

Predictive Auto-scaling in the Kubernetes Cluster Manager

■ THANK YOU

- Professor Albrecht, Professor, Williams College
- Brendan Burns, Lead Engineer for Kubernetes, Google

1.

Goals

Why do we care?



General Goal

Contribute to distributed system's ability to reliably and resourcefully do large, varying amounts of computation.

■ A typical use case

Seek to have hbogo.com reliably and resourcefully served during the season premier of Silicon Valley...



But **who** will serve it and
how?

Accomplishing General Goals

How do cluster managers reliably and resourcefully perform large, varying amounts of work?

■ What is a cluster manager?

- A **cluster** is a collection of commodity computers linked by a local-area network.
- A **cluster manager** is like an operating system for a cluster.



Benefits of **Cluster Managers**



Cluster managers allow us to perform computational work that could never be performed on a single computer.



Kubernetes





Specific Goal

To maximize the sum of two Kubernetes's metrics: Efficient Resource Utilization and Quality of Service

■ Unpacking this Goal

Kubernetes

An open-source cluster manager from Google.

Efficient Resource Utilization (ERU)

Is the application efficiently using the resources it is given?

Quality of Service (QOS)

Is the application accomplishing its stated purpose?

The goal is **balance**.

Increasing ERU/QOS while decreasing the other is easy, we seek to increase the summation.

Kubernetes specific terms

Pods

A stateless, replicable wrapper around related containerized applications (ex. a pod for hbogo.com contains an Apache web server and a cache of video content)

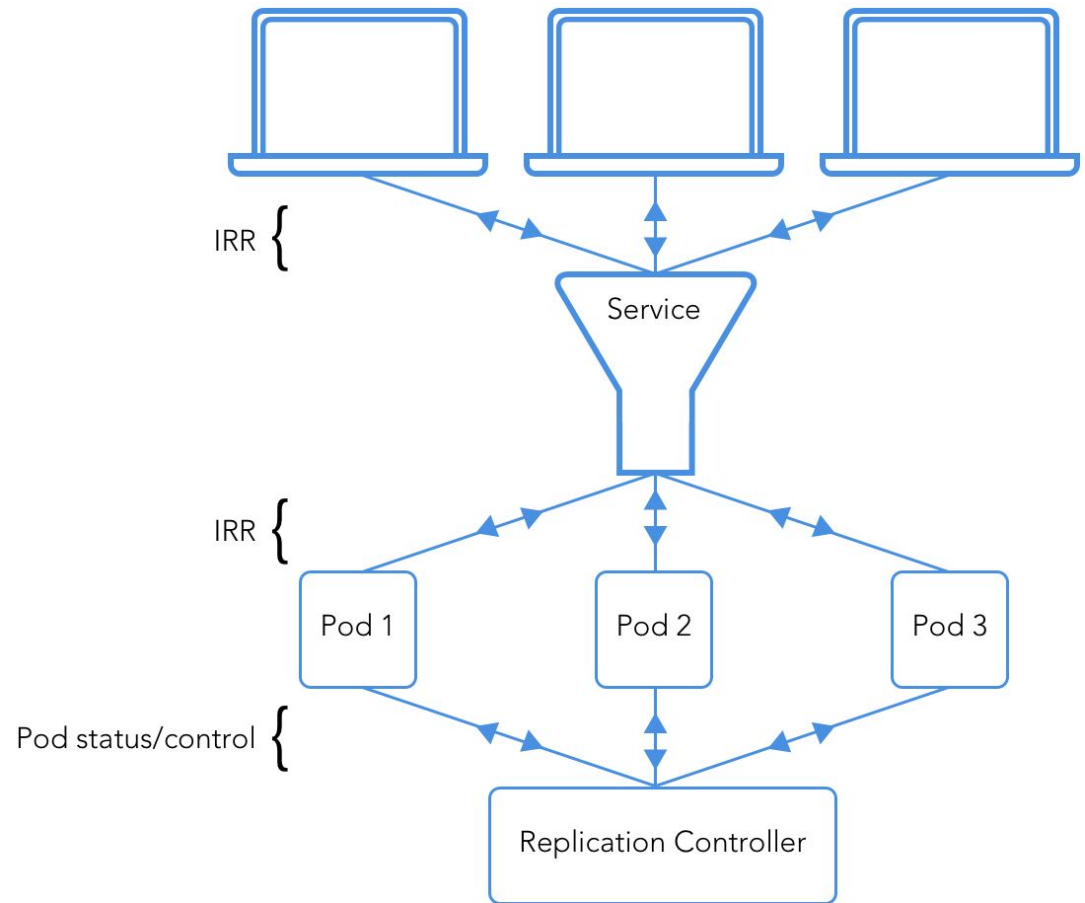
Replication Controllers

A controller for ensuring a given number of replica pods exist.

