Creating an Effective Developer Experience on Kubernetes

Daniel Bryant
@danielbryantuk | @datawireio

"Developer Experience"
CMA CGM
APM TERMINALS

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

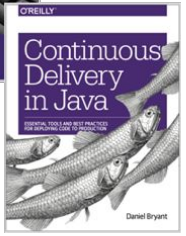
@danielbryantuk

Independent Technical Consultant, Product Architect at Datawire

Previously: Academic, software developer (from startups to gov), consultant, CTO, trainer…

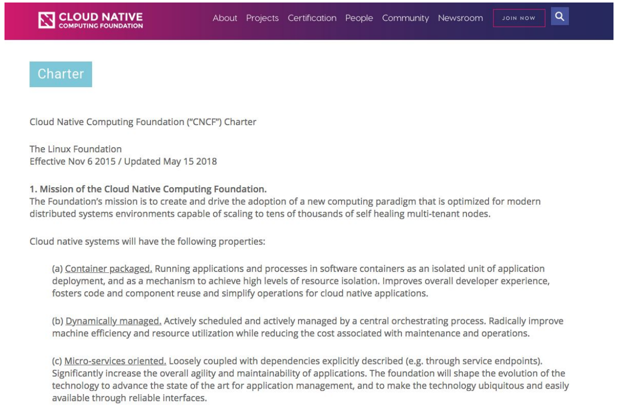
Leading change through technology and teams

O'REILLY
Containerizing Continuous Delivery in Java
Docker Integration for Build Pipelines and Application Architecture
Daniel Bryant



Setting the Scene

What is Cloud Native?



Charter

Cloud Native Computing Foundation ("CNCF") Charter

The Linux Foundation
Effective Nov 6 2015 / Updated May 15 2018

1. Mission of the Cloud Native Computing Foundation.
The Foundation's mission is to create and drive the adoption of a new computing paradigm that is optimized for modern distributed systems environments capable of scaling to tens of thousands of self healing multi-tenant nodes.

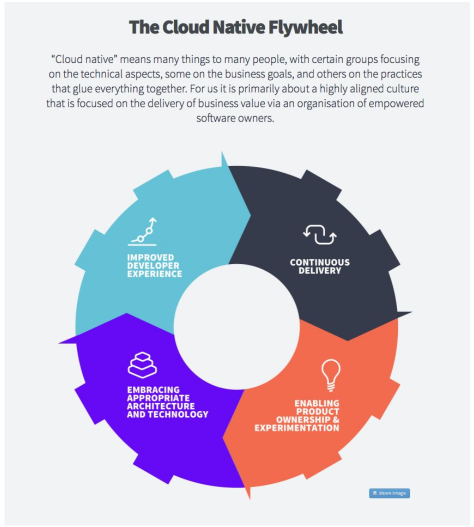
Cloud native systems will have the following properties:

(a) Container packaged. Running applications and processes in software containers as an isolated unit of application deployment, and as a mechanism to achieve high levels of resource isolation. Improves overall developer experience, fosters code and component reuse and simplify operations for cloud native applications.

(b) Dynamically managed. Actively scheduled and actively managed by a central orchestrating process. Radically improve machine efficiency and resource utilization while reducing the cost associated with maintenance and operations.

(c) Micro-services oriented. Loosely coupled with dependencies explicitly described (e.g. through service endpoints). Significantly increase the overall agility and maintainability of applications. The foundation will shape the evolution of the technology to advance the state of the art for application management, and to make the technology ubiquitous and easily available through reliable interfaces.

https://www.cncf.io/about/charter/

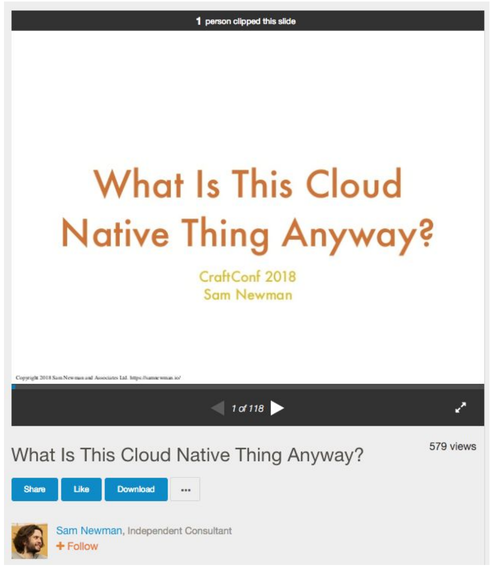


The Cloud Native Flywheel

"Cloud native" means many things to many people, with certain groups focusing on the technical aspects, some on the business goals, and others on the practices that glue everything together. For us it is primarily about a highly aligned culture that is focused on the delivery of business value via an organisation of empowered software owners.

IMPROVED DEVELOPER EXPERIENCE
CONTINUOUS DELIVERY
EMBRACING APPROPRIATE ARCHITECTURE AND TECHNOLOGY
ENABLING PRODUCT OWNERSHIP & EXPERIMENTATION
Share image

https://www.datawire.io/what-is-cloud-native/



What Is This Cloud Native Thing Anyway?

CraftConf 2018
Sam Newman

Copyright 2018 Sam Newman and Associates Ltd. https://samnewman.io/

What Is This Cloud Native Thing Anyway?

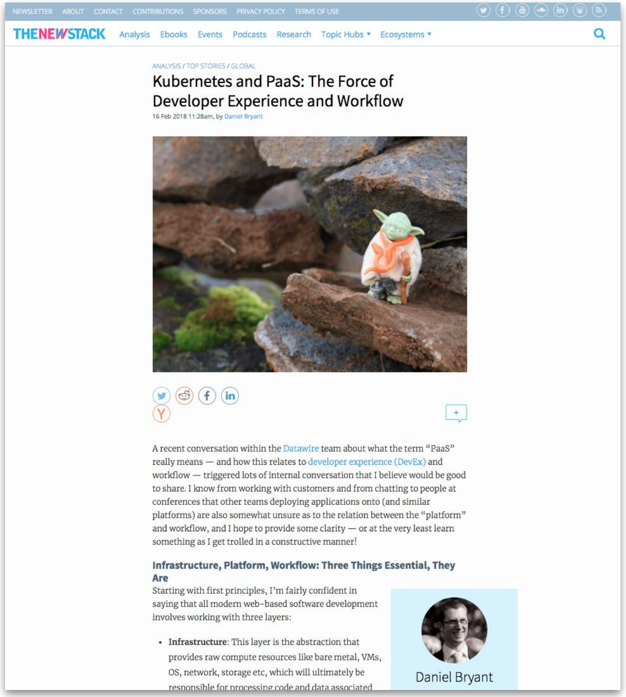
579 views

Sam Newman, Independent Consultant

https://www.slideshare.net/spnewman/what-is-this-cloud-native-thing-anyway

Infrastructure, Platforms, Workflow

- ❏ Infrastructure
  - ❏ Compute, network, IAM
- ❏ Platform
  - ❏ System building blocks
- ❏ Workflow
  - ❏ Design, build, test, deploy


NEWSLETTER ABOUT CONTACT CONTRIBUTIONS SPONSORS PRIVACY POLICY TERMS OF USE
THENEWSTACK Analysis Ebooks Events Podcasts Research Topic Hubs Ecosystems
ANALYSIS / TOP STORIES / GLOBAL

