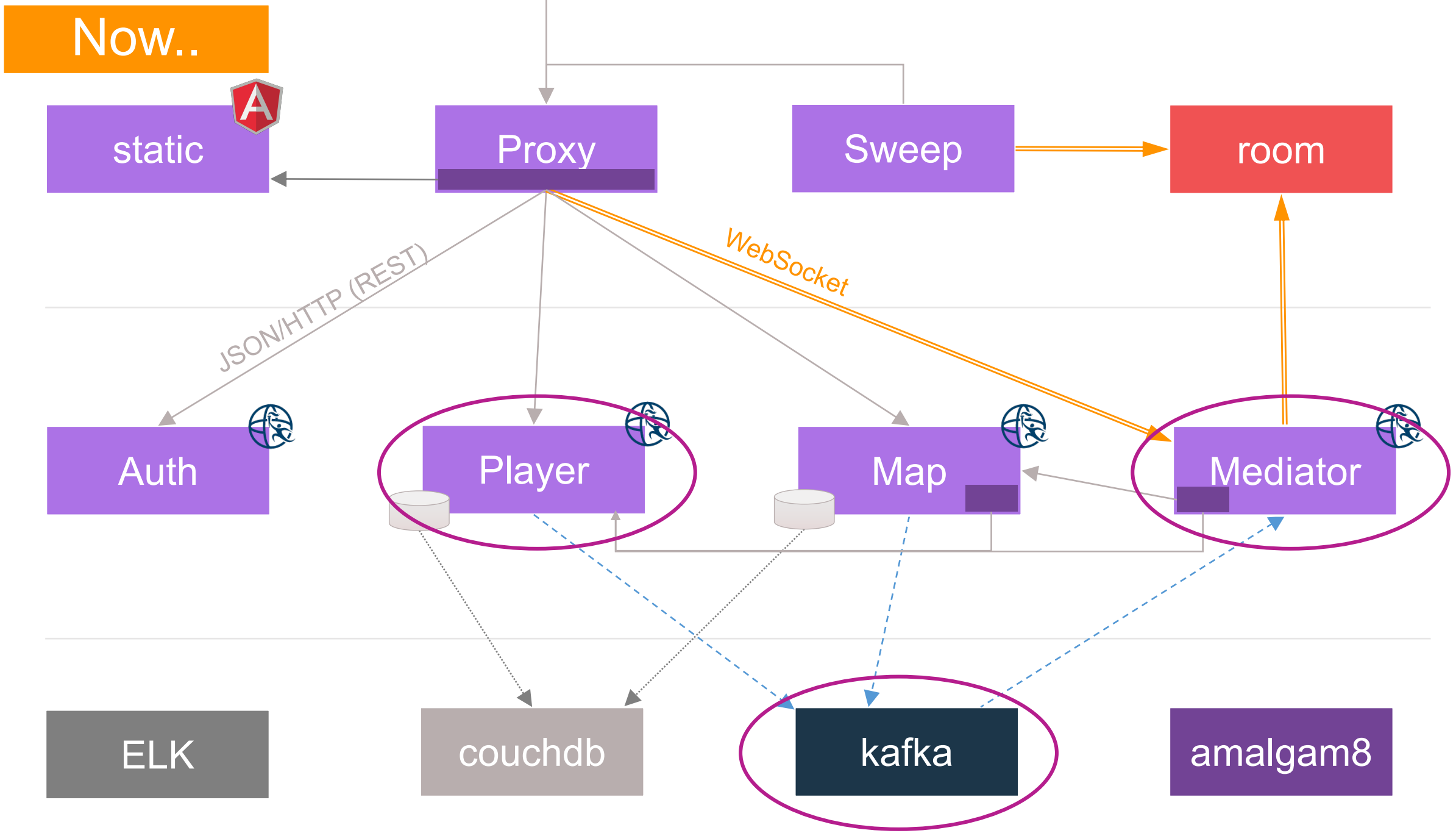


- Player managed ”all interactions with the player”
- Concierge: tell player service which room is next (/go N)
- Room (had to have one!)


