

Krane

Daniel Turner and Katrina Verey

Production Engineering at Shopify



Why are we here?

Overview

- What is Krane?
- How do you use it?
- How is it structured?
- Lessons learned

What is Krane?

**Krane is an open source
deploy tool for Kubernetes**

We use it *a lot* at Shopify

2000+

invocations per business day

**“This Kubernetes thing
seems pretty cool.”**

– Some Shopifolk in early 2016



kubectl apply -f config/deploy/production

  Deploy of `shopify/production` to revision `dfdae9db` succeeded

Meanwhile, in the cloud...

<code>web-1572547800-d8l4k</code>	<code>0/1</code>	<code>CrashLoopBackOff</code>
<code>web-1572547800-s4hfh</code>	<code>0/1</code>	<code>CrashLoopBackOff</code>
<code>upload-assets-869434</code>	<code>0/1</code>	<code>Error</code>
<code>jobs1-647994f679-zjv4s</code>	<code>0/1</code>	<code>CreateContainerConfigError</code>
<code>jobs2-7658d9bd46-99fpj</code>	<code>0/1</code>	<code>ImagePullBackOff</code>



Wanted: a tool that empowers
developers to deploy confidently
to Kubernetes' eventually
convergent system

History

- Written in Ruby
- Started as a script in Shopify/shipit-engine
- Used at Shopify since early 2017
- Version 1.0 released this month
- 50 contributors and counting... will YOU be next? 😄

```
krane deploy my-ns my-ctx -f deploy/production
```

Our focus:

Developer experience

- Accurate pass/fail result
- Output actionable by non-experts
- Many conveniences, e.g.:
 - Deploy sequencing
 - Task running

~~Live~~ Demos

Demo 1:

Deploy a web app

1. Validate inputs
2. Detect cluster state
3. Deploy the resources
4. Monitor the rollout



Validation

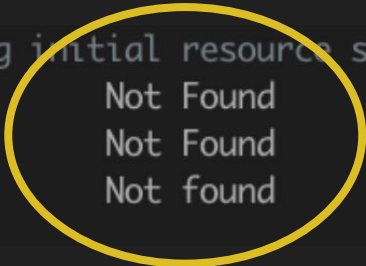
- Reachable cluster
- Existing namespace
- Resources are valid

```
-----Phase 1: Initializing deploy-----  
All required parameters and files are present  
Discovering resources:  
- Ingress/web  
- Service/web  
  Deployment/web
```


Detect cluster state

- Know where you started

```
-----Phase 2: Checking initial resource statuses-----  
Deployment/web          Not Found  
Ingress/web            Not Found  
Service/web           Not found
```



Deploy resources

- Kubectl apply by default

```
-----Phase 3: Deploying all resources-----  
Deploying resources:  
- Deployment/web (progress deadline: 3s)  
- Ingress/web (timeout: 30s)  
- Service/web (timeout: 420s)
```

Monitor the rollout

- Per resource success criteria

Result: SUCCESS

Successfully deployed 3 resources

Successful resources

Deployment/web

Ingress/web

Service/web

1 replica, 1 updatedReplica, 1 availableReplica

Created

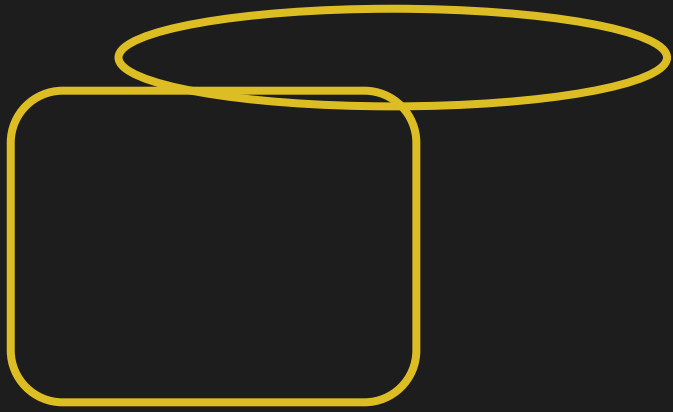
Selects at least 1 pod

Demo 2:

Web app and ConfigMap

1. Validate inputs
2. Detect cluster state
- 3. Deploy priority resources**
4. Deploy the resources
5. Monitor the rollout

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-app-configmap-data
  labels:
    name: test-app-configmap-data
    app: test-app
datas:
  LOG_LEVEL: INFO
```



```
apiVersion: v1
```

```
kind: ConfigMap
```

```
metadata:
```

```
  name: test-app-configmap-data
```

```
  labels:
```

```
    name: test-app-configmap-data
```

```
    app: test-app
```

```
data:
```

```
  LOG_LEVEL: INFO
```



Priority resource deployment

- Pre-deploy resources
 - Referenceable resources: Secret, ConfigMap, ServiceAccount
 - State modifiers: RBAC, NetworkPolicy, ResourceQuota
 - Tasks: Pod

```
-----Phase 3: Predeploying priority resources-----  
Deploying ConfigMap/test-app-configmap-data (timeout: 30s)  
Successfully deployed in 0.3s: ConfigMap/test-app-configmap-data
```

Demo 3:

Web app ~~and ConfigMap~~

1. Validate inputs
2. Detect cluster state
- ~~3. Deploy priority resources~~
4. Deploy the resources **and prune**
5. Monitor the rollout



5 for the price of 1

- **krane deploy**: manages a namespace
- **krane global-deploy**: manages a partition of the global namespace
- **krane restart**: performs a rolling restart of deployment(s)
- **krane run**: deploys and watches a single pod
- **krane render**: renders ERB templates

Demo 4:

Restarting an app

1. Find workloads to restart
2. Trigger the restart
3. Monitor the restart



→ test-app



Ways to run Krane

1. From a laptop
2. Ruby library
3. Continuous delivery tooling



shopify

cumulus-cat production

Shipit v0.29.0

[Refresh statuses & commits](#)

[Commits & Deploys](#)

[Settings](#)

[Timeline](#)

[Tasks](#) ▼

[View on GitHub](#)

[View website](#)

Perform a rolling restart in all contexts where this app is deployed

