Best Practices for Orchestrating the Cloud with Kubernetes

Carter Morgan
@_askcarter

#io16

Best Practices for Orchestrating the Cloud with Kubernetes

Carter Morgan
@_askcarter

Hi, I'm Carter Morgan -- a brand new Developer Programs Engineer on the Cloud Platform Team at Google and today I'm going to talk about some of the best practices I've picked up for writing and scaling applications in the cloud, since I joined Google.

But before that I want to tell you a quick story…

#K8story

Photo © Annette Shaff via Canva.com

When I started at Google six months ago -- I knew nothing about the cloud… so when I was to ramp up on it, I was overwhelmed.

There were so many tools, so many new developments that I started wondering if maybe I was smart enough to be at Google or to program the Cloud. Until one day I ended up on a bus next to Kelsey Hightower -- the guy who's literally writing the book on a container automation framework called Kubernetes. We got to talking about how overwhelming everything was and he had one piece of advice for me:

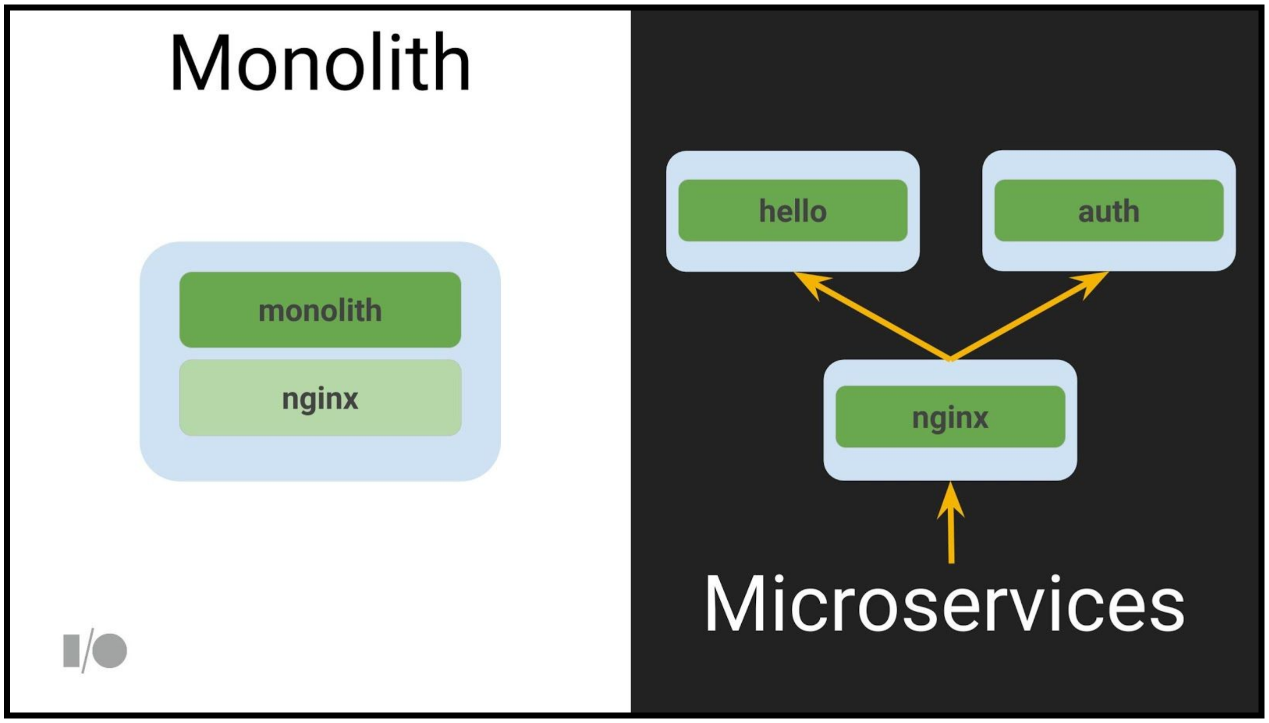
"Use Kubernetes"

And I said, "You want me to learn *another* tool, on top of everything else?"

But then he spend the next two hours showing me Kubernetes and I had to admit: He was right. Kubernetes offered abstractions that made the cloud more approachable.

Over the next few months -- I ended up learning a lot while creating a course with Kelsey over what he showed me on that bus ride.   And the rest of this talk is going to share some of those learnings.

By the end of this talk, you'll be able to handle the hurdles to -- and best practices for -- managing an application at scale on the cloud.

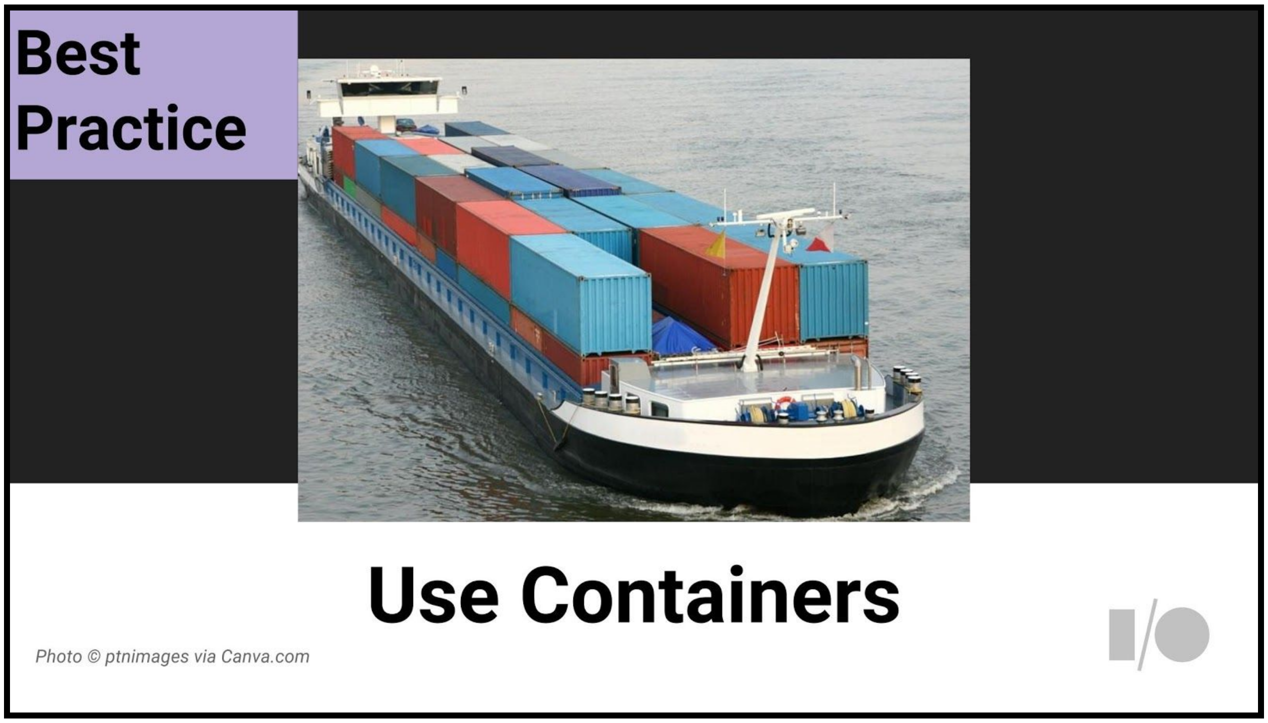


The first hurdle when writing scalable apps for the cloud is the App. How do you write it? How do you share it? Over the last decade -- new demands from users and new technological advances have led to changes in the way we write code that's meant to be scalable in Cloud.

On screen you'll see two versions of the same application: Both applications provide the same functionality -- they send back a "hello" message to the user and can restrict access of this service to authenticated users. One is a traditional monolithic application and, on the other side, the same application broken up into microservices.

One of the first things I had to learn was that neither approach is the one true way to make an application. There are benefits to both approaches. If you do choose to go the microservices route, you're basically taking a UNIX approach to application design. Instead of having one "do everything" application -- you break an app up into smaller, focused applications that do one thing well and no how to play nice with other applications.

This gives you independent deployments, flexibility on toolings, and faster builds. In our picture above -- we could update or scale our hello application independently from everything else. We could also choose to write in any programming language (independent of what the other services were written in) and each application is going to build fast.



Best Practice

Use Containers

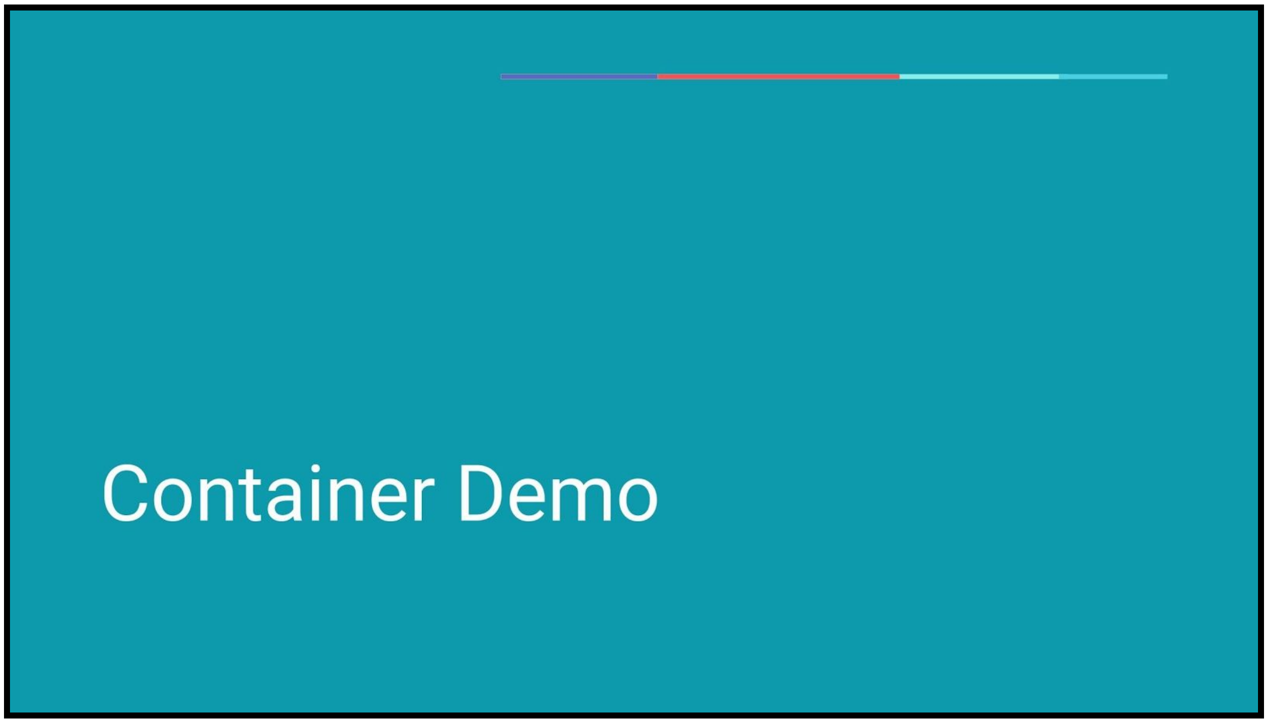
Photo © ptnimages via Canva.com

These advances in application design were precipitated by the development of application isolation technologies like VMs and Containers. Container bundle you app code and dependencies into a unit, abstracting app from infrastructure. Make apps easy to deploy across dev, test, and prod. And it’s easy to move containers to different cloud providers, so you’re not locked in. Performance, Repeatability, Isolation, Portability

In the picture above let’s pretend the boat is an Operating System -- to run an isolated application using containers is a simple as loading and unloading the containers onto the boat. This is true because containers achieve their isolation from within the Operating system.

