KubeCon | CloudNativeCon

China 2018

Migration of an Enterprise UI Microservice System from Cloud Foundry to Kubernetes

Tony Erwin, Senior Technical Staff Member, IBM
Jonathan Schweikhart, Console DevOps Lead, IBM

Agenda

- Overview of IBM Cloud Console Architecture
- What is Cloud Foundry? What is Kubernetes? Why Switch?
- Experiences And Lessons Learned During Migration
- Conclusion

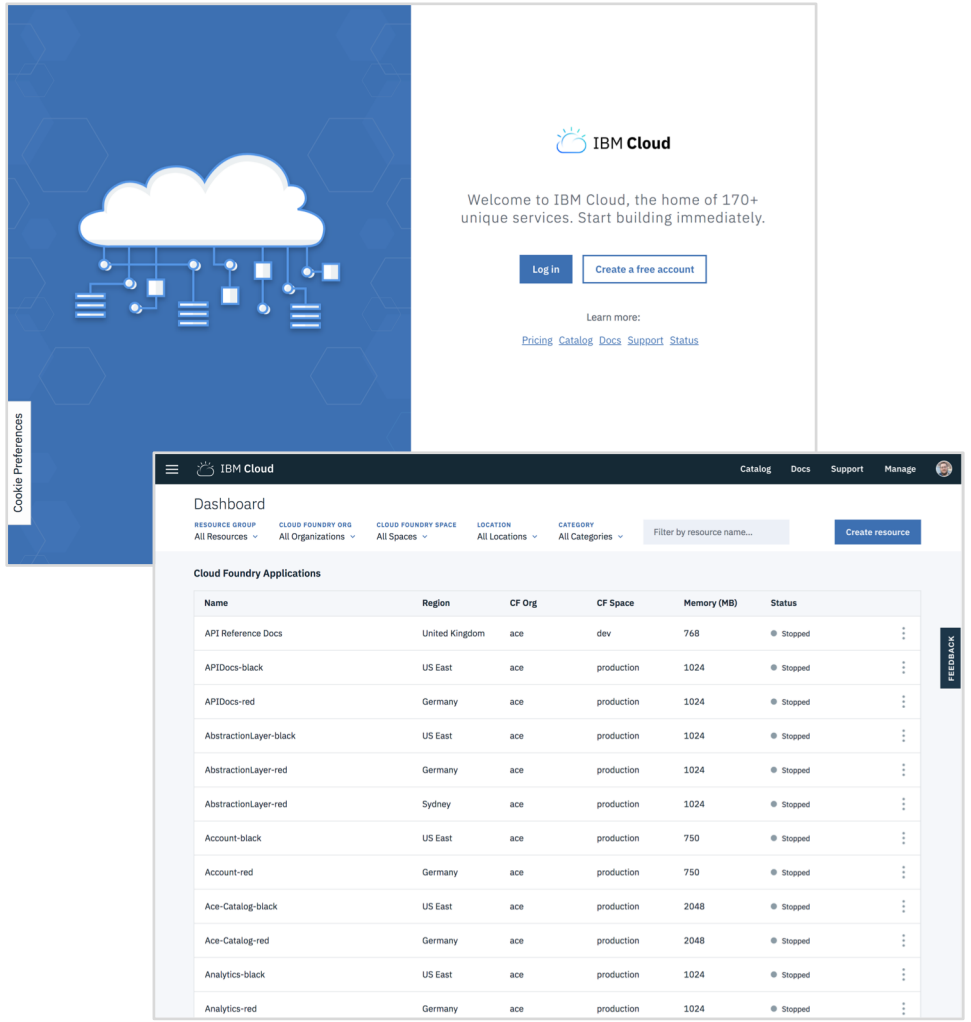
KubeCon | CloudNativeCon

China 2018

Overview of IBM Cloud Console Architecture

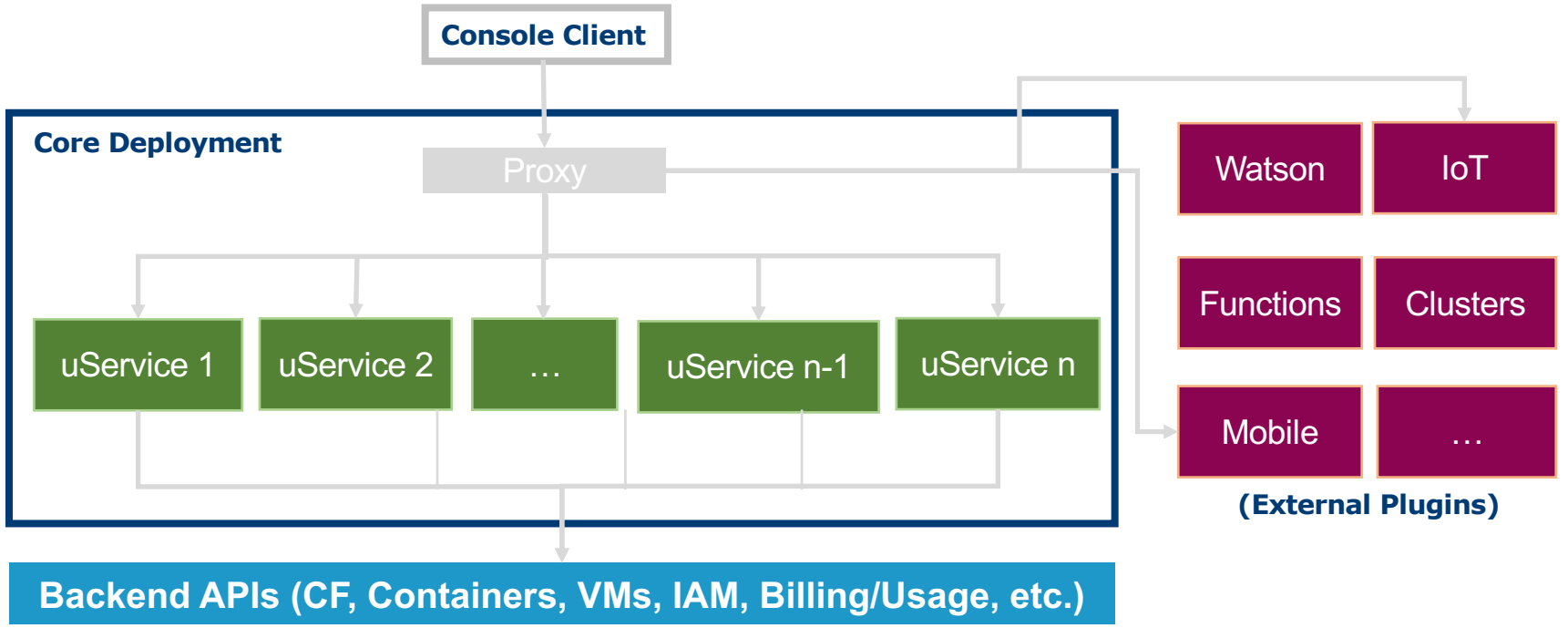
IBM Cloud Console

- Large UI serving as front-end to the IBM Cloud
- Lets users create, view, and manage PaaS/IaaS resources:
  - Cloud Foundry apps & services
  - Kubernetes clusters
  - Virtual servers
  - Bare metal
- Provides additional functionality for:
  - Registration/onboarding
  - Identity and Access Management (IAM)
  - Billing/usage
