2024

Joe Mordica

Founder/CTO @ VOXO

voxo.co

Scaling/Managing RTC Workloads in K8s

Things we've learned operating a multi-region communications platform 100% in k8s

Agenda

GitHub Repo

github.com/voxoco/k8s

Visual Diagram

voxo.co/technology

3+

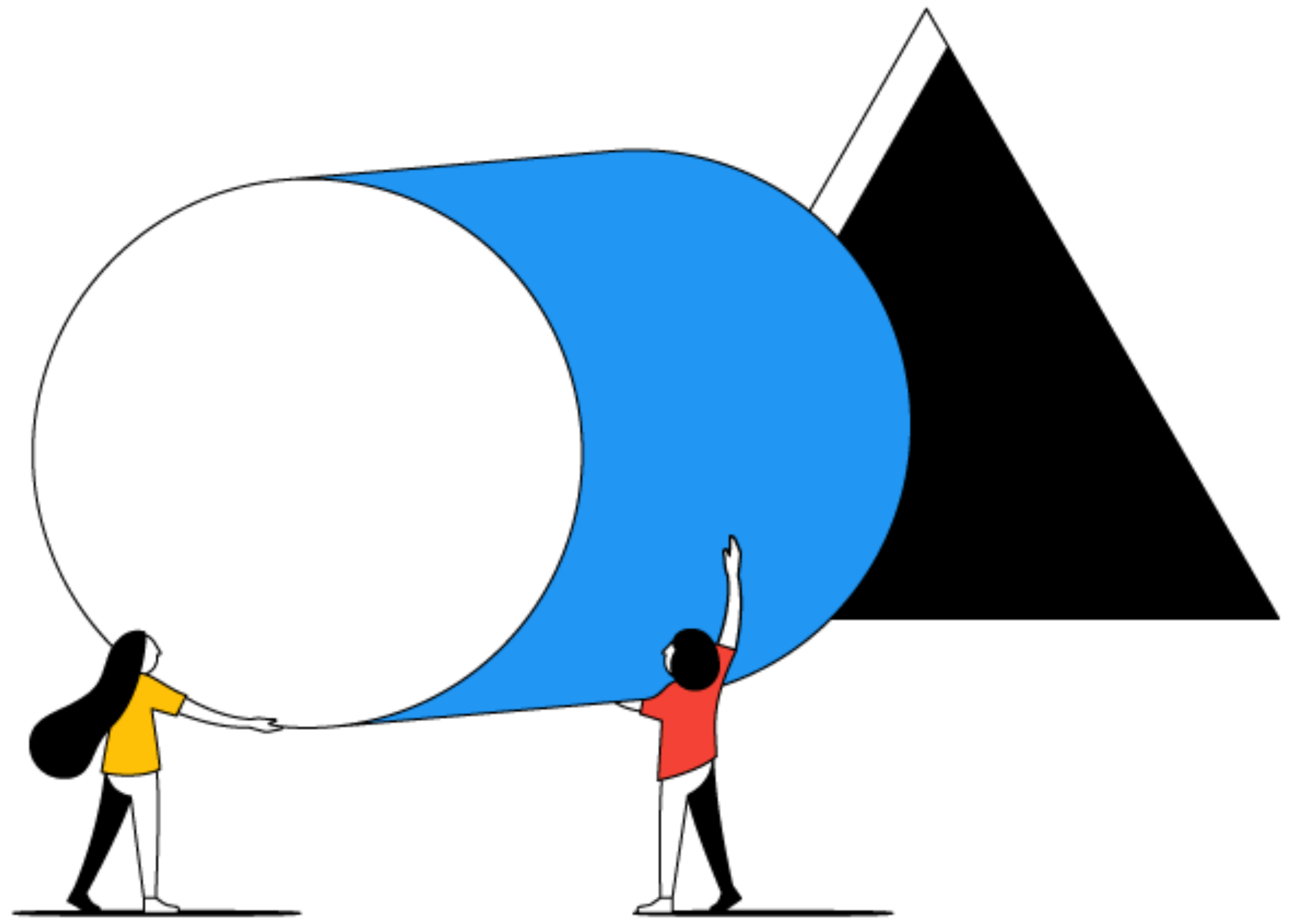
Regions

0

Single point of failure

0

Maintenance windows



DOES IT FIT?

Managing stateful workloads globally in the very ephemeral world of Kubernetes.

Problems

1. Plan for Failure (mindset shift)
2. Dropped calls on shutdown
3. Scaling
4. Maintenance downtime

Failure (Mindset shift)

Kubernetes

Let Kubernetes do what it does best. Scale your apps and move stuff around based on CPU/Node health

Ephemeral

- Make apps as ephemeral as possible. Expect them to fail and/ or move between nodes anytime
- No PVC's.
- Orchestrator Example
- Use sidecar containers

Global KV

We use Consul KV by HashiCorp as a global service discovery. Consul coupled with consul-template is awesome

2

Dropped calls on shutdown

- ✓ Graceful shutdowns
- ✓ Use sidecar containers
- ✓ Update KV store
- ✓ terminationGracePeriodSeconds
- ✓ Ship logs

3

Scaling

- ✓ Let the HPA take care of it
- ✓ Tweak HPA scale up/scale down policy
- ✓ Set proper CPU requests on pods
- ✓ Application lifecycle needs to be dialed in
- ✓ Make small iterations and test

4

Maintenance downtime

- ✓ 0 downtime maintenance
- ✓ Create the ability to delete anything anytime
- ✓ Use rolling restarts for resources
- ✓ Use Google Global Load Balancer
- ✓ QA well

Questions

joe@voxo.co

OUR WEBSITE

voxo.co