



Buzzwords: Microservices, containers and serverless


PRESENTED BY: Dave Nugent - Developer Advocate




Why Serverless

readwrite DEVICES DEALS TRANSPORT DATA/SECURITY INDUSTRIAL CITIES PLATFORMS MORE ▾ 🔍

Why The Future Of Software And Apps Is Serverless

Posted on October 15, 2012

 **KEN FROMM**
Contributing Writer

117 Shares   

About Me

Dave Nugent | Dev Advocacy

- ◆ CMU alum
- ◆ Astrobiology at NASA
- ◆ Consultant to PayPal, Kaiser Permanente, Deutsche Börse Group, SETI Institute
- ◆ SF JavaScript, SF IoT meetups; ForwardJS, Forward Swift
- ◆ Joined Iron.io March 2016

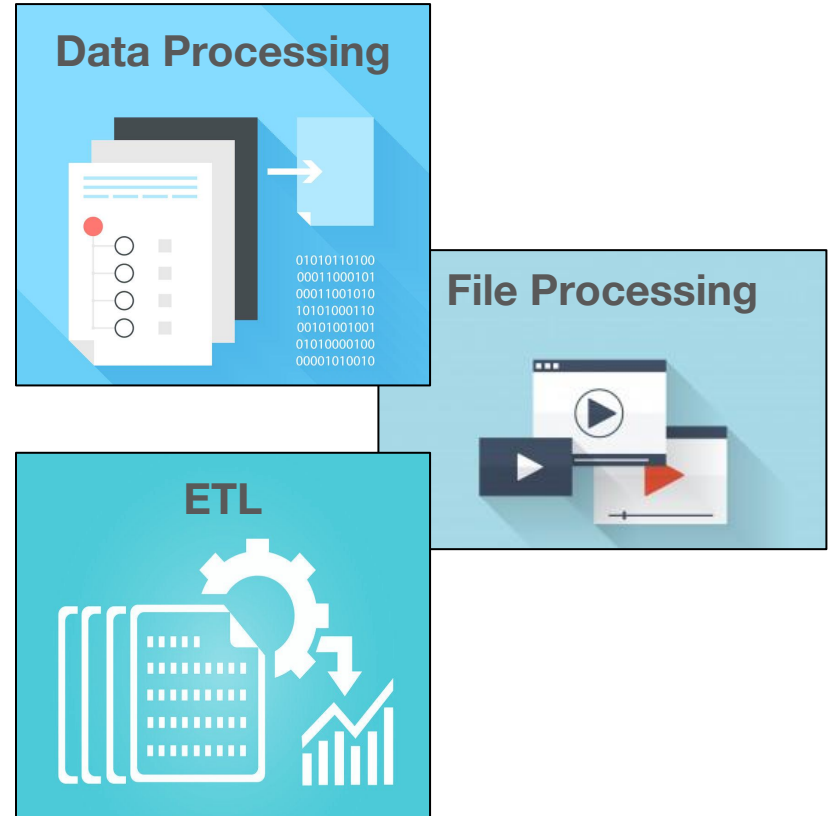


@drnugent

What We Do

Iron.io delivers Docker-based
job processing as a service
for modern enterprises

70% of IT processes still performed in batch - Gartner