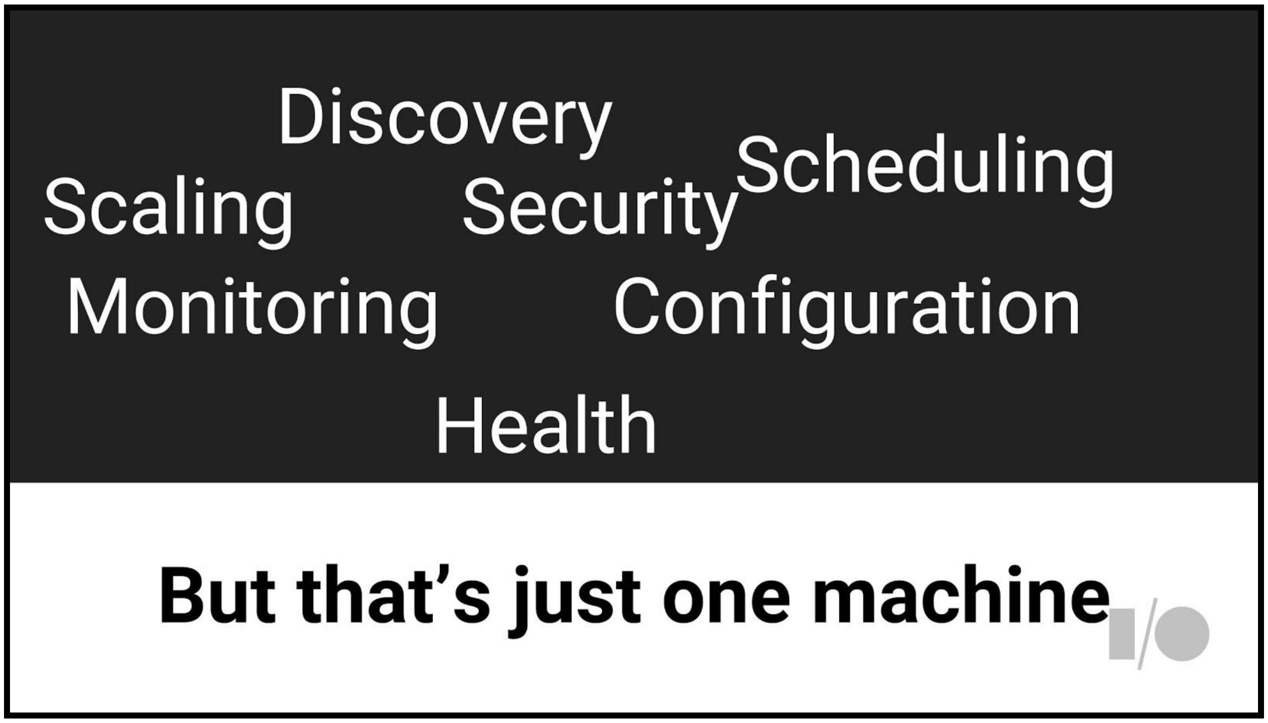
Running an isolated application inside of a VM, on the other hand, is like creating a whole entire new boat to run the application.

Effective… not efficient. So as a best practice, use containers to package and distribute your applications.



Container Demo

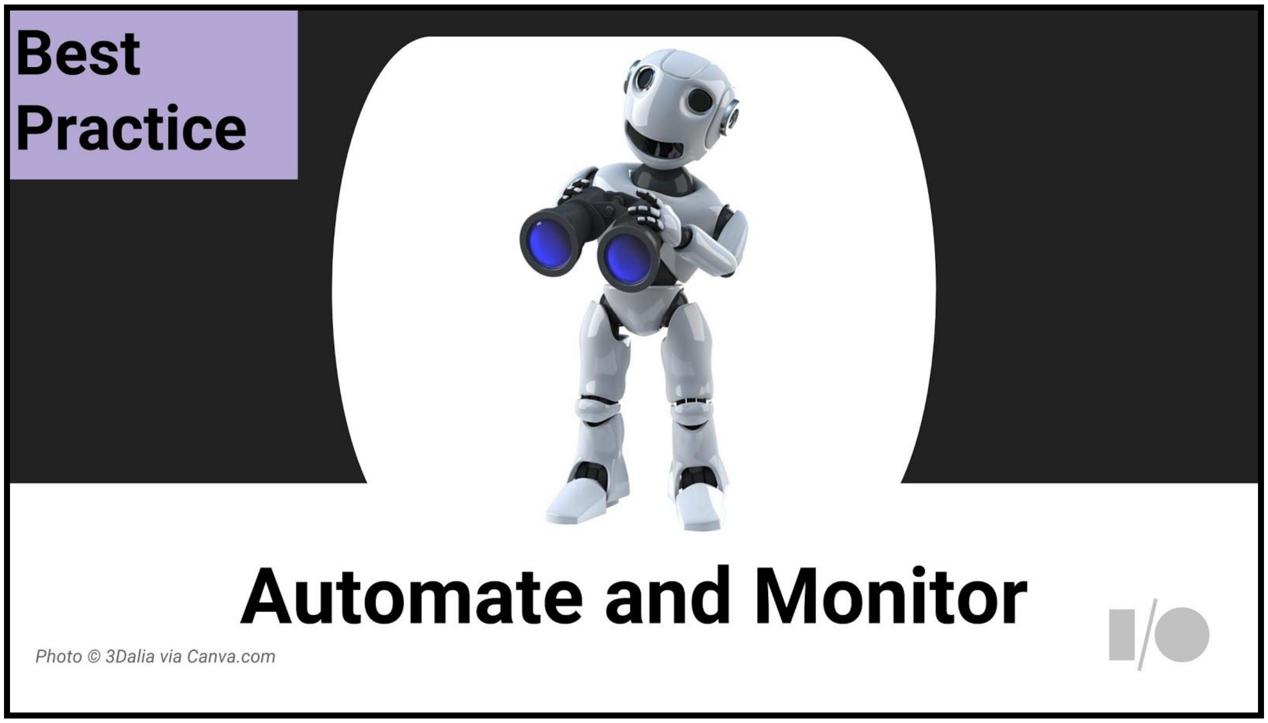
Code is here: https://github.com/askcarter/io16



Discovery
Scaling
Security
Scheduling
Monitoring
Configuration
Health

But that’s just one machine

Now, I’ll be honest, at this point I thought I knew everything I need to know to get an application up and running on the cloud. But it turns out that Packaging and distributing an app is just a small percent of the problem.   The next hurdle is infrastructure.



Best Practice

Automate and Monitor

Photo © 3Dalia via Canva.com

Once deployed, we want to make sure that our applications are running and are performant. We want to make sure they're discoverable so that we can access them. We want them to be secure and to scale up or down as needed to serve our users.

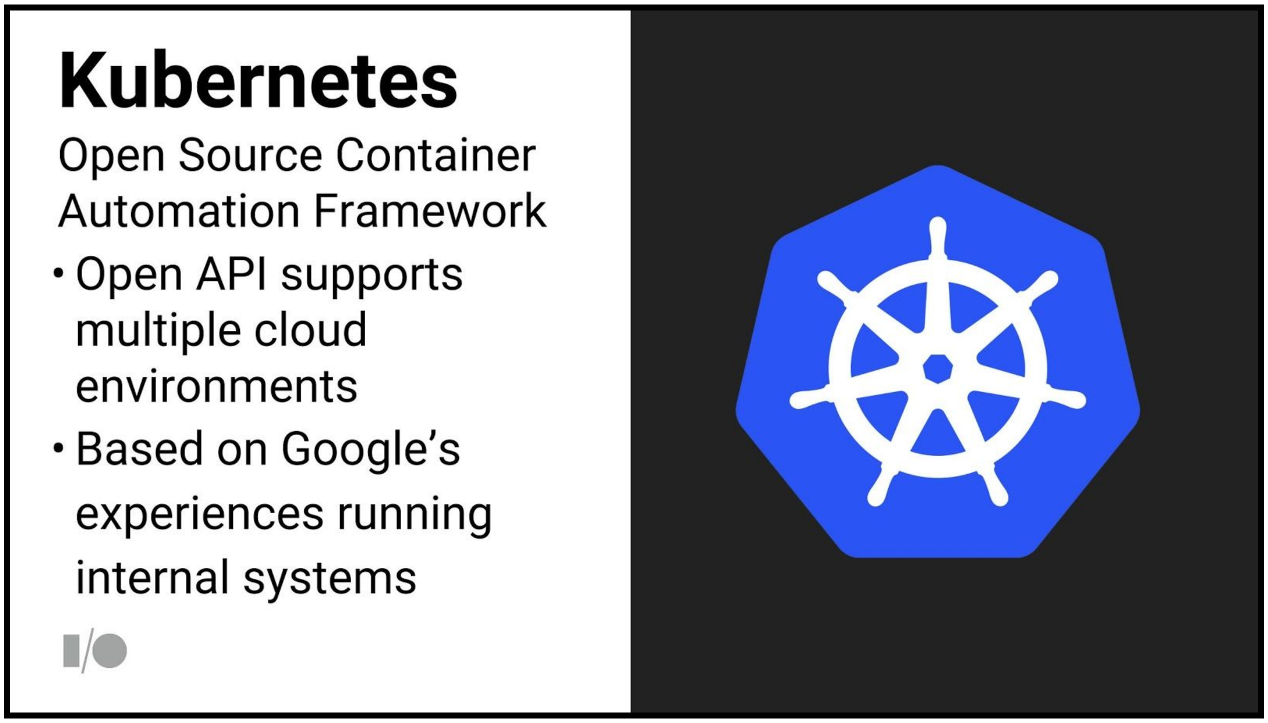
And Ideally we want all of this to happen automatically.

Best Practice

Avoid Vendor Lock-in

Photo © Michele Piacquadio via Canva.com

And we want all of this complexity managed for us without being locked in to anyone cloud provider, container format, or way of doing things.   Ideally, we'll be able to use tools that let us write our own story.



Kubernetes

Open Source Container Automation Framework

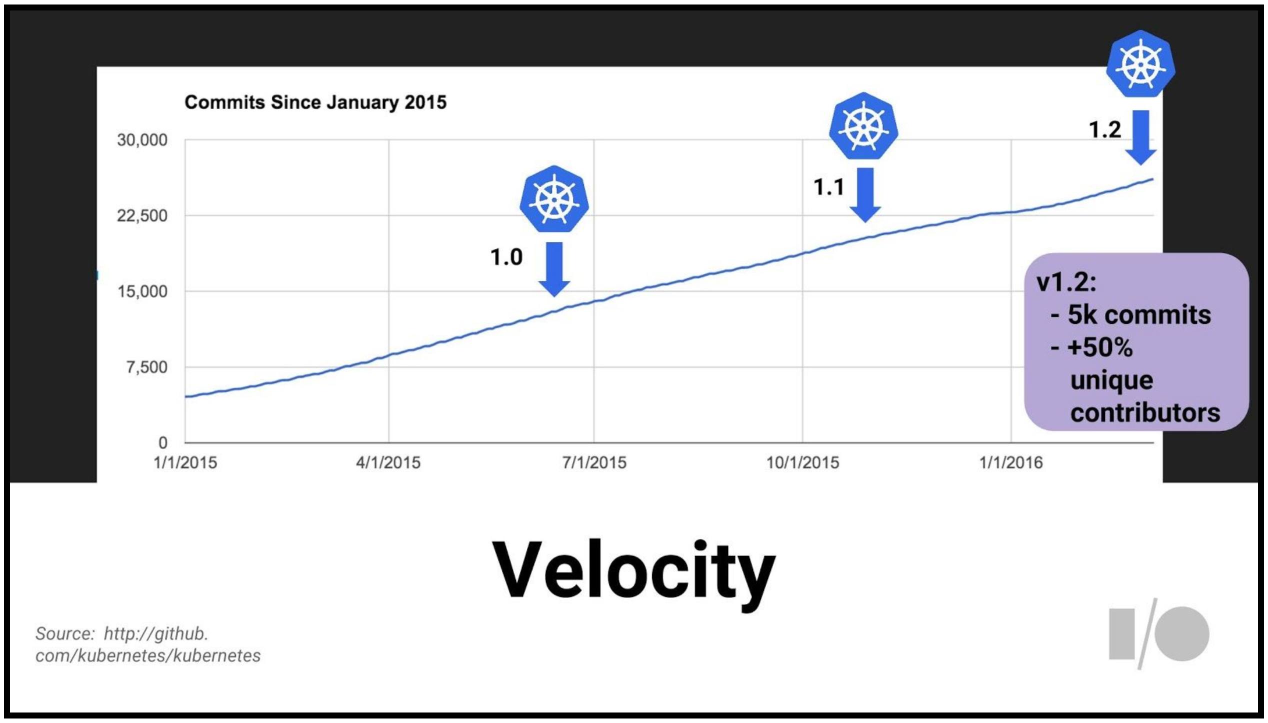
- Open API supports multiple cloud environments
- Based on Google's experiences running internal systems

Which leads us to Kubernetes…

Kubernetes is an open source project that google founded about a year and a half ago. It is an underlying framework for container management and automation. It is how look at, manage, and engage with containers across multiple cloud providers. This means that as your applications grow, kubernetes help you manage that application (at scale) while still providing portability and options in case you need it.

Kubernetes is based on learnings from how Google itself has been running applications and containers, internally. These learnings have given rise to new primitives, new ways of looking at orchestrating the cloud in order to abstract away the underlying machines.