Now..
static
Proxy
Sweep
room
JSON/HTTP (REST)
WebSocket
Auth
Player
Map
Mediator
ELK
couchdb
kafka
amalgam8


Now..
static
Proxy
Sweep
room
JSON/HTTP (REST)
WebSocket
Auth
Player
Map
Mediator
ELK
couchdb
kafka
amalgam8

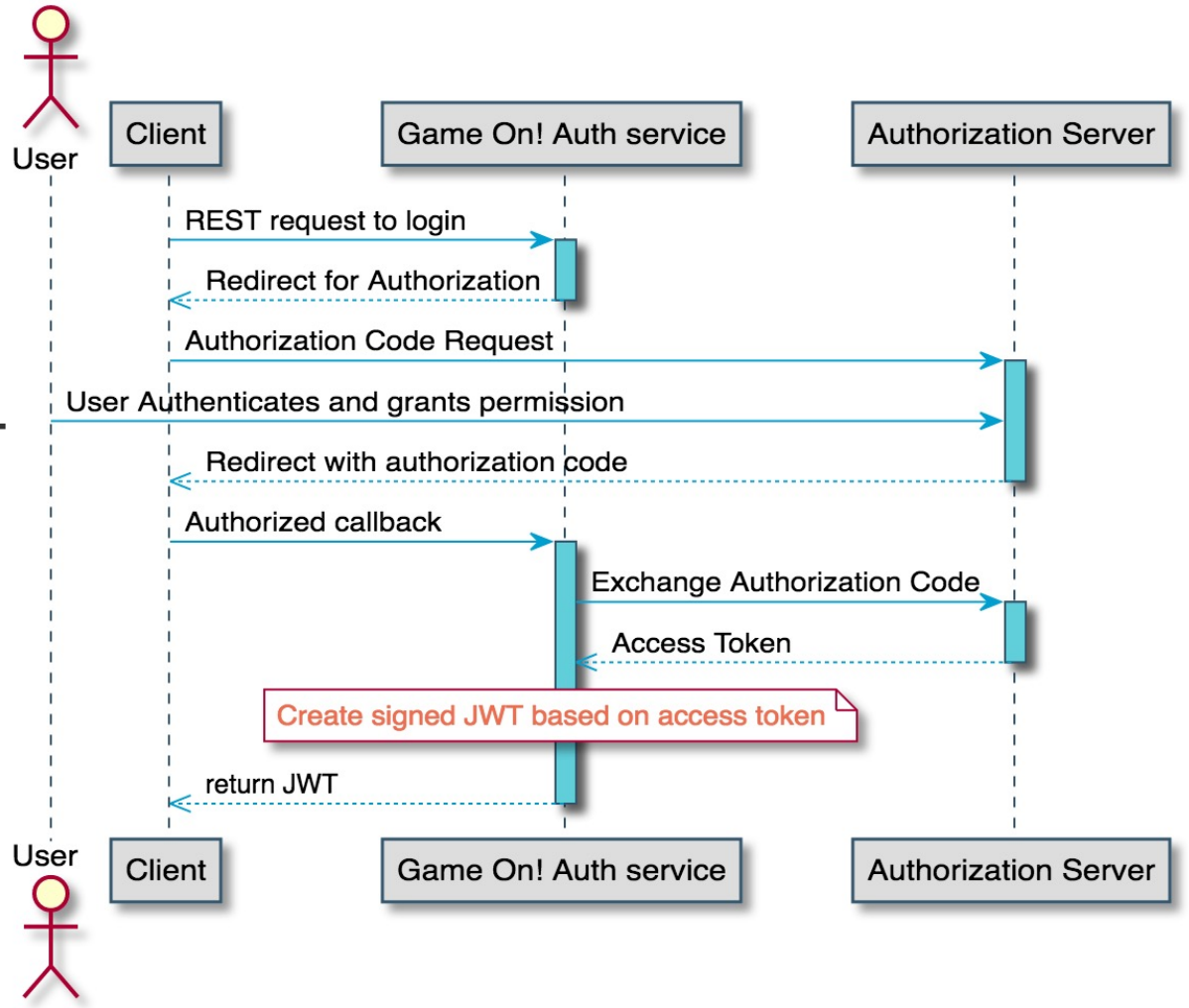

Now..
static
Proxy
Sweep
room
JSON/HTTP (REST)
WebSocket
Auth
Player
Map
Mediator
ELK
couchdb
kafka
amalgam8


Now..
static
Proxy
Sweep
room
JSON/HTTP (REST)
WebSocket
Auth
Player
Map
Mediator
ELK
couchdb
kafka
amalgam8

Security

OAuth & JWTs

- OAuth proxy
  - Application id w/ different front-end
  - Could be a gateway instead
- Access token converted into signed JWT
- System services deal only with JWT
  - gameontext.org SSL certificate
  - Well-known public key



