Kubernetes VMware User Group

Intro: Best Practices for Running on VMware

Steven Wong
Open Source Software Engineer
VMware

Myles Gray
Senior Technical Architect
VMware

November 19, 2020

Agenda

Overview of vSphere cloud provider and related storage plugins:

- coverage of recent features/changes.
- Recommended path for migration from the deprecated in-tree storage plugin to CSI.

New features for running Kubernetes on "desktop" hypervisors.

How to get involved in the User Group to meet other users to share advice and experiences.

The vSphere Cloud Provider

What does it do?

Cloud providers are what makes Kubernetes “cloud native”

- plug-in abstraction layer that links to underlying infrastructure in a public cloud or on-prem

The vSphere Cloud Provider:

- Supports reporting the availability zones of underlying infrastructure tying this in with the CSI storage plugin
- Does not support a specific load balancer, routes, or interface to return a cluster list

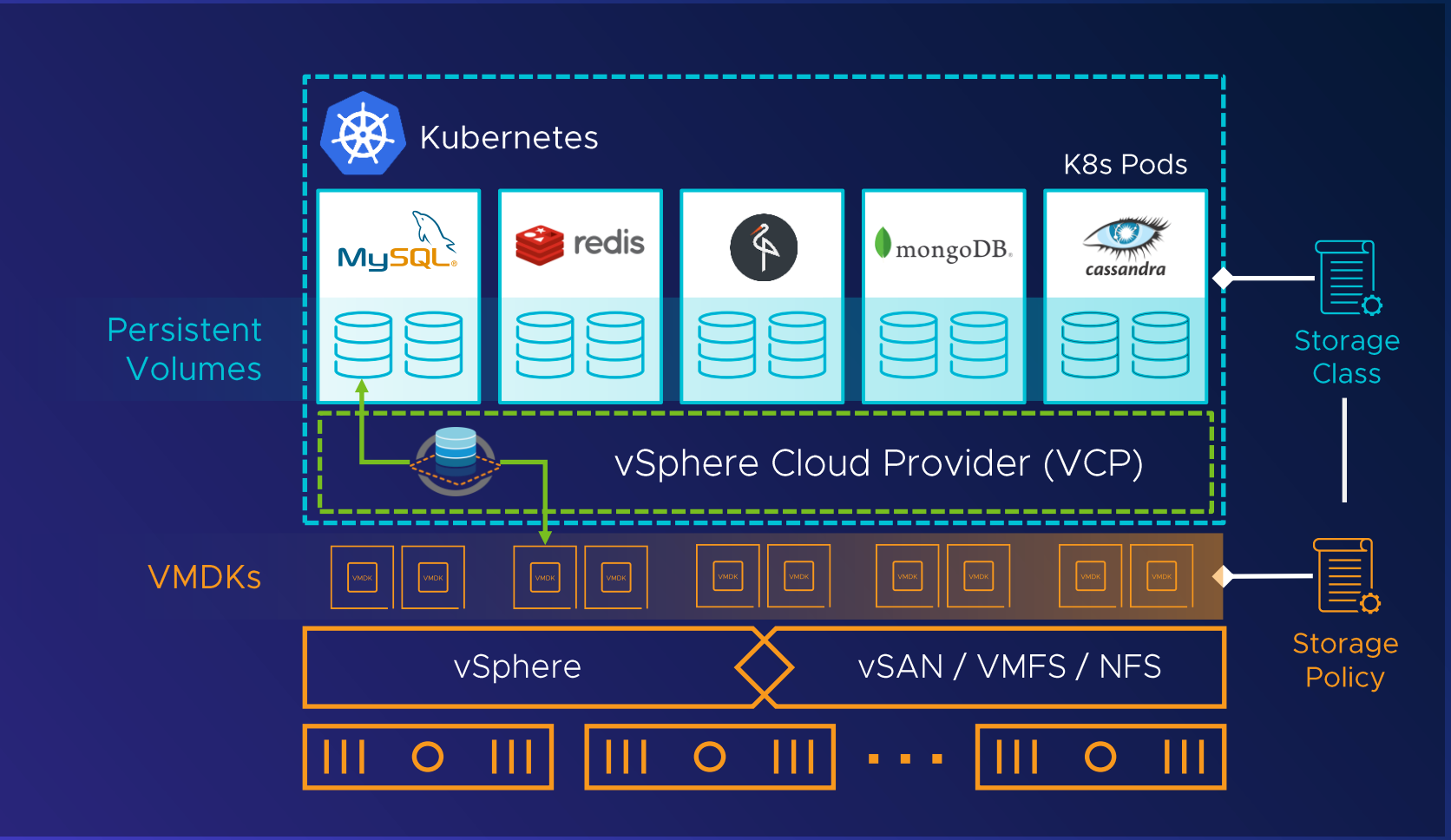
For more details see: cloud-provider-vsphere.sigs.k8s.io/