The Evolution of Deployed Application



Unit of Scale

Server

VM

Container

Application Architecture

Monolith

N-Tiered

Microservices

Deployment Model

Major Release

Software Updates

Continuous Delivery

Workload Processing

DIY

Software Defined

Event-Driven

We are leading the Enterprise towards a “serverless” computing world

Impact on Organizations

Technology

Smaller single purpose services

Independently developed and deployed components

Lightweight cloud-native communication and processing

Standardizing on Containers



Business

Agile Teams

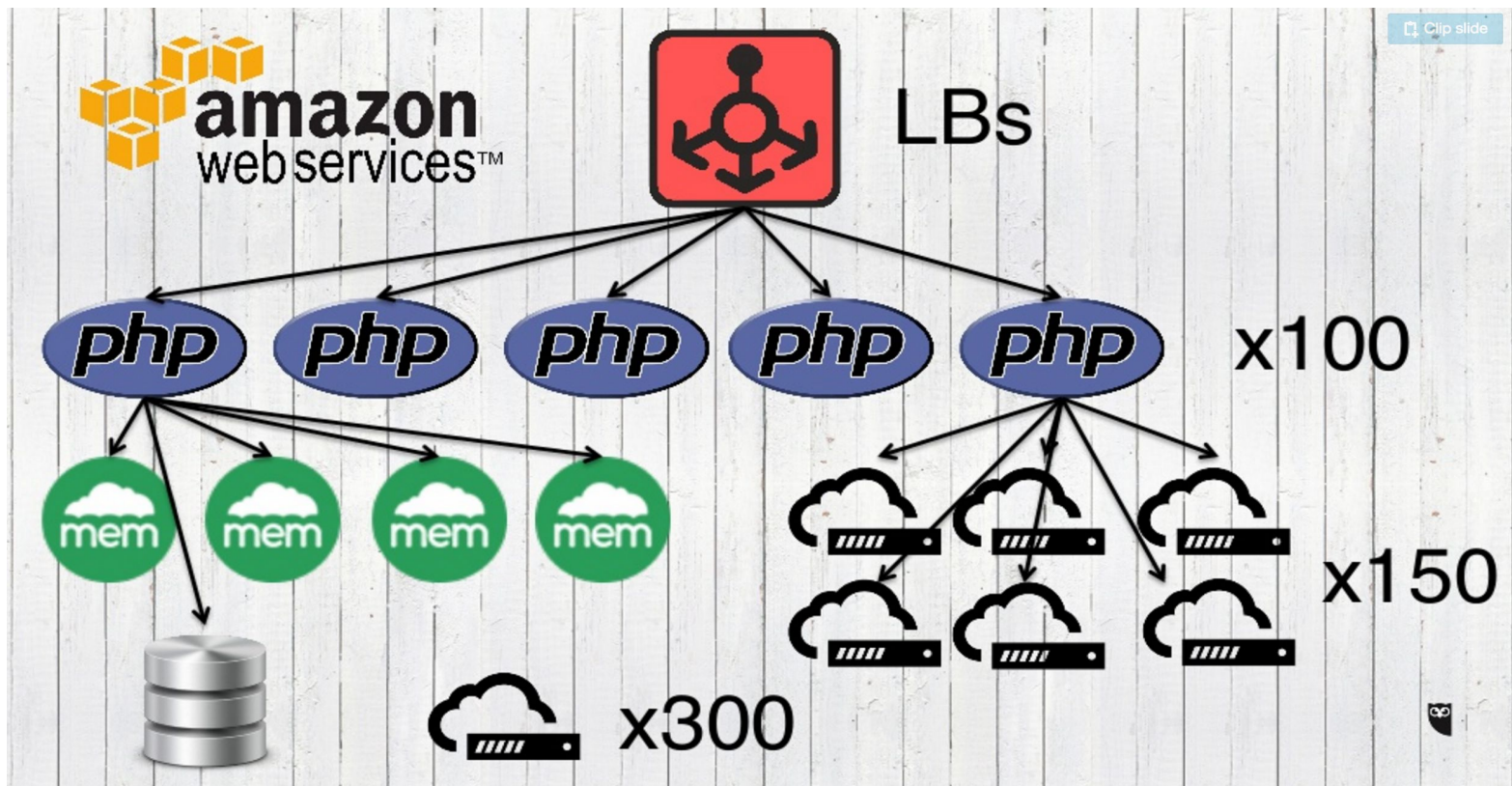
Shorter release cycles

Cost efficient scaling

No vendor/tech lock in

First: Let's look at the Most Recent Paradigm

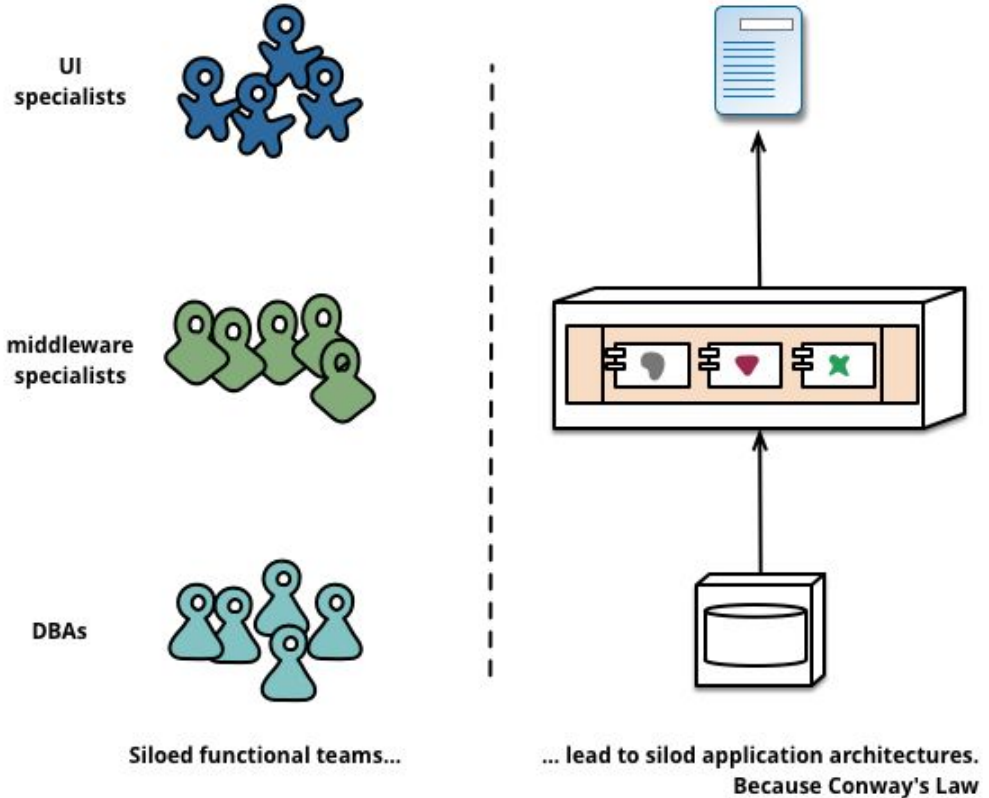
Why are we still doing this?



