KubeCon | CloudNativeCon

North America 2019

KubeCon | CloudNativeCon

North America 2019

Helm 3 Deep Dive

It’s more than just Tiller!

Taylor Thomas & Martin Hickey

Who are we?

Taylor Thomas

- Helm core maintainer
- AKS and Helm at Microsoft
- Social
  - Twitter: @_oftaylor
  - GitHub: @thomastaylor312
  - Kubernetes Slack: @oftaylor

Martin Hickey

- Helm core maintainer
- Developer at IBM
- Social
  - Twitter: @mhickeybot
  - GitHub: @hickeyma
  - Kubernetes Slack: @mhickey

Agenda

- Announcements
- New Functionality
- Architectural Changes
- Migration
- Security Model

Announcements

Helm 3!

Thank you.

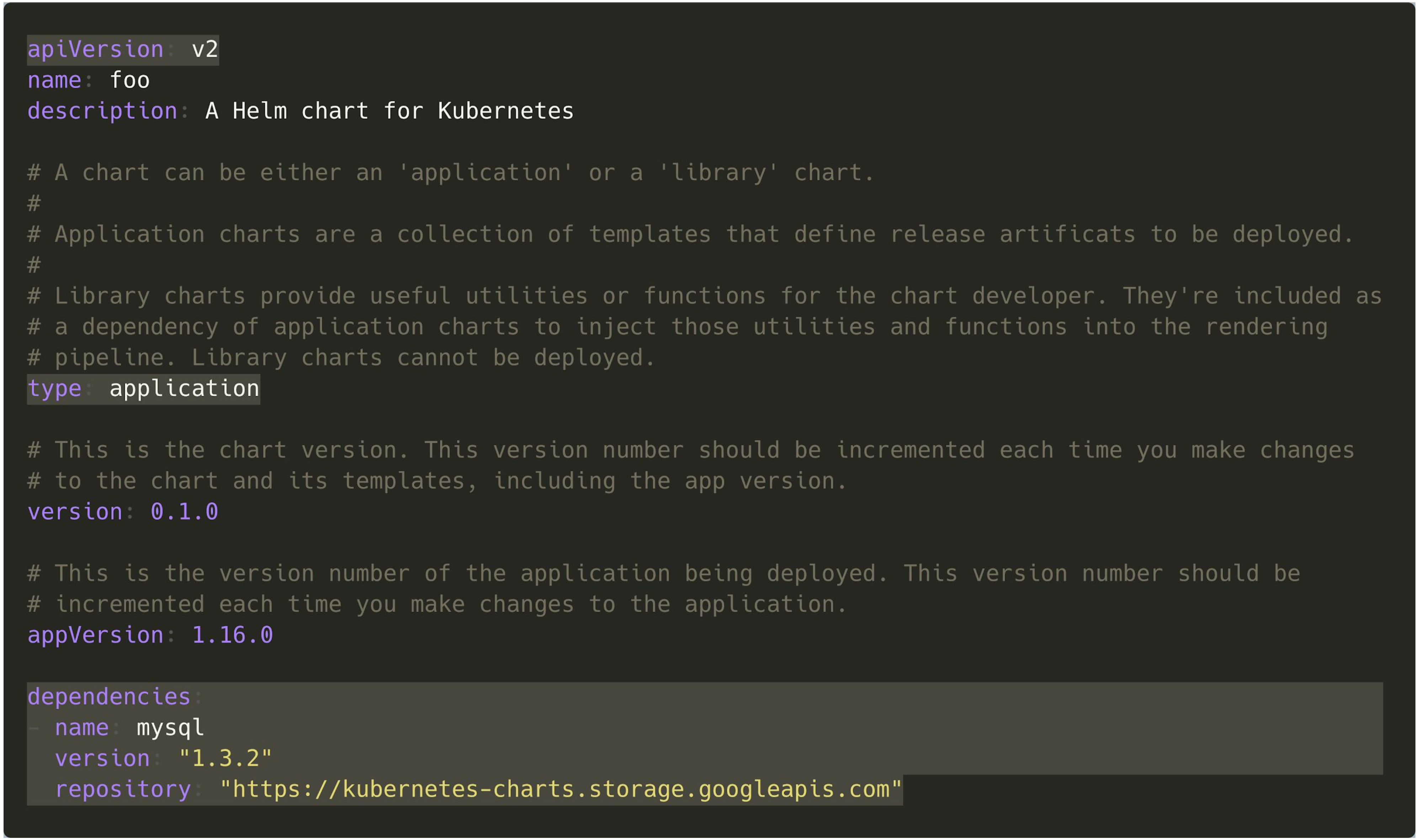
🎺 Security audit passed!
Hoping to graduate next year 🎺