```
krane restart cumulus-cat-production tier4
```

[Restart application \(all contexts\)](#)



shopify

cumulus-cat production

Refresh statuses & commits

[Commits & Deploys](#) [Settings](#) [Timeline](#) [Tasks](#) ▾

[View on GitHub](#) [View website](#)

Danny Turner executing restart about an hour ago ([view raw output](#))

```
$ krane restart cumulus-cat-production tier4
pid: 851345
[INFO][2019-11-11 05:15:00 +0000]
[INFO][2019-11-11 05:15:00 +0000] -----Phase 1: Initializing restart-----
[INFO][2019-11-11 05:15:03 +0000] Configured to restart all deployments with the `shipit.shopify.io/restart` annotation
[INFO][2019-11-11 05:15:03 +0000]
[INFO][2019-11-11 05:15:03 +0000] -----Phase 2: Triggering restart by touching ENV[RESTARTED_AT]-----
[INFO][2019-11-11 05:15:03 +0000] Triggered `jobs` restart
[INFO][2019-11-11 05:15:03 +0000] Triggered `web` restart
[INFO][2019-11-11 05:15:03 +0000]
[INFO][2019-11-11 05:15:03 +0000] -----Phase 3: Waiting for rollout-----
[INFO][2019-11-11 05:15:29 +0000] Successfully restarted in 25.4s: Deployment/jobs
[INFO][2019-11-11 05:15:29 +0000] Continuing to wait for: Deployment/web
[INFO][2019-11-11 05:15:37 +0000] Successfully restarted in 33.9s: Deployment/web
[INFO][2019-11-11 05:15:37 +0000]
[INFO][2019-11-11 05:15:37 +0000] -----Result: SUCCESS-----
[INFO][2019-11-11 05:15:37 +0000] Successfully restarted 2 resources
[INFO][2019-11-11 05:15:37 +0000]
[INFO][2019-11-11 05:15:37 +0000] Successful resources
[INFO][2019-11-11 05:15:37 +0000] Deployment/jobs          3 replicas, 3 updatedReplicas, 3 availableReplicas
[INFO][2019-11-11 05:15:37 +0000] Deployment/web          3 replicas, 3 updatedReplicas, 3 availableReplicas
```

Completed successfully

Krane internals

Key classes

Krane::DeployTask

#new

#run

Krane::ResourceDeployer

Krane::ResourceWatcher

Krane::KubernetesResource

Key classes:

ResourceWatcher

```
def run(...)
  while remainder.present?
    give_up(...) if global_timeout?
    sleep_until_next_sync(...)

    sync_resources(...)

    report_what_just_happened(...)
    report_what_is_left(...)
  end
  record_statuses_for_summary(...)
end
```

Key concept:

Sync

```
def run(...)
  while remainder.present?
    give_up(...) if global_timeout?
    sleep_until_next_sync(...)

    sync_resources(...)

    report_what_just_happened(...)
    report_what_is_left(...)
  end
  record_statuses_for_summary(...)
end
```

Key classes:

KubernetesResource

```
module Krane
  class KubernetesResource
    def sync(cache)
    end

    def deploy_failed?
    end

    def deploy_succeeded?
    end

    def deploy_timed_out?
    end
  end
end
```

Key classes:

KubernetesResource

```
module Krane
  class ConfigMap < KubernetesResource
    TIMEOUT = 30.seconds

    def deploy_succeeded?
      exists?
    end

    def status
      exists? ? "Available" : "Not Found"
    end

    def deploy_failed?
      false
    end

    def timeout_message
      UNUSUAL_FAILURE_MESSAGE
    end
  end
end
```

```
def sync(cache)
  @instance_data = cache.get_instance(kind, name)
end
```

```
def deploy_failed?
  @instance_data["status"]["phase"] == "Lost"
End
```

```
def deploy_succeeded?
  @instance_data["status"]["phase"] == "Bound"
end
```

```
def deploy_timed_out?
  Time.now.utc - @deploy_started_at > timeout
end
```


How you can help:

- Resource modelling
- Fast-failure detection
- Documentation
- Bug reports
- *Your idea here**



Lessons Learned

Mistakes

- Extensive rendering at deploy time adds risk
- If you claim to manage everything in a namespace, pruning should be blacklist-based
 - Adding new kinds to a whitelist is *painful!*

Hard problems: Timeouts

- Lots of reasons for slow starting pods
 - Large images
 - Long app start-up
 - Cluster issues
- Drains trust



Hard problems:

kubectl apply

last-applied annotation

- Pruning
- 3-way merge

Tips

- K8s != must use Golang 😄
- “Look again later” for resiliency
- Annotations for per-resource settings

Tips

(continued)

- Provide separate debug-level logging
- Have an official supported K8s versions list, and run CI against all of them

Recap

- Krane is a developer-centric deploy tool
- Highly scalable
- Open source and contributors welcome

Questions?

github.com/Shopify/krane

[#krane](https://kubernetes.slack.com)

