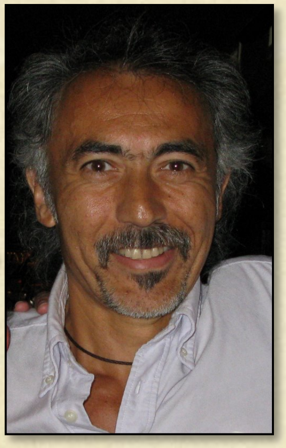
WebSocket Perspectives 2015

Clouds, Streaming, Microservices and the Web of Things

@frankgreco

Background



- Director of Technology
- Chairman NYJavaSIG (javasig.com)
- Largest Java UG in NA 8k+ members
- First Java UG ever! Sept 1995
- email: frank.greco@kaazing.com
- Twitter: @frankgreco
- Yell: “Hey Frank!”

WIN A COPY!

KAAZING

WIN A COPY!

1. Introduction to HTML5 WebSocket
2. The WebSocket API
3. The WebSocket Protocol
4. Building Instant Messaging and Chat over WebSocket with XMPP
5. Using Messaging over WebSocket with STOMP
6. VNC with the Remote Frame Buffer Protocol
7. WebSocket Security
8. Deployment Considerations

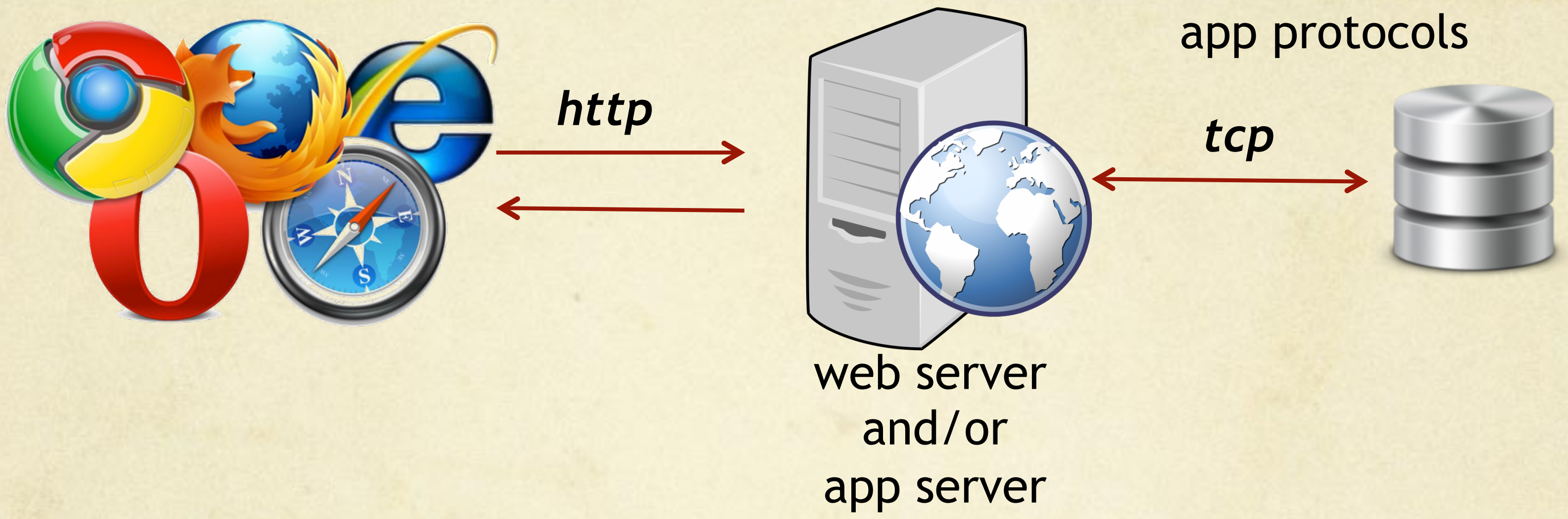
KAAZING

Outline – Things to Consider

- Web Communications Then and Now
- Web APIs
- Communications Models, Protocols, Frameworks
- Where WebSocket Fits
  - IoT/WoT
  - Microservices Transports
  - Cloud Connectivity

Web Communication

Web – "over the firewall" (early 90's – 2011)

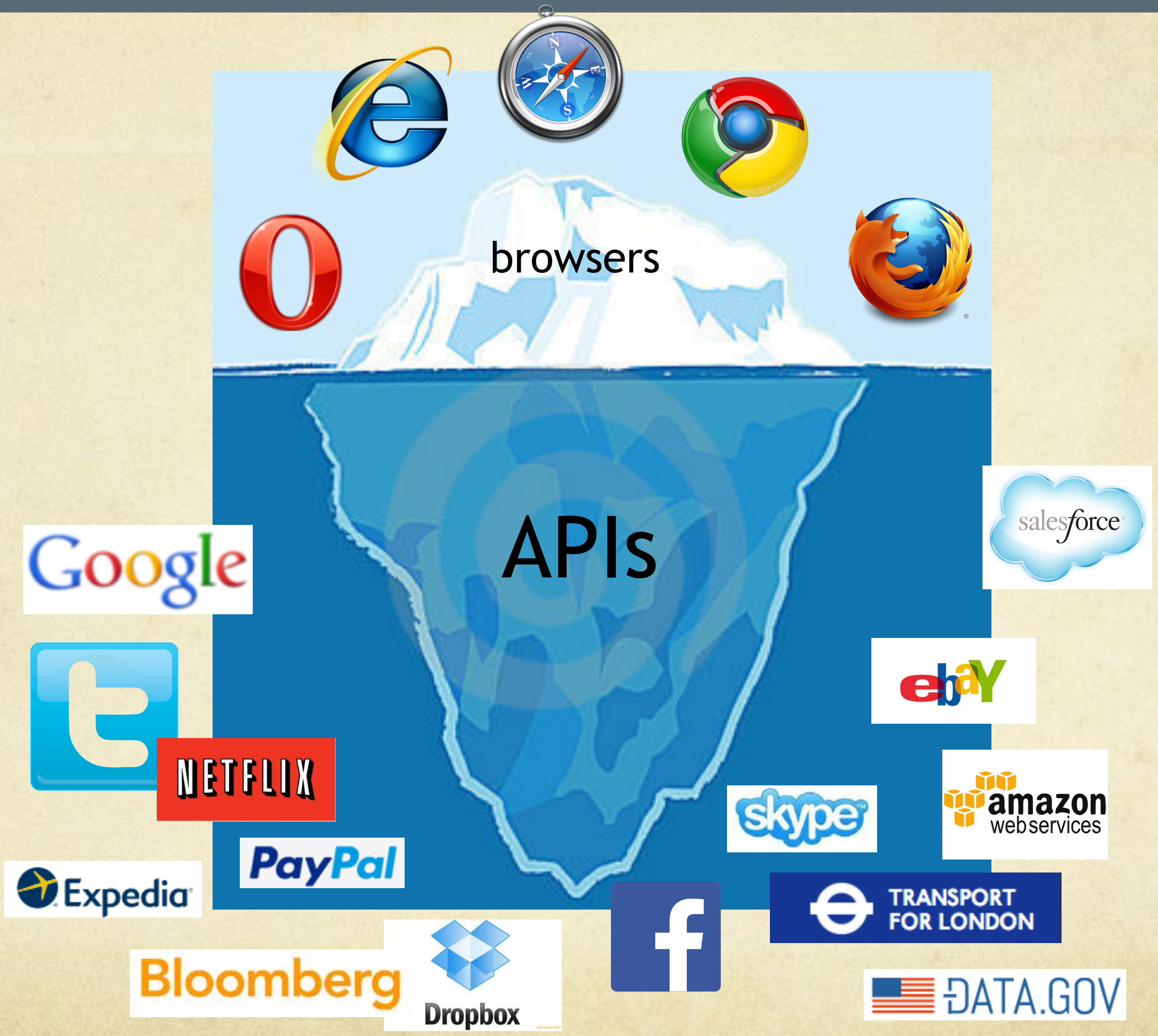


Page and Visitor hits used to be the report card

You are visitor ←Remember these?