So that you can Manage applications, not machines

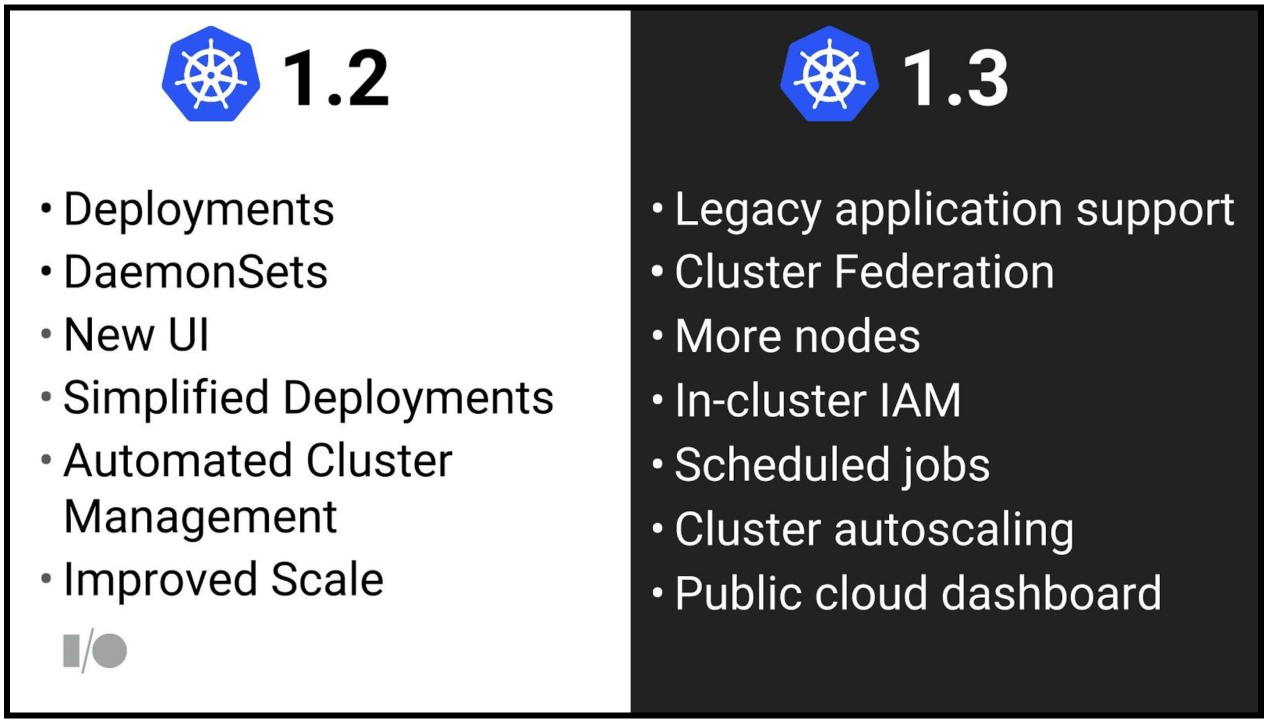


Velocity

Source: http://github.com/kubernetes/kubernetes

Now Kubernetes isn't perfect -- it's developing so fast that the documentation can be confusing at times and setting Kubernetes up on your own requires a lot of knowledge about Kubernetes.

That's because there are so many people and organization getting involved with contributing -- and that's a good thing. For version v1.2 there were around 5000 commits of new code and many of those commits came from new contributors -- so get involved! Kubernetes is open.



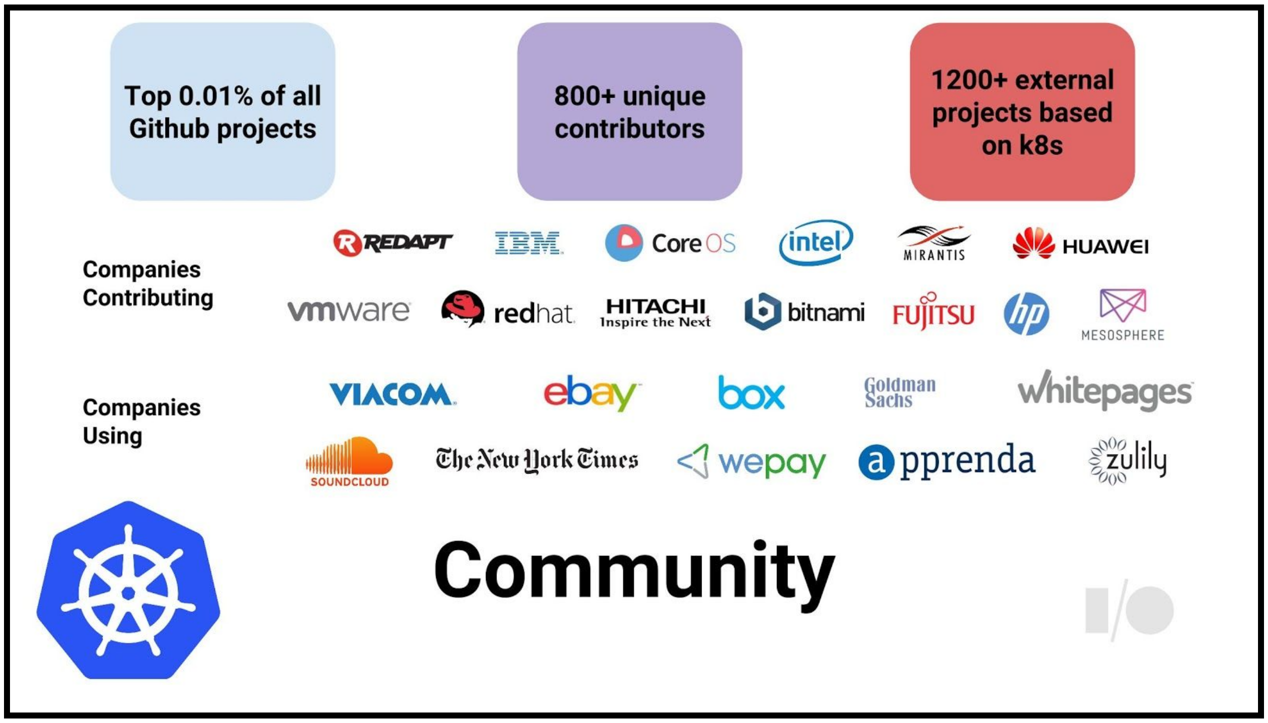
1.2

- Deployments
- DaemonSets
- New UI
- Simplified Deployments
- Automated Cluster Management
- Improved Scale

1.3

- Legacy application support
- Cluster Federation
- More nodes
- In-cluster IAM
- Scheduled jobs
- Cluster autoscaling
- Public cloud dashboard

There are a ton of amazing features in Kubernetes (currently) and many more coming. This talk is only going to cover the basics. You'll know enough to get started and to deploy your apps -- but check out the 1.3 release notes and the docs over at kubernetes.io.



Top 0.01% of all Github projects

800+ unique contributors

1200+ external projects based on k8s

Companies Contributing

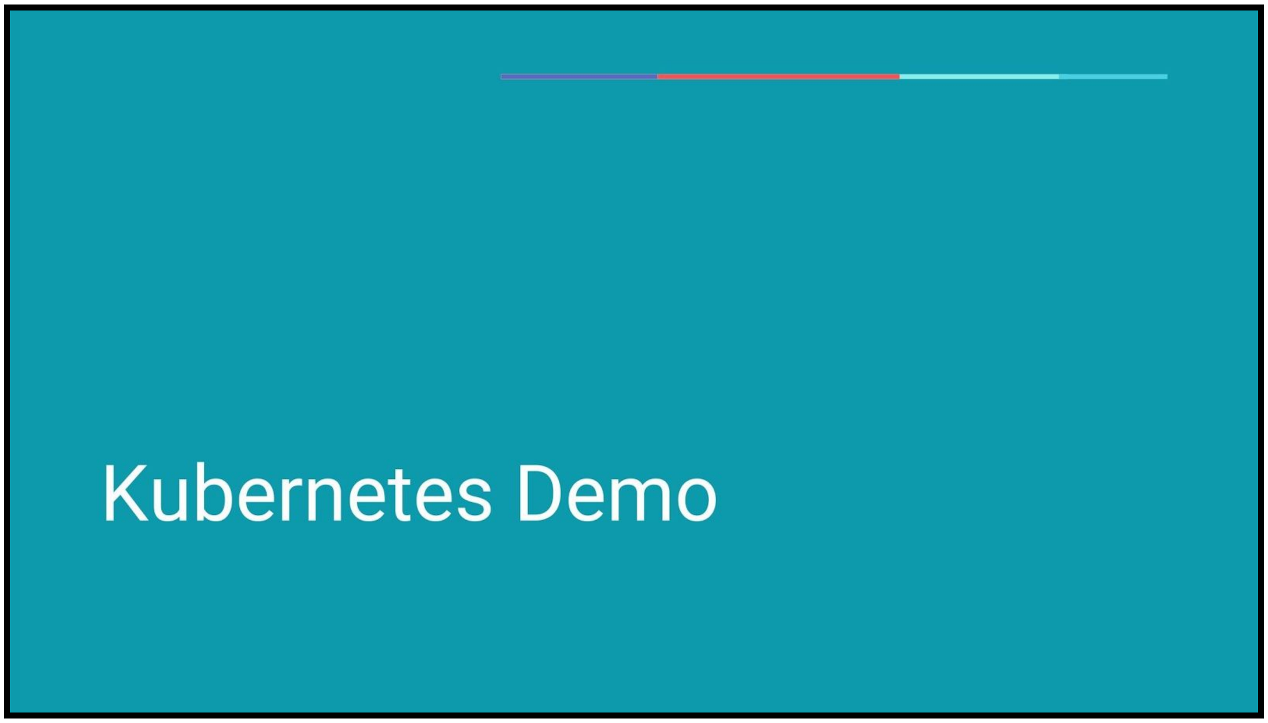
REDAPT IBM CoreOS intel MIRANTIS HUAWEI vmware redhat HITACHI Inspire the Next bitnami FUJITSU hp MESOSPHERE

Companies Using

VIACOM ebay box Goldman Sachs whitepages SOUNDCLOUD The New York Times wepay apprenda zulily

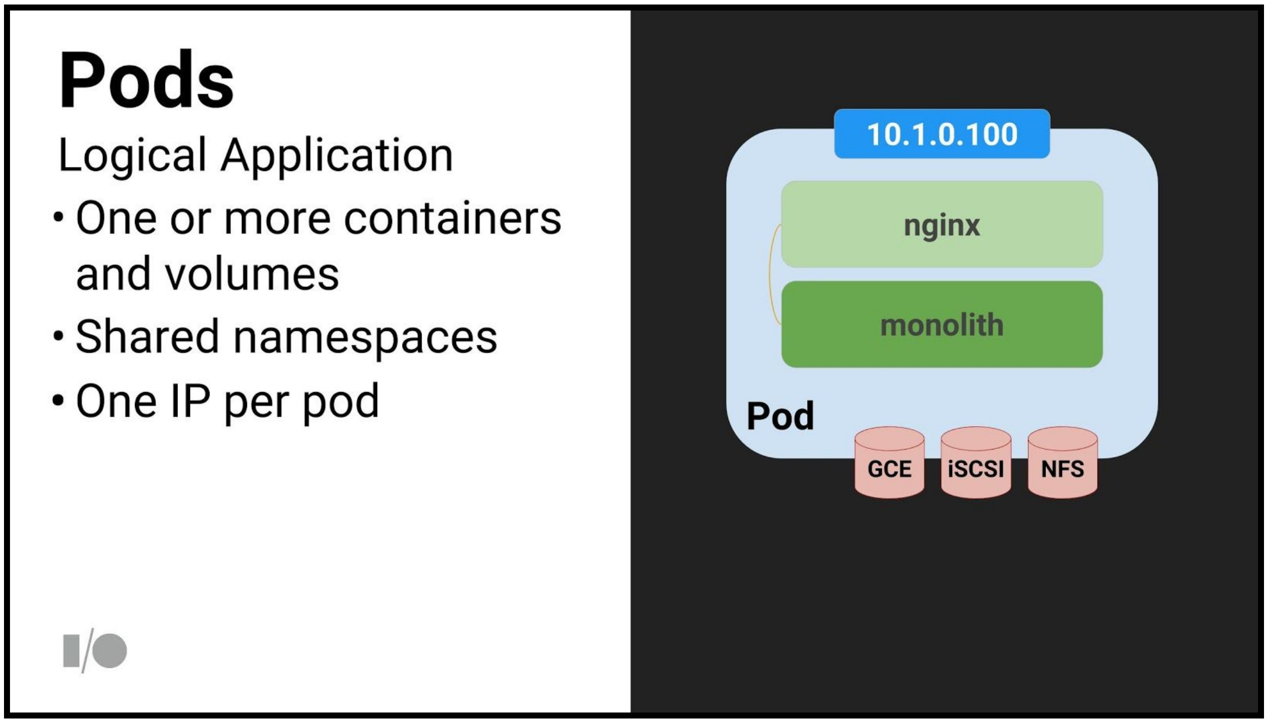
Community

In fact, Kubernetes is in the top 1 of the top 1% of all github projects, there are over 800 unique contributors, and over 1200 external projects based on Kubernetes. Now today we only showed a small demo but many large companies are running Kubernetes in production with 100 or even 1000 node clusters. So Kubernetes is something that can let you start small but can keep up with the demands you throw at it as you scale.