Intro to storage on vSphere

The VCP, CPI and CSI

vSphere Cloud Provider (VCP) for Kubernetes

The past



Natively built into Kubernetes

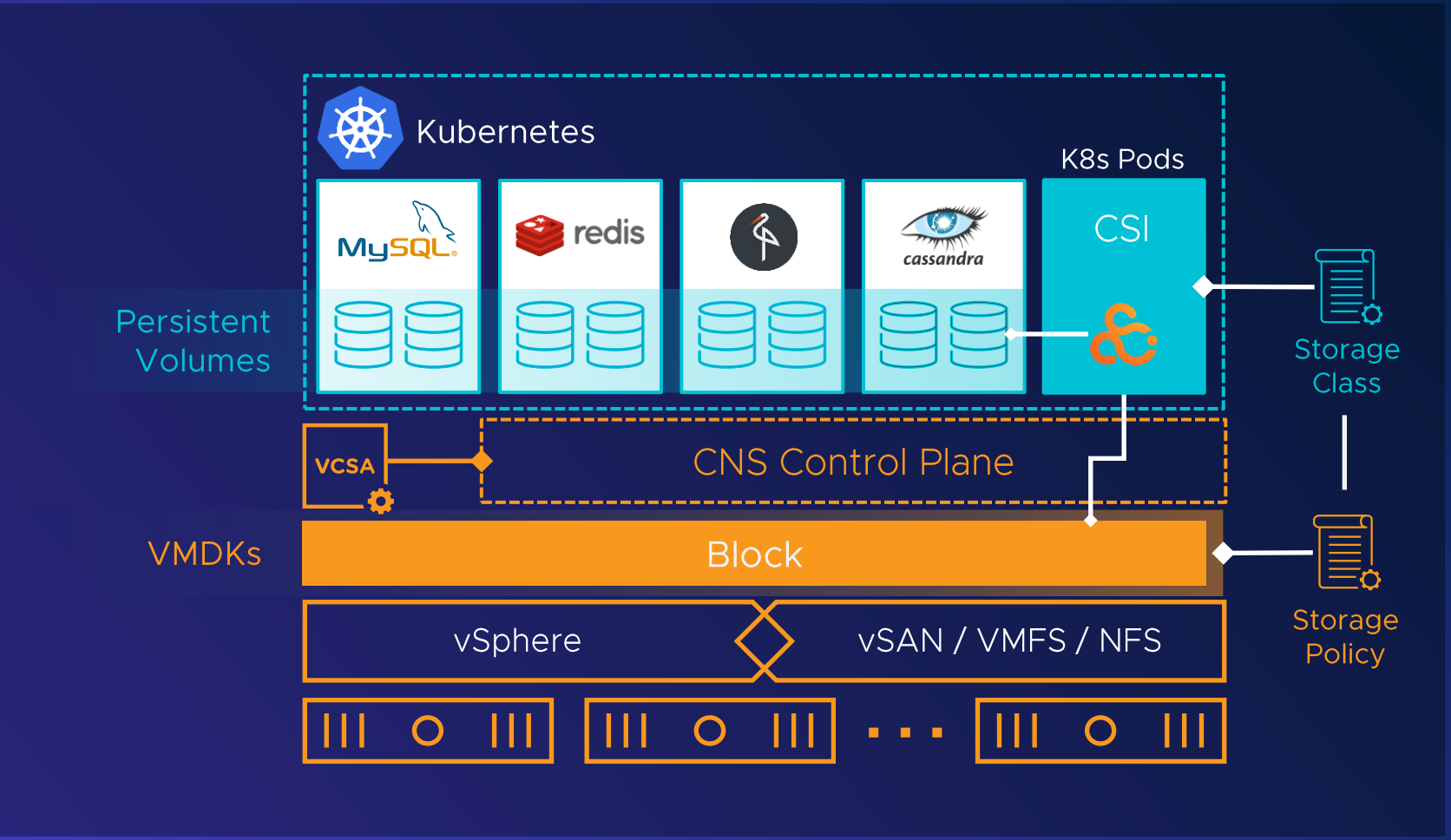
Policy driven dynamic provisioning of Kubernetes persistent volumes

