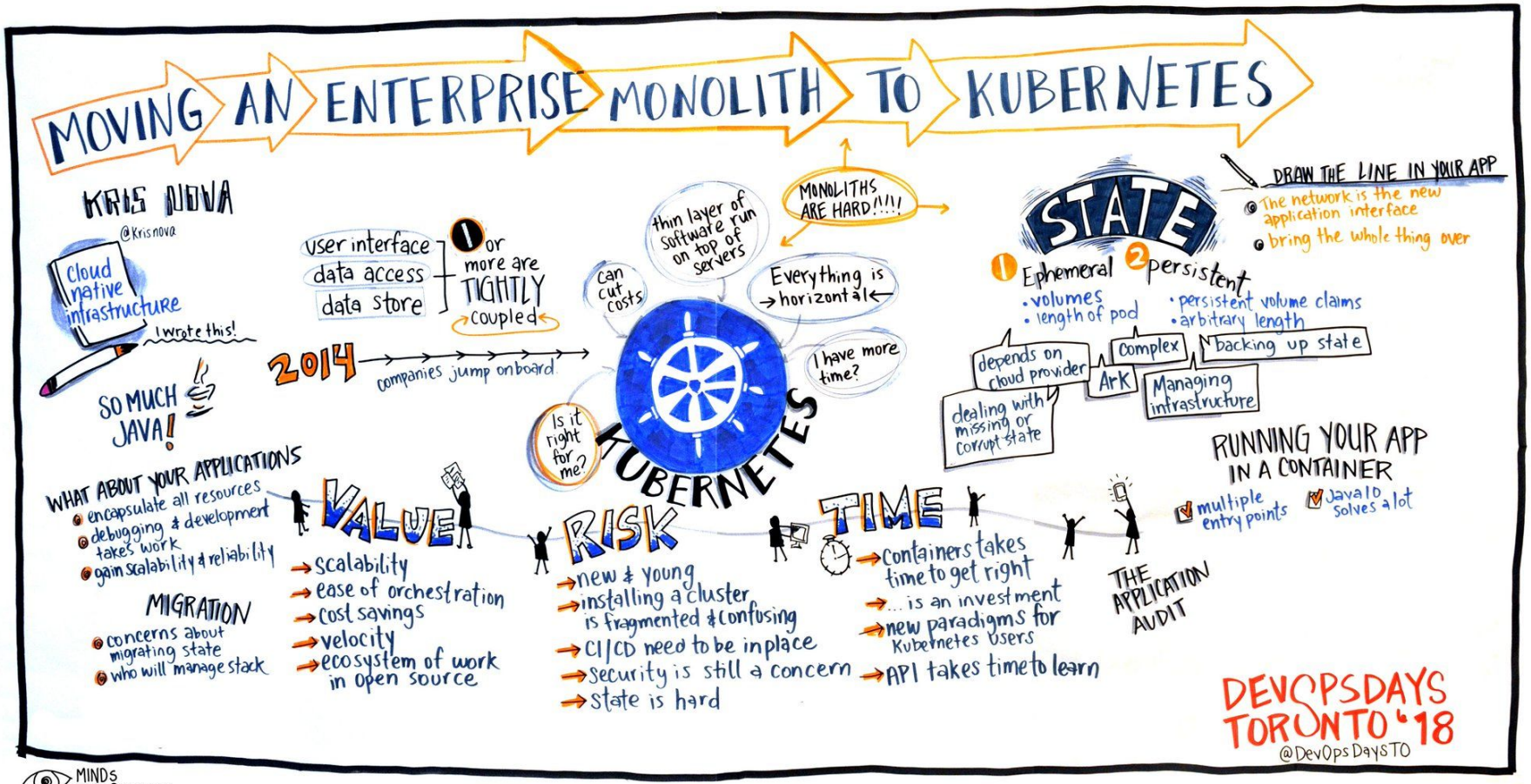
Moving Monolithic Apps to Kubernetes

Kris Nova

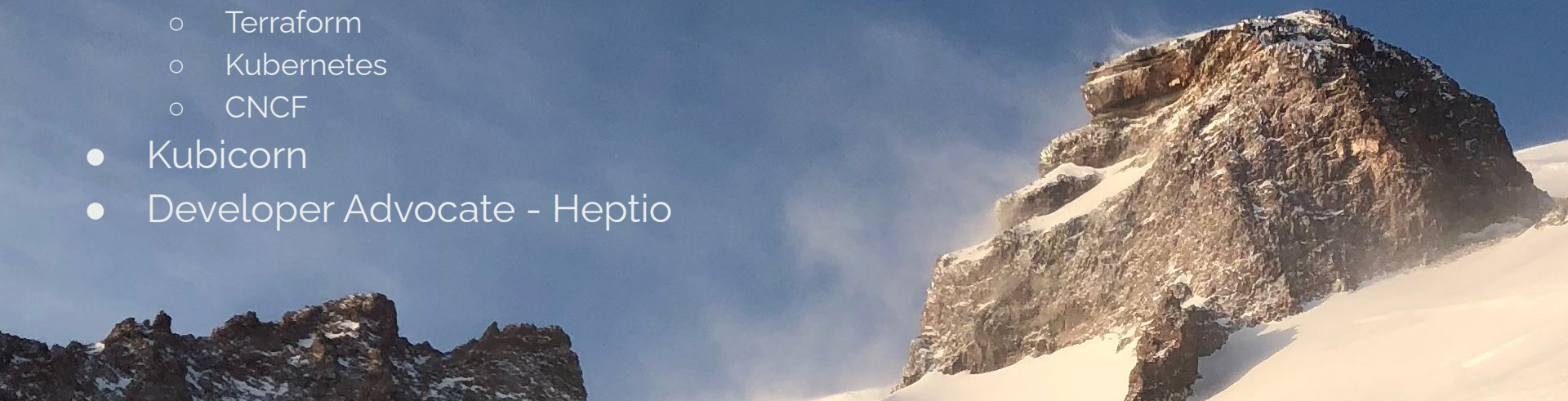
heptio

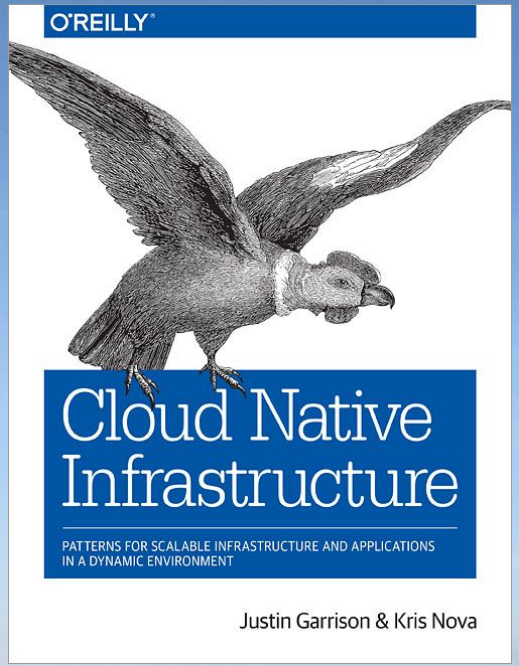

MOVING AN ENTERPRISE MONOLITH TO KUBERNETES
KRIS NOVA
@Krisnova
Cloud native infrastructure
I wrote this!
SO MUCH JAVA!
user interface
data access
data store
1 or more are TIGHTLY coupled
2014
companies jump onboard
thin layer of software run on top of servers
MONOLITHS ARE HARD!!!!
can cut costs
Everything is → horizontal ←
I have more time?
Is it right for me?
KUBERNETES
STATE
1 Ephemeral
• volumes
• length of pod
2 persistent
• persistent volume claims
• arbitrary length
depends on cloud provider
dealing with missing or corrupt state
Ark
Complex
backing up state
Managing infrastructure
DRAW THE LINE IN YOUR APP
The network is the new application interface
bring the whole thing over
WHAT ABOUT YOUR APPLICATIONS
encapsulate all resources
debugging & development takes work
gain scalability & reliability
MIGRATION