New Functionality

Lets take a look at ....

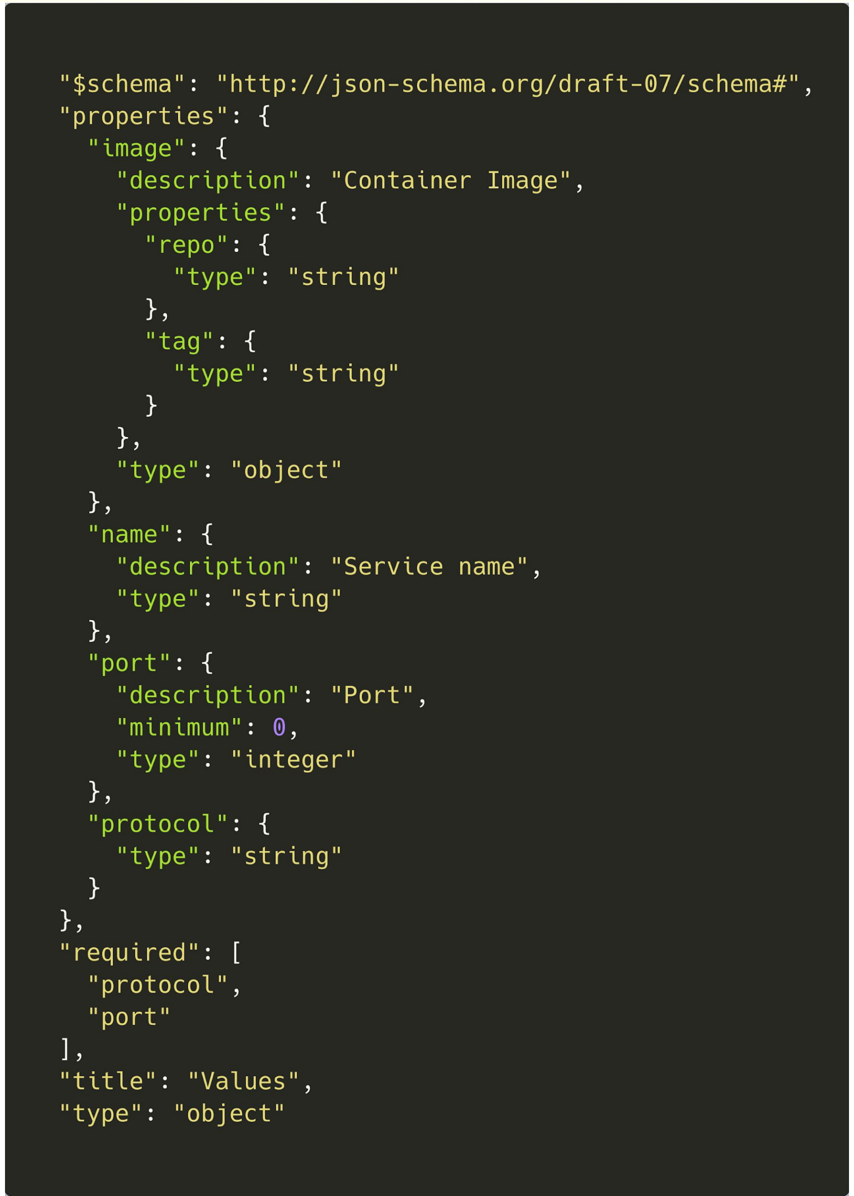
- Chart Changes
- Chart Repository
- Release upgrade strategy
- Test framework
- OCI registry support

