DevOps INSTITUTE

SKILUP SM

Kubernetes Basics by Tracy Ragan on behalf of the Continuous Delivery Foundation

About Tracy

Tracy Ragan is CEO and Co-Founder of DeployHub. DeployHub is the first microservice management platform designed to facilitate the sharing, relationship mapping and deployments of microservices. Tracy is expert in configuration management and pipeline life cycle practices with a hyper focus on microservices and cloud native architecture. She currently serves as a board member of the Continuous Delivery Foundation (CDF) where she is the elected General Member Representative.

Synopsis

Containers, microservices and Kubernetes is a tsunami size wave of disruption from our traditional ways of developing software. As a result, the CD Landscape will need tweaking to support a K8s CD Pipeline. In this session we will explore basic concepts of Kubernetes, containers and microservices, and review how this new architecture changes the traditional CD Landscape. We will look at changes occurring now, what the future may look like including the potential demise of the Dev, Test, Prod waterfall practice.

DevOps INSTITUTE

Key Takeaways

Kubernetes and **microservices are big shift**.

For most organizations, **microservices will have their own repository and workflows**. CD tools will need to support workflow templates.

Configuration management will be lost as large monolithic builds are replaced or non-existence. Link decision making is done at runtime – not by a build manager.

Organizing microservices into Domains becomes critical for sharing and re-use.

Library Management and Security moves to containers.

A Tsunami

The Modern Architecture of containers, Kubernetes and microservices is a massive shift from monolithic practices. It will impact:

1) The way we develop software.
2) The way we deliver software and our CI/CD Pipeline.
3) The way we manage and monitor our ‘data center.’

In other words, we will need to surf this wave into a new form of software development.

DevOps INSTITUTE

DevOps INSTITUTE

Let's Talk Kubernetes

From the Top

Containers and Docker:

A container is a standard unit of software that packages up code and all dependencies so the application runs quickly and reliably.

Kubernetes Container Orchestration:

An open source platform from Google for orchestrating containers across clusters of servers providing auto scaling and fault tolerance.

Microservices:
An architectural style that structures an application as a collection of loosely coupled services.

In a microservices architecture, services are fine-grained and **independently deployable.**

Why Kubernetes and Not Just Containers?

A container provides an isolated context in which an application, together with its environment can run.

- Need to be managed.
- Networking is hard.
- Must scheduled, distributed and load balanced.
- Data must persist somewhere.

Kubernetes Orchestration

Kubernetes ensures that the cluster continues to serve users no matter how its environment is altered, *with minimal intervention from operations teams.*

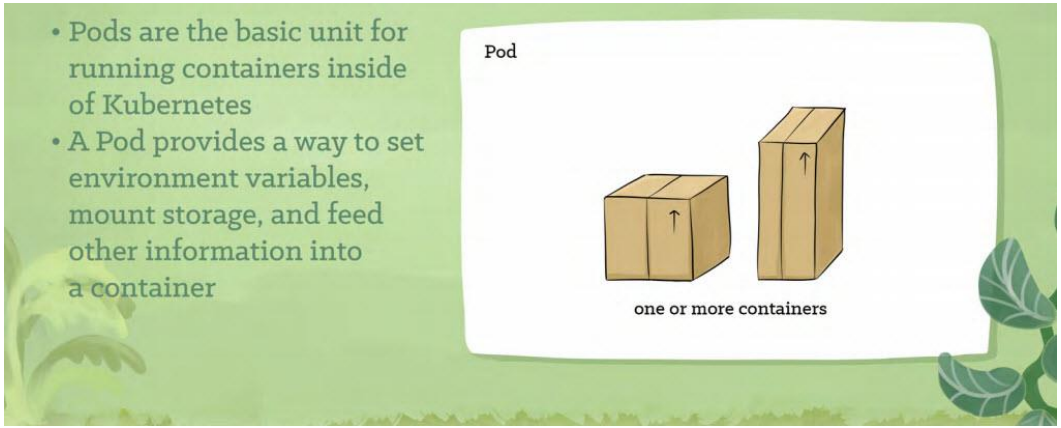
Self-Healing

- Finds and restarts failed containers.
- Finds and reschedules failed Nodes.
- Destroys unresponsive containers.

Kubernetes Parts and Pieces

Pods and Nodes

Nodes are a VM for Pods.