Data services at a granularity of a container volume via SPBM

Not without its drawbacks

vSphere 6.7 Update 3 – CNS Platform Introduction

The present



Built on the CSI standard for container storage

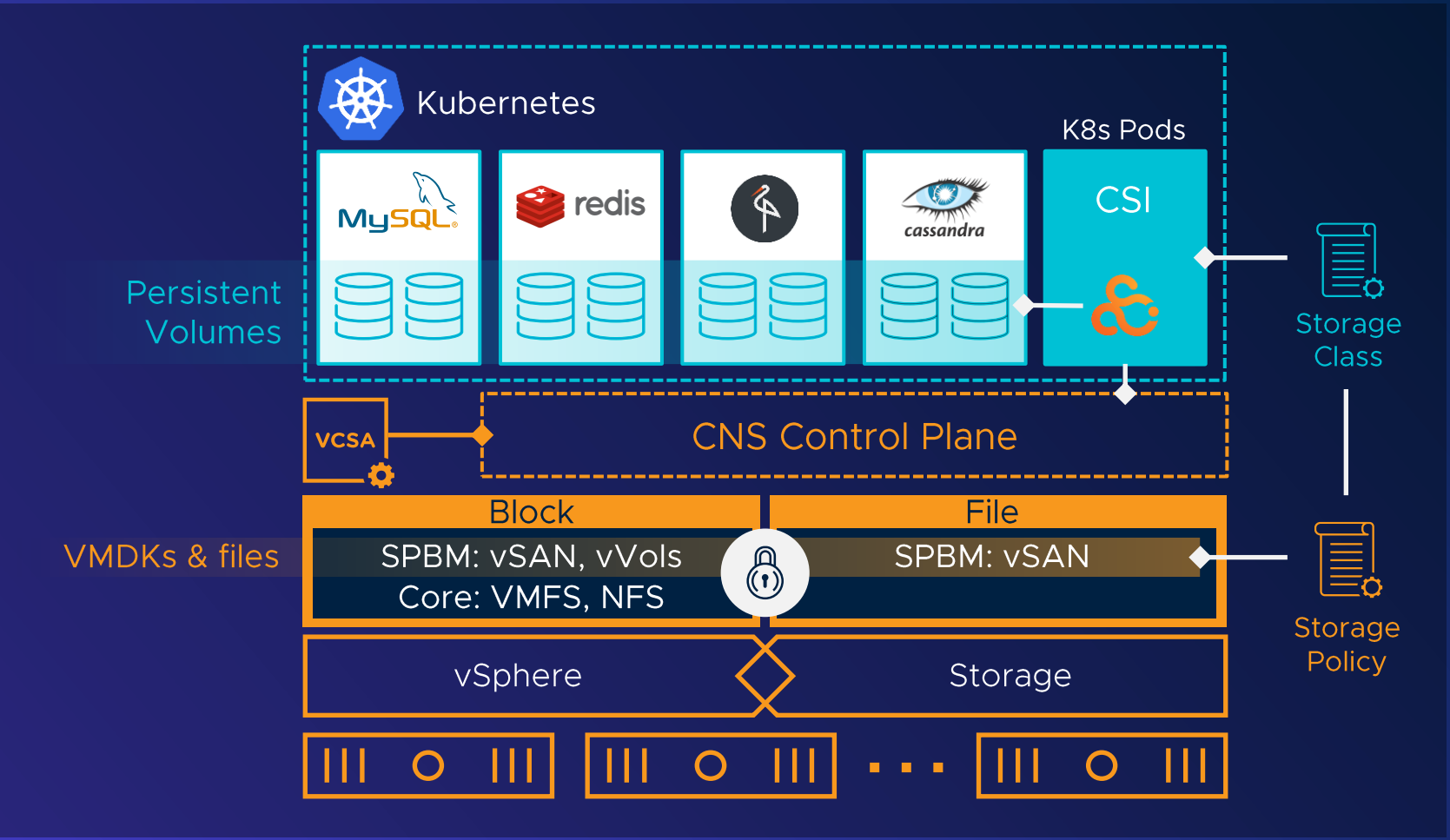
Policy driven dynamic provisioning of Kubernetes persistent volumes

Enabling operational consistency between VM and container infrastructure management