Chart updates: *Chart.yaml*



```yaml
apiVersion: v2
name: foo
description: A Helm chart for Kubernetes

# A chart can be either an 'application' or a 'library' chart.
#
# Application charts are a collection of templates that define release artificats to be deployed.
#
# Library charts provide useful utilities or functions for the chart developer. They're included as
# a dependency of application charts to inject those utilities and functions into the rendering
# pipeline. Library charts cannot be deployed.
type: application

# This is the chart version. This version number should be incremented each time you make changes
# to the chart and its templates, including the app version.
version: 0.1.0

# This is the version number of the application being deployed. This version number should be
# incremented each time you make changes to the application.
appVersion: 1.16.0

dependencies:
- name: mysql
  version: "1.3.2"
  repository: "https://kubernetes-charts.storage.googleapis.com"
```

Chart updates: *Value validation*



```
"$schema": "http://json-schema.org/draft-07/schema#",
"properties": {
  "image": {
    "description": "Container Image",
    "properties": {
      "repo": {
        "type": "string"
      },
      "tag": {
        "type": "string"
      }
    },
    "type": "object"
  },
  "name": {
    "description": "Service name",
    "type": "string"
  },
  "port": {
    "description": "Port",
    "minimum": 0,
    "type": "integer"
  },
  "protocol": {
    "type": "string"
  }
},
"required": [
  "protocol",
  "port"
],
"title": "Values",
"type": "object"
```

- Place *values.schema.json* in chart folder
- Validation on commands:
  - *helm install*
  - *helm upgrade*
  - *helm template*
  - *helm lint*

Chart Repository

- Helm Hub – central catalog for chart repos
- Search updated in Helm for Helm Hub
- No chart repos added OOTB anymore

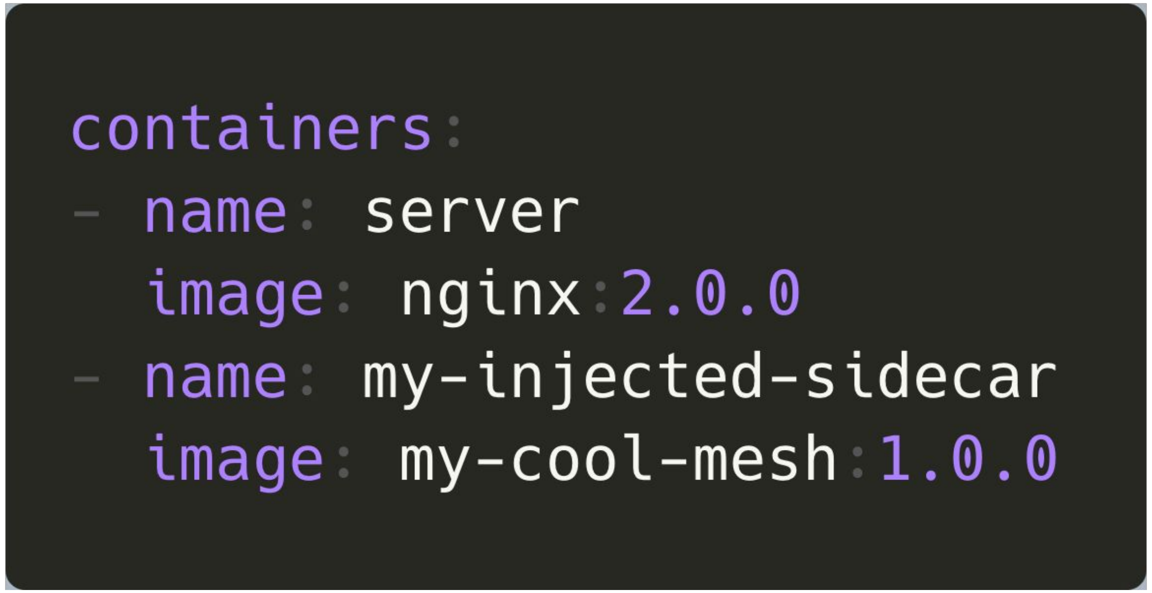
Improved upgrade strategy

- Helm 2 merge/upgrade compares: most recent chart manifest and the proposed chart manifest