Services

A single point of load-balancing access for requests to replica pods.

Architecture



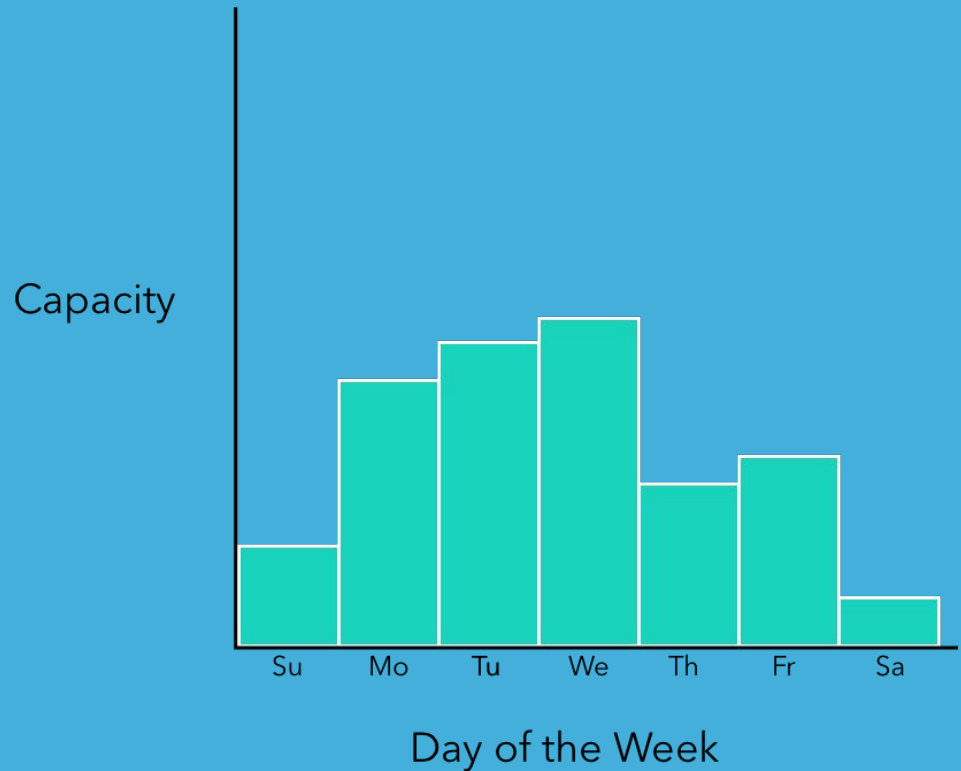
IRR = Individual responses and requests

Accomplishing Specific Goals

How does (predictive) auto-scaling in Kubernetes improve the summation of ERU and QOS?