concerns about migrating state
who will manage stack
VALUE
→scalability
→ease of orchestration
→cost savings
→velocity
→ecosystem of work in open source
RISK
→new & young
→installing a cluster is fragmented & confusing
→CI/CD need to be in place
→security is still a concern
→state is hard
TIME
→containers takes time to get right
→... is an investment
→new paradigms for Kubernetes users
→API takes time to learn
THE APPLICATION AUDIT
RUNNING YOUR APP IN A CONTAINER
multiple entry points
Java 10 solves a lot
DEVOPSDAYS TORONTO '18
@DevOpsDaysTO
MINDS EYE CREATIVE.CA


Who am I?

Kris Nova

- Kubernetes Contributor and Maintainer
  - Kops
  - Kubeadm
  - Cluster API
- Author: Cloud Native Infrastructure
  - Go
  - Terraform
  - Kubernetes
  - CNCF
- Kubicorn
- Developer Advocate - Heptio




O'REILLY
Cloud Native Infrastructure
PATTERNS FOR SCALABLE INFRASTRUCTURE AND APPLICATIONS IN A DYNAMIC ENVIRONMENT
Justin Garrison & Kris Nova


So why monolithic applications?

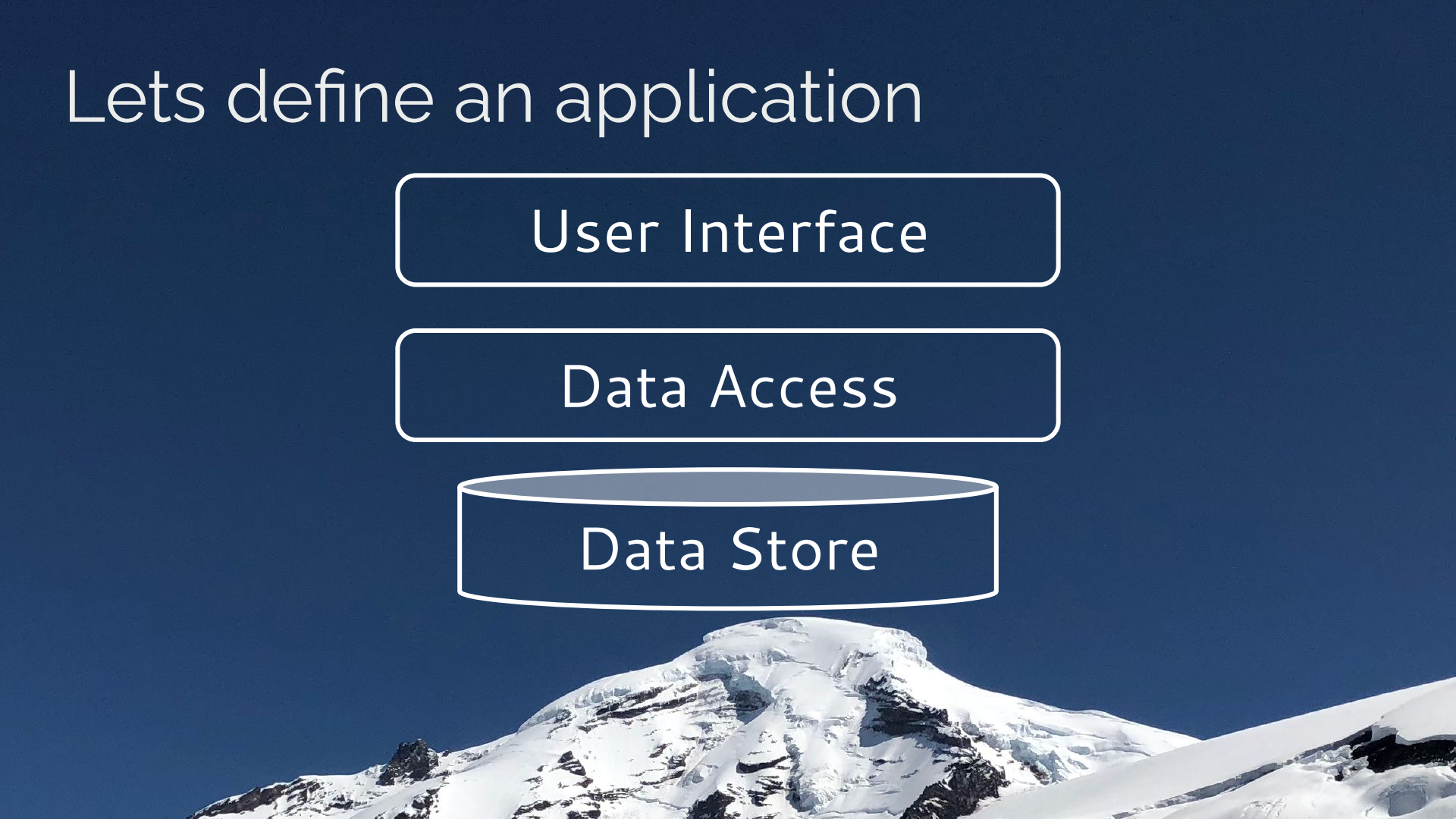

They’re hard.