Abstracts the storage infrastructure for developers

Continued Integration of Cloud Native Storage in vSphere and vSAN

The present



Offer file-based persistent volumes on vSAN

Supports basic vVol primitives

Enable persistent volume encryption and snapshots

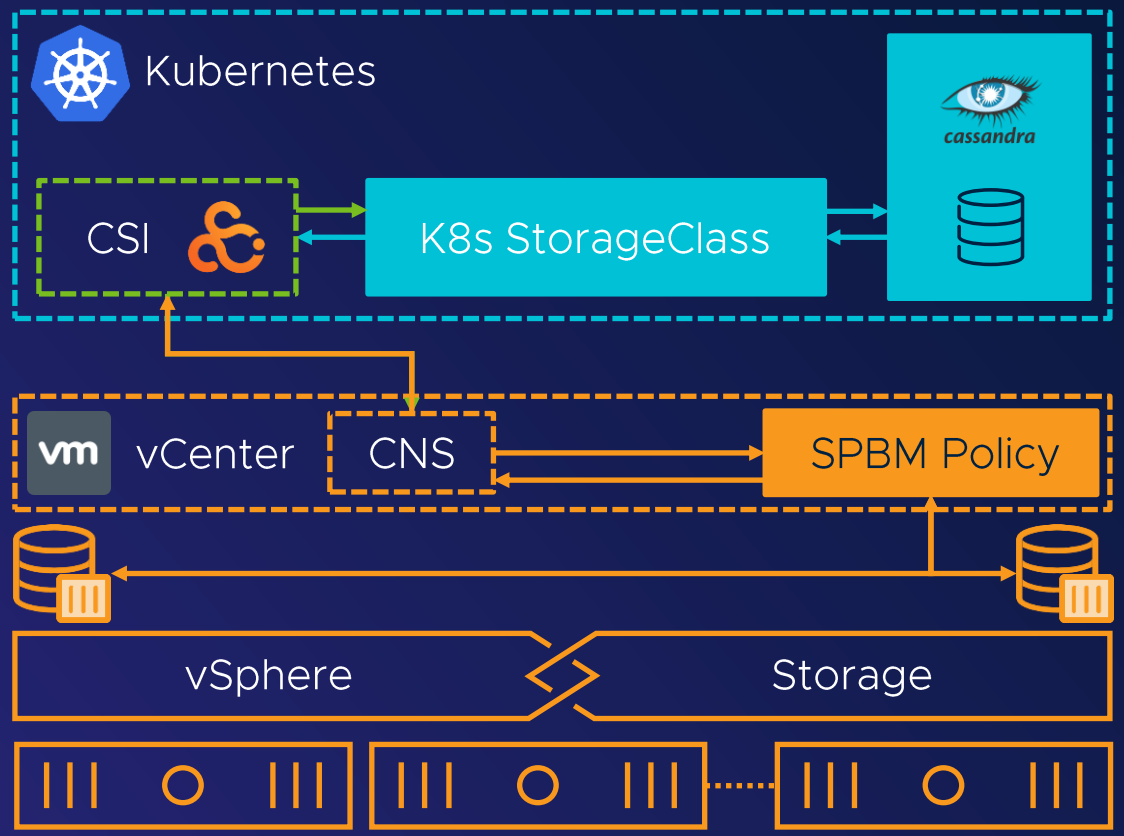
Supports volume resizing

Supports a mix of tooling

- Wavefront
- Prometheus
- vR Ops

Policy Based Management for Kubernetes Workloads

Dynamic Provisioning Workflow for Block Container Volumes



Dynamically create volumes on tiers of storage

Preferred method of storage provisioning

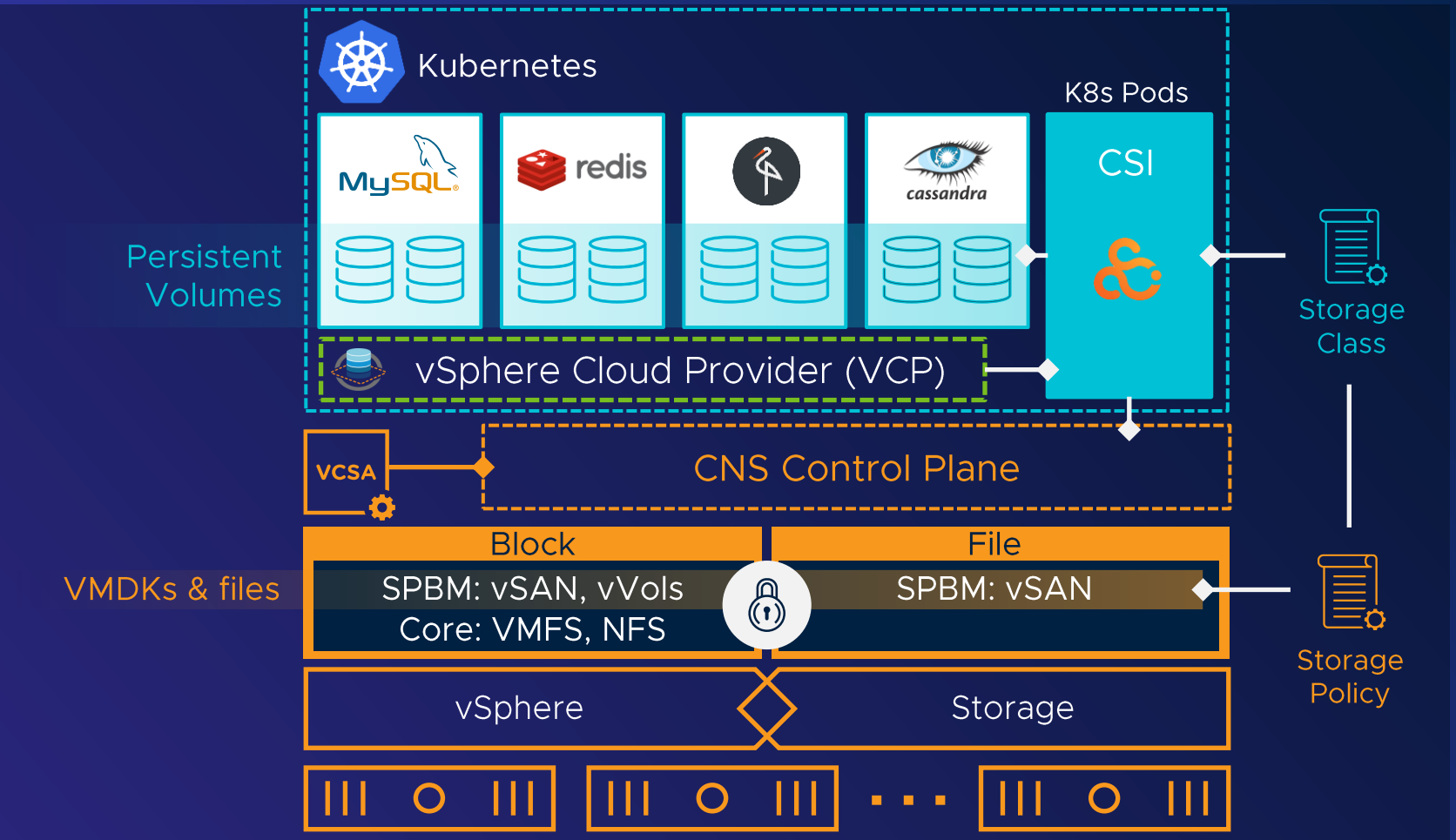
Admin intervention not required

Completely automated volume LCM

VCP to CSI Migration

Beta in K8s 1.19 and vSphere 7.0 U1

Beta



- Offers migration from legacy VCP to supported CSI driver
- Transparent to the application
- Volumes converted to FCDs and included in CNS UI
- Requires new CSI driver and vSphere version

Migration from in-tree to new out-of-tree

What do you need to know

Projected cut off date. Is around K8s v1.21

Out of tree is already recommended for most new installations

Older vSphere versions will be deprecated

How to migrate:

- Cloud Provider Configuration
- CSI from in-tree
  - Configuration
  - Existing persistent volumes

Recently added Features + Changes

What are they? How to Use Them

Recent / Planned Changes

Warning this session was pre-recorded so this is based on status as of mid October - we will update actual status during Q&A

[BUG] Fix a bug when discovering a VM's address by it's subnet or VM network name. PR: #378
[BUG] Fix a bug where a node may be prematurely deleted if vmtools is slow to report it's hostname. PR: #387

Kubernetes on Desktop Hypervisors

Why?
How?

Why run Kubernetes on a desktop/laptop?

Can't I just use a cloud hosted cluster?

local has advantages in some circumstances:

- It may be cheaper …. and also exhibit faster dev cycle times when internet is limited
- It can be very attractive as a Kubernetes learning environment
- For development local dev clusters to boost productivity:
  - You need to insure that when you develop (builds + test) on your laptop, it is a match to the tooling and environment used for production builds and deployments. Kubernetes in both places can cover this requirement
  - Consider how well your local dev cluster can fit into your CI/CD patterns and tooling.

So I want a local Kubernetes dev cluster

What are my options?

minikube

VM

KinD runs Kubernetes inside a “Docker” container and has low resource demands and fast cycle times.