Kubernetes Demo

Code is here: https://github.com/askcarter/io16



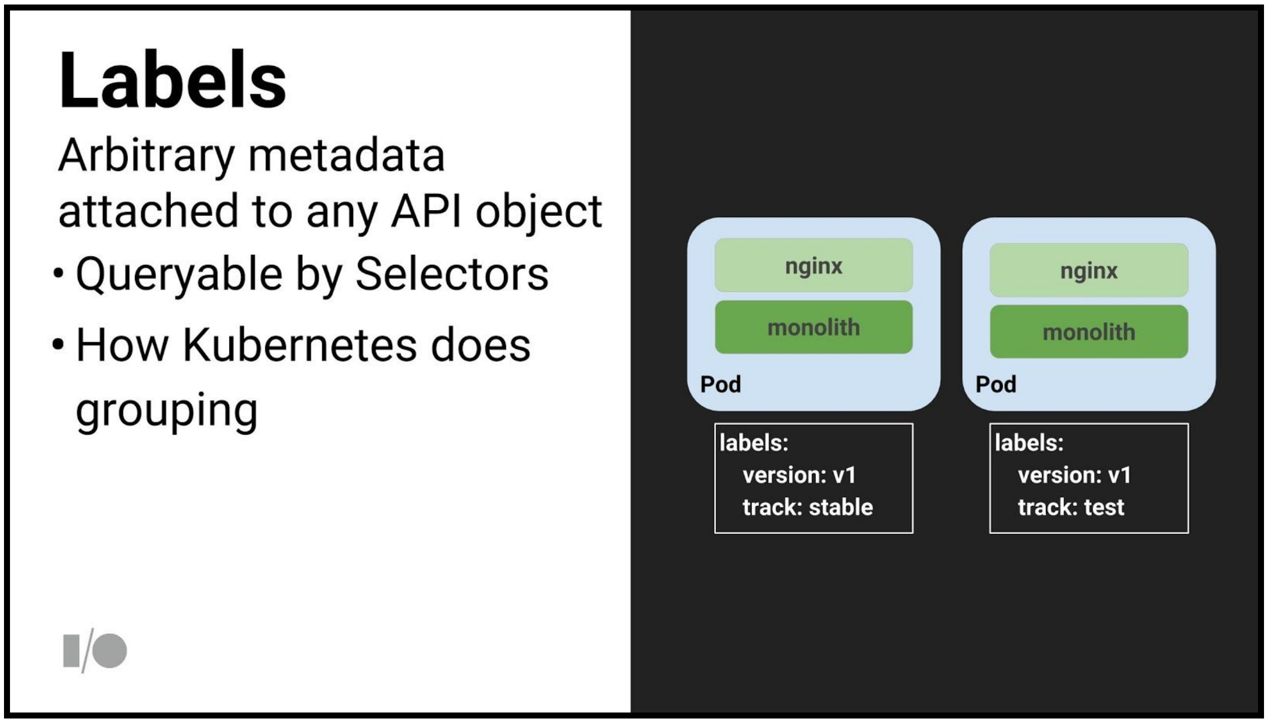
Pods

Logical Application

- One or more containers and volumes
- Shared namespaces
- One IP per pod

10.1.0.100
nginx
monolith
Pod
GCE
iSCSI
NFS

Pods represent a logical application in Kubernetes. They hold one or more containers and containers that are part of the same pod are guaranteed to be scheduled together onto the same machine, and can share state via local volumes. Because pods share a namespace -- pods can access any data mounted onto the pods: called Volumes. Volumes are are data disks that live as long as the pods lives -- and can be used by the containers in that pod. Additionally, each pods gets a unique IP address -- which means if I had two of the Pod on screen running in production -- nginx could use port 80 for both instances with no conflicts.

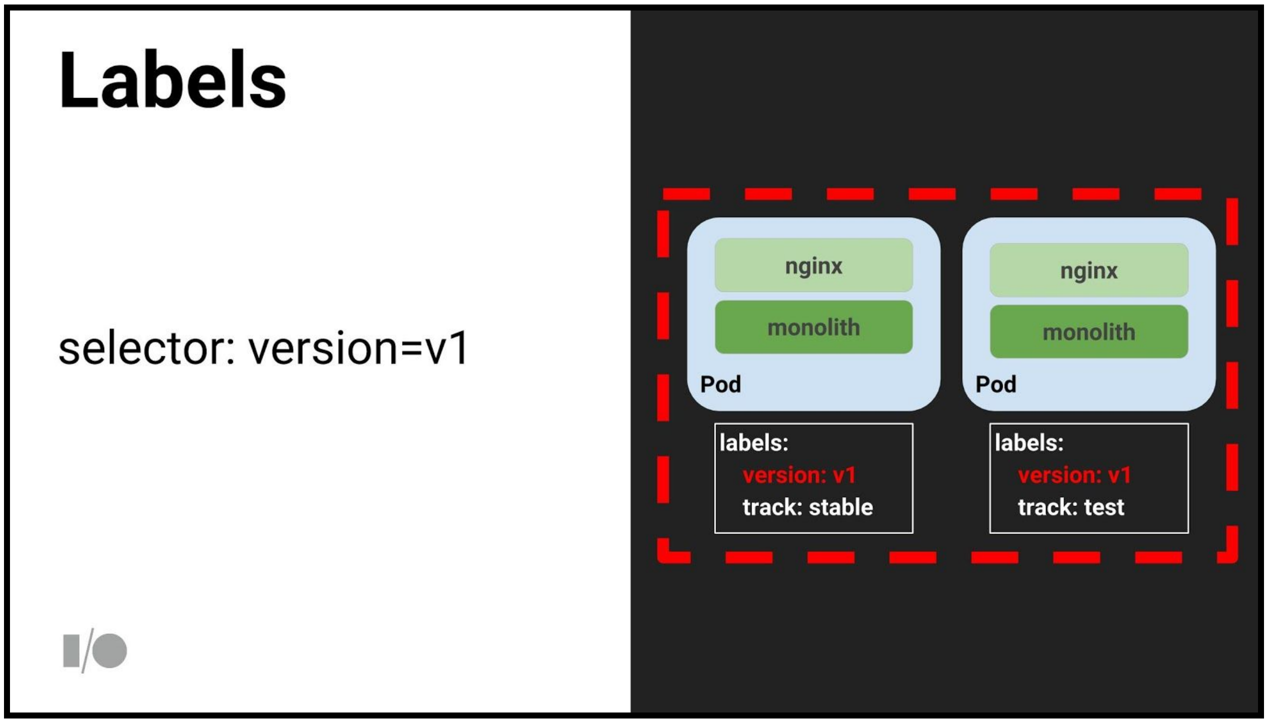


Labels

Arbitrary metadata attached to any API object

- Queryable by Selectors
- How Kubernetes does grouping

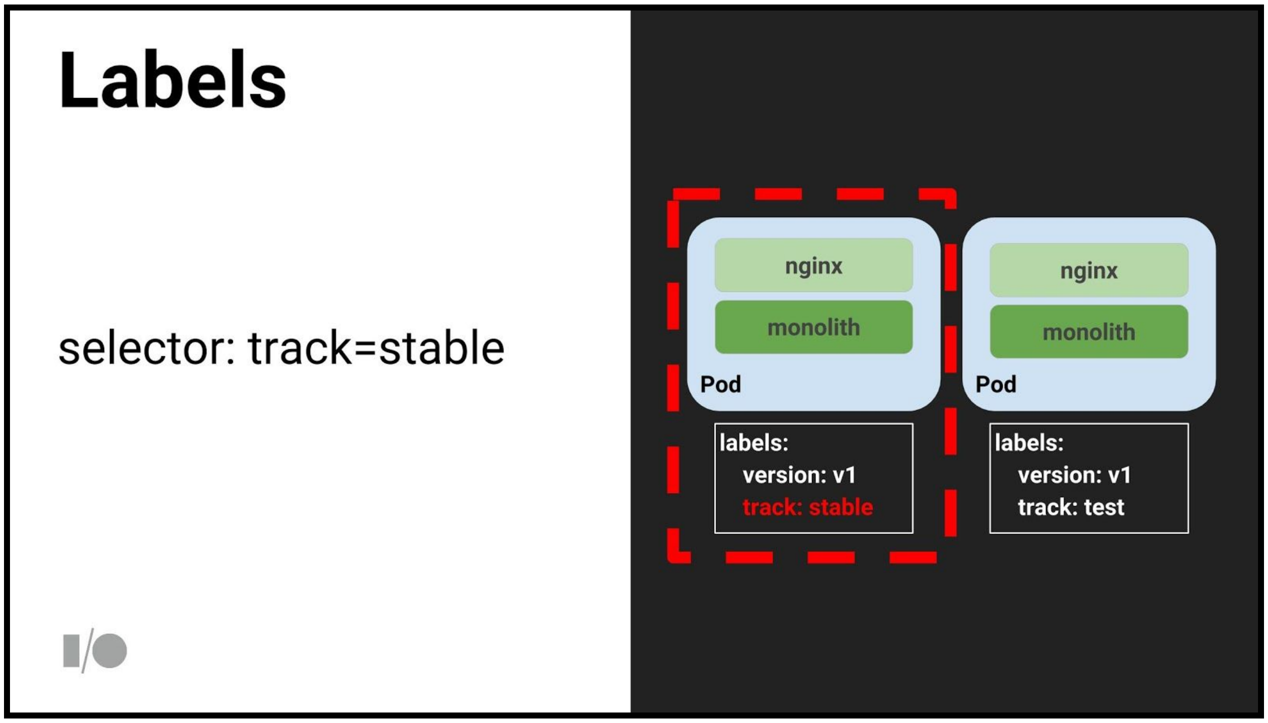
Kubernetes supports labels which are arbitrary key/value pairs that users attach to pods (and in fact to any object in the system). Users can use additional labels to tag the service name, service instance (production, staging, test), and in general, any subset of their pods. A label query (called a “label selector”) is used to select which set of pods an operation should be applied to. Taken together, labels and deployments allow for very flexible update semantics.



