Orchestrating containers with Kubernetes

What’s Kubernetes?

- Kubernetes is a Linux container manager

- “manual” container management is hard

- offload scheduling, health & process checking, resource management, volume management (,…) to Kubernetes

Containers?

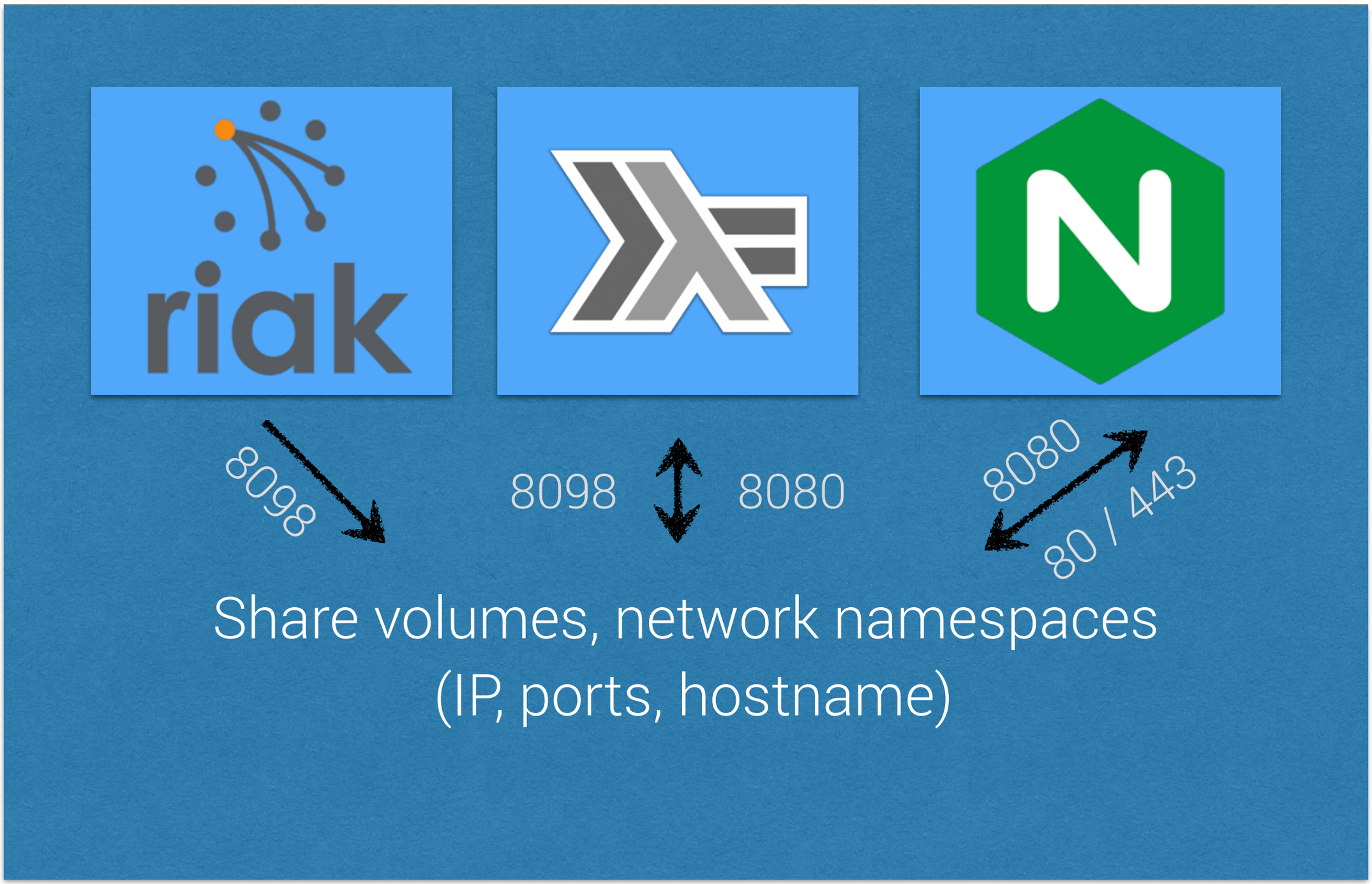
- Containers are not VMs
- “chroot on steroids”
- Docker improves concept of containers with image handling and other tooling
- Kubernetes is not restricted to Docker engine