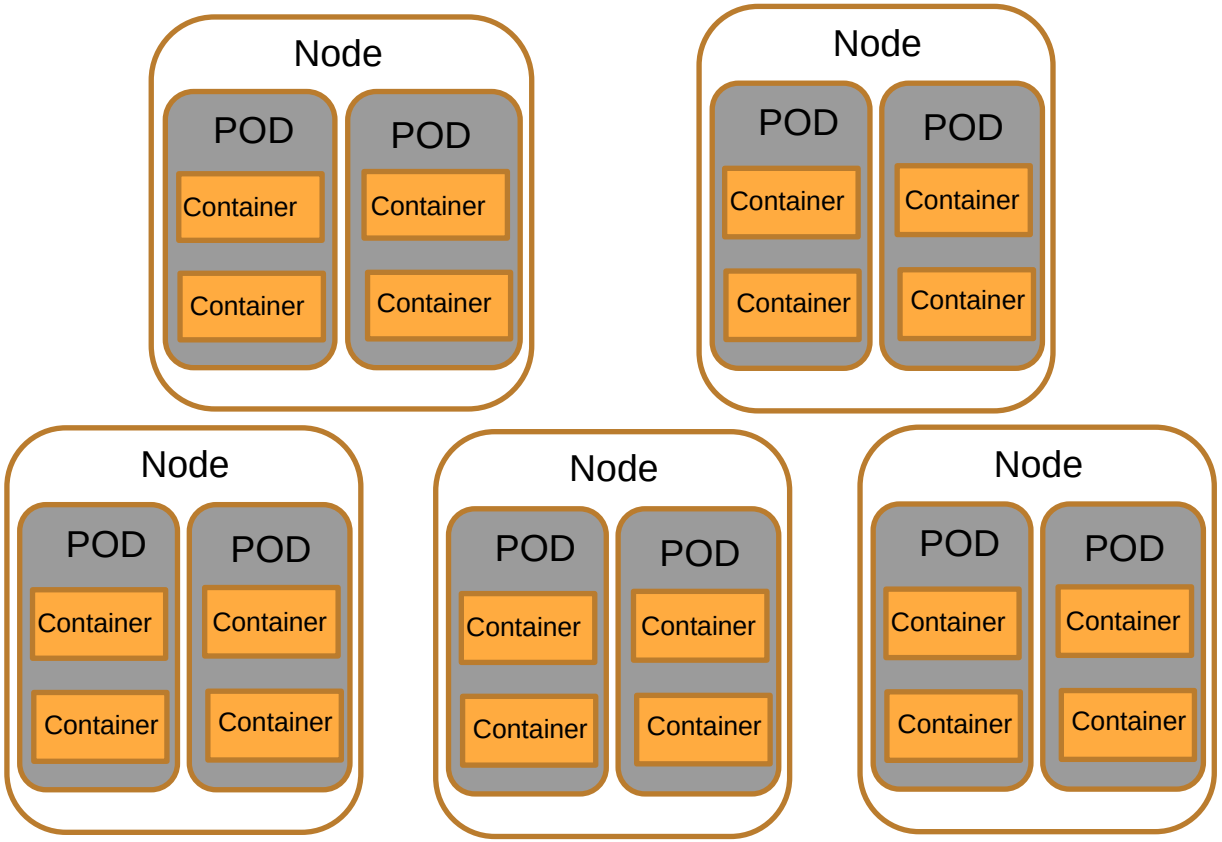


"Pods run your containers. There is at least one container for every Pod.

The Pod controls the execution of that container. When the container exit, the pod dies too."

Phippy Goes to the Zoo –A Kubernetes Story, by Matt Butcher & Karen Chu, Illustrated by Bailey Beougher, Renee French