- Helm 3 merge/upgrade compares: most recent chart manifest, proposed chart manifest and cluster live state

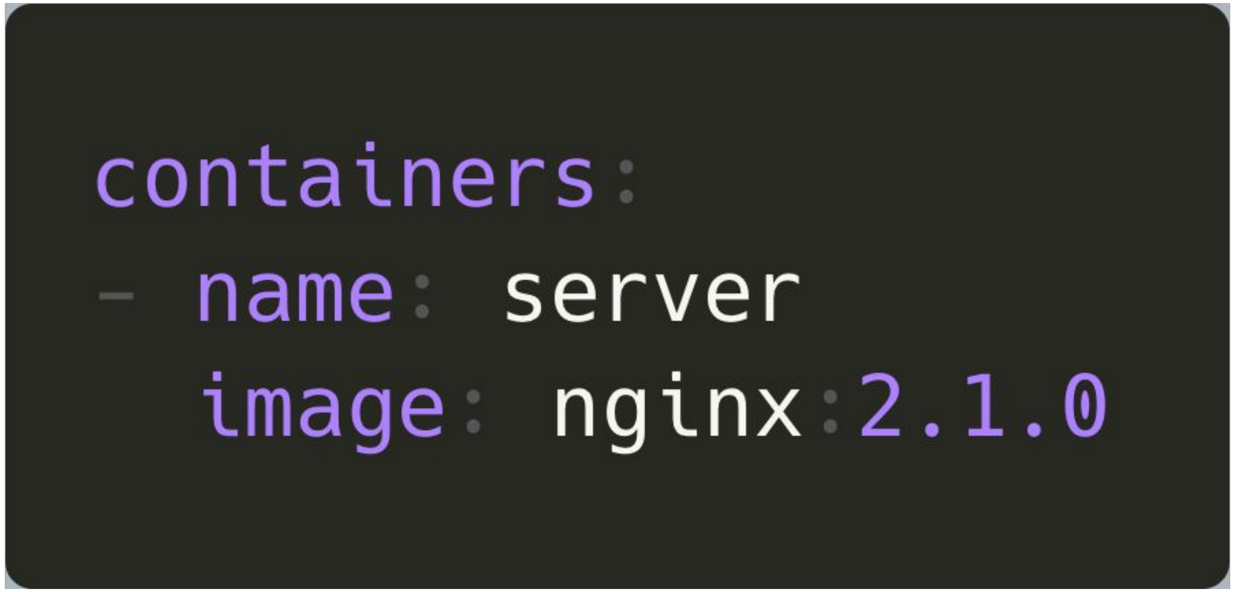
3-way merge: Example

Current State of Cluster



```
containers:
- name: server
  image: nginx:2.0.0
- name: my-injected-sidecar
  image: my-cool-mesh:1.0.0
```

