CITRIX

CLOUD NATIVE COMPUTING FOUNDATION

8x8

Migrating Real-Time Communication Applications to Kubernetes at Scale: Learnings from 8×8's Experience

Pankaj Gupta, Citrix
Lance Johnson, 8x8
Michael Laws, 8x8

PankajOnCloud

AUGUST 12, 2020

Microservices-Based Applications for Business Continuity

Read “A Pandemic Plan for Application Architecture” at https://tinyurl.com/**K8sApp**

Modular App Architecture

Break application in small services, which can be developed, deployed and auto-scaled **independently**

Faster Releases

Very frequent application updates (even many times a day)

Continuous delivery to automate deployment

Portability

Self-contained runtime

Application portability across clouds

Diverse Stakeholders Have Unique Needs

DevOps

Faster release & deployment cycles
CI/CD & automation
Canary & progressive rollout

Developers

User experience
Troubleshooting
Microservice discovery & routing

Platform Team

Platform governance
Operational efficiency
Developer agility

SRE

Application availability
Observability
Incident response
Postmortems

NetOps

Network policy & compliance
Manage, control & monitor network
Resources & capacity planning (visibility)

DevSecOps

Application & infrastructure security
Container security & API gateways
Automation

Microservices App Delivery Solution Principles

Read “7 Key Considerations for Microservices-Based Application Delivery” at https://tinyurl.com/**7Consideration**

Architecture Flexibility

Move at speed of your IT skill set
Balance benefits & complexity

Integrated With Your Platform & Tools

Get apps to production fast with vast K8s platform & opensource tools support

Performance & Scale

Support large clusters & very dynamic microservices

Consistent App & API Security

Break the silos of monoliths & microservices

Complete Observability Stack

Gain visibility & troubleshoot problems faster, Break the silos

Production-Grade Solution at the Speed of Business


8x8
Communications.
Transformed.
DevOps “Three Body Problem”

8x8 Introductions

Lance Johnson, Director of Engineering, Cloud R&D
https://twitter.com/johnsonrl
https://github.com/rljohnsn
https://www.linkedin.com/in/lancejohnsn/

Michael Laws, Sr. Site Reliability Engineer @8x8
https://www.linkedin.com/in/mikelaws/


BASH
CHEF STYLE
DEVOPS
KUNG FU
I Voted
CHEF
WHIPPING UP AWESOME
HyperL
docker
HELM
amazon web services
8x8
GitLab
CODE CAN DEPLOY
Code
CHEF
CHEF.IO
$ helm yeah
HAWAII
FEST 2015
INSPEC
aws
OCEAN ORIGIN
CHEF CONF
2014


8x8
All-in-One
Communications
A single platform that combines cloud phone, video collaboration, team chat, contact center and analytics.

8x8 At a Glance

NYSE: EGHT

1,600+ Employees in 8 Countries

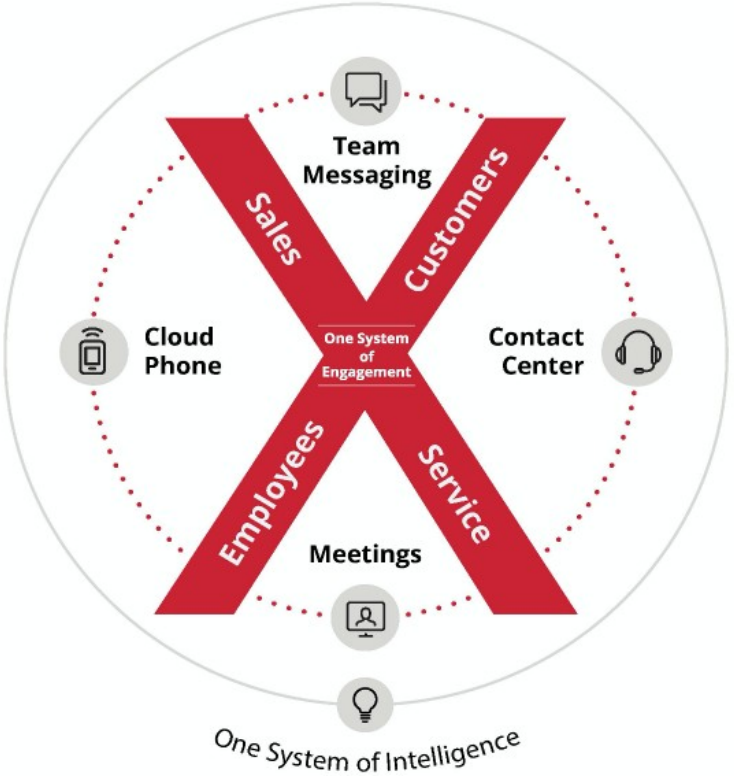
1+ million business users in 150+ countries