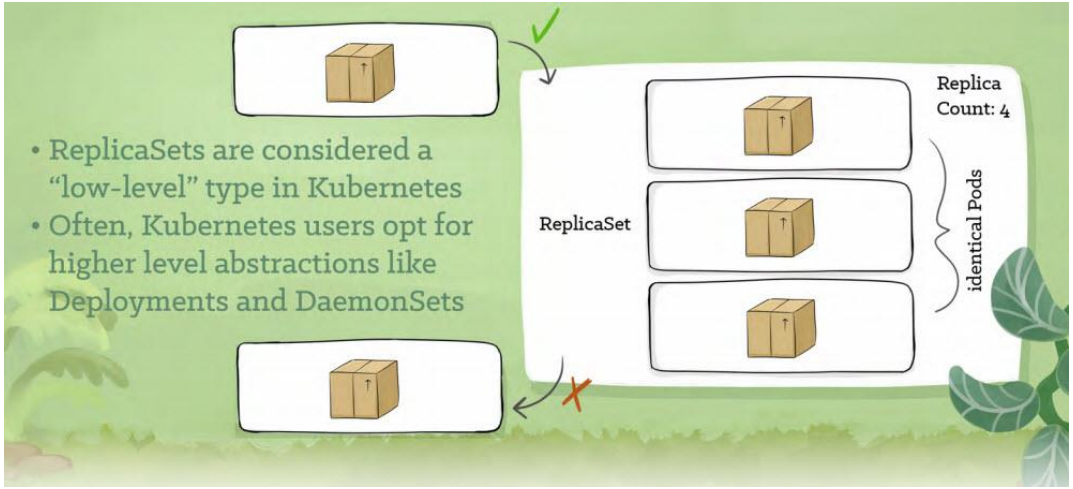
Kubernetes Cluster



DevOps INSTITUTE

Kubernetes Parts and Pieces

Replica Sets



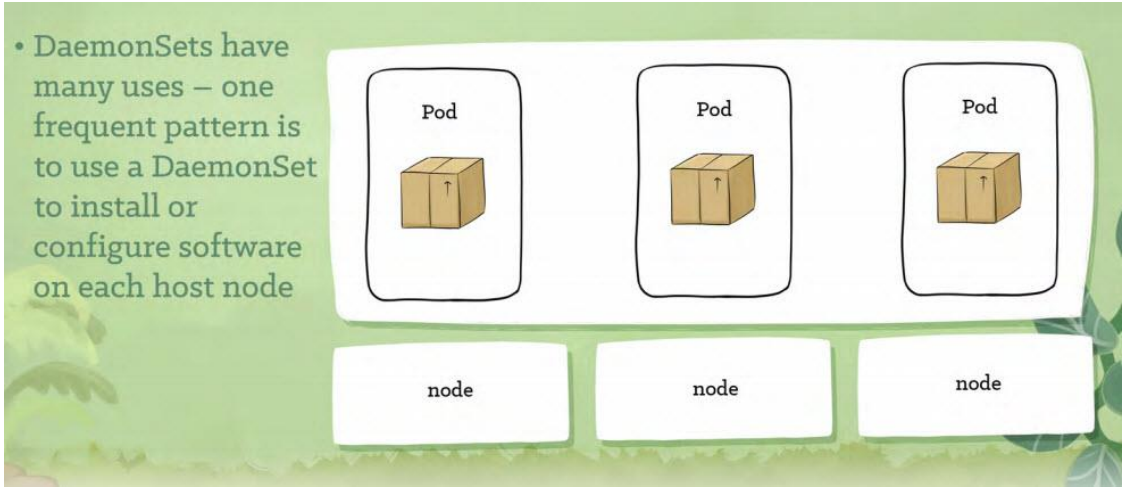
ReplicaSet ensures that a set of identically configured Pods are running at the desired replica count. If a Pod drops off, the ReplicaSet brings a new one online as a replacement."

Phippy Goes to the Zoo –A Kubernetes Story, by Matt Butcher & Karen Chu, Illustrated by Bailey Beougher, Renee French

DevOps INSTITUTE

Kubernetes Parts and Pieces

DaemonSets

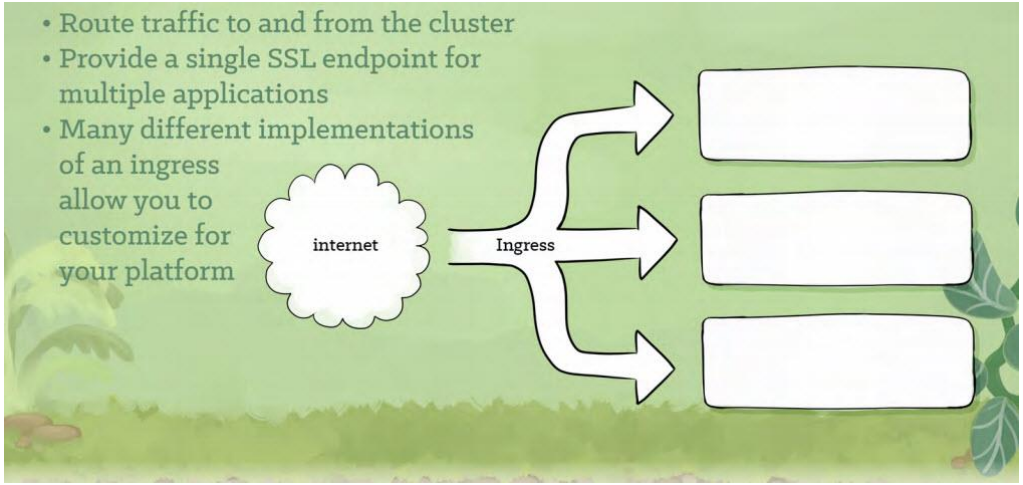


DaemonSets provide a way to ensure that a copy of a Pod is running on every node in a cluster.
As a cluster grows and shrinks, the DaemonSets spreads these specially labeled Pods across all of the nodes."