Kubernetes and PaaS: The Force of Developer Experience and Workflow

16 Feb 2018 11:28am, by Daniel Bryant

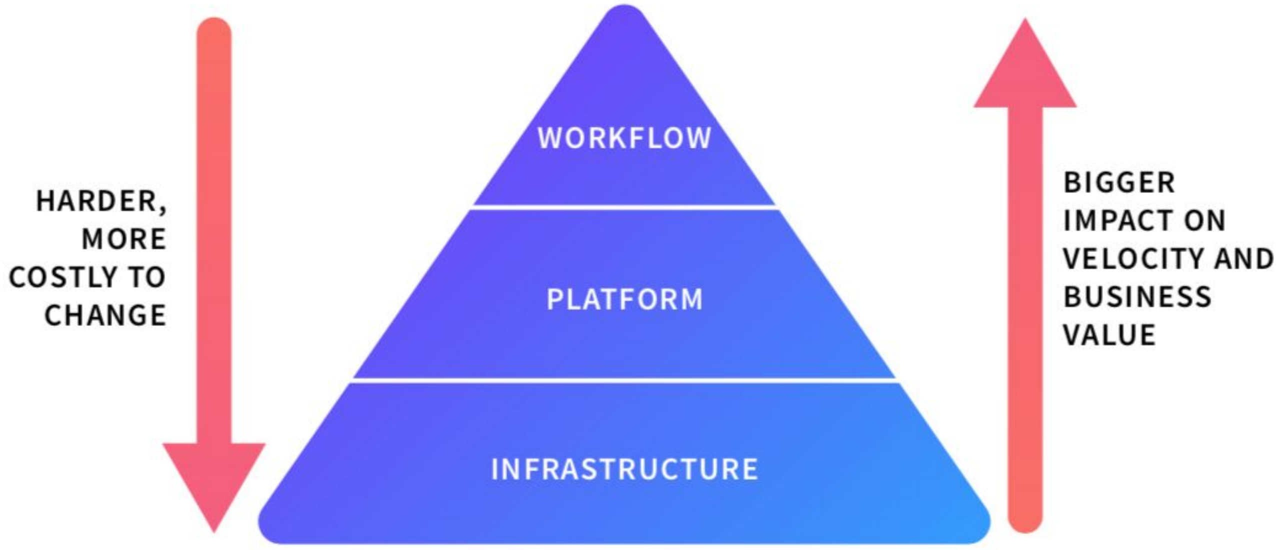
A recent conversation within the Datawire team about what the term "PaaS" really means — and how this relates to developer experience (DevEx) and workflow — triggered lots of internal conversation that I believe would be good to share. I know from working with customers and from chatting to people at conferences that other teams deploying applications onto (and similar platforms) are also somewhat unsure as to the relation between the "platform" and workflow, and I hope to provide some clarity — or at the very least learn something as I get trolled in a constructive manner!

Infrastructure, Platform, Workflow: Three Things Essential, They Are

Starting with first principles, I'm fairly confident in saying that all modern web-based software development involves working with three layers:

- **Infrastructure**: This layer is the abstraction that provides raw compute resources like bare metal, VMs, OS, network, storage etc, which will ultimately be responsible for processing code and data associated

Infrastructure, Platforms, Workflow


HARDER, MORE COSTLY TO CHANGE
WORKFLOW
PLATFORM
INFRASTRUCTURE
BIGGER IMPACT ON VELOCITY AND BUSINESS VALUE

https://dzone.com/articles/creating-a-positive-developer-experience-for-conta

Focusing on Platform and Workflow

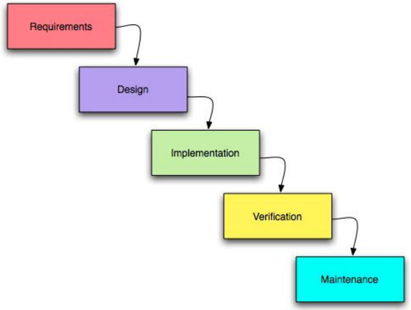
What is workflow?

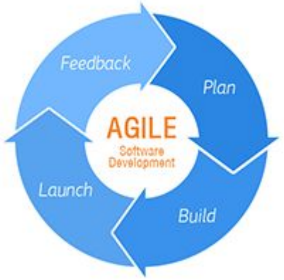
The platform heavily influences the workflow

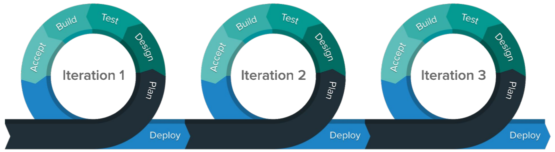
Diving deeper: patterns of good practice

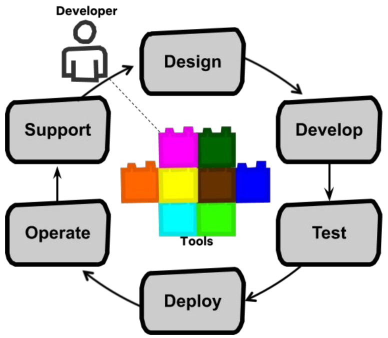
What is “Workflow”?