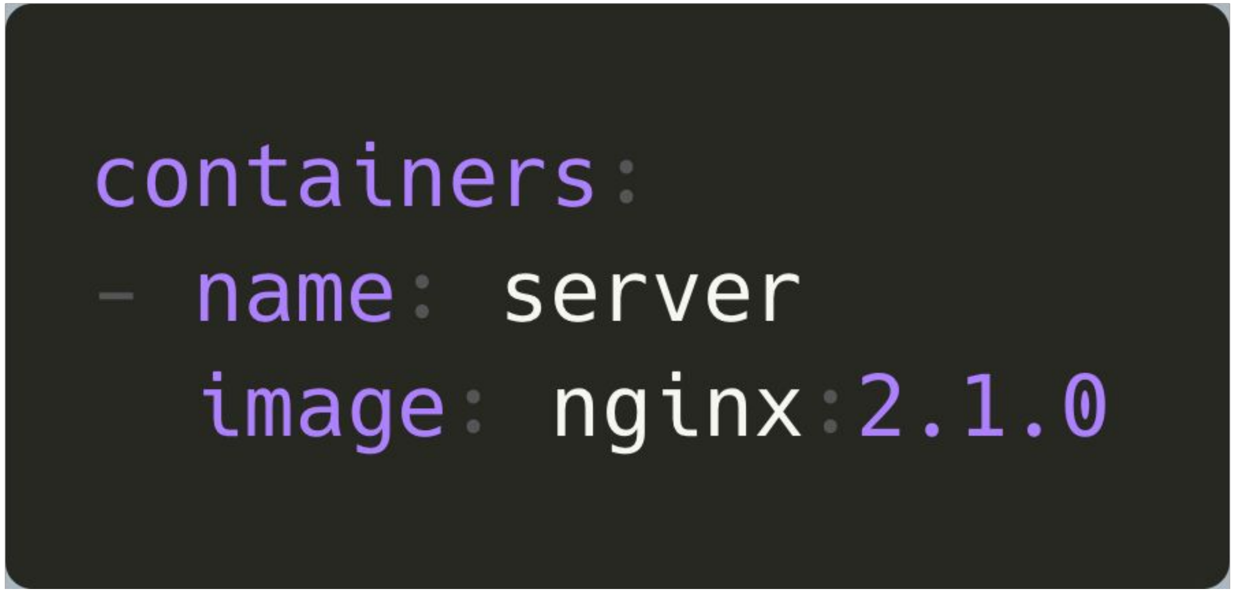
New Manifest



```
containers:
- name: server
  image: nginx:2.1.0
```

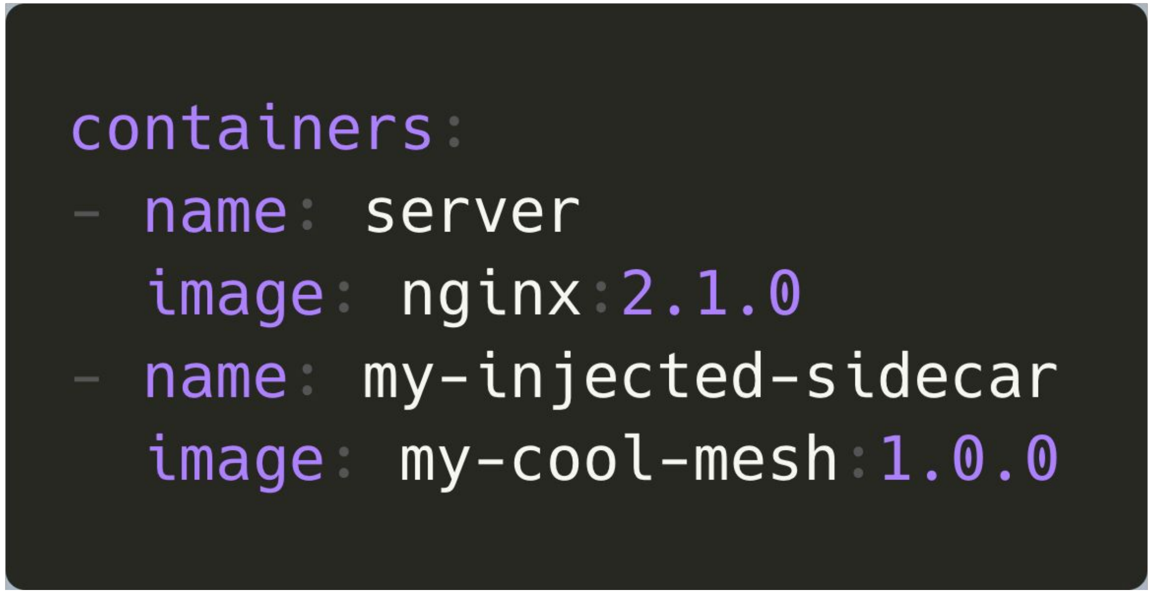
3-way merge: Example

Helm 2



```
containers:
- name: server
  image: nginx:2.1.0
```

Helm 3



```
containers:
- name: server
  image: nginx:2.1.0
- name: my-injected-sidecar
  image: my-cool-mesh:1.0.0
```