Phippy Goes to the Zoo –A Kubernetes Story, by Matt Butcher & Karen Chu, Illustrated by Bailey Beougher, Renee French

Kubernetes Parts and Pieces

Ingresses

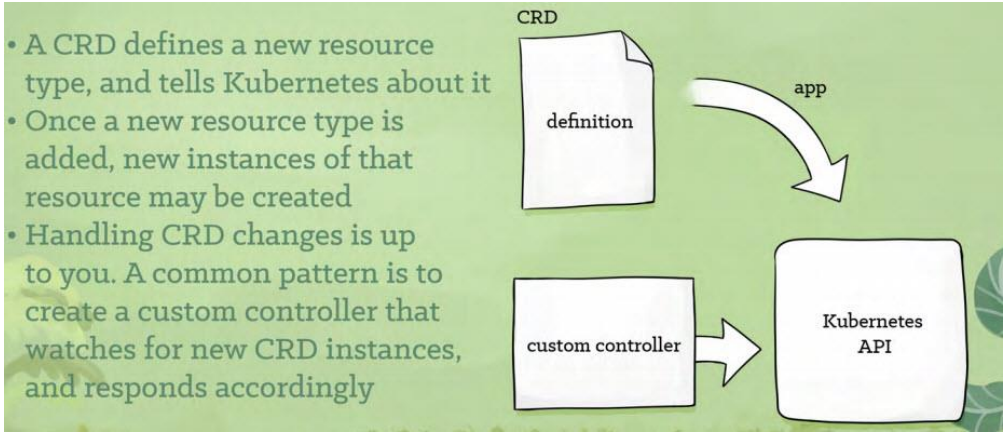


"Ingresses provide a way to declare that traffic ought to be channeled from the outside of the cluster into destination points within the cluster. One single external Ingress point can accept traffic destined to many different internal services."

Phippy Goes to the Zoo –A Kubernetes Story, by Matt Butcher & Karen Chu, Illustrated by Bailey Beougher, Renee French

Kubernetes Parts and Pieces

Custom Resource Definitions



CRDs provide an extension mechanism that cluster operators and developers can use to create their own resource types.“

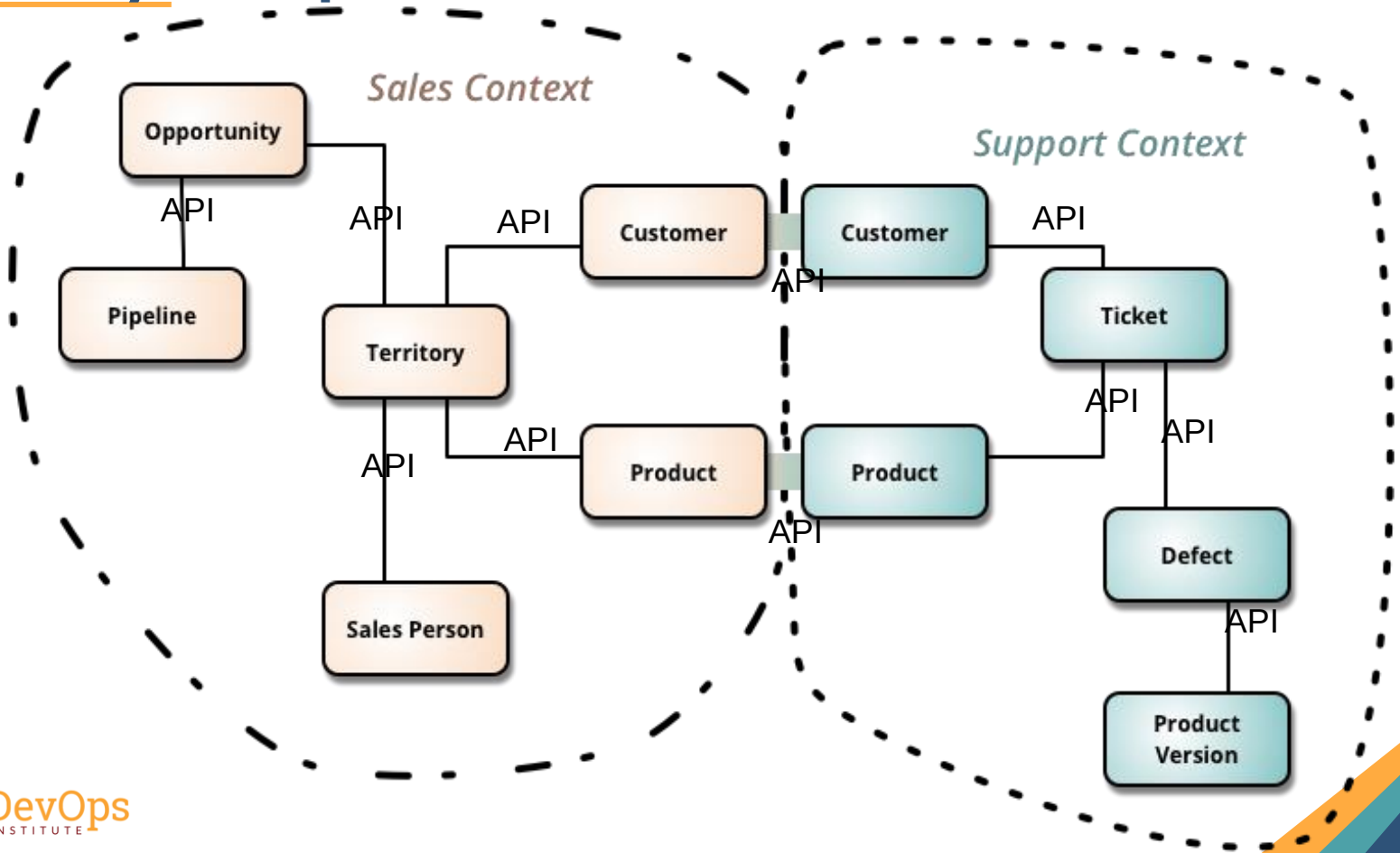
Phippy Goes to the Zoo –A Kubernetes Story, by Matt Butcher & Karen Chu, Illustrated by Bailey Beougher, Renee French

