Creating an Effective Developer Experience on Kubernetes

Daniel Bryant
@danielbryantuk | @datawireio
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)

DevEx: DevOps, Lean, and UX
DevEx and Workflow
The Ideal Workflow
The Platform Drives DevEx
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

Kelsey Hightower @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.

You know you’re building a PaaS when you start work on that custom templating engine for deployments and configuration files.

You know you’re building a PaaS when you start stitching together 1,000,000 other tools in order to get one-click deployments.

Nothing wrong with building a PaaS; just know that’s what you’re doing.

https://twitter.com/kelseyhightower/status/851935087532945409
The "Paved Road" PaaS for Microservices at Netflix: Yunong Xiao at QCon NY

At QCon New York 2017, Yunong Xiao presented "The Paved Road 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 organizational goals of velocity and reliability. The Netflix PaaS team attempts to provide a sensibly configured but customizable "paved road" platform for developers by offering standardized and scaffolded components, pre-assembling the platform, and by providing extensive automation and testing.

Xiao Principal Software Engineer at Netflix, began the talk by referencing the Wikipedia definition of "Paved."

Full Code Developers at Netflix from Windows to Test-Driven Development by July 16, 2015

At Netflix, backend engineers are typically deployed on a Java/OM piece which is hosted off an Edge A1 hosting service that is created to meet the needs of their associated user delivery technologies like smart TVs, iOS and W8/Windows. These services are typically developed using Java/Netty and Node.js, and the delivery teams are not necessarily familiar with backend operations and platforms. Netflix immersively embraces a culture of freedom and responsibility ("F&R"), and the infrastructure and the overall organizational goals of velocity and reliability. Functionality provided by a PaaS can help with this balance, and this is implemented in these ways: in order to provide a homogenized but configurable "paved road" for developers, including the provision of standardized components, pre-assembled platform, and automation and testing.


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

At QCon New York 2018, Niko Kurtti presented "From Dev Our Journey to Kubernetes," and described the Shopify engineering teams journey to building their own Paas with Kubernetes as the foundation. Key takeaway for other teams looking to build their own Paas and associated developer workflow included: target high 98% of deployment and operational use cases, create platforms and tools in the underlying platform completely, educate and get people excited about the project and be conscious of vendor lock-in.

Full Code Developers at Netflix from Windows to Test-Driven Development by July 16, 2015

Kurtti production engineer at Shopify, began the talk by discussing how Shopify is a rapidly growing Canadian e-commerce company that operates a propriety e-commerce platform for online stores and retail point-of-sale systems. Shopify currently has 3000+ employees, and the company processed $3 billion in transactions in 2017. The underlying e-commerce software platform sees 800+ 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), in AWS using Chef and Marathon. Developers lead the development experience of Marathon, and Kurtti commented that this platform actually scales out as well, with "simple UI clients" to increase the number of instances and an associated GPU and RAM. Although the platform had built-in 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.

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
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!)
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

- 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”
“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

https://www.battery.com/powered/%EF%BB%BFwho-doesnt-like-lock-in/
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

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

TLDR
- **Draft**
  - deploy code to k8s cluster (automates build-push-deploy)
  - deploy code in draft's supported languages without writing dockerfile or k8s manifests
  - needs Draft's CLI, helm CLI, tiller on cluster, local docker, docker registry
- **Gitkube**
  - deploy code to k8s cluster (automates build-push-deploy)
  - get 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 mode 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 operations of applications. The roles and responsibilities of different tools are, at times, a little blurred. Sometimes find that there’s the case with Helm, is a configuration management tool like Draft or a package manager like apt? This even beg 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?

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.

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
Pattern: CI/CD Enforces Policy

- Make is 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
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”
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”