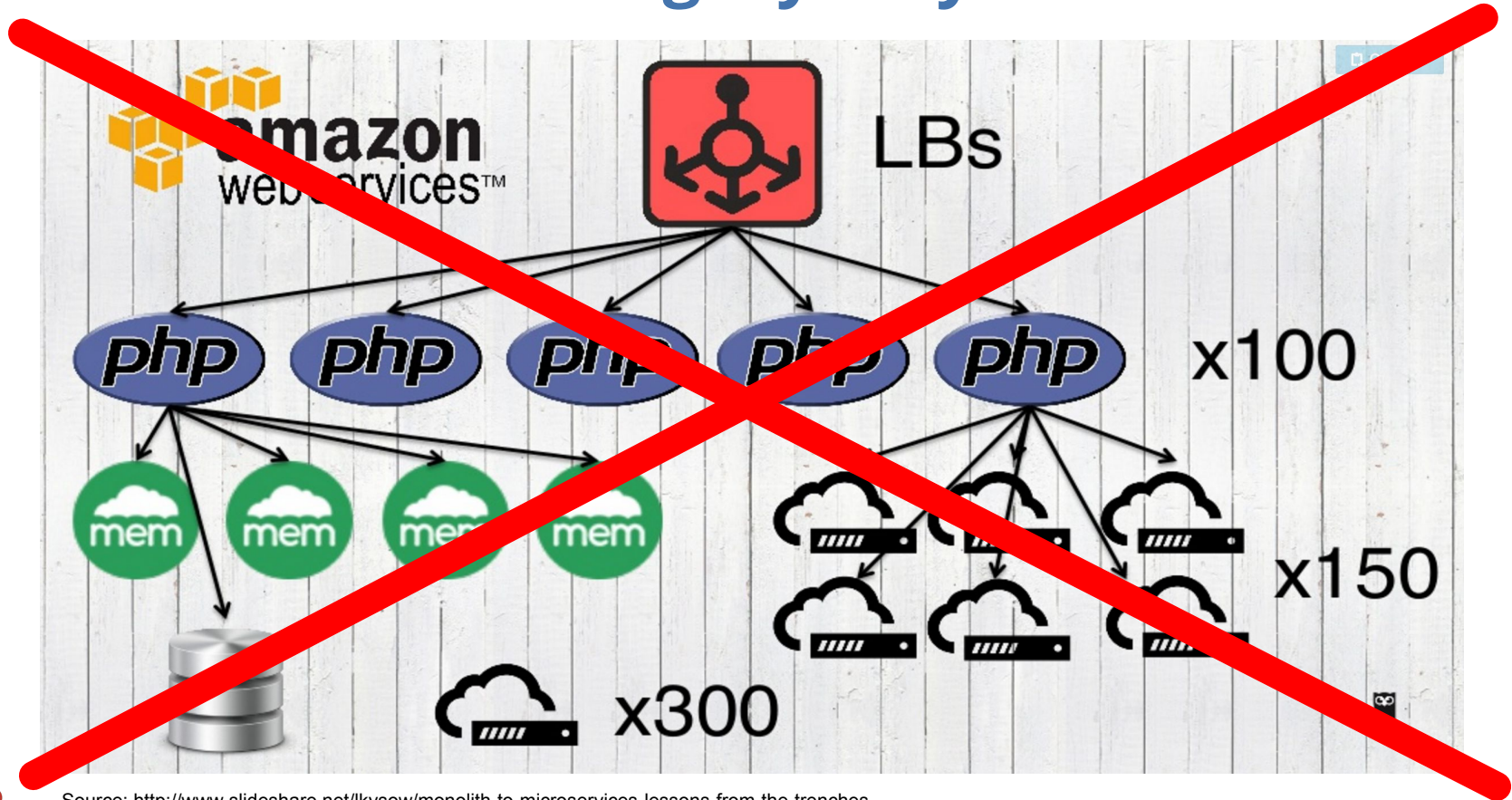
One Hypothesis: Cultural Constructs

"Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure."

