Self Managing Clusters at Scale

Kevin Downey | Eytan Avisror

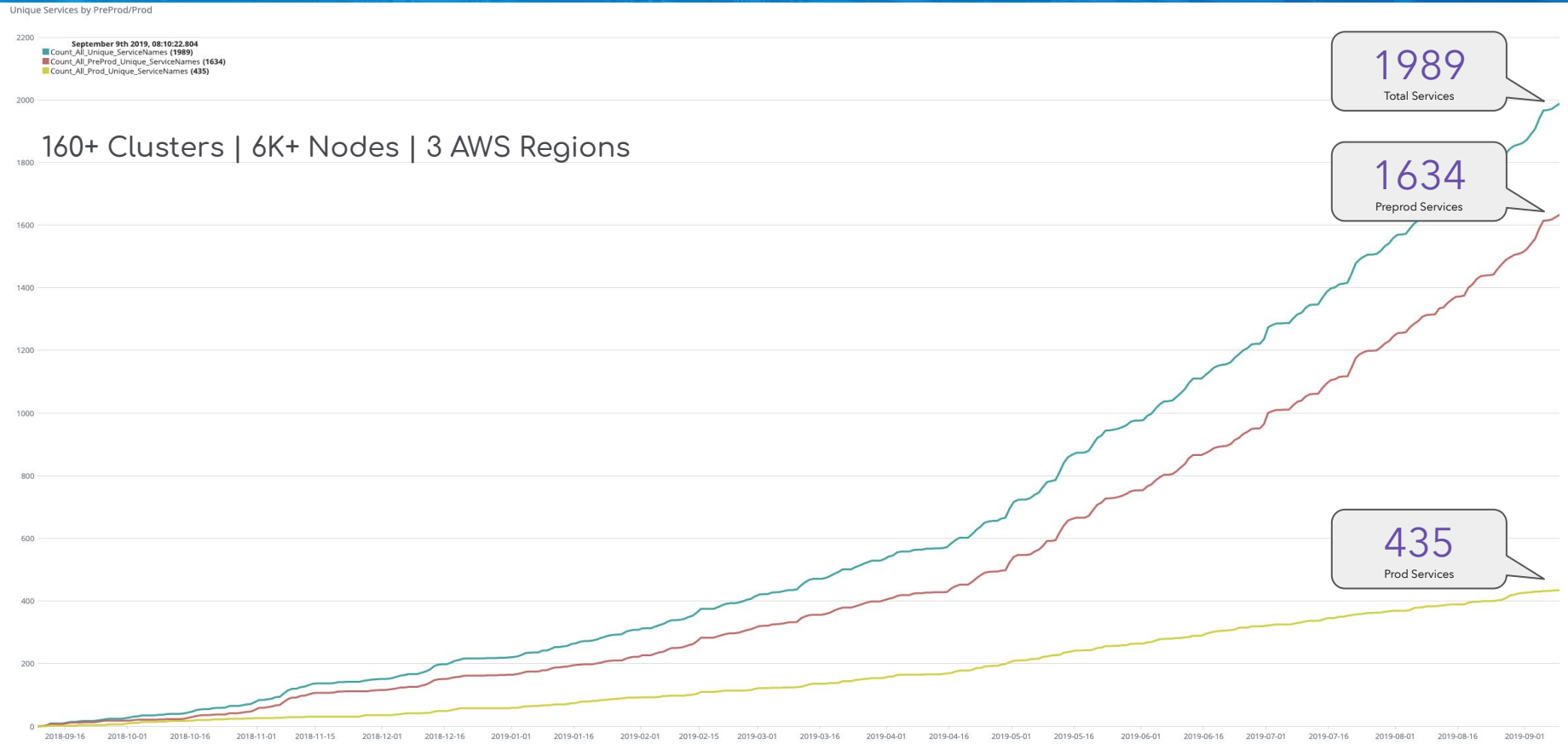
Eytan Avisror

Staff Software Engineer, Intuit
Core contributor, Keikoproj
github @eytan-avisror

Kevin Downey

Staff Software Engineer, Intuit
Core contributor, Keikoproj
github @kevdowney

Kubernetes @ Intuit



Modern Developer Platform

Kubernetes Powered, GitOps Driven

Kubernetes Powered, GitOps Driven

Application Repository

A git repository containing a set YAML and Kustomize files that make up your deployment.