  - Docs



IBM Cloud Console Architecture

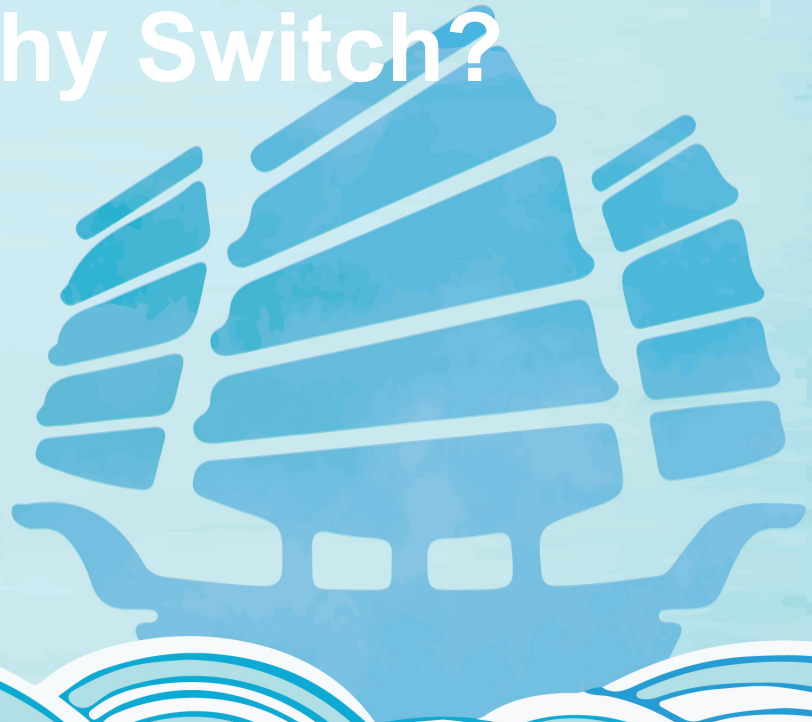
- Started life about 5 years ago as a monolithic Java app
- Now composed of around 50 Node.js, cloud-native microservices + more than 30 external plugins
- Originally deployed as apps to Cloud Foundry
- Currently deployed as containers on Kubernetes



KubeCon | CloudNativeCon

China 2018

What is Cloud Foundry?
What is Kubernetes?
Why Switch?



What is Cloud Foundry*?

- Provides a PaaS with an abstraction at the *application* level
  - Developers can focus on code rather than underlying infrastructure
- Leverages the Open Service Broker API to make it easy to use services from apps
- Manages apps as Diego containers (internally)

CLOUD FOUNDRY APPLICATION RUNTIME™

* Technically describing the Cloud Foundry Application Runtime which is one of the two open source components from the CF Foundation.

