

# Creating an Effective Developer Experience on Kubernetes

---

Daniel Bryant  
@danielbryantuk | @datawireio

# tl;dr

The developer experience is primarily about minimising the friction from idea to code to delivering observable business value

How you construct your 'platform' impacts the developer experience greatly

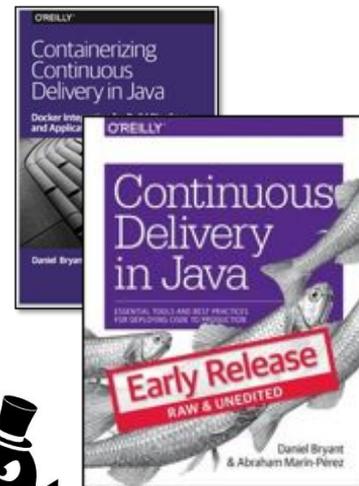
High productivity (and fun) comes from intentionally designing experience of: local development, packaging apps, CI/CD, deployment control, and observability

# @danielbryantuk

Independent Technical Consultant, Product Architect at Datawire

Previously: Academic, software developer (from startups to gov), consultant, CTO, trainer, conference tourist...

*Leading change through technology and teams*



# DevEx 101

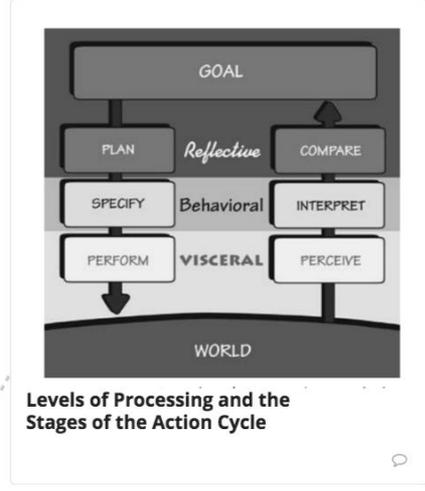
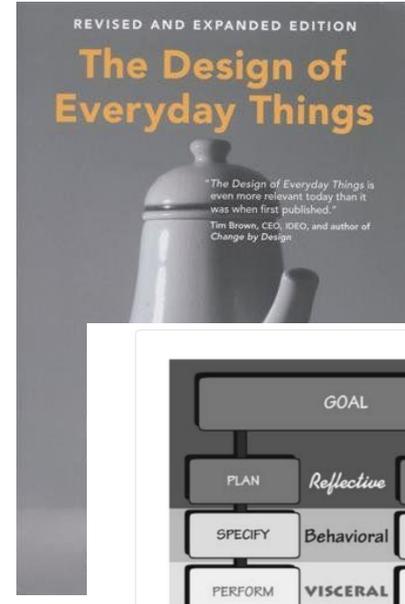
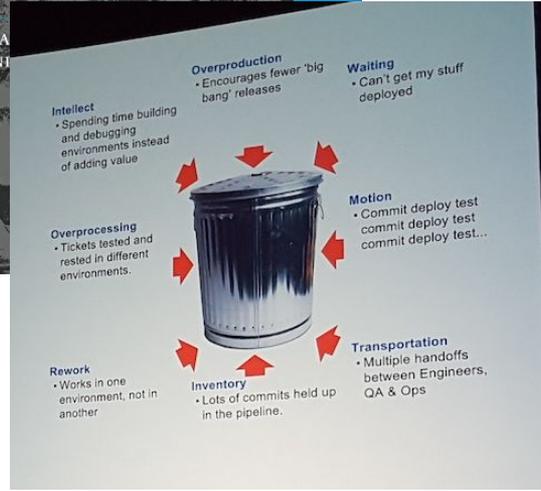
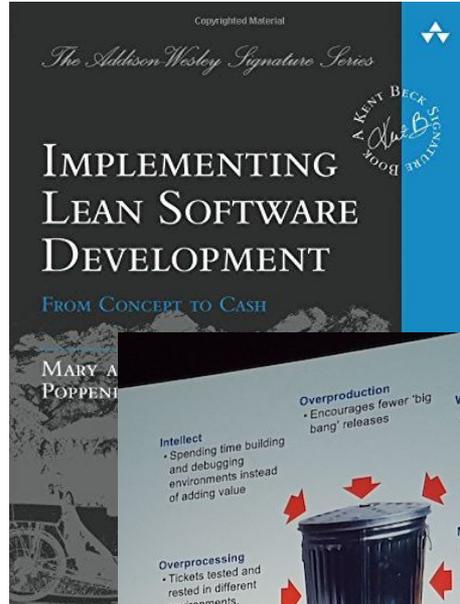
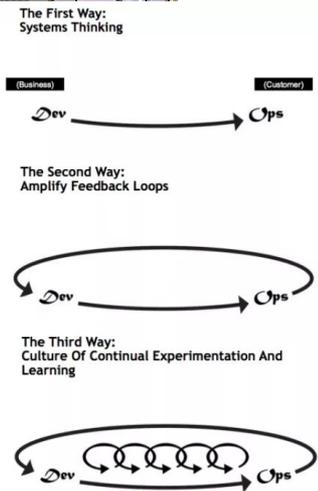
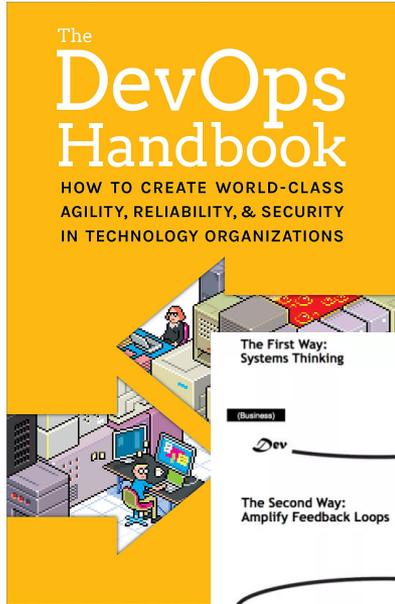
# DevEx...

