DevOps 2.0

Using Modern Tools and Practices to Develop, Maintain, and Manage Scalable Microservices

Joe McCormick, Architect
Boeing

Joseph E. McCormick III

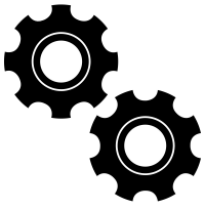
Joe McCormick has more than 25 years of software experience in companies ranging from Dot Com startups to large Fortune 100 companies, usually filling roles in Development, Architecture, and Development Management. He has extensive experience in designing, using, creating, and implementing Software Configuration Management systems, build and deployment systems, Application Lifecycle Management tools, and other software delivery pipeline enablers, making him an expert in the concepts of Continuous Integration and Delivery, Agile development processes, DevOps, and Service-Oriented Architecture concepts like Microservices.

The former Long Island, NY Firefighter, Emergency Medical Technician, and Army Crew Chief (OH-6, OH-58, UH-1, and UH-60 helicopter airframes) now lives in Charleston, SC and races sailboats in his free time (this will become evident during the following presentation).

Joe is currently working as an Architect in Boeing's Future State Technology Architecture group under the Enterprise Architecture organization of Information Technology.

Overview

Core Concepts



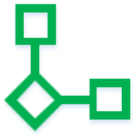
Microservices

DevOps

Blue Green Deployments

Self Healing

Lightweight APIs



Agile

Logging & Analytics

Discovery

Continuous Monitoring & Response

Proxy Services

Test Driven Development

Automated Scaling

Orchestration

Continuous Delivery

Infrastructure & Configuration as code

RESTful JSON

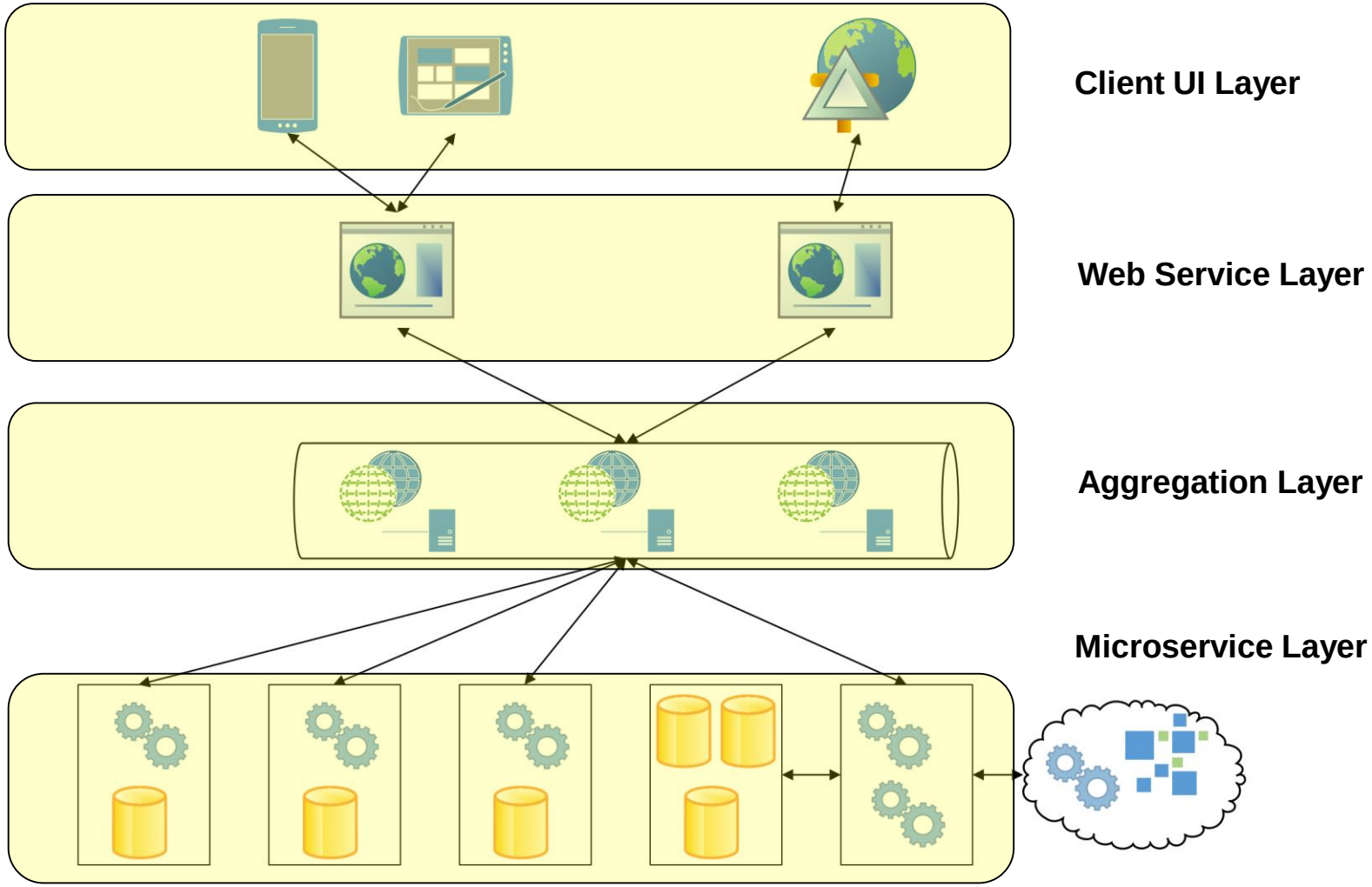
Before We Begin

Request: This presentation is an **overview** and **integration** of more than a few concepts, many of which could be **presented on their own** (or could even be their own workshops or courses). The first third of this presentation contains definitions and overviews of **high-level ideas** that will be covered quickly and the details will be covered in later slides. **Please hold your questions** until solicited. Thank You.

Note: If you are viewing this material on your own (i.e. Joe is not presenting the deck), please view this in **"presentation mode"**. As a time and slide saving measure, animations are used throughout the presentation and the slides will not render correctly (or will not be readable at all) if you are not in "presentation mode".

DISCLAIMER: The software and tools used in this presentation are for **conceptual demonstration** and **do not represent the standard tools and development patterns** of any organization or company. Please **consult YOUR organization** for standard tooling, patterns, processes, and best practices.

Our Example Application: Sailboat Management



Our Test Application: Sailboat Management

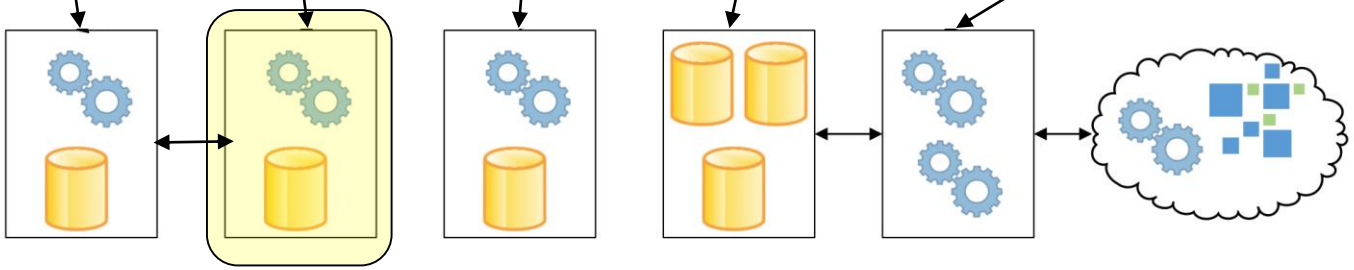
crew-assignments Microservice

User Story:
As a Boat Manager, I need to see the age of each crewmate so that the company can better plan for the Youth Sailing Program events and races.

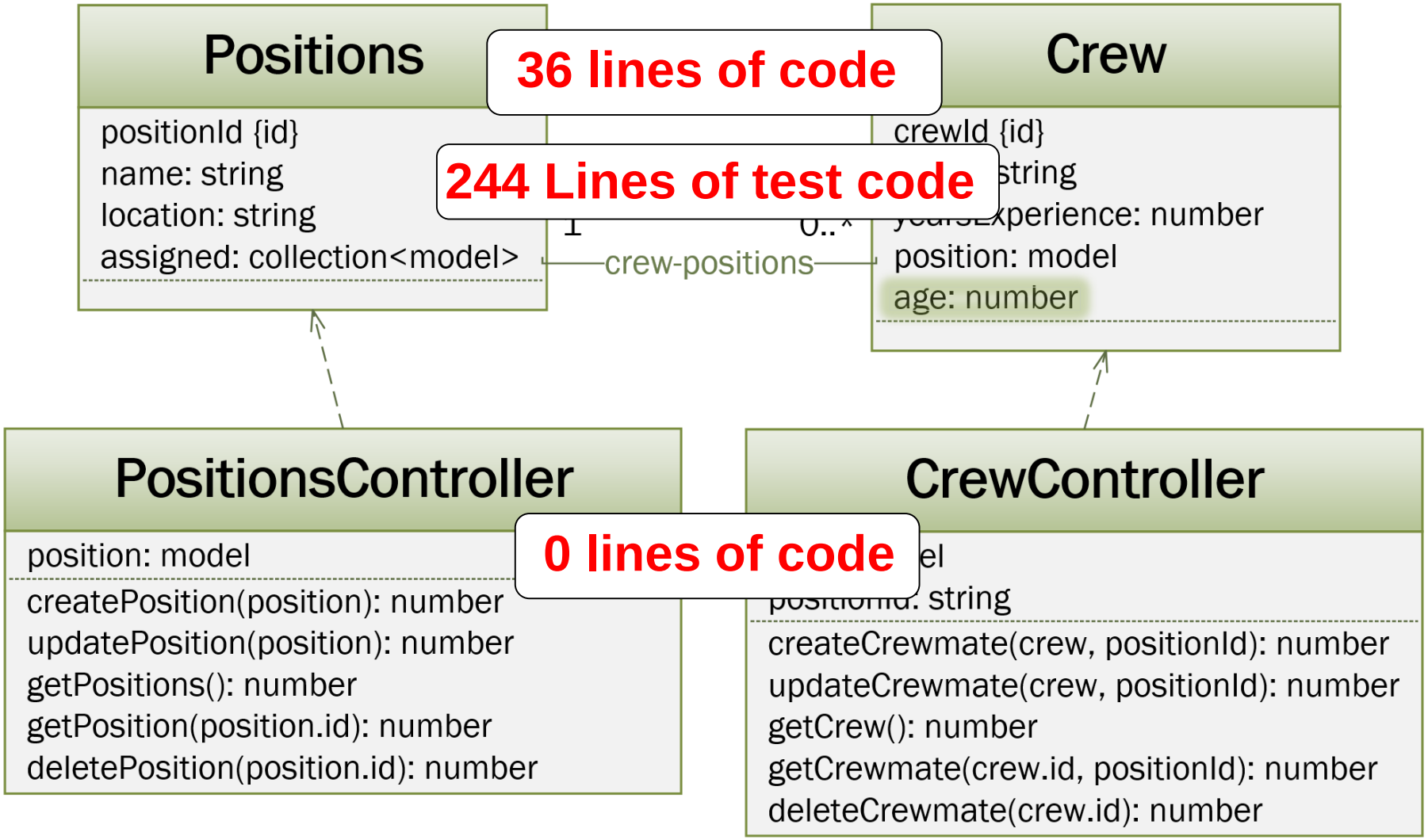
- **Task**: Add an age text field on the crew information page.
  - Status: Completed
  - Comment: Implemented, but is Feature Flagged to not show until the crew-assignments service is updated.
- **Task**: Update the data model of the crew-assignments service to accept age as a new 'float' property.
  - Status: Not Yet Started