What is Kubernetes?

- Abstracts at the *container* level
- Provides many of the benefits of PaaS with the flexibility of IaaS
  - Often referred to as IaaS+
- Orchestrates computing, networking, and storage infrastructure on behalf of user workloads
- Enables portability across infrastructure providers

Why Did We Switch?

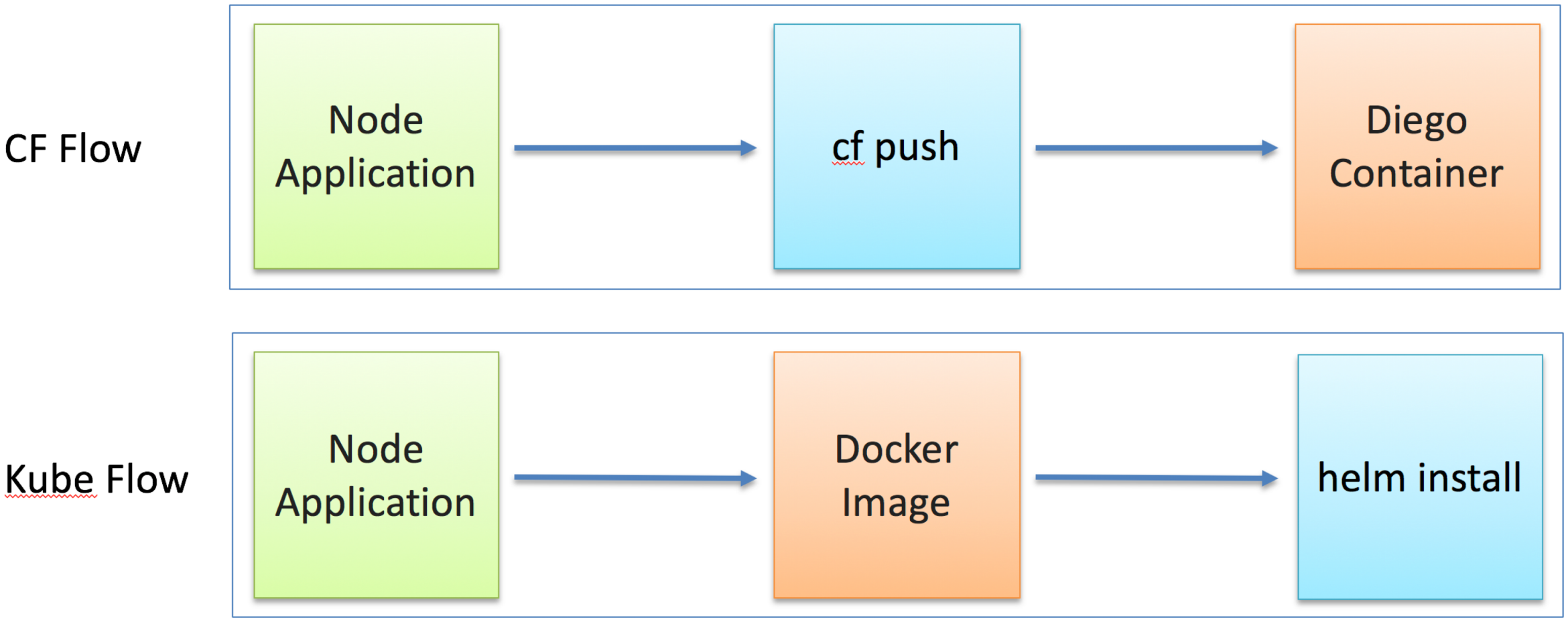
- Nothing “wrong” with CF
  - Very easy to get apps running, relatively low learning curve, etc.
  - Used in some way by at least half of the Fortune 500
- Kubernetes offers several advantages for our use case
  - More granular control to better manage our large, complex microservice system
  - Dedicated clusters to avoid performance/availability problems from friendly fire
    - In fairness, CF can be installed in a dedicated manner as well (even on Kubernetes!)
  - Simpler “front door” stack with built-in Ingress proxy to avoid extra network hops
  - Private host names
    - All apps *in* CF have public host names, so not possible to have a “private” microservice
  - Private networking
    - Calls between microservices in CF require going out over the public internet
  - Improved memory and CPU usage (dynamic allocation)
  - Ability to run our own local services (like Redis)
  - Integrated monitoring with Prometheus

KubeCon | CloudNativeCon

China 2018

Experiences And Lessons Learned During Migration

Need to Dockerize



Migrating Manifest to Helm

- Helm - Deployment
  - Docker image
  - CPU & memory
  - Environment variables
- Helm – Service
  - Single alias for the deployment
- Helm – Ingress
  - Hostname/URL mapping to service