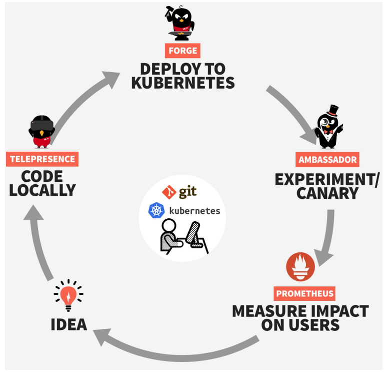
The Ideal Workflow

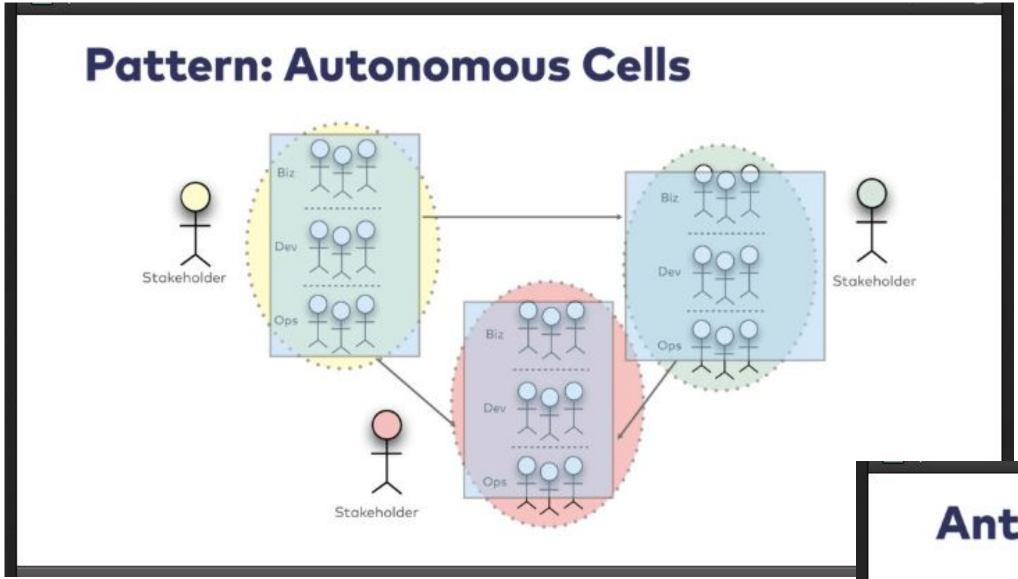




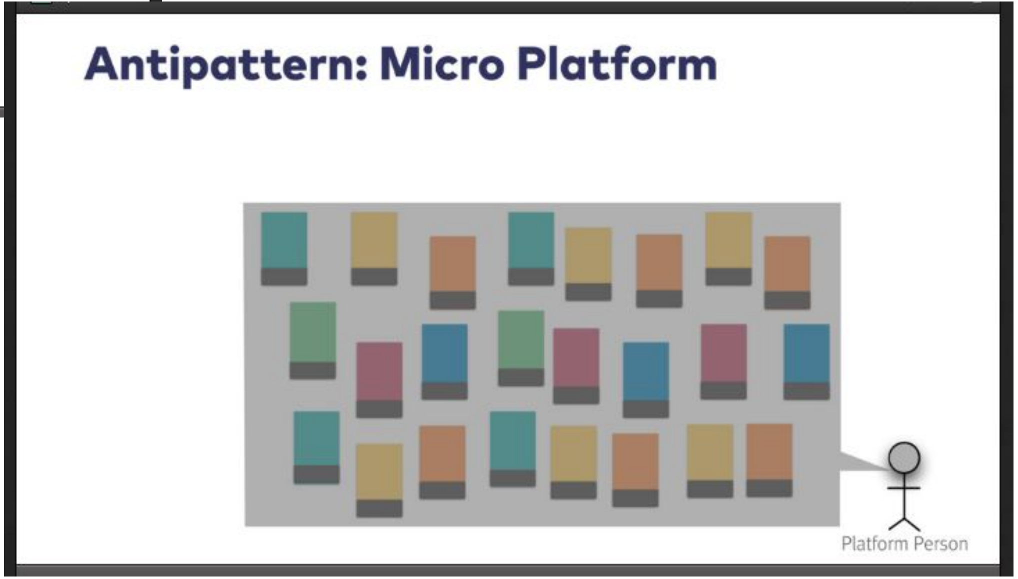








Pattern: Autonomous Cells



Antipattern: Micro Platform

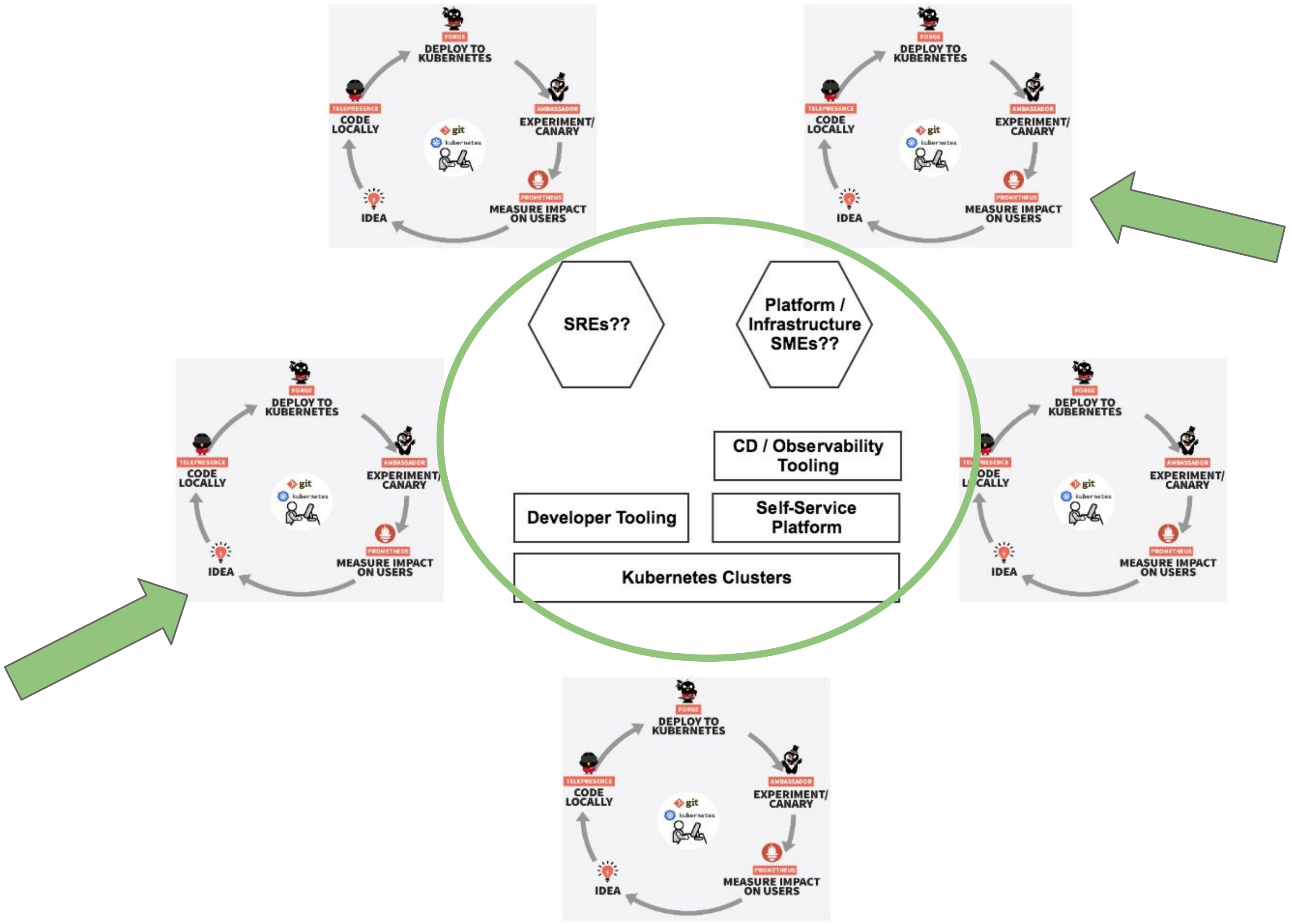
MicroXchg 2018, Berlin

Microservices: Patterns & Antipatterns

Stefan Tilkov
stefan.tilkov@innoq.com
@stilkov

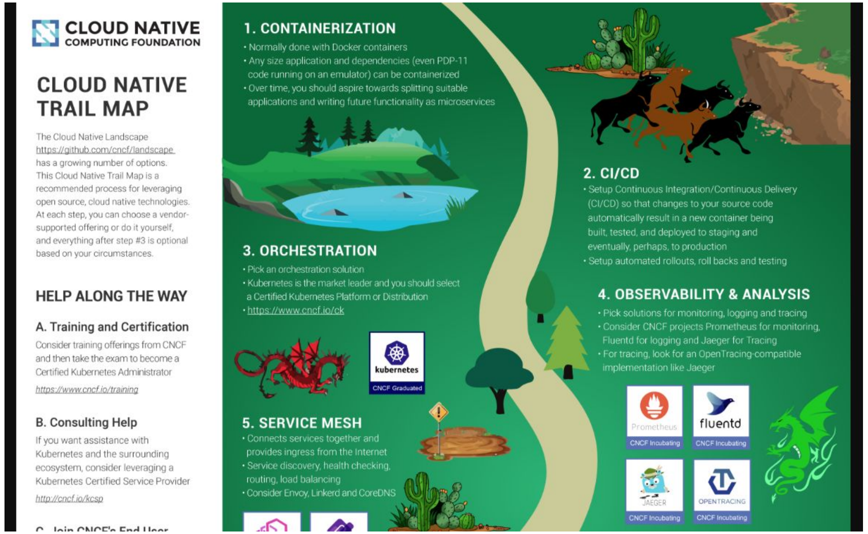
INNOQ

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


SREs??
Platform / Infrastructure SMEs??
CD / Observability Tooling
Developer Tooling
Self-Service Platform
Kubernetes Clusters
FORGE
DEPLOY TO KUBERNETES
AMBASSADOR
EXPERIMENT/ CANARY
PROMETHEUS
MEASURE IMPACT ON USERS
IDEA
TELEPRESENCE
CODE LOCALLY
git
kubernetes

The Platform Drives DevEx

CNCF Vision: Macro-Level Guidance



CLOUD NATIVE COMPUTING FOUNDATION
kubernetes
CNCF Graduated
Prometheus
CNCF Incubating
fluentd
CNCF Incubating
JAEGER
CNCF Incubating
OPENTRACING
CNCF Incubating

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape https://github.com/cncf/landscape has a growing number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator

https://www.cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider

http://cncf.io/kcsp

C. Join CNCF's End User

1. CONTAINERIZATION
- Normally done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

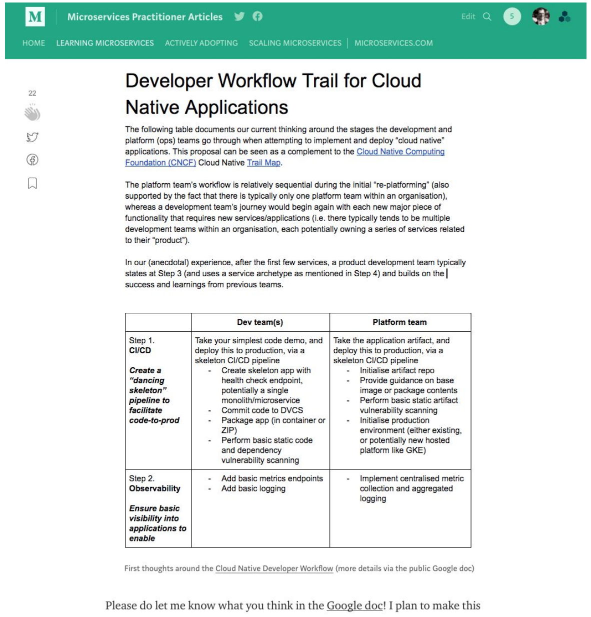
2. CI/CD
- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing

3. ORCHESTRATION
- Pick an orchestration solution
- Kubernetes is the market leader and you should select a Certified Kubernetes Platform or Distribution
- https://www.cncf.io/ck

4. OBSERVABILITY & ANALYSIS
- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger

5. SERVICE MESH
- Connects services together and provides ingress from the Internet
- Service discovery, health checking, routing, load balancing
- Consider Envoy, Linkerd and CoreDNS

https://github.com/cncf/landscape#trail-map



Developer Workflow Trail for Cloud Native Applications

The following table documents our current thinking around the stages the development and platform (ops) teams go through when attempting to implement and deploy "cloud native" applications. This proposal can be seen as a complement to the Cloud Native Computing Foundation (CNCF) Cloud Native Trail Map.

The platform team's workflow is relatively sequential during the initial "re-platforming" (also supported by the fact that there is typically only one platform team within an organisation), whereas a development team's journey would begin again with each new major piece of functionality that requires new services/applications (i.e. there typically tends to be multiple development teams within an organisation, each potentially owning a series of services related to their "product").

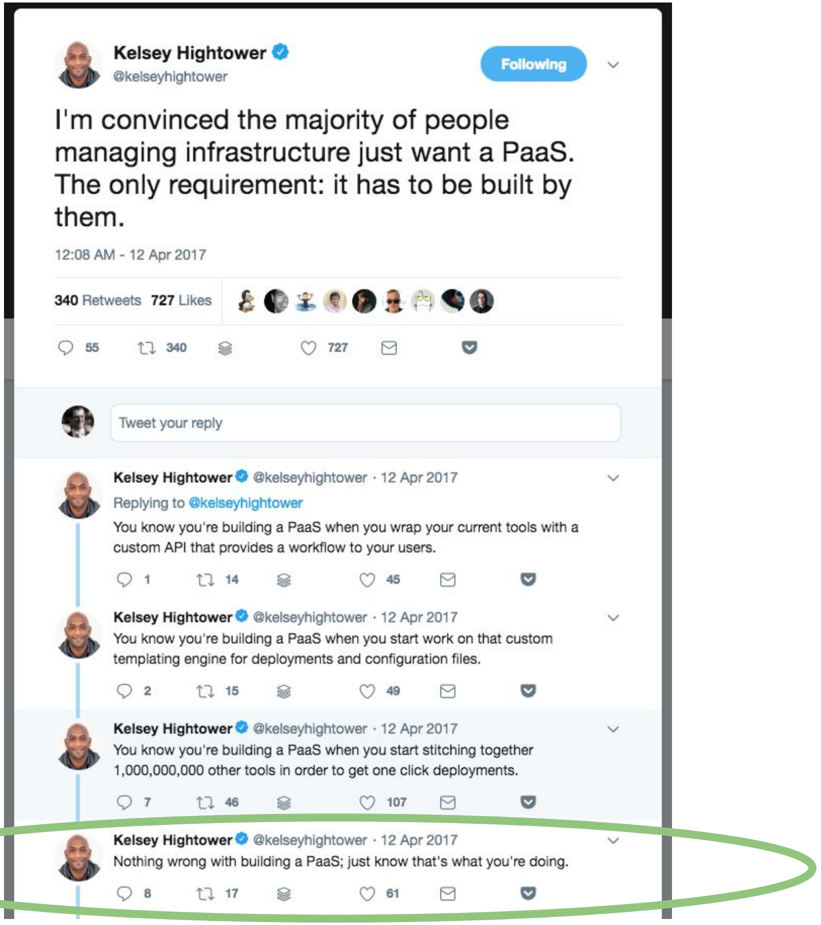
In our (anecdotal) experience, after the first few services, a product development team typically states at Step 3 (and uses a service archetype as mentioned in Step 4) and builds on the success and learnings from previous teams.

	Dev team(s)	**Platform team**
Step 1. **CI/CD** ***Create a "dancing skeleton" pipeline to facilitate code-to-prod***	Take your simplest code demo, and deploy this to production, via a skeleton CI/CD pipeline - Create skeleton app with health check endpoint, potentially a single monolith/microservice - Commit code to DVCS - Package app (in container or ZIP) - Perform basic static code and dependency vulnerability scanning	Take the application artifact, and deploy this to production, via a skeleton CI/CD pipeline - Initialise artifact repo - Provide guidance on base image or package contents - Perform basic static artifact vulnerability scanning - Initialise production environment (either existing, or potentially new hosted platform like GKE)
Step 2. **Observability** ***Ensure basic visibility into applications to enable***	- Add basic metrics endpoints - Add basic logging	- Implement centralised metric collection and aggregated logging

First thoughts around the Cloud Native Developer Workflow (more details via the public Google doc)

Please do let me know what you think in the Google doc! I plan to make this

https://articles.microservices.com/developer-workflow-trail-for-cloud-native-applications-request-for-feedback-a9365b64c790


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

Tweet your reply

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

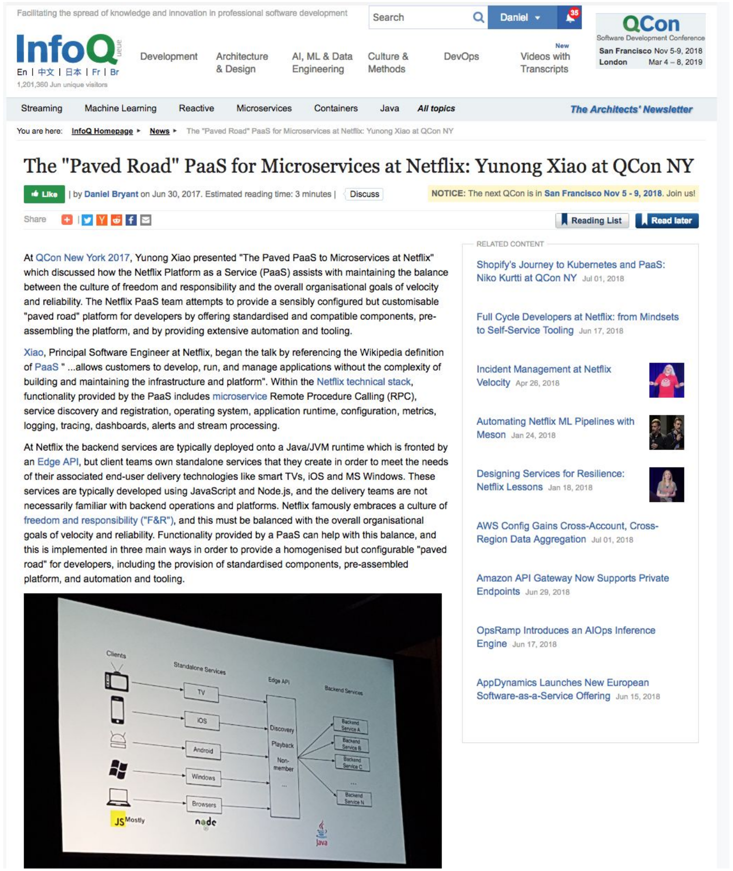
Kelsey Hightower @kelseyhightower · 12 Apr 2017
You know you're building a PaaS when you start stitching together 1,000,000,000 other tools in order to get one click deployments.