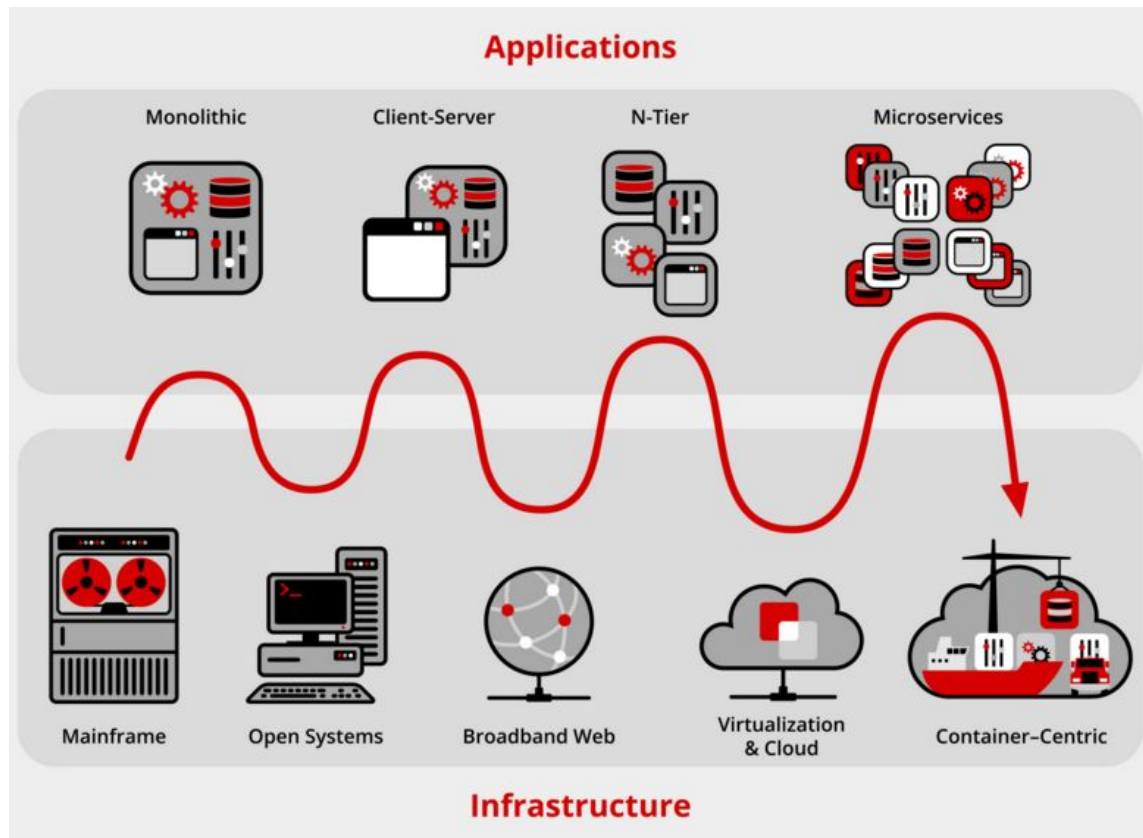
-- Melvyn Conway, 1967



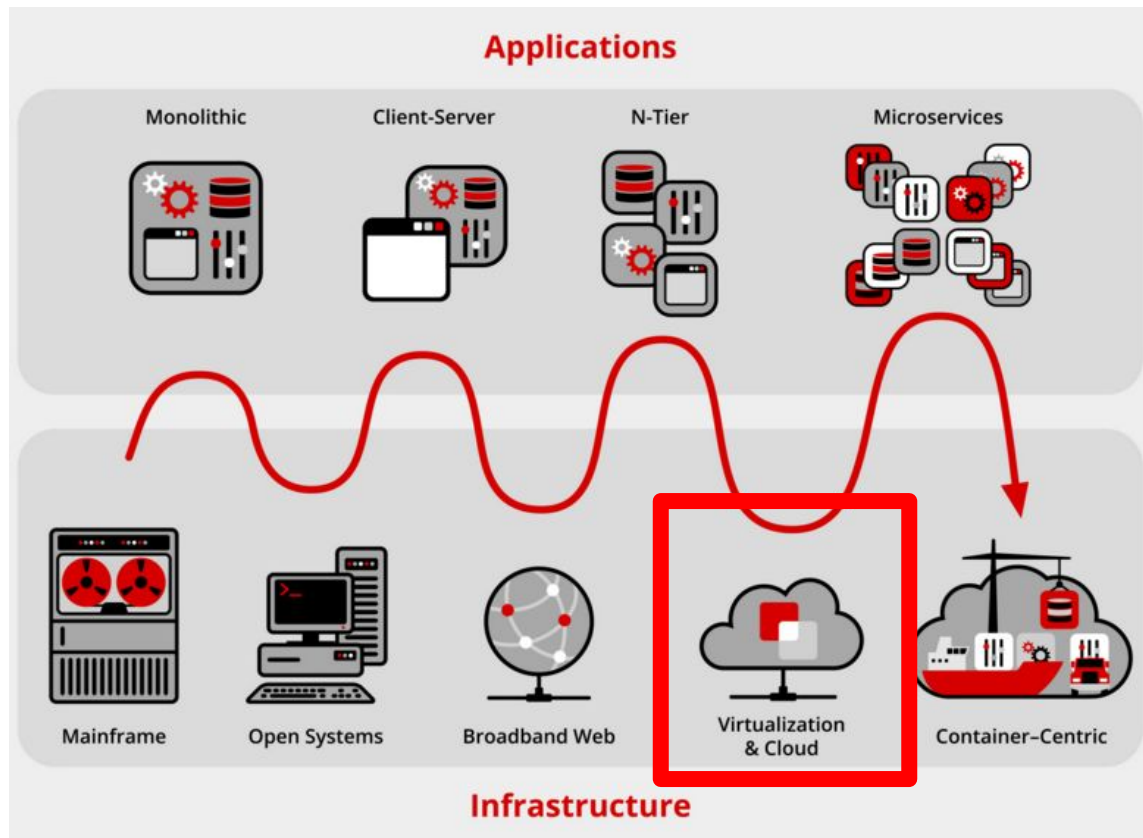
Point of Interest: Legacy Players Flexible