Experience at Heptio

- Looking at real life situations with large stateful applications
- Discovered there is way more Java than we thought
- Discovered there wasn't really a good story for these large applications
- Started working on figuring out a migration story

What is a monolithic application?

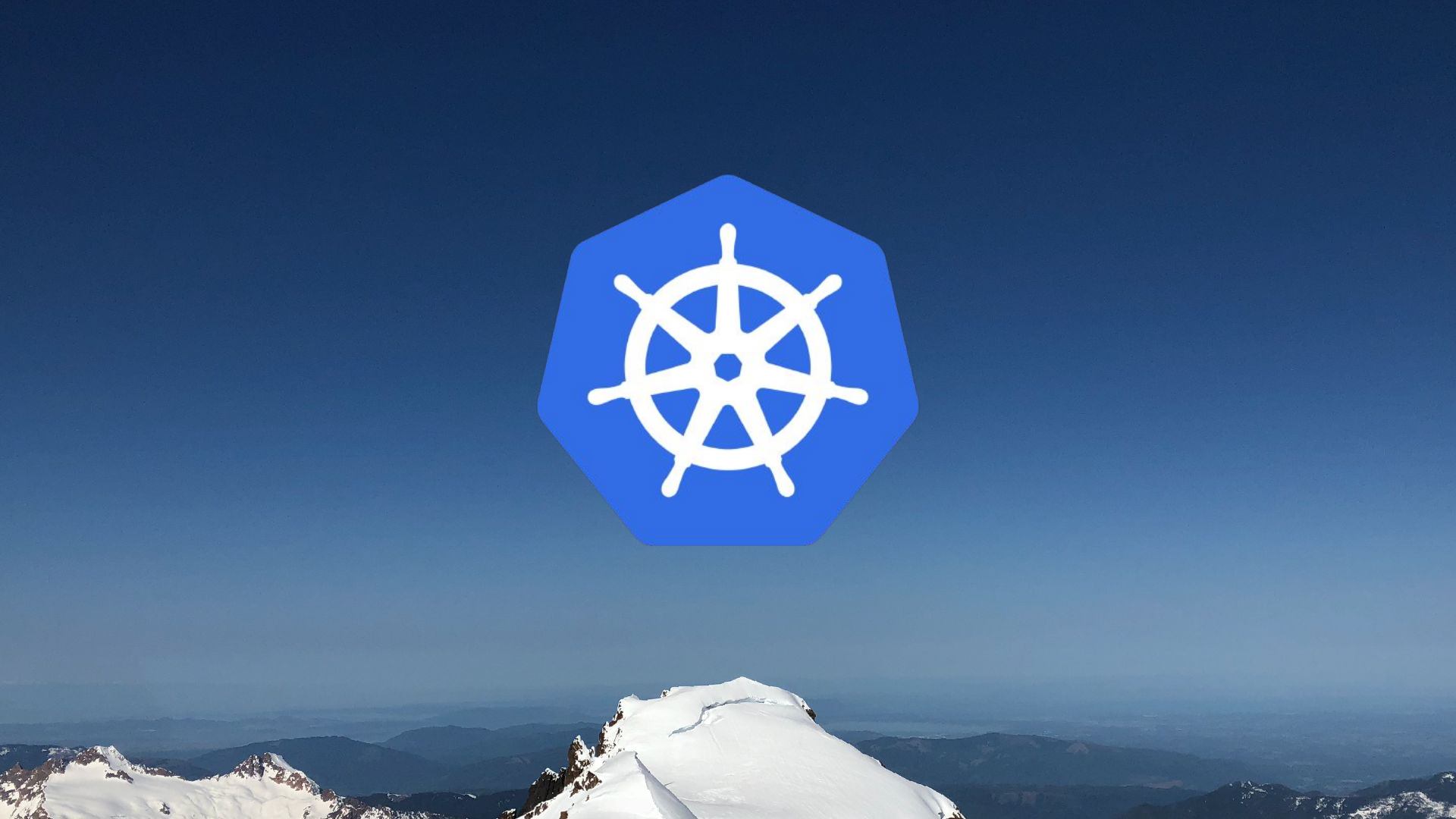
Lets define an application



Monoliths have one or more tightly coupled


hadoop
Drupal
There are a lot of monolithic applications!
elasticsearch
MySQL®
RabbitMQ