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



Facilitating the spread of knowledge and innovation in professional software development

You are here: InfoQ Homepage ▸ News ▸ The "Paved Road" PaaS for Microservices at Netflix: Yunong Xiao at QCon NY

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

| by Daniel Bryant on Jun 30, 2017. Estimated reading time: 3 minutes |

NOTICE: The next QCon is in San Francisco Nov 5 - 9, 2018. Join us!

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.

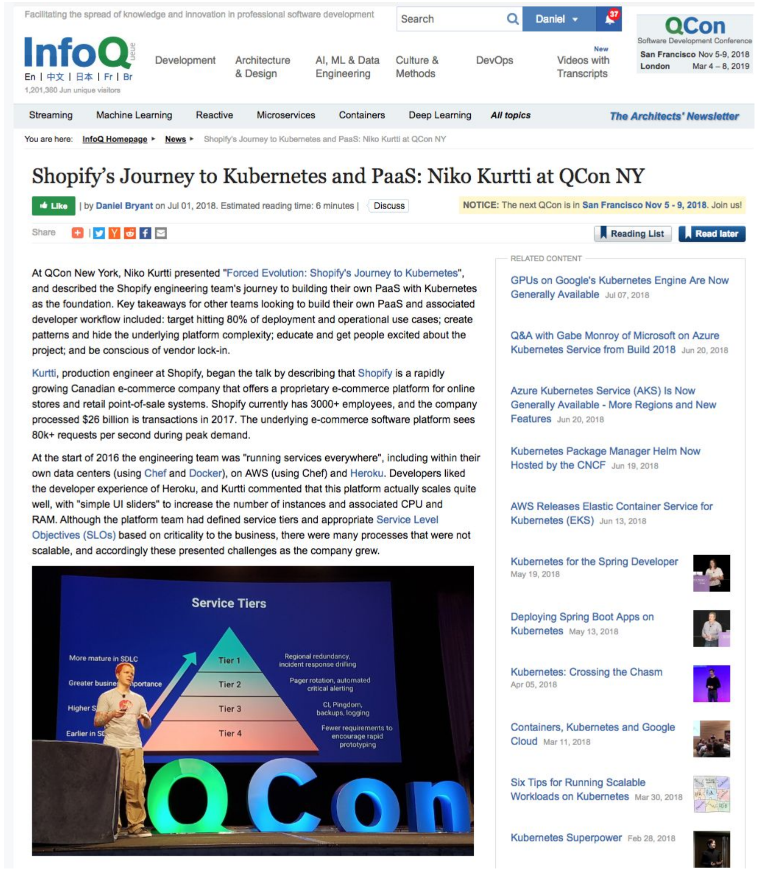
Xiao, Principal Software Engineer at Netflix, began the talk by referencing the Wikipedia definition of PaaS " ...allows customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure and platform". Within the Netflix technical stack, functionality 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.

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 26, 2018
- Automating Netflix ML Pipelines with Meson Jan 24, 2018
- Designing Services for Resilience: Netflix Lessons Jan 18, 2018
- AWS Config Gains Cross-Account, Cross-Region Data Aggregation Jul 01, 2018
- Amazon 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

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


Search
Daniel
QCon
Software Development Conference
San Francisco Nov 5-9, 2018
London Mar 4 – 8, 2019
En | 中文 | 日本 | Fr | Br
1,201,360 Jun unique visitors
Development
Architecture & Design
AI, ML & Data Engineering
Culture & Methods
DevOps
New Videos with Transcripts
Streaming
Machine Learning
Reactive
Microservices
Containers
Deep Learning
All topics
The Architects' Newsletter
Like
Discuss
Share
Reading List
Read later
Service Tiers
More mature in SDLC
Tier 1
Tier 2
Tier 3
Tier 4
Regional redundancy, incident response drilling
Pager rotation, automated critical alerting
CI, Pingdom, backups, logging
Fewer requirements to encourage rapid prototyping
QCon

Facilitating the spread of knowledge and innovation in professional software development

You are here: InfoQ Homepage ▸ News ▸ Shopify's Journey to Kubernetes and PaaS: Niko Kurtti at QCon NY

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

| by Daniel Bryant on Jul 01, 2018. Estimated reading time: 6 minutes |

NOTICE: The next QCon is in San Francisco Nov 5 - 9, 2018. Join us!

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 is 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.

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 19, 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

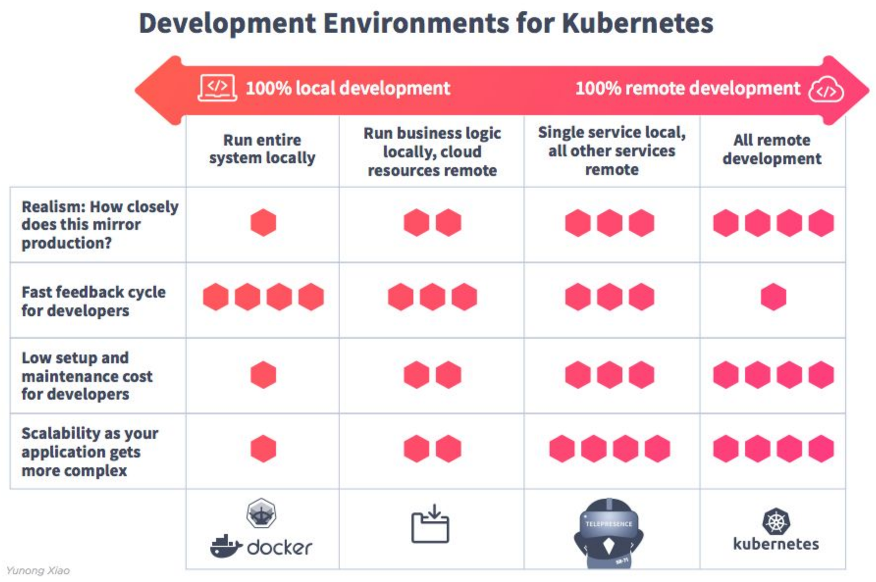
https://www.infoq.com/news/2018/07/shopify-kubernetes-paas

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 dev envs
  - Or hide Docker/k8s from devs
- Local/remote container dev tools like Telepresence and Squash allow hybrid



Development Environments for Kubernetes

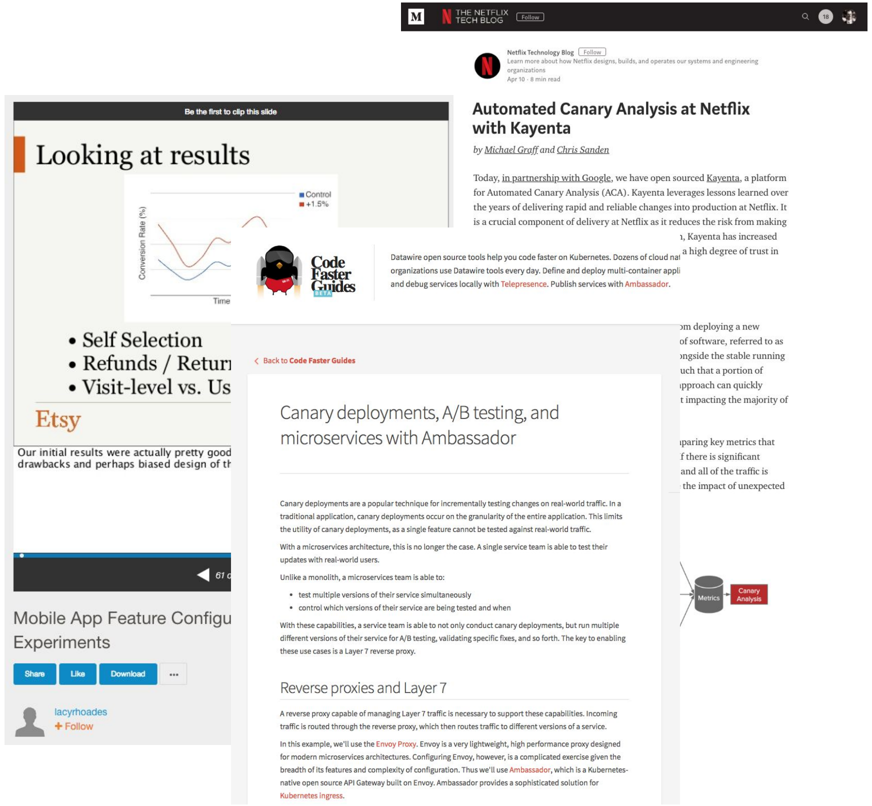
100% local development ⟷ 100% remote development

	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?	⬢	⬢⬢	⬢⬢⬢	⬢⬢⬢⬢
Fast feedback cycle for developers	⬢⬢⬢⬢	⬢⬢⬢	⬢⬢⬢	⬢
Low setup and maintenance cost for developers	⬢	⬢⬢	⬢⬢⬢	⬢⬢⬢⬢
Scalability as your application gets more complex	⬢	⬢⬢	⬢⬢⬢⬢	⬢⬢⬢⬢
	docker		TELEPRESENCE	kubernetes

Yunong Xiao

How quick do you need user feedback?

