Authoring Helm Charts for DayTrader

Lessons Learned Writing the Helm Charts for DayTrader

Dominic Dumrauf

23.1.2019

Outline

Outline

TL;DR: Show me the Code!

- https://github.com/jpmorganchase/daytrader-helm-charts
- https://github.com/jpmorganchase/daytrader-helm-charts-repo

About the Speaker

Dominic is a Public Cloud Architect with a hands-on background in AWS, Azure, and K8s. He also has an application developer background in Python and likes to recite PEP 20.

K8s is Great!

The Power of K8s

- Allows to deploy, run, and manage containerised apps

K8s is Great!

The Power of K8s

- Allows to deploy, run, and manage containerised apps
- Has abstracted network support built in which makes it easy to build and run micro service based applications in production

K8s is Great!

The Power of K8s

- Allows to deploy, run, and manage containerised apps
- Has abstracted network support built in which makes it easy to build and run micro service based applications in production
- Can be hosted on numerous IaaS platforms (incl. AWS, Azure, Google, Alibaba)

K8s is Great!

The Power of K8s

- Allows to deploy, run, and manage containerised apps
- Has abstracted network support built in which makes it easy to build and run micro service based applications in production
- Can be hosted on numerous IaaS platforms (incl. AWS, Azure, Google, Alibaba)
- Comes with built-in
  - durability
  - persistence
  - resiliency
  - scalability

K8s is Great!

The Power of K8s

- Allows to deploy, run, and manage containerised apps
- Has abstracted network support built in which makes it easy to build and run micro service based applications in production
- Can be hosted on numerous IaaS platforms (incl. AWS, Azure, Google, Alibaba)
- Comes with built-in
  - durability
  - persistence
  - resiliency
  - scalability
- In short: K8s might as well be "The Cloud we always wanted...?!"

K8s is Great!

The Power of K8s

- Allows to deploy, run, and manage containerised apps
- Has abstracted network support built in which makes it easy to build and run micro service based applications in production
- Can be hosted on numerous IaaS platforms (incl. AWS, Azure, Google, Alibaba)
- Comes with built-in
  - durability
  - persistence
  - resiliency
  - scalability
- In short: K8s might as well be "The Cloud we always wanted...?!"
- For a bit of Cargo Cult: Google is actively developing/using it...

K8s Can Be Hard

Potential Hiccups

- Everything is a RESTful API call
- Everything is JSON or YAML

K8s Can Be Hard

Potential Hiccups

- Everything is a RESTful API call
- Everything is JSON or YAML
- "Out of the box", only `kubectl` is included for K8s cluster interaction

K8s Can Be Hard

Potential Hiccups

- Everything is a RESTful API call
- Everything is JSON or YAML
- "Out of the box", only `kubectl` is included for K8s cluster interaction
- Debugging can be tricky due to the distributed and eventual consistent nature

K8s Can Be Hard

Potential Hiccups

- Everything is a RESTful API call
- Everything is JSON or YAML
- "Out of the box", only `kubectl` is included for K8s cluster interaction
- Debugging can be tricky due to the distributed and eventual consistent nature
- New Features are constantly being added

K8s Can Be Hard

Potential Hiccups

- Everything is a RESTful API call
- Everything is JSON or YAML
- "Out of the box", only `kubectl` is included for K8s cluster interaction
- Debugging can be tricky due to the distributed and eventual consistent nature
- New Features are constantly being added
- Securing K8s clusters can be tricky

K8s Can Be Hard

Potential Hiccups

- Everything is a RESTful API call
- Everything is JSON or YAML
- "Out of the box", only `kubectl` is included for K8s cluster interaction
- Debugging can be tricky due to the distributed and eventual consistent nature
- New Features are constantly being added
- Securing K8s clusters can be tricky
- It truly embraces the cloud where "everything fails, all the time" (Werner Vogels, Amazon CTO)

K8s Can Be Hard

Potential Hiccups

- Everything is a RESTful API call
- Everything is JSON or YAML
- "Out of the box", only `kubectl` is included for K8s cluster interaction
- Debugging can be tricky due to the distributed and eventual consistent nature
- New Features are constantly being added
- Securing K8s clusters can be tricky
- It truly embraces the cloud where "everything fails, all the time" (Werner Vogels, Amazon CTO)
- In short: K8s can have a mind of its own when not used carefully

Working with K8s

This sounds great!