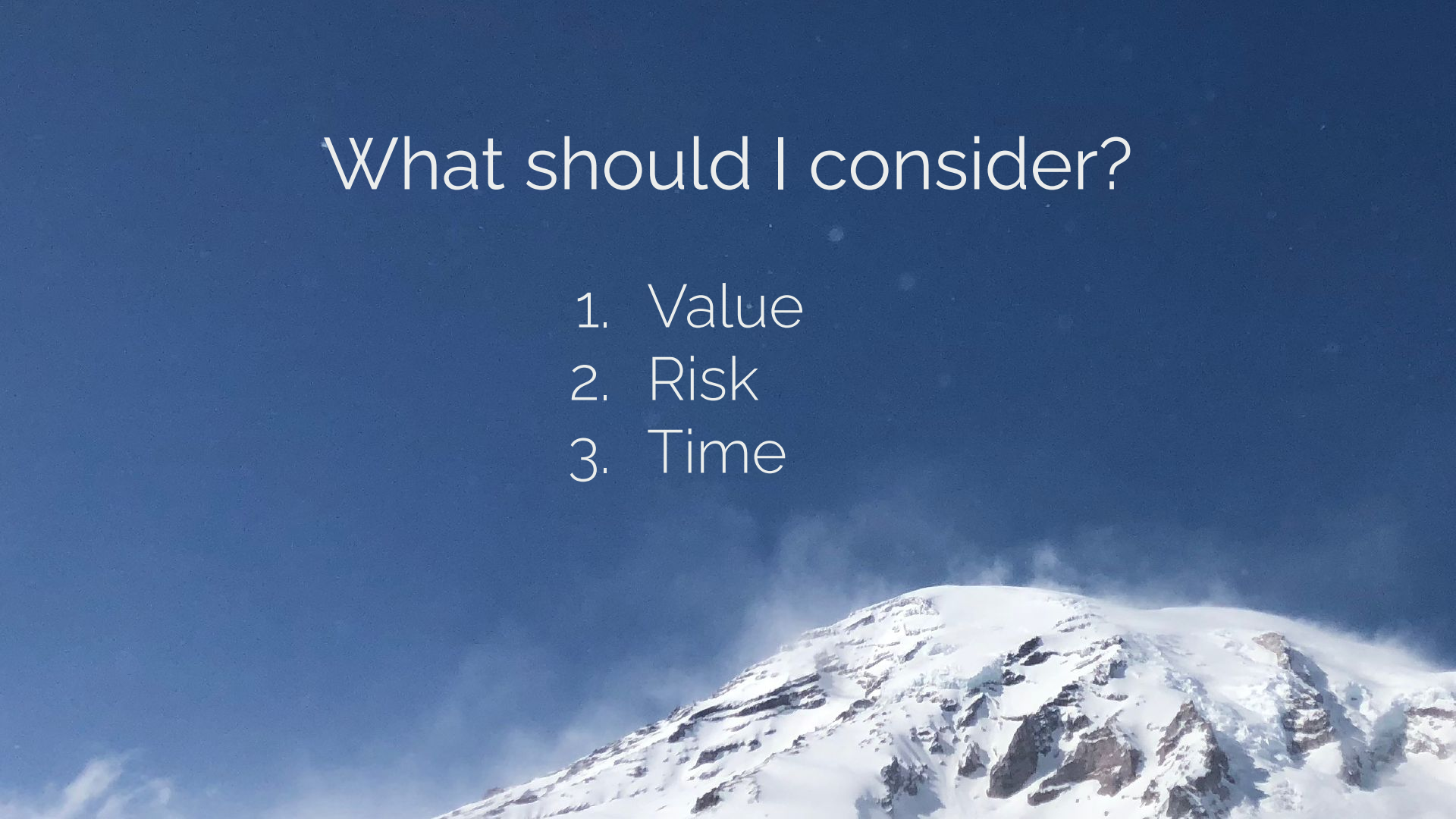
So what about Kubernetes?



Is Kubernetes right for me?

What should I consider?

1. Value
2. Risk
3. Time



What do we gain in VALUE?

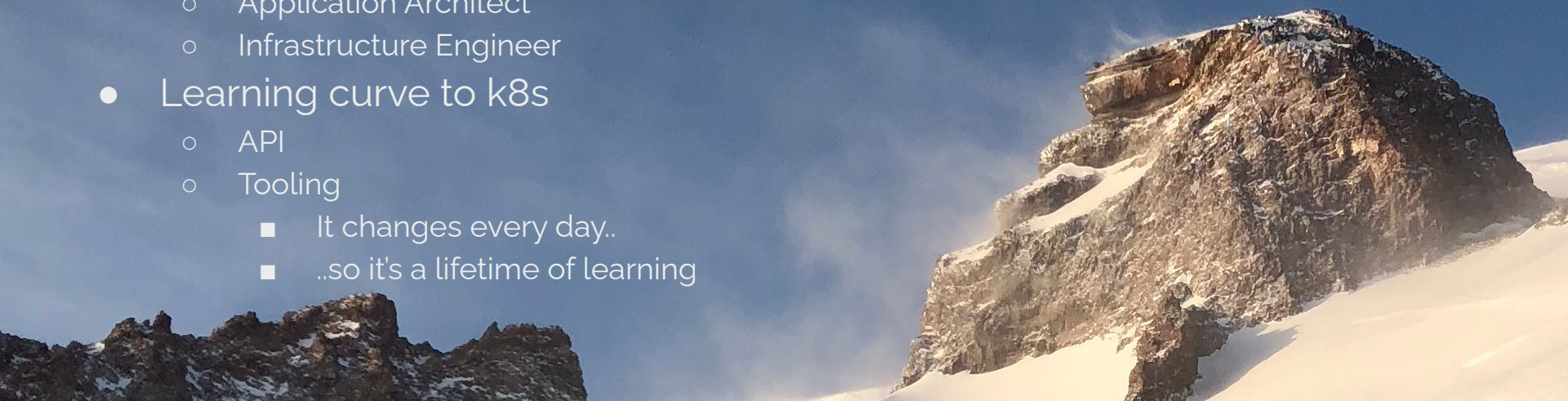
- Scalability
  - Extensibility, Observability, Velocity
- Ease of orchestration
  - More time for customers and engineering
- Ecosystem of work in open source
  - Storage, CNI, Logging, Alerting, Monitoring
- Cost savings
  - Case studies of 40-50% cost in hardware savings
- API of the cloud

What are the RISKS?

1. Kubernetes is NEW and YOUNG
    a. New: most people are less than a year or two in production (learning curve)
    b. Young: the project was open sourced in 2014
2. Installing a cluster is still fragmented and confusing
3. Still have most of the same concerns as you would without Kubernetes
4. Most large applications are not containerized
5. CI/CD systems need to be built out and understood
6. Security is still a concern
7. State is hard

What about the TIME?

- Containers take time to get right
- Kubernetes is an investment, it takes time and effort to adopt
  - It promises stability, scalability, and ease in the future
- There are new paradigms for Kubernetes users
  - Cluster Engineer/Operator
  - Application Engineer
  - Application Architect
  - Infrastructure Engineer
- Learning curve to k8s
  - API
  - Tooling
    - It changes every day..
    - ..so it's a lifetime of learning



Technical Concerns?


