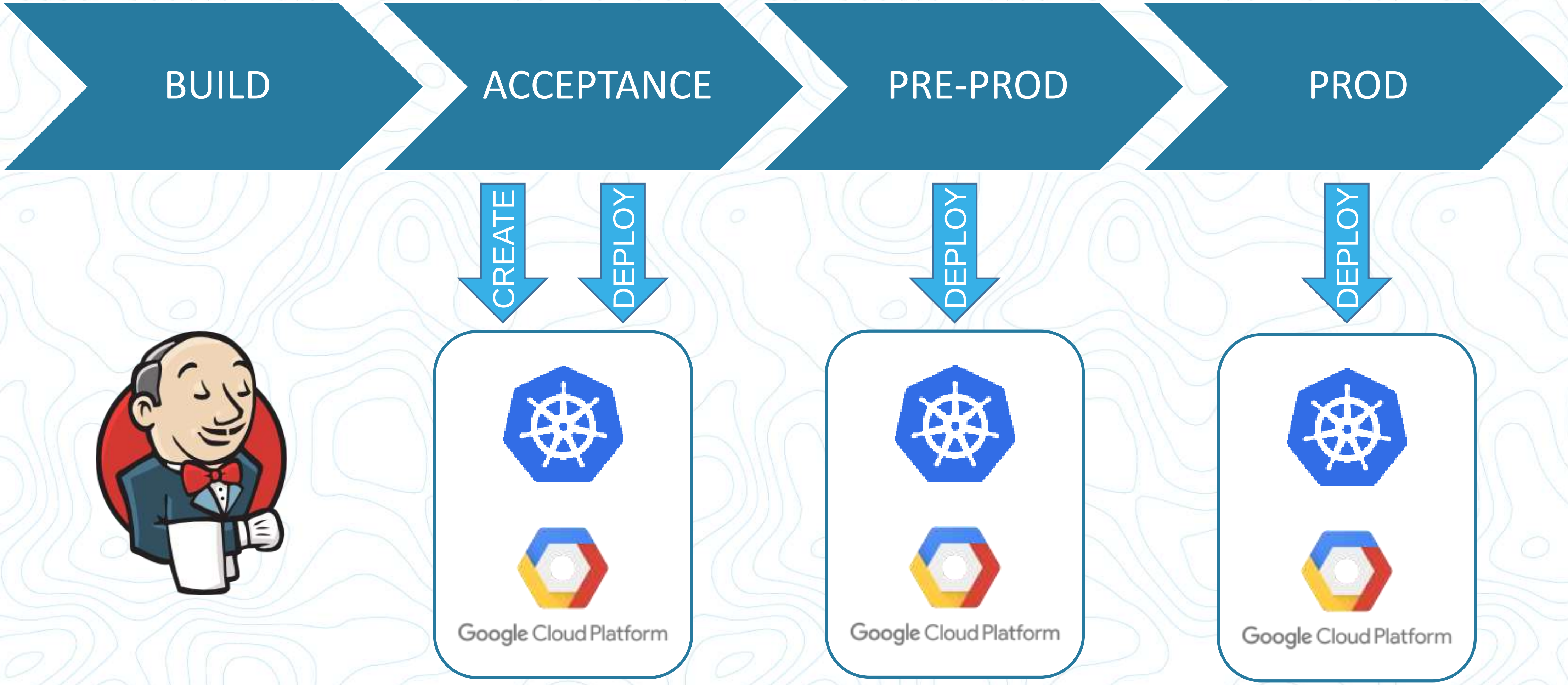
Steering the course with HELM

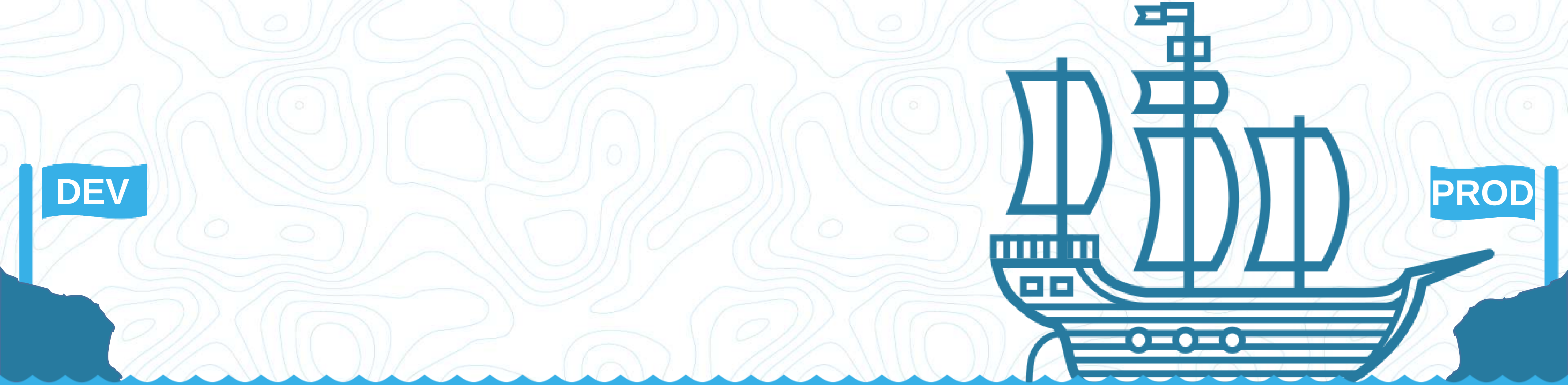
Dirk Jablonski
ePages GmbH

Motivation

How we use Kubernetes

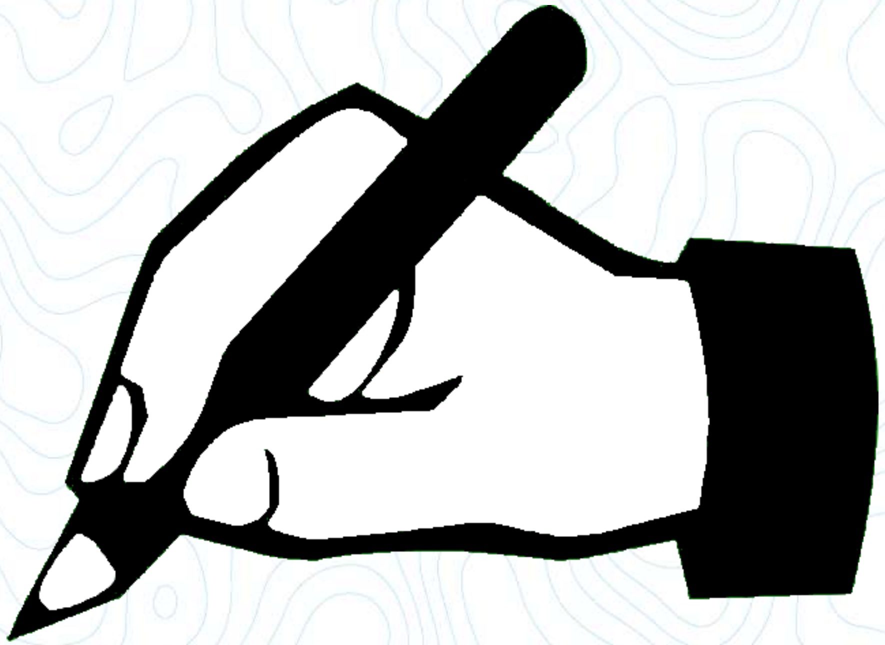


How we want out deployments to be…



...and how it often ended up

What Kinds of Icebergs we hit



handwritten manifests are error-prone

What Kinds of Icebergs we hit

overriding values per environment is quirky

What Kinds of Icebergs we hit

manifests are mostly duplicated code

HELM to the rescue!

What is Helm?