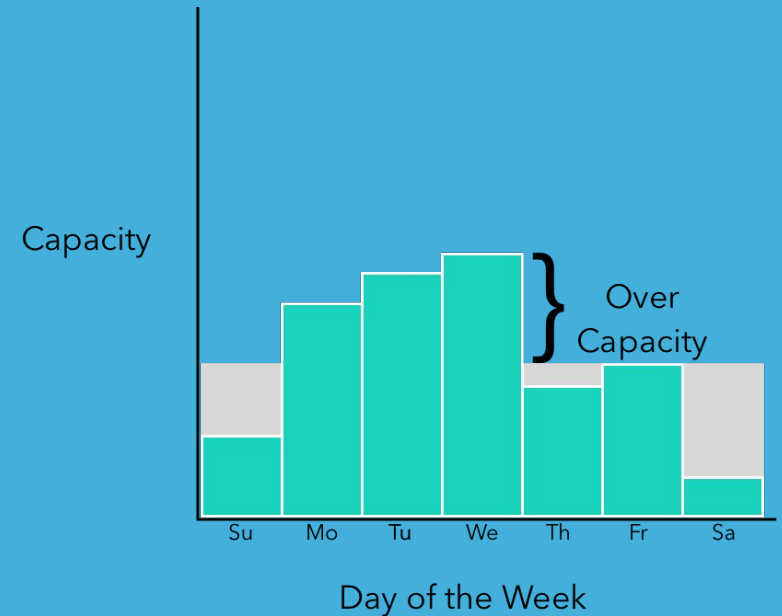
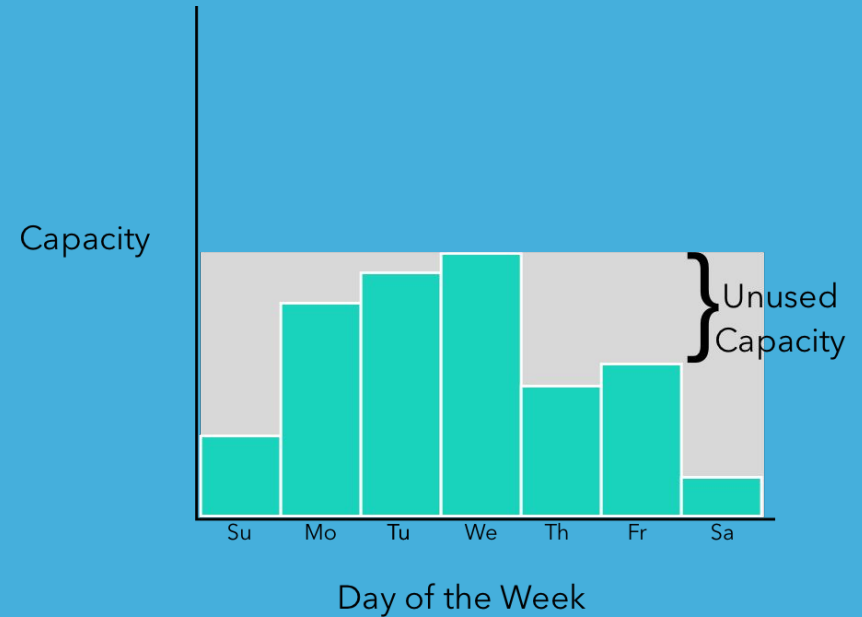
Benefits of auto-scaling

Imagine the following capacity for hbogo.com when running on a cluster manager...



If we do **not** have auto-scaling

No matter what, either poor QoS or poor ERU.

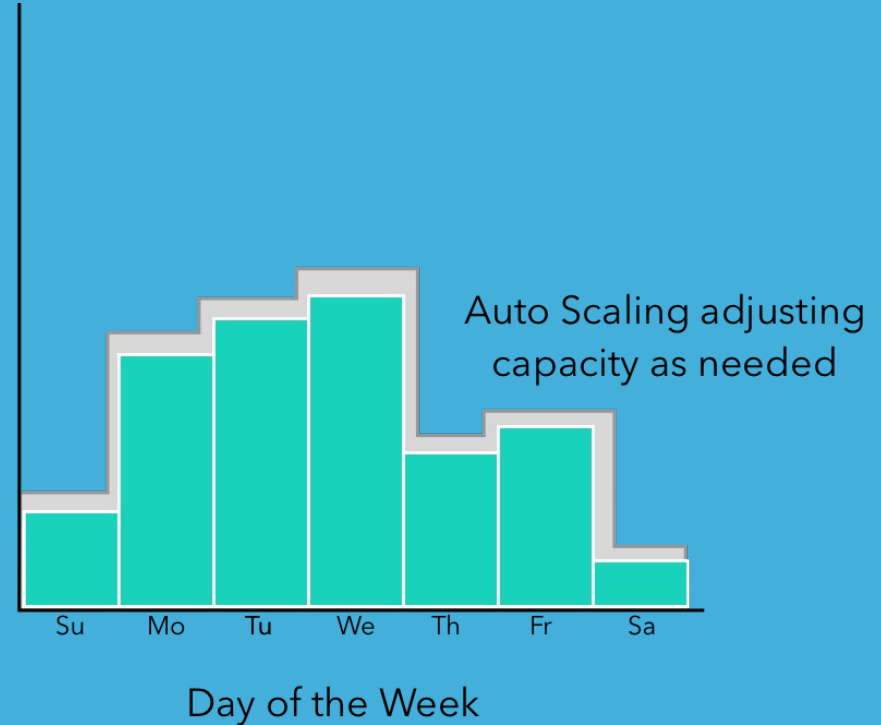


■ If we do have auto-scaling

We can assign the application the exact resources it needs, when it needs them...

improving the summation of ERU and QOS.