Deployment Configuration

- Cloud Foundry
  - Configuration per deployment environment
- Kubernetes
  - Helm cli makes hierarchical simple
  - Global
  - Global-<Environment>
  - Cluster
  - Cluster-<namespace>

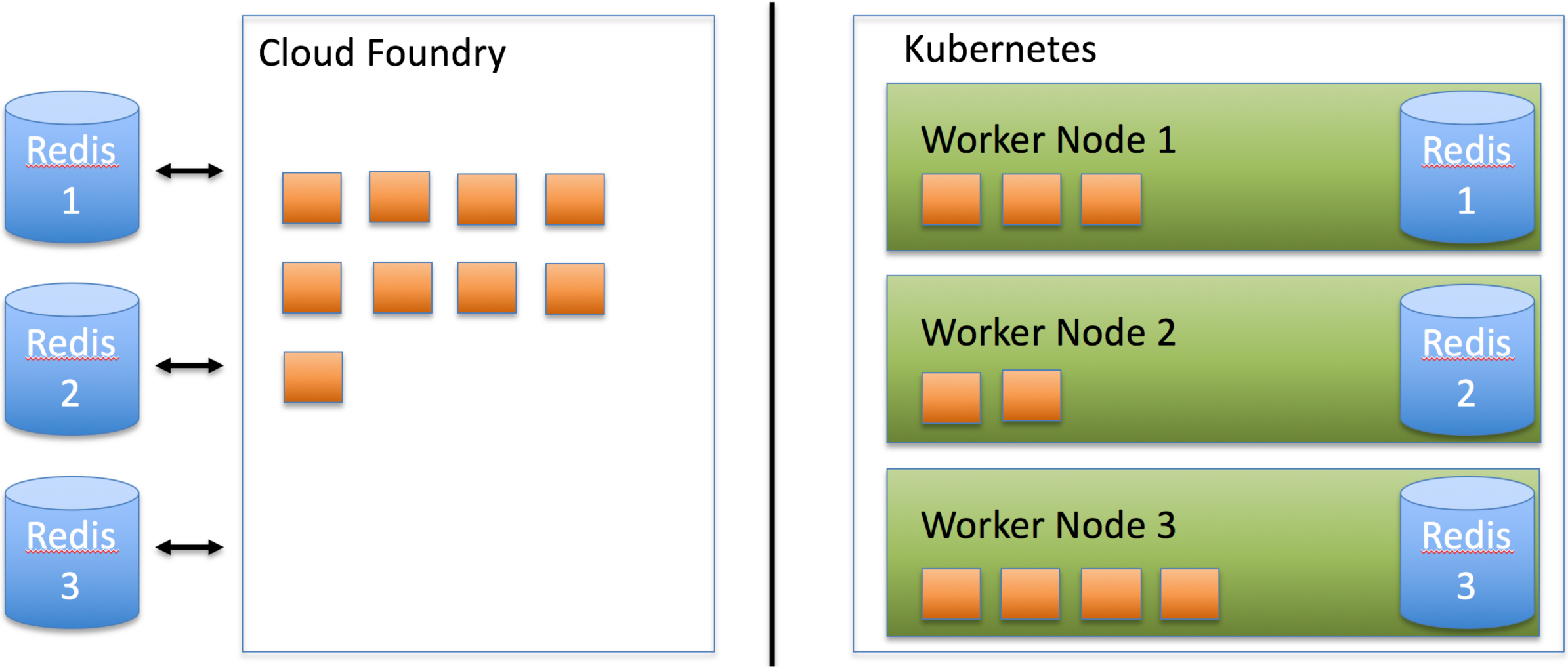
Exposure of Microservices

- Cloud Foundry
  - Public URL per microservice
  - Each microservice has to protect against direct access
    - Security concerns
    - Common code repeated
- Kubernetes
  - Microservice gets to choose exposure
    - Service – Allows an internal only route to the application
    - Ingress – Allows external routes to be defined to map to Services
  - Protections take place at a higher level to allow microservices to focus less on exposure issues