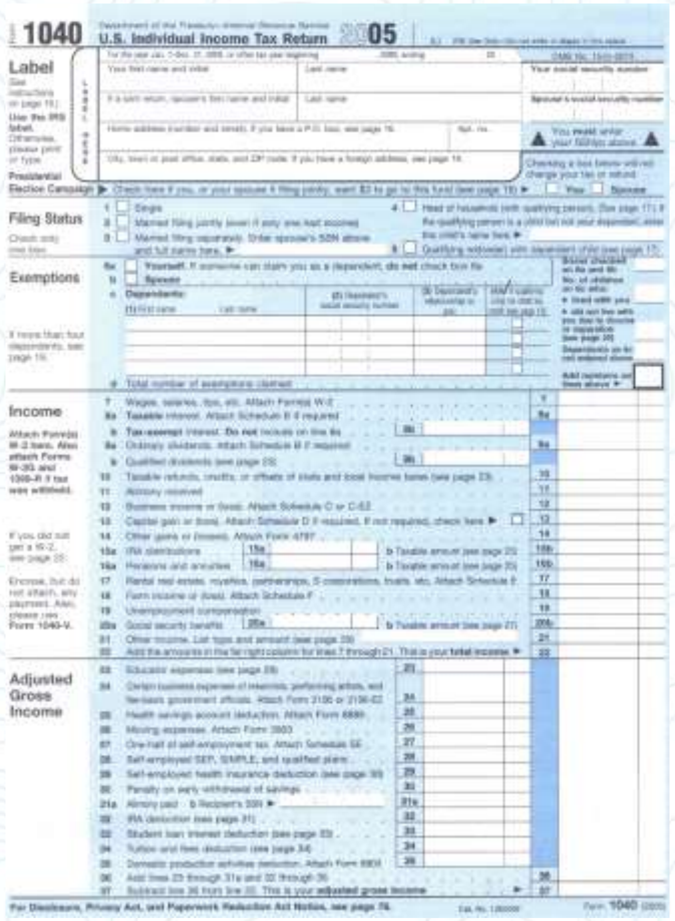
“Kubernetes Package Manager”

What does Helm do?

- bundles related manifests into ***charts***, e.g.
  - deployment.yaml
  - service.yaml
  - ingress.yaml
  - etc.
- when installing a chart, Helm creates a ***release***

What does Helm do?

- provides solid templating
- utilizes Go templates & Sprig template library



1040 U.S. Individual Income Tax Return 2005

Label

Filing Status

Exemptions

Income

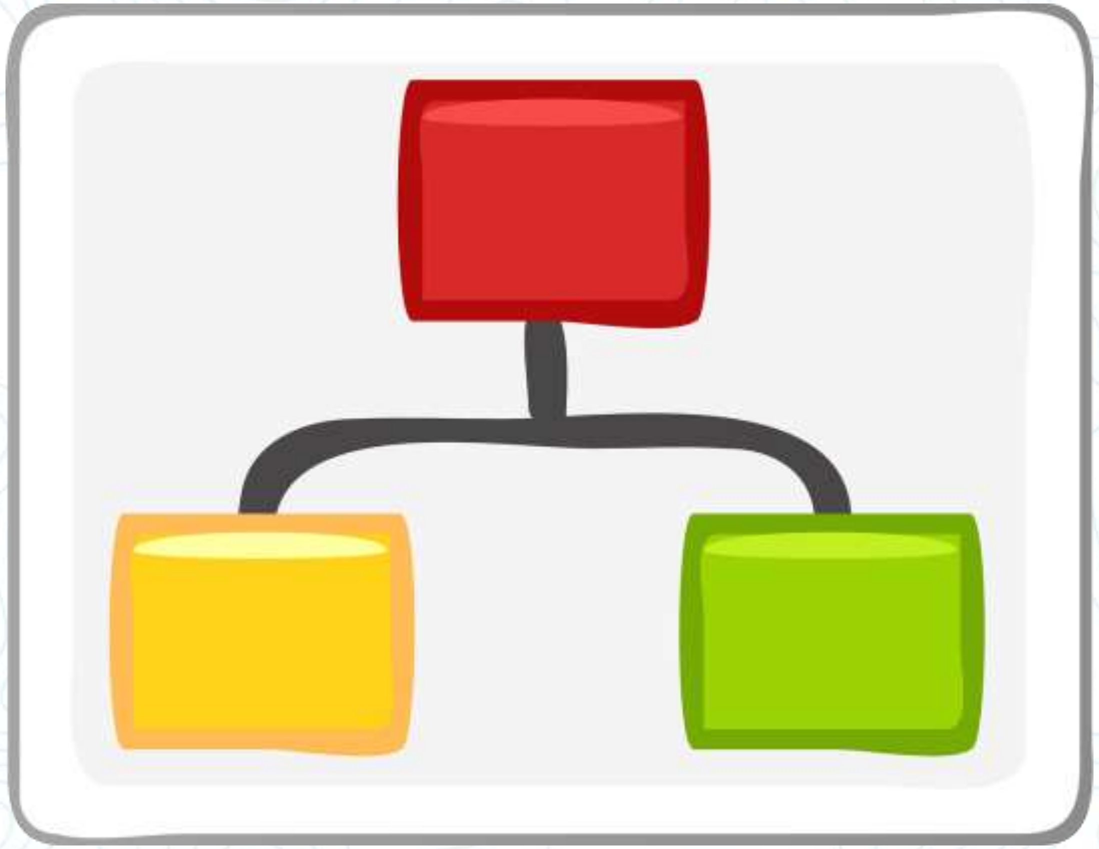
Adjusted Gross Income

What does Helm do?