Existing Enterprise Databases

Inventory / Supplier / Ordering Microservices



crew-assignments Service Description



crew-assignments Development Environment

(“DevOps in a Box”)

./jq

MOCHA

python

SuperTest

docker

ANSIBLE

GO

CONSUL TEMPLATE

mongoDB

node + npm

sails

git

Jenkins

Docker Compose

Java

UI For Docker

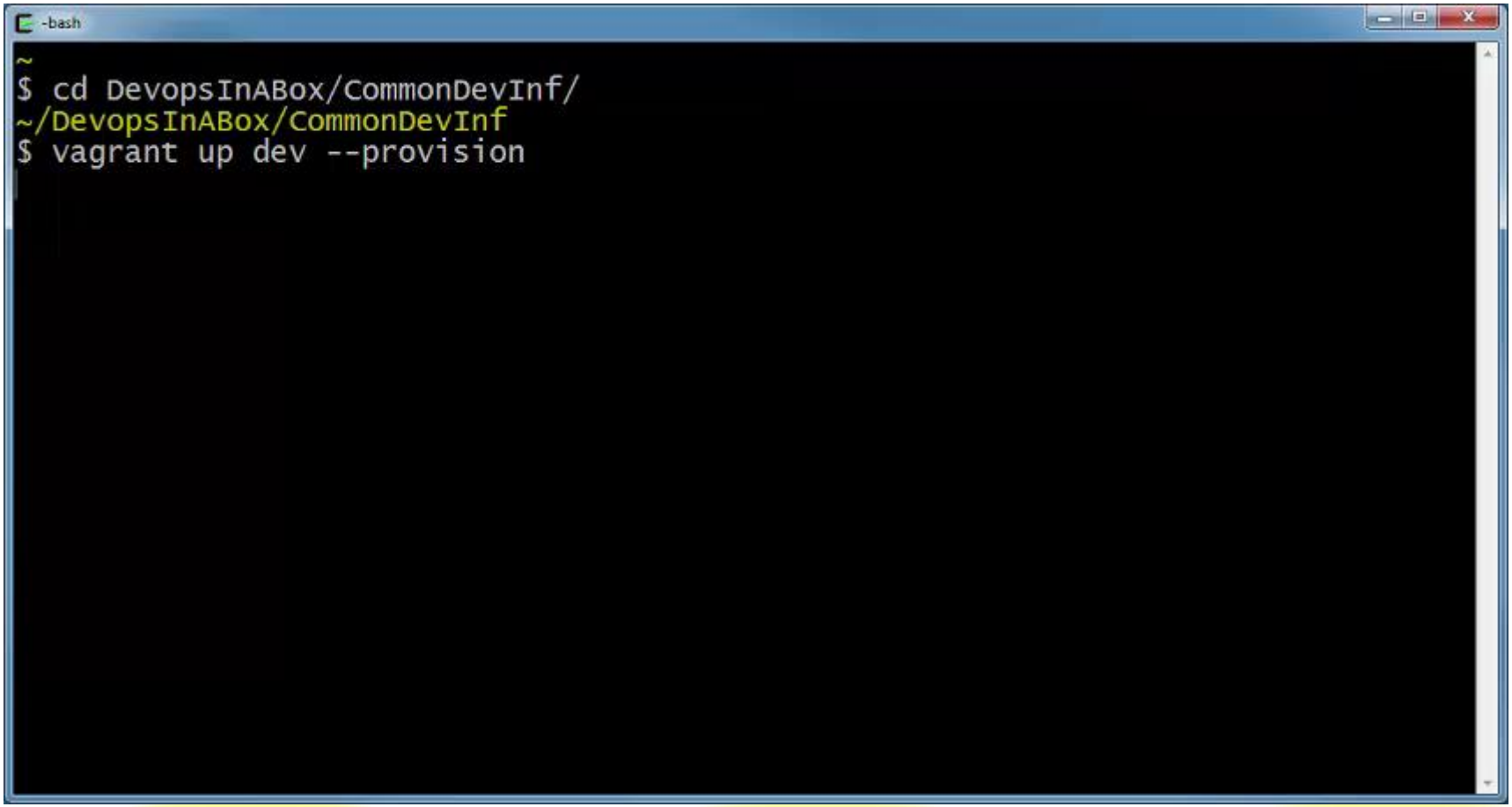
docker-py

SHOULD JS

Groovy

Workflow Aggregator
Multibranch
Log Parser
Copy Artifact
Template
Plugins

Provisioned Environment

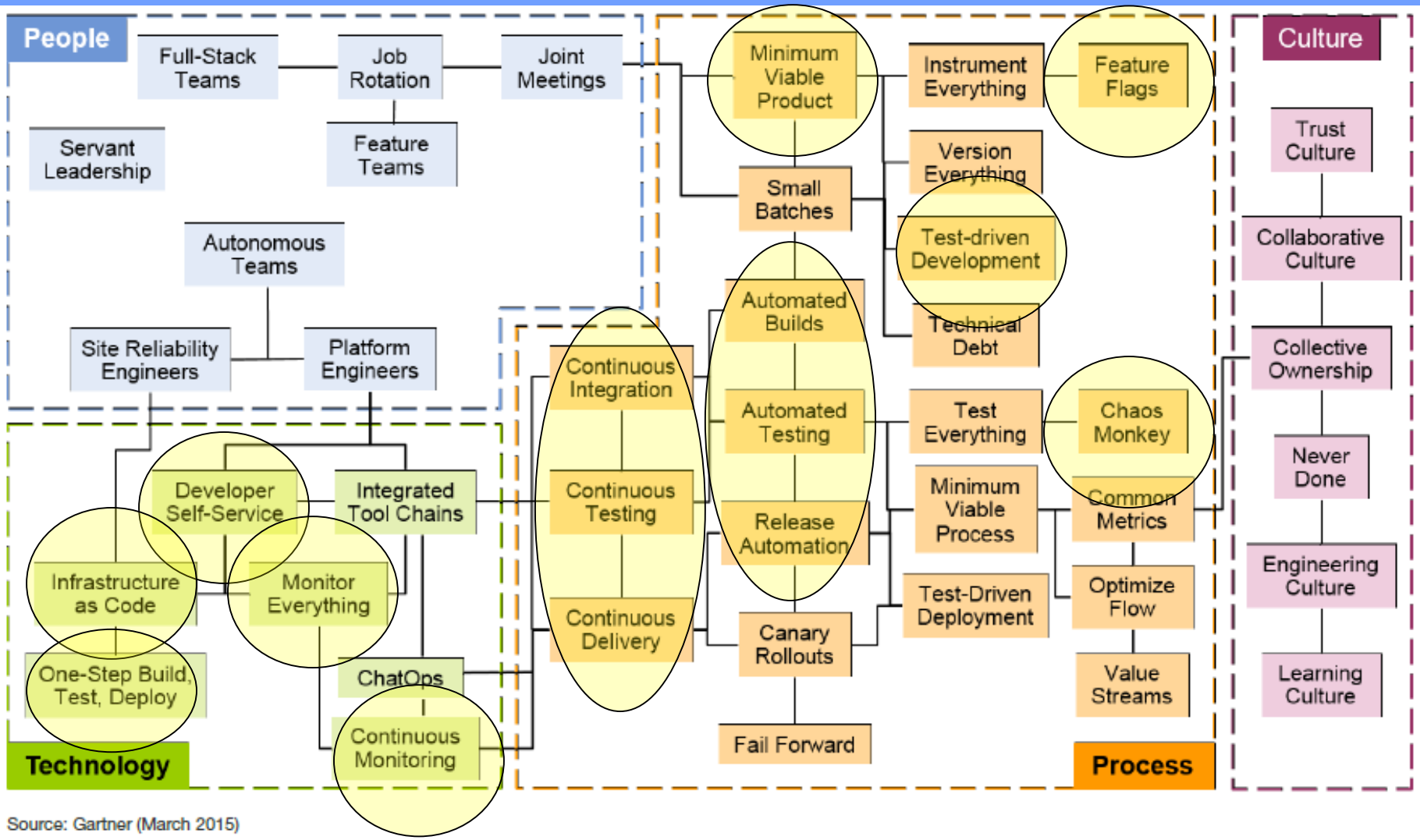



```
real    11m47.712s
user    0m0.031s
sys     0m0.110s
```

```
real    11m49.978s
user    0m0.015s
sys     0m0.093s
```