Guaranteed call reliability, quality

Leading security and compliance

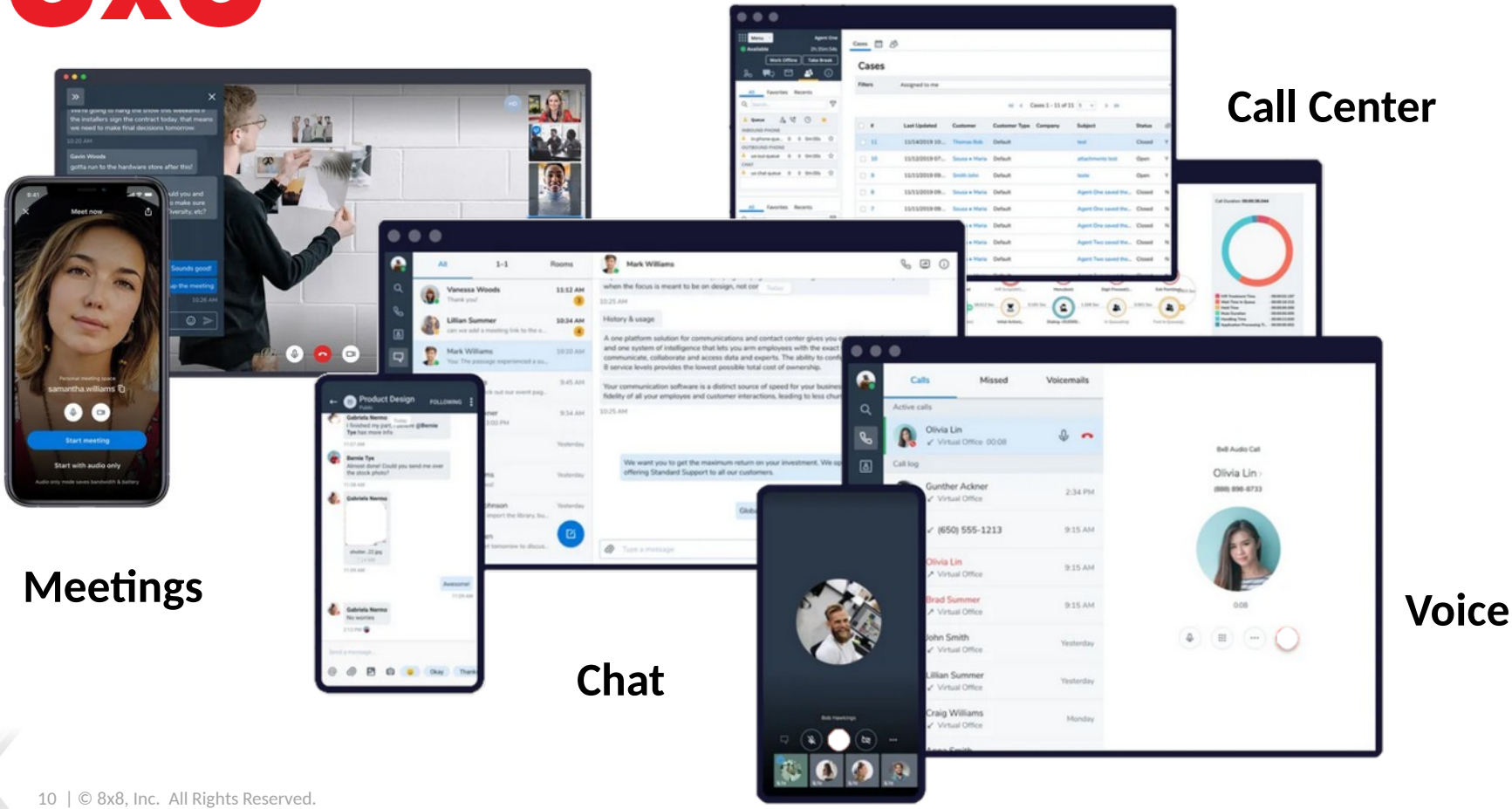
Global capabilities

Proven, trusted partner



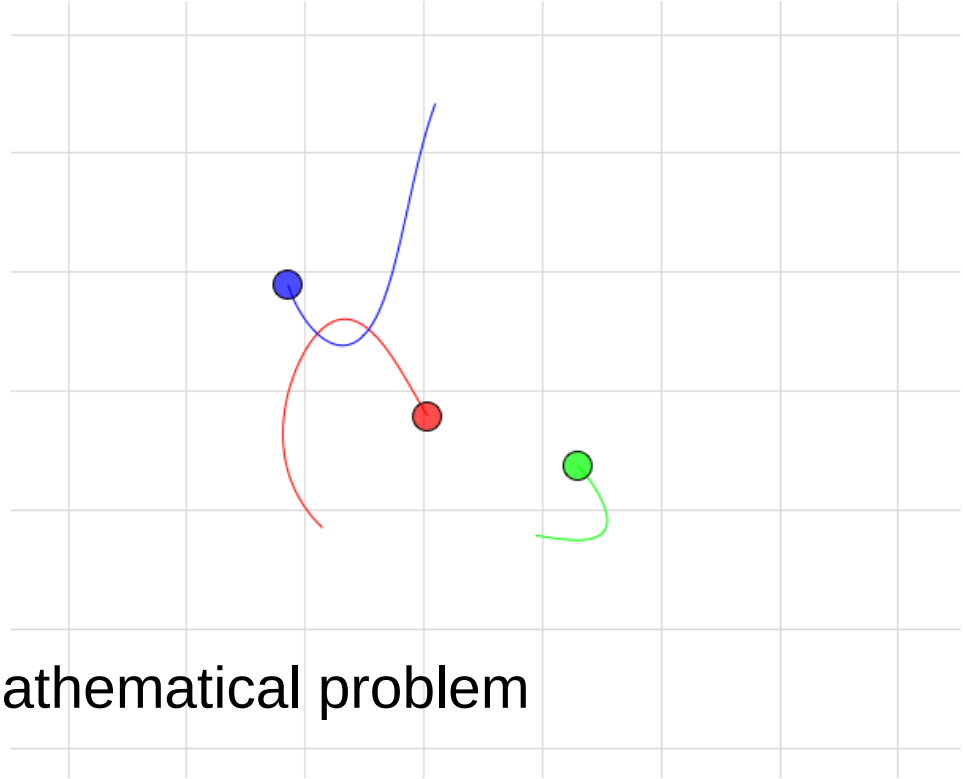
8x8

8x8 Unified Communications



8x8

The Three Body Problem



Unsolvable mathematical problem

DevOps Three Body Problem

Deployments

- Ssh
- Jenkins Pipelines
- HyperLoop
- Ansible
- Chef
- Terraform
- Cloudformation

Customers

- DNS
- LTM
- GTM
- Route53
- Geo
- Proxy
  - CloudFlare
  - Ambassador/Envoy

Dependencies