Application & Platform Evolution



Application & Platform Evolution

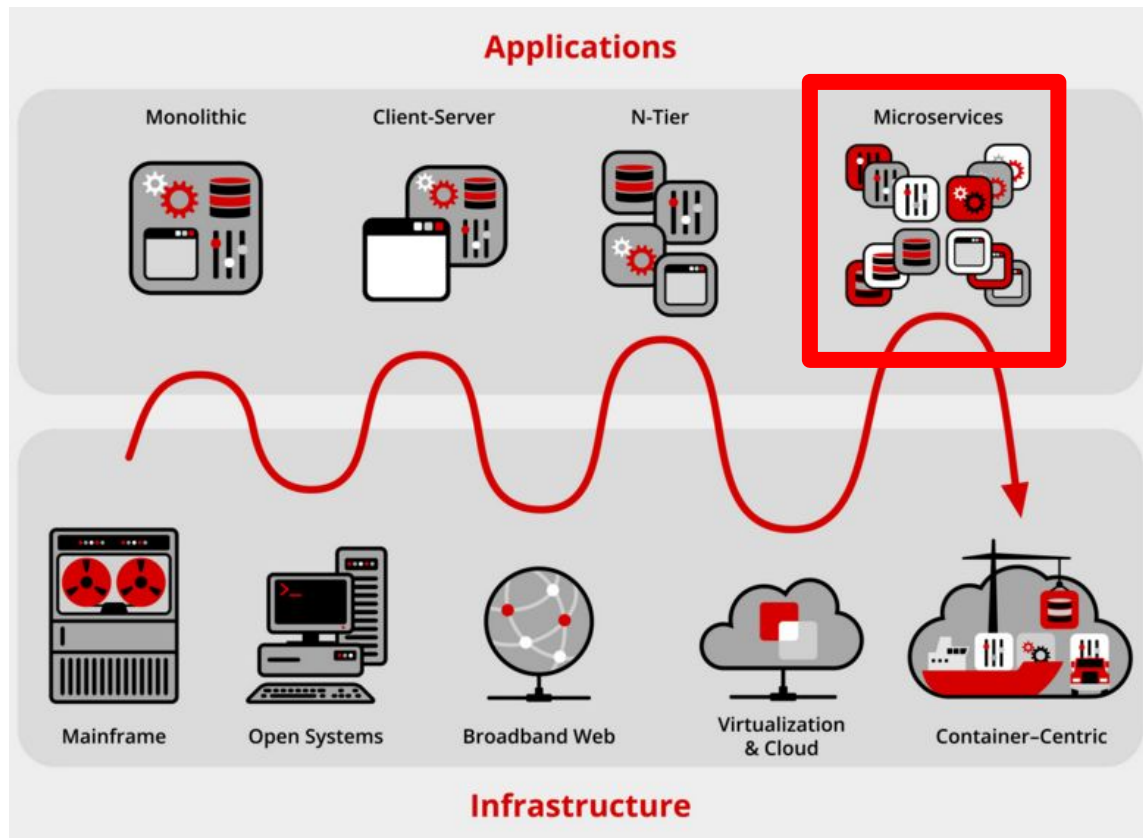


Virtualization & Cloud

- Decoupled hardware and software
- Software-driven hardware

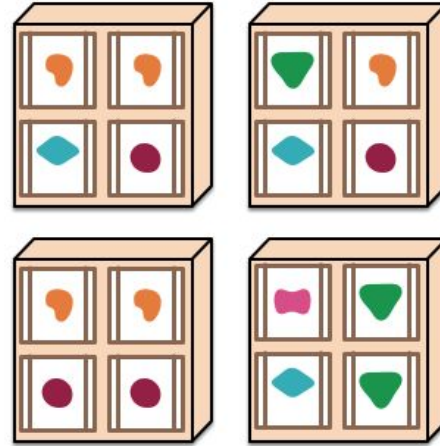


Application & Platform Evolution

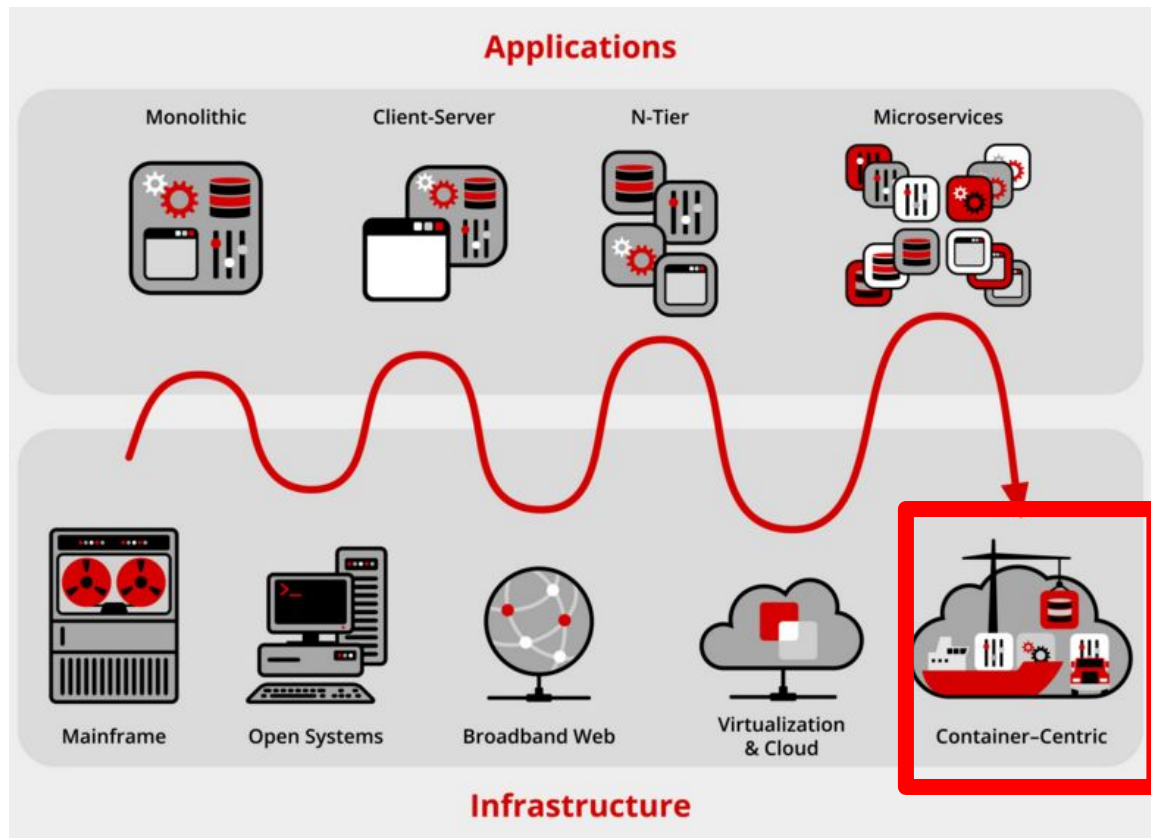


Microservices

- Trust and policy between distributed services
- Horizontally scalable applications built for cloud