...reducing engineering friction between creating a hypothesis to delivering an observable experiment (or value) in production

- Adrian Trenaman (SVP Engineering, HBC)

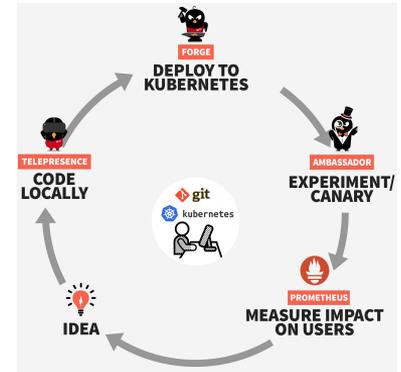
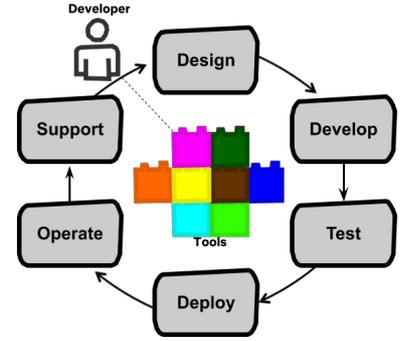
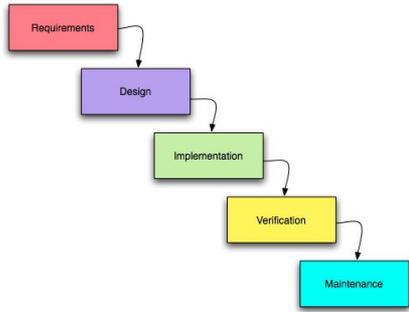
<https://www.infoq.com/news/2017/07/remove-friction-dev-ex>