Let's release a containerised app on a K8s cluster!

Typical Workflow for Releasing Containerised Apps on K8s

1. Take containerised app

Typical Workflow for Releasing Containerised Apps on K8s

1. Take containerised app
2. Start K8s cluster

Typical Workflow for Releasing Containerised Apps on K8s

1. Take containerised app
2. Start K8s cluster
3. Open YAML editor to prepare release file `how-hard-can-it.be.yaml`

Typical Workflow for Releasing Containerised Apps on K8s

1. Take containerised app
2. Start K8s cluster
3. Open YAML editor to prepare release file `how-hard-can-it.be.yaml`
4. Switch to Terminal to execute `kubectl apply -f how-hard-can-it.be.yaml`

Typical Workflow for Releasing Containerised Apps on K8s

1. Take containerised app
2. Start K8s cluster
3. Open YAML editor to prepare release file `how-hard-can-it.be.yaml`
4. Switch to Terminal to execute `kubectl apply -f how-hard-can-it.be.yaml`
5. Google cryptic error message and eventually find the error message on StackOverflow

Typical Workflow for Releasing Containerised Apps on K8s

1. Take containerised app
2. Start K8s cluster
3. Open YAML editor to prepare release file `how-hard-can-it.be.yaml`
4. Switch to Terminal to execute `kubectl apply -f how-hard-can-it.be.yaml`
5. Google cryptic error message and eventually find the error message on StackOverflow
6. Copy-paste the answer and also fix any whitespace typos

Typical Workflow for Releasing Containerised Apps on K8s

1. Take containerised app
2. Start K8s cluster
3. Open YAML editor to prepare release file `how-hard-can-it.be.yaml`
4. Switch to Terminal to execute `kubectl apply -f how-hard-can-it.be.yaml`
5. Google cryptic error message and eventually find the error message on StackOverflow
6. Copy-paste the answer and also fix any whitespace typos
7. Goto 4 until release works as expected

Typical Workflow for Releasing Containerised Apps on K8s

1. Take containerised app
2. Start K8s cluster
3. Open YAML editor to prepare release file `how-hard-can-it.be.yaml`
4. Switch to Terminal to execute `kubectl apply -f how-hard-can-it.be.yaml`
5. Google cryptic error message and eventually find the error message on StackOverflow
6. Copy-paste the answer and also fix any whitespace typos
7. Goto 4 until release works as expected
8. Enjoy feeling of accomplishment

Typical Workflow for Releasing Containerised Apps on K8s

1. Take containerised app
2. Start K8s cluster
3. Open YAML editor to prepare release file `how-hard-can-it.be.yaml`
4. Switch to Terminal to execute `kubectl apply -f how-hard-can-it.be.yaml`
5. Google cryptic error message and eventually find the error message on StackOverflow
6. Copy-paste the answer and also fix any whitespace typos
7. Goto 4 until release works as expected
8. Enjoy feeling of accomplishment
9. Repeat for all other environments

So, Where's the Problem?!

YAML

- Whitespace really(!) matters in YAML
- One blank can change the entire meaning of the block
- The error can mask itself quite well
- YAML can be very brittle as a result

So, Where's the Problem?!

YAML

- Whitespace really(!) matters in YAML
- One blank can change the entire meaning of the block
- The error can mask itself quite well
- YAML can be very brittle as a result

The Manual Maintenance Problem

- YAML files are most likely environment specific
- YAML files for different environments are very similar
- Separating config from YAML file needed

Enter Helm

What's Helm?

"The package manager for Kubernetes" (quote from https://helm.sh/)

What to Use Helm For

What's Helm Really Good At?