Web APIs

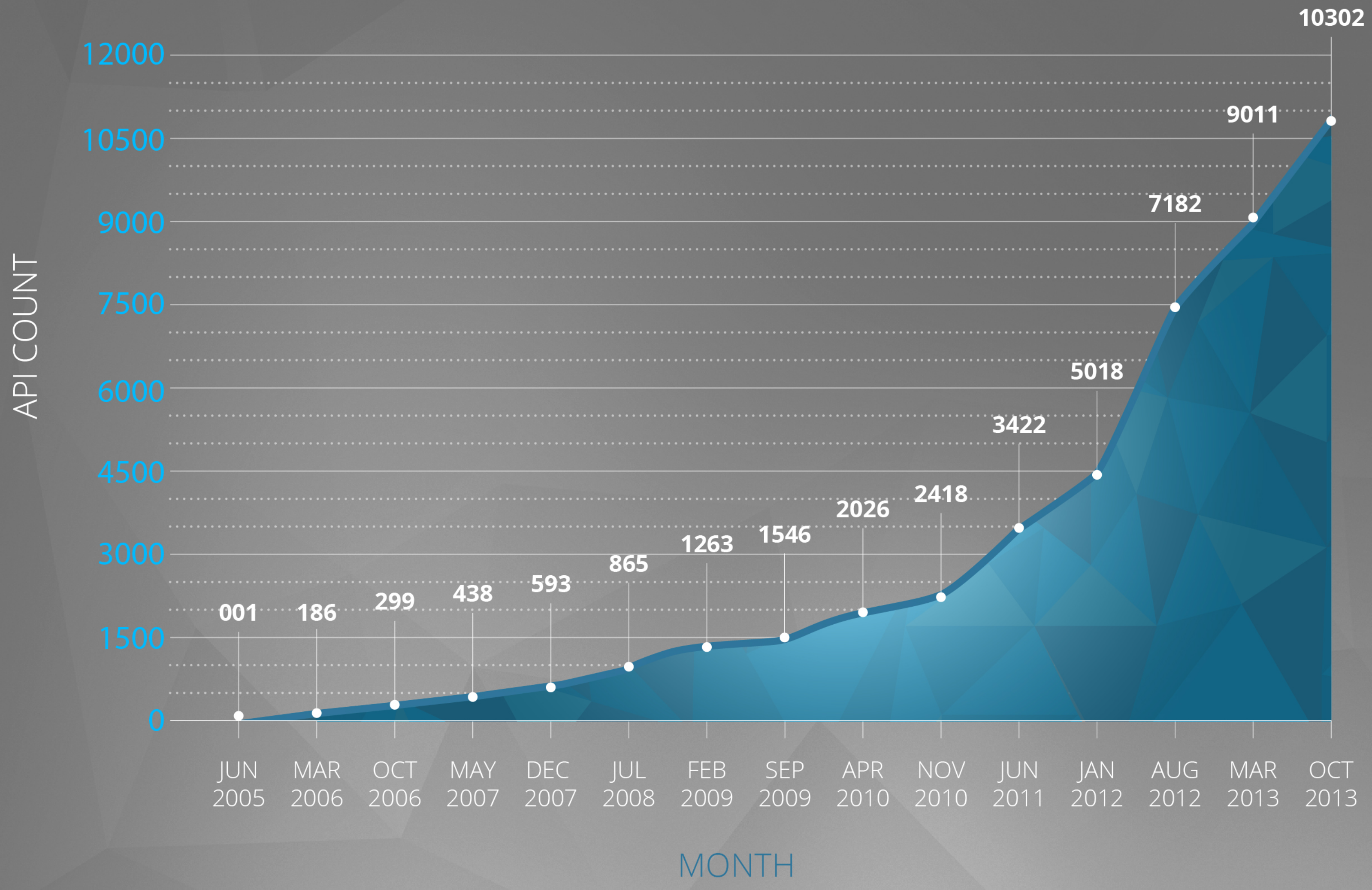
The Hidden Web – Most of the Web is Not Visible



http://thumbs.dreamstime.com/x/iceberg-23503494.jpg

ProgrammableWeb

Growth In Web APIs Since 2005



2015

You are API call # 3,617,293,229 ***today***

2012

You are visitor # 2,391 this year

KAAZING

Explosion of Open Web APIs

- APIs from Everywhere, Consumed by Every [one|thing]
- ~14K public APIs and even more Mashups
  - programmableweb.com/apis/directory
  - Amazon, Facebook, LinkedIn, AT&T, Google, Microsoft, NYTimes, Orange, SalesForce, Telefonica, Twitter, Visa, Vodafone, Bloomberg, NYSE, Thomson-Reuters, etc.
- Over time, more will be event-based
- Enterprise and B2B APIs
- Services... Services... Services...

programmableweb

API FOR THAT

DATA.GOV

PublicAPIs by Mashape API Platform

KAAZING

Chuck Norris has an API and It can Kill you

```
% groovy -e \
'println new URL(\"http://api.icndb.com/jokes/random\").getText()”
| grep joke | tr -d "{}" | sed -e 's/.*joke": "//' -e 's/".*$//’
```

Chuck Norris compresses his files by doing a flying round house kick to the hard drive.

```
% groovy -e \
'println new URL(\"http://api.icndb.com/jokes/random\").getText()”
| grep joke | tr -d "{}" | sed -e 's/.*joke": "//' -e 's/".*$//’
```

Chuck Norris played Russian Roulette with a fully loaded gun and won.

Using the Web without a Browser

```
% REPO=kaazing/gateway

% printf 'As of %s, repo [%s] has %s forks\n' \
    "`date +%D`"\
    $REPO\
    `curl --user "XXXXX:YYYYY" https://api.github.com/repos/$REPO 2>&1\
    grep -i forks_count |\
    cut -d: -f2 |\
    tr -d ,`


As of 10/16/15, repo [kaazing/gateway] has 34 forks
```

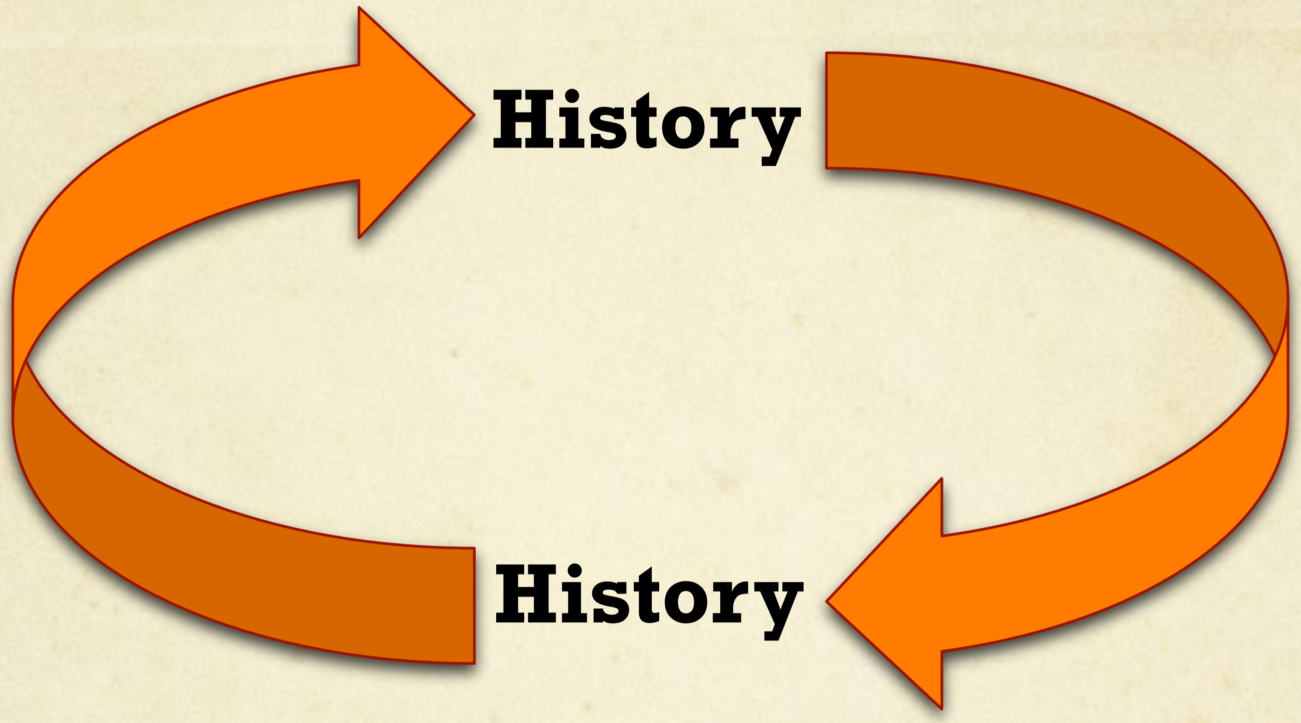
Services integration from anywhere on the planet to any device.

Why Am I Mentioning This?

Why Am I Mentioning This?

Services integration is important.
Asynchronicity is next.

A Primary Tenet of Computing



If History Repeats Itself, Is There No Future?

KAAZING

Reactive Programming, Streams and Events – The Alive Web

The Reactive Manifesto

Reactive programming is programming with asynchronous data streams.

Apache Kafka
A high-throughput distributed messaging system.

Spark

Amazon Kinesis

Clojure

Reactive-Extensions / RxJS

Watch 288 Sta

The Reactive Extensions for JavaScript http://reactivex.io

180 contributors

mattp

t commit e45382 7 hours ago

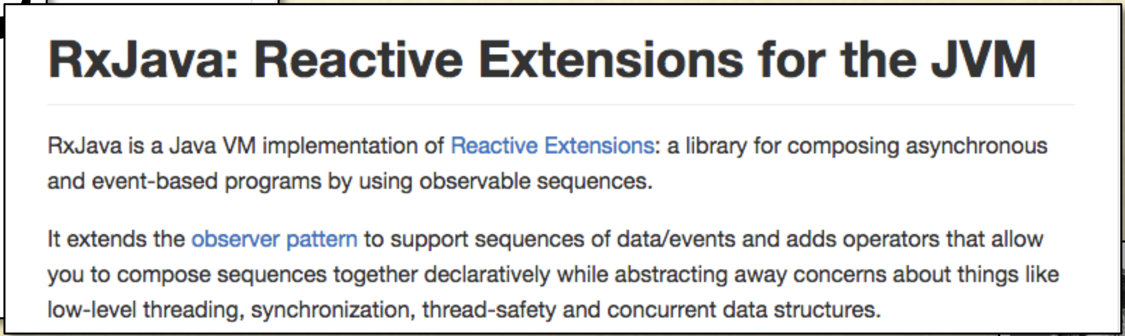
.nuget 2 years ago

dist

akka

Scala

Java 8 λ


RxJava: Reactive Extensions for the JVM

RxJava is a Java VM implementation of Reactive Extensions: a library for composing asynchronous and event-based programs by using observable sequences.