# DevEx: DevOps, Lean, and UX

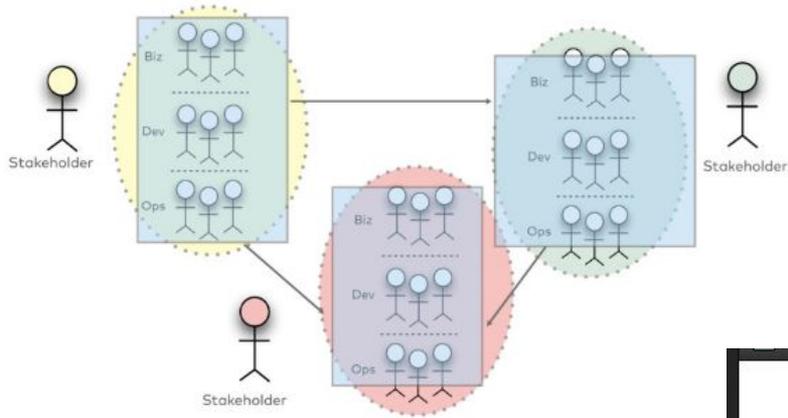


# **DevEx and Workflow**

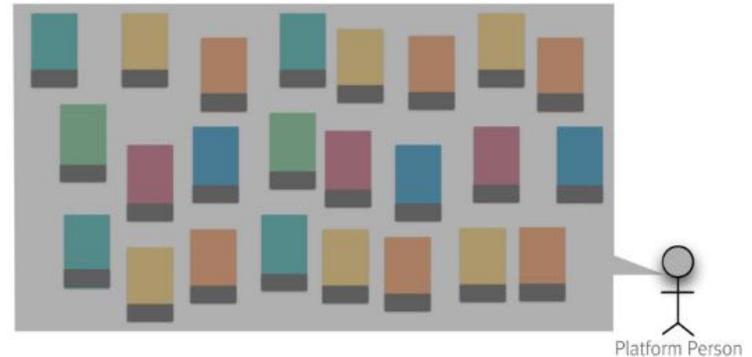
# The Ideal Workflow



# Pattern: Autonomous Cells



# Antipattern: Micro Platform



<https://speakerdeck.com/stilkov/microservices-patterns-and-antipatterns-1>

**The Platform  
Drives DevEx**

**Kelsey Hightower**  @kelseyhightower Following

I'm convinced the majority of people managing infrastructure just want a PaaS. The only requirement: it has to be built by them.

12:08 AM - 12 Apr 2017

340 Retweets 727 Likes

55 340 727

**Kelsey Hightower**  @kelseyhightower · 12 Apr 2017  
Replying to [@kelseyhightower](#)  
You know you're building a PaaS when you wrap your current tools with a custom API that provides a workflow to your users.

1 14 45

**Kelsey Hightower**  @kelseyhightower · 12 Apr 2017  
You know you're building a PaaS when you start work on that custom templating engine for deployments and configuration files.

2 15 49

**Kelsey Hightower**  @kelseyhightower · 12 Apr 2017  
You know you're building a PaaS when you start stitching together 1,000,000,000 other tools in order to get one click deployments.

7 46 107

**Kelsey Hightower**  @kelseyhightower · 12 Apr 2017  
Nothing wrong with building a PaaS; just know that's what you're doing.

8 17 61

<https://twitter.com/kelseyhightower/status/851935087532945409>

## The "Paved Road" PaaS for Microservices at Netflix: Yunong Xiao at QCon NY

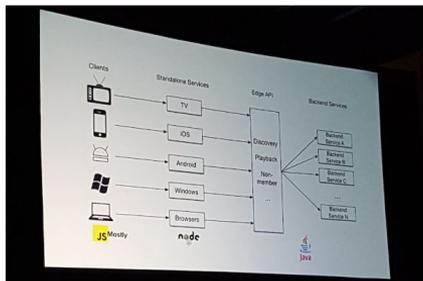
Like  | by Daniel Bryant on Jun 30, 2017. Estimated reading time: 3 minutes | [Discuss](#) NOTICE: The next QCon is in San Francisco Nov 5 - 9, 2018. Join us!