- Data Centers
- Clouds
- Files
- Databases
- Random VPC
- Public
- Private

How did 8x8 approach the problem?

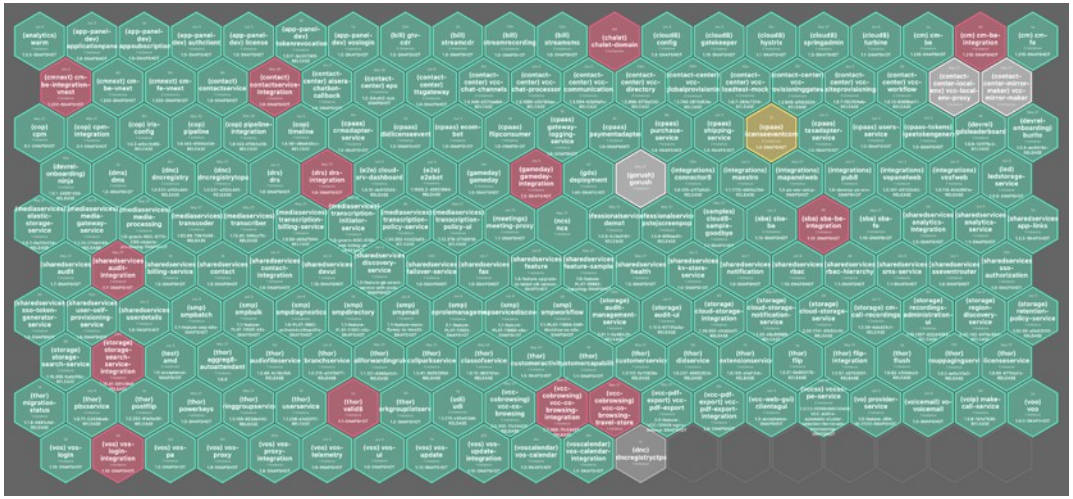

GANGNAM STYLE

8x8 DevOps Fundamentals

- **Safety**
  - Provide a way for changes to be validated and tested.
- **Automation**
  - Deliver the changes the same way for any environment
- **Knowledge Sharing**
  - Learn from each other with open collaboration
- **Autonomy**
  - Immediate visibility that changes deployed are safe
  - Continuously verified for every change