Labels

selector: version=v1

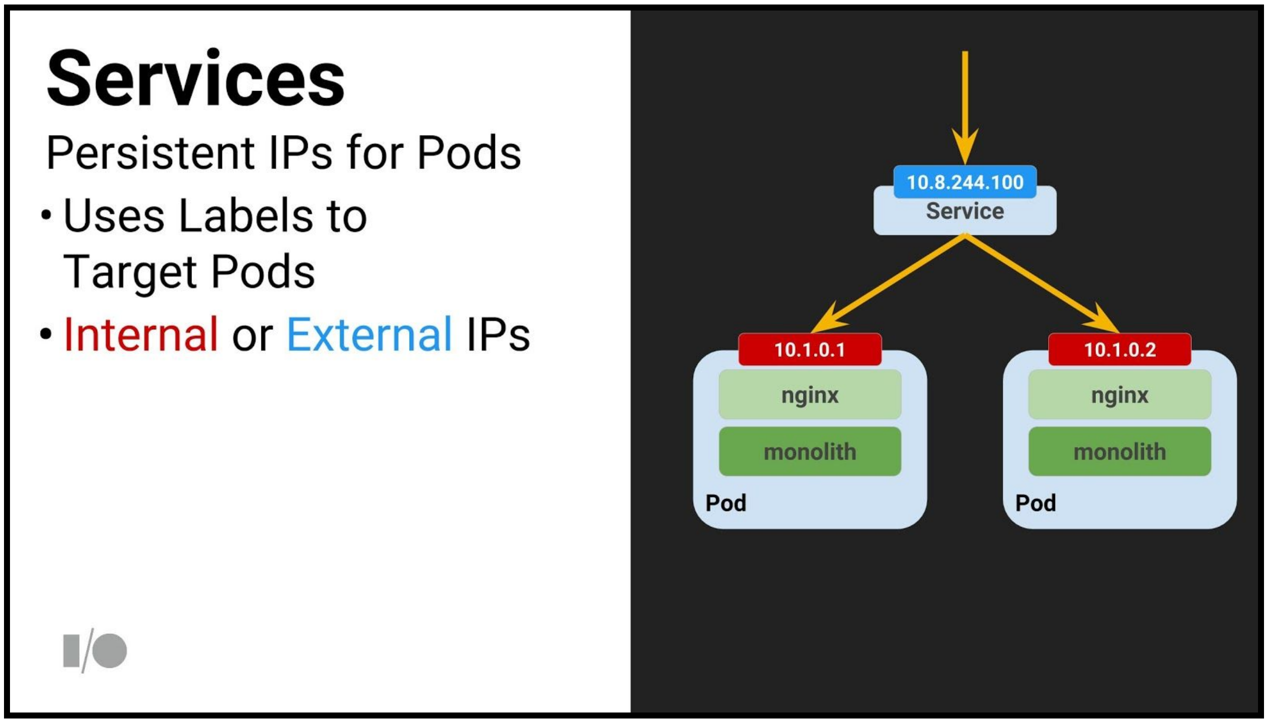
…



Labels

selector: track=stable

…



Services

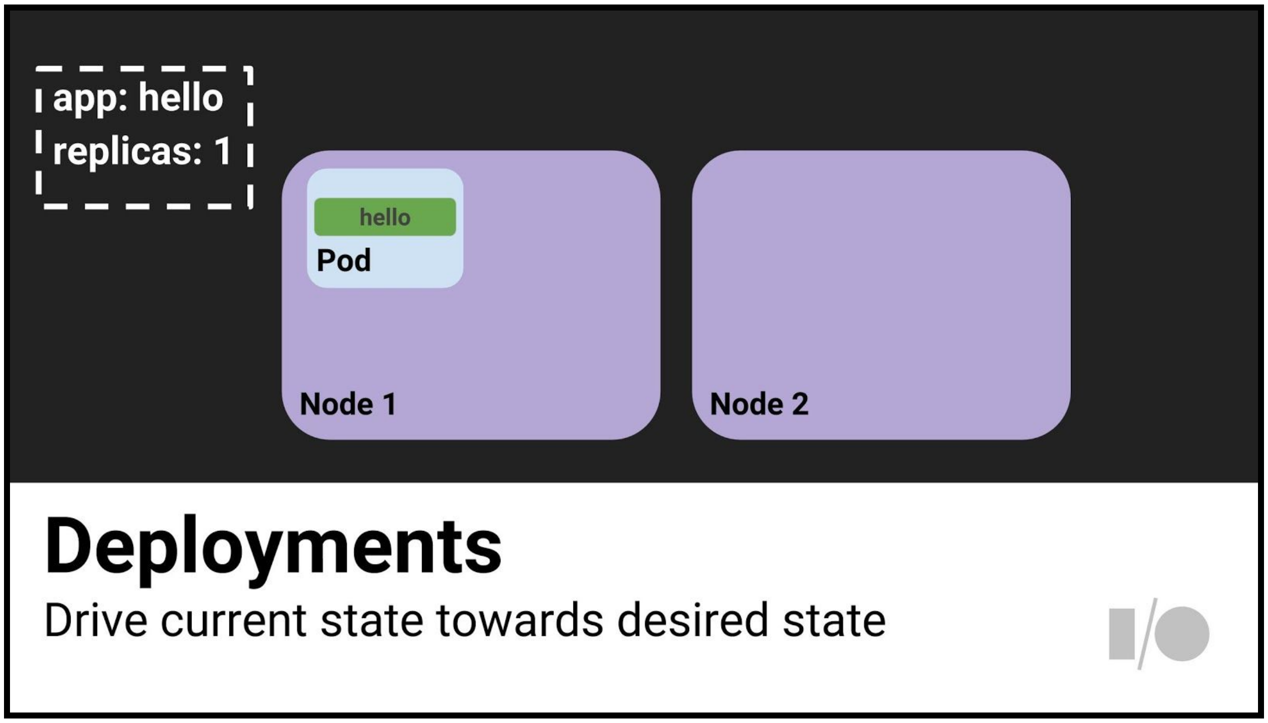
Persistent IPs for Pods

- Uses Labels to Target Pods
- Internal or External IPs

Pods can be restarted for all kinds of reason -- like failed liveness or readiness checks -- and this leads to a problem: what happens if we want to communicate with a set of Pods? When they get restarted they might have a different IP address. That's where services come in.

Services are stable endpoints for Pods which get their own unique IP address. The pods that a service exposes is based on a set of labels. If pods have the correct labels, they are automatically picked up ap and exposed by our services.

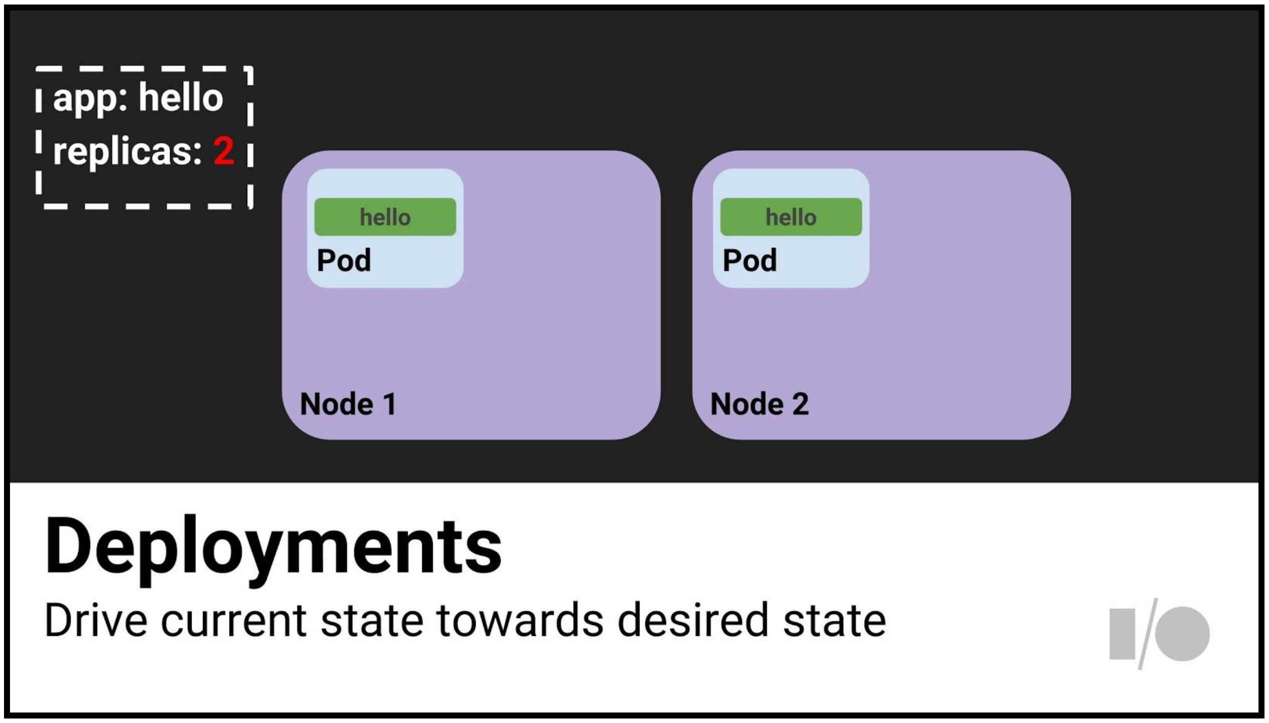
The level of access a service provides to a set of pods depends on the Service’s type -- depending on if you want the services visible internally, externally, or if you want the service to load balance incoming requests.



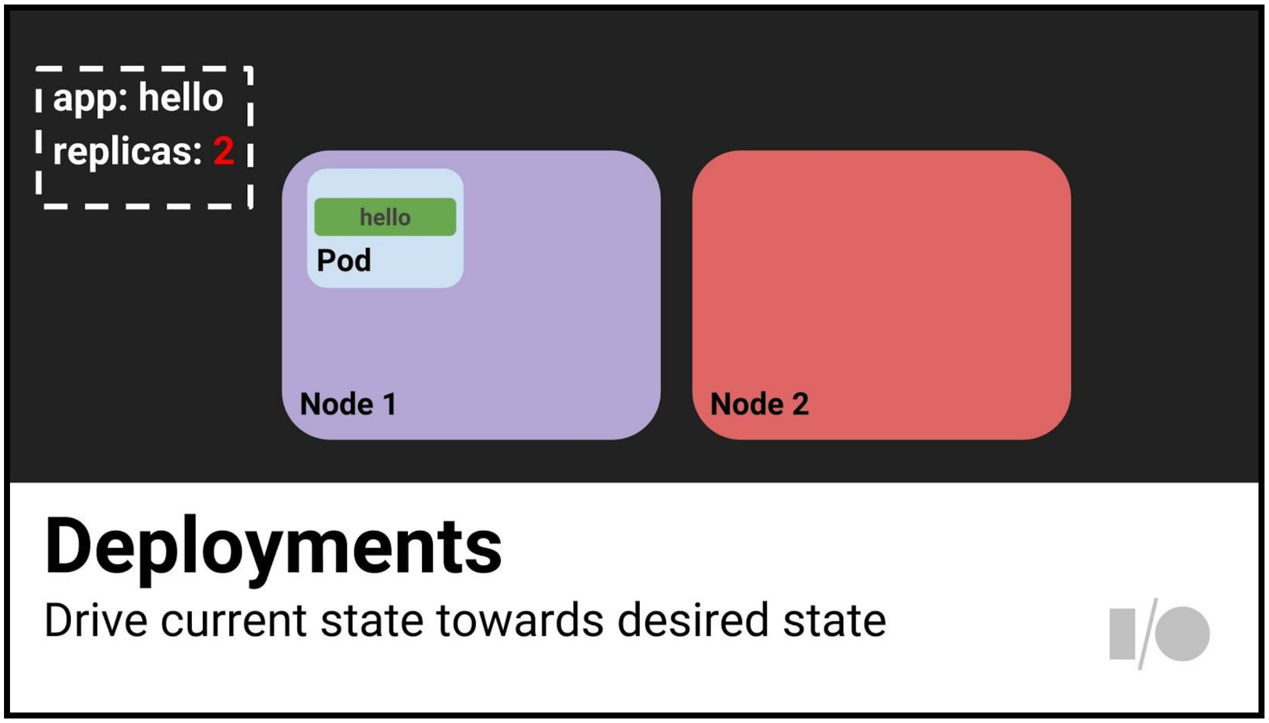
Deployments

Drive current state towards desired state

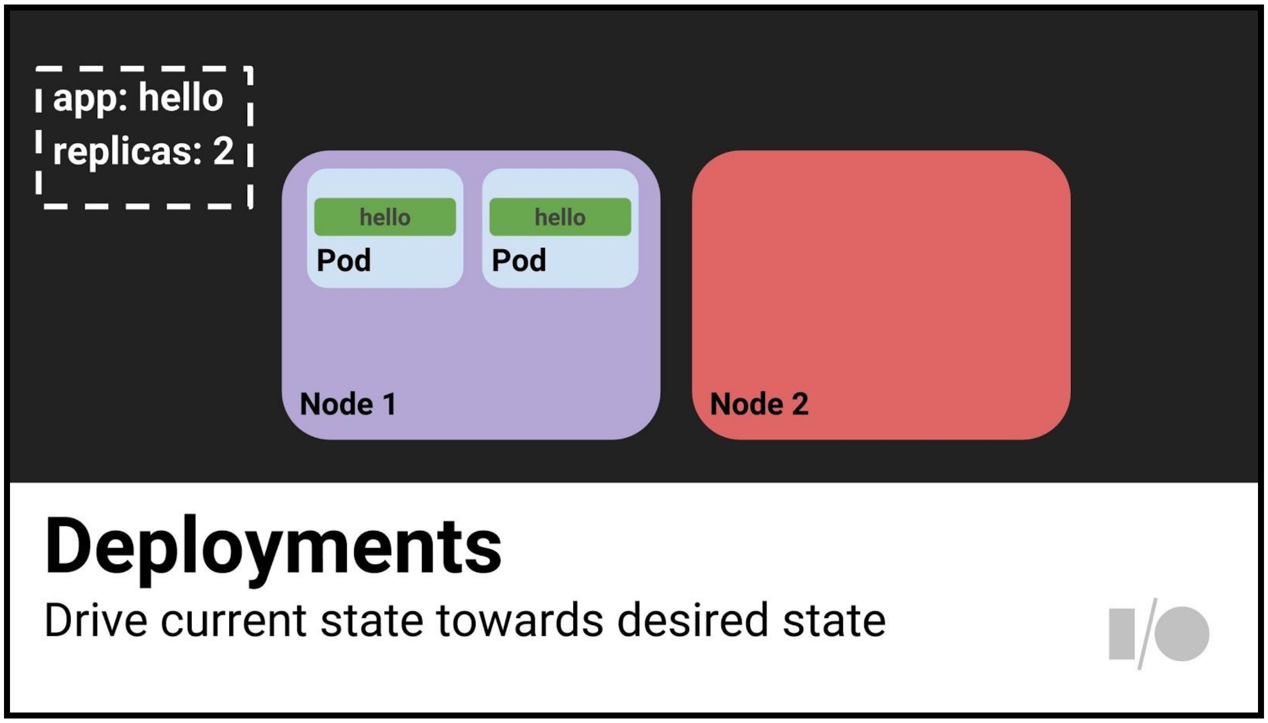
Deployments are used to ensure the number of pods *actually* running -- matches up to the number we want to be running. They handle scheduling the pods (onto machines) and they also handle scale the number of pods we have.