It extends the observer pattern to support sequences of data/events and adds operators that allow you to compose sequences together declaratively while abstracting away concerns about things like low-level threading, synchronization, thread-safety and concurrent data structures.

node js

bacon.js

Functional Reactive Programming

Web Communication Models for Asynchronous Streams

KAAZING

Web Communication Protocols for Event-Driven World

http:// HTTP/1.1 - 1997 RFC 2068	Great for caching, synchronous req/resp, XHR for client async polling (AJAX), Comet for push
HTTP/2 HTTP/2 - 2015 RFC 7540	Binary, mux over TCP, header compression, server can push into client cache, AJAX/Comet, 30-50%+
HTML SSE HTML5 - 2009 W3C	Standardization of Comet (push), uni-directional, uses HTTP
WebSocket - 2011 RFC 6455	Binary/Text, full-duplex, persistent connection, *TCP for the Web* *JSR 356/JEE7*

KAAZING

Web Communication Mechanisms for Event-Driven World

Web Notifications **2015 - W3C**	Browser Notifications outside webpage, can use with Service Workers (and WS)
Push API **2015 - W3C**	Scripted access to push data, use with Service Workers (and WS)
Project Tyrus **2013-2014**	Java/WebSocket JSR 356 – Glassfish OpenMQ Java/C API
webhooks building a more programmable web **2007**	User-defined HTTP callbacks (POST)
node JS socket.io **2009**	Event-driven architecture, non-blocking I/O API, JS for server

KAAZING

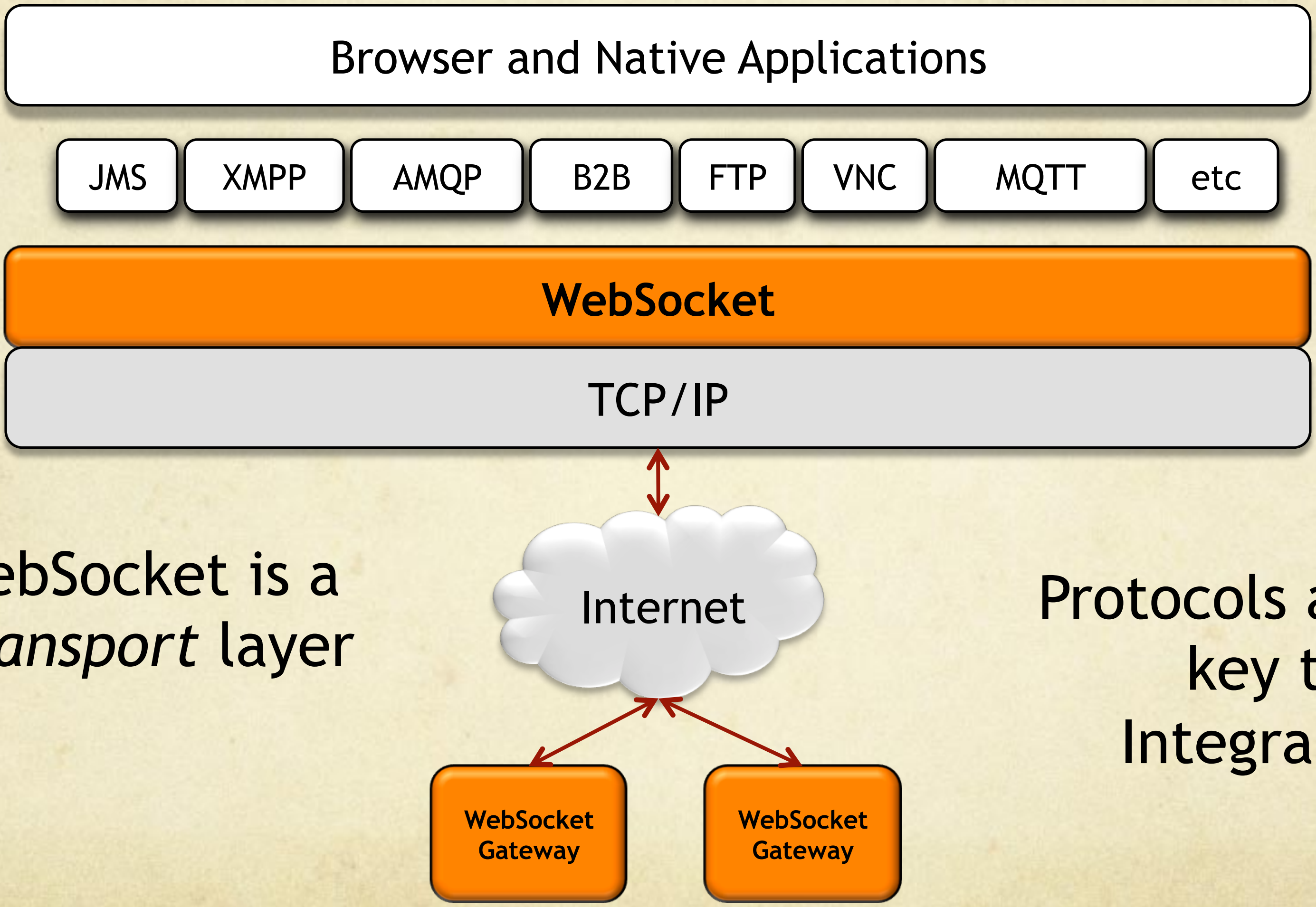
Do I Still Use WebSocket with HTTP/2?

- WebSocket is not a REST (JAX-RS/Jersey) replacement.
- WebSocket is complementary to HTTP (and REST)
- Simple Notifications can be easily done with HTTP
- Higher level APIs for Polyglot world Needed (e.g., JS)
- WebSocket used for Full-duplex Persistent connection… a *TCP for the Web*
- Non-Browser use is where it gets interesting

WebSocket Projects

- Kaazing
- Java EE – **JSR 356, Project Tyrus, Grizzly**
- Node.js/socket.io/SockJS/engine.io
- ActiveMQ
- Tomcat
- Jetty
- Oracle Glassfish
- Play Framework – Reactive Apps
- Rabbit MQ
- JBoss
- IIS/ASP .NET 4.5
- PHP, Objective-C, Ruby, Python, C/C++, JVM-langs…
- Many more… (100+ implementations)

Protocol Layering is Possible



WebSocket is a *Transport* layer

Protocols are the key to Integration

KAAZING

What do Protocols give us?

Who handles retries?

How can we guarantee delivery?

How do we handle publish/ subscribe semantics?

What do we do with slow consumers, last value cache, etc?

How do we handle market data?

What if the client is not currently active?

How do I handle entitlements? ACL?

What about partial data?

KAAZING

WoT Now?

The Web of Things

Internet of Things (IoT) – Java ME

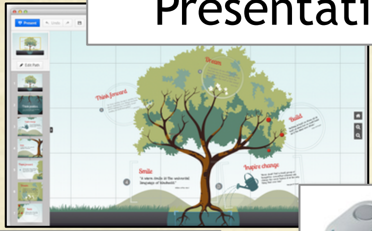
+

Heterogeneity + Scale + Usability

The World is Naturally Event-based (“real-time”)

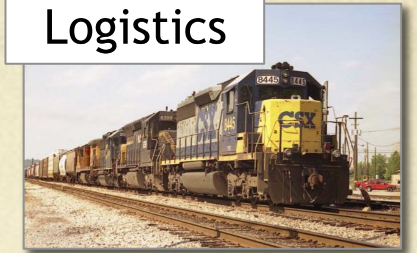
Are you in class right now?
I hate this class. Nothing ever happens.
Hey have you eaten lunch yet?
Send