Share  [Reading List](#) [Read later](#)

At QCon New York 2017, Yunong Xiao presented "The Paved PaaS to Microservices at Netflix" which discussed how the Netflix Platform as a Service (PaaS) assists with maintaining the balance between the culture of freedom and responsibility and the overall organisational goals of velocity and reliability. The Netflix PaaS team attempts to provide a sensibly configured but customisable "paved road" platform for developers by offering standardised and compatible components, pre-assembling the platform, and by providing extensive automation and tooling.

Xiao, Principal Software Engineer at Netflix, began the talk by referencing the Wikipedia definition of PaaS "...allows customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure and platform". Within the Netflix technical stack, functionality provided by the PaaS includes microservice Remote Procedure Calling (RPC), service discovery and registration, operating system, application runtime, configuration, metrics, logging, tracing, dashboards, alerts and stream processing.

At Netflix the backend services are typically deployed onto a Java/JVM runtime which is fronted by an Edge API, but client teams own standalone services that they create in order to meet the needs of their associated end-user delivery technologies like smart TVs, iOS and MS Windows. These services are typically developed using JavaScript and Node.js, and the delivery teams are not necessarily familiar with backend operations and platforms. Netflix famously embraces a culture of freedom and responsibility ("F&R"), and this must be balanced with the overall organisational goals of velocity and reliability. Functionality provided by a PaaS can help with this balance, and this is implemented in three main ways in order to provide a homogenised but configurable "paved road" for developers, including the provision of standardised components, pre-assembled platform, and automation and tooling.



<https://www.infoq.com/news/2017/06/paved-paas-netflix>

### RELATED CONTENT

Shopify's Journey to Kubernetes and PaaS: Niko Kurtti at QCon NY Jul 01, 2018

Full Cycle Developers at Netflix: from Mindsets to Self-Service Tooling Jun 17, 2018

Incident Management at Netflix Velocity Apr 28, 2018

Automating Netflix ML Pipelines with Meson Jun 24, 2018

Designing Services for Resilience: Netflix Lessons Jun 18, 2018

AWS Config Gains Cross-Account, Cross-Region Data Aggregation Jul 01, 2018

Aman API Gateway Now Supports Private Endpoints Jun 29, 2018

OpsRamp Introduces an AIOps Inference Engine Jun 17, 2018

AppDynamics Launches New European Software-as-a-Service Offering Jun 15, 2018

## Shopify's Journey to Kubernetes and PaaS: Niko Kurtti at QCon NY

Like  | by Daniel Bryant on Jul 01, 2018. Estimated reading time: 6 minutes | [Discuss](#) NOTICE: The next QCon is in San Francisco Nov 5 - 9, 2018. Join us!

Share  [Reading List](#) [Read later](#)

At QCon New York, Niko Kurtti presented "Forced Evolution: Shopify's Journey to Kubernetes", and described the Shopify engineering team's journey to building their own PaaS with Kubernetes as the foundation. Key takeaways for other teams looking to build their own PaaS and associated developer workflow included: target hitting 80% of deployment and operational use cases, create patterns and hide the underlying platform complexity, educate and get people excited about the project; and be conscious of vendor lock-in.

Kurtti, production engineer at Shopify, began the talk by describing that Shopify is a rapidly growing Canadian e-commerce company that offers a proprietary e-commerce platform for online stores and retail point-of-sale systems. Shopify currently has 3000+ employees, and the company processed \$26 billion in transactions in 2017. The underlying e-commerce software platform sees 80k+ requests per second during peak demand.

At the start of 2016 the engineering team was "running services everywhere", including within their own data centers (using Chef and Docker), on AWS (using Chef) and Heroku. Developers liked the developer experience of Heroku, and Kurtti commented that this platform actually scales quite well, with "simple UI sliders" to increase the number of instances and associated CPU and RAM. Although the platform team had defined service tiers and appropriate Service Level Objectives (SLOs) based on criticality to the business, there were many processes that were not scalable, and accordingly these presented challenges as the company grew.