…



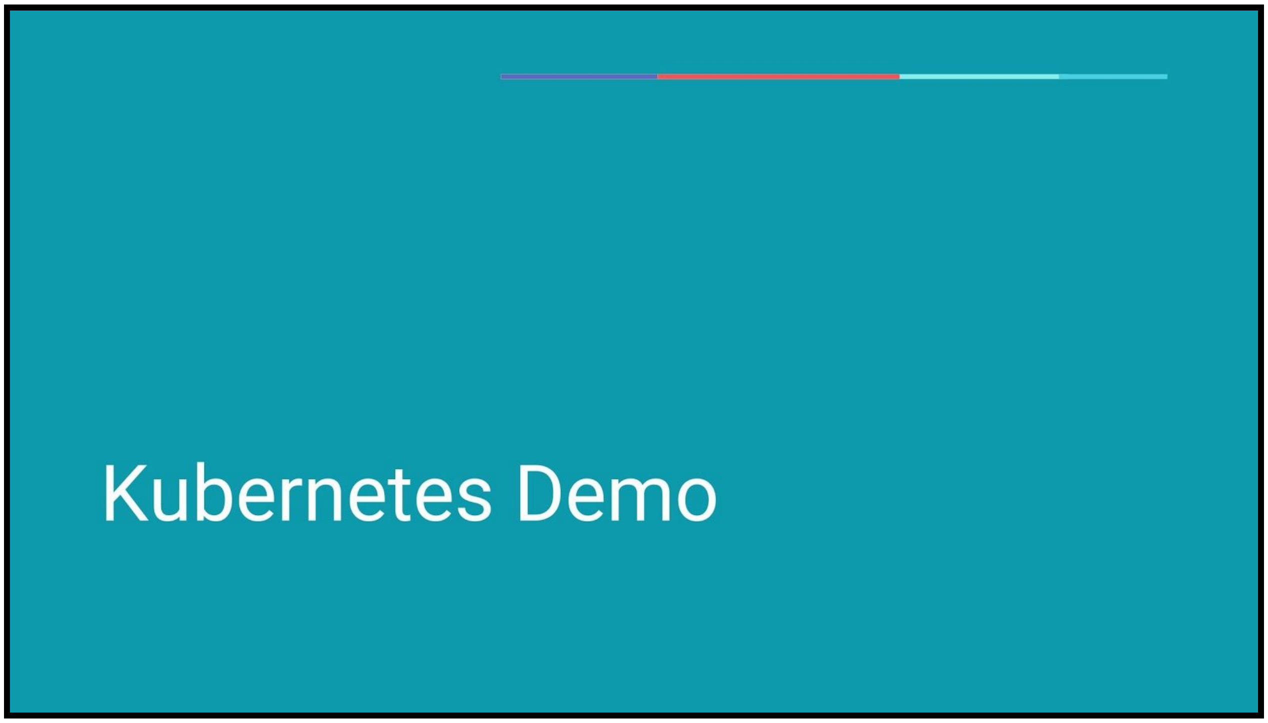
Have you noticed we haven't talked about the machines our code is running on at all, yet? Remember, Kubernetes lets you manage applications not machines -- and we're going to see one of the ways how here. In Kubernetes machines that our pods run on are called Nodes. In this example Node 2 went down leaving us with only one Pod running.



Deployments

Drive current state towards desired state

But we desire 2 -- so our deployment will start another pod on Node 1 -- and evey thing is good in the world.



Kubernetes Demo

Code is here: https://github.com/askcarter/io16

…

What about code updates?

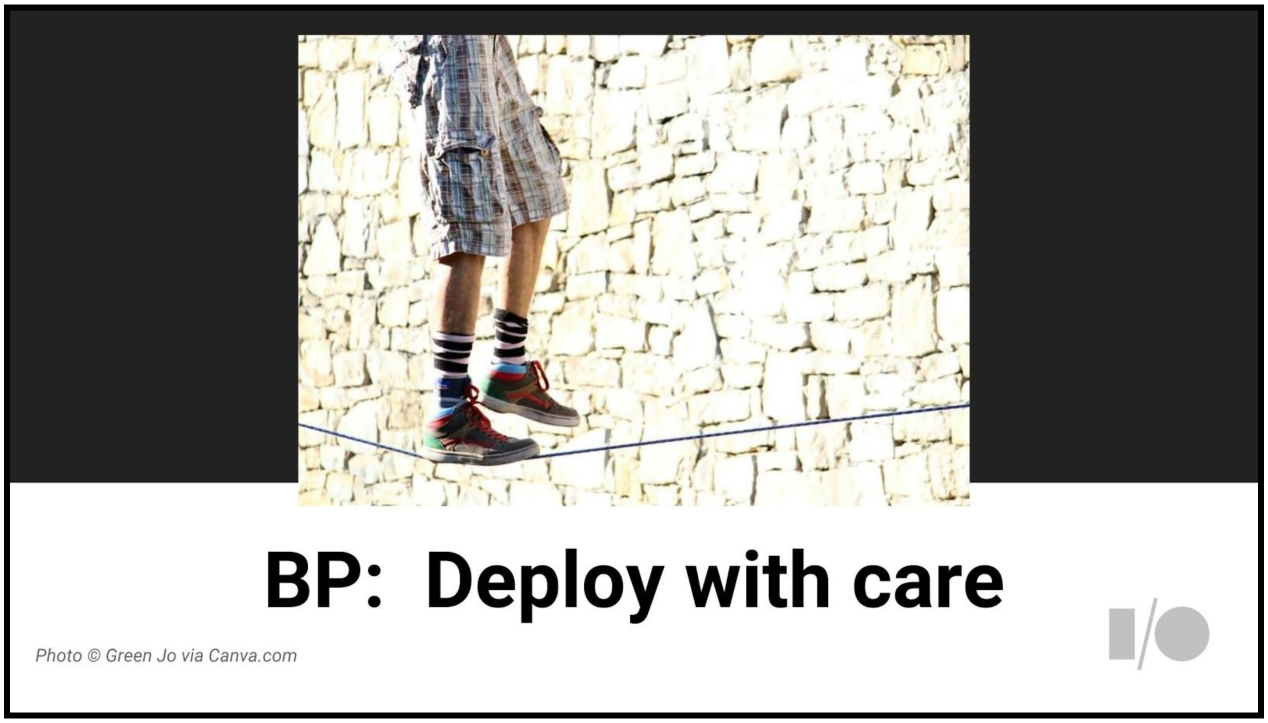
Photo © hjalmeida via Canva.com

Now I'll be honest…

At this point, I thought I knew everything I needed to know about scalable application in the cloud.   I knew how to work on the app level: by making application more deployable and wrapping them into containers. I knew how to work at infrastructure level: by monitoring and automating everything and by using open APIs to avoid vendor lock in...

What was there even left for Kelsey to teach me? And then he asked me about code updates.

What happens if the new intern, Bob, is watching tests past with his blindfold on. Again.

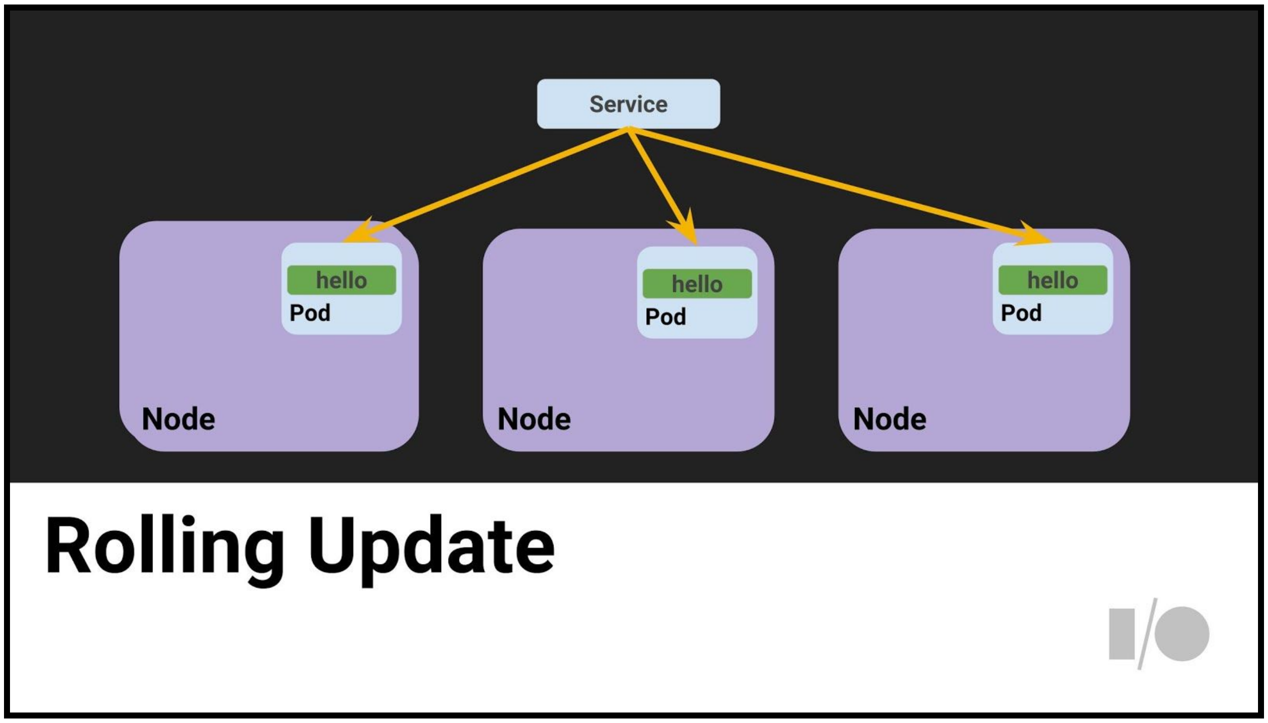


BP: Deploy with care

Photo © Green Jo via Canva.com

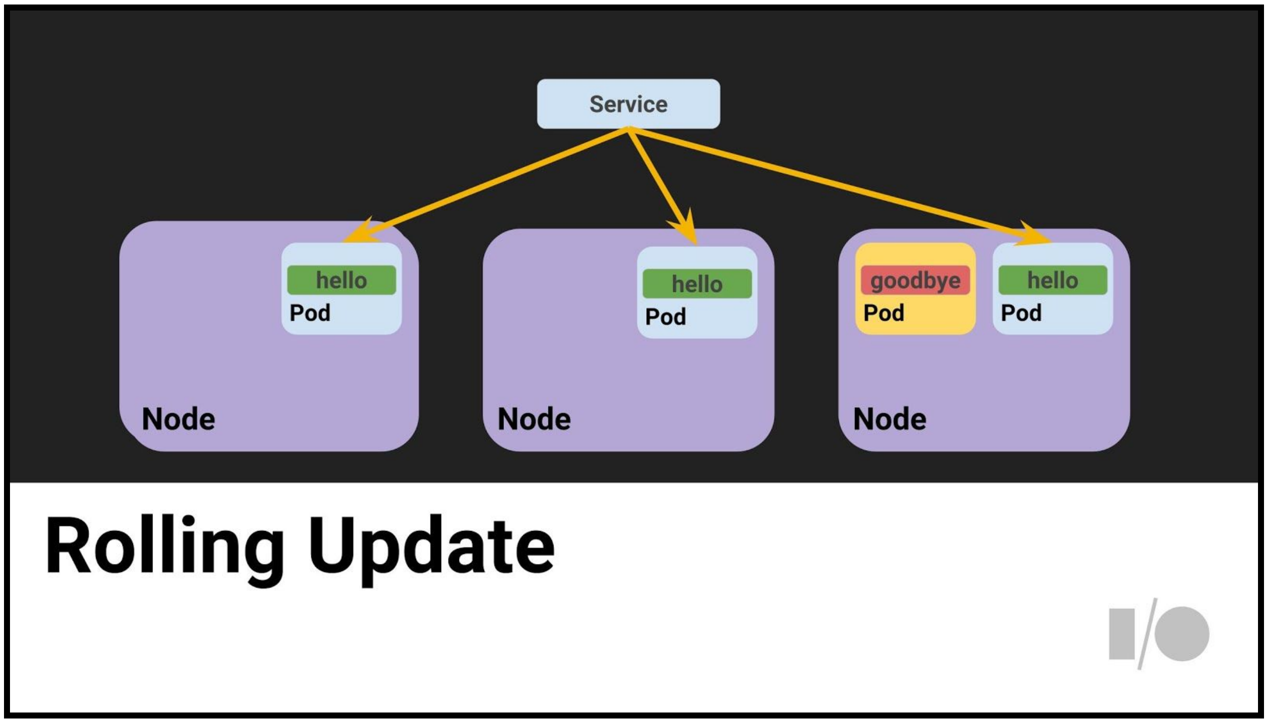
This is the last hurdle: the Wild. Because inevitably, we're going to have to rollout and rollback updates to code, in production at some point. How do we do this while minimizing (or ideally eliminating) down-time? The answer is we need to have a story that lets us cautiously roll out updates and quickly roll them back in case they fail for whatever reason. Patterns have developed in this space like Blue/Green and Canary deployments.

But let's see what Kubernetes offers.