Capacity



What are the **different characteristics** of auto-scaling?

Horizontal vs Vertical

How is an application given the extra resources that it needs?

Reactive vs Predictive

Does auto-scaling occur based on the current or future state of the cluster?

What are the **major types** of auto-scaling?

Threshold-based Rule Policies

Scale if the current resource usage is not in accordance with a set of predefined rules (i.e. Amazon EC2).

Time-series Analysis

Auto-scale based on repeating pattern in the application load (i.e. Netflix).

Control-theory (Feedback Control)

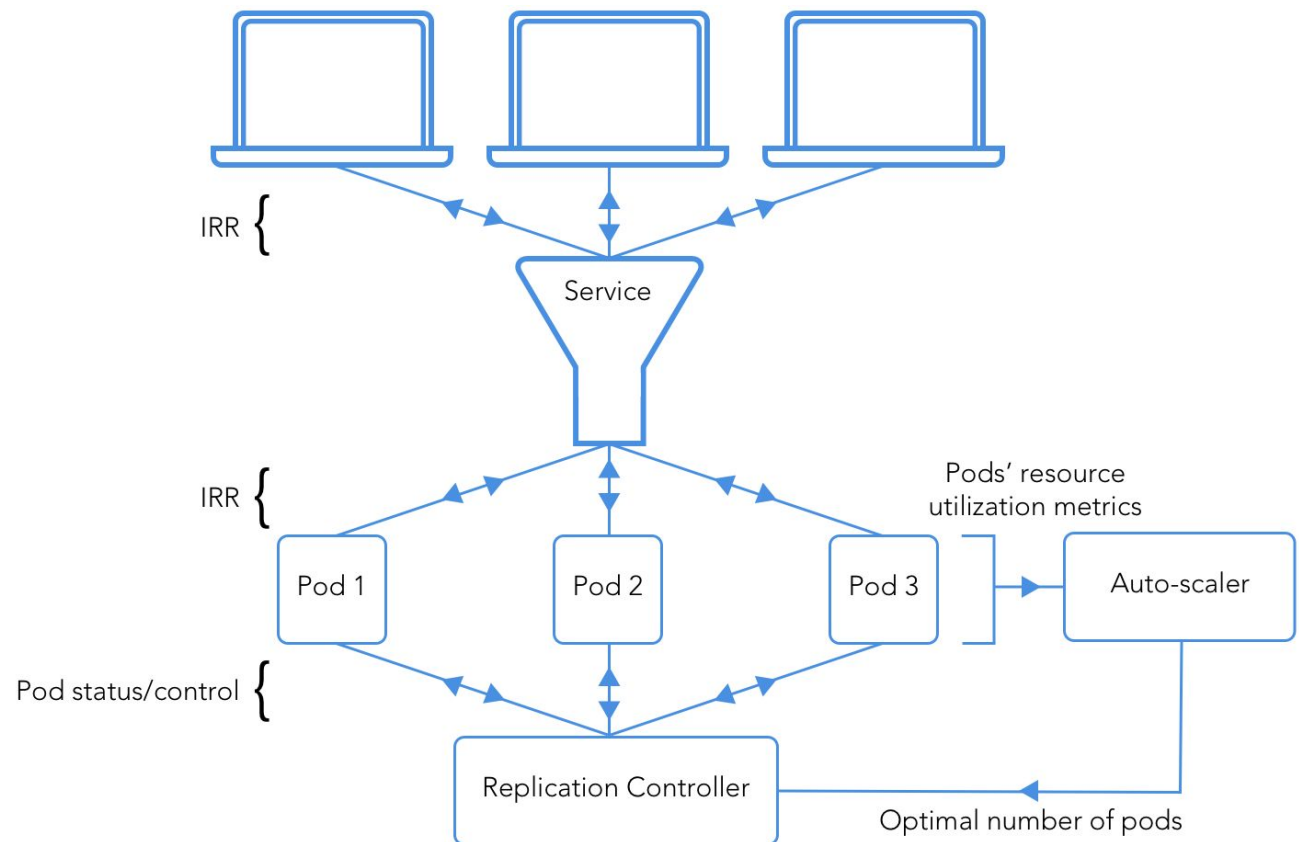
Scale such that the resource usage is in accordance with predefined guidelines (i.e. Kubernetes!).