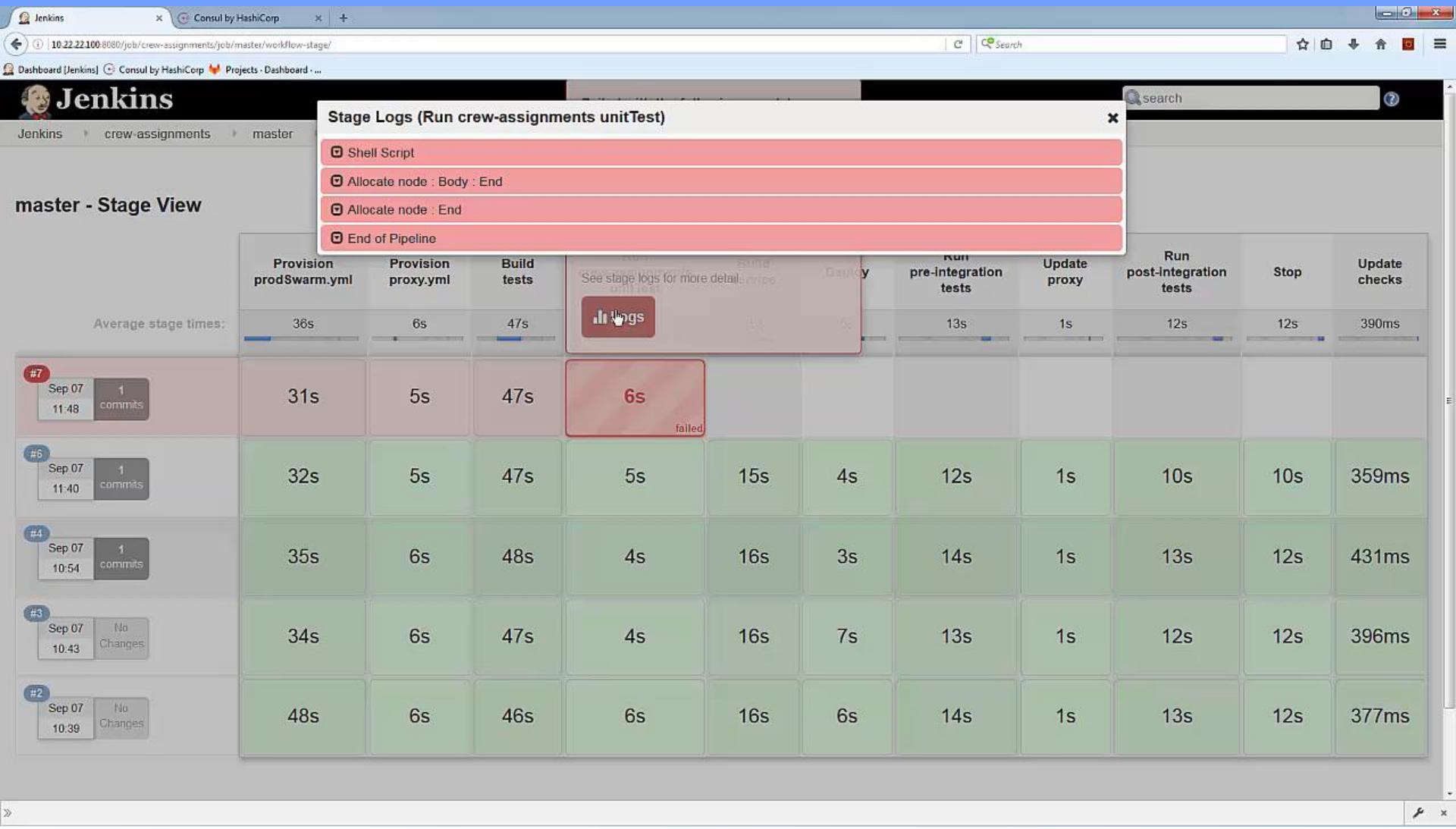
```
real    4m12.002s
user    0m0.000s
sys     0m0.031s
```

Gartner DevOps Model (Gartner 2015)

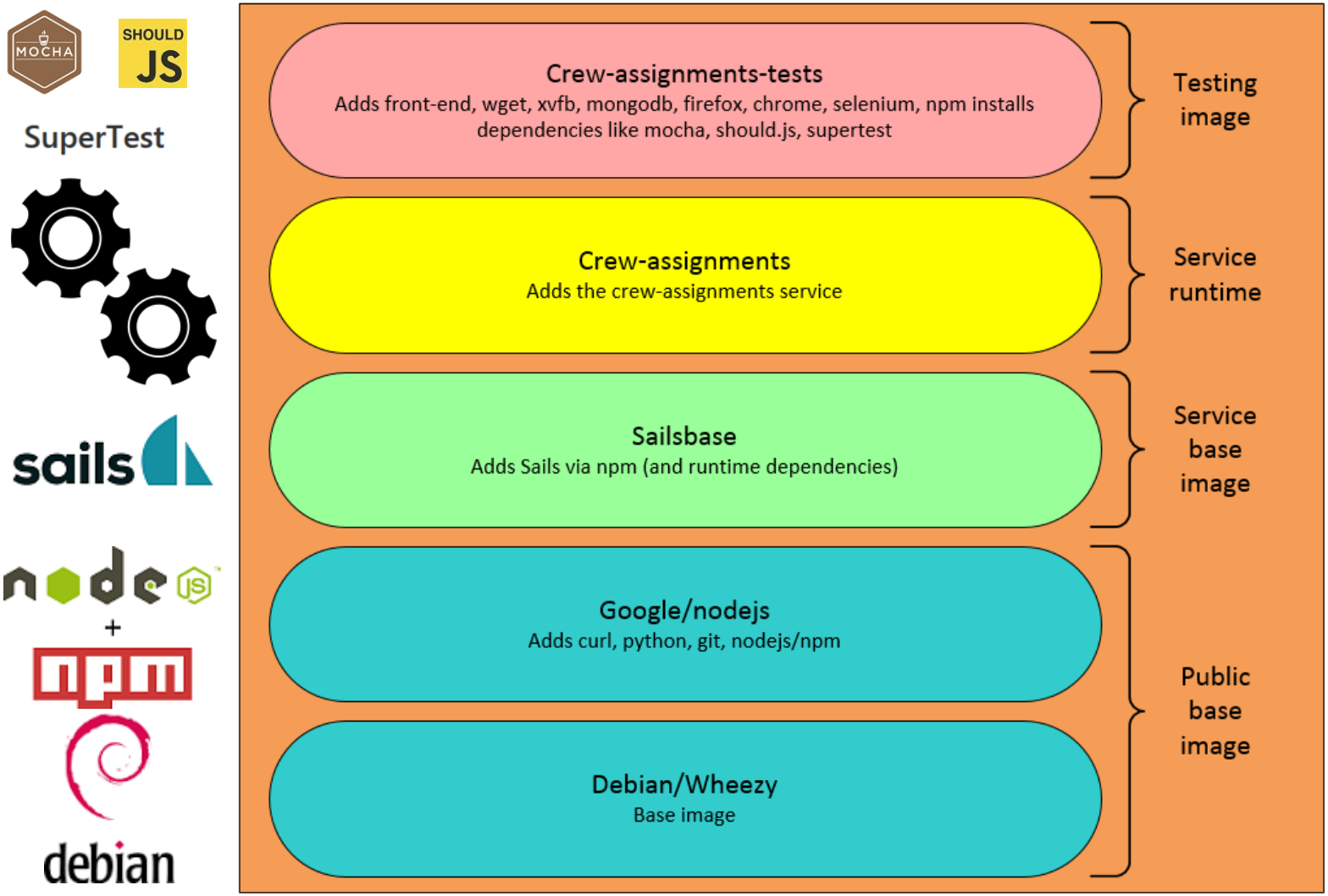


Source: Gartner (March 2015)