Application & Platform Evolution

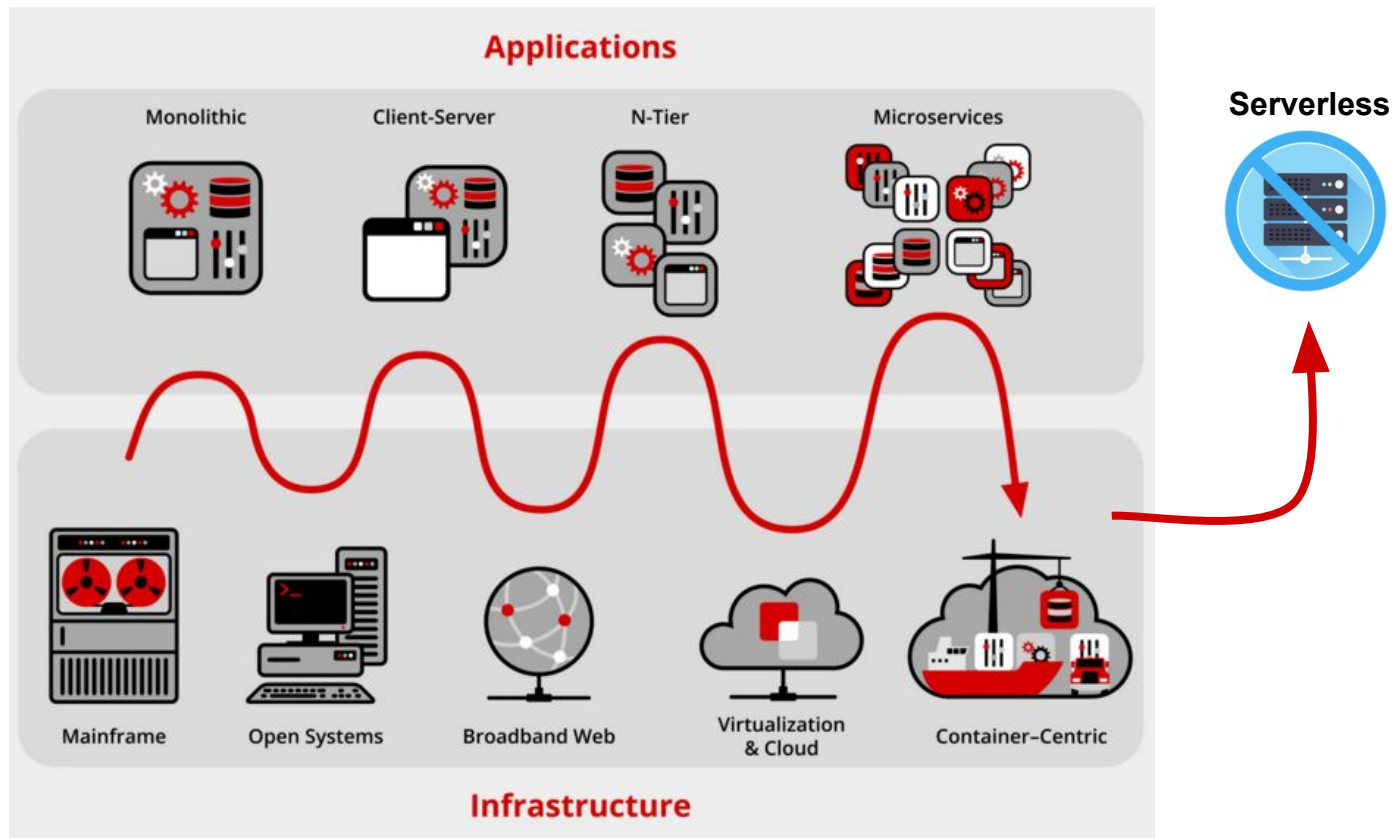


Containers

- Allow extremely higher efficient sharing of resources
- Provides standard and minimizes software packaging
- Further decouples software from underlying host w/ no hypervisor



Application & Platform Evolution



Workloads: Legacy vs Serverless



Legacy App

Pushed

Running

Requested

Load Balanced

Elastic



Serverless Job

Uploaded

Ephemeral

Triggered

Queued

Concurrent

Job-centric workloads have a different behavior than app-centric workloads

Event Driven

- Evented invocation of function/worker/task
- Producer to consumer one-way invocation
- Fire and forget model

Containerized

- Skip the virtual machine
- Microcontainers reduce network traffic
 - Alpine Linux, CoreOS, etc.
- Code becomes ultra-portable abstracting server and VM
- iron.io/microcontainers-tiny-portable-containers/

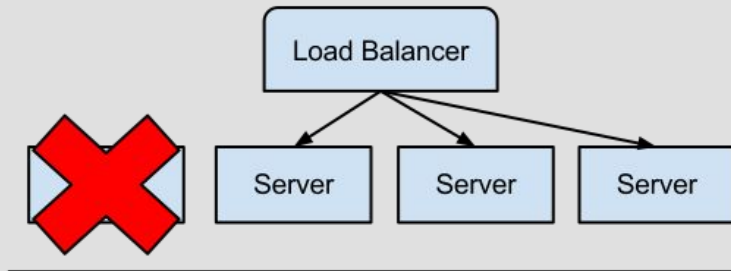
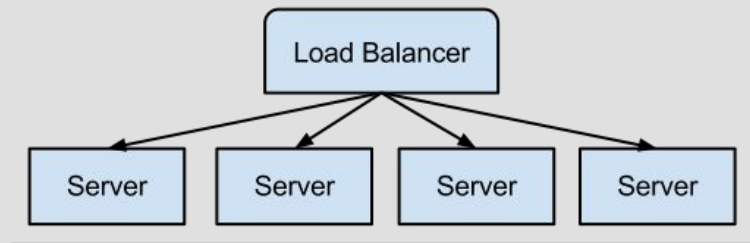
Composable

- Serverless roots grounded in SOA
- 3rd party providers bring scalable component code
- Examples: Algolia, Algorithmia, Cloudinary, Auth0, DynamoDB, on and on and on.

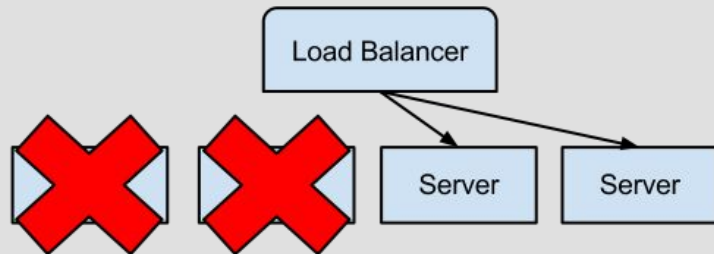
Workload Aware

- Understanding of the workload characteristics and properties
- Allows for self-healing and directing of workloads across specialized infrastructure

Colossal Clusterf**k Visualized



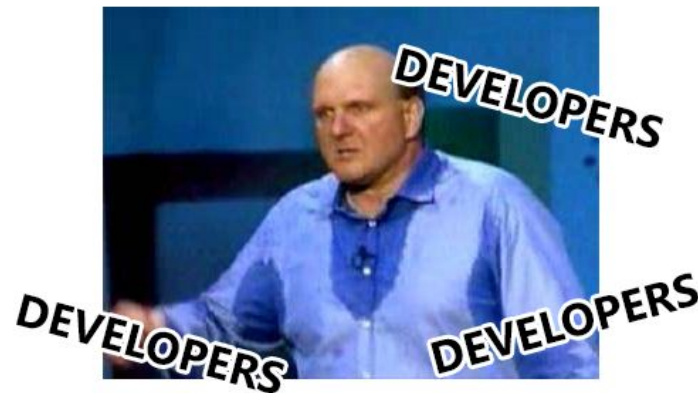
One server goes down which puts more load on the remaining servers.



Another goes down which puts more load on the remaining servers.

Developer Empowerment

- Moves the abstraction level up
- Spend more time on feature code
- Implement and extend 3rd party code