Let's talk about
state

Types of State in Kubernetes

- Ephemeral State
  - Volumes
  - Length of pod
- Persistent State
  - Persistent Volume Claims
  - Arbitrary length

What you need to know about state

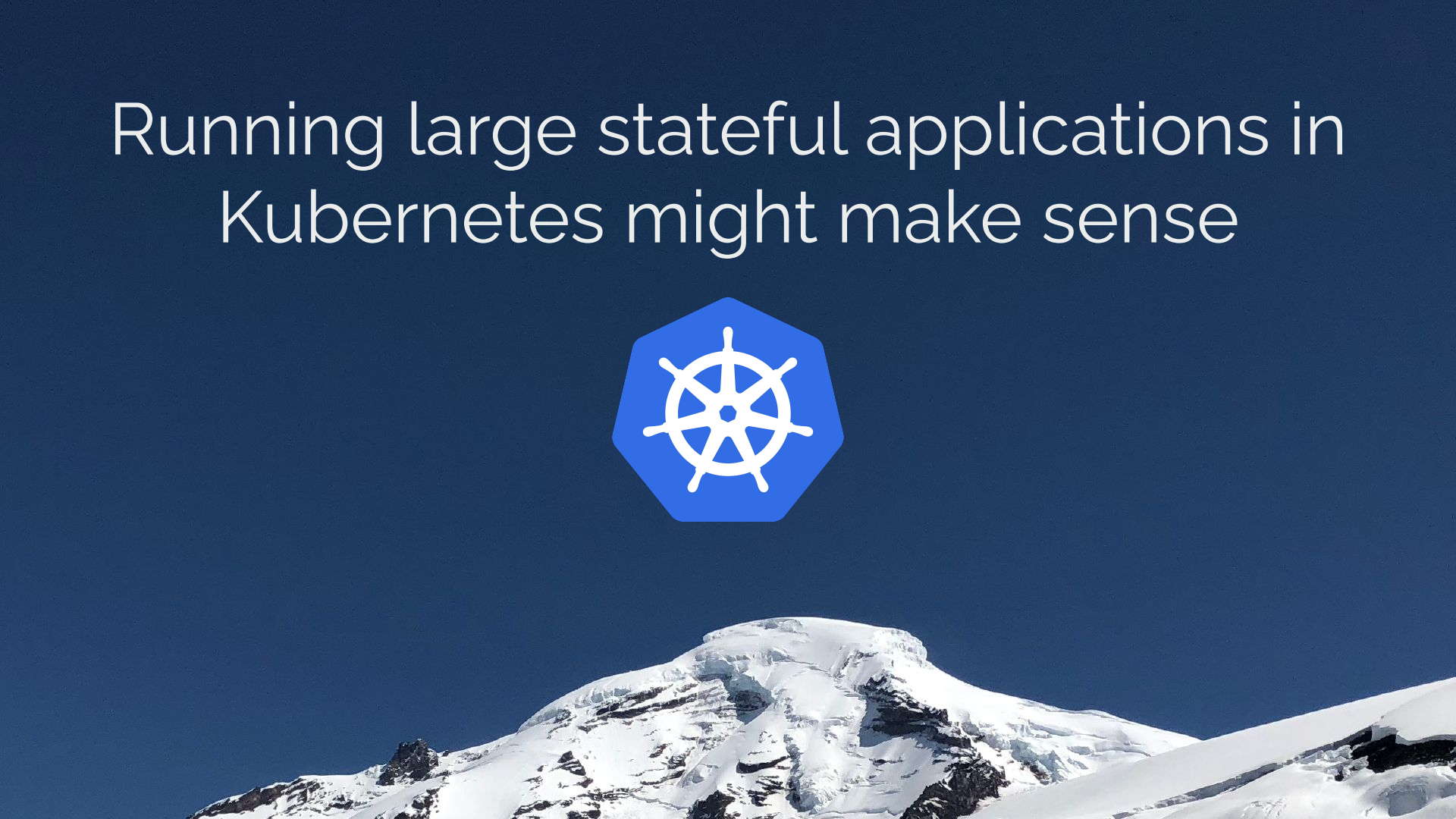
- Depends on cloud provider
- Complex - room for errors
- Backing up state
  - Ark
- Dealing with missing or corrupt state
- Managing infrastructure