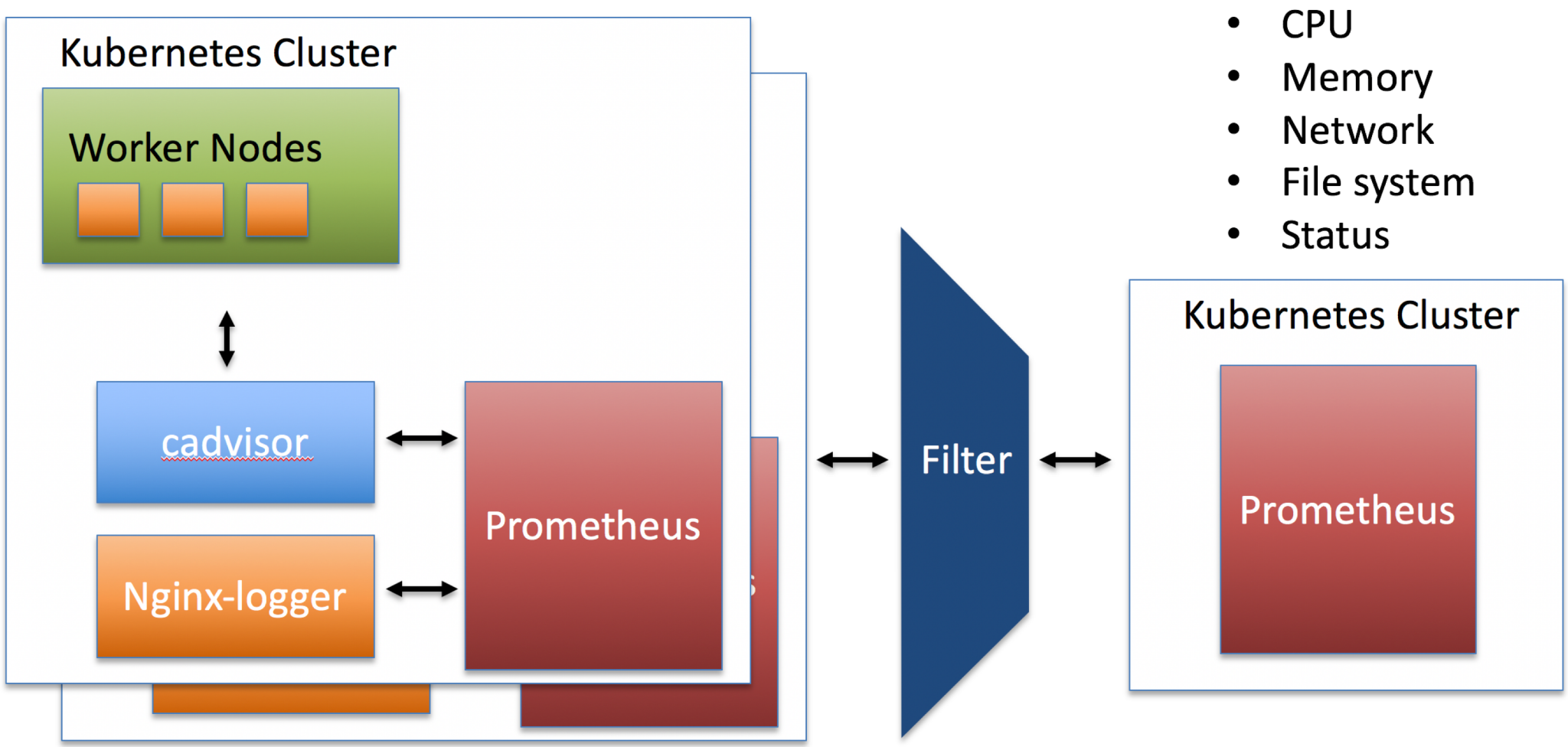
Common Code Migration Problems

- Cloud Foundry assumptions
  - Environment variable assumptions
    - VCAP_SERVICES
    - PORT
    - Invalid OS name characters like hyphens
  - URL format for intra-microservice communication
    - CF: https://ace-common-production.us-south.bluemix.net
    - Kubernetes: http://common
    - URL construction vs URL variables