- Rendering YAML templates, provided some input
- Versioning YAML templates and default inputs

What to Use Helm For

What's Helm Really Good At?

- Rendering YAML templates, provided some input
- Versioning YAML templates and default inputs
- Dependency management
- Bundeling versioned YAML templates and default inputs into *Charts*
- Providing a standard way of adding and searching *Helm chart repositories*

What to Use Helm For

What's Helm Really Good At?

- Rendering YAML templates, provided some input
- Versioning YAML templates and default inputs
- Dependency management
- Bundeling versioned YAML templates and default inputs into *Charts*
- Providing a standard way of adding and searching *Helm chart repositories*
- Keeping an audit trail of what has been installed in the K8s cluster
- Releasing, upgrading, and rolling back releases

What to Use Helm For

What's Helm Really Good At?

- Rendering YAML templates, provided some input
- Versioning YAML templates and default inputs
- Dependency management
- Bundeling versioned YAML templates and default inputs into *Charts*
- Providing a standard way of adding and searching *Helm chart repositories*
- Keeping an audit trail of what has been installed in the K8s cluster
- Releasing, upgrading, and rolling back releases
- Being backed by some of the biggest names in the industry (people and code)

What Not to Use Helm For

What's Helm Not So Good At?

- A *silver bullet* approach to everything
- Solving all problems that arise when working with K8s clusters

What Not to Use Helm For

What's Helm Not So Good At?

- A *silver bullet* approach to everything
- Solving all problems that arise when working with K8s clusters
- Debugging K8s templates
- Debugging YAML templates
- Solving YAML brittleness

What Not to Use Helm For

What's Helm Not So Good At?

- A *silver bullet* approach to everything
- Solving all problems that arise when working with K8s clusters
- Debugging K8s templates
- Debugging YAML templates
- Solving YAML brittleness
- Out-of-the-box security

Helm and Security

Wait a minute...

The internet if full of articles telling me how insecure Helm is?!

Helm and Security - It Starts With You

Yes, but...

We also don't remove `sudo` from all Linux boxes as it allows escalation to *root*. Rather, we *manage who has access to what when and how*.

Helm and Security - It Starts With You

Yes, but...

We also don't remove `sudo` from all Linux boxes as it allows escalation to *root*. Rather, we *manage who has access to what when and how*.

So,...

Securing Helm is Your Responsibility!

Helm and Security - It Takes Effort!

Options for Securing Helm

- It depends on the use case!
- Helm can be installed with different privileges (RBAC role/binding)
- There's a tiller-less install
- More to come in Helm 3...

Helm and Security - It Takes Effort!

Options for Securing Helm

- It depends on the use case!
- Helm can be installed with different privileges (RBAC role/binding)
- There's a tiller-less install
- More to come in Helm 3...

In any case...

- Don't leave it at the default settings unless *you know what you are doing and are comfortable with it*!
- Read the Helm documentation on how to secure your Helm/tiller installation

Installing Helm

macOS

```
$ brew install helm
```

Installing Helm

macOS

```
$ brew install helm
```

Other OS

See OS package manager or download from https://github.com/helm/helm/releases

Installing Helm

macOS

```
$ brew install helm
```

Other OS

See OS package manager or download from https://github.com/helm/helm/releases

What's in the Box?!

- `helm` CLI
- Built-in access to `stable` Helm Chart repository
- Access to hundreds of curated Helm charts

Install a Helm Chart

Installing a Local Helm Chart

```
$ helm install /path/to/local/dir --name my-app
```

Install a Helm Chart

Installing a Local Helm Chart

```
$ helm install /path/to/local/dir --name my-app
```

Installing a Helm Chart

```
$ helm install stable/nginx-ingress --name nginx-ingress
```

Install a Helm Chart

Installing a Local Helm Chart

```
$ helm install /path/to/local/dir --name my-app
```

Installing a Helm Chart

```
$ helm install stable/nginx-ingress --name nginx-ingress
```

Installing a Helm Chart Using Custom Values

```
$ helm install stable/nginx-ingress --name nginx-ingress -f
my-values.yaml
```