And Then There are Microservices

Microservices are a software development technique—a variant of the service-oriented architecture architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight.”

DevOps INSTITUTE

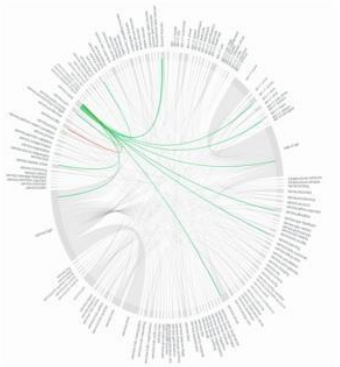
Loosely Coupled



DevOps INSTITUTE

Your New Challenge – The Death Star

450 microservices

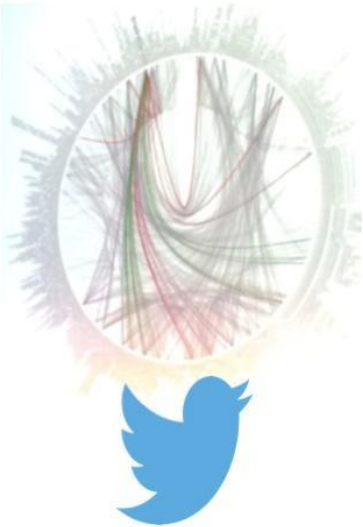


HAIL O

500+ microservices

NETFLIX

500+ microservices



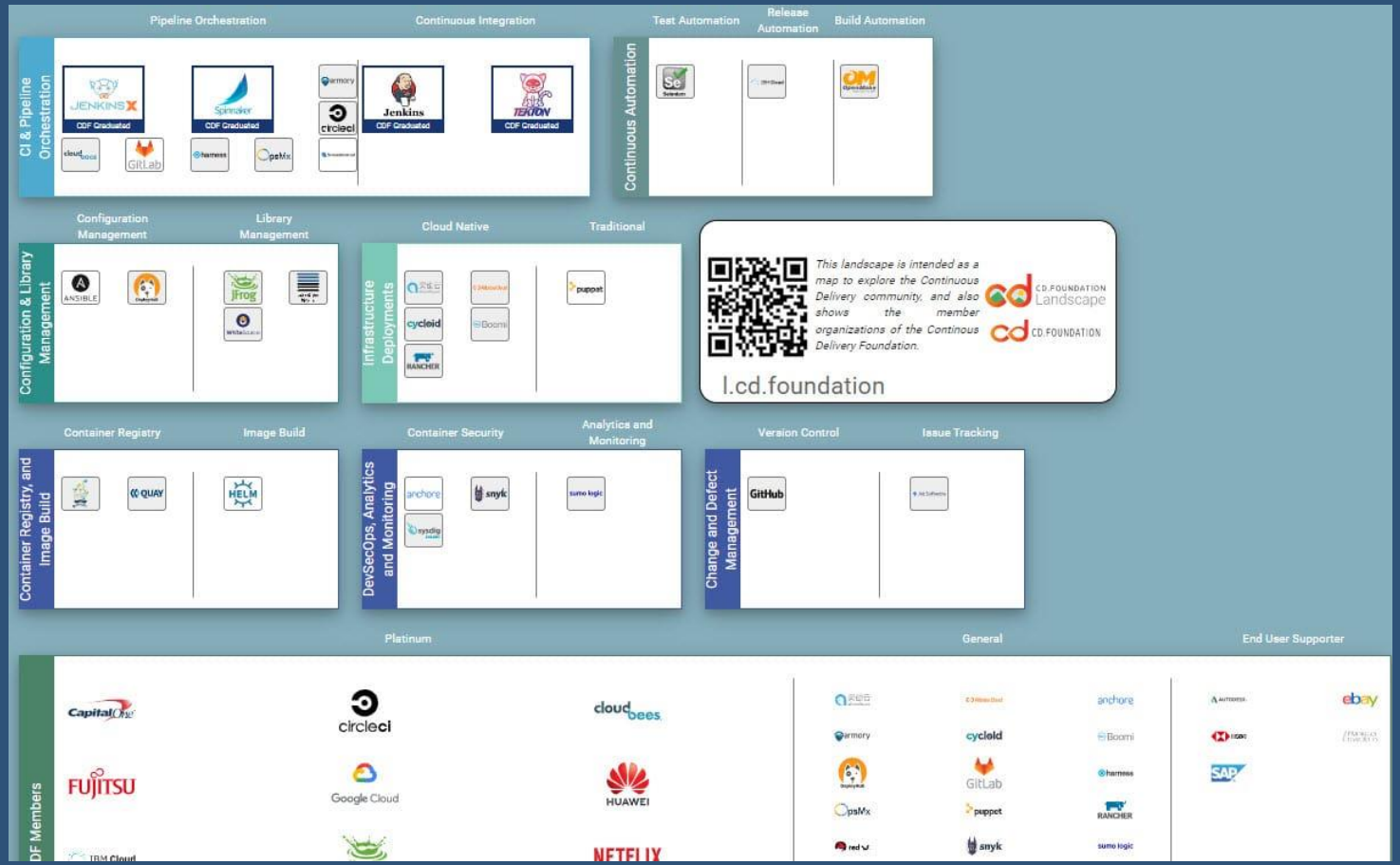
Source:
Netflix: http://www.slideshare.net/BruceWong3/the-case-for-chaos
Twitter: https://twitter.com/adrianco/status/441883572618948608
Hail-o: https://sudo.hailoapp.com/services/2015/03/09/journey-into-a-microservice-world-part-3/

DevOps INSTITUTE

DevOps INSTITUTE

Let's Talk CD Landscape

CD Landscape



Change Happens

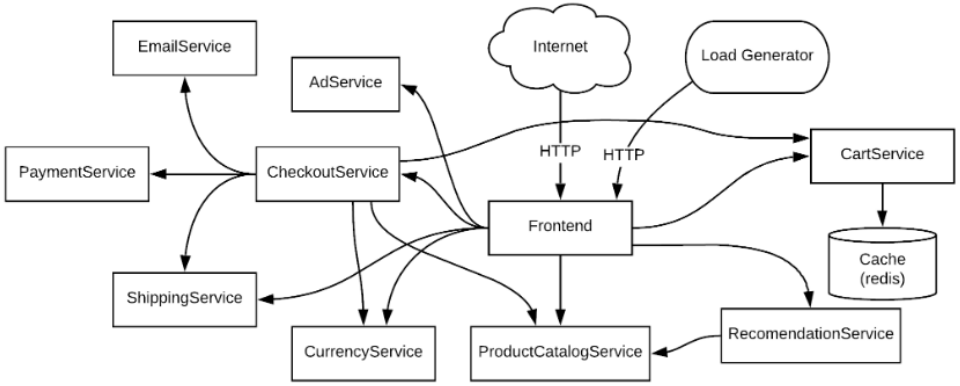
Shifting to a modern architecture will disrupt our traditional CI/CD pipeline.

Why is the CI/CD process disrupted?

Microservices are deployed independently and that change impacts everything.