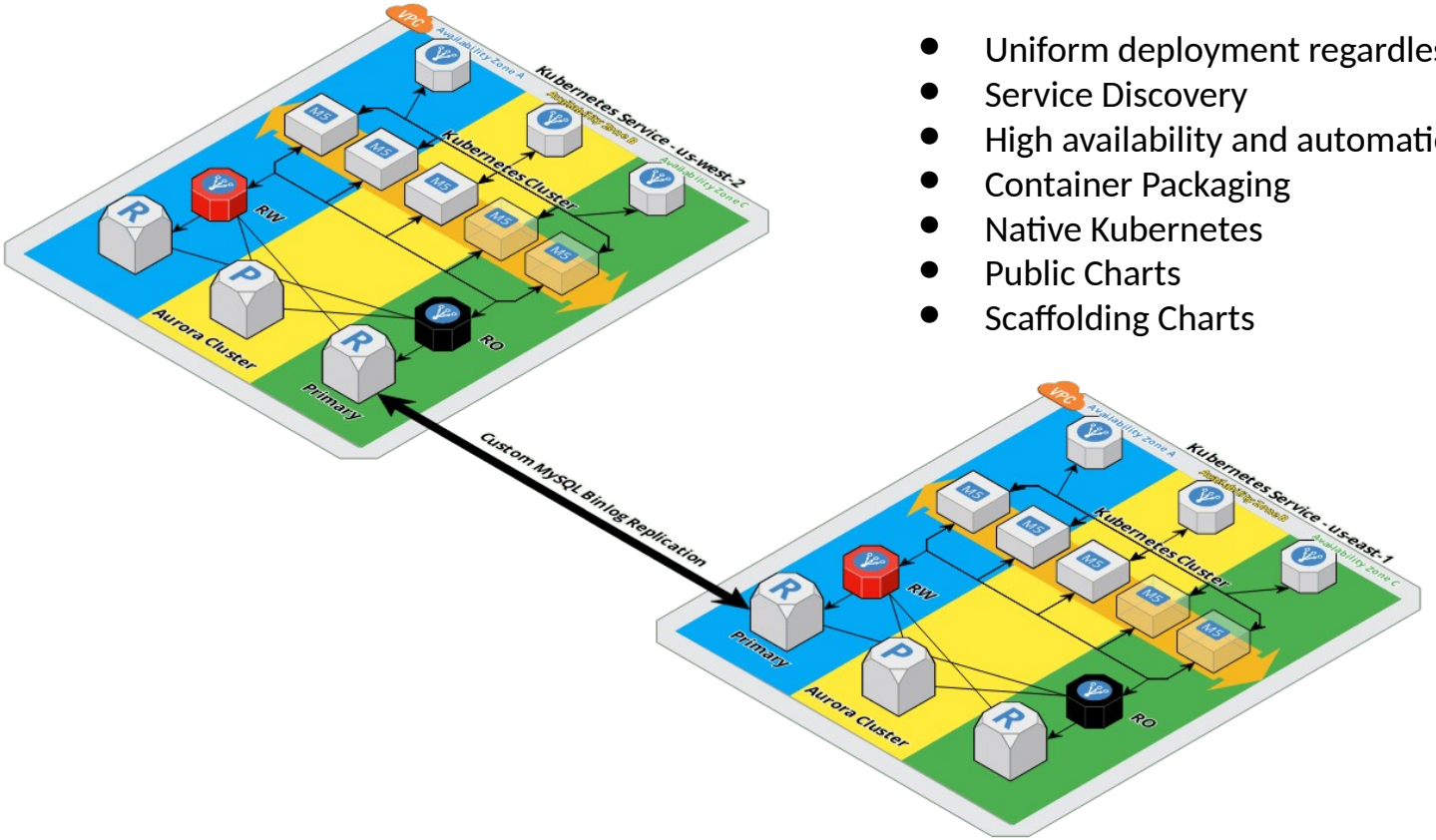
8x8 Adoption of Microservices: 100s of Them!