<https://www.infoq.com/news/2018/07/shopify-kubernetes-paas>

### RELATED CONTENT

GPUs on Google's Kubernetes Engine Are Now Generally Available Jul 07, 2018

Q&A with Gabe Monroy of Microsoft on Azure Kubernetes Service from Build 2018 Jun 20, 2018

Azure Kubernetes Service (AKS) is Now Generally Available - More Regions and New Features Jun 20, 2018

Kubernetes Package Manager Helm Now Hosted by the CNCF Jun 19, 2018

AWS Releases Elastic Container Service for Kubernetes (EKS) Jun 13, 2018

Kubernetes for the Spring Developer May 18, 2018

Deploying Spring Boot Apps on Kubernetes May 13, 2018

Kubernetes: Crossing the Chasm Apr 05, 2018

Containers, Kubernetes and Google Cloud Mar 11, 2018

Six Tips for Running Scalable Workloads on Kubernetes Mar 30, 2018

Kubernetes Superpower Feb 28, 2018

**Should I Build a  
PaaS on k8s?**

# Key Questions to Ask...

# Develop and test services locally, or within the cluster (or both)?

- Working locally has many advantages
  - Reduce ops cost of multi-cluster
- Some want to maintain minimal local development envs
  - Or hide Docker/k8s from devs
- Local/remote container dev tools like Telepresence and Squash allow hybrid

## Development Environments for Kubernetes

	100% local development	100% remote development		
	Run entire system locally	Run business logic locally, cloud resources remote	Single service local, all other services remote	All remote development
Realism: How closely does this mirror production?	1 hexagon	2 hexagons	3 hexagons	4 hexagons
Fast feedback cycle for developers	4 hexagons	3 hexagons	2 hexagons	1 hexagon
Low setup and maintenance cost for developers	1 hexagon	2 hexagons	3 hexagons	4 hexagons
Scalability as your application gets more complex	1 hexagon	2 hexagons	3 hexagons	4 hexagons
	docker			kubernetes

Yunong Xiao

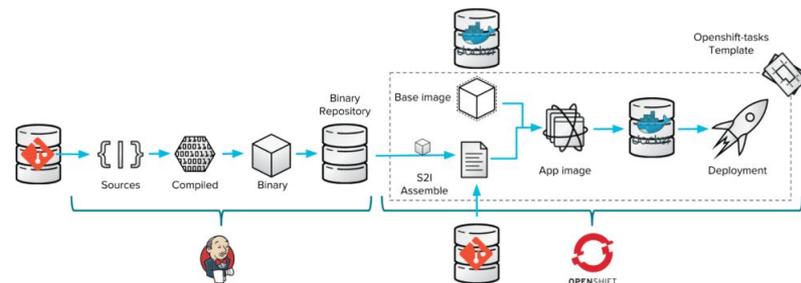
# How quick do you need user feedback?

- Canary testing is very powerful
  - As is developing in prod and shadowing
  - K8s tools like Istio & Ambassador enable this
- Needs app and platform support
- Some teams can be nervous about testing in production (quite rightly!)

The screenshot shows a blog post from 'THE NETFLIX TECH BLOG'. The main title is 'Automated Canary Analysis at Netflix with Kayenta' by Michael Coffey and Chris Sinden. The post discusses the use of Kayenta for Automated Canary Analysis (ACA) at Netflix, highlighting its role in reducing risk and increasing the degree of trust in deployments. It mentions that Kayenta is a platform for ACA that leverages lessons learned from years of delivering rapid and reliable changes into production. The post also notes that Kayenta has increased the degree of trust in deployments and is used by dozens of cloud native organizations. A line graph titled 'Looking at results' shows 'Conversion Rate (%)' over 'Time' for a 'Control' group (blue line) and a '+1.5%' group (red line). The red line shows a significant dip and then a recovery, while the blue line remains relatively flat. Below the graph, there are bullet points: 'Self Selection', 'Refunds / Return', and 'Visit-level vs. Us'. The post also mentions 'Code Faster Guides' and 'Datawire open source tools help you code faster on Kubernetes'. A diagram on the right side of the page shows a flow from 'Notice' to 'Canary Analysis'.