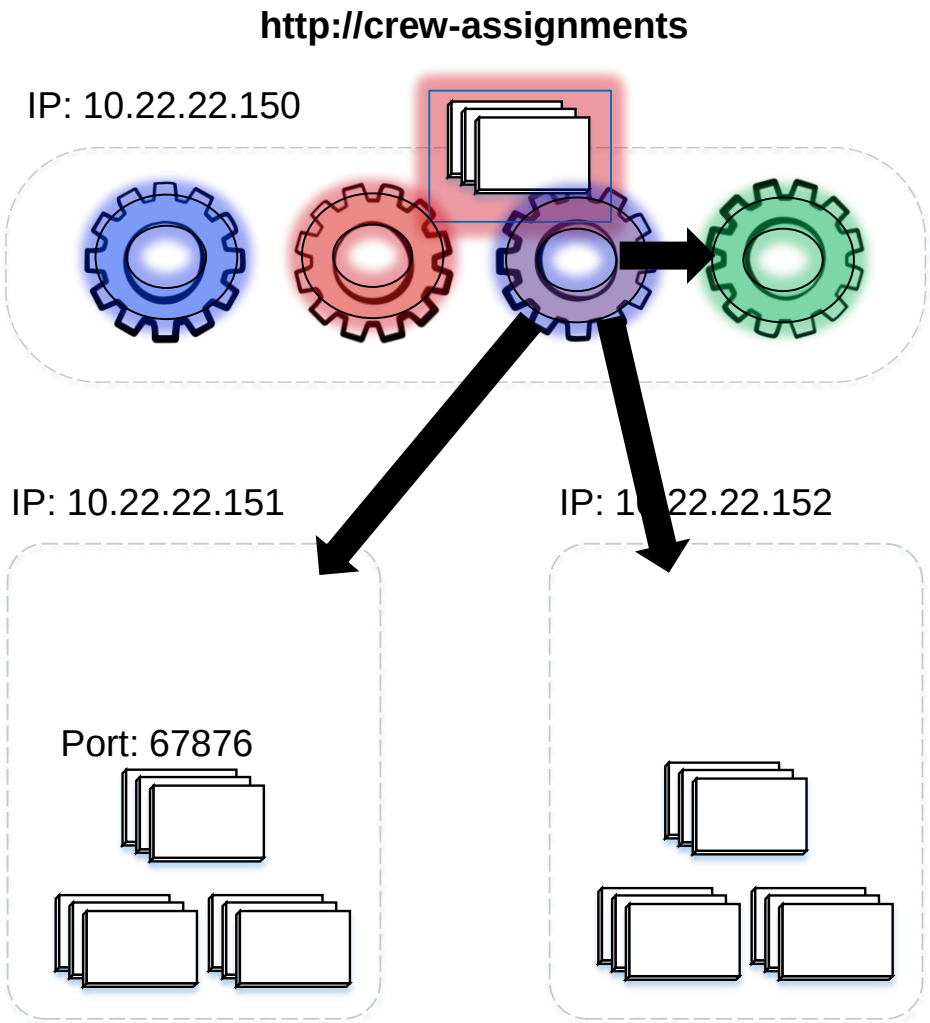
The Build Pipeline



crew-assignments Docker Layers



Managing a Single Service on Multiple Nodes



Jenkins Pipeline Deploys to the Swarm Master

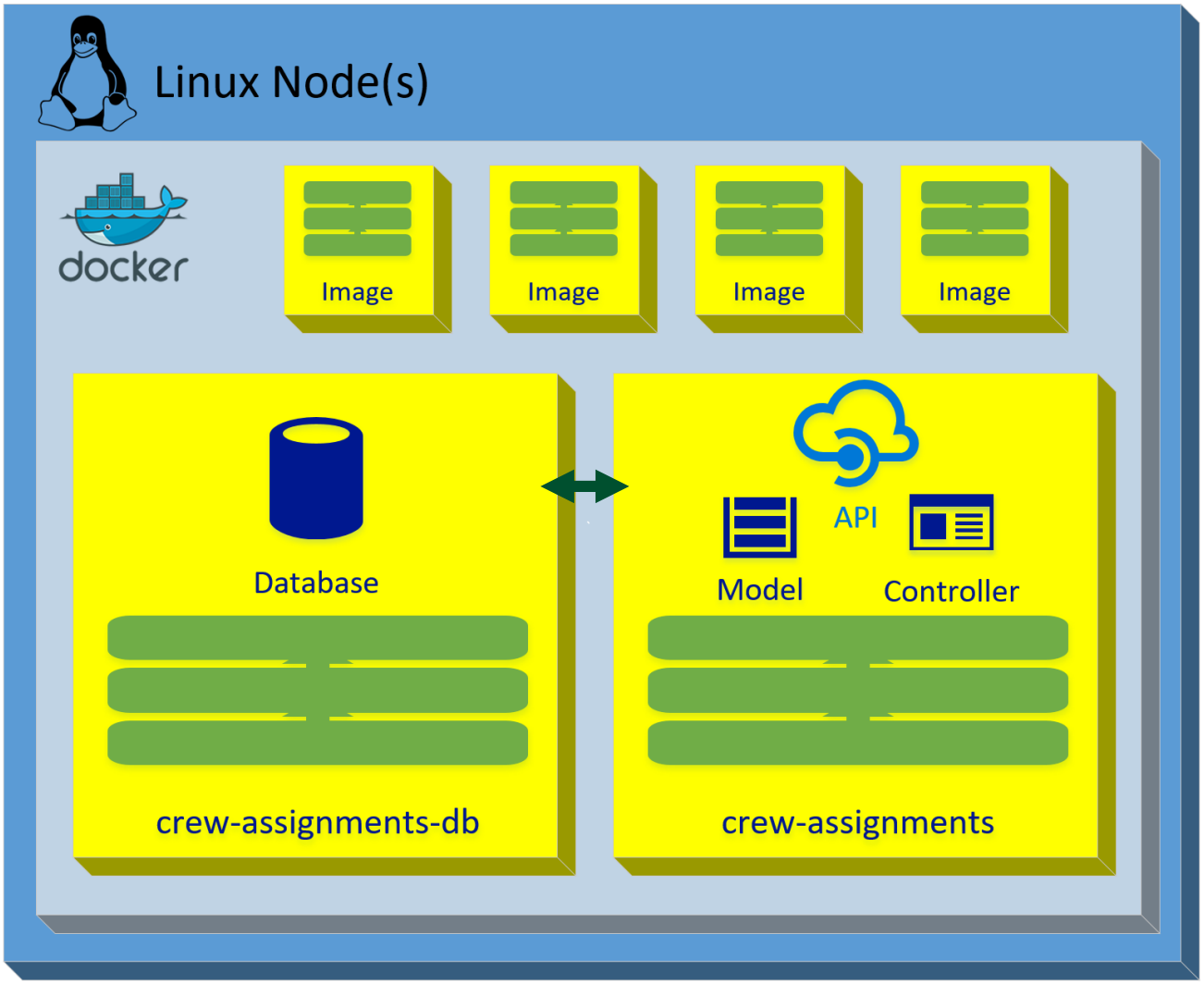
Swarm determines where the service should go

Registrator sees the deployment and notes relevant information

Consul updates our reverse proxy, consul instances on nodes and updates / creates service checks

Nginx is now our gateway with our predetermined ip, name, and / or port (and can do more than we show – like ssl