- Why
  - Divide and conquer
  - Technology refresh
  - Team Autonomy



- Which
  - Foundational Framework services (Cloud8)
  - Shared Services
  - Stateless Application Services

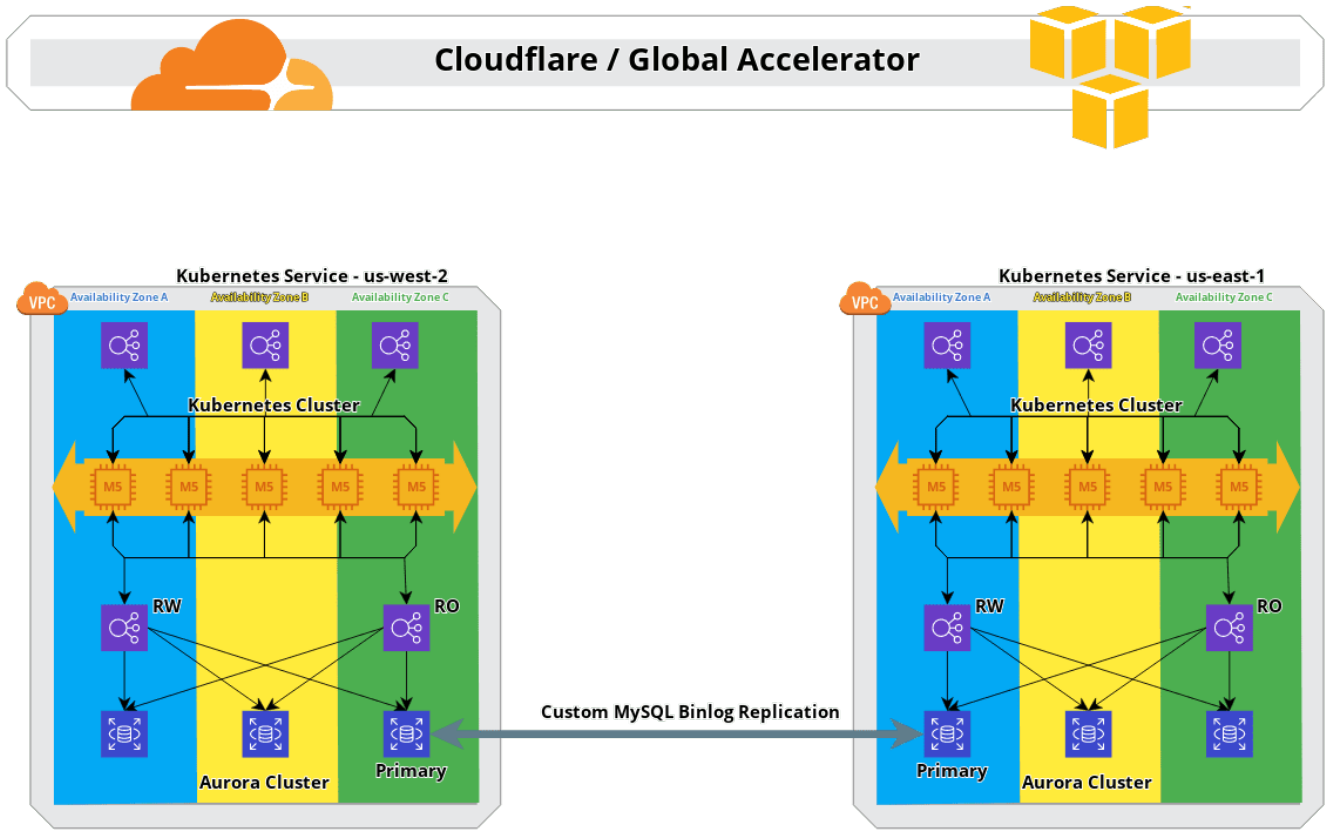
8x8 Design Goals