# Do you want to implement “guide rails” for your development teams?

- Larger teams often want to provide comprehensive guide rails
- Startups and SMEs may instead value team independence
  - #YOLO



<https://blog.openshift.com/multiple-deployment-methods-openshift/>

- Hybrid? Offer platform, but allow service teams freedom and responsibility

# **Workflow Tooling and Techniques**

# Pattern: K8s as a Foundation

- Kubernetes becoming de facto CoaaS (the new cloud broker?)
  - Lots of hosted options

- Highly extensible
  - Custom Controllers
  - Operators
  - CloudBuddies

- Extension enables custom workflow
  - “Kubernetes Custom Resource, Controller and Operator Development Tools”



## Overview

An Operator is a method of packaging, deploying and managing a Kubernetes application. A Kubernetes application is an application that is both deployed on Kubernetes and managed using the Kubernetes APIs and subject tooling. To be able to make the most of Kubernetes, you need a set of cohesive APIs to extend in order to service and manage your applications that run on Kubernetes. You can think of Operators as the runtime that manages this type of application on Kubernetes.

## The Operator Framework

The Operator Framework is an open source project that provides developer and runtime Kubernetes tools, enabling you to accelerate the development of an Operator. The Operator Framework includes:



### OPERATOR SDK

Enables developers to build Operators based on their expertise without requiring knowledge of Kubernetes API complexities.



### OPERATOR LIFECYCLE MANAGER

Oversees installation, updates, and management of the lifecycle of all of the Operators (and their associated services) running across a Kubernetes cluster.



# DIY K8s? What About Vendor Lock-in?

*“Engineers go to great lengths to avoid vendor lock-in. The irony is that in doing so, they often become their own vendor... with just as troublesome behaviour and rules”*

- Paraphrasing Adrian Cockcroft

The screenshot shows the Battery Ventures website. The header includes the BV logo and navigation links: About, Companies, Team, Venture + Growth, and Private Equity. The main content area features the article title "Who Doesn't Like Lock-In?" by Adrian Cockcroft, dated April 22, 2016. The article includes a photograph of a bride and groom figurine on a bed of white petals. Below the photo, the text reads: "That was my opening line at an OpenStack technology summit presentation I delivered last year. Many people raised their hands after I asked the question. I followed up by asking, 'How many of you are also married?' The remaining people looked uncomfortable, so of course I made a joke about the Ashley Madison website, which facilitates extra-marital affairs. (It was funny at the time, when the site had just been hacked and cheaters were being publicly exposed.) Like a marriage, enterprise IT lock-in—or using only one vendor's product for key functions, such as enterprise cloud computing—is a good thing if managed well for mutual benefit. But like a divorce, it gets ugly and expensive when things go wrong." The article is categorized under "Trends mitigating lock-in". On the right side of the page, there are sections for "Search", "Venture + Growth Capital", "Battery Ventures", and "Categories".

# Pattern: Automate Inner Dev Loop

- [Draft](#)
  - Automates “inner loop” build-push-deploy
  - Utilises Helm
- [Gitkube](#)
  - Automates build-push-deploy
  - Provides heroku / CF like experience
- [Skaffold](#)
  - Automates build-push-deploy
  - Watches source code
  - Provides “dev” and “run” (CD) modes
- [Telepresence](#) (\*)
  - Enables local-to-prod development
- [Helm](#) (\*)
  - Package manager for k8s
  - Deploy and manage (ready-made) charts
- [Ksonnet](#)
  - Define k8s manifests in jsonnet
  - Create composable config/services
- [Metaparticle](#)
  - “Standard library for cloud native apps”
  - Language-specific binding
- [Ballerina](#)
  - “Microservice programming language”
  - Annotations for package and deploy

(\*) CNCF projects

# Pattern: Automate Inner Dev Loop



Shahid K. Muhammed [Follow](#)  
Design Engineer by training, Polyglot by passion, @HasuraHQ by choice, Kubernetes by chance!  
Mar 29 · 12 min read

## Draft vs Gitkube vs Helm vs Ksonnet vs Metaparticle vs Skaffold

A comparison of tools that help developers build and deploy their apps on Kubernetes