crew-assignments Runtime Environment



What our Setup Can Do

Blue Green Deployments

Zero-Downtime Deployments

Continuous Monitoring & Response

Know the health of every node and every container in those nodes

Automated Scaling

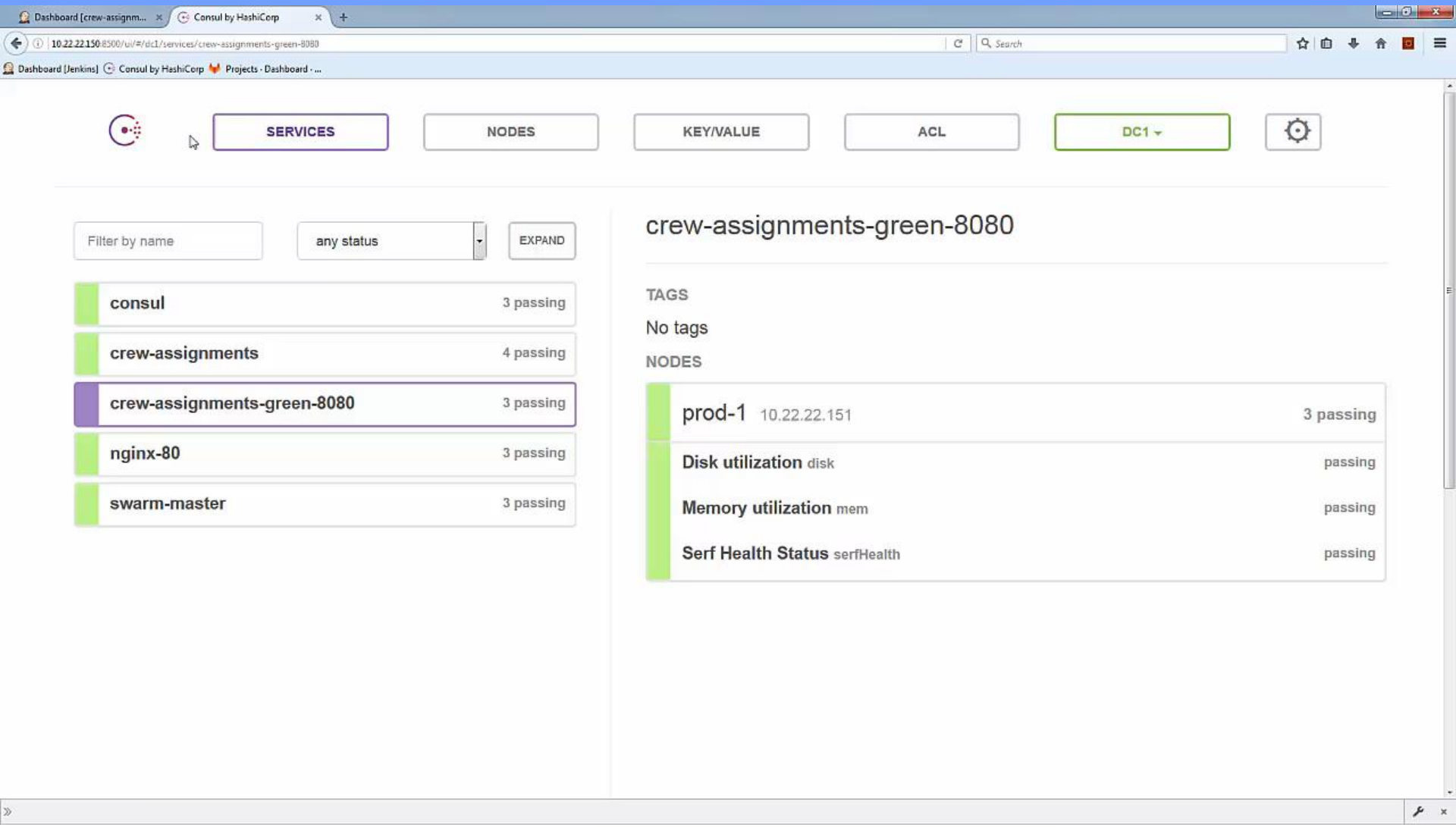
Plan for Load
React to Load

Self Healing

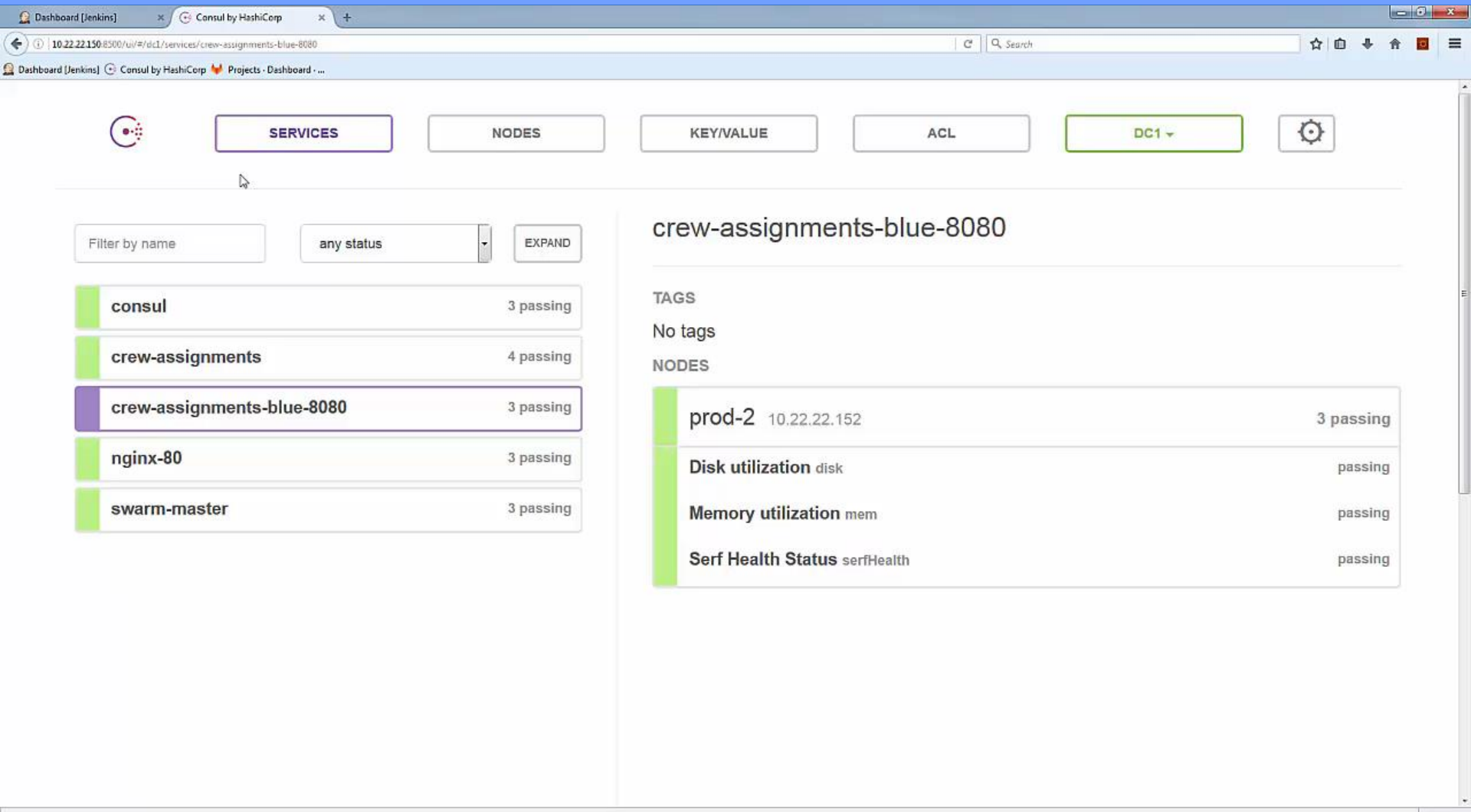
Automatically recover when services stop responding (or even when nodes or entire datacenters stop responding)

Zero Downtime Blue | Green Deployments

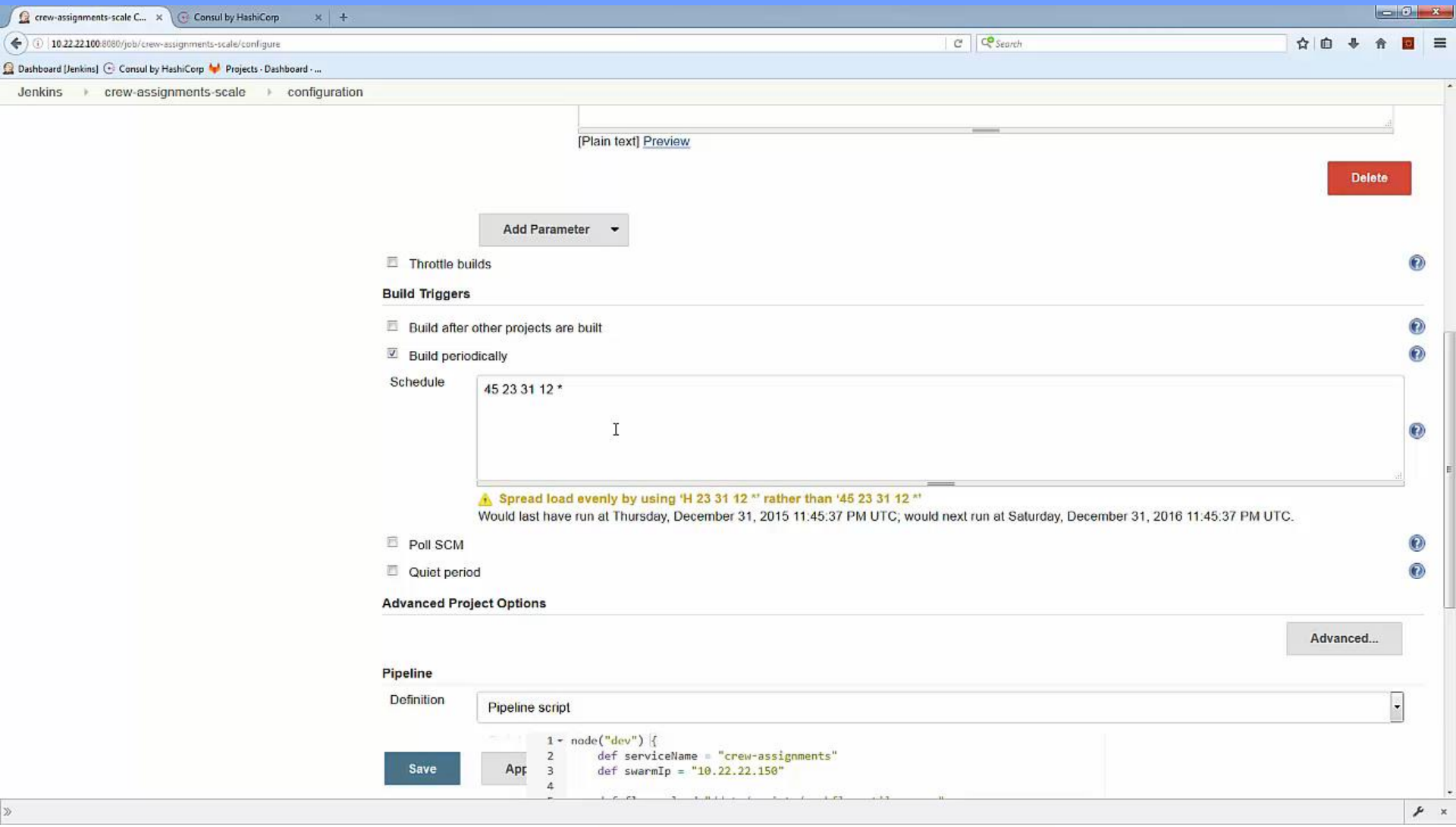
Global Product Data Interoperability Summit | **2016**



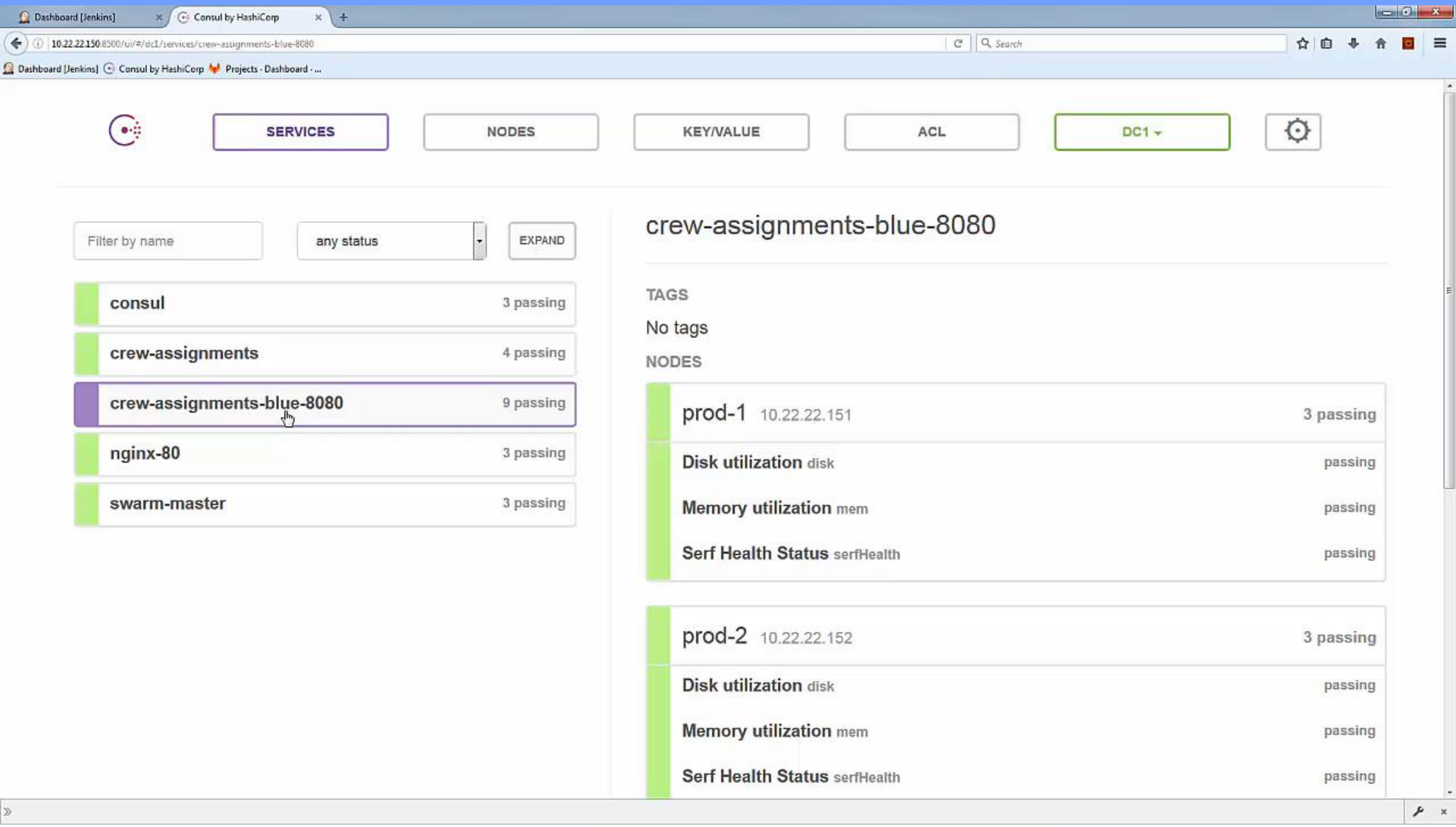
Scaling (X-Axis)