Update to test framework

- Tests as Jobs
- `test-failure` hook removed
- `test-success` hook replaced with `test`
- `--logs` flag
- `--cleanup` flag removed

OCI registry support

- Helm 3 enables charts to be stored/shared in OCI-based registries
- Experimental feature: `export HELM_EXPERIMENTAL_OCI=1`
- Check out `helm help chart` and `helm help registry`

Architecture Changes

No more Tiller

- The universe is safe again! 😊
- `helm` now talks to Kubernetes API server directly
- XDG base specification support
  - Cache: `$XDG_CACHE_HOME`, for example, `${HOME}/.cache/helm/`
  - Configuration: `$XDG_CONFIG_HOME`, for example, `${HOME}/.config/helm/`
  - Data: `$XDG_DATA_HOME`, for example `${HOME}/.local/share/helm`
- `helm init` has been removed

Release storage update

- Stored as *Secrets* by default
- Stored in the namespace of the release
- Helm “Release” object has changed
- Release name prefixed with `sh.helm.release.v1`

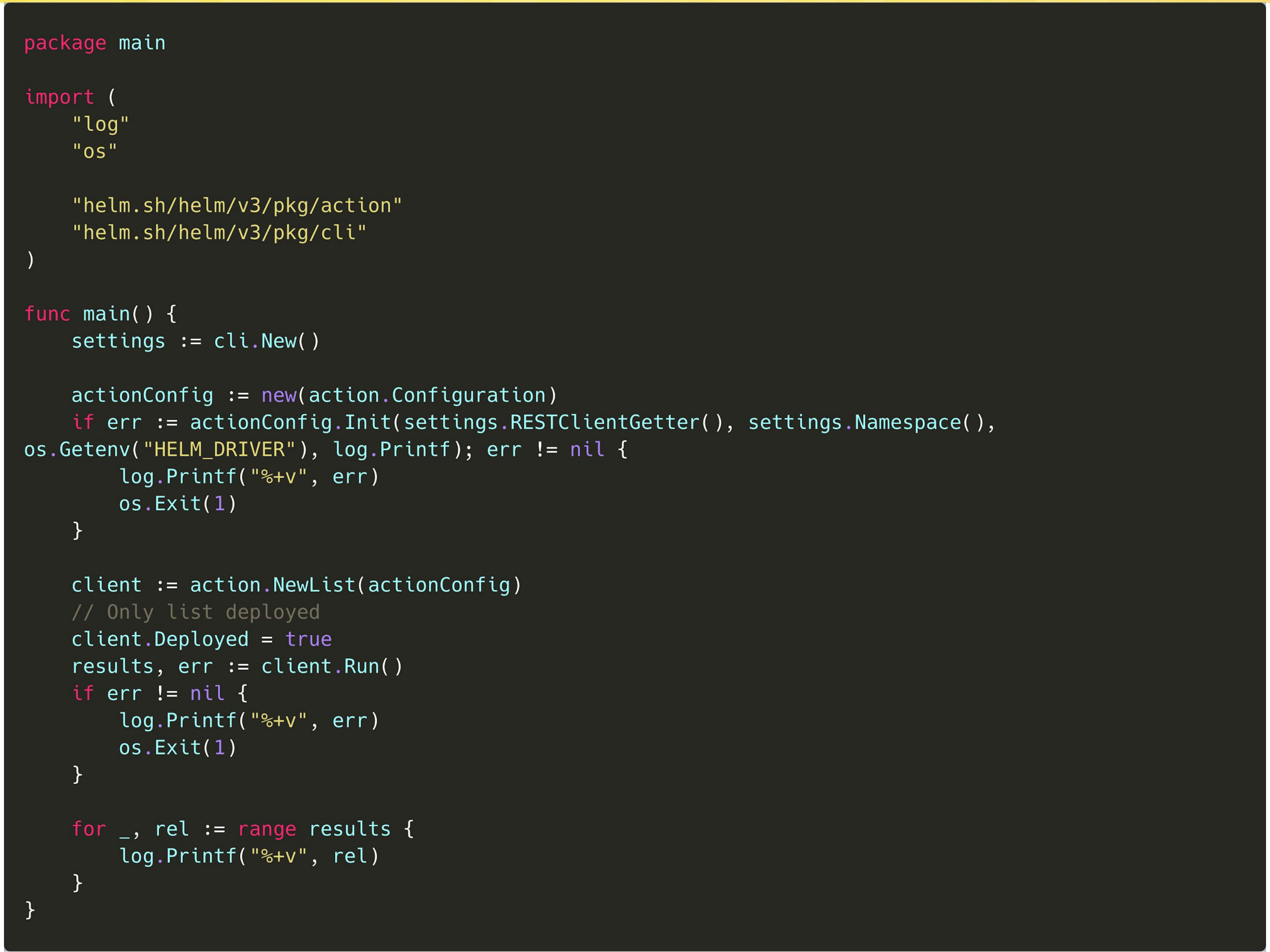
CRD Changes

- CRDs are installed only – no modifications or delete
- CRDs now go in the `crds/` directory to be automatically installed
- Installation can be skipped with `--skip-crds`
- `crd-install` hooks are ignored with a warning in Helm 3

New Go SDK