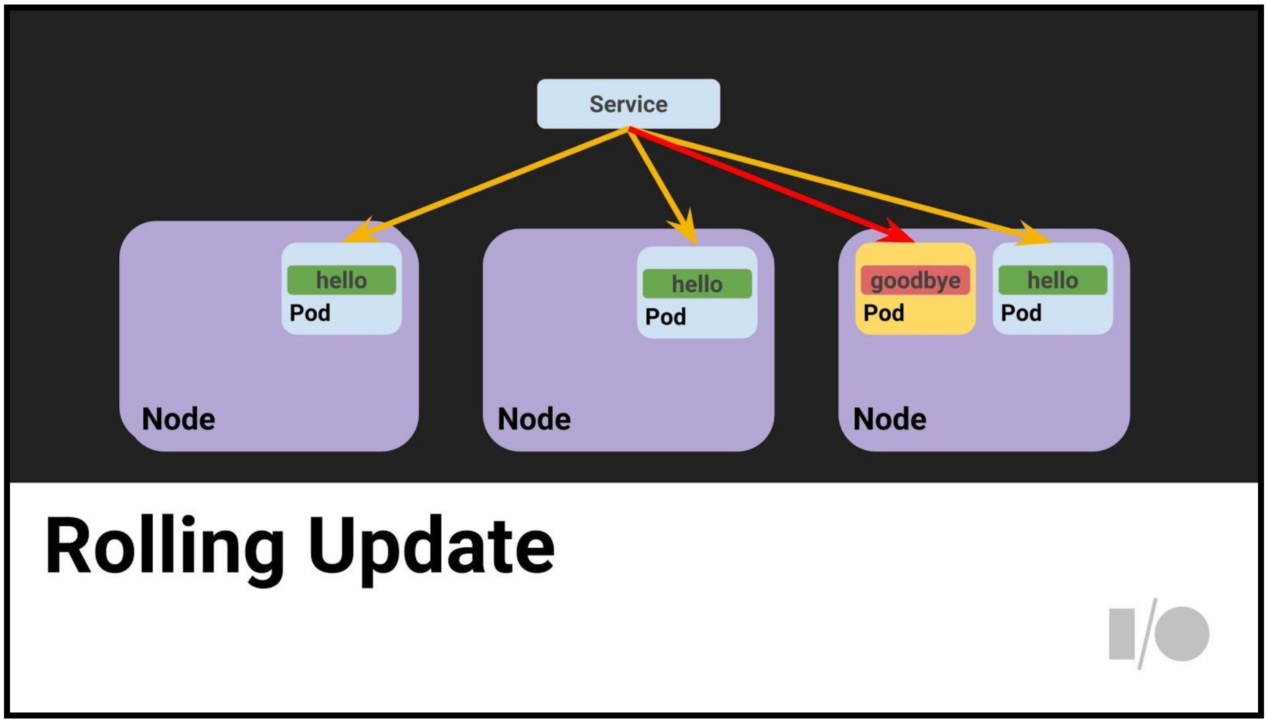
Rolling Update

One of the update strategies for Deployments are what's called rolling updates.   What rolling updates do is allow you to gradually scale up a new version of a pod and rollback quickly if necessary. Let's watch the how this scales up.



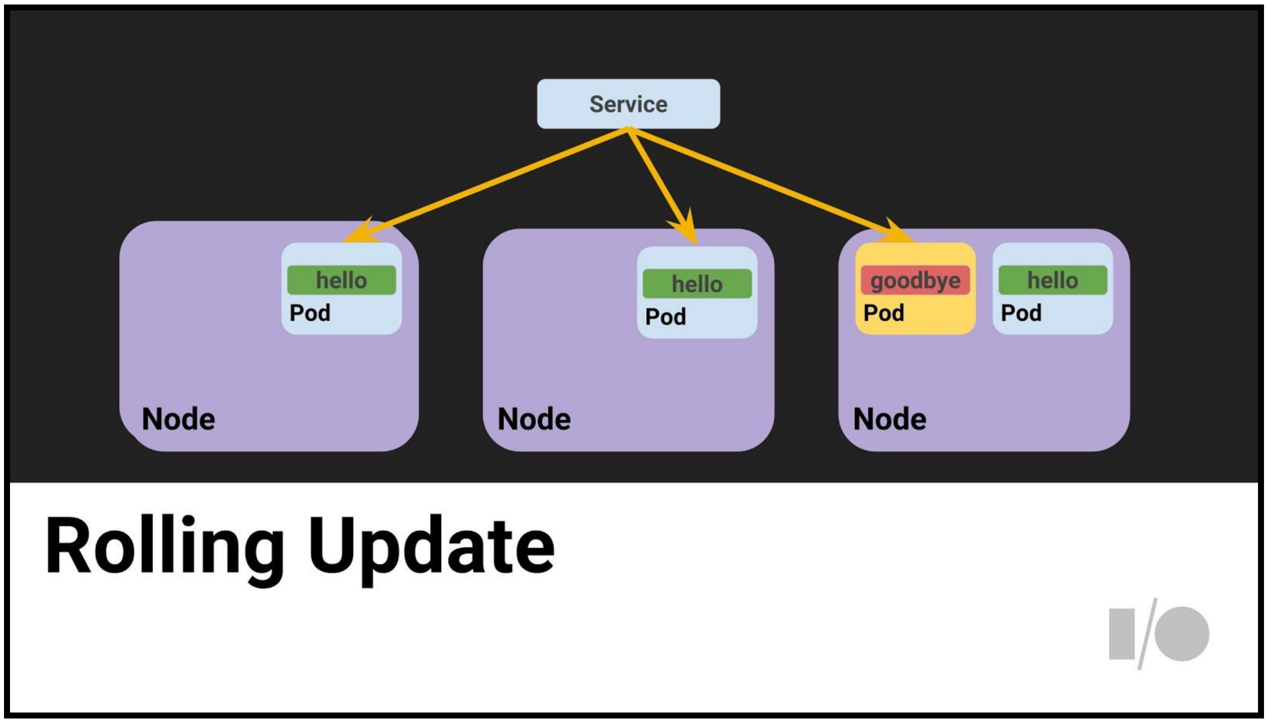
Rolling Update

…



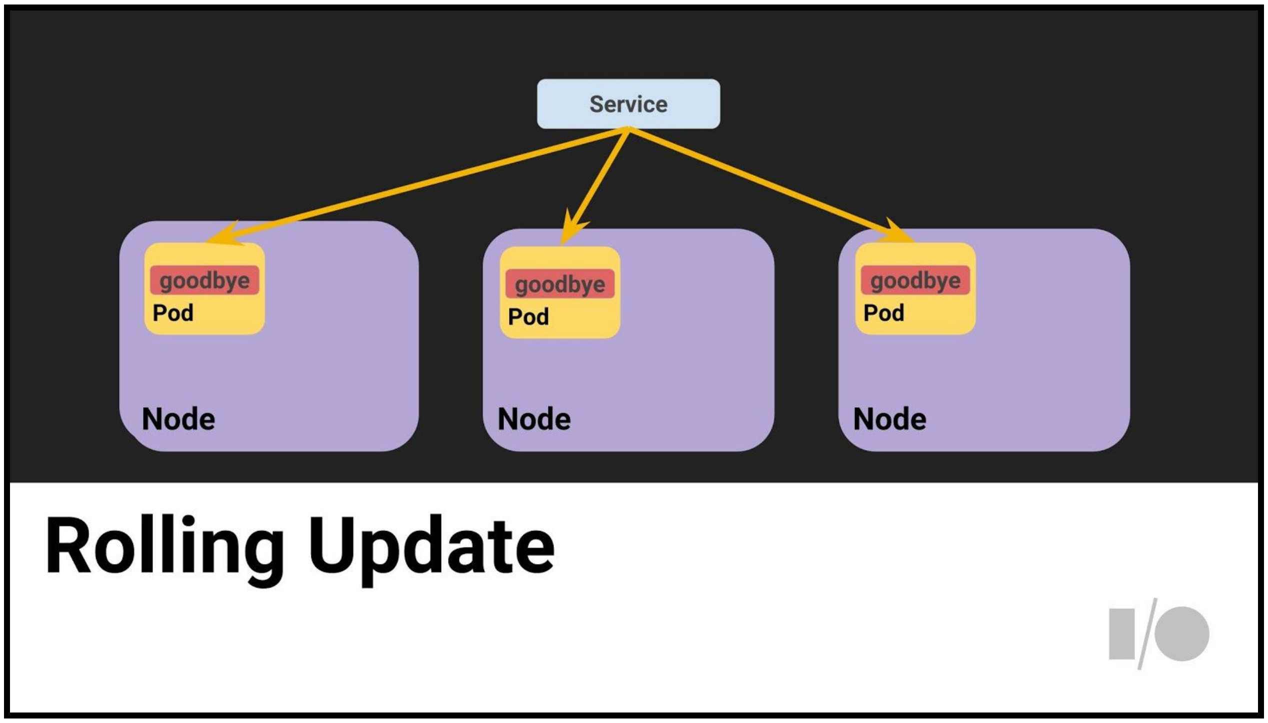
Rolling Update

[Slides omit]
..