Current State of **Auto-scaling** in Kubernetes

Kubernetes currently implements reactive,
horizontal feedback control based auto-
scaling of pods.

Architecture



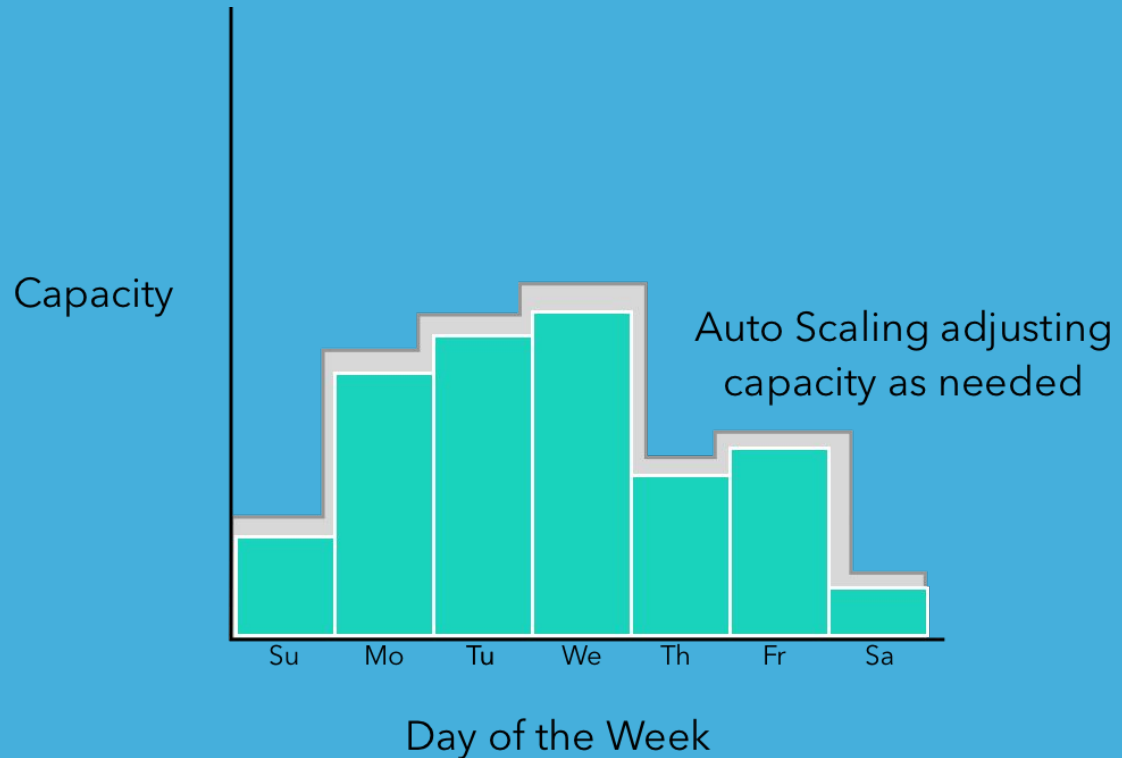
IRR = Individual responses and requests

Concerns with Auto-scaling in Kubernetes

Are there ways to improve the summation of ERU and QOS?

Delayed Pod Initialization Time

What if it takes a long time for a pod to be ready to share in the computational work?



Improvements to Auto-scaling in Kubernetes

What if we add prediction?