Communication

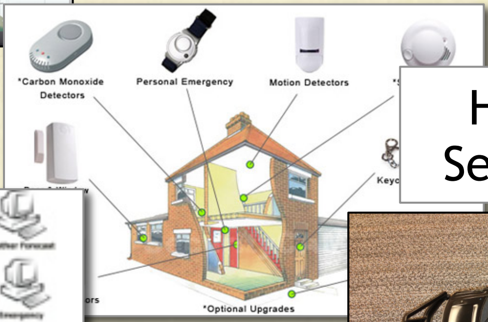


Presentation

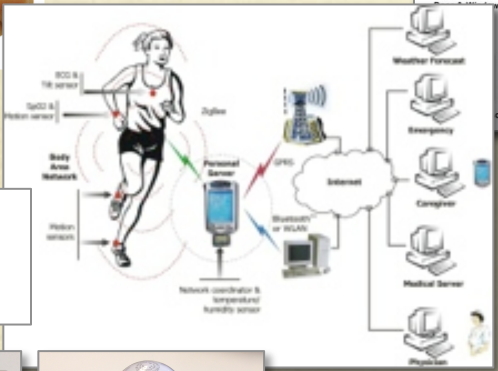
Music



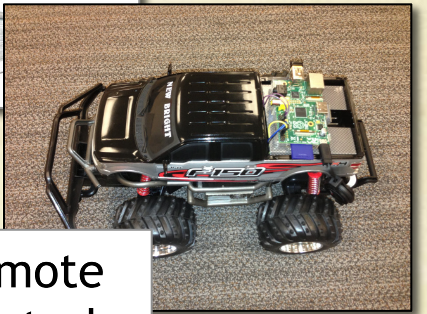
Logistics



Home Security



Health Monitoring

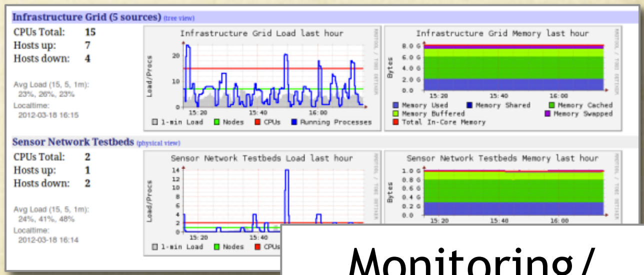


Remote control

Risk Management

Intelligent Appliances

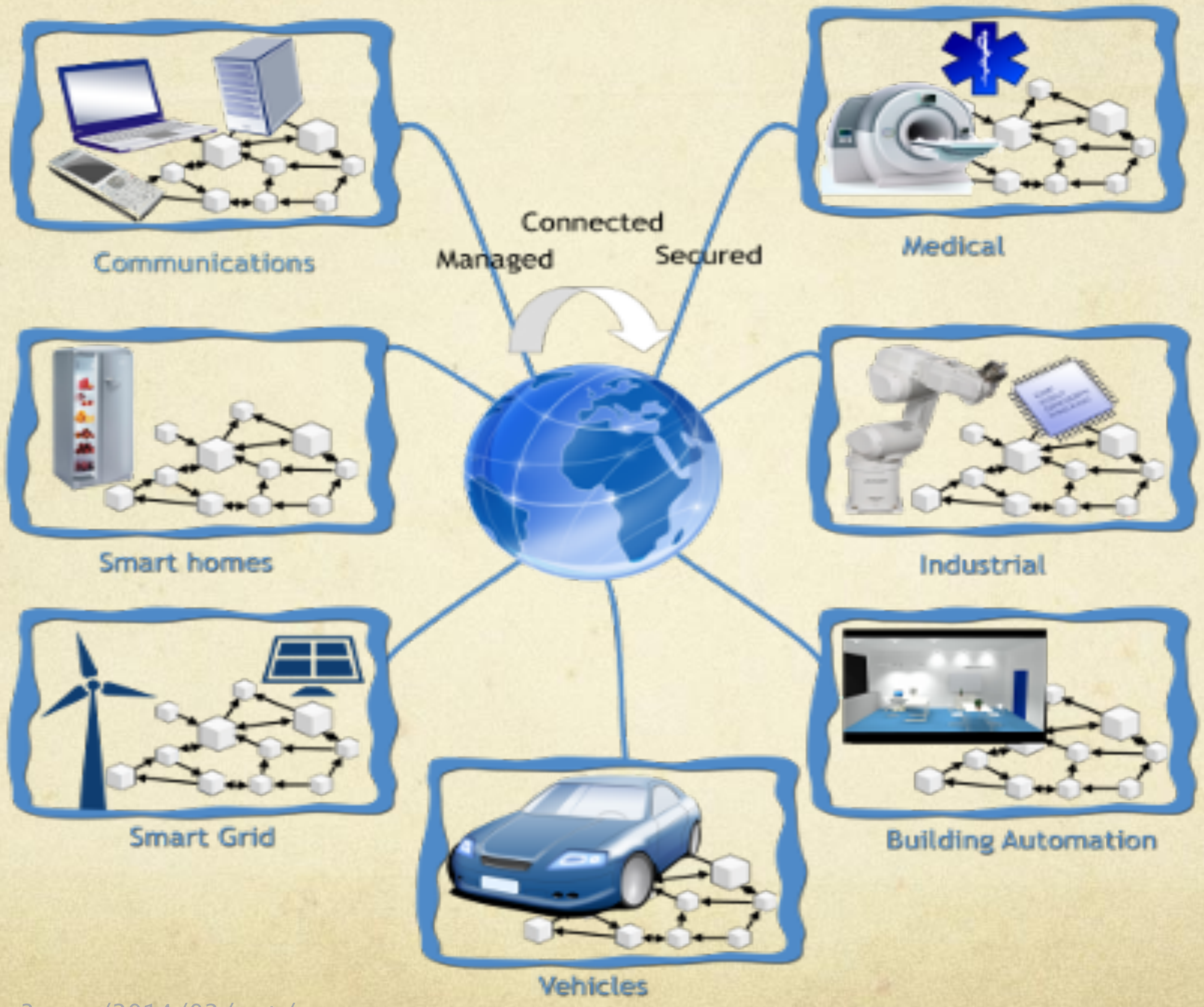
Local Transportation


Infrastructure Grid (5 sources)
CPUs Total: 15
Hosts up: 7
Hosts down: 4
Infrastructure Grid Load last hour
Infrastructure Grid Memory last hour
Sensor Network Testbeds
CPUs Total: 2
Hosts up: 1
Hosts down: 2
Sensor Network Testbeds Load last hour
Sensor Network Testbeds Memory last hour

Monitoring/ Management

KAAZING

Web of Things – Its All About SERVICES!



http://www.w3.org/2014/02/wot/

WoT does this have to do with the Web?

IoT/IIoT – Connectivity isn’t Sufficient

- No formal API standards
- Many protocol standards – interoperability low
- No common, wide-reaching frameworks
- No composition possibilities
- Difficult to leverage economies of scale
- Barrier to entry is high for millions of app developers

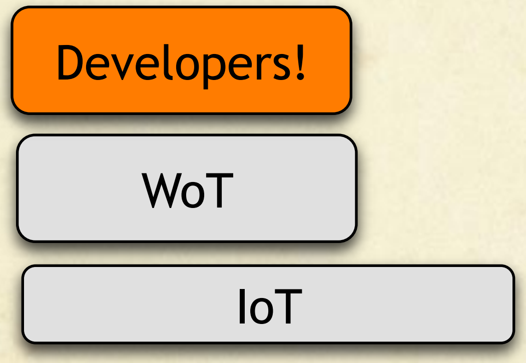
Here's Where the Web Comes In

- IoT – Internet of Things

  Embedded computing endowed with Internet connectivity

- WoT – Web of Things

  Application and Services layer over IoT



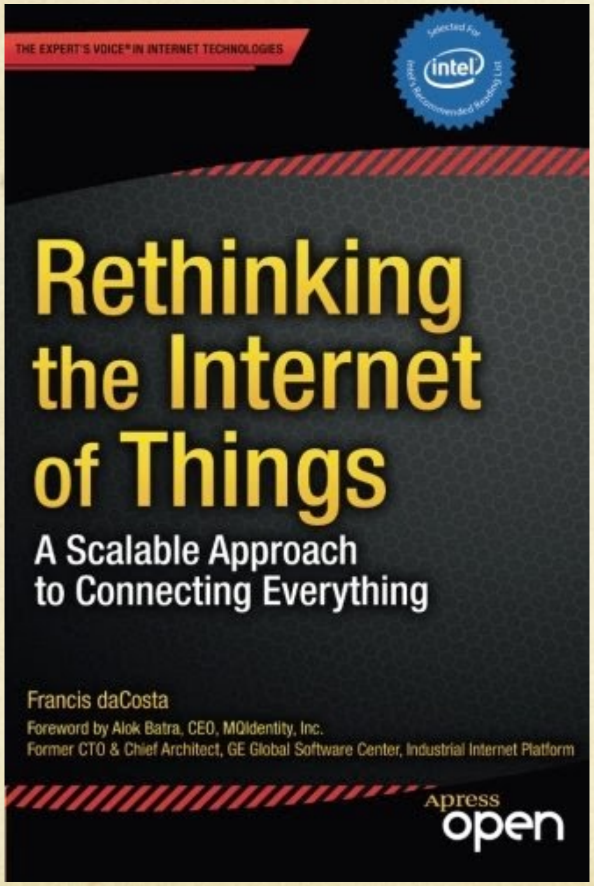
Here’s Where the Web Comes In

- Apply the benefits of the Web to IoT
- WoT is a uniform interface to access IoT functionality
- Provides the abstraction for control/monitoring (sensors/actuators)
- Accelerates innovation
- Deployment, development, interoperability, economy of scale…

But Is HTTP the Right Choice?

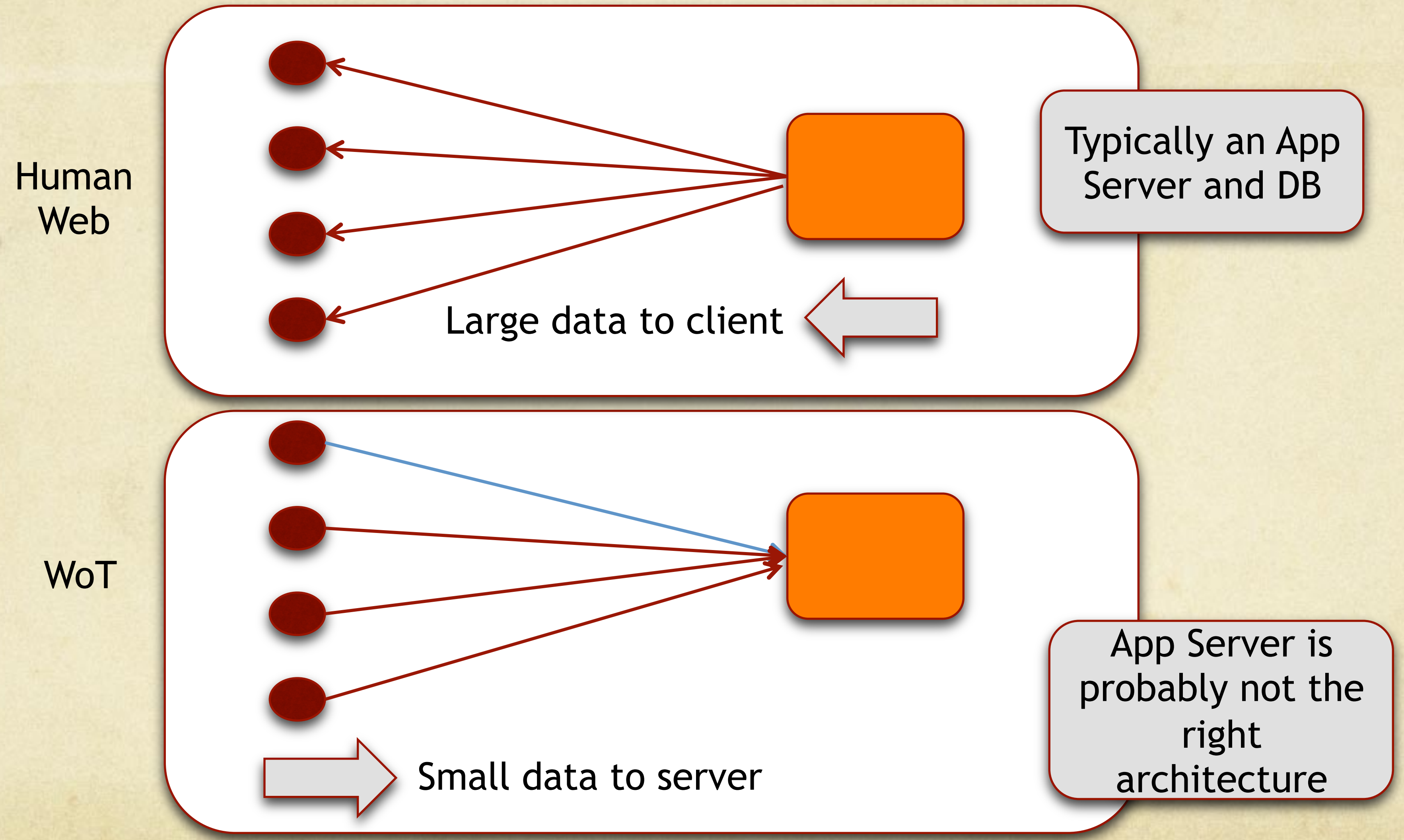
- Disadvantages of HTTP Request/Response
- Lack of resiliency and robustness
- Enterprise events retrieved by resource intensive polling techniques
  - Much bandwidth is wasted
  - Information can be delayed
- Composite services brittle and lack transactionality
- Enterprises learned advantages of ESB 10+ years ago
- See failures of CORBA, Sun RPC, etc.
- Clumsy AJAX/Comet workarounds to simulate real-time