Upgrading and Deleting a Helm Chart

Uprading a Helm Chart Using Custom Values

```
$ helm upgrade nginx-ingress -f my-values.yaml
```

Upgrading and Deleting a Helm Chart

Uprading a Helm Chart Using Custom Values

```
$ helm upgrade nginx-ingress -f my-values.yaml
```

Deleting a Release

```
$ helm delete nginx-ingress
```

Upgrading and Deleting a Helm Chart

Uprading a Helm Chart Using Custom Values

```
$ helm upgrade nginx-ingress -f my-values.yaml
```

Deleting a Release

```
$ helm delete nginx-ingress
```

Deleting a Release and Purging Release History

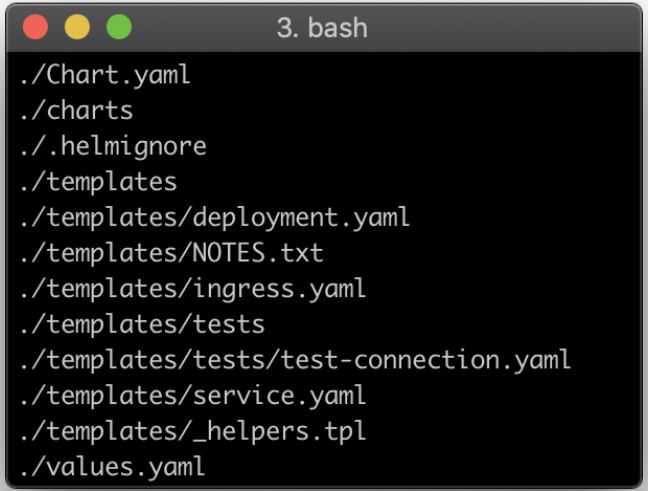
```
$ helm delete nginx-ingress --purge
```

Authoring Helm Charts for DayTrader

Starting a Helm Chart

```
$ helm create daytrader-web
```

What's in the Box?!



Authoring a Helm Chart for DayTrader

Steps

1. Create new Helm Chart using `helm create daytrader`
2. Add Helm charts for individual micro services to `requirements.yaml`

Updating Dependencies and Packaging Charts

Updating Dependencies for a Helm Chart

```
$ helm dependencies update
```

Updating Dependencies and Packaging Charts

Updating Dependencies for a Helm Chart

```
$ helm dependencies update
```

Packaging a Helm Chart

```
$ helm package <directory>
```

Updating Dependencies and Packaging Charts

Updating Dependencies for a Helm Chart

```
$ helm dependencies update
```

Packaging a Helm Chart

```
$ helm package <directory>
```

A Bash Script

The Bash script package-charts.sh updates the dependencies and packages all Helm charts in a given directory.

Creating a Helm Chart Repository

A Home for Helm Charts

Numerous options:

- AWS S3 bucket
- Google Cloud Storage bucket
- GitHub repository

Creating a Helm Chart Repository

A Home for Helm Charts

Numerous options:

- AWS S3 bucket
- Google Cloud Storage bucket
- GitHub repository

A Home on GitHub

Package charts and then execute the following commands:

```
$ git add .
$ git commit -m "Adding packaged Helm charts for daytrader v1.0.17"
$ git push
```

Using the Custom Repository via `helm` CLI

Adding the Repository to `helm`

```
$ helm repo add jpmc 'https://raw.githubusercontent.com/
jpmorganchase/daytrader-helm-charts-repo/master/'
$ helm repo update
```

Using the Custom Repository via `helm` CLI

Adding the Repository to `helm`

```
$ helm repo add jpmc 'https://raw.githubusercontent.com/
jpmorganchase/daytrader-helm-charts-repo/master/'
$ helm repo update
```

Searching the Repository via `helm`

```
$ helm search jpmc
```

Installing DayTrader via `helm`

```
$ helm install jpmc/daytrader -f my-values.yaml --name daytrader
```

That's It!

Happy Helming!