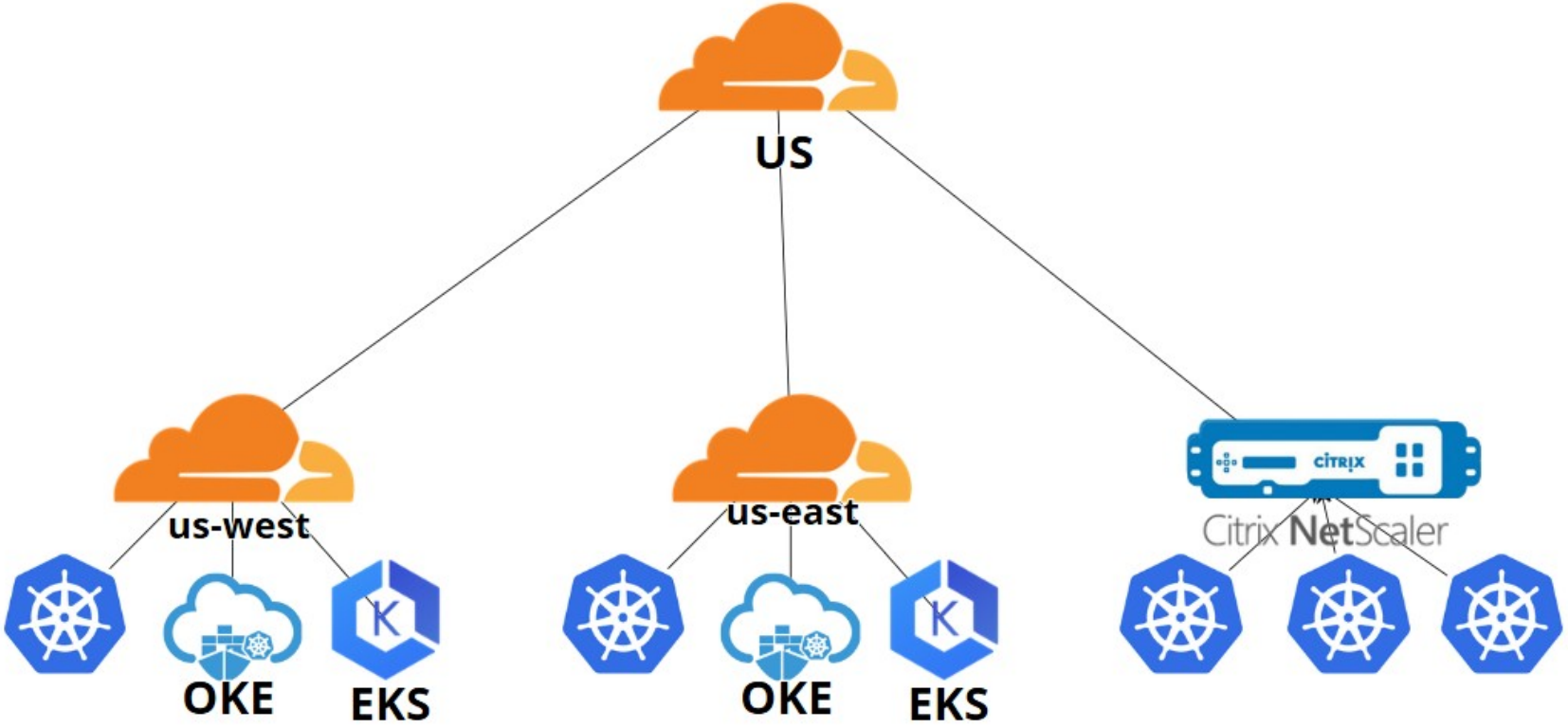
- Uniform deployment regardless of environments
- Service Discovery
- High availability and automatic recovery
- Container Packaging
- Native Kubernetes
- Public Charts
- Scaffolding Charts