- Things have been restructured for a better experience
- The CLI consumes the same SDK
- Packages to care about
  - pkg/action
  - pkg/{chart,chartutil}
  - pkg/cli and its subpackages

Go SDK: Example



```go
package main

import (
    "log"
    "os"

    "helm.sh/helm/v3/pkg/action"
    "helm.sh/helm/v3/pkg/cli"
)

func main() {
    settings := cli.New()

    actionConfig := new(action.Configuration)
    if err := actionConfig.Init(settings.RESTClientGetter(), settings.Namespace(),
os.Getenv("HELM_DRIVER"), log.Printf); err != nil {
        log.Printf("%+v", err)
        os.Exit(1)
    }

    client := action.NewList(actionConfig)
    // Only list deployed
    client.Deployed = true
    results, err := client.Run()
    if err != nil {
        log.Printf("%+v", err)
        os.Exit(1)
    }

    for _, rel := range results {
        log.Printf("%+v", rel)
    }
}
```

Migration

Helm 2 v Helm 3

- Helm 2 charts still work in Helm 3*
- The difference between Helm 2 and Helm 3 is mostly architectural and infrastructure changes (e.g. no Tiller)
- How do I migrate my cluster from Helm 2 to Helm 3?

* CRDs and namespace creation are exceptions

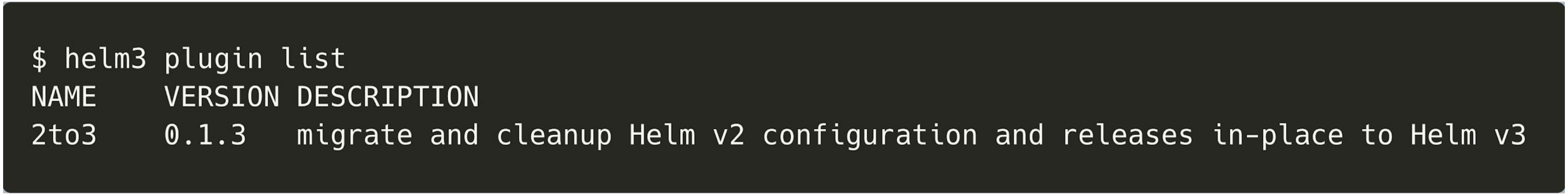
Migration? That doesn’t sound good

Welcome migration!