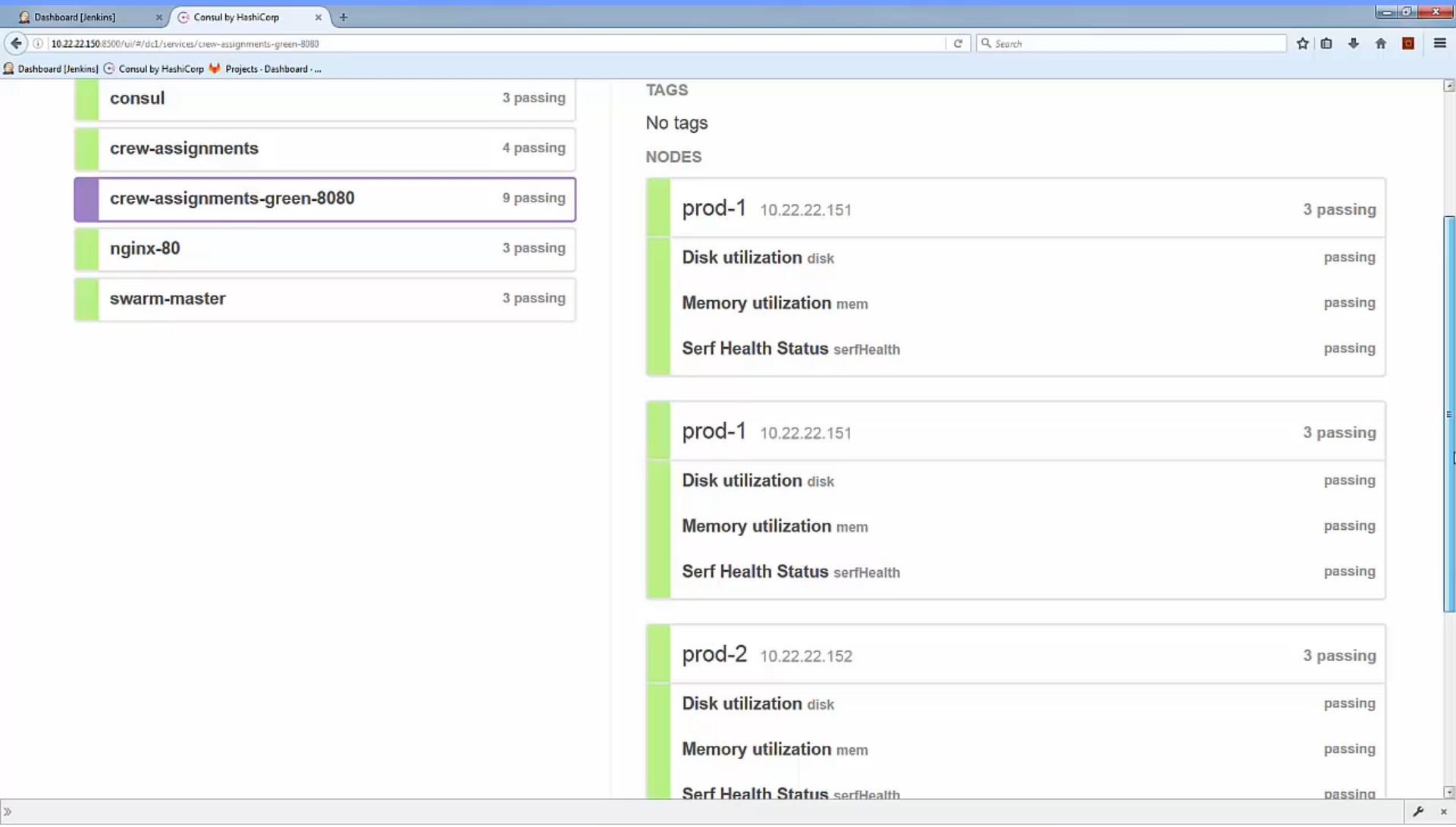
Scaling and Descaling



Rollback to Previous Runtime



Healing



Logging and Analytics

Logging & Analytics

Predict and Avoid

elastic

(Formerly ELK)

Data Visualization — kibana

Data Analytics — elasticsearch

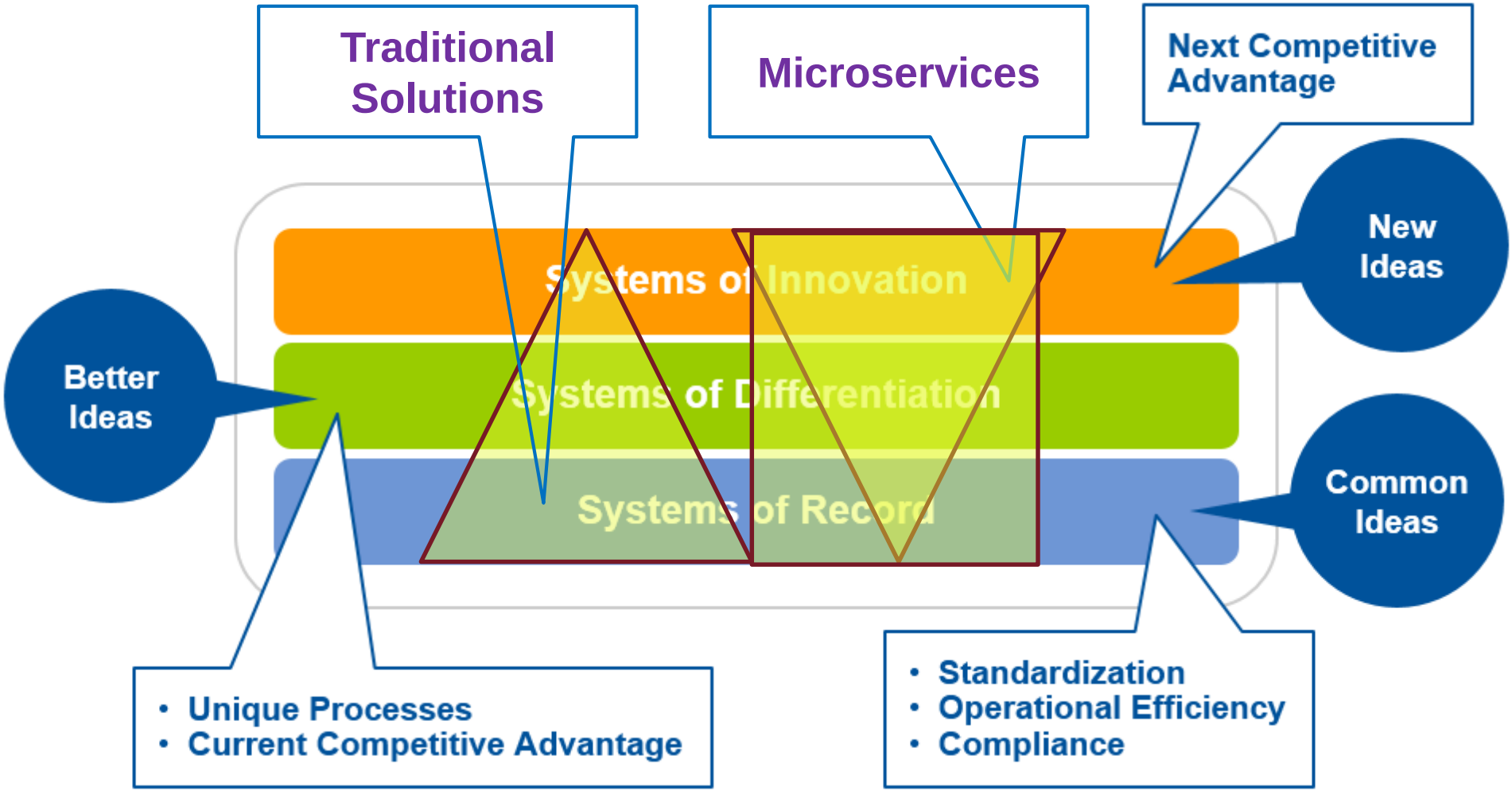
Data Collection
(Centralized Logging) — logstash

Our setup has all three preconfigured (via Ansible) and each is running in a container

Are We Missing Anything?

- **Of course we are! DevOps *must include* Continuous Improvement**
  - **Our example database is running in one container. We need to apply X- Axis scaling to our Z-Axis solution**
  - **Our Build Pipeline does not include any static analysis or security testing**
  - **We have a HUGE architectural problem: Our aggregation layer itself is not redundant or scaled**
    - **Thankfully, Docker Swarm, Consul, and Nginx all support clustering themselves**
  - **We need ALM Integration**
  - **Maybe implement an enterprise service registry so we can find service available to develop against? Maybe not so that we keep coupling loose.**
  - **More?**

Microservices: Not for Everything



Note: Adapted from Gartner (2014)

Summary – Why Microservices?