Kubernetes Namespace

A share of a Kubernetes cluster using either a shared or dedicated node group.

Continuous Deployment with ArgoCD

ArgoCD enables GitOps between the repository and the Kubernetes cluster.

The Pipeline

Declarative build and deploy in seconds.

Common Problems

How do I manage nodes?

How do I manage critical cluster services?

How do I optimize the cost?

How do I ensure resiliency?

Keiko -
Enable Kubernetes at scale ...

What is Keiko?

Keiko provides a set of independent open-source tools for management of multi-tenant, reliable, secure and efficient Kubernetes clusters at scale.

github.com/keikoproj

Components

Orchestration

Addon-Manager Instance-Manager Upgrade-Manager

Security
Kube-Forensics

Monitoring
Active-Monitor

Reliability
Governor Lifecycle-Manager

Cost
Minion-Manager

Addon-Manager

How do I manage critical cluster addons?

What are cluster addons?

Critical Cluster Wide Services

1. **Metrics and Logging**
   We want Metrics and Logging using Prometheus, Fluentd, Eventrouter, etc.

2. **Networking**
   We want DNS and maybe mesh with CoreDNS, Istio, etc.

3. **Autoscaling**
   We want to scale up cluster nodes using Cluster Autoscaler.

4. **Ingress**
   We want to provide Ingress to our cluster using ALB Ingress Controller, Nginx Ingress Controller, etc.

Cluster Addon Management

Declarative. Defined Lifecycle.

1

Single Pane View of Addons

We want to see what cluster services are installed and it's status in one view

2

Managed Addon Lifecycle

Encapsulate Pre-install, Install and Delete workflows as part of each addon

3

Dependency Management

Define relationship dependencies between addons, order of installation, dependent status

4

Deployment Agnostic

Use your favorite deployer, helm, ship, kubectl, etc.

Addon Lifecycle

Prereqs

Create the Addon prerequisites e.g, cloud resources, rbac, etc.

Install/Update

Install or update an Addon configuration.