- Canary testing is very powerful
- Needs app and platform support
- Some teams can be nervous about testing in production



Netflix Technology Blog
Learn more about how Netflix designs, builds, and operates our systems and engineering organizations
Apr 10 · 8 min read

Automated Canary Analysis at Netflix with Kayenta

by Michael Graff and Chris Sanden

Today, in partnership with Google, we have open sourced Kayenta, a platform for Automated Canary Analysis (ACA). Kayenta leverages lessons learned over the years of delivering rapid and reliable changes into production at Netflix. It is a crucial component of delivery at Netflix as it reduces the risk from making
ı, Kayenta has increased
a high degree of trust in

ɔm deploying a new
of software, referred to as
ɔngside the stable running
uch that a portion of
ıpproach can quickly
t impacting the majority of

ıparing key metrics that
f there is significant
and all of the traffic is
the impact of unexpected

Looking at results

- Self Selection
- Refunds / Returı
- Visit-level vs. Us

Etsy

Our initial results were actually pretty good
drawbacks and perhaps biased design of th

61 c

Mobile App Feature Configu
Experiments

Datawire open source tools help you code faster on Kubernetes. Dozens of cloud nat
organizations use Datawire tools every day. Define and deploy multi-container appli
and debug services locally with Telepresence. Publish services with Ambassador.

‹ Back to Code Faster Guides

Canary deployments, A/B testing, and microservices with Ambassador

Canary deployments are a popular technique for incrementally testing changes on real-world traffic. In a traditional application, canary deployments occur on the granularity of the entire application. This limits the utility of canary deployments, as a single feature cannot be tested against real-world traffic.

With a microservices architecture, this is no longer the case. A single service team is able to test their updates with real-world users.

Unlike a monolith, a microservices team is able to:

- test multiple versions of their service simultaneously
- control which versions of their service are being tested and when

With these capabilities, a service team is able to not only conduct canary deployments, but run multiple different versions of their service for A/B testing, validating specific fixes, and so forth. The key to enabling these use cases is a Layer 7 reverse proxy.

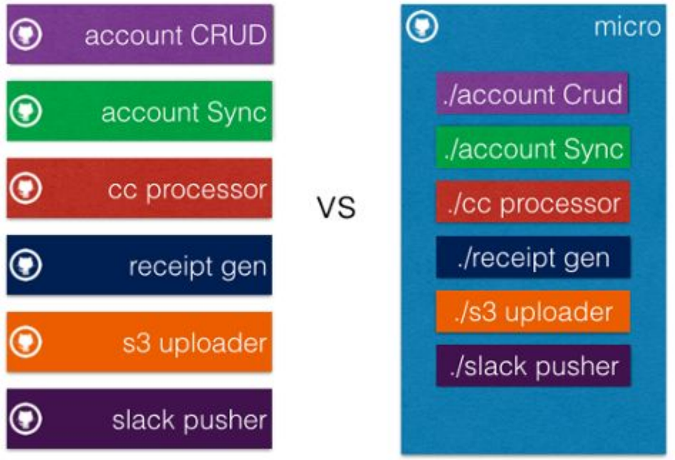
Reverse proxies and Layer 7

A reverse proxy capable of managing Layer 7 traffic is necessary to support these capabilities. Incoming traffic is routed through the reverse proxy, which then routes traffic to different versions of a service.

In this example, we'll use the Envoy Proxy. Envoy is a very lightweight, high performance proxy designed for modern microservices architectures. Configuring Envoy, however, is a complicated exercise given the breadth of its features and complexity of configuration. Thus we'll use Ambassador, which is a Kubernetes-native open source API Gateway built on Envoy. Ambassador provides a sophisticated solution for Kubernetes ingress.

Do you have a strong opinion on code repository structure?

- Monorepo:
  - Coordination of integration and testing across services is generally easier,
  - Service dependency management easier
- Multi-repo:
  - Clearer ownership
  - Can promote loose coupling
  - Refactoring and code-level standardization can be challenging



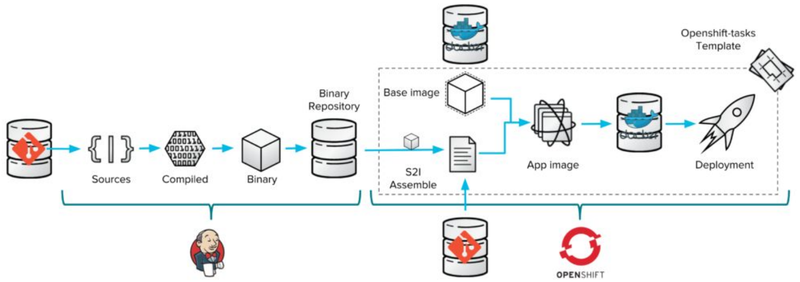
http://blog.shippable.com/our-journey-to-microservices-and-a-mono-repository

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

- Larger teams often want to provide comprehensive guide rails

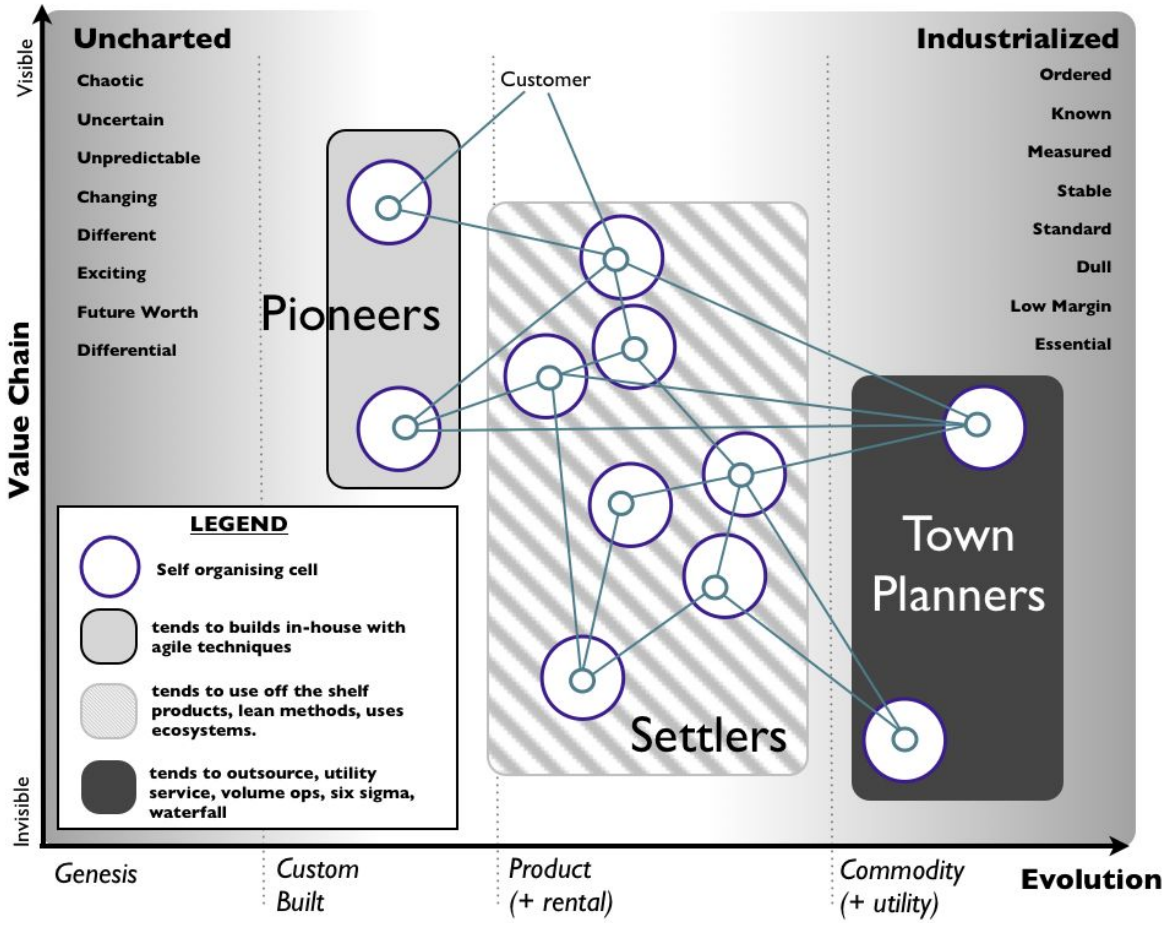
- Startups and SMEs may instead value team independence

