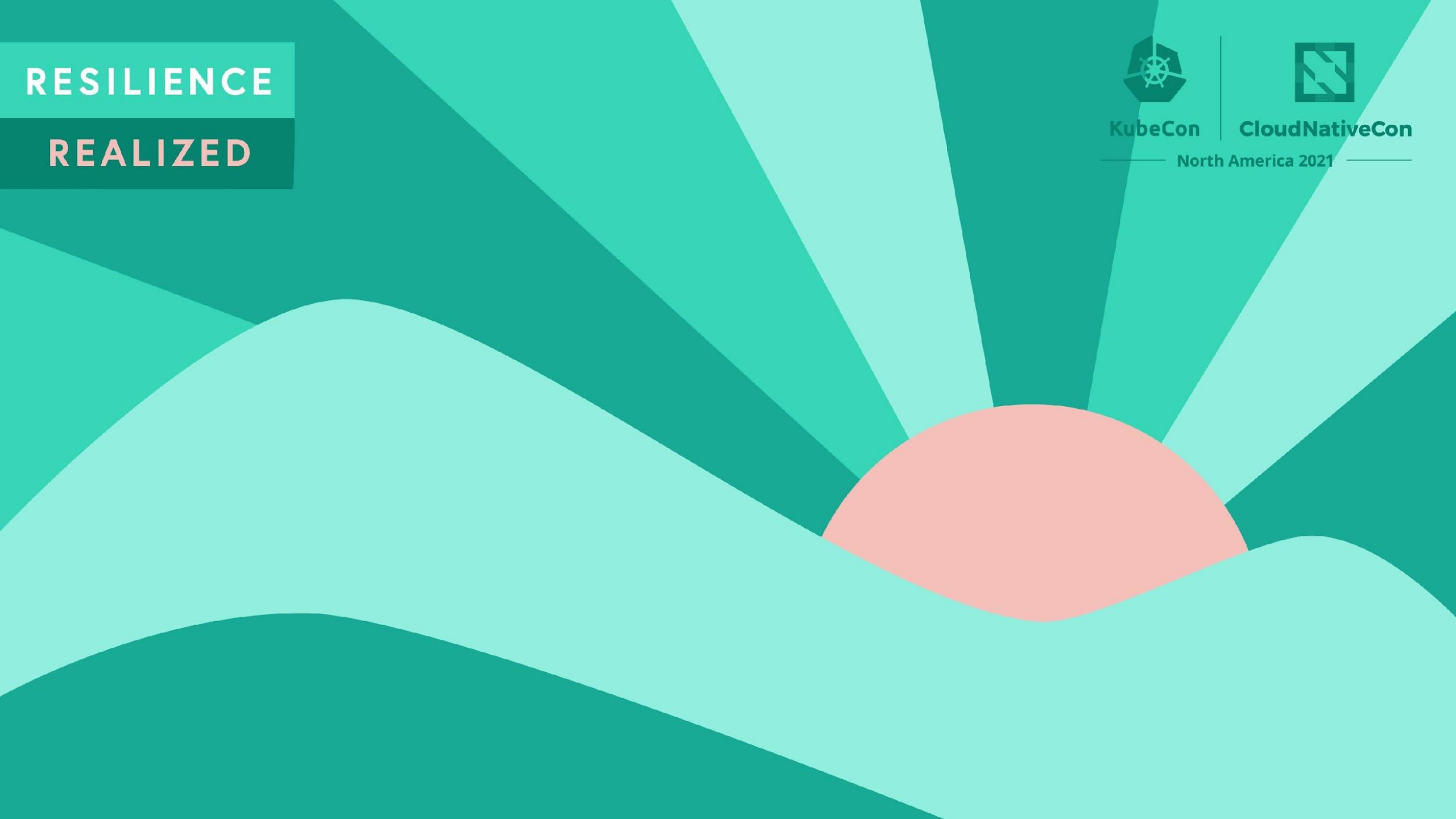
RESILIENCE

REALIZED



KubeCon | CloudNativeCon

North America 2021

RESILIENCE
REALIZED

SIG CLI Updates

Katrina Verey (@KnVerey), Apple
Sean Sullivan (@seank8s), Google
Eddie Zaneski (@eddiezane), AWS

Who?

Chairs:

- Maciej Szulik (@soltysh), Red Hat
- Sean Sullivan (@seans3), Google
- Eddie Zaneski (@eddiezane), AWS
- Katrina Verey (@KnVerey), Apple

Tech Leads:

- Maciej Szulik (@soltysh), Red Hat
- Katrina Verey (@KnVerey), Apple

https://github.com/kubernetes/community/tree/master/sig-cli#leadership

What?

We focus on:

- development and standardization of the **CLI framework** and its dependencies
- the establishment of **conventions** for writing CLI commands
- POSIX **compliance**
- improving the command line tools from a developer and devops **user experience and usability** perspective

https://github.com/kubernetes/community/tree/master/sig-cli#cli-special-interest-group

What?