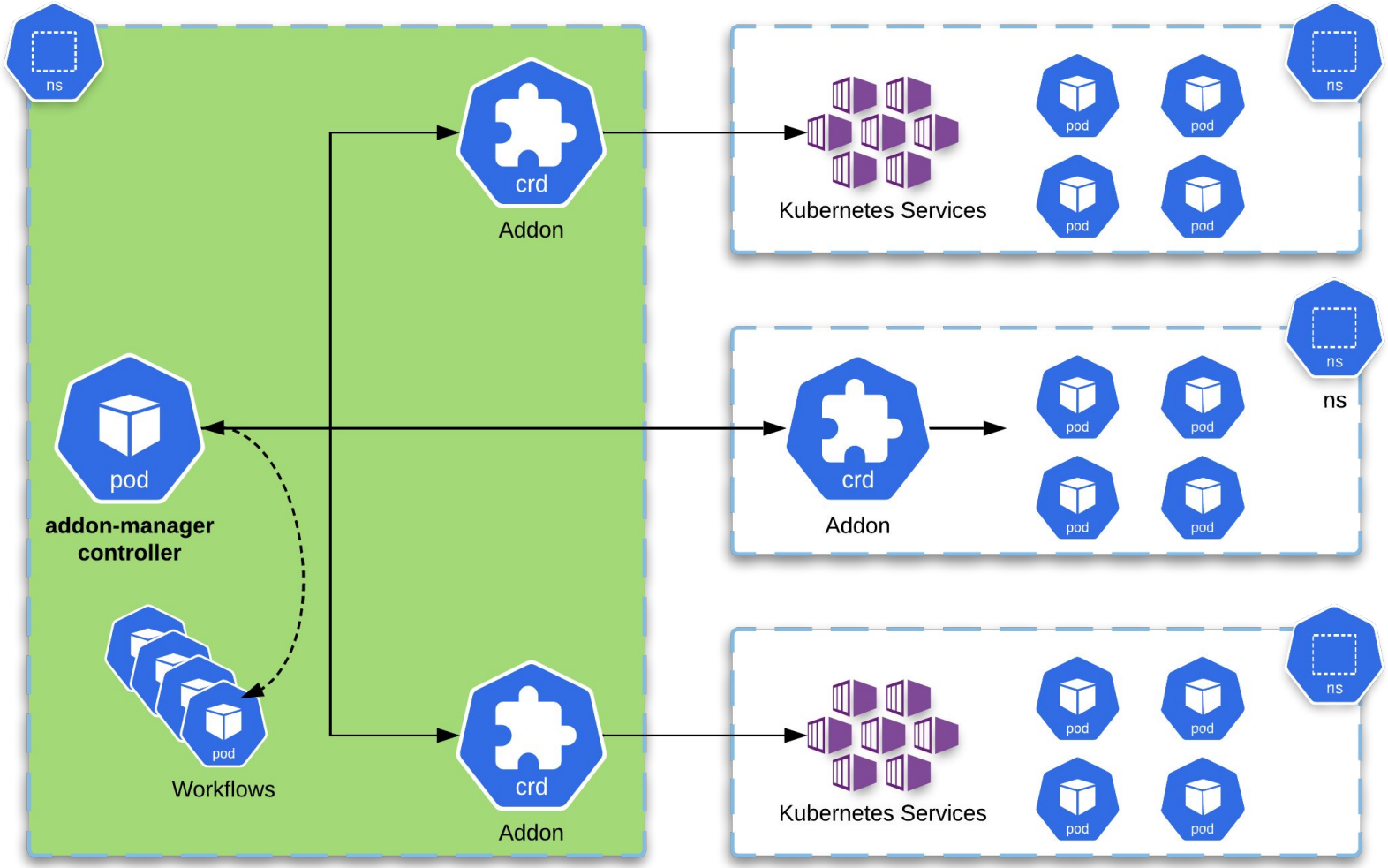
Validate

Validate the components of an Addon.

Delete

Delete the Addon and it's prereqs.

Demo


ns
crd
Addon
Kubernetes Services
pod
pod
pod
pod
ns
pod
addon-manager
controller
crd
Addon
pod
pod
pod
pod
ns
ns
pod
Workflows
crd
Addon
Kubernetes Services
pod
pod
pod
pod
ns

```yaml
apiVersion: addonmgr.keikoproj.io/v1alpha1
kind: Addon
metadata:
 name: cluster-autoscaler
 namespace: addon-manager-system
spec:
 pkgChannel: cluster-autoscaler
 pkgName: cluster-autoscaler-addon
 pkgVersion: v0.1
 pkgType: composite
 pkgDescription: "cluster-autoscaler"
 params:
   namespace: addon-cluster-autoscaler-ns
   context:
     clusterName: "my-example.cluster.k8s.local"
     clusterRegion: us-west-2
 lifecycle:
   prereqs:
     template: |
       apiVersion: argoproj.io/v1alpha1
       kind: Workflow
       spec:
       ...

   install:
     template: |
       apiVersion: argoproj.io/v1alpha1
       kind: Workflow
       spec:
       ...

   delete:
     template: |
       apiVersion: argoproj.io/v1alpha1
       kind: Workflow
       spec:
         ...
```

Addon API

Descriptive package declaration

pkg* describes the unique packaging and version of an addon

params defines the injected params into Argo workflows.

lifecycle are the optional Argo Workflow templates you can define to hook into the lifecycle of an Addon