- allows to override values easily
- overrides can originate from
  - child charts
  - additional values files
  - command-line values
- override order is clearly defined

What does Helm do?

- enables reuse & composition via dependencies



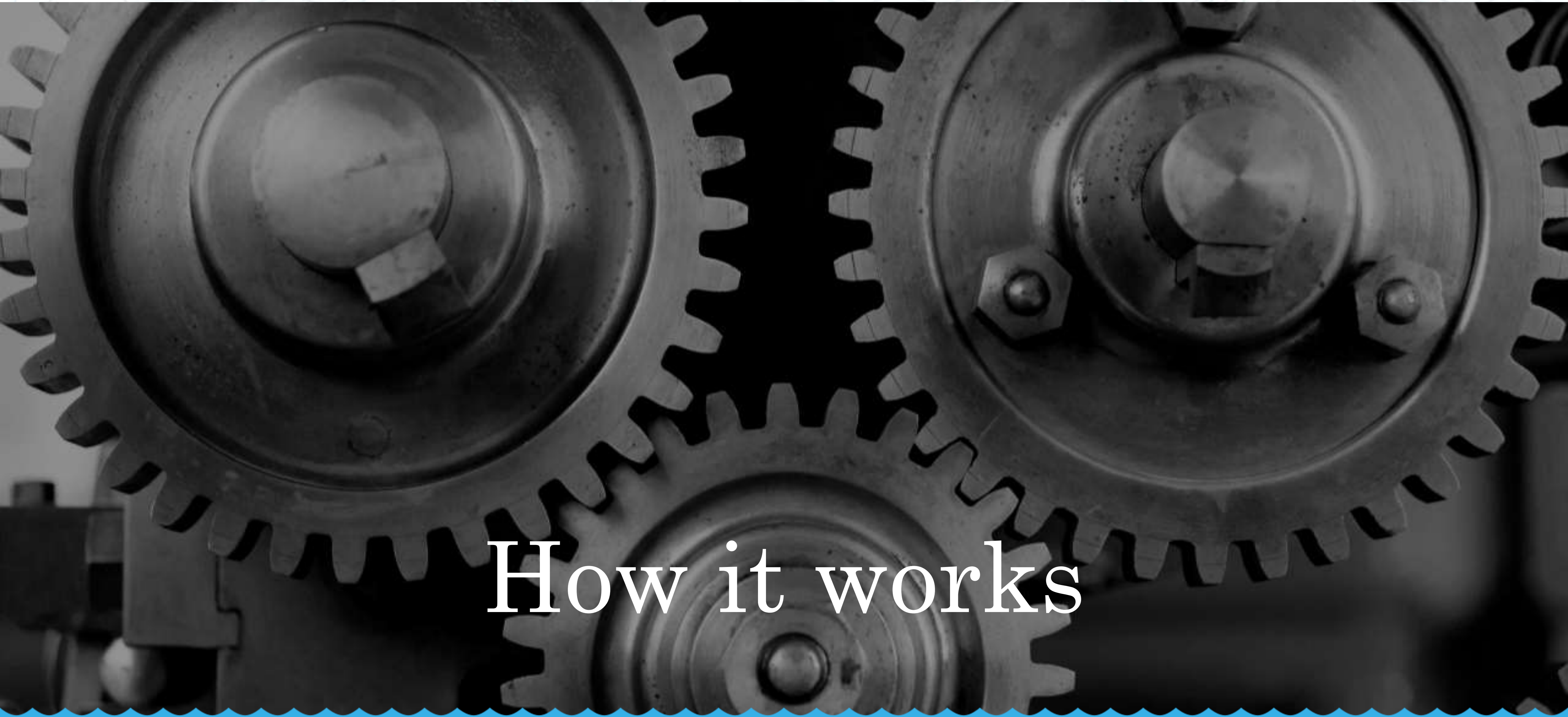
Use Cases?