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



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

How much platform should we build?



https://blog.gardeviance.org/2015/03/on-pioneers-settlers-town-planners-and.html

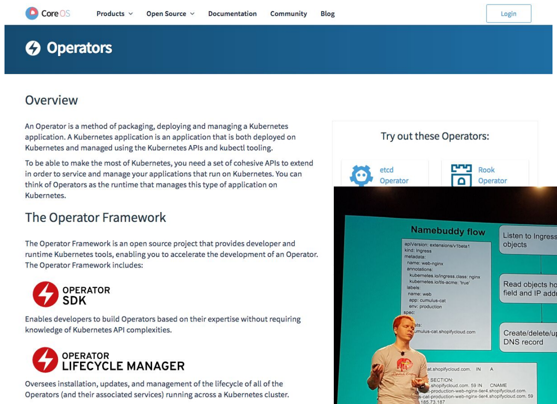
Some thoughts on this...

	Prototype	**Production**	**Mission Critical**
Dev and test	Local / hybrid	Local / hybrid	Hybrid / local
Deployment	Canary	Canary / pre-prod test	Pre-prod test / Canary
Code repo	Mono / multi	Multi / Mono	Multi
Guide rails	YOLO	Limited	Strong
Where to focus?	CI/CD	Scaffolding	Testing (env creation)

Workflow Tooling and Techniques

Pattern: K8s for Ops

- 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 kubectl 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.

Try out these Operators:

etcd Operator

Rook Operator

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

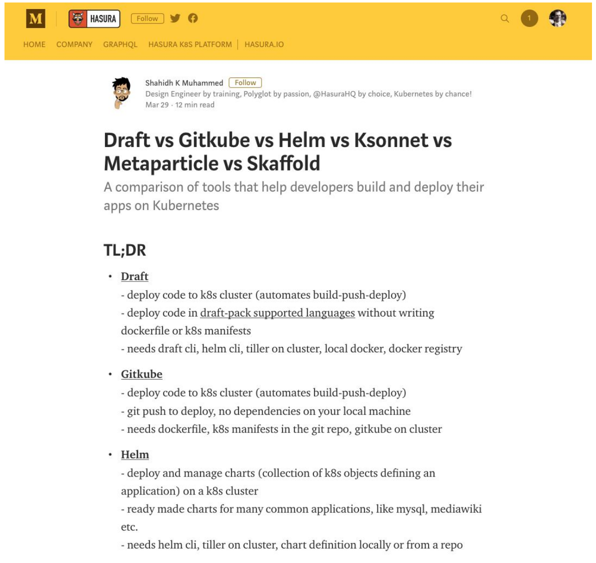
Enables developers to build Operators based on their expertise without requiring knowledge of Kubernetes API complexities.

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.

Namebuddy flow
Listen to Ingress objects
K8s API
create delete update
Read objects hosts field and IP address
Create/delete/update DNS record
DNS provider API
create delete update
QCon

Pattern: Development and Deployment



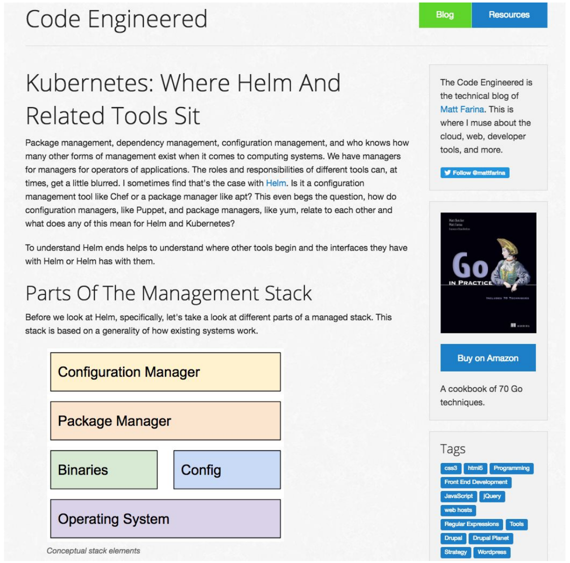
Shahidh K Muhammed Follow
Design Engineer by training, Polyglot by passion, @HasuraHQ by choice, Kubernetes by chance!
Mar 29 · 12 min read

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



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.

A cookbook of 70 Go techniques.

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

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

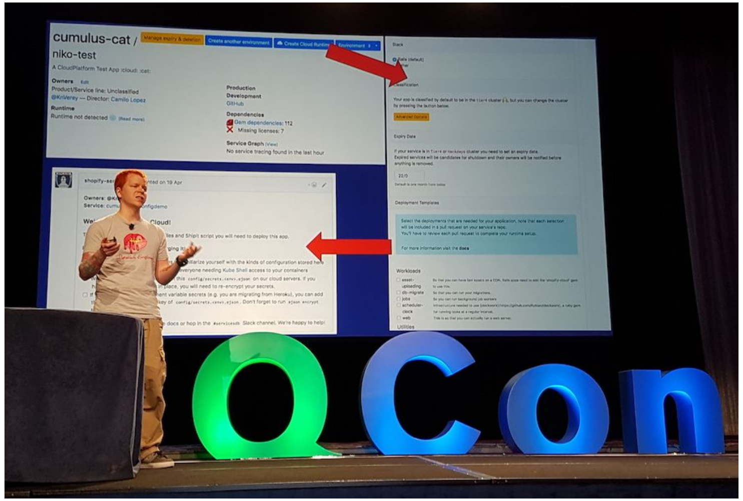
Pattern: Development and Deployment

- 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
- Forge
  - Automates build-push-deploy
  - Templates k8s/Ambassador config