Organizational Impact

Technology

Smaller single purpose services

Independently developed and deployed functional components

Lightweight cloud-native communication and processing

Standardizing on Containers



Business

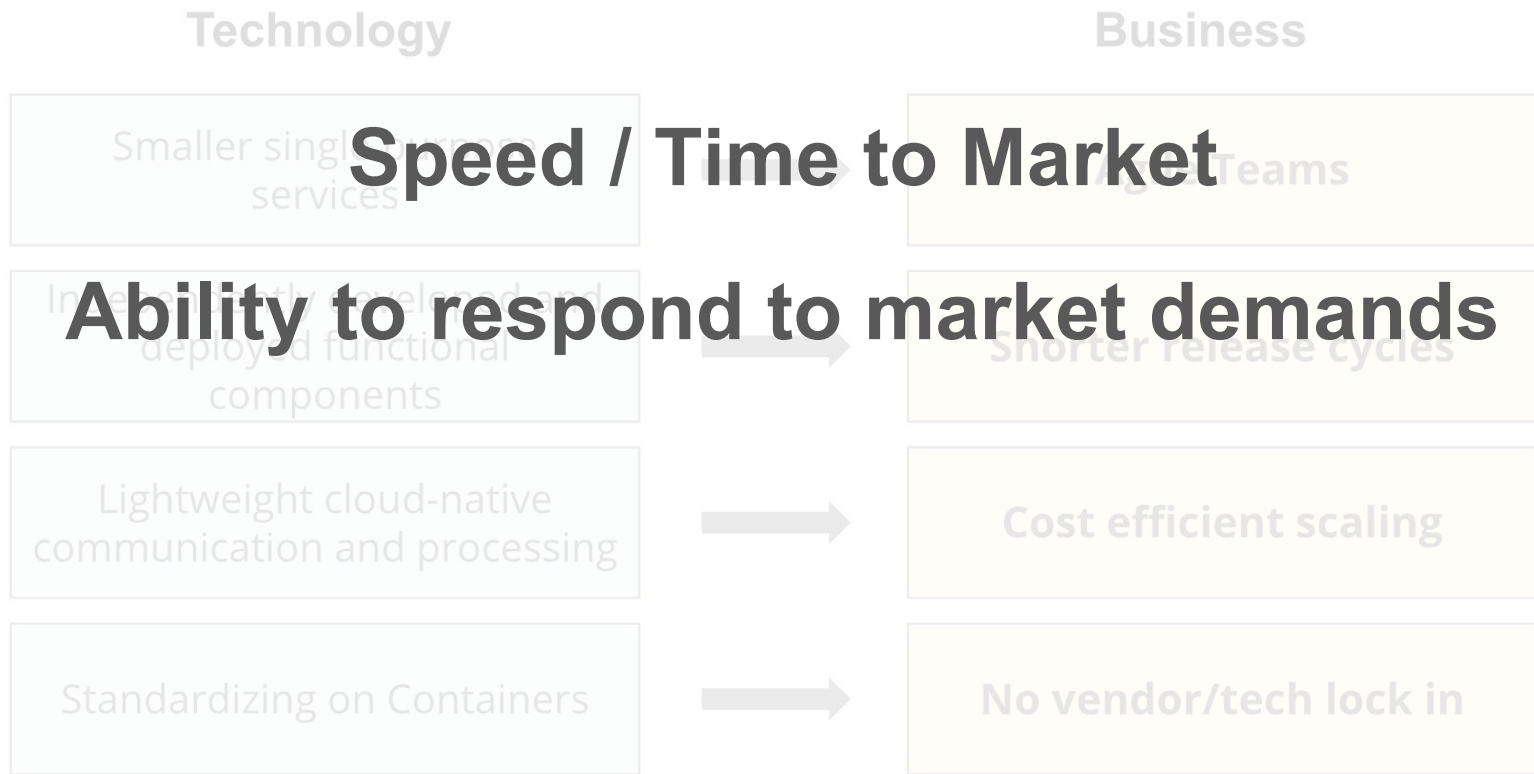
Agile Teams

Shorter release cycles

Cost efficient scaling

No vendor/tech lock in

Organizational Impact



Serverless Platforms

AWS Lambda

- Pros
 - Native integration with all other AWS services
 - Scales nicely
 - Cheap
- Cons
 - Native integration with all other AWS services
 - Stuck with their machine images
 - IAM

Google Cloud Functions

- Pros
 - Integration to strong machine learning tools
 - Generally better performance per dollar
- Cons
 - TBD - We don't know yet

Microsoft Azure Functions

- Pros
 - Private cloud support
 - Quickly innovating service feature set
- Cons
 - Azure

Iron.io

- Pros
 - Supports all public/private clouds
 - Docker-based w/ rich API
- Cons
 - More work to build triggers

FAQs

- Isn't serverless computing impossible?

FAQs

- Aren't the servers just managed by someone else?

FAQs

- Aren't the servers just managed by someone else?

FAQs

- If these aren't just buzzwords, how do I use them?

Just Published: Serverless White Paper

Executive Summary

Imagine a future where the notion of provisioning and managing infrastructure is completely removed from the development process - in production and at any scale. The 'serverless experience' abstracts away the complexities from the developer experience so they don't have to think about the resources associated with powering their workloads.

Once you've succeeded in implementing a serverless experience for the developer, you allow them to focus on a clear path to build features more quickly. In addition, going 'serverless' allows you to break up resources and make the most out of what you have - and what you're spending. A fully automated workflow leaves less need for oversight and leaves you investing in workload optimization and intelligent systems design upfront - setting the future for future growth.



<http://go.iron.io/serverless-computing-white-paper>

The Future is of Serverless



Questions?