Installing a Local Redis w/ Stateful Sets



Monitoring in Kubernetes

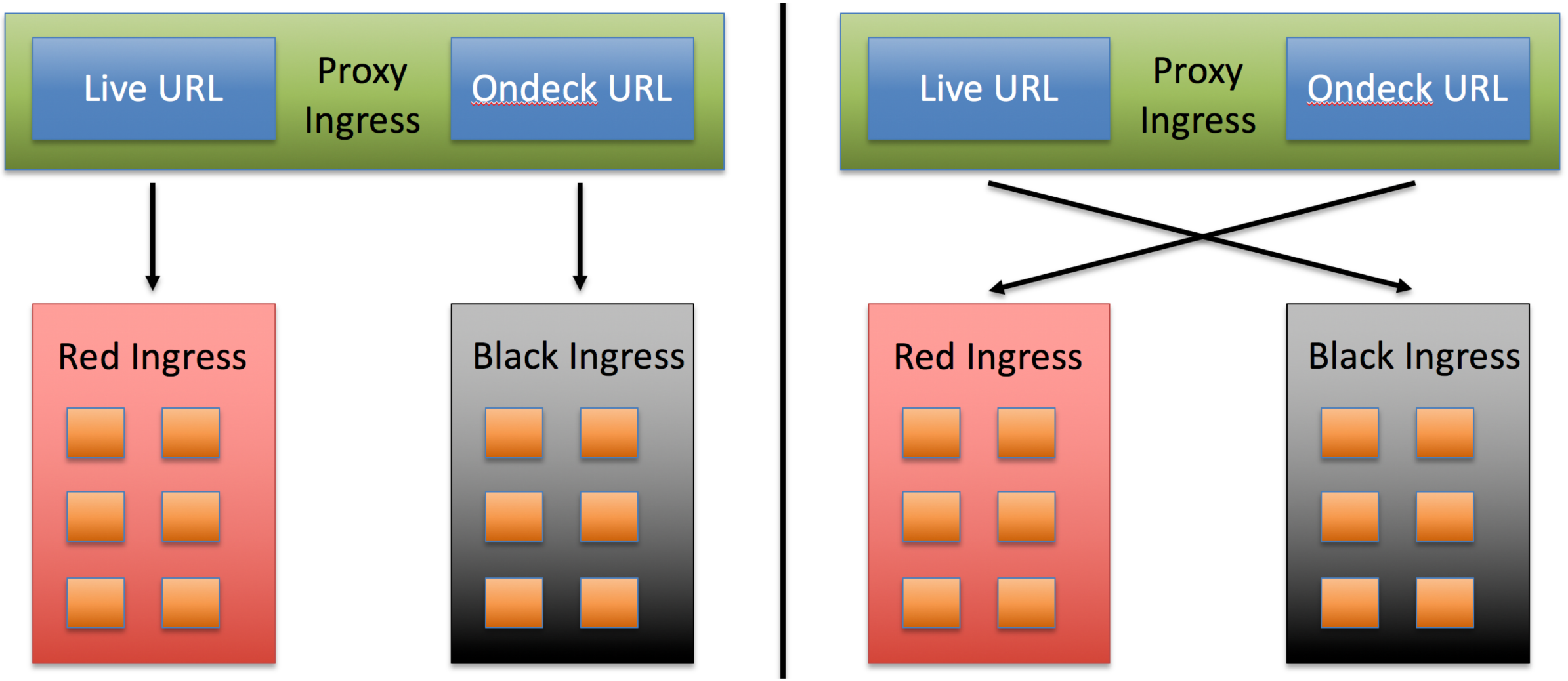


- CPU
- Memory
- Network
- File system
- Status

Monitoring NGINX Ingress

- Nginx logs contain invaluable metrics about incoming calls
  - Timestamp
  - HTTP method
  - HTTP status codes
  - Headers
  - URI
  - Response time
- Implemented custom solution for accessing those metrics
  - Configure nginx to log to syslog
  - Create microservice that scrapes the syslog and exposes the data to Prometheus
  - Filter, monitor, and alert

Red/Black Deployments



Built-in Liveness/Readiness Checks

- /readiness
  - ”I am ready to accept traffic”
  - One time initialization checks
    - Connections to resources (URLs, DBs, etc..)
  - Periodic checks
    - Circuit breakers
    - Current status
    - Content Throttling
- /liveness
  - “I should keep living“
  - Unrecoverable situations/Unexpected Failures
  - “Have you tried turning it off and on again?”