The Message is the Medium


THE EXPERT'S VOICE® IN INTERNET TECHNOLOGIES
Selected for intel Intel's Recommended Reading List
Rethinking the Internet of Things
A Scalable Approach to Connecting Everything
Francis daCosta
Foreword by Alok Batra, CEO, MQIdentity, Inc.
Former CTO & Chief Architect, GE Global Software Center, Industrial Internet Platform
Apress open

"…terse, self-classified messages, networking overhead isolated to a specialized tier of devices, and ***publish/subscribe*** relationships are the only way to fully distill the power of the coming Internet of Things" – Francis daCosta

Data Flow – Human Web vs WoT

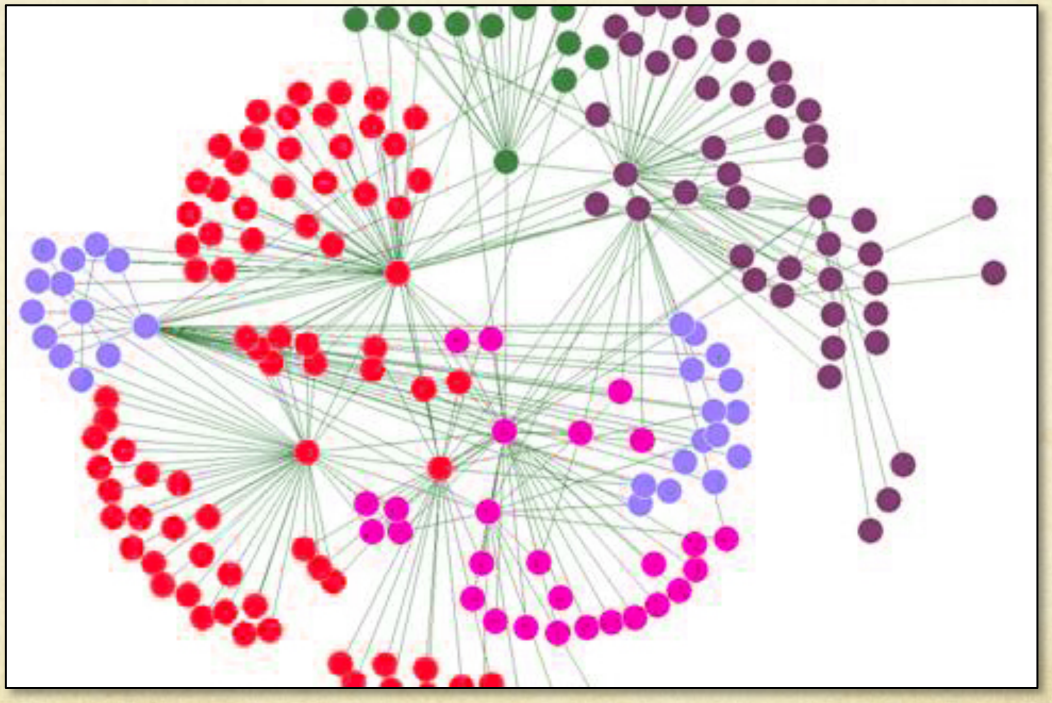


Human Web
Typically an App Server and DB
Large data to client
WoT
Small data to server
App Server is probably not the right architecture

Do human-readable protocols make sense for non-humans?

KAAZING

Microservices



Why are we talking about Microservices?

- It's an SOA (*lightweight SOA*)
- It's SOA without WS-*, SOAP, etc, crap
- Older technique now useful with modern infrastructure
- An App is a Collection of Services
- Nothing really "micro" about Microservices
- If you need more than two pizzas to feed the team with the largest service, its not small enough

Monoliths vs Microservices

Monoliths

- Long builds, complex internals, scale issues
- Scale by replicating entire monolith on multiple servers
- Hard to modify
- Not necessarily bad – depends on team

Monoliths vs Microservices

Microservices

- Small services – more agile
- Scale by replicating services
- Independent distributed services
- The Unix way
  - `% cat myfile | tr "A-Z" "a-z" | tr -cs 'a-z' '\n' | sort | uniq`
- Requires more management
- Still early

But we've had this idea for a while…
Let's take a step back

A Trip Down Memory Lane…

IBM VM/370

In the beginning…

History of Separate, Protected Environments

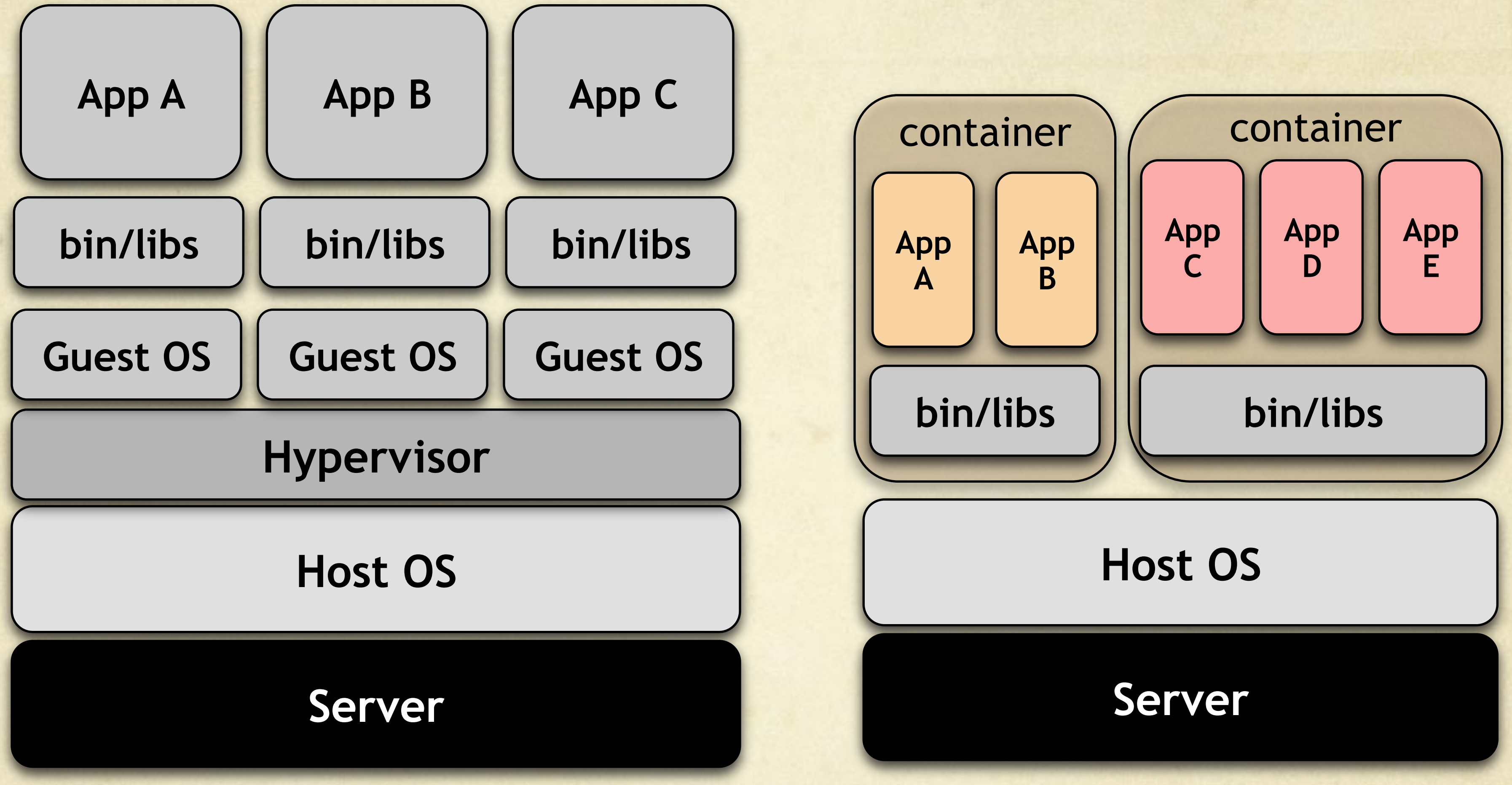
- IBM VM/370 – 1972
  - hypervisor emulated a machine
  - Separate addr space, virtual devices, fs
- Unix chroot(2) – 1979 Unix V7
  - Created a virtual root of fs
  - Useful for testing a clean environment
  - Shared users, procs, network – imperfect
- BSD Jails - 2000
  - Virtual root fs, hostname, IP addr, users, su
  - Still shared host OS

History of Separate, Protected Environments

- Solaris – Zones/Containers - 2004
  - Totally isolated, secure system resources
  - “Zones” renamed “Containers” then back to “Zones”
  - Separate CPU resources, memory and network
  - Very low overhead. No hypervisor required
- LXC – Linux Containers – 2008
  - Run multiple Linux instances on a single Linux
  - Uses cgroups – to manage cpu, memory, i/o, of a collection of processes
- Docker – 2013
  - Auto deployment
  - Adds its own libcontainer for linux virtualization
  - Rides PaaS trend