DevOps INSTITUTE

Microservices and CD



The key to understanding microservices is to think ‘functions.’ With a microservice environment the concept of an ‘application’ goes away. It is replaced by a grouping of loosely coupled services connected via APIs at runtime, running inside of containers, nodes and pods.

Microservices are immutable. You don’t ‘copy over’ the old one, you deploy a new version to the cluster and manage them with *Labels*.

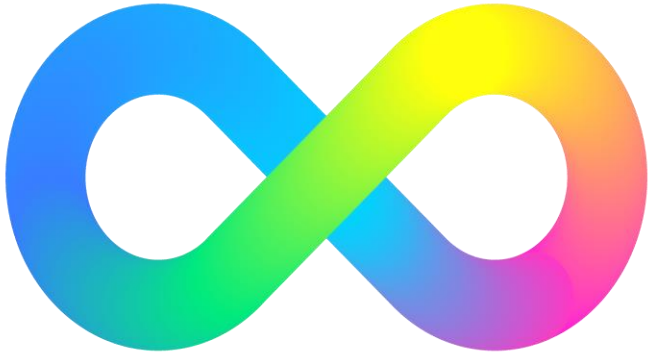
DevOps INSTITUTE

Changes in the CD Pipeline

Continuous Delivery Orchestration (CD)

Monolithic:

Continuous Delivery auto executes workflow processes between development, testing, and production orchestrating external tools to get the job done. Continuous Delivery calls on all players in the lifecycle process to execute in the correct order and centralizes their logs.

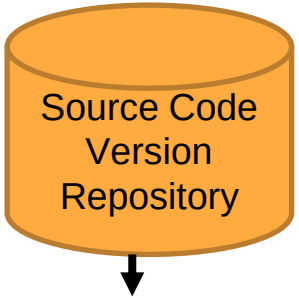


Microservices:

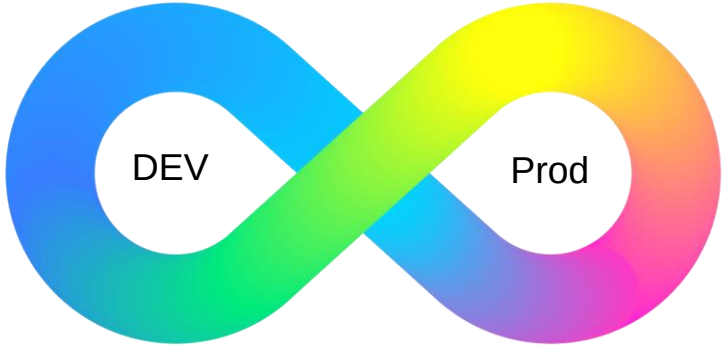
Because microservices are independently deployed, most organizations moving to a microservice architecture tell us they use a single pipeline workflow for each microservice. This means you are going to have hundreds if not thousands of workflows. CD tools will need to include the ability to template workflows allowing a fix in a shared template to be applied to all child workflows. Look for ‘event driven’ solutions to enter the market.

Shifting Configuration

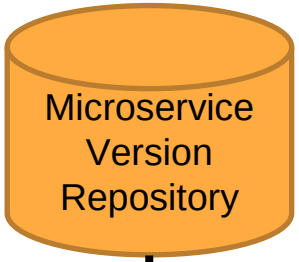
Monolithic CI/CD



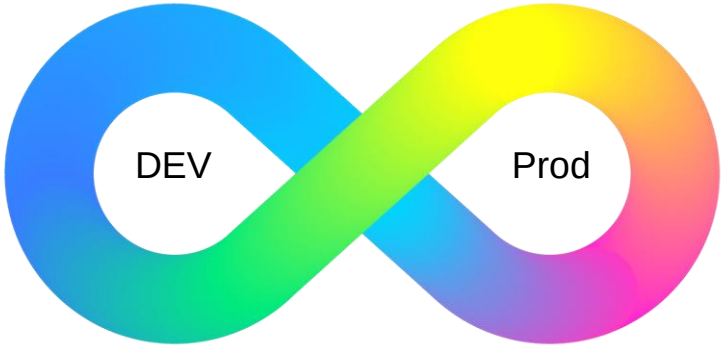
Compile and Link



Microservices CI/CD



API Link



DevOps INSTITUTE

Changes in the Build

Builds – Compile/Link

Monolithic:

Executes a serial process for calling compilers and linkers to translate source code into binaries (Jar, War, Ear, .Exe, .dlls, docker images). Common languages that support the build logic includes Make, Ant, Maven, Meister, NPM, PIP, and Docker Build.