Managing your own charts

- create charts for your own applications

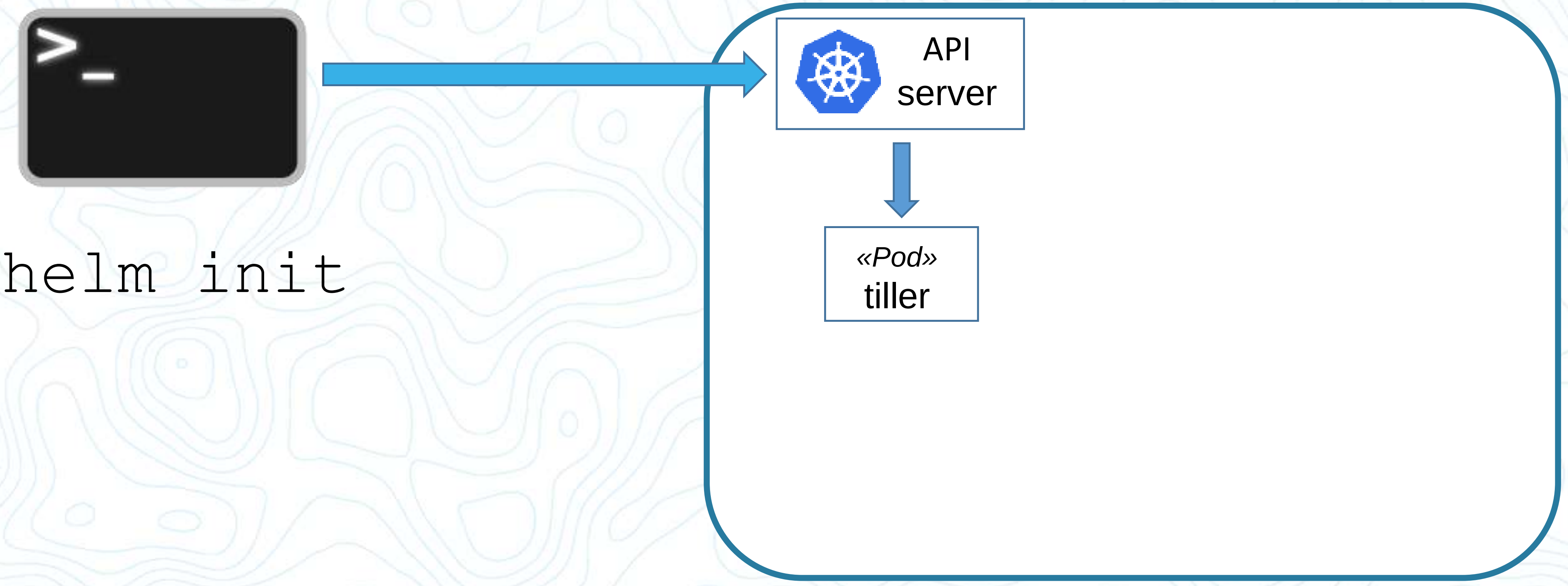
Utilizing community charts

- easily install standard applications, e.g. Prometheus



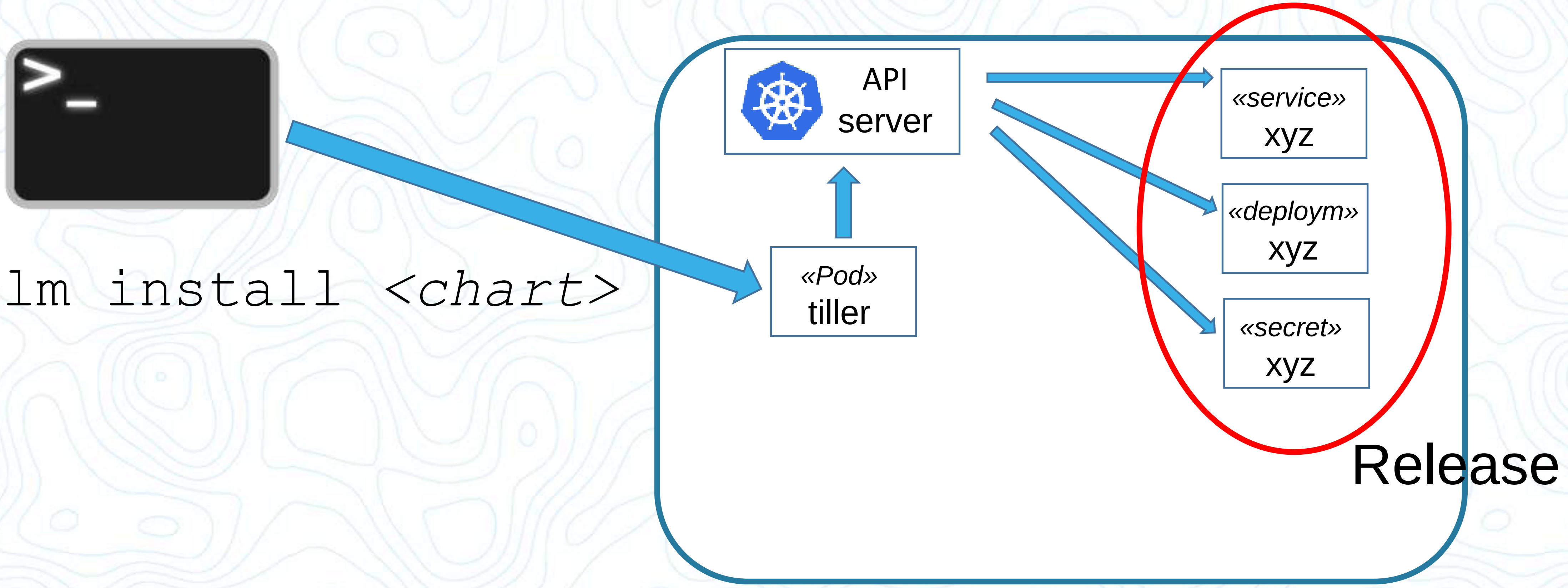