### TL;DR

- **Draft**
  - deploy code to k8s cluster (automates build-push-deploy)
  - deploy code in **draft-pack supported** languages without writing dockerfile or k8s manifests
  - needs draft cli, helm cli, tiller on cluster, local docker, docker registry
- **Gitkube**
  - deploy code to k8s cluster (automates build-push-deploy)
  - git push to deploy, no dependencies on your local machine
  - needs dockerfile, k8s manifests in the git repo, gitkube on cluster
- **Helm**
  - deploy and manage charts (collection of k8s objects defining an application) on a k8s cluster
  - ready made charts for many common applications, like mysql, mediawiki etc.
  - needs helm cli, tiller on cluster, chart definition locally or from a repo

<https://blog.hasura.io/draft-vs-gitkube-vs-helm-vs-ksonnet-vs-metaparticle-vs-skaffold-f5aa9561f948>

## Code Engineered

[Blog](#) [Resources](#)

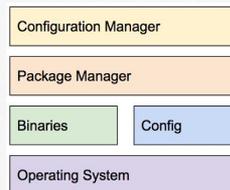
### Kubernetes: Where Helm And Related Tools Sit

Package management, dependency management, configuration management, and who knows how many other forms of management exist when it comes to computing systems. We have managers for managers for operators of applications. The roles and responsibilities of different tools can, at times, get a little blurred. I sometimes find that's the case with **Helm**. Is it a configuration management tool like Chef or a package manager like apt? This even begs the question, how do configuration managers, like Puppet, and package managers, like yum, relate to each other and what does any of this mean for Helm and Kubernetes?

To understand Helm ends helps to understand where other tools begin and the interfaces they have with Helm or Helm has with them.

### Parts Of The Management Stack

Before we look at Helm, specifically, let's take a look at different parts of a managed stack. This stack is based on a generality of how existing systems work.



Conceptual stack elements

The Code Engineered is the technical blog of [Matt Farina](#). This is where I muse about the cloud, web, developer tools, and more.

[Follow @mattfarina](#)



[Buy on Amazon](#)

A cookbook of 70 Go techniques.

### Tags

[Linux](#) [K8s](#) [Programming](#)  
[Front End Development](#)  
[JavaScript](#) [jQuery](#)  
[Web Tools](#)  
[Backend Development](#) [Tools](#)  
[Cloud](#) [DevOps](#)  
[Strategy](#) [Workflows](#)

<https://codeengineered.com/blog/2018/kubernetes-helm-related-tools/>

# Pattern: Envoy for Managing L7 Traffic

- Allows fine-grained deploy/release

- Enables real-time development in production (with shadowing)

- Many control planes (for Envoy)

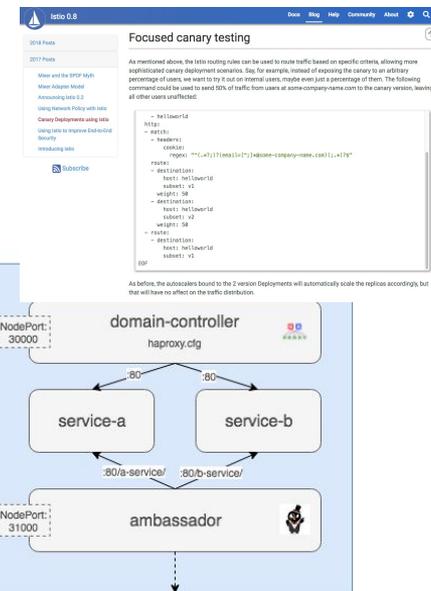
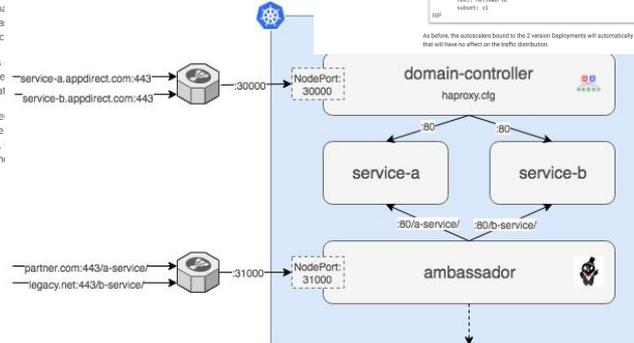
- [Ambassador](#)
- [Gloo](#)
- [Istio](#)