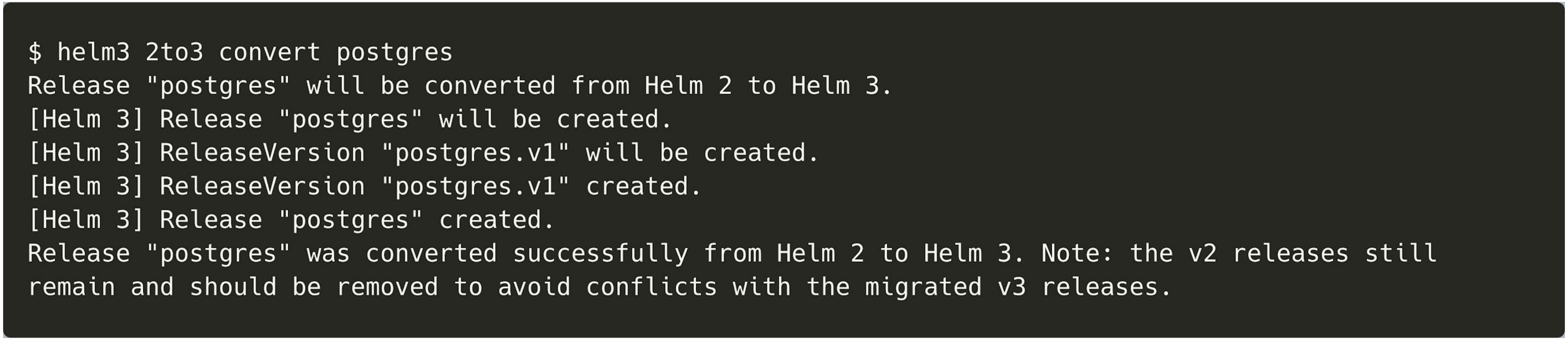
- 2 migration use cases:
  - *Strangler pattern* - Helm 2 and Helm 3 co-exist on the same cluster
  - *In situ* - Migrating Helm 2 to Helm 3

- 2to3 plugin helps with *in situ*: https://github.com/helm/helm-2to3

2to3 plugin example



```
$ helm3 plugin list
NAME    VERSION DESCRIPTION
2to3    0.1.3   migrate and cleanup Helm v2 configuration and releases in-place to Helm v3
```



```
$ helm3 2to3 convert postgres
Release "postgres" will be converted from Helm 2 to Helm 3.
[Helm 3] Release "postgres" will be created.
[Helm 3] ReleaseVersion "postgres.v1" will be created.
[Helm 3] ReleaseVersion "postgres.v1" created.
[Helm 3] Release "postgres" created.
Release "postgres" was converted successfully from Helm 2 to Helm 3. Note: the v2 releases still
remain and should be removed to avoid conflicts with the migrated v3 releases.
```

New security model

Helm 2 security

Kubernetes

- RBAC
- PodSecurityPolicy
- NetworkPolicy
- APIServer cert management
- User management
- Service account management
- Auditing
- Container signing

Helm

- Chart provenance
- Tiller cert management
- Tiller RBAC management

Securing Helm 3: https://sched.co/S8tr

Helm 3 security

Kubernetes

- RBAC
- PodSecurityPolicy
- NetworkPolicy
- APIServer cert management
- User management
- Service account management
- Auditing
- Container signing

Helm

- Chart provenance
- ~~Tiller cert management~~
- ~~Tiller RBAC management~~

Securing Helm 3: https://sched.co/S8tr

In other words...

Security is now on a per-user basis

In other words...

Security is now on a per-user basis

Delegated to K8s user cluster security

In Conclusion

Summary

- Lots of new functionality!
- Several major architecture changes
- Supported migration path
- A new security model that follows Kubernetes
- What’s next?

Helpful Links

- Helm community
- Helm 3 docs
- Helm 2 to Helm 3 migration
- Helm 3 changes

Thank You!