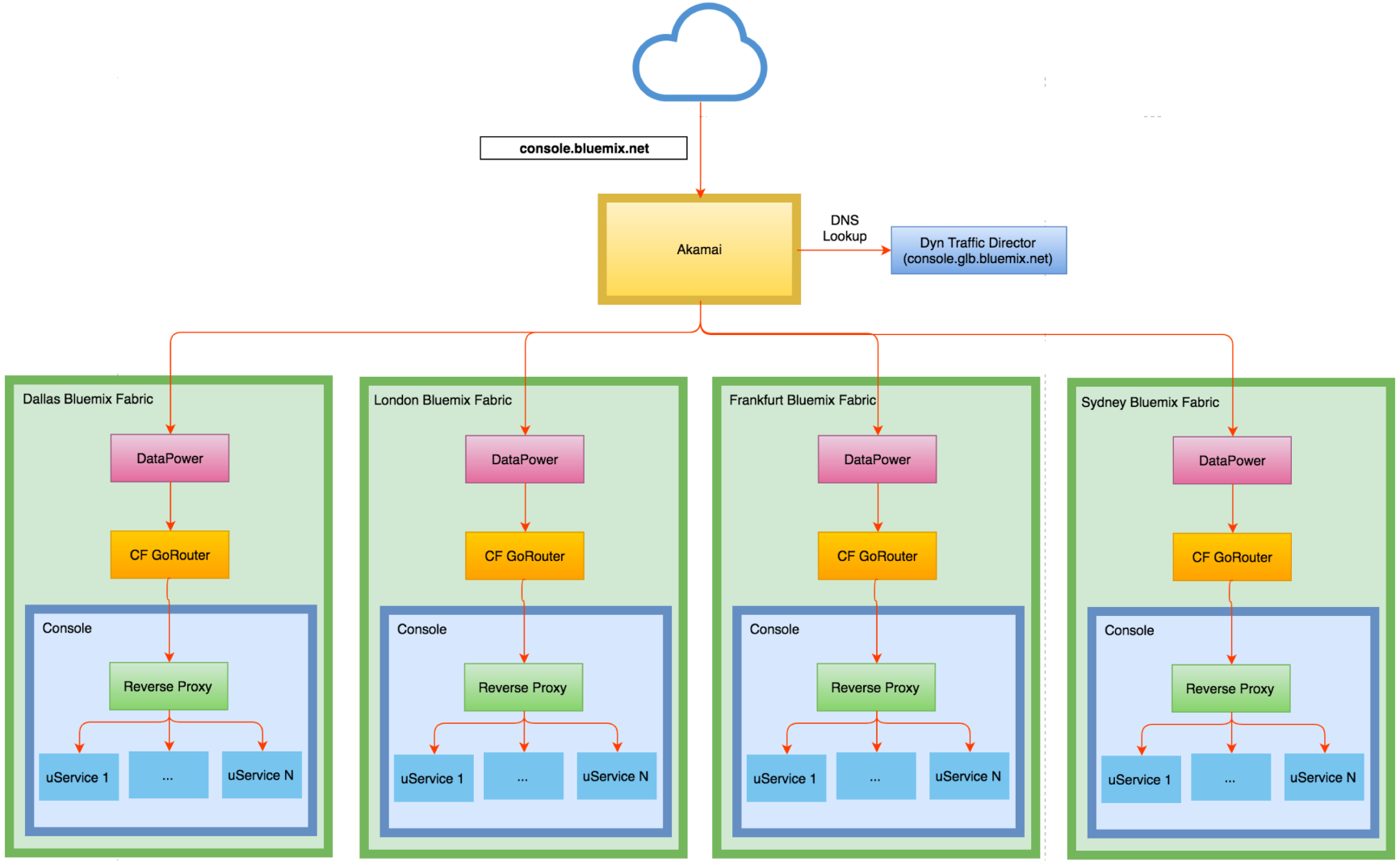
Rolling Out Kubernetes

KubeCon | CloudNativeCon

China 2018

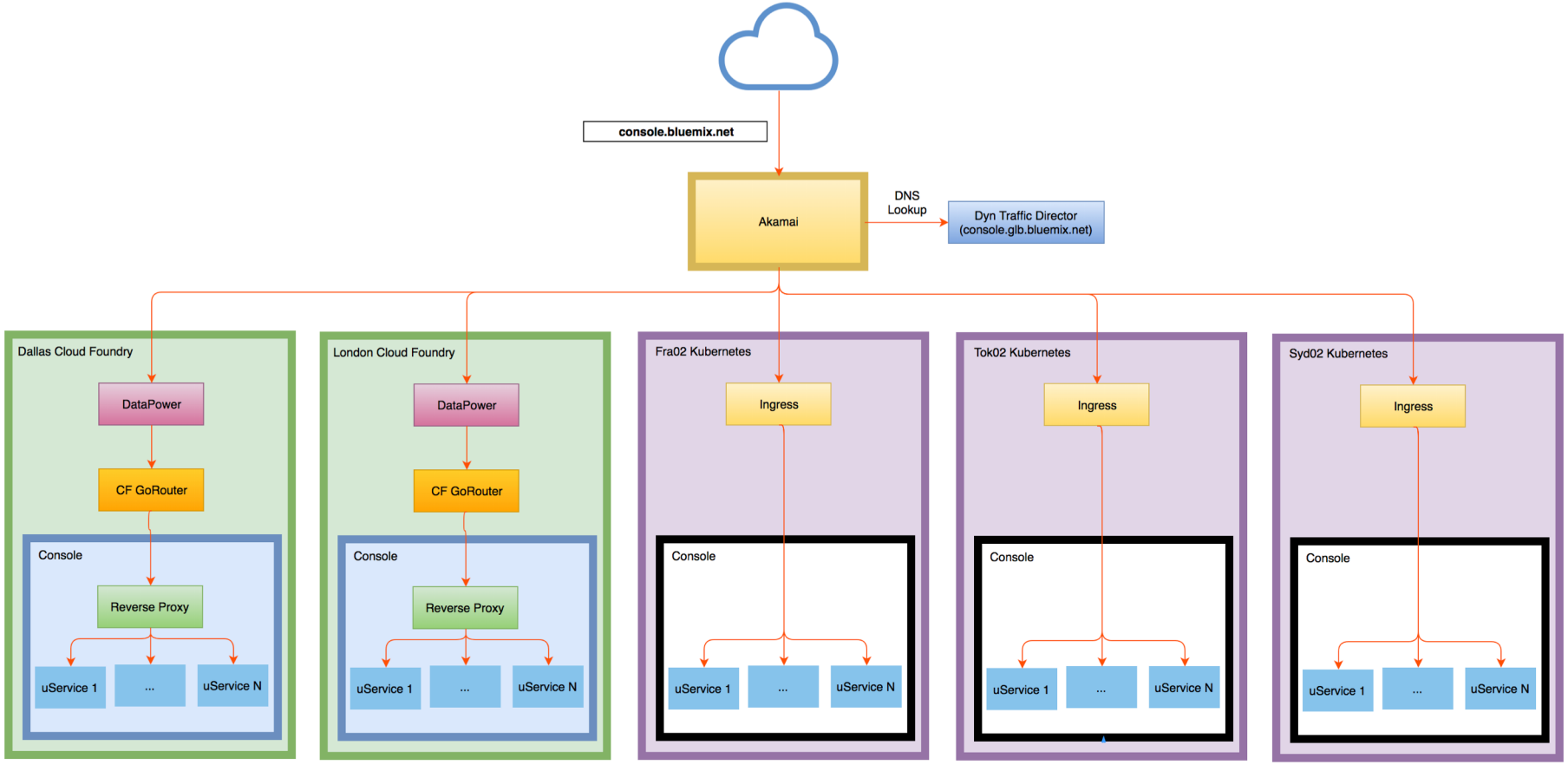
Geo Load Balancing and Failover (CF)