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
spec:
 entrypoint: entry
 serviceAccountName: addon-manager-workflow-installer-sa
 templates:
   - name: entry
     steps:
       - - name: prereq-resources
           template: submit
           arguments:
             artifacts:
               - name: doc
                 path: /tmp/doc
                 raw:
                   data: |
                     apiVersion: v1
                     kind: Namespace
                     metadata:
                       name: "{{workflow.parameters.namespace}}"
                     ---
                     apiVersion: v1
                     kind: ServiceAccount
                     metadata:
                       labels:
                         k8s-addon: cluster-autoscaler.addons.k8s.io
                         k8s-app: cluster-autoscaler
                       name: cluster-autoscaler
                       namespace: "{{workflow.parameters.namespace}}"
                     ---
                     apiVersion: rbac.authorization.k8s.io/v1beta1
                     kind: ClusterRole
                     metadata:
                       name: cluster-autoscaler
                       labels:
                         k8s-addon: cluster-autoscaler.addons.k8s.io
                         k8s-app: cluster-autoscaler
```

Workflow API

Descriptive workflow step declaration

entrypoint describes the first step of the workflow

templates are named sequence of steps that a workflow is comprised of

steps a list of references to actions that will be executed as part of this workflow

https://github.com/argoproj/argo

Instance-Manager

How do I manage worker nodes?

Multitenant Clusters

The user requirements

1. **Dedicated Nodes**
   No one likes a noisy neighbor.

2. **Different Scales**
   Some service teams needed more nodes others needed less.

3. **Different Node Types**
   Some service teams needed memory optimized, others needed compute, or even GPU.

Worker Node Management

The platform challenges

1

Frequent upgrades

Rotating nodes running production workloads isn't always easy, but tend to happen often.

2

Node group ownership

Kubernetes has no concept of grouping for nodes, similar to an autoscaling group, also node objects are not namespaced.

3

Multiple control planes

Changes to nodes have to be made through the cloud provider account, not directly from the cluster.

The Instance Group Lifecycle

Create

Create the instance group.

Update

Modify the instance group configuration.

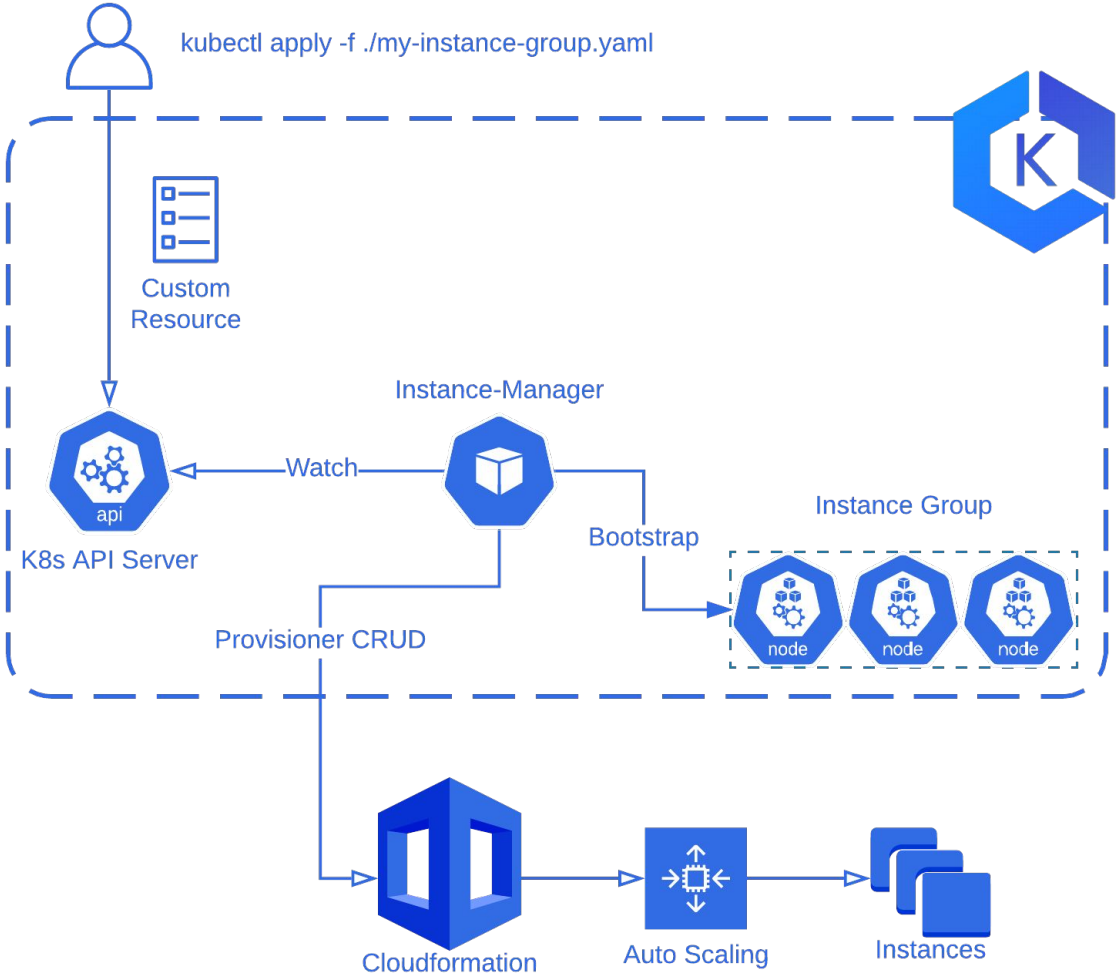
Upgrade

Rotate the instances in the group.

Delete

Delete the instance group.

Demo


kubectl apply -f ./my-instance-group.yaml
Custom
Resource
Instance-Manager
api
K8s API Server
Watch
Bootstrap
Instance Group
node
node
node
Provisioner CRUD
Cloudformation
Auto Scaling
Instances