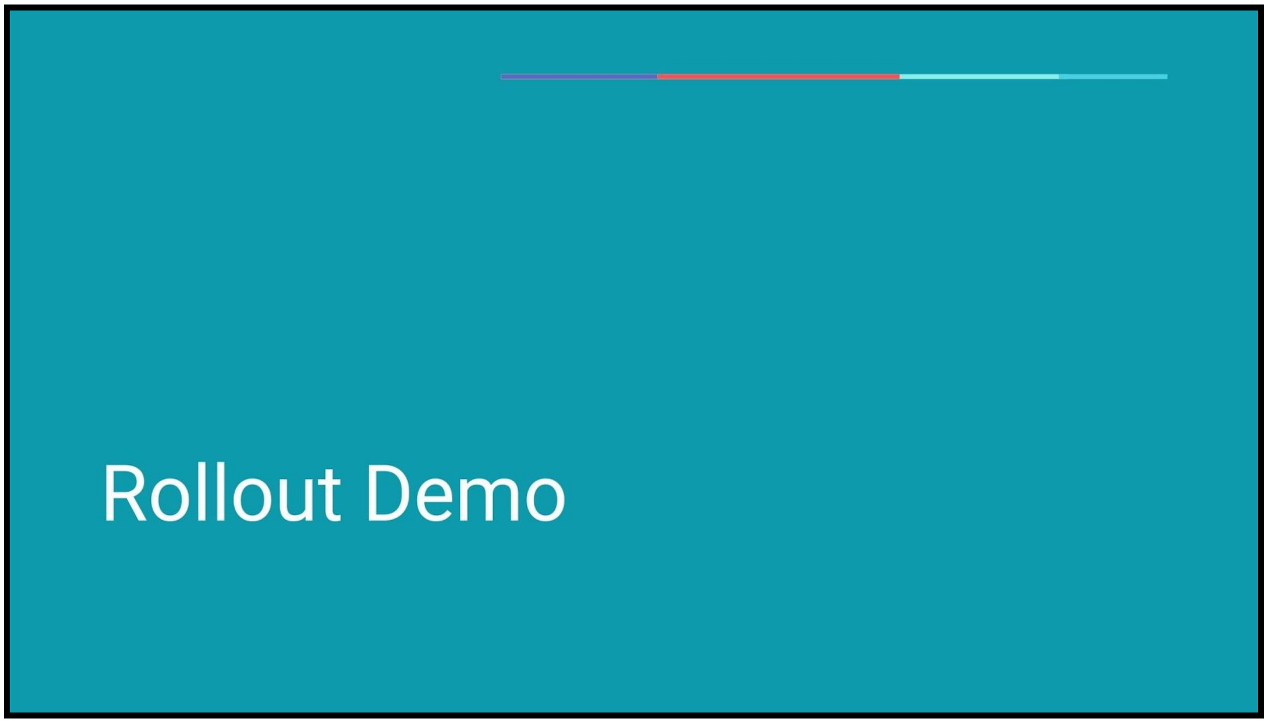
Rolling Update

…



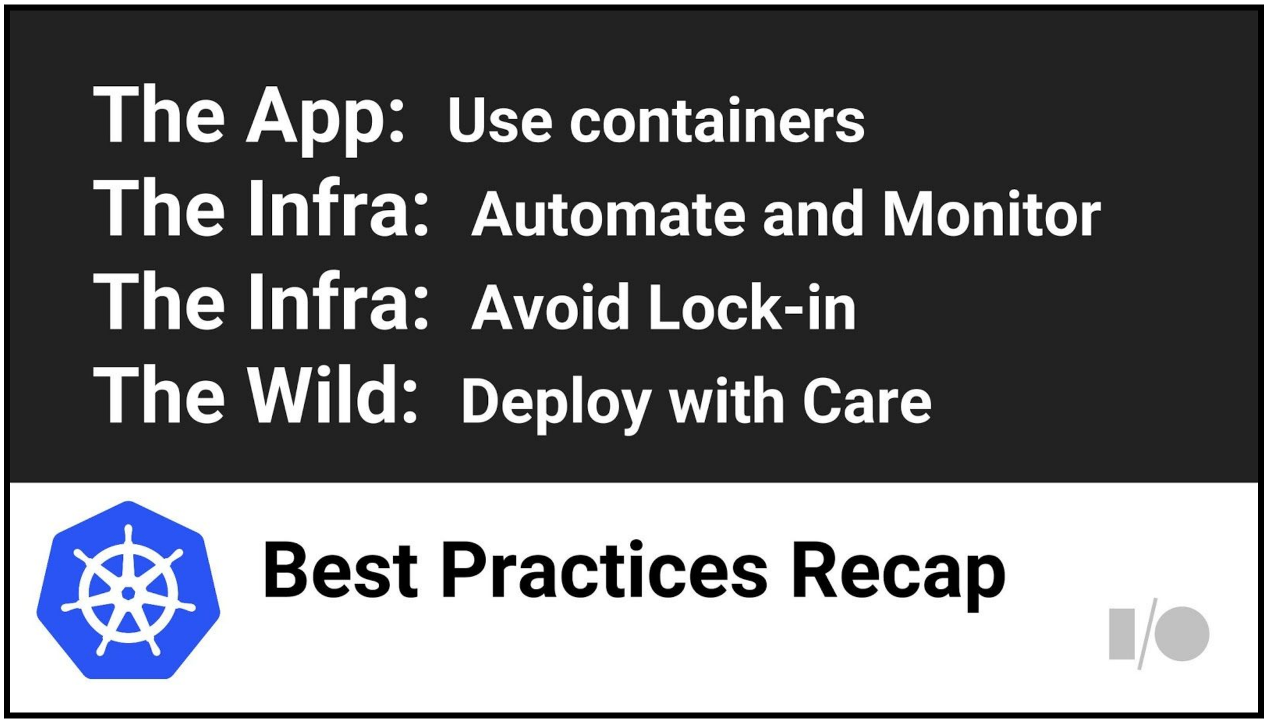
Rolling Update

And this continues until you have a full set of your updated application in production!



Rollout Demo

Code is here: https://github.com/askcarter/io16



The App: Use containers

The Infra: Automate and Monitor

The Infra: Avoid Lock-in

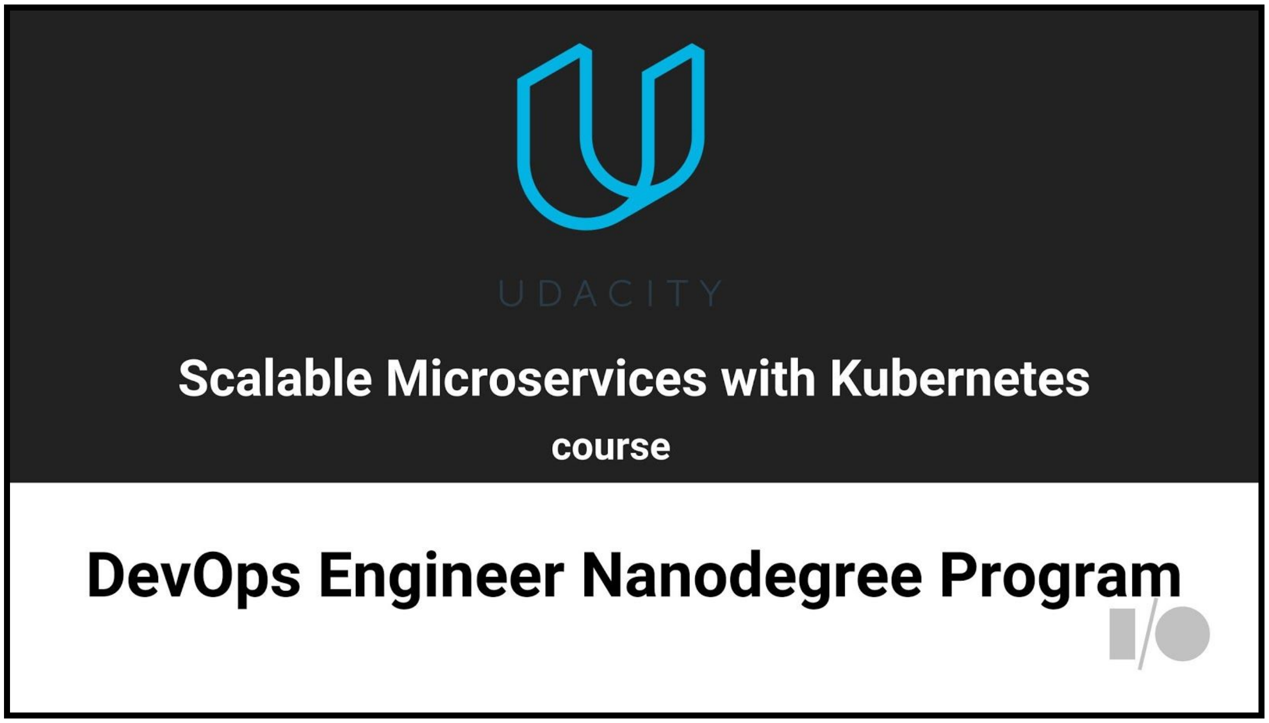
The Wild: Deploy with Care

Best Practices Recap

Now, I'll be honest....
At this point I thought I knew everything I needed to get started with managing scalable applications with kubernetes.

So let's recap
I knew how to handle the app level: containers. I knew how to handle the infra: ... . I knew how to handle the wild: ... .
What more was there to learn?

A lot. But that was a good start.



UDACITY

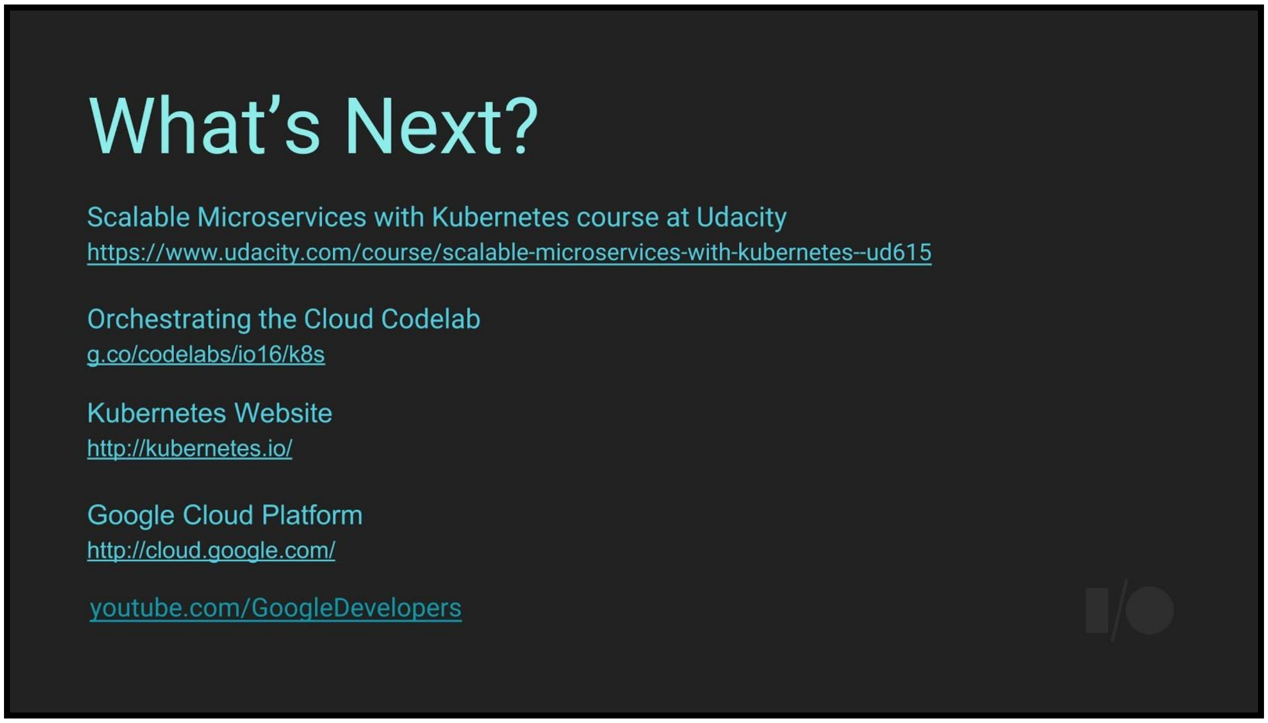
Scalable Microservices with Kubernetes

course

DevOps Engineer Nanodegree Program

And so -- When it came to learning Kubernetes -- I got lucky… I found a mentor on a bus. Now I don't recommend taking work advice from strangers on a bus, so instead if you want to learn more you should check out this the new Scalable Microservices with Kubernetes course Kelsey and I developed. It's part of Udacity's new DevOps Engineer Nanodegree Program made in partnership with Google and others and I'm really proud of it.

In the course you'll be able to hear about from leading industry experts like that guy over there and we even got Adrian Cockcroft to come teach me a little about the evolution on microservices and container infrastructure. So check it out!



What's Next?

Scalable Microservices with Kubernetes course at Udacity
https://www.udacity.com/course/scalable-microservices-with-kubernetes--ud615

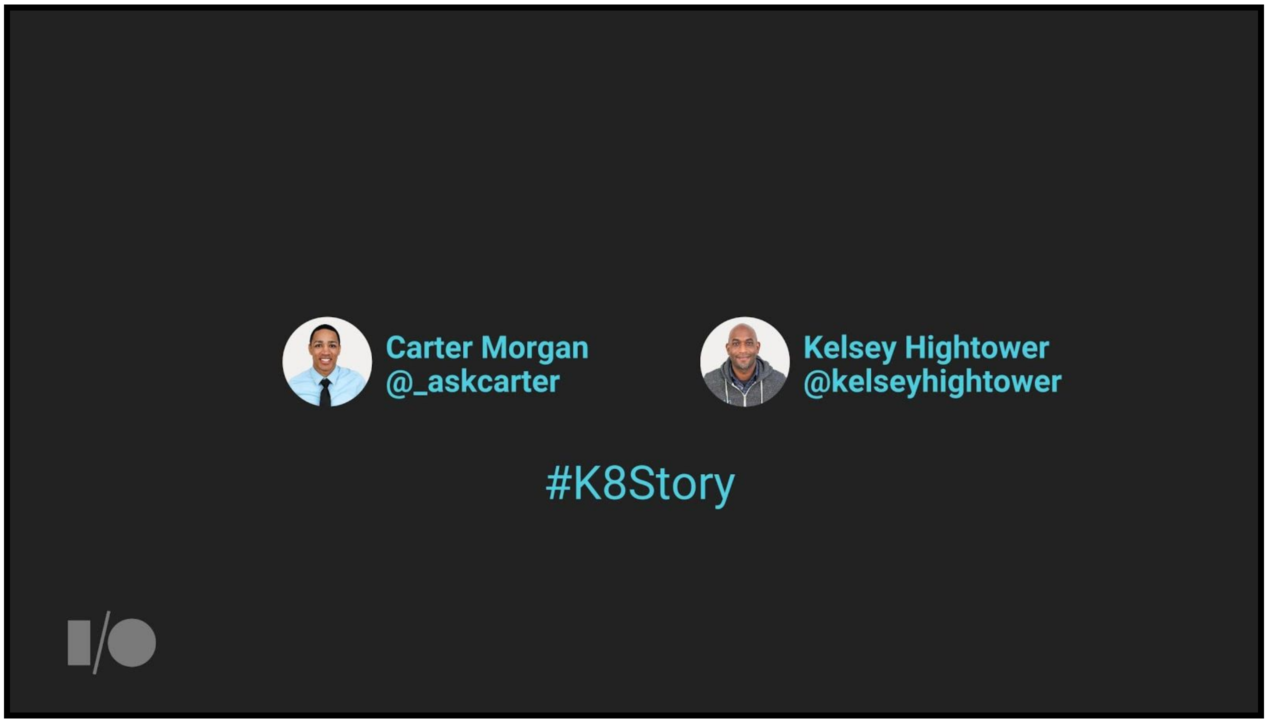
Orchestrating the Cloud Codelab
g.co/codelabs/io16/k8s

Kubernetes Website
http://kubernetes.io/

Google Cloud Platform
http://cloud.google.com/

youtube.com/GoogleDevelopers

There's also a Codelab that'll walk you through some of today's talk -- Google Cloud Platform and Kubernetes websites where you can go to learn more.



Carter Morgan
@_askcarter

Kelsey Hightower
@kelseyhightower

#K8Story

So, to wrap up, I want to say that was my Kubernetes story…

Kubernetes is an open source container automation framework that you tell your story. And I want to hear it.

So, if you have any questions or and stories to share -- hit me up at @_askcarter (or hit up the guy I learned all of this from @kelseyhightower) with the hashtag #K8story.

Thanks