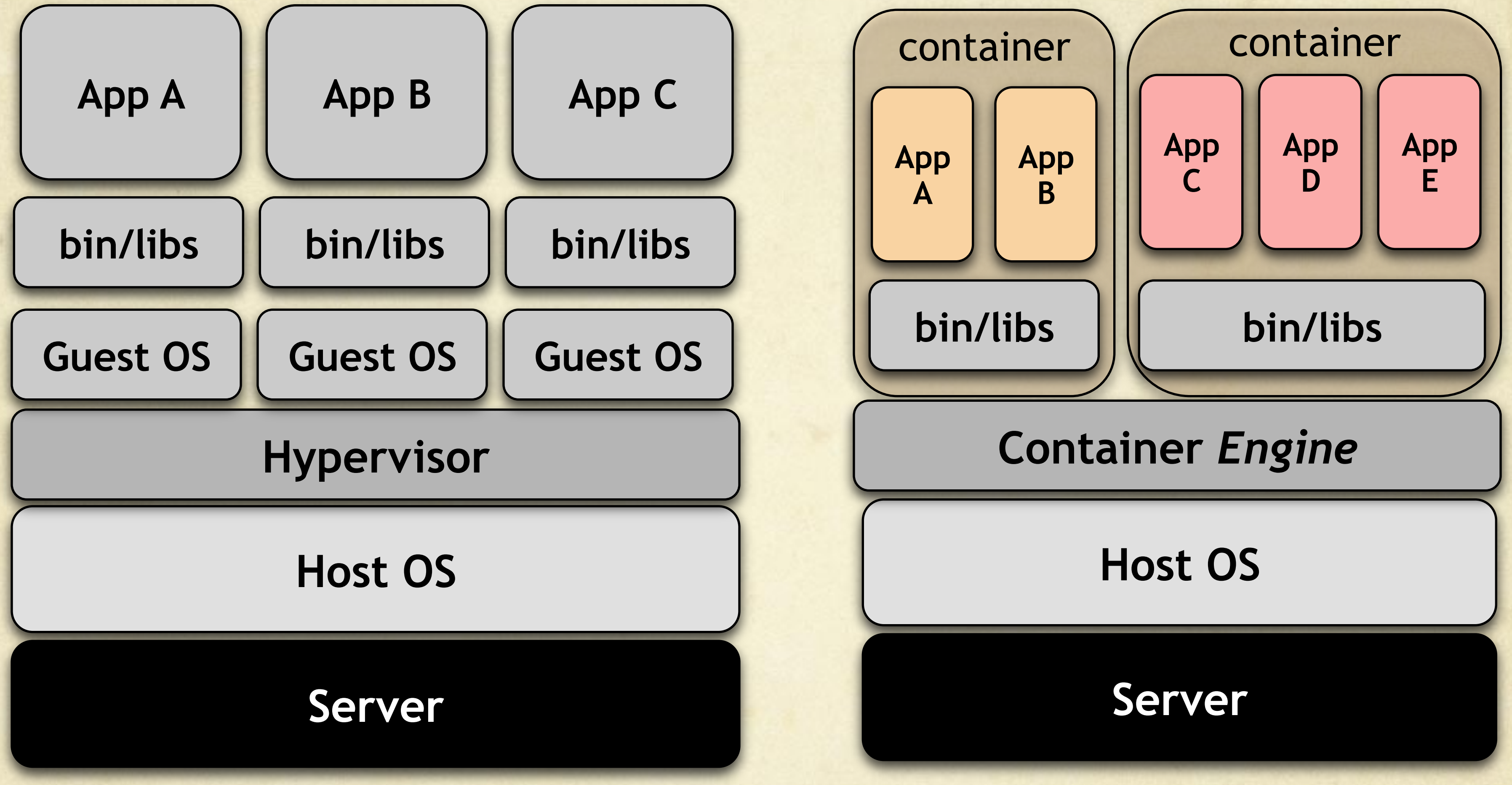
KAAZING

Containers vs Virtual Machines (VM)



KAAZING

Containers vs Virtual Machines (VM)



KAAZING

Clouds and Microservices – the bottom line

- More services per an OS
- Greater Services Mobility – dev and ops
- Easier application patching
- Faster provisioning
  - 10 min for VM, 10 sec (or less) for microservice
- Container internals visible to help maximize optimization
- Avoids cloud framework lock
- Intercloud portability
- Allows services to be located most appropriate part of architecture
- Allows policies to be applied per container

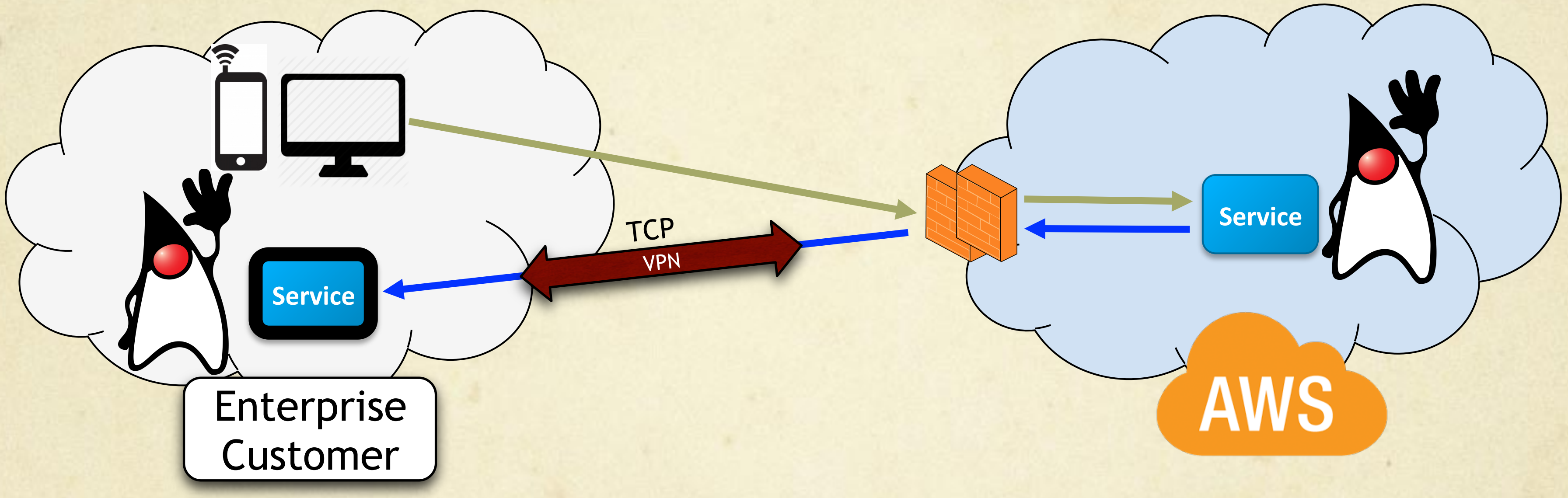
Clouds and Microservices – the bottom line

- The *Microservices Synchronicity Penalty*
- Many ecommerce sites use 150-200 microservices for personalization. Amazon.com
- Many are REST-based… ie, synchronous (wait for a reply). And many are chained, so the penalty is additive.
- Significant resources are needed for high levels of scalability (S = G/C)

Cloud Connectivity

WebSocket for Hybrid Cloud Connectivity

Cloud services frequently require on-premises access



- Access must be on-demand, secure and real-time
- Requires lengthy VPN installation process, open ports or worse

Demos

WIN A COPY!

KAAZING

WIN A COPY!

1. Introduction to HTML5 WebSocket
2. The WebSocket API
3. The WebSocket Protocol
4. Building Instant Messaging and Chat over WebSocket with XMPP
5. Using Messaging over WebSocket with STOMP
6. VNC with the Remote Frame Buffer Protocol
7. WebSocket Security
8. Deployment Considerations

Stretching Web Communication to Its Limits

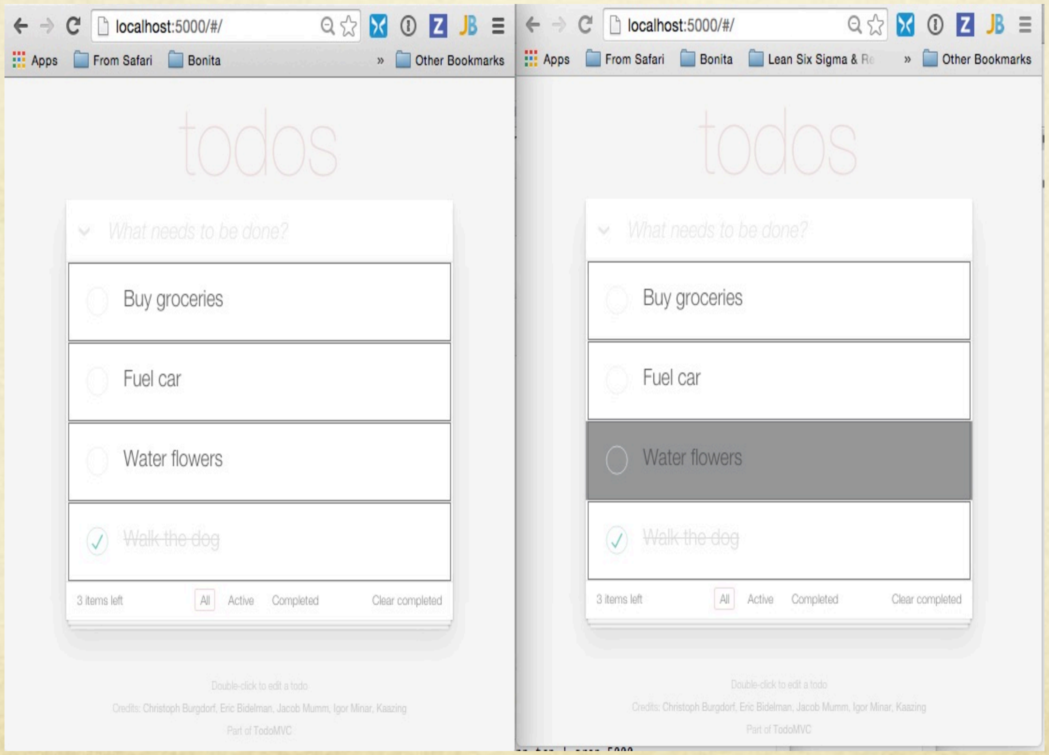
Stretching Web Communication to Its Limits