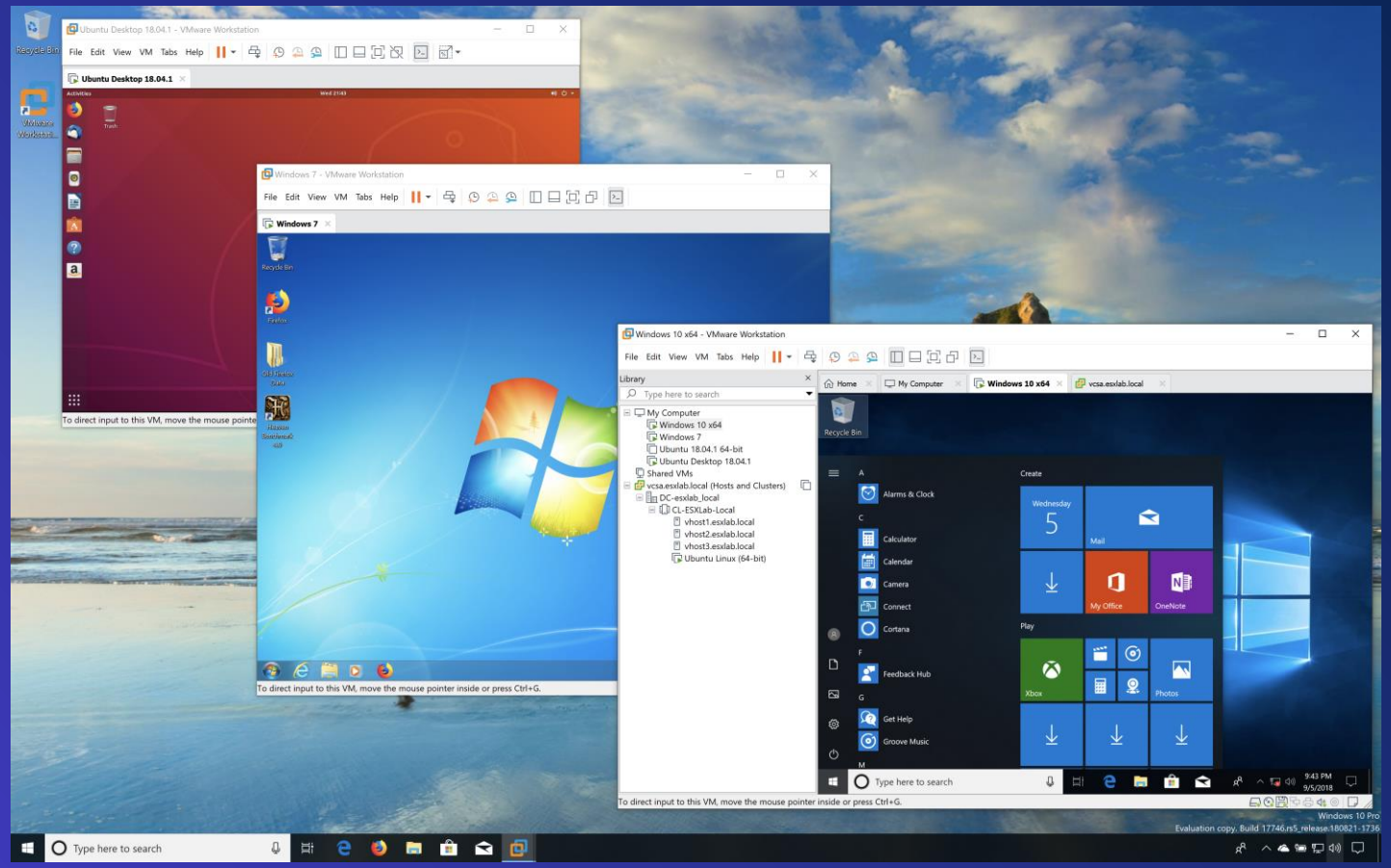
The other options might offer a more “high fidelity” cluster at great resource and cycle time costs. Kind is routinely used by many of the devs of Kubernetes itself.

Kind’s fast cluster creation and tear downs can lead to short lived clusters. You aren’t tempted to postpone patches and maybe you can afford to test against multiple K8s versions.

See the KubeCon Europe recording for minikube on desktop hypervisor coverage

VMware Desktop Hypervisors: Fusion + Workstation

New feature: lightweight VM for container hosting including built-in Kubernetes kind



VMware Workstation

Version 16 for Windows and Linux

VMware Fusion

Version 12 for MAC

Kubernetes on Desktop Hypervisor

Demo steps - Windows shown here (Linux and MAC are similar)

1. Install Workstation 16 on Windows or Linux, or Fusion 12 on MAC
2. Install Chocolatey at admin command prompt

```
@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -
InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object
System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET
"PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin“
```

3. Use Chocolatey to install the standard kubectl Kubernetes CLI

```
choco install kubernetes-cli
```

4. Also install the Tekton CLI to be used later in the demo

```
choco install tekton-cli --confirm
```

Demo stage 1 – basic Docker like functionality built in

Demo steps - Windows shown here (Linux and MAC are similar)

So the new container and kind feature is based on technology similar to what went into vSphere 7 for hosting containers within a rapidly stageable VM. kind is a binary for deploying Kubernetes components into a Docker runtime. With Workstation 16 what you have is a version of kind that thinks it is talking to a standard container runtime interface like the one used by Docker.

So the desktop hypervisors start with a cli that is intentionally similar the the Docker cli you are already familiar with.

Let me show you by typing in vctl with no parameters. You get usage guidance and you see the familiar build, push, pull run, tag, etc.

```
vctl
```

So let me prove it with the classic hello-word

```
vctl pull hello-world
```

I know your saying wow Steve messed up the demo already – but trust me I did it on purpose. The error tells you what to do to fix the issue. The container engine needs to be started. This is intentional so that when you don't need the container runtime, it can be completely shut down. Something you find appreciate when you need the most of you memory, cpu or battery on a laptop.

Demo stage 2– enable container runtime

and I will type in vctl system start as suggested by the previous hint

```
vctl system start
```

And after a few seconds we see that a container runtime is available.