8x8

8x8 Deployment Environments - Cloud



8x8 Deployment Environments - Hybrid

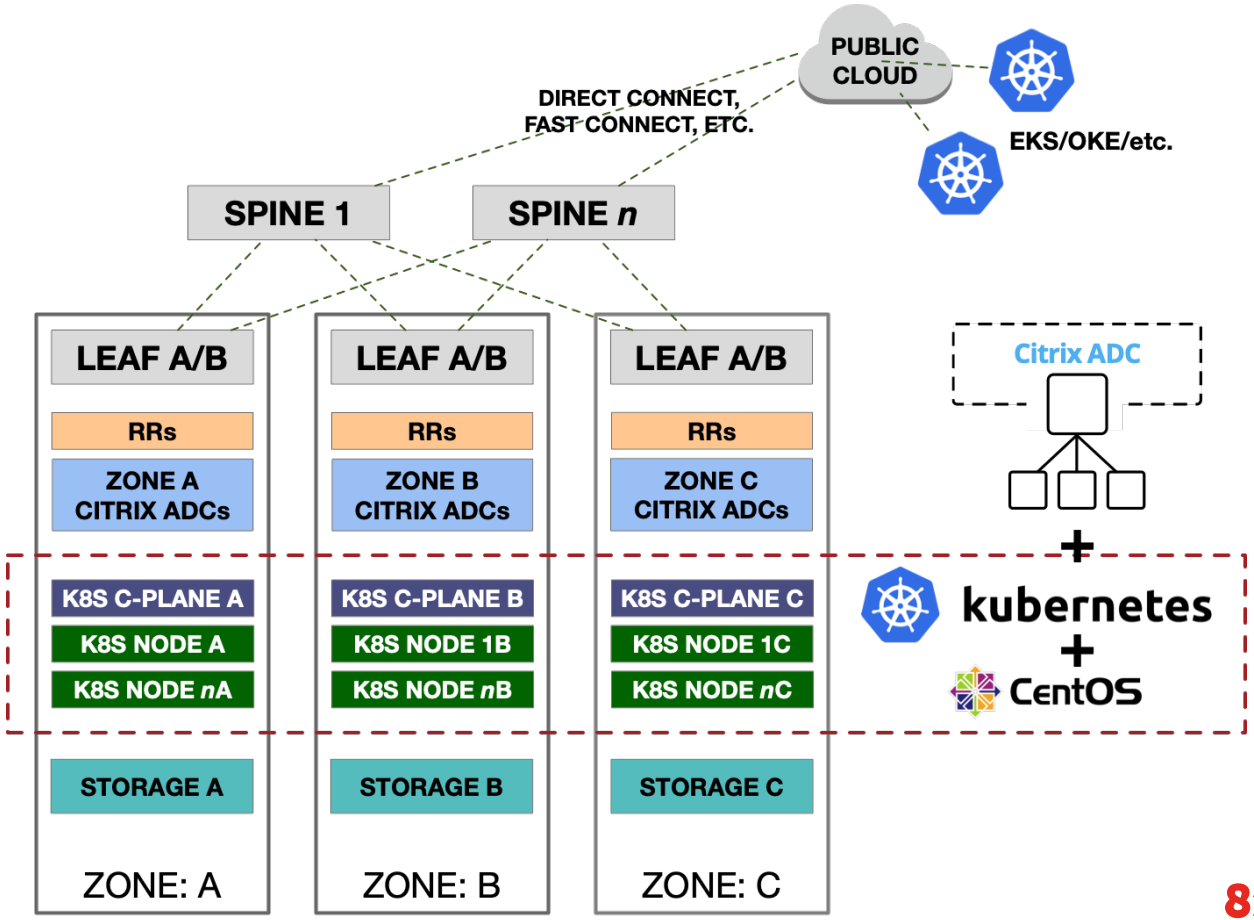


 |

8x8

8x8 Deployment Environment - On Premise

* Mainline K8s (1.17+)
* L3 Leaf-Spine
* 3 Zones/DC
* Zone Storage
  (& CSI Topology)
* BGP (Calico + Bird)
  - Citrix ADC
  - Route Reflectors
  - BGP to Host



8x8

8x8 Deployment Environment - Anatomy of a Zone

Requirements:

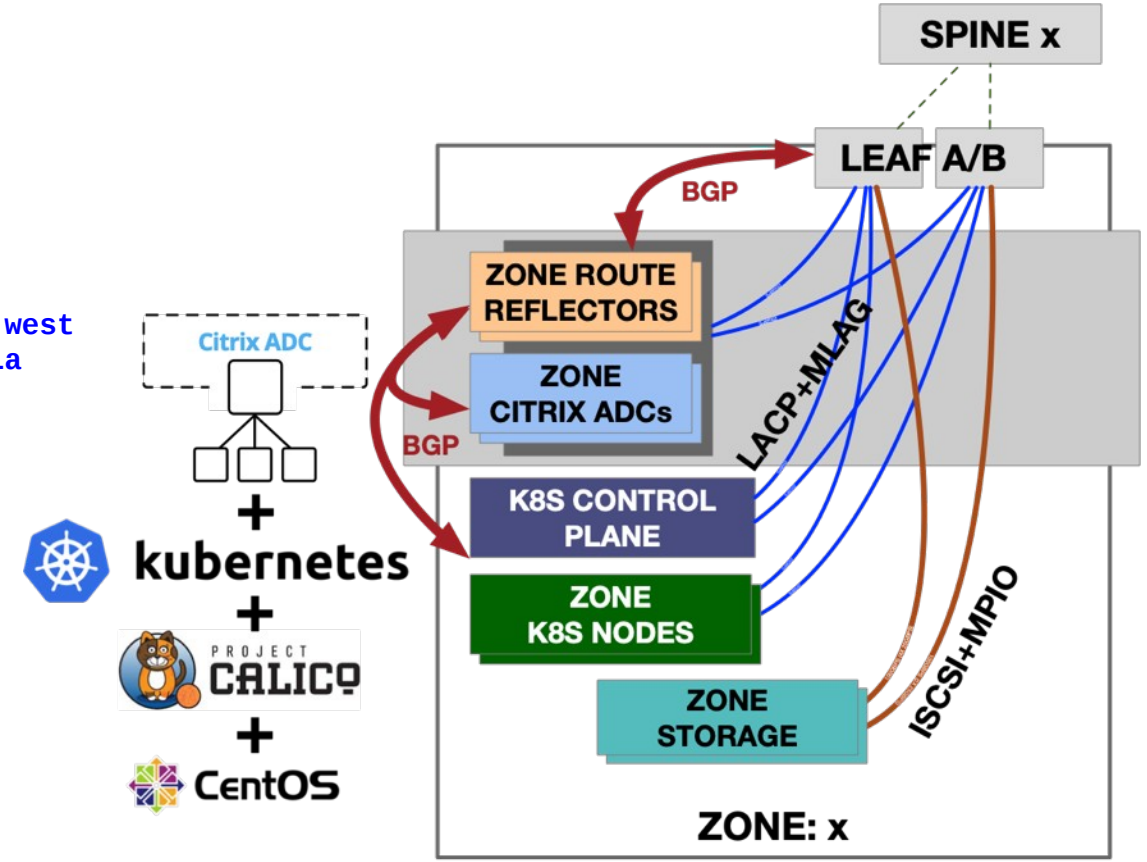
* Horizontal Scale

* Zone isolation
  `topology.kubernetes.io/region=us-west`
  `topology.kubernetes.io/zone=sjc01a`

* Specialized network
  Use-cases (DSR, ECMP, Anycast)

* Service IPs (VIPs) via BGP

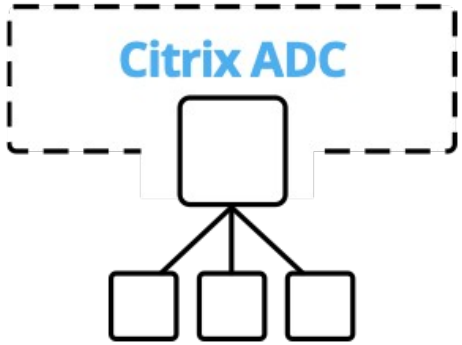
* K8s-native APIs/resources
  `kind: Service`
  `type: LoadBalancer`



8x8

8x8 Load Balancing Tier - Citrix ADC

Key Decision Factors:



- Supports Kubernetes Services and Ingresses (and Ingress Classes)
- Supports Complex, Zone-Aware Infrastructures
- Full-Feature BGP Support (e.g. Route Reflectors, Auth, RHI, ECMP)
- Horizontal/Linear Scale (using VPX in VMware)
- Rich Feature Support via CRD/Annotation (Cert Mgmt, Rules, Policy, IPAM, etc.)
- Simple, Hands-Off, Reliable Deployment and Configuration
- Flexible Licensing Options & Editions (MPX, VPX, CPX, etc.)

8x8 Recommendations

01	Choose your battles	• Only stateless microservices initially • New framework from the ground up
02	Alignment	• Strong collaboration between operations and development • Shared goals between teams
03	Self Service	• Both engineering and operations could react independently
04	Cloud First	• Started in the cloud for a uniform experience • 100% of the resources were automatable
05	Reserve System Resources	• System level, guarantee the OS a percentage • Namespace LimitResource defaults • Right size your workers

8x8 Recommendations

06	Embrace A Service Mesh	• Telemetry • Better Security • Connect desperate clusters
07	Security	• Limit kubectl usage • Helm safer • Cluster Role for a power user
08	Ingress vs Services	• Use an ingress strategy • Saves money • Provides Telemetry
09	Common Logging Format	• Simply it for the teams, send to standard out • Build into your framework or endorse a specific format
10	Centralize Metrics and Logs	• Prometheus has vast library of existing dashboards • Collect and ship using Daemonsets • Newrelic with Pagerduty