Subprojects:

- kubectl
- kustomize
- krew + krew index
- kui
- cli-utils + cli-runtime + cli-experimental

https://github.com/kubernetes/community/tree/master/sig-cli#subprojects

Where?

Slack:

#sig-cli

Mailing list:

https://groups.google.com/g/kubernetes-sig-cli

https://github.com/kubernetes/community/tree/master/sig-cli#contact

When?

Biweekly meeting:

Wednesday at 09:00 PT

Biweekly Kustomize/Kubectl bug scrubs:

Wednesday at 09:00 PT

https://github.com/kubernetes/community/tree/master/sig-cli#meetings

Enhancements

KEP 2775 - kubectl delete protections

Summary:

- Adds a new `--interactive | -i` flag to kubectl delete that will require confirmation before deleting resources. This flag will be false by default.
- Adds a warning to `kubectl delete [--all | --all-namespaces]` about the destructive action that will be performed and artificially delays the command for x seconds, allowing users a chance to abort.

Update:

- Provisional: seeking feedback

Enhancements

KEP 1441 - kubectl debug

Summary:

- Adds a `kubectl debug` command to improve the user experience of troubleshooting pods, nodes and containers.

Update:

- New "debugging profiles" will provide more configurability for generated pods and containers. For example, a user may use `--profile=netadmin` when debugging a node to create a pod with the `NET_ADMIN` capability.

Enhancements

KEP 555 - Server-side apply

Summary:

- Move "apply" and declarative object management from kubectl to the apiserver in order to fix many of the existing bugs that we can't fix today. Also use that opportunity to add "field ownership".

Update:

- Graduated to GA in release v1.22!
- Usage: `kubectl apply --server-side` (NOT on by default)

Enhancements

KEP 859 - kubectl command metadata in http request headers

Summary:

- kubectl adds http headers with the subcommand name, which flags were specified and a session id.

Update:

- In beta as of release v1.22
- This is a step towards getting telemetry on kubectl usage

Enhancements

KEP 2953 - Kustomize Plugin Graduation

Summary:

- Converge Kustomize's various alpha extension mechanisms into a single KRM-driven feature that has an enhanced story around plugin distribution, discovery and trust.

Update:

- Provisional: seeking feedback
- Related KEPs outlining details of sub-enhancements:
  - KEP 2299: Kustomize Plugin Composition API (implementable)
  - KEP 2906: Kustomize KRM Plugin Catalog (provisional)
  - KEP 2985: Public KRM functions registry (provisional)

Enhancements

KEP 2950 - Add subresource support to kubectl

Summary:

- Add a new flag `--subresource` to kubectl commands like get, patch, edit and replace commands to allow fetching and updating subresources like status, scale, etc.

Update:

- Targeting alpha in release v1.23

Building kubectl

- kubectl is a "staging repo"
- We don't take PRs to kubernetes/kubectl (yet)
- `make kubectl`

Testing kubectl

- We have different types of testing for kubectl
- Unit
- e2e
- Integration (bash)

Unit testing

- https://github.com/kubernetes/kubernetes/blob/master/staging/src/k8s.io/kubectl/pkg/cmd/apply/apply_test.go

```
make test WHAT=./staging/src/k8s.io/kubectl/...
```

e2e testing

- Need a cluster (local or remote)
- https://github.com/kubernetes/kubernetes/blob/master/test/e2e/kubectl/kubectl.go

```
make ginkgo

make WHAT=test/e2e/e2e.test

_output/bin/ginkgo --nodes 5 --focus="Simple pod" --skip="should support exec through kubectl proxy|should support exec through an HTTP proxy|should handle in-cluster config" _output/bin/e2e.test -- --provider=local --kubeconfig=/home/eddiezane/.kube/config
```

Integration testing

- Will try to spin up a cluster
- https://github.com/kubernetes/kubernetes/blob/master/test/cmd/get.sh

```
make test-cmd WHAT="deployment impersonation"
```