- One global URL (https://console.bluemix.net)
- Use Dyn geo load balancing to serve UI from the nearest healthy region
- If healthcheck in a region shows a problem, Dyn routes to the next closest healthy region
- Odds of all regions being down at the same time much less than one region being down
- Reduces regional latency



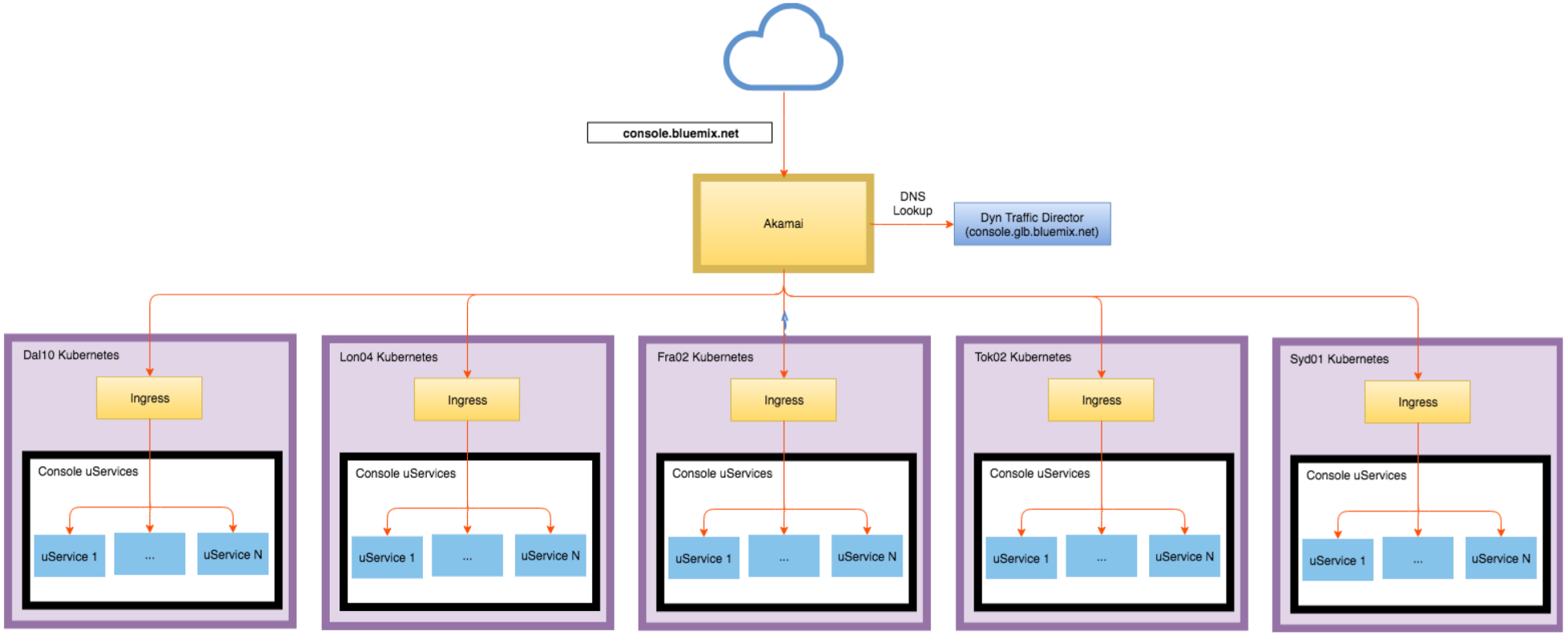
Geo Load Balancing and Failover (Migration)

- Needed to verify stability of Kube clusters before turning off CF deployments in production
- Solution: Add Kube clusters to Dyn rotation and run CF deployments side-by-side with Kube deployments



Geo Load Balancing and Failover (Final)

- Once satisfied, removed CF deployments from rotation and only Kube deployments remained



Conclusion

KubeCon

CloudNativeCon

China 2018

Conclusion

- CF is a great technology, but Kubernetes better meets the needs of our microservice system
- Nothing is free, and we had to solve several new problems along the way
- Allowed us to achieve greater performance, scalability, reliability, and security than we had before

Questions?

- Tony Erwin
  - Email: aerwin@us.ibm.com
  - Twitter: @tonyerwin

- Jonathan Schweikhart
  - Email: jschweik@us.ibm.com

The End

KubeCon

CloudNativeCon

China 2018