At AppDirect, we embraced Kubernetes—an open-source system for automating deployment, scaling, and many days of the project. We ran sa applications and served traffic

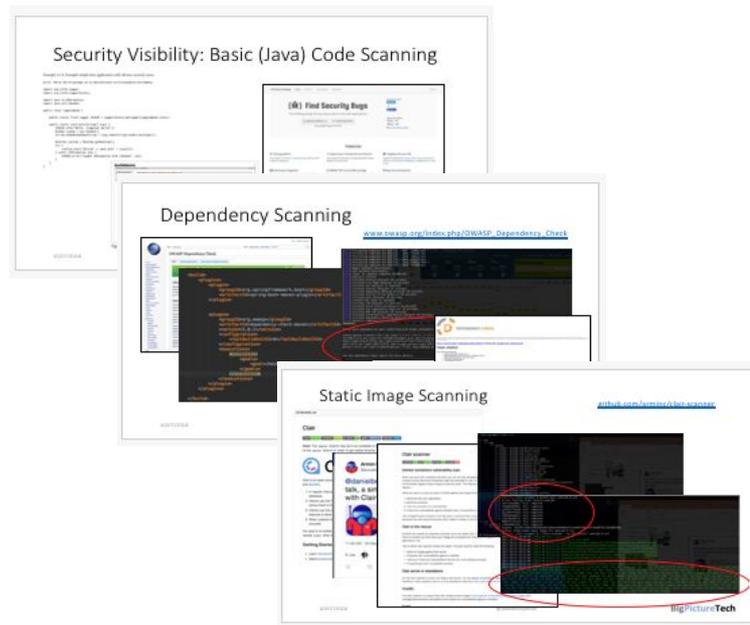
In October of 2016, members Montreal Meetup, and one que heck do you handle and dispat

Back then it seemed there wev years since the meetup we've many different environments, environments, cloud based an question has evolved.



# Pattern: CI/CD Enforces Policy

- Make it easy to do the right thing
  - Self-service pipeline creations
  - Bake-in hooks/slots for platform functionality
- Testing of NFRs is vital
  - Security, Performance, Quality
- Post-pipeline: Run chaos tests to codify properties and assert in production



[https://www.slideshare.net/dbryant\\_uk/codemotion-rome-2018-continuous-delivery-with-containers-the-good-the-bad-and-the-ugly](https://www.slideshare.net/dbryant_uk/codemotion-rome-2018-continuous-delivery-with-containers-the-good-the-bad-and-the-ugly)

# Pattern: Observability > Testing

- Essential part of the platform and developer workflow/experience
  - Monitoring, logging and tracing
  - Bake-in hooks to scaffolding

- Global/service dashboards

- [“Observability and Avoiding Alert Overload from Microservices at the Financial Times”](#)



Elasticsearch



Fluentd



Kibana

**Conclusion**

# In Summary

The developer experience is primarily about minimising the friction from idea to code to delivering observable business value

How you construct your 'platform' impacts the developer experience greatly

You must intentionally curate the experience of: local development, packaging apps, CI/CD, deployment control, and observability

# Thanks for Listening!

Questions, comments, thoughts...

[db@datawire.io](mailto:db@datawire.io)

[@danielbryantuk](https://twitter.com/danielbryantuk)

More info: [dzone.com/articles/creating-a-positive-developer-experience-for-conta](https://dzone.com/articles/creating-a-positive-developer-experience-for-conta)

[datawire.io/what-is-cloud-native](https://datawire.io/what-is-cloud-native) | [getambassador.io](https://getambassador.io) | [istio.io](https://istio.io) | [telepresence.io](https://telepresence.io),  
[prometheus.io](https://prometheus.io) | “[Kubernetes Up and Running](#)”