Risk

Time

Value

Running large stateful applications in Kubernetes might make sense



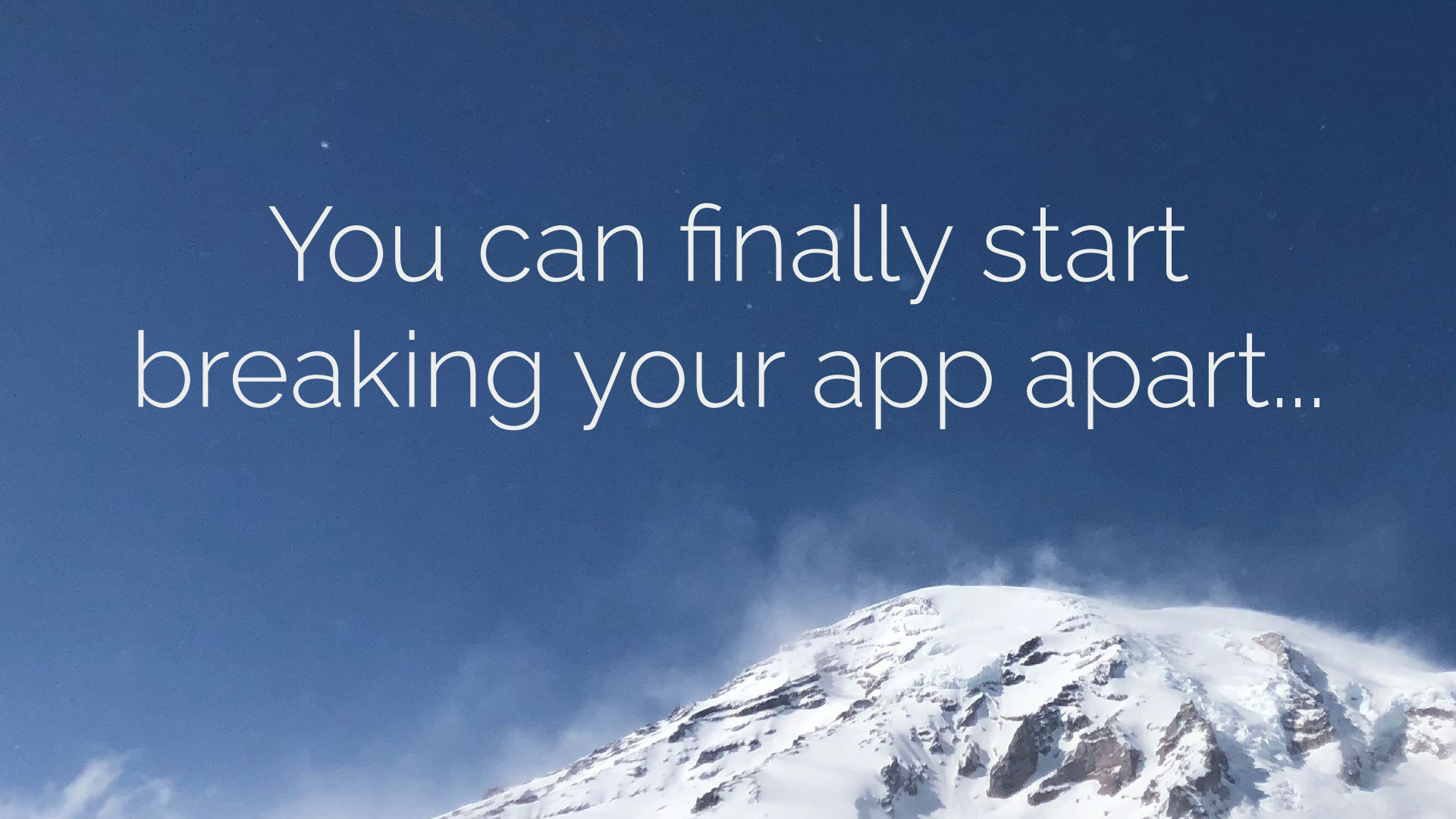
Running your app in a container

- There are a lot of developers tools to help with this
- Java 10 solves most* Java concerns with containers!
- Gain security, repeatability, and packaging
- CI/CD (something) needs to be put in place
- Multiple entry points
- You can either have one container to rule them all or...
    - See next slide

Risk

Time

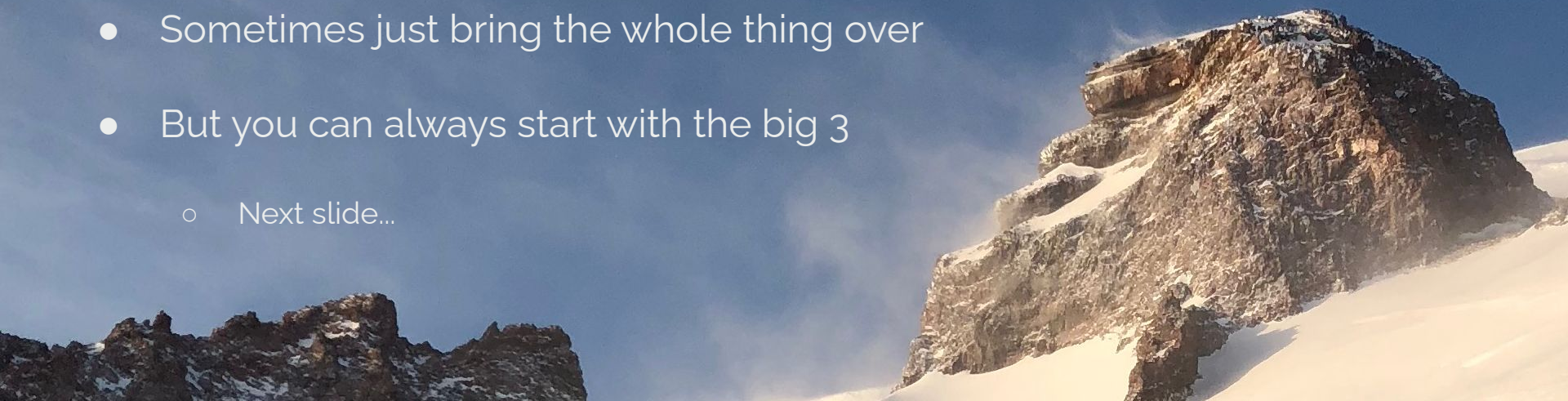
Value


You can finally start
breaking your app apart...

Microservices! Agh!

Where do you draw the line in your app?

- The network is the new application interface
  - gRPC, HTTP, Istio, Service Meshes
- Any time you start to transfer large, complete data structures in your app
- Sometimes just bring the whole thing over
- But you can always start with the big 3
  - Next slide...



Monoliths have one or more are tightly coupled

Awareness that containerizing your app might take time, but has benefits

What about your applications?

- Encapsulate all resources for your app
  - Static manifests, ksonnet, helm, git
- Debugging and developing your applications take work
  - New logging paradigms, new development stories
- Gain scalability, and reliability
  - Scheduler is rad

Risk

Time

Value

Running applications takes time, but offers a lot of gained value.

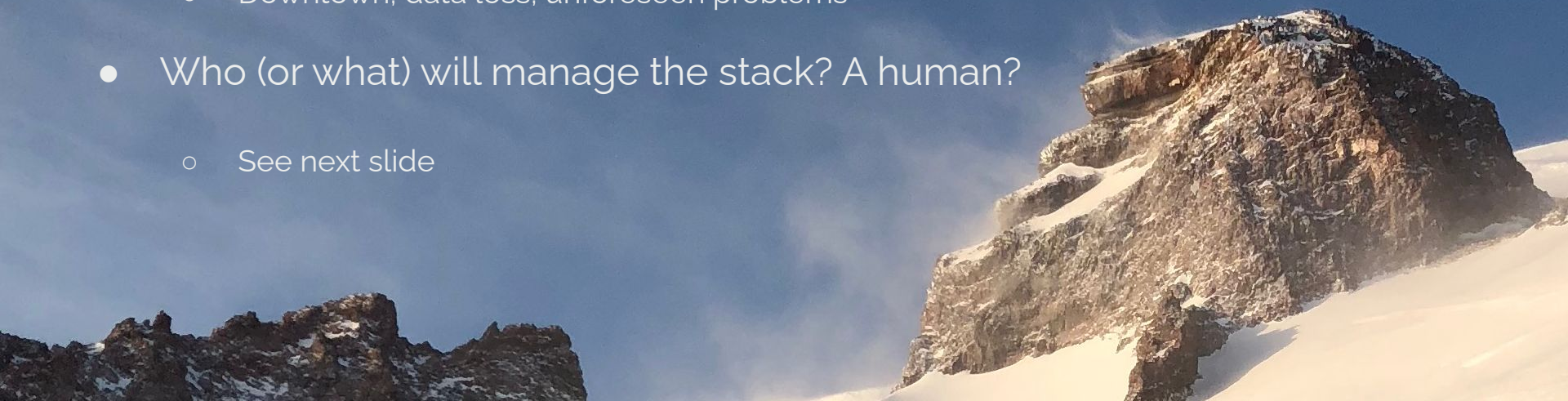
What about the migration?