We can type in vctl system info to see what is available to the container runtime if needed

```
vctl system info
```

This is configurable if needed and as you see the output explains what the current settings are and where the config file and log lives

Now in the interest of time I am not going to show it, but if I was to look at me system resource, the amount of cpu and memory shown here is NOT in use at this time because no containers are running.

In fact we can use the hypervisor CLI to show that no VMs are running

```
vmrun list
```

Demo stage 3 – basic container support

Now that the container runtime is available let's try that Docker hello-world exercise again

```
vctl pull hello-world
```

```
vctl run hello-world
```

The hello-world example is a short lived container and we can see that after its run, we are back to no VMs running.

Now if this was just a container runtime for Windows it wouldn't really be that interesting. We want full Kubernetes, not just Docker.

Demo stage 4 – create a Kubernetes cluster

Lets deploy a kind cluster.

```
vctl kind
```

You see a new window popped up – and this windows is set up to use kind

```
kind create cluster
```

This will take a few minutes, likely more than one and less than 5 - the time depends on your speed of your system and your network connection

I am going to speed this up in this recorded demo video

Demo stage 5 – examine Kubernetes cluster

Now that it's finished, Lets try a few standard kind, Docker and Kubernetes commands.

```
vctl images
kubectl version
kubectl –n kube-system get all
kubectl cluster-info
kind get clusters
kubectl get nodes -o wide
vctl ps
vmrun list
```

That last one is a hypervisor cli command showing that we have a VM running to host this

Demo stage 6 – segue to CI/CD

So we have a standard Kubernetes cluster with standards functionality. Nice, but for that developer workflow I talked about –it's a foundation but still not a complete toolkit.

What about CI/CD.
Tekton is a popular open source framework for driving CI/CD by hosting steps and pipeline in a portable way by taking advantage of Kubernetes. The goal is that you build test deploy steps can run anywhere and can't tell the difference between when they run on my laptop or when they run in a public cloud. The result should be identical anywhere, no snowflake build steps.

We can use our kind cluster to host the additional tools we need.

This isn't going to be a talk on Tekton but let me at least show you how you can use a desktop hypervisor as part of your CI/CD process.

Demo stage 7 – deploy Tekton to Kubernetes

We will start by installing Tekton into our Kubernetes cluster with kubectl. I am going to downloadi and apply the yaml from the Tekton open source project. Maybe downloading directly from the latest tag on github is living a little dangerously, but hey this is recorded. I would do the tired pray to demo gods thing. If it blows up I can secretly re-record and you'll never know it happened. But I am confident this will work.

```
kubectl apply --filename https://storage.googleapis.com/tekton-releases/pipeline/latest/release.yaml
```

```
kubectl get pods --all-namespaces
```

And it looks like we've got a couple Tekton pods starting up.

Demo stage 8 – a Tekton hello world example

Tekton runs CI/CD steps as something it calls tasks. It is a framework for running a CI/CD process of your choice on Kubenetes.

Rather than run a real full build test deploy, In the interest of time I am going to run a Tekton Hello World example that pretends emitting Hello World is a step like a buil within your process.

This comes from a blog post from Brian McClain that I will link in the deck

```
kubectl apply -f https://raw.githubusercontent.com/BrianMMcClain/tekton-examples/main/hello-task.yml
```

Demo stage 9 – wrapup

I'll use the previously installed Tekton CLI to show the status of the task run.

```
tkn taskrun describe echo-hello-world-task-run
```

Not much to see before this is a contrived example with no inputs or outputs but this did run

And we can use another Tekton CLI command to examine the logs

```
tkn taskrun logs echo-hello-world-task-run
```

I am going to quit here because this isn't a session about CI/CD – I just wanted to show how Kubernetes on a desktop hypervisor has the potential to be useful as a development process including one based on CI/CD. A real workflow could build Docker images, test them and push the result to an image registry.

Post Demo cleanup – shut things down and reclaim resources

```
kind delete cluster
```

Verify that no VMs are running

```
vmrun list
```

Shut down the container subsystem

```
vctl system stop
```

Verfify the container subsystem is stopped

```
vctl system info
```

Where to experience more material like this and interact with other users

The Kubernetes VMware User Group

Kubernetes VMware User Group

What is it?

Similar to SIGs and Working Groups - intended to serve the needs of users running Kubernetes on particular platforms.

The VMware User group is the first (and currently only) K8s UG for a platform - covers running K8s on all VMware hypervisors.

Why is this important?

Create community culture among our users

- Users can help each other
- Users can help us make Kubernetes better – and strengthen user experience on our platforms:
  - Feature requests
  - Feedback + issue resolution

Who is involved?