- Leverage DevOps concepts and tooling to drastically **decrease release cycle** time so much so that **Continuous Deployment** can be implemented
- Infrastructure and Tooling can be easily **replicated**
- **Development teams are small** – maybe even one developer (but, there are others involved – Architect, QA, System Administrators, etc.)
- Containers ensure all parts of an application are **developed, tested, and deployed** via the **same process** and that the service can run anywhere (bare metal, VM, cloud)
- They can be **scaled** (up or down) very easily – **even automated**
- Monitoring can not only **detect issues**, but actually **heal the system** or **prevent** an issue from ever happening

Note: Adapted from Gartner (2014)

Additional Resources and Informational Slides

- Some of the tooling and setup of the demonstrations used in the presentation were modified after reading the book *The DevOps 2.0 Toolkit* by Viktor Farcic (Farcic, 2016). I highly recommend this book for anyone who wants a hands-on look at these concepts.

- For a better understanding of scaling, scalability, and related concepts, a great resource is *The Art of Scalability* (Abbott and Fisher, 2015).

- Boeing personnel please look for us on inSite.

Note: Adapted from Gartner (2014)

What are Microservices?

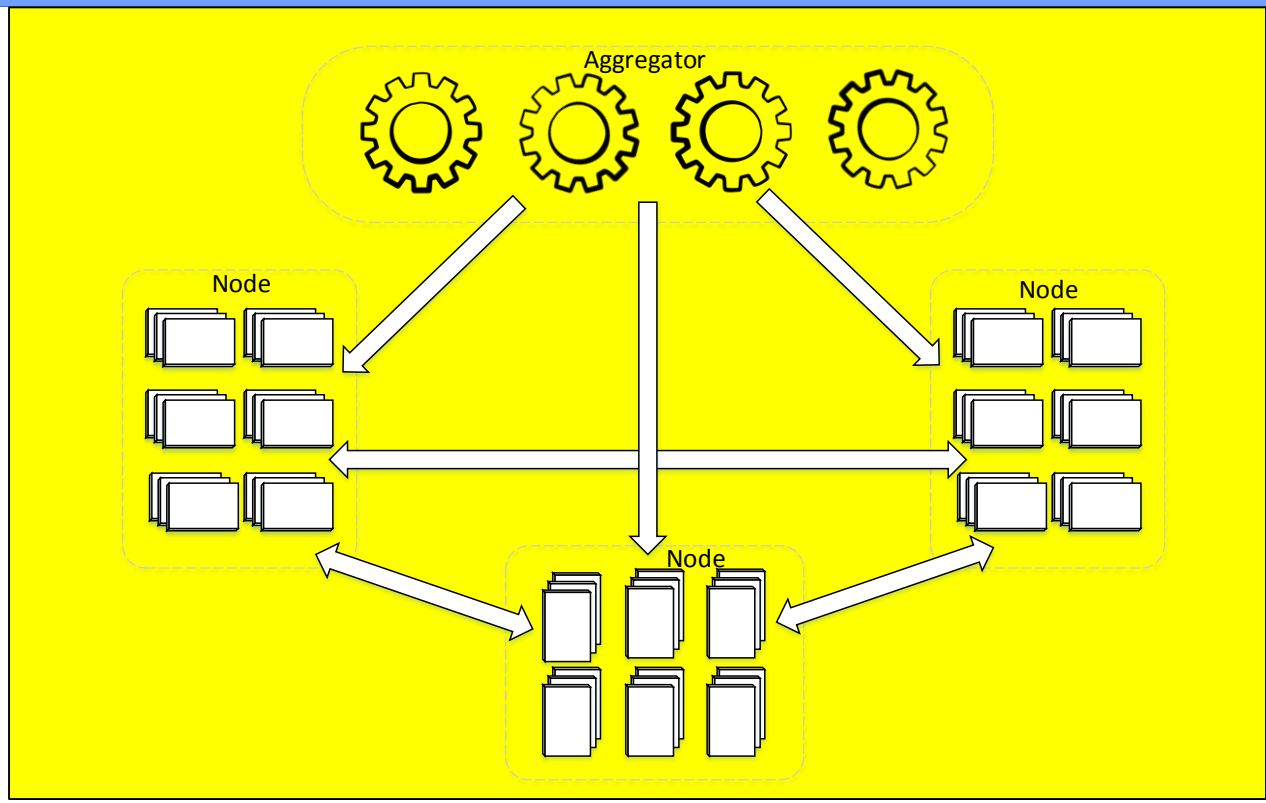
- **Definitions**
  - Microservices are a more concrete and **modern** interpretation of **service-oriented architectures (SOA)** used to build **distributed** software systems. It is an architectural style that is a first realization of SOA after the introduction of **DevOps** and this is becoming the standard for building **continuously deployed** systems. (Microservices, August 9, 2016)
  - Microservices are an approach to developing a single application as a **suite of small services**, each running in its own process and communicating with **lightweight** mechanisms, often an HTTP resource API. These services are built around **business capabilities** and **independently deployable** by **fully automated** deployment machinery. (Fowler, 2014)

COMCAST amazon UBER NETFLIX Capital One Bank ebay

Company list source: Richardson (2014)

Microservices Characteristics

- **Quickly** developed
  - Speed over elegance
- **Stateless**
- Designed for **Failure**
  - Netflix Simian Army
- **Elastic**
- Easily **replaceable**
- Use **Continuous Delivery**
- **Modular** in structure
- **Independently** deployable
- Technology **agnostic**
- **Finely-grained**
  - "**Do one thing and do it well**" (Unix philosophy, July 31, 2016)
- Typically implemented via **APIs** (Application Programming Interface) over **HTTP/REST** (Representational State Transfer) using **JSON** (JavaScript Object Notation)
  - Not a requirement



What is DevOps?

Gartner's Definition of DevOps (Gartner, 2014) :

- "a change in **IT culture**, focusing on **rapid** IT service **delivery** through the adoption of **agile**, **lean** practices in the context of a **system-oriented** approach."
- "emphasizes people (and culture), and seeks to improve **collaboration** between **operations** and **development** teams. Implementations utilize technology - especially **automation** tools that can leverage an increasingly **programmable** and **dynamic infrastructure** from a lifecycle perspective."

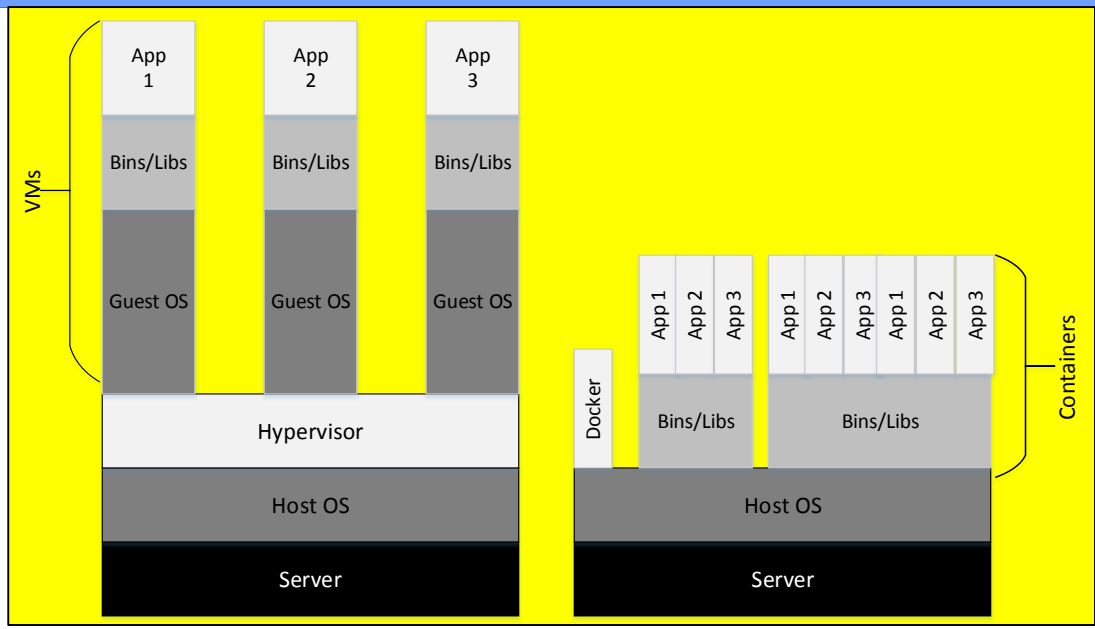
Gartner further identifies 5 primary principles that underpin DevOps (Gartner, 2015) :

- Iterative: well aligned with uncertainty; exhaustive planning is not optimal.
- Continuous: delivery & deployment; optimizing/experimenting with new processes, tools and org structures.
- Collaborative: agreement on the mission and metrics; transparent and frequent communications.
- Systemic: Agile initiatives not just focused on *development*, but downstream *operations*
- Automated: technologic facilitator to deliver speed and scale with human involvement only by exception

What are Containers?

Characteristics of Containers

- **Build once**, run anything anywhere
  - Completely portable -- no inconsistencies between development, test, production, or customer environments
- **Complete**
  - Dependent libraries and binaries
  - Configuration files
  - Middleware
  - Environment changes are built with the code and not as a separate process
- **Immutable**
  - No more “it ran fine on my box”, debug production issues using the production image in another environment.
  - Simpler scaling (X axis)
  - Enabler for self healing
- **Lightweight**
  - Easy to store, retrieve, change, deploy, and redeploy
  - Lower cost and higher performance than VMs alone.