- Concerns about migrating state
  - Or having a fragmented system
- All the major concerns of any migration
  - Downtown, data loss, unforeseen problems
- Who (or what) will manage the stack? A human?
  - See next slide

Risk

Time

Value



What about the migration?

The migration is similar to any other migration, and risky.

Why are monoliths harder

- Probably a code change
  - Entrypoint matters
- How do you manage config
- Applications not designed to be ran in a container
- Engineering effort to change already brittle application
- Big

Risk

Time

Value

The application audit

- Huge lesson on even knowing concretely what you have
- Where is the list of dependencies your application needs?
- Where do your configs live?
- Does your application care what OS it's running ?

Monolithic applications are significantly harder

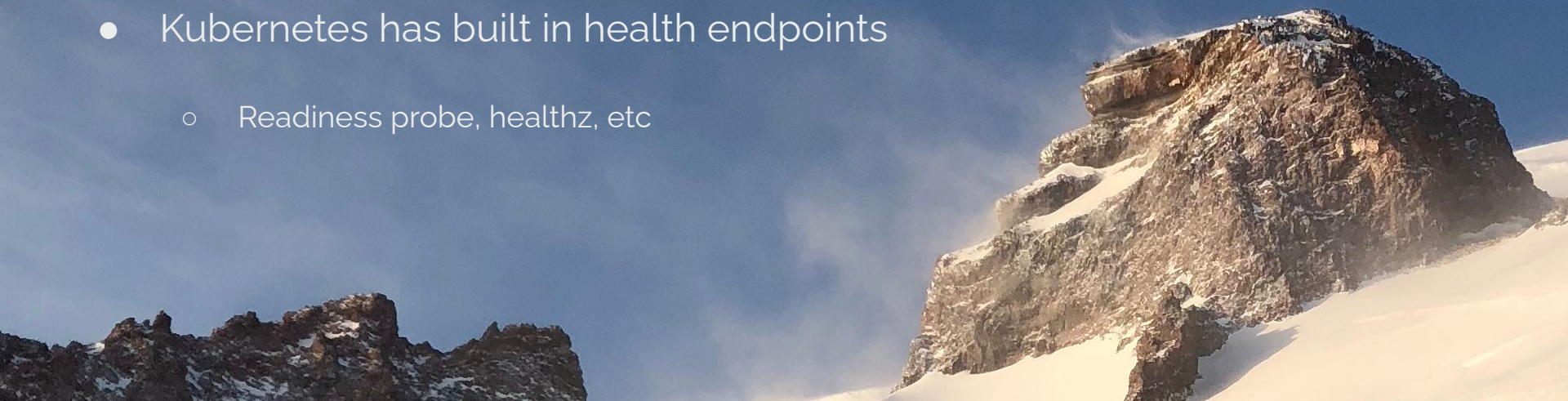
What about logging, monitoring, alerting?

- Plethora of open source solutions
  - Prometheus, Heapster, Grafana, etc
- Kubernetes has built in health endpoints
  - Readiness probe, healthz, etc

Risk

Time

Value



The Kubernetes ecosystem can help cut costs

Where did we learn this?

We created a prototype application

- Written in Java
- Hard to run and manage
- Designed for cloud foundry
- Never containerized
- github.com/heptio/monolith