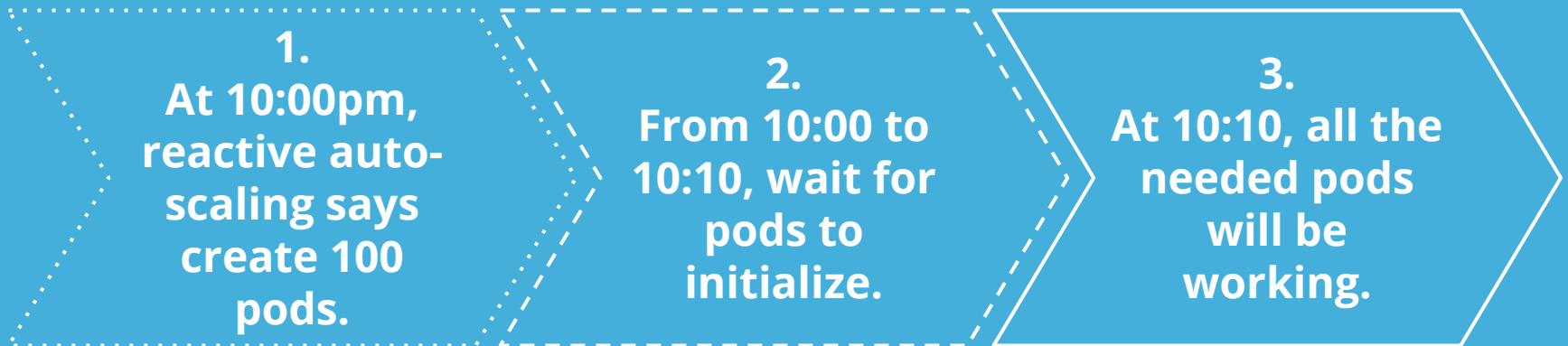
■ **Benefits** of adding prediction

- Predictive, horizontal feedback control based auto-scaling of pods
- Improves QOS without decreasing ERU

■ A case study

- Imagine at 9:50pm, hbogo.com needs 100 pods, and at 10pm the season of premier of Silicon Valley is shown, so we need 200 pods.
- Imagine pods take 10 minutes to download the video files they will serve.

Reactive



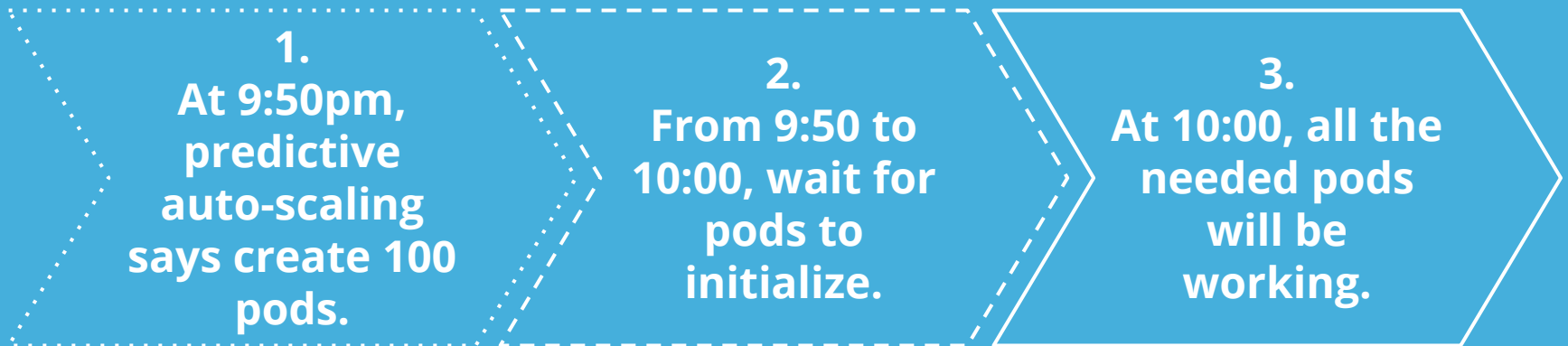
For **10 minutes**, hbogo.com operates with only **half** the resources it needs, while we wait for the replica pods to initialize.



HBO GO



Predictive



hbogo.com **always** has the resources that it needs.





Implementing Predictive Auto- scaling

How do we actually do this?

Implementation questions?

How long does it take for a pod to be ready to share in the work?

How can we predict the future resource utilization of an application?

How does a Kubernetes user enable predictive auto-scaling?

Evaluation

How will we know if we're successful?

Does predictive auto-scaling **increase** ERU + QOS?

How to combine ERU
and QOS?

What applications
will we try to auto-
scale?

What will be the
external environment
of these applications?

THANKS!

Any questions?

CREDITS and CITATIONS

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash
- Thanks to Andrew Udell for assistance with the graphs and images.
- All Kubernetes info is from <http://kubernetes.io/>.
- Lorigo-Botrán, T., Miguel-Alonso, J., and Lozano, J. A. Auto-scaling Techniques for Elastic Applications in Cloud Environments. Research EHU-KAT-IK, Department of Computer Architecture and Technology, UPV/EHU, 2012.