TodoMVC – Angular with WebSocket

TodoMVC

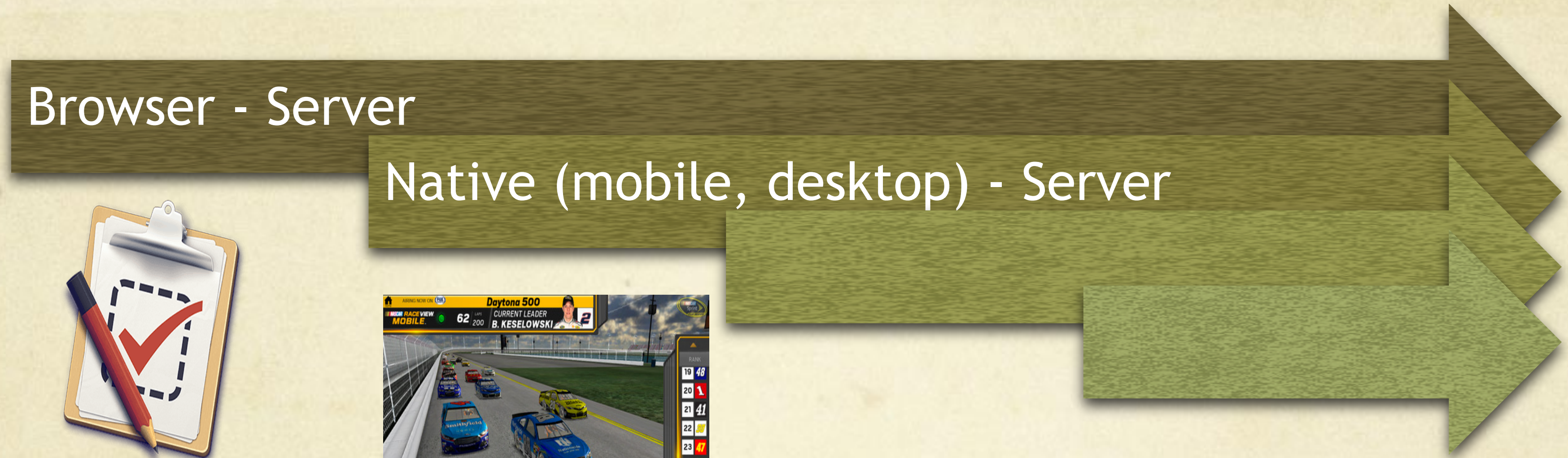
Helping you **select** an MV* framework

Download | View on GitHub | Blog



KAAZING

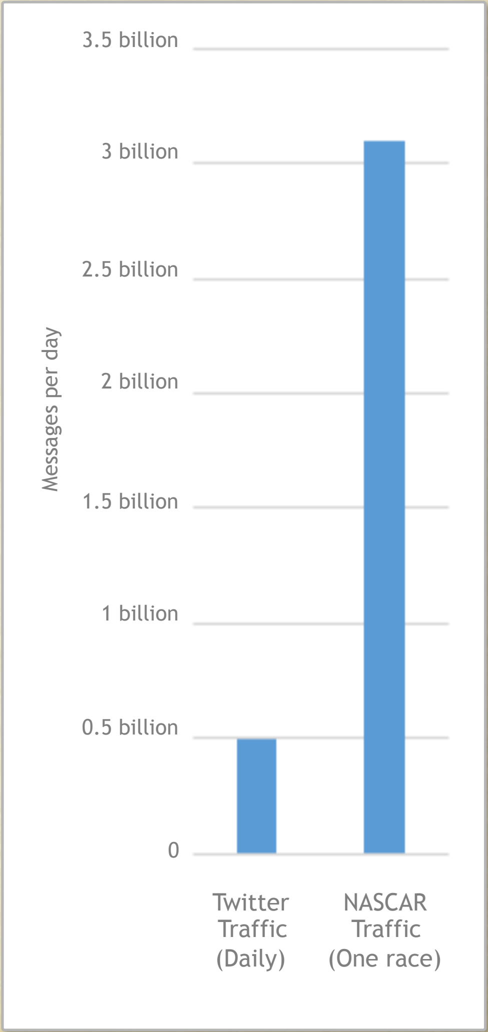
Stretching Web Communication to Its Limits



KAAZING

Over a Billion Messages an Hour

In a single 3½ hour race, [racing company] broadcasts over 6 times as many messages as Twitter does in an entire day