So in conclusion

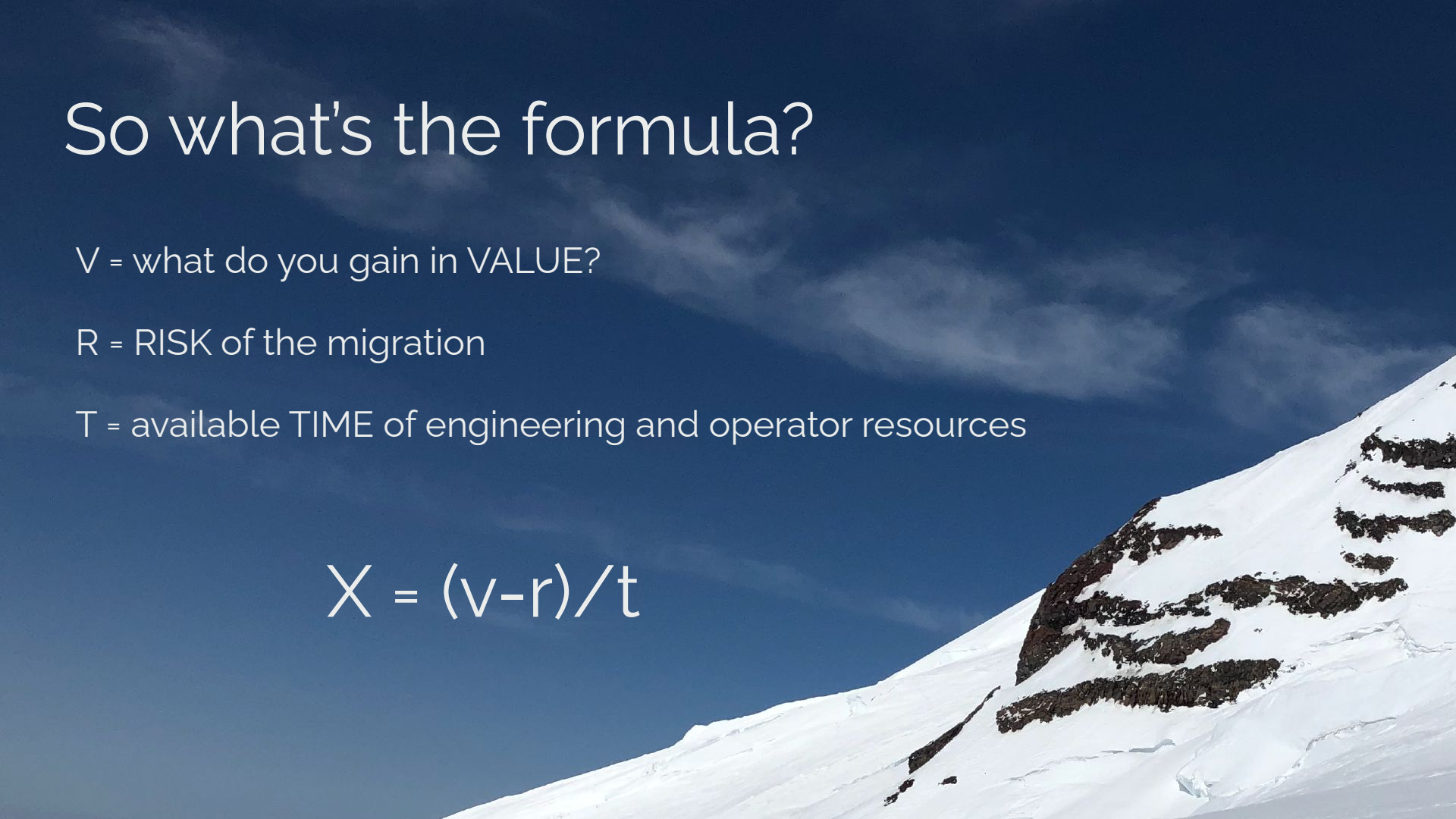
So what's the formula?

V = what do you gain in VALUE?

R = RISK of the migration

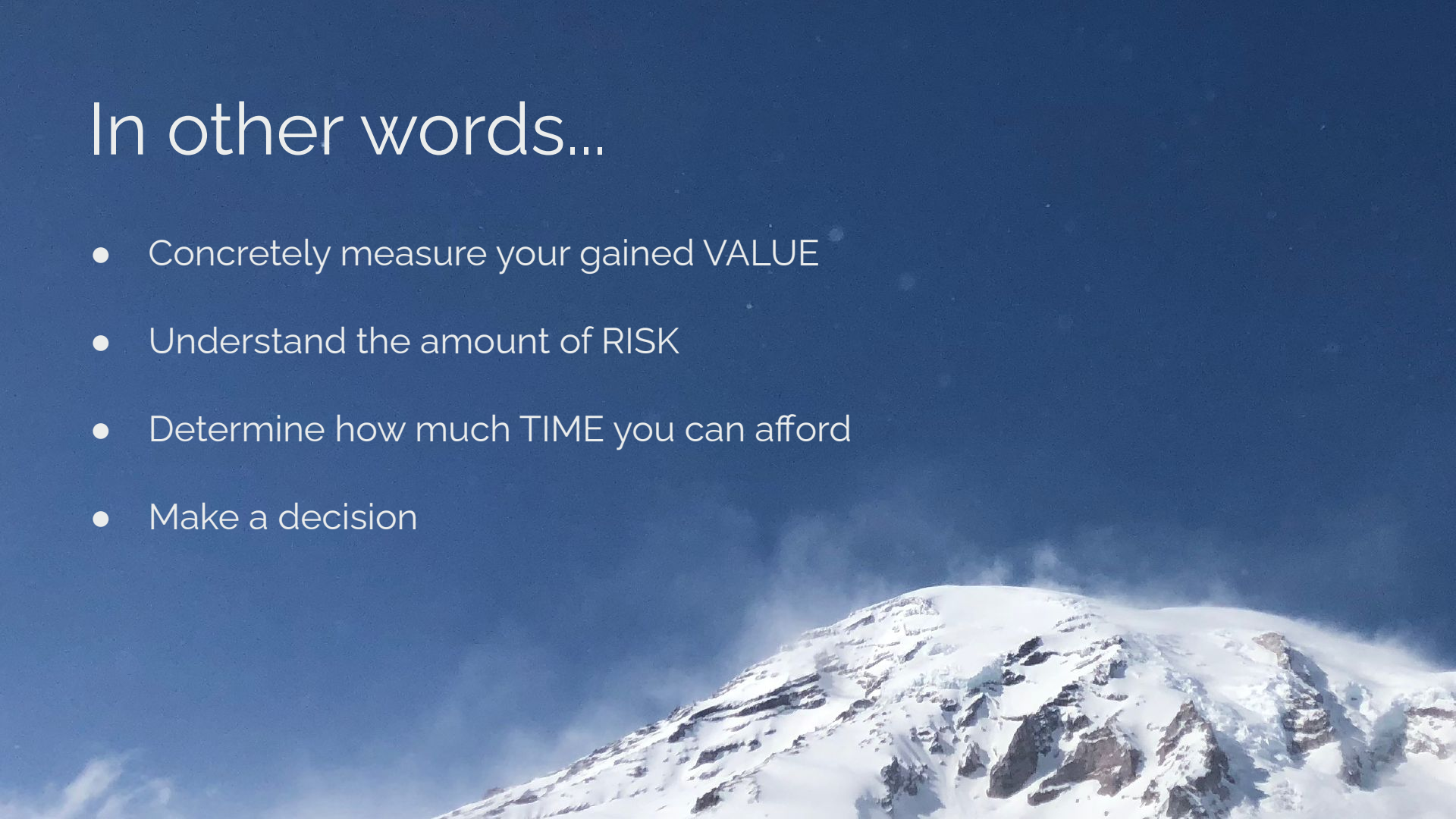
T = available TIME of engineering and operator resources

$$X = (v-r)/t$$



In other words...

- Concretely measure your gained VALUE
- Understand the amount of RISK
- Determine how much TIME you can afford
- Make a decision



Kris Nova

@krisnova