Hashed message authentication codes (HMACs)

- Shared secrets
  - Credentials not sent on the wire
  - Used to verify identity of sender

- Map operations
  - Mutable operations require HMAC signature
  - Hashed signature used to prevent replays

- Room handshake for WebSocket
  - It is the game calling the room
  - Room answering the game

Shared Library

https://book.gameontext.org/microservices/ApplicationSecurity.html

Twelve Factors

Twelve factor applications

- “a methodology for building software-as-a-service applications”
  - Created by developers at Heroku
- Factors are independent of
  - programming language,
  - backing services,
  - cloud provider
- http://12factor.net/

The Twelve Factors

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing Services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

Git + Submodules (Factor 1: codebase)

- Root repository: https://github.com/gameontext/gameon
  - Optional use of submodules

- Key: Only builds update submodule commit levels
  - Prevents conflicts and confusion caused by humans

Containers

(Factor 2: dependencies, 5: build/release/run,
6: Processes, 8: concurrency, 10: dev/prod parity)

- Encapsulation of all dependencies
- Parity: dev -> test -> prod
- Configuration passed in via environment

- Local: Docker Compose or Vagrant
  - Pre-built images in dockerhub (this came later..)
  - Overlays for local editing

- Independent build pipelines per service to deploy containers

Liberty (Factor 2, 10, 3: config, 4: backing services, 7: port binding, 9: disposability)

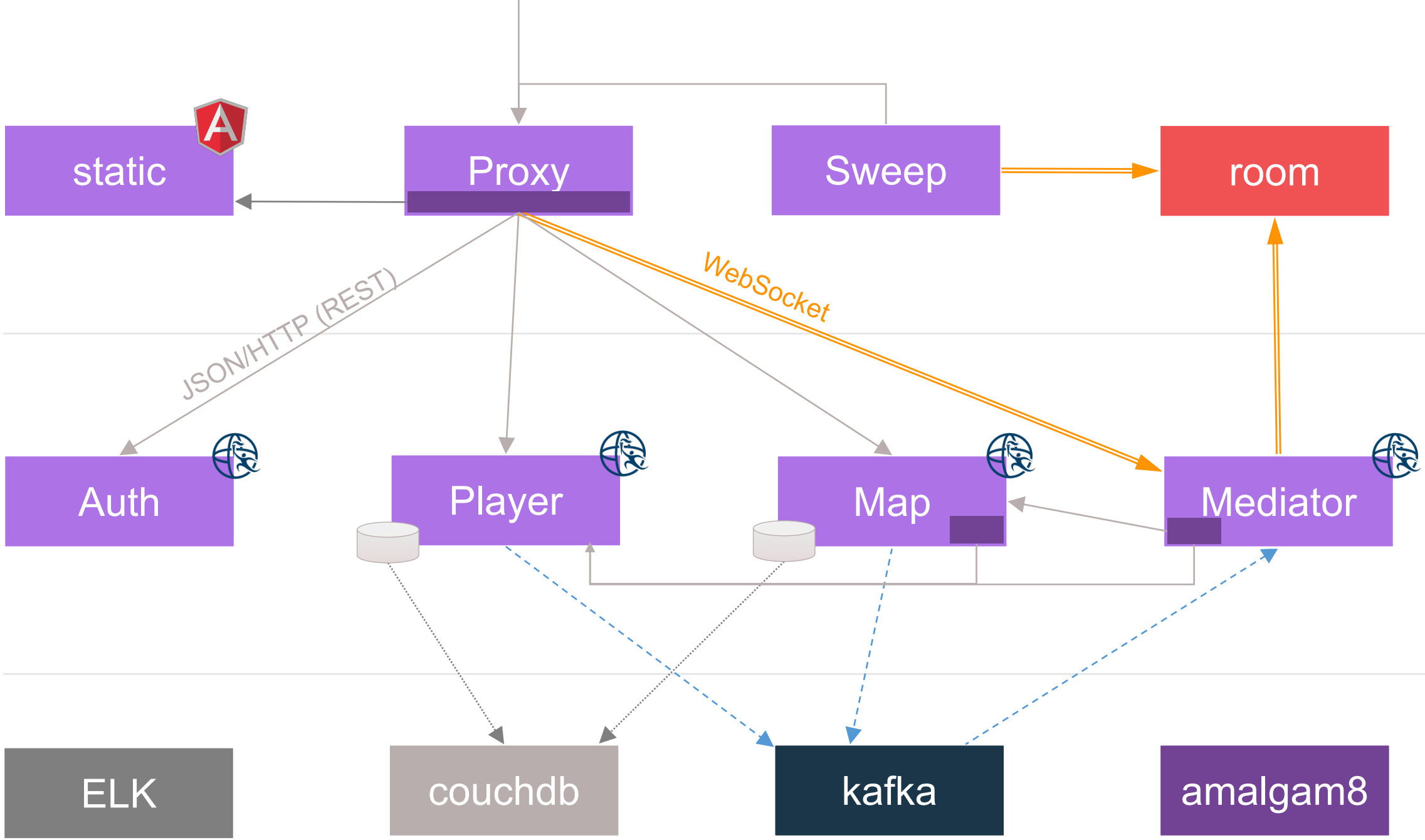
- Java services are Liberty-based
- Customizable features: Cachable Docker Layers
  - Explicit app server dependencies
  - Self-contained immutable artifact
  - Smaller war (smaller delta)