Kubernetes overview

- In simple terms: A set of containers is replicated N times
  - Organised into Pods, ReplicationControllers and Services
- Kubernetes abstracts networking, scheduling and the hosts themselves away
- Built-In support for rolling updates, scaling, DNS in cluster
- Designed to reflect Google's internal infrastructure

Pods

Pods



Share volumes, network namespaces
(IP, ports, hostname)

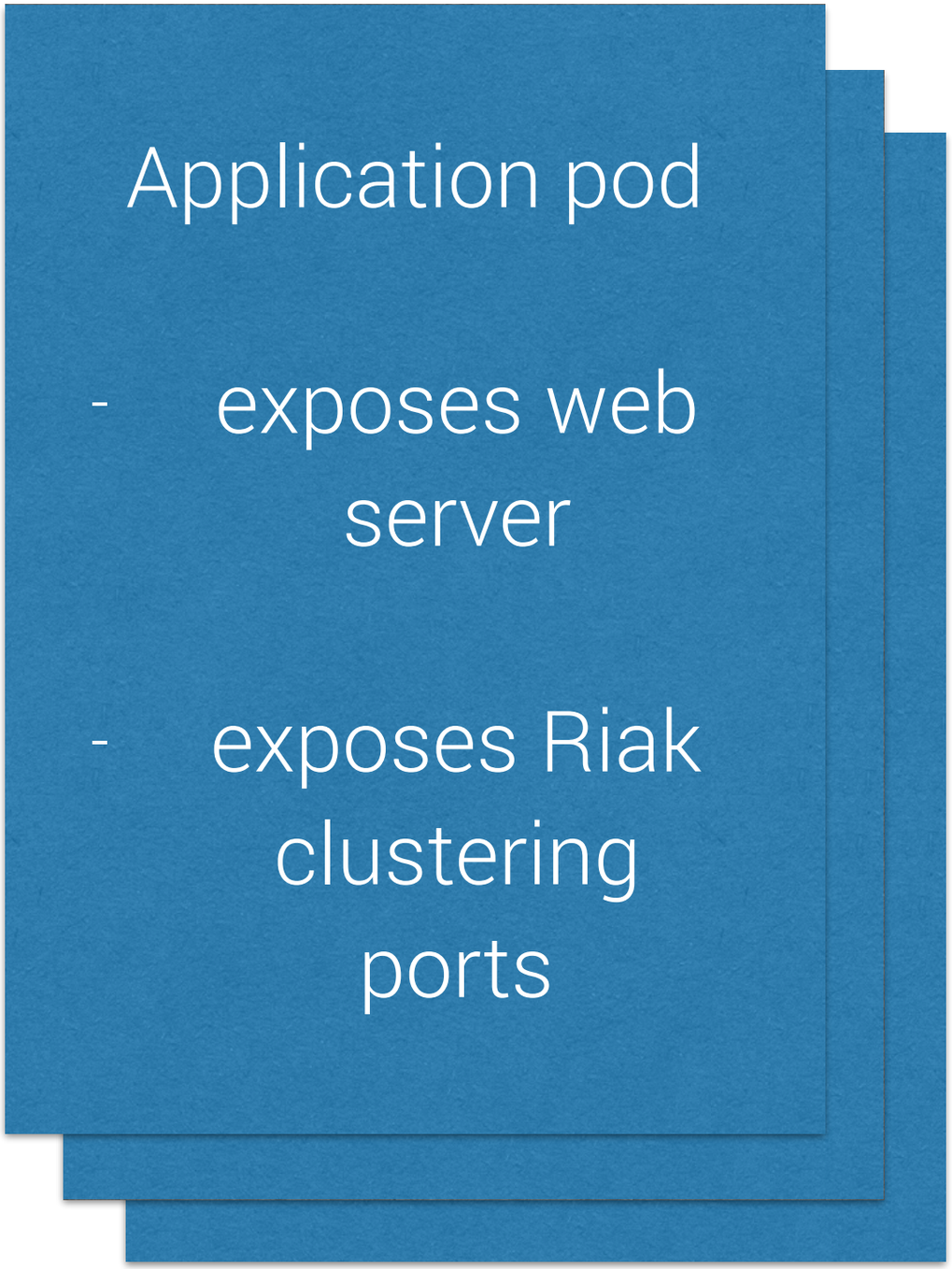
ReplicationControllers

Application pod

- exposes web server
- exposes Riak clustering ports

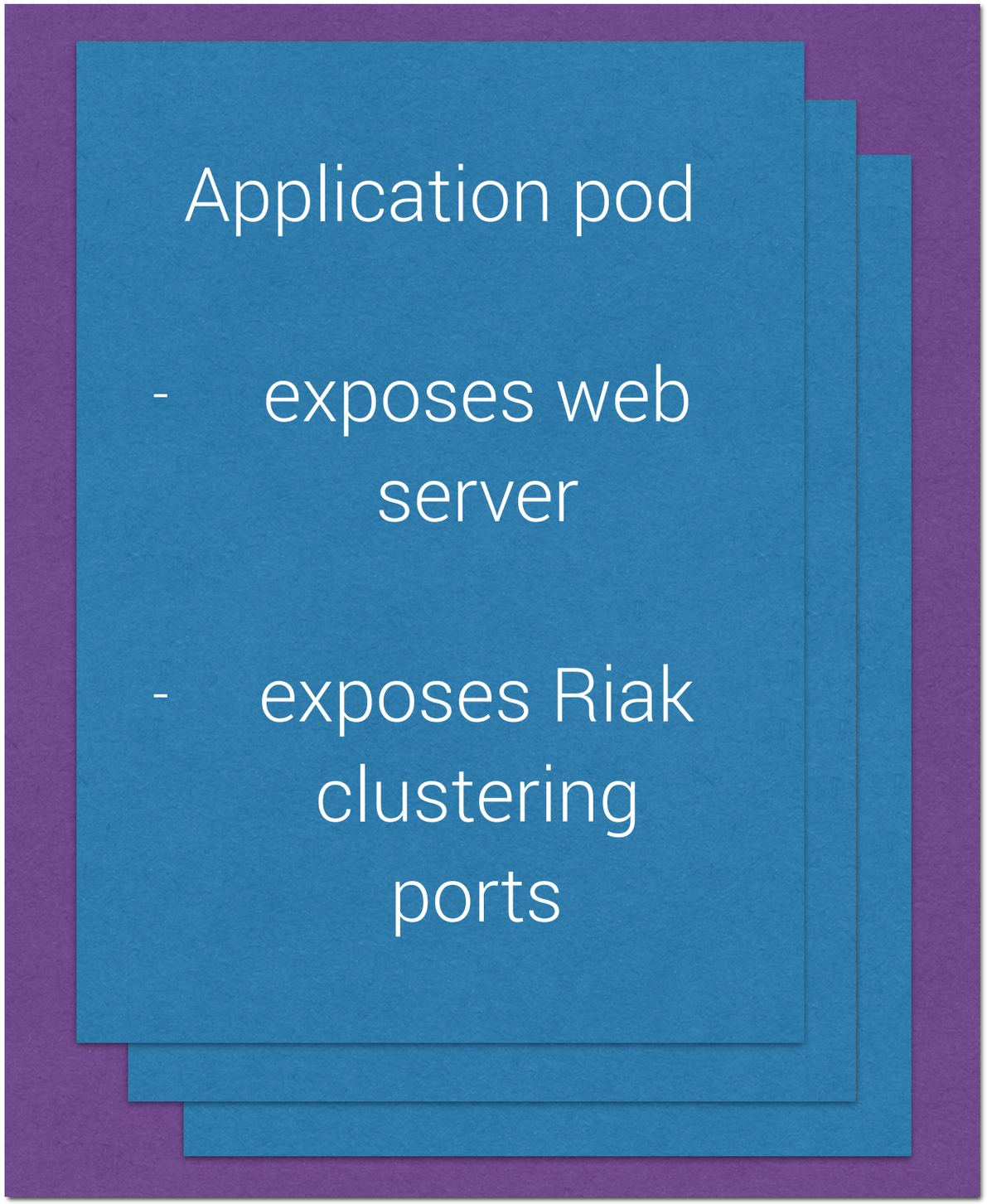
replicas=1

ReplicationControllers



replicas=3

ReplicationControllers



replicas=3

k8s-app=my-app

Services

ReplicationController

metadata:
  k8s-app: my-app
replicas: 3

Pods expose ports:
8098, 8080, 80, 443

Service

selector:
  k8s-app: my-app
ports:
  - name: https
    port: 443
  - name: http
    port: 80
type: LoadBalancer

Scheduling

- Currently relatively simple scheduling:
  - Pods can be configured to request certain resources
  - Configured resources are hard limits
  - Pods will be scheduled on to nodes with available resources and least usage

Demo time

Questions?

- Slides and resources available on Github:

- github.com/tazjin/k8s-elk-demo