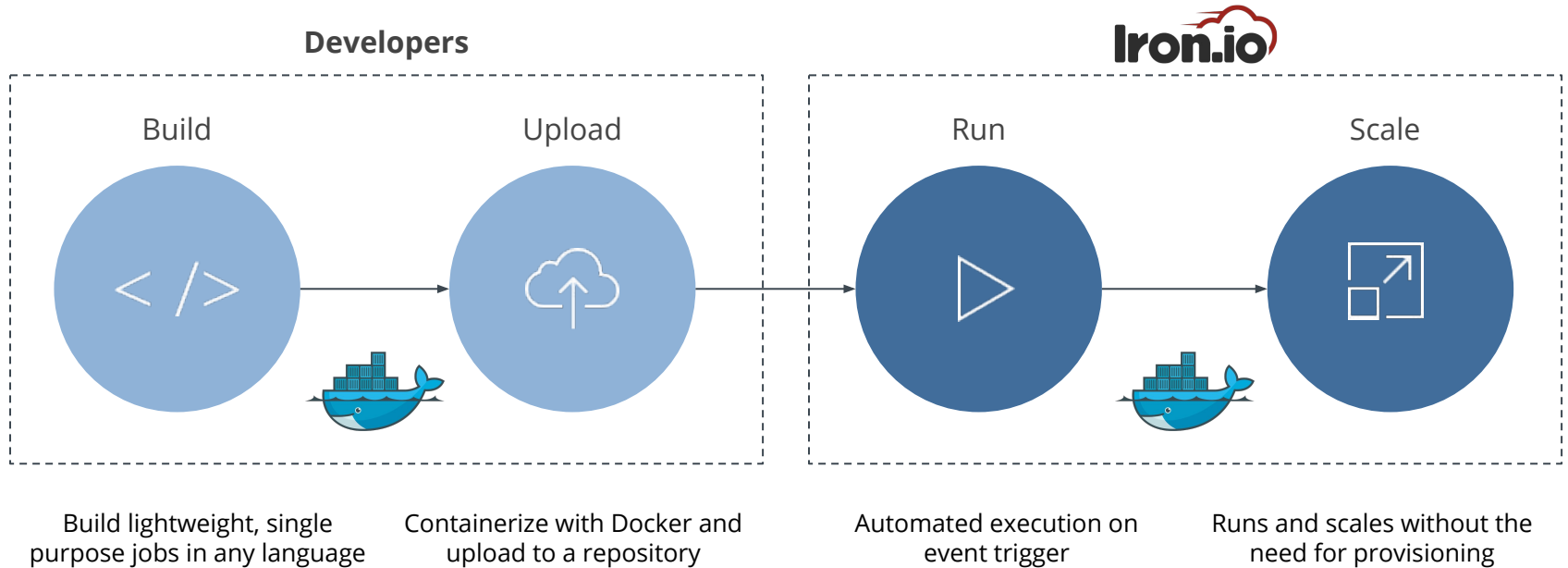
@drnugent

<http://go.iron.io/serverless-computing-white-paper>

Iron.io
325 9th St
San Francisco, CA 94103

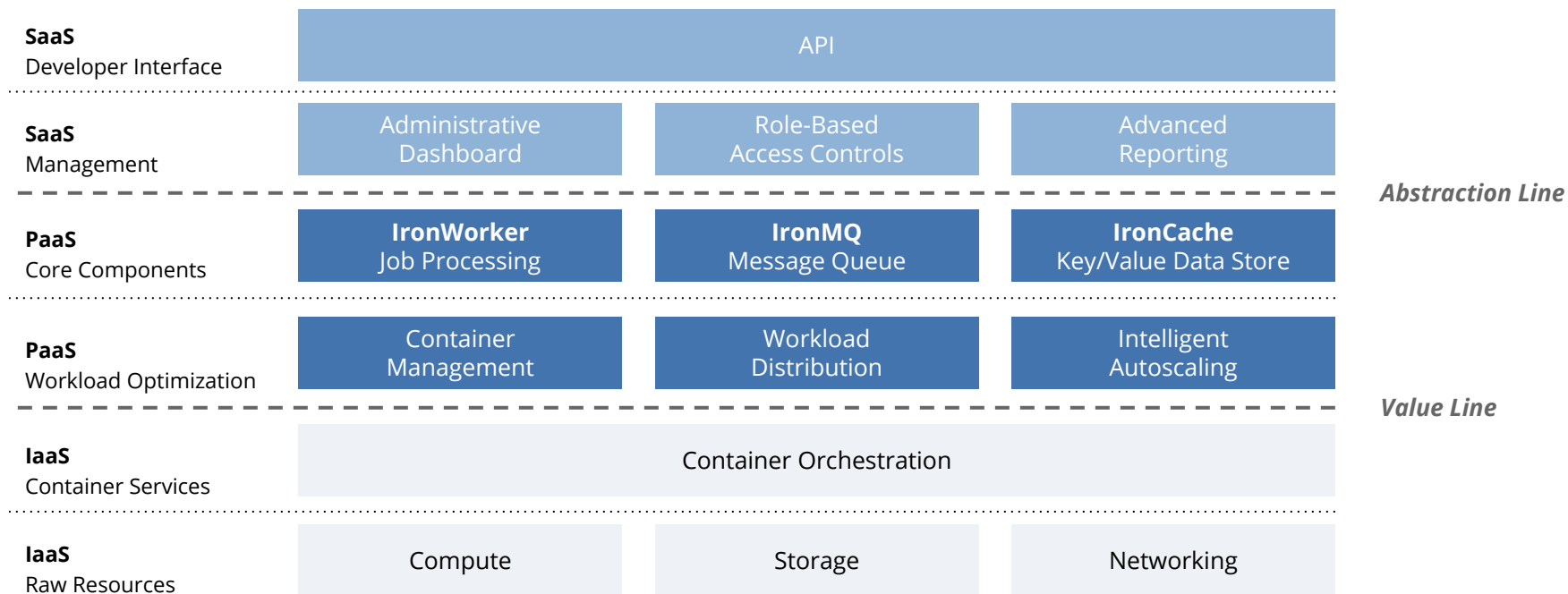
1-888-939-4623
www.iron.io
dave@iron.io

Docker-Based Workflow



To the developer, working with Iron.io is a “serverless” experience

The Iron.io Platform



Key Features



Queuing Jobs

Once your code is uploaded, you can queue up jobs and Iron.io handles the provisioning and execution.



Scheduling Jobs

Scheduling API replaces CRON with an HA service that withstands node failures.



Job Priorities

Includes a built-in priority manager, allowing users to set the importance of specific jobs to be run.



Webhooks

Create event-driven workflows between APIs, services, and endpoints through an HTTP POST callback.



Logging

STDOUT captured for every job and exposed via API and dashboard, and can stream to syslog or 3rd party



Failure Handling

Job state change provides error and timeout handling, with alerting and auto-retry capabilities.

Why Businesses Choose Iron.io

“Serverless” Experience

Power large-scale workloads without the need to provision and manage infrastructure.

Developer Friendly

Cloud-native REST API with client libraries across all major languages.

Workload Scalability

Scale effectively and efficiently at the task level through lightweight and loosely coupled containers.

Multi-cloud Portability

Container-based to allow for flexible and portable workloads that can be run on any cloud of choice.

Speed to Market

Operates as a service and can be easily integrated with various platforms and services.

Hybrid Capable

Deploy components and distribute workloads to any cloud environment, public or private.

Popular Use Cases

THIS SLIDE IS **NOT** FOR PUBLIC CONSUMPTION