How it works

Initializing the Cluster



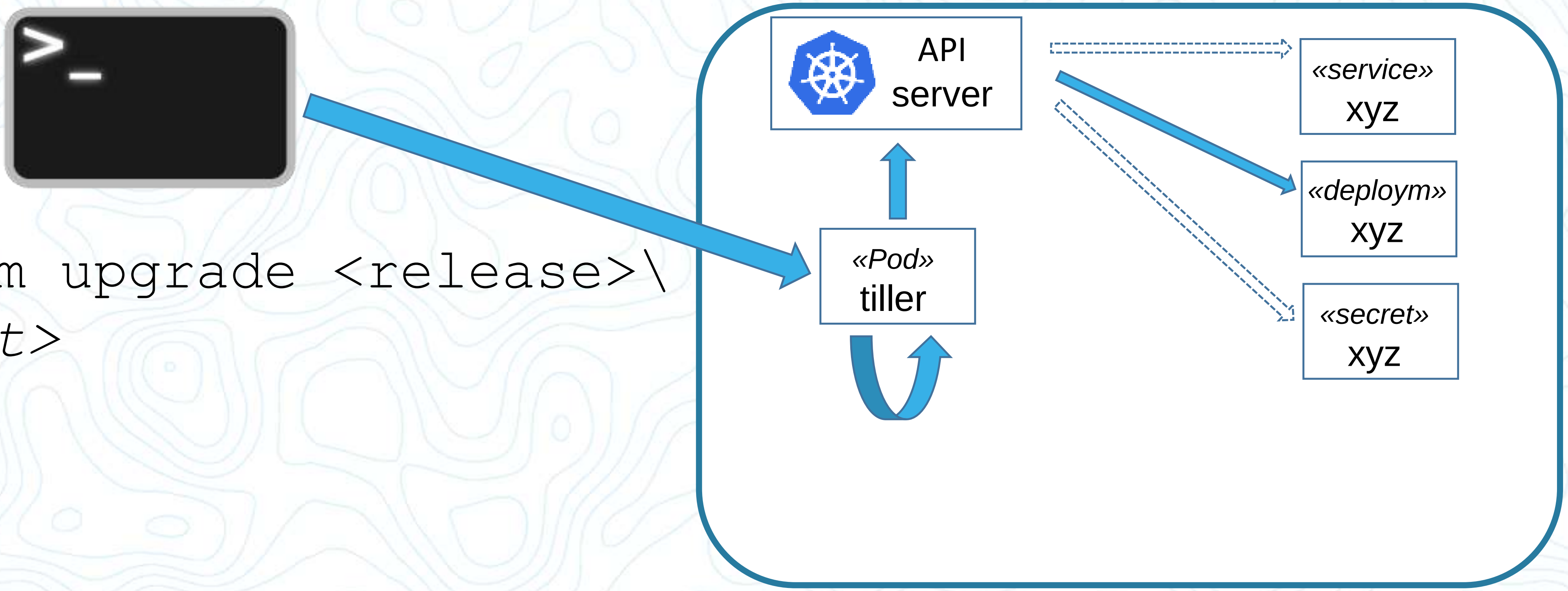
```
$ helm init
```

Installing a Chart



```
$ helm install <chart>
```

Upgrading a Chart

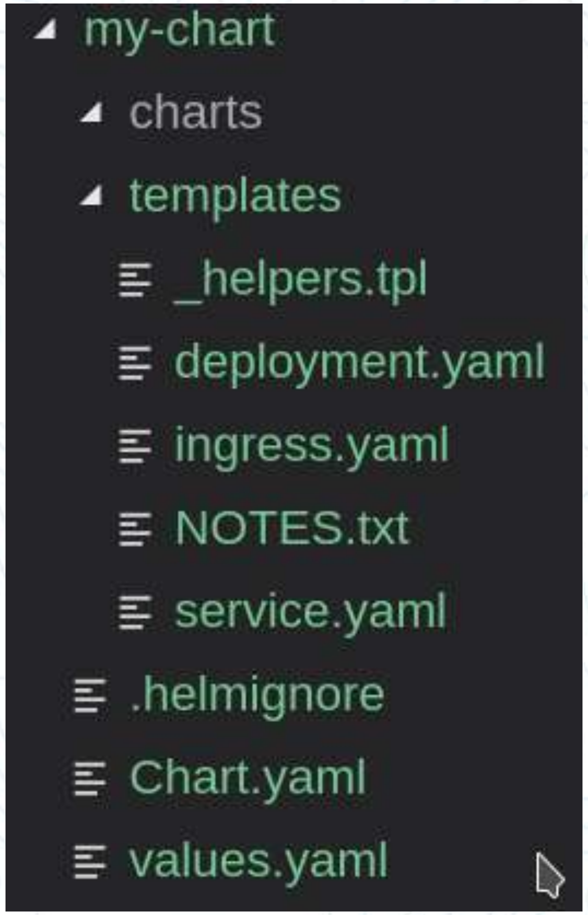