KAAZING

Stretching Web Communication to Its Limits

Browser - Server

Native (mobile, desktop) - Server

IoT/Embedded - Server

Stretching Web Communication to Its Limits

Browser - Server

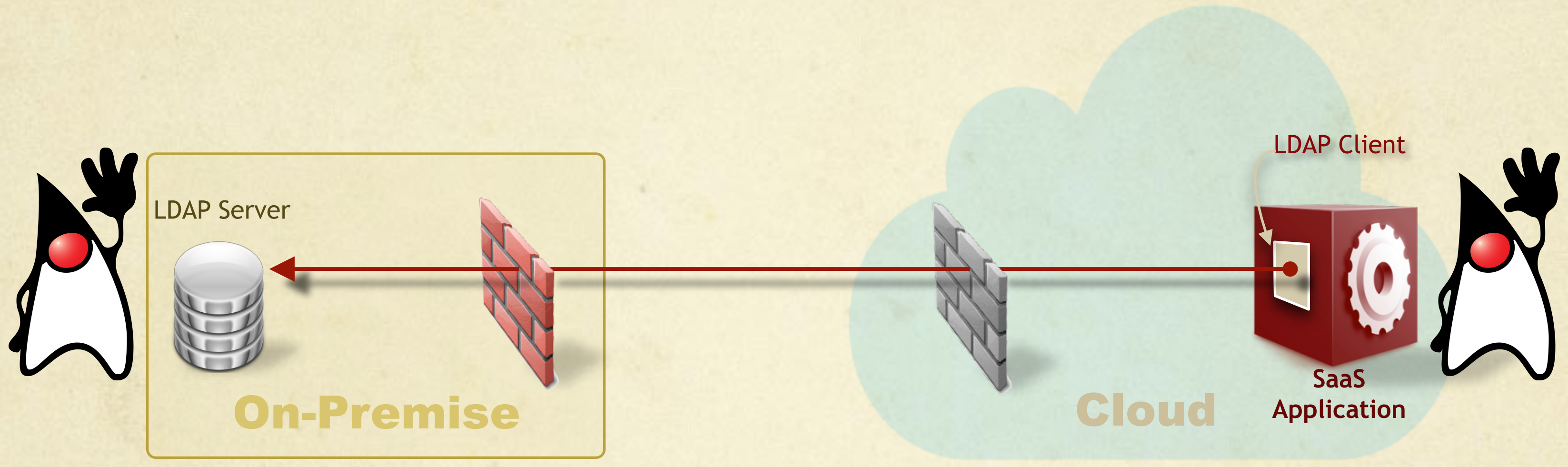
Native (mobile, desktop) - Server

IoT/Embedded - Server

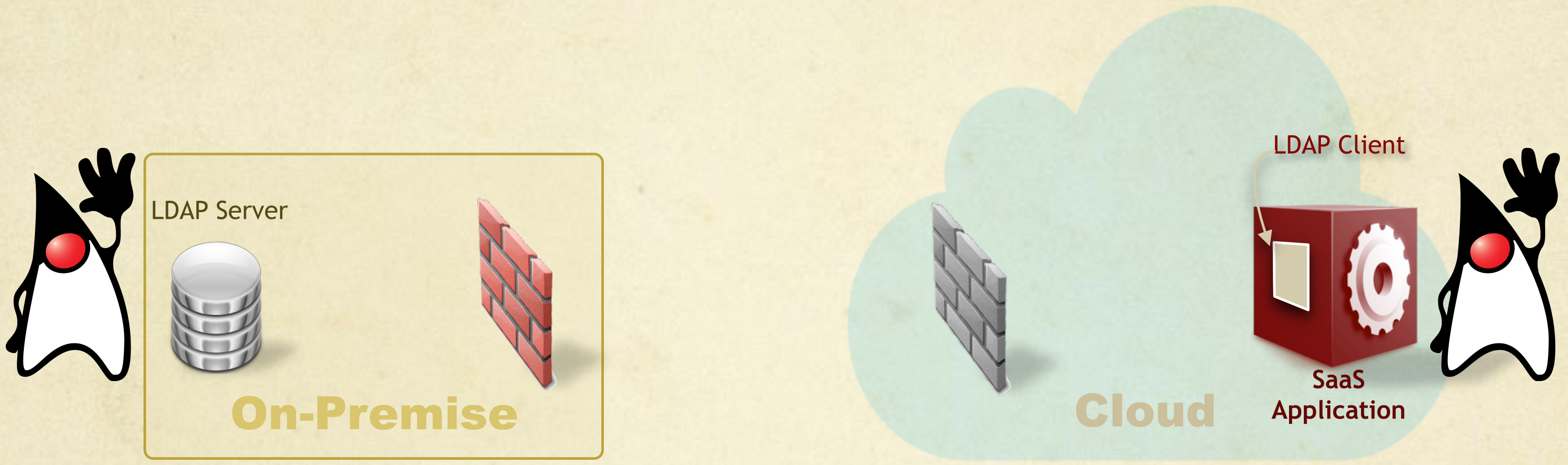
Server - Server

KAAZING

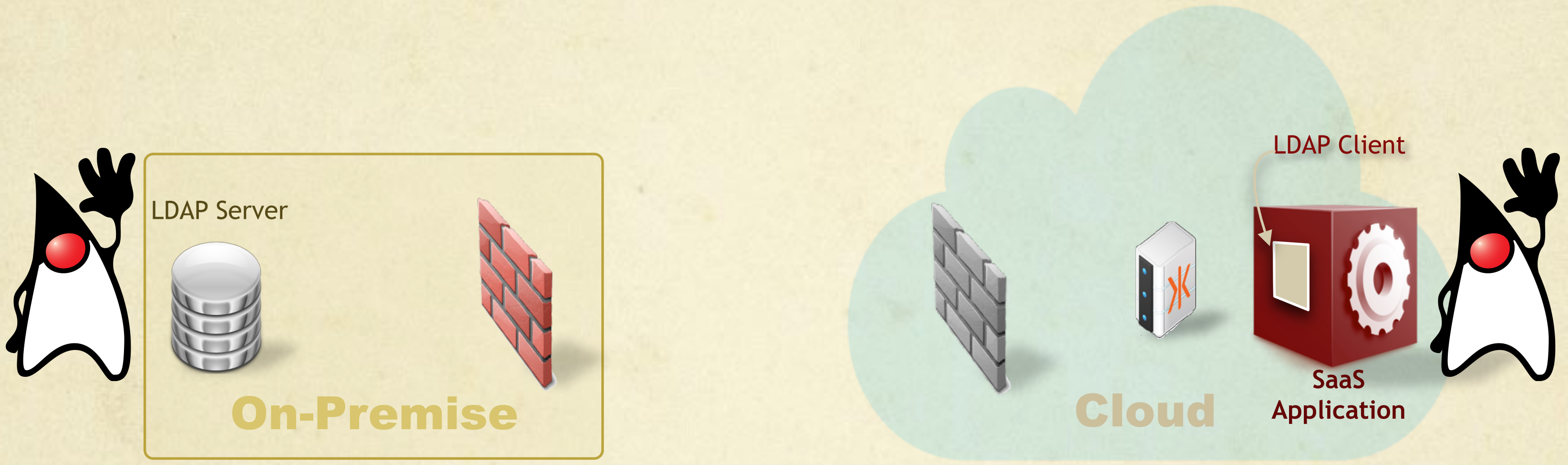
A2A for B2B. No VPN Needed.



A2A for B2B. No VPN Needed.

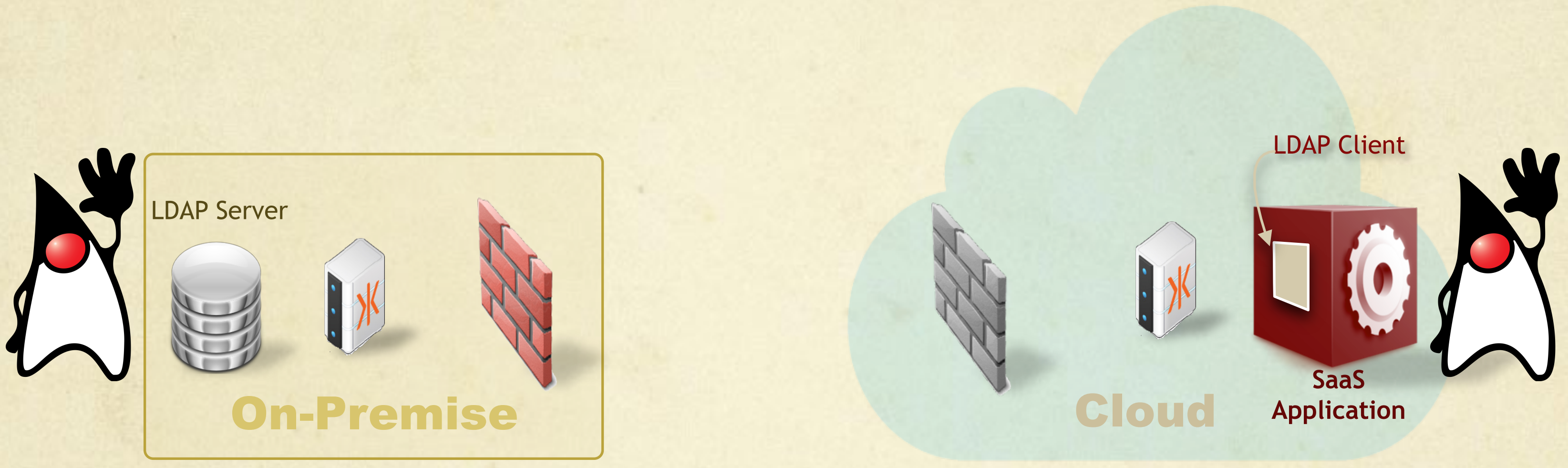


A2A for B2B. No VPN Needed.



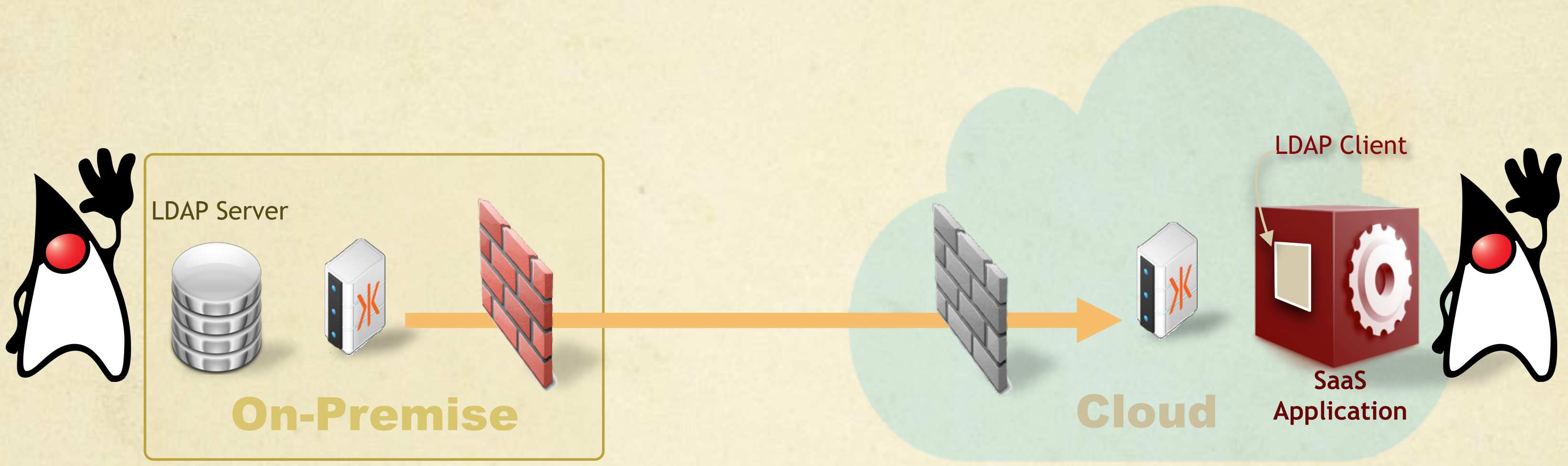
KAAZING

A2A for B2B. No VPN Needed.



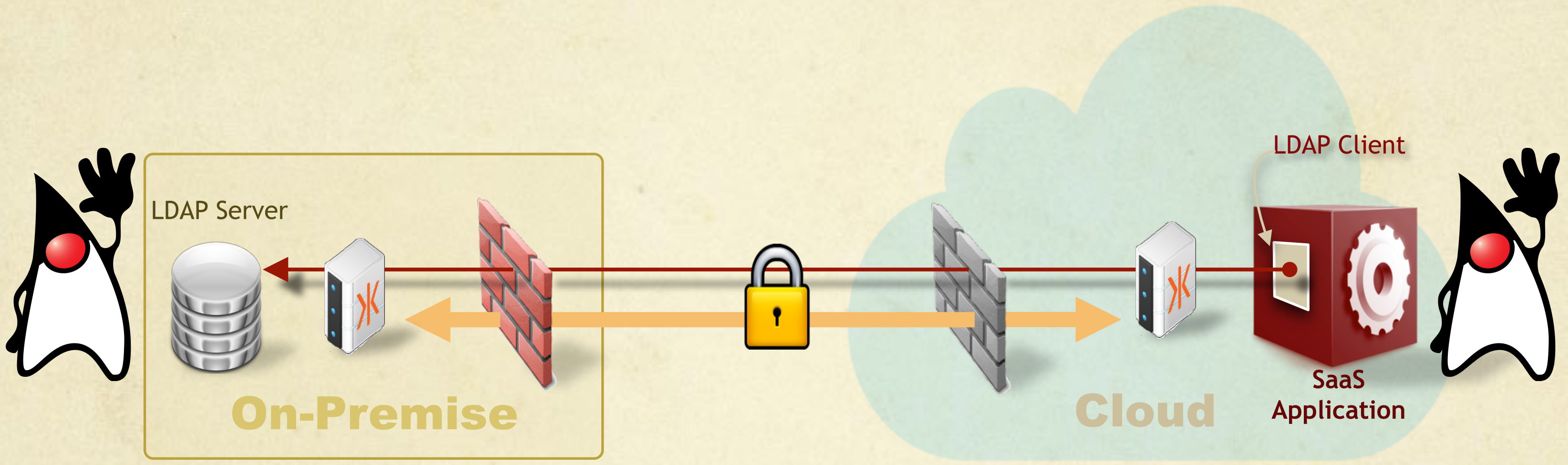
KAAZING

A2A for B2B. No VPN Needed.



KAAZING

A2A for B2B. No VPN Needed.



KAAZING

WIN A COPY!

1. Introduction to HTML5 WebSocket
2. The WebSocket API
3. The WebSocket Protocol
4. Building Instant Messaging and Chat over WebSocket with XMPP
5. Using Messaging over WebSocket with STOMP
6. VNC with the Remote Frame Buffer Protocol
7. WebSocket Security
8. Deployment Considerations

KAAZING

Raffle time

```
% echo $(( $RANDOM % 50 + 1 ))
45
```

Thank You!

@frankgreco