Microservices:

A build of a microservice will involve creating a container image and resolving the dependencies needed for the container to run. You can think of a container image to be our new binary. This will be a relatively simple step and not involve a monolithic compile/link of an entire application. It will only involve a single microservice. Linking is done at runtime with the restful API call coded into the microservice itself.

Changes in CI

Continuous Integration (CI)

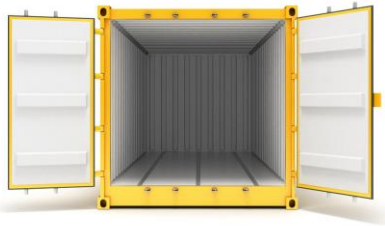
Monolithic:

CI is the triggered process of pulling code and libraries from version control and executing a Build based on a defined ‘quiet time.’ This process improved development by ensuring that code changes were integrated as frequently as possible to prevent broken builds, thus the term continuous integration.

Microservices:

Continuous Integration was originally adopted to keep us re-compiling and linking our code as frequently as possible in order to prevent the build from breaking. The goal was to get to a clean ’10-minute build’ or less. With microservices, you are only building a single ‘function.’ This means that an integration build is no longer needed. CI will eventually go away, but the process of managing a continuous delivery pipeline will remain important with a step that creates the container.

Changes in Artifact Repositories



Artifact (Binary) Repository

Monolithic:

Originally built around Maven, an artifact repository provides a central location for publishing jar files, node JS Packages, Java scripts packages, docker images, python modules. At the point in time where you run your build, the package manager (maven, NPM, PIP) will refer to the artifact repository to perform the dependency management for tracking transitive dependencies.

Microservices:

As we move away from monolithic builds, these solutions need to scale build hundreds of containers where container specific dependencies will be resolved.

DevOps INSTITUTE

Changes in SCM

Software Configuration Management (SCM)

Monolithic:

The build process is the central tool for performing configuration management. The build performs configuration management by pulling code from version control based on a ‘trunk’ or ‘branch. A Software Bill of Material can be created to show all artifacts that were used to create the application.

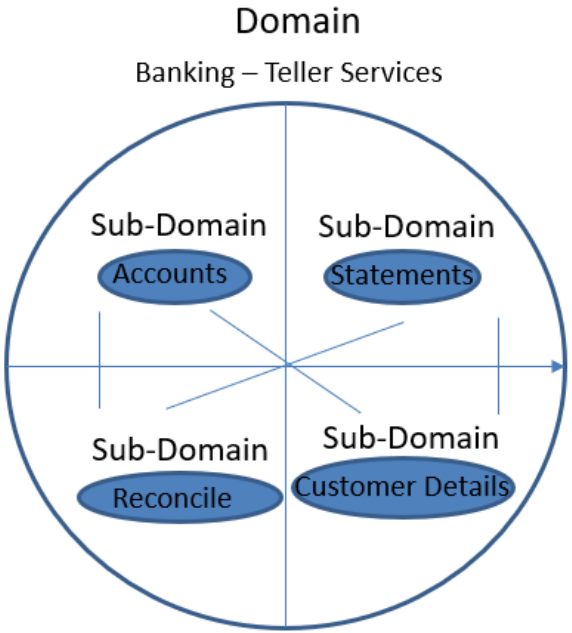
Microservices:

‘Builds’ as we know them go away in a microservice pipeline. For the most part, the version and build configuration shifts to runtime with microservices. While the container image has a configuration, the broader picture of the configuration happens at run-time in the cluster via the APIs. New tools will enter the market to track configurations of microservices.

Changes in the Use of Domains

Domain Driven Design
Monolithic:
Does not exist.

Microservices:
SCM will begin to bring in the concept of Domain Driven Design where you are managing an architecture based on the microservice 'problem space.' New tooling will enter the market to help with managing your Domains, your logical view of your application and to track versions of applications to versions of services. In general, SCM will become more challenging as we move away from resolving all dependencies at the compile/link step and must track more of it across the pipeline.



DevOps INSTITUTE

Changes in Security

DevSecOps

Monolithic:
Security solutions allow you to define or follow a specific set of standards. They include code scanning, container scanning and monitoring.

Microservices:
Security solutions will shift further ‘left’ adding more scanning around the creation of containers. As containers are deployed, security tools will begin to focus on preventing known security vulnerabilities and open source licensing violations from making it to production.

Conclusion

Modern Architecture is disrupting business as usual. A new way of running software has emerged.

This new modern architecture directly impacts the way we write and deliver software solutions.

Skilling up now is more important than ever.