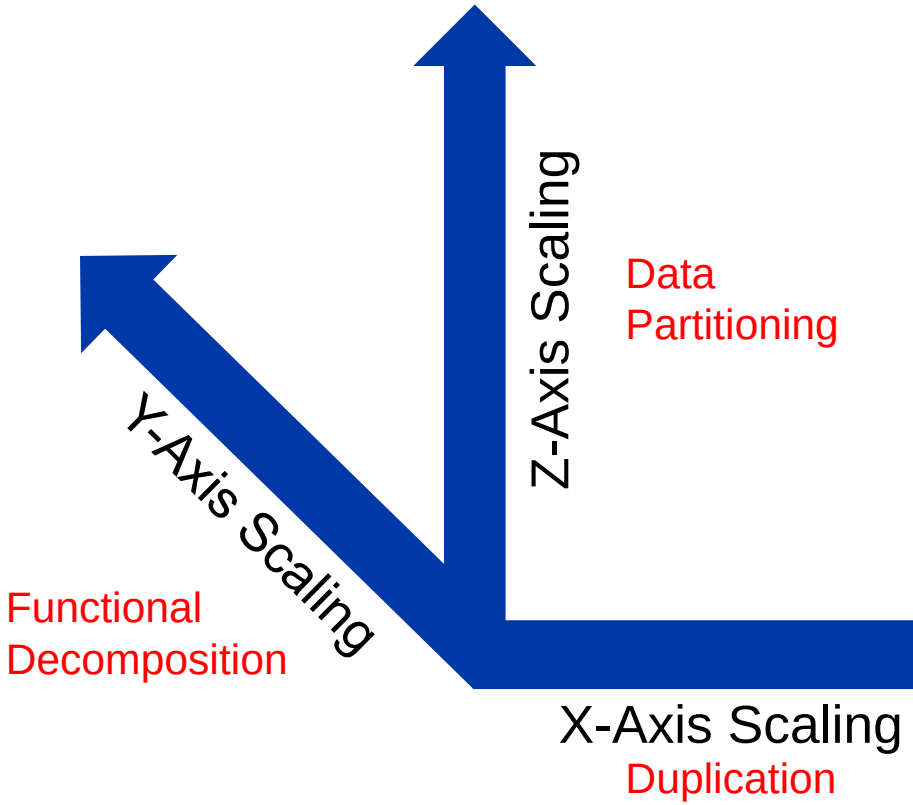


A container is an isolated user-space virtualization instance. Think of them as managed chroot jails.

Containers are **isolated** but **share** OS and binaries and libraries where appropriate.

The result is significantly **faster** deployment, **less** overhead, **easier** migration, and **quicker** restarts.

Three Dimensional Scaling



X-Axis

- What most think of as scaling
- Clones running behind load balancers
- Can be resource intensive
  - Especially with large applications

Y-Axis

- Scale by splitting
- Typically implemented via SOA and microservices

Z-Axis

- Scale by partitioning data
- No centralized monolithic database system
- Each piece of an application is "responsible" for its own data

Adapted from Abbott and Fisher (2015)

Microservices Versus SOA and Monolithic Applications

Category	Microservice	Traditional SOA	Monolith
Typical Lines of Code	Typically less than 100	Hundreds to Thousands	Thousands to Millions
Data Model	NoSQL or Small SQL databases with existing RDBMS	Large RDBMS	Large RDBMS
Communication	Fast, lightweight, asynchronous messaging	Enterprise Service Bus, synchronous connections	N/A
Development Team	Very Small – possibly a single Developer	Normal Development teams, each focusing on one area.	Large teams of teams, with institutional knowledge
System Changes	Create a new service, abandon the old one	Modify existing services and architecture	Requires more architectural analysis, knowledge of large code bases, and seasoned Developers
Release Schedule	Continuous Delivery	Weeks to Months, coordination needed	Long cycles, Blockpoints
Scaling	Scales well X, Y, and Z axis	Scales in X, limited Y and Z axis	Difficult to scale in X axis, No Y axis scaling, Limited Z

Orchestration and Managing More than One Node

Here's what we don't want: The maintenance nightmare of keeping up with deployment scripts and configurations in each of our services, in multiple nodes, especially since they may need to know about each other.

We need something to manage a "cluster" of nodes for us

This is the first part of our scaling (more to come), so how do we implement basic X-Axis scaling?

"**Docker Swarm** is native **clustering** for **Docker**. It turns a pool of Docker hosts into a single, **virtual Docker** host." (Docker, n.d.)

- Runs on the **aggregation layer**
- Is a **Docker Container**
- The **CD Pipeline** (Jenkins) has a step to make sure it is running and available (using **Ansible**)
- Installed as a Swarm **Master** and Swarm **nodes**

Discovery, Registration, and Reverse Proxy

Consul has multiple components, but as a whole, it is a tool for discovering and configuring services in your infrastructure. It provides several key features:

- **Service Discovery**
- **Health Checking**
- **Key/Value Store**
- **Multi Datacenter**

(Consul, n.d.)

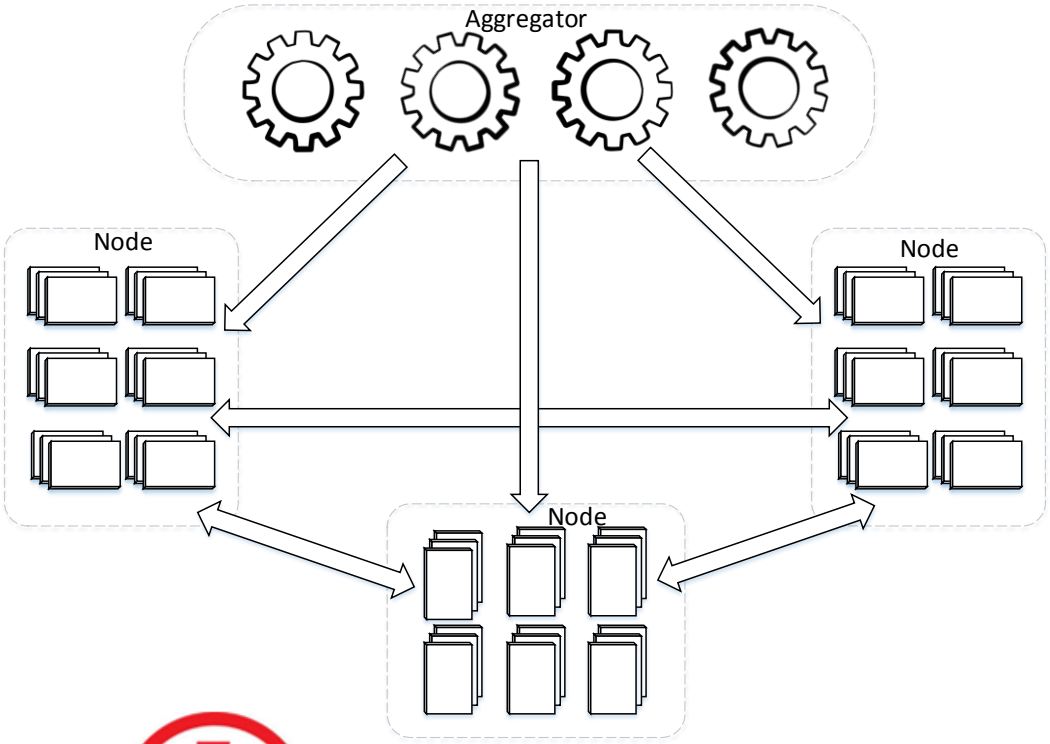
NGINX is a free, open-source, high-performance HTTP server and **reverse proxy**, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. (Nginx, n.d.)

Our Nginx runs in a Docker container.

Registrator automatically **registers and deregisters** services for any **Docker** container by inspecting containers as they come online. (Gliderlabs n.d.)

Our Registrator runs in a Docker container

Production Overview



Orchestration

Proxy Services

Automated Scaling

Logging & Analytics

Continuous Monitoring & Response

Discovery

Self Healing

Blue Green Deployments

References

Richardson, Chris (2014). *Microservice architecture patterns and best practices*. Retrieved from http://microservices.io

Microservices (August 9, 2016) *Wikipedia.* Retrieved August 16, 2016 from https://en.wikipedia.org/wiki/Microservices

Fowler, M (2014). *Microservices, A Definition of This New Architectural Term.* Retrieved from http://martinfowler.com/articles/microservices.html

Unix philosophy (July 31, 2016) *Wikipedia.* Retrieved August 12, 2016 from https://en.wikipedia.org/wiki/Unix_philosophy

Abbott, Martin L. & Fischer, Michael T. (2015). Introduction to AFK scale cube. In *The art of scalability: Scalable web architecture, processes, and organizations for the modern enterprise* (2nd ed., pp 343-356). Mark L. Taub (Ed.). Old Tappan, NJ: Pearson Education

Gartner, Mangi, L & Gaughan, D (April 23, 2015). How to Develop a Pace-Layered Application Strategy (ID: G00276478). Retrieved from Gartner database.

Gartner, Colville, R (July 22, 2014). *Hype Cycle for IT Operations Management, 2014* (ID: G00263503). Retrieved from Gartner database.

Gartner, Haight, C (March 12, 2015). *Principles and Practices of DevOps* (ID: G00272990). Retrieved from Gartner database.

References

Docker (n.d.). Docker Swarm overview in *Docker Swarm.* Retrieved from https://docs.docker.com/swarm

Gliderlabs (n.d.). *Registrator*. Retrieved from http://gliderlabs.com/registrator/latest

Consul (n.d.). *Introduction to Consul.* Retrieved from https://www.consul.io/intro

Nginx (n.d.). *NGINX Wiki Documentation.* Retrieved from https://www.nginx.com/resources/wiki

Farcic, Viktor. (July 20, 2016). *Devops 2.0. Automating the continuous deployment pipeline with containerized microservices. CreateSpace Independent Publishing Platform*

All statements in this report attributable to Gartner represent Boeing's interpretation of data, research opinion or viewpoints published as part of a syndicated subscription service by Gartner, Inc., and have not been reviewed by Gartner. Each Gartner publication speaks as of its original publication date (and not as of the date of this presentation/). The opinions expressed in Gartner publications are not representations of fact, and are subject to change without notice.