```
$ helm upgrade <release>\
<chart>
```

Charts

Charts

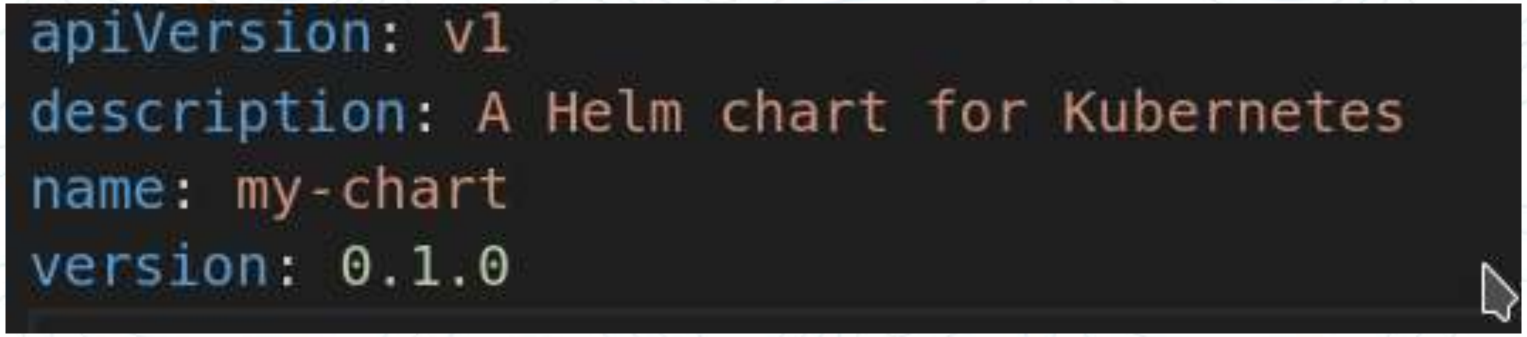
- bundle meta data, templates, default values & docs
- use “`helm create`“ to generate skeleton
- need semantic version numbers