- 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

Pattern: Development and Deployment



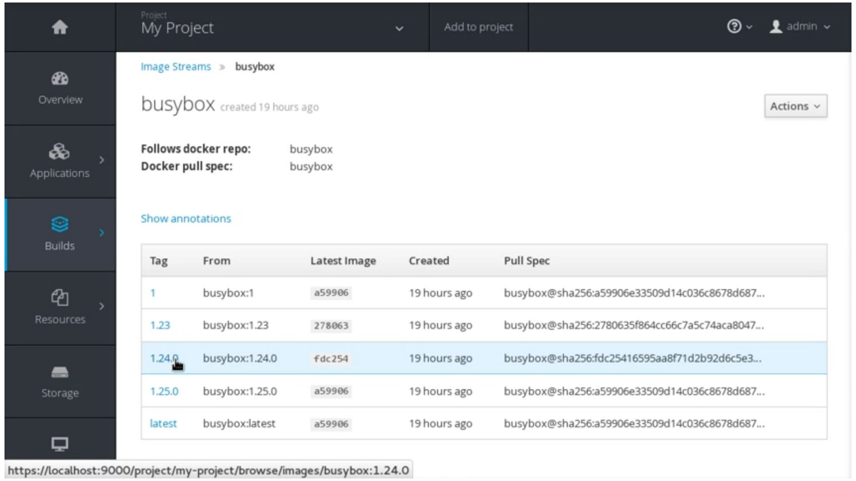


Image Streams » busybox

busybox created 19 hours ago

Follows docker repo: busybox
Docker pull spec: busybox

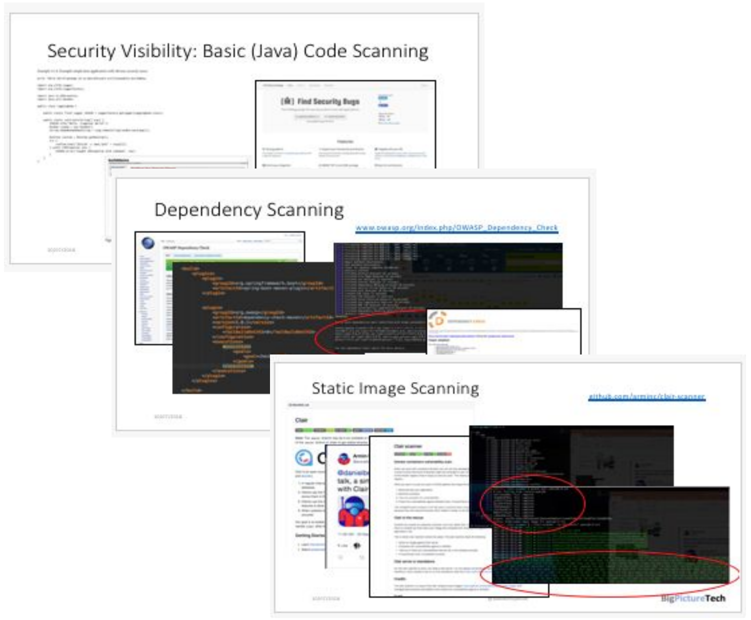
Show annotations

Tag	From	Latest Image	Created	Pull Spec
1	busybox:1	a59906	19 hours ago	busybox@sha256:a59906e33509d14c036c8678d687...
1.23	busybox:1.23	278063	19 hours ago	busybox@sha256:2780635f864cc66c7a5c74aca8047...
1.24.0	busybox:1.24.0	fdc254	19 hours ago	busybox@sha256:fdc25416595aa8f71d2b92d6c5e3...
1.25.0	busybox:1.25.0	a59906	19 hours ago	busybox@sha256:a59906e33509d14c036c8678d687...
latest	busybox:latest	a59906	19 hours ago	busybox@sha256:a59906e33509d14c036c8678d687...

https://localhost:9000/project/my-project/browse/images/busybox:1.24.0

Pattern: CI/CD

- Make is easy to do the right thing
  - Self-service pipeline creations
  - Bake-in hooks/slots for platform
- Testing of NFRs is vital
  - Security
  - Performance
  - Quality


Security Visibility: Basic (Java) Code Scanning

Dependency Scanning

www.owasp.org/index.php/OWASP_Dependency_Check

Static Image Scanning

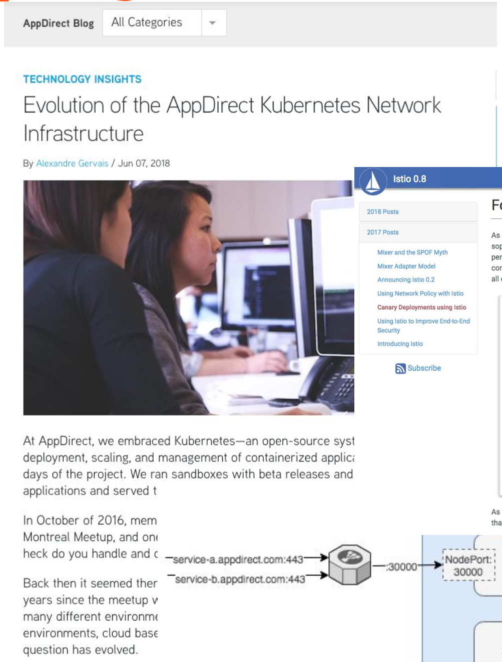
github.com/arminc/clair-scanner

BigPictureTech

https://www.slideshare.net/dbryant_uk/codemotion-rome-2018-continuous-delivery-with-containers-the-good-the-bad-and-the-ugly

Pattern: Layer 7 Deployment Control

- Allows fine-grained control
- Envoy is de facto proxy data plane
- Many control planes
    - Ambassador
    - Gloo
    - Istio



TECHNOLOGY INSIGHTS

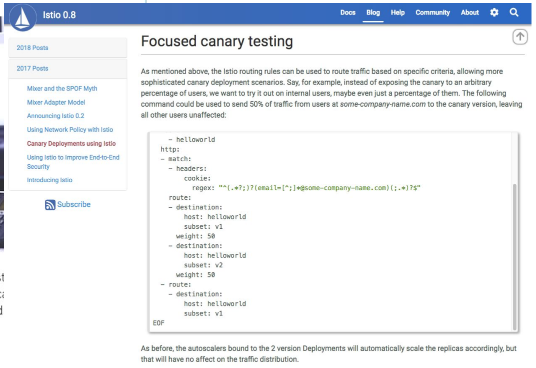
Evolution of the AppDirect Kubernetes Network Infrastructure

By Alexandre Gervais / Jun 07, 2018

At AppDirect, we embraced Kubernetes—an open-source syst
deployment, scaling, and management of containerized applic
days of the project. We ran sandboxes with beta releases and
applications and served t

In October of 2016, mem
Montreal Meetup, and on
heck do you handle and c

Back then it seemed ther
years since the meetup v
many different environme
environments, cloud base
question has evolved.

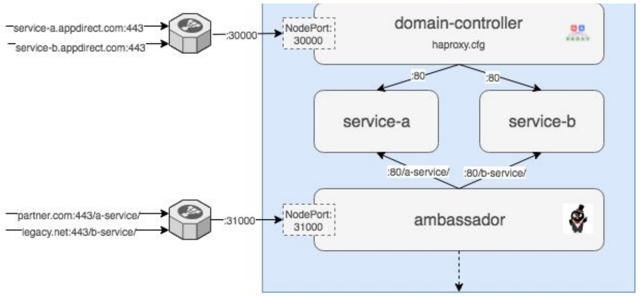


Focused canary testing

As mentioned above, the Istio routing rules can be used to route traffic based on specific criteria, allowing more sophisticated canary deployment scenarios. Say, for example, instead of exposing the canary to an arbitrary percentage of users, we want to try it out on internal users, maybe even just a percentage of them. The following command could be used to send 50% of traffic from users at *some-company-name.com* to the canary version, leaving all other users unaffected:

```
    - helloworld
  http:
  - match:
    - headers:
        cookie:
          regex: "^(.*?;)?(email=[^;]*@some-company-name.com)(;.*)?$"
    route:
    - destination:
        host: helloworld
        subset: v1
      weight: 50
    - destination:
        host: helloworld
        subset: v2
      weight: 50
  - route:
    - destination:
        host: helloworld
        subset: v1
EOF
```

As before, the autoscalers bound to the 2 version Deployments will automatically scale the replicas accordingly, but that will have no affect on the traffic distribution.



Pattern: Observability

- Essential part of the platform and developer workflow/experience
  - Monitoring, logging and tracing
  - Bake-in hooks to skaffolding
- Global/service dashboards
- "Observability and Avoiding Alert Overload from Microservices at the Financial Times"

Prometheus

ZIPKIN

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

@danielbryantuk

More info: dzone.com/articles/creating-a-positive-developer-experience-for-conta

datawire.io/what-is-cloud-native | getambassador.io | istio.io | telepresence.io, prometheus.io | “Kubernetes Up and Running”