```yaml
apiVersion: instancemgr.keikoproj.io/v1alpha1
kind: InstanceGroup
metadata:
  name: hello-world
  namespace: instance-manager
spec:
  strategy:
    type: rollingUpdate
    rollingUpdate:
      minInstancesInService: 1
      maxBatchSize: 1
      pauseTime: PT5M
  provisioner: eks-cf
  eks-cf:
    maxSize: 6
    minSize: 3
    configuration:
      bootstrapArguments: '--node-labels=some-label=true'
      clusterName: my-eks-cluster
      subnets:
        - subnet-abb1c79de3EXAMPLE
        - subnet-cdd1c79de3EXAMPLE
        - subnet-eff1c79de3EXAMPLE
      keyPairName: my-key-pair
      image: ami-fe2dc743aEXAMPLE
      instanceType: m3.medium
      volSize: 80
      securityGroups:
        - sg-a04adb633bEXAMPLE
      tags:
        - key: my-key
          value: my-value
```

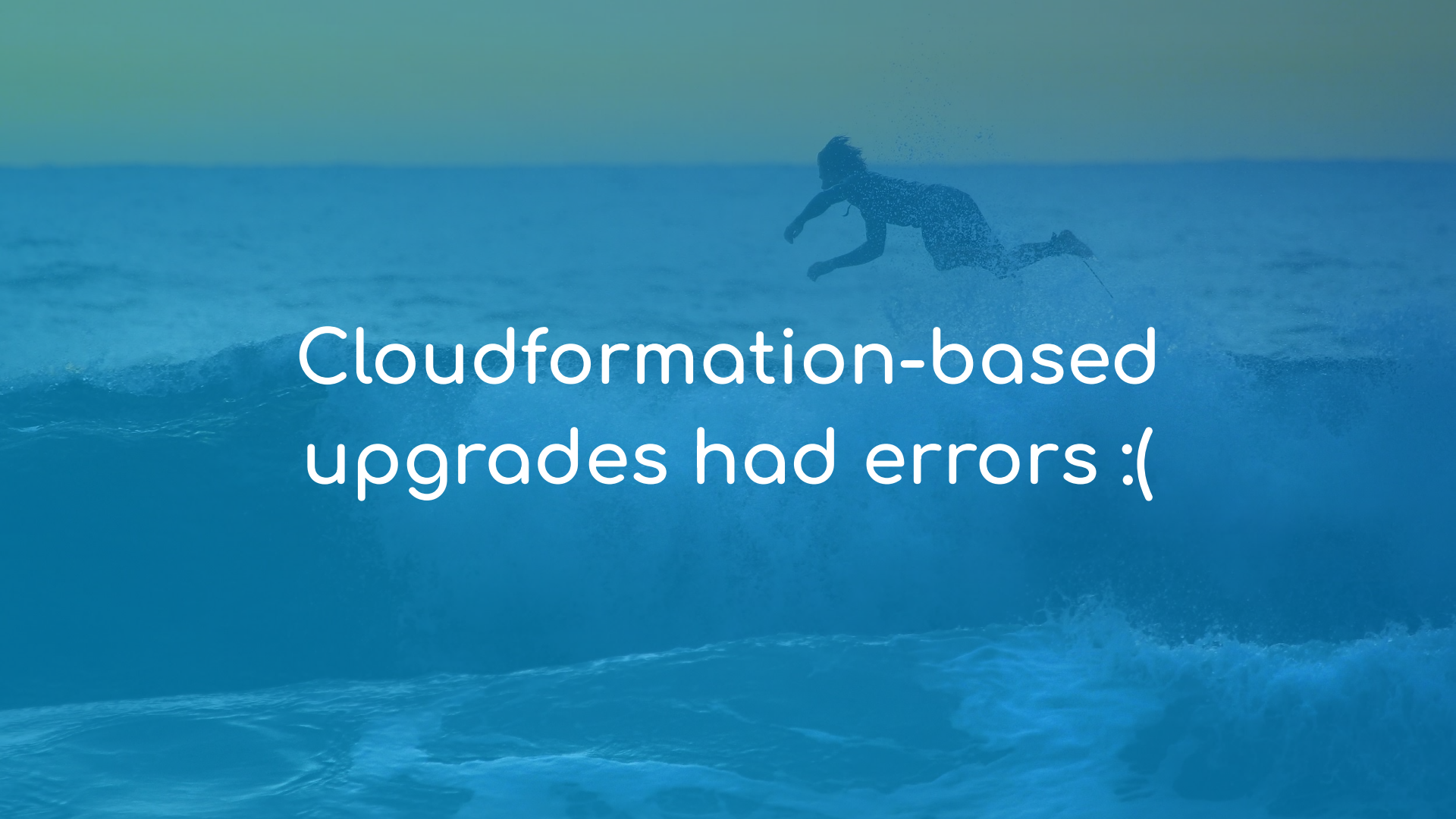
InstanceGroup API

Basic instance group with a Rolling Update strategy

strategy defines the upgrade behavior.

provisioner defines the type of instance group.

eks-cf or the named provisioner defines the configuration of the instance group.


Cloudformation-based
upgrades had errors :(

```yaml
apiVersion: upgrademgr.keikoproj.io/v1alpha1
kind: RollingUpgrade
metadata:
  name: my-rolling-upgrade
  namespace: instance-manager
spec:
  postDrainDelaySeconds: 30
  nodeIntervalSeconds: 300
  region: us-west-2
  asgName: my-auto-scaling-group
```

Upgrade-Manager

What does it do?

Custom Resource for Kubernetes-Aware Rolling Upgrade

Drains an instance in a scaling group

Wait for drain completion

Terminate EC2 Instance

Post-terminate wait

Move to next instance

More Keiko projects!

1. **Lifecycle-Manager**
   Graceful EC2 terminations using AWS lifecycle hooks

2. **Governor**
   Terminates failing nodes and stuck pods

3. **Active-Monitor**
   Deep monitoring of cluster capabilities

4. **Minion-Manager**
   Cost savings by using spot instances

5. **Kube-Forensics**
   Captures a running container state for security analysis

github.com/keikoproj