Co-chairs

- Steven Wong, MAPBU CET
- Myles Gray, VMware Storage Tech Marketing, UK

User Co-leads

- Bryson Shepherd, Walmart
- Joe Searcy, T-Mobile

150+ Slack channel participants as of October 2020

Kubernetes VMware User Group

User Group Meeting:

First Thursday each month 11am PT
calendar link

Link to join the group

- groups.google.com/forum/#!forum/kubernetes-ug-vmware

Link to join Slack channel

- https://kubernetes.slack.com/messages/ug-vmware

Speaker contact info

Deck link: https://sched.co/ekHS

Some other related sessions:

Cloud Provider out of tree (next): sched.co/ZeuY

K8s User experience (Thursday): sched.co/Zeue

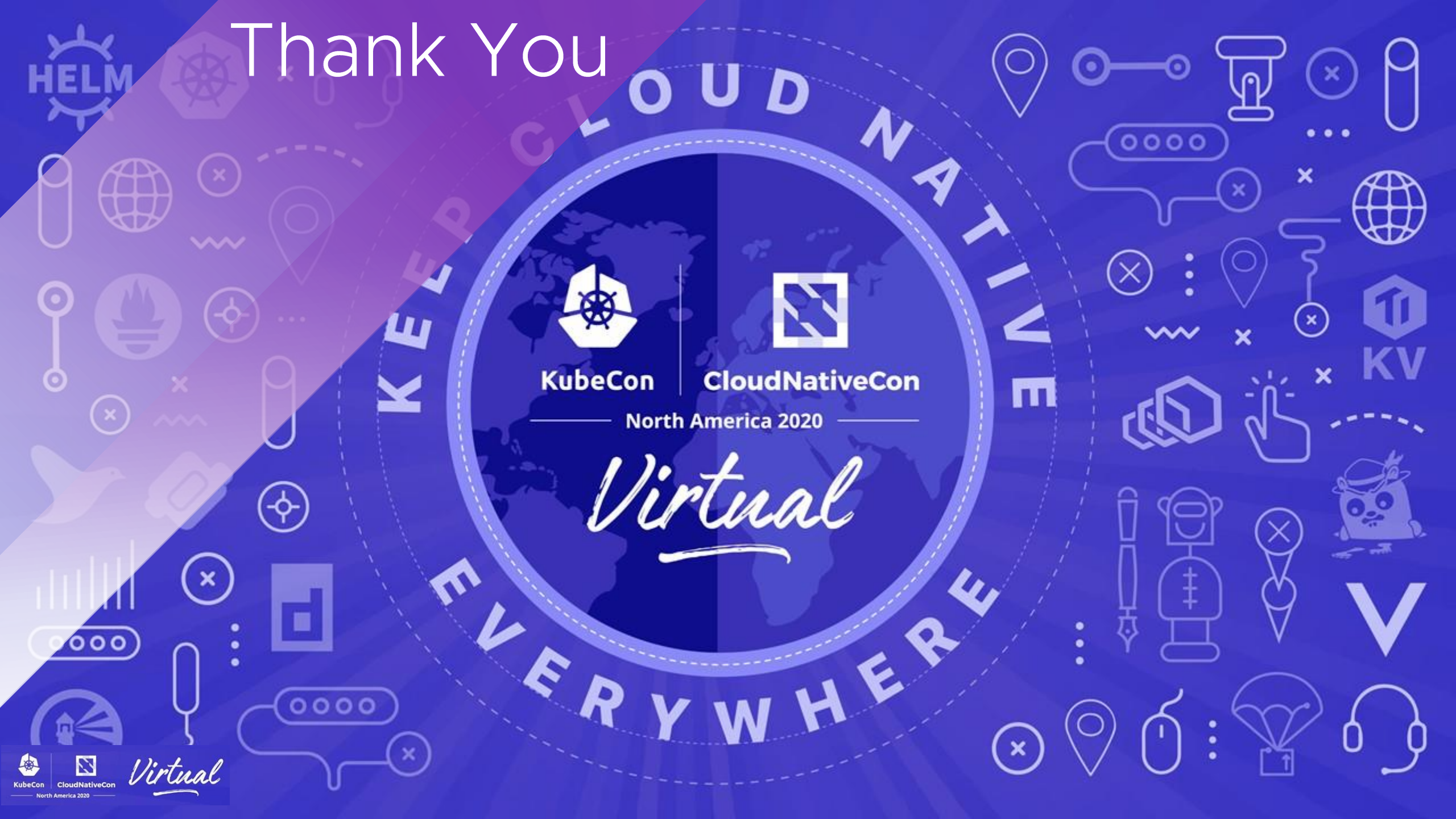
vSphere Cloud Provider (Thursday): sched.co/ZevZ

Myles Gray
VMware

@mylesagray

Steve Wong
VMware
@cantbewong

Thank You


KEEP CLOUD NATIVE
EVERYWHERE
KubeCon
CloudNativeCon
North America 2020
Virtual
HELM
KV