kubectl apply/prune

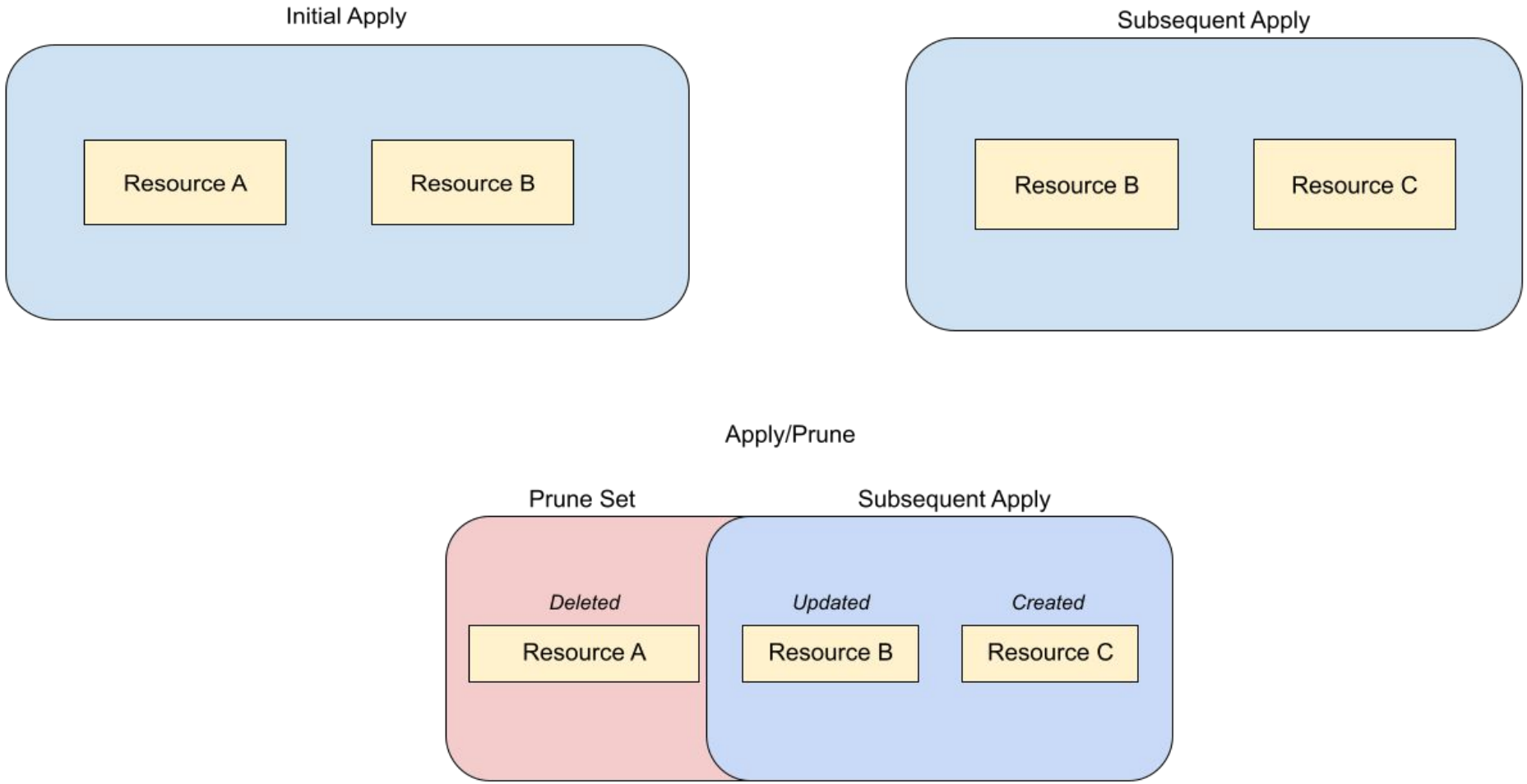
What is Pruning

- During apply, pruning attempts to delete obects that are “no longer needed”
- It requires kubectl users to specify a previously applied set of objects
- Pruning deletes objects previously applied that are not in the current apply set

Prune Set = Previous Apply Set - Current Apply Set

kubectl apply/prune

What is Pruning



kubectl apply/prune

Simple Example: Using Labels to Specify Previous Apply Set

```
$ cat config-maps-1.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-a
  labels:
    app: cm-label
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-b
  labels:
    app: cm-label
```

kubectl apply/prune

Simple Example: Using Labels to Specify Previous Apply Set

```
$ cat config-maps-2.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-b
  labels:
    app: cm-label
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-c
  labels:
    app: cm-label
```

kubectl apply/prune

Simple Example: Using Labels to Specify Previous Apply Set

```
$ # Apply config maps; all have label app=cm-label
$ kubectl apply -f config-maps-1.yaml
configmap/cm-a created
configmap/cm-b created
$ # Next, apply a slightly different set of config maps
$ # Prune, specifying previous applied resources with label
$ kubectl apply -f config-maps-2.yaml --prune -l app=cm-label
configmap/cm-b unchanged
configmap/cm-c created
configmap/cm-a pruned
```

kubectl apply/prune

Another Example: Using prune-whitelist

```
$ kubectl apply -f config-maps-1.yaml
configmap/cm-a created
configmap/cm-b created
$ # Next, apply a slightly different set of config maps
$ # Prune, specifying previous applied resources with label
$ kubectl apply -f config-maps-2.yaml --prune --all
--prune-whitelist=core/v1/ConfigMap
configmap/cm-b unchanged
configmap/cm-c created
configmap/cm-a pruned
```

kubectl apply/prune

Drawbacks

- Using labels to determine previous apply set is error prone
- The default set of GVK's is hard-coded; CRD's don't work by default
- Differing namespaces between applied resources is problematic
- Dangerous!
- `kubectl apply/prune` is alpha and probably won't ever graduate to beta
- https://github.com/kubernetes-sigs/cli-utils

Top of mind

- Refactoring kubectl commands
- How do we better handle flags
- kubectl performance
- New contributors!

Thanks!

Questions?

@KnVerey
@seank8s
@eddiezane