- Environment variables in server config
  - Common configuration across environments
  - Config munging not necessary
  - Composable configuration w/ dropins if required

```
<couchdb id="couchdb"
         jndiName="couchdb/connector"
         libraryRef="couchdb-lib"
         password="${env.COUCHDB_PASSWORD}"
         url="${env.COUCHDB_SERVICE_URL}"
         username="${env.COUCHDB_USER}"/>
```

```
# Install required features
RUN /opt/ibm/wlp/bin/installUtility install
    apiDiscovery-1.0 \
    bluemixLogCollector-1.1 \
    cdi-1.2 \
    concurrent-1.0 \
    couchdb-1.0 \
    localConnector-1.0 \
    jaxrs-2.0 \
    jndi-1.0 \
    jsonp-1.0 \
    ssl-1.0 \
    websocket-1.1
```

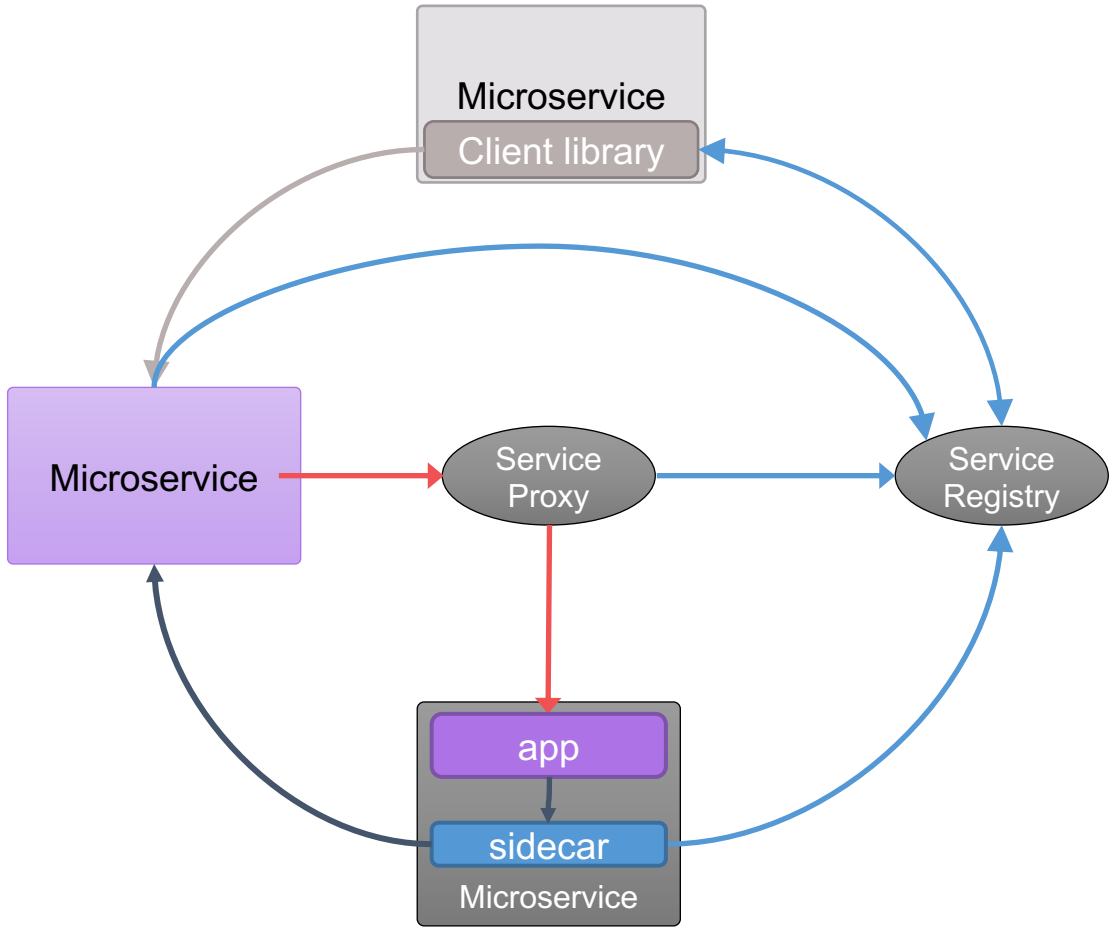
Service discovery


static
Proxy
Sweep
room
JSON/HTTP (REST)
WebSocket
Auth
Player
Map
Mediator
ELK
couchdb
kafka
amalgam8

Service registration and discovery

- Required for load balancing and scaling
- Services need to find each other
- Environment changes constantly

- Client-side or server-side?
- Client library, sidecar, or proxy?



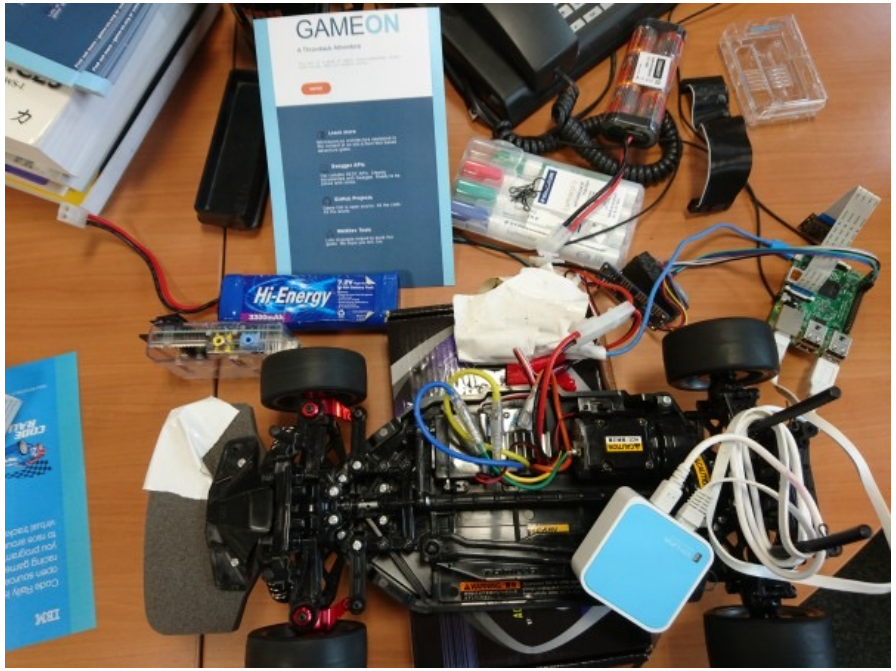
Successful?

Example rooms

- Map room
- Weather room
- Liberty car
- Pi-Cam

	-4	-3	-2	-1	0	1	2	3
3					Empty			
2			Empty	Empty	WalkThruThur (GWC)	Empty		
1		Empty	A Very Simple Room. (Alvin_Uy)	MyRoom (Simon_Boyden)	GO.CONTAINER.4 (GWC)	The Room with The Mug (game-on.org)	Empty	
0	Empty	The Node-JS Room (game-on.org)	WalkthruWed (GWC)	Rec Room (game-on.org)	The First Room (game-on.org)	GoMonday (GWC)	This is a CF in bluemix (Ant_Mazzone)	Empty
-1		Empty	A Very Simple Room. (hongsign)	A dusty room beneath the capstone... (Ozzy)	Basement (game-on.org)	This is a CF in bluemix (GWC)	Empty	
-2			Empty	Empty	A Very Simple Room. (LargeMuffin)	Empty		
-3					Empty			

(LastUpdated: 2016-04-14T21:46:43.829Z)



Notices and disclaimers

Notices and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular, purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Blueworks Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services®, Global Technology Services®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli® Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

InterConnect
2017