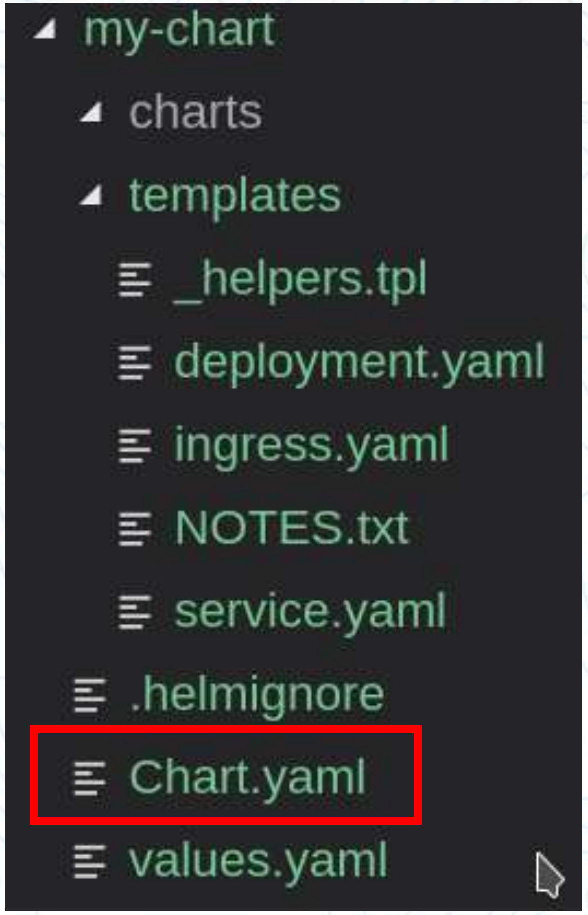
```
my-chart
  charts
  templates
    _helpers.tpl
    deployment.yaml
    ingress.yaml
    NOTES.txt
    service.yaml
  .helmignore
  Chart.yaml
  values.yaml
```

Chart.yaml

- defines name & version
- may include additional meta data, e.g. app version, maintainer etc.



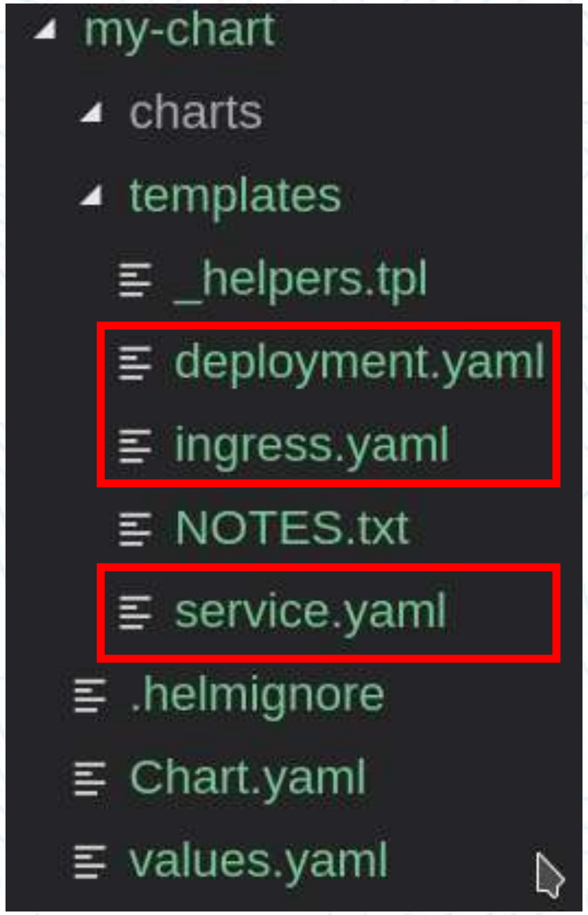
```yaml
apiVersion: v1
description: A Helm chart for Kubernetes
name: my-chart
version: 0.1.0
```



```
my-chart
  charts
  templates
    _helpers.tpl
    deployment.yaml
    ingress.yaml
    NOTES.txt
    service.yaml
  .helmignore
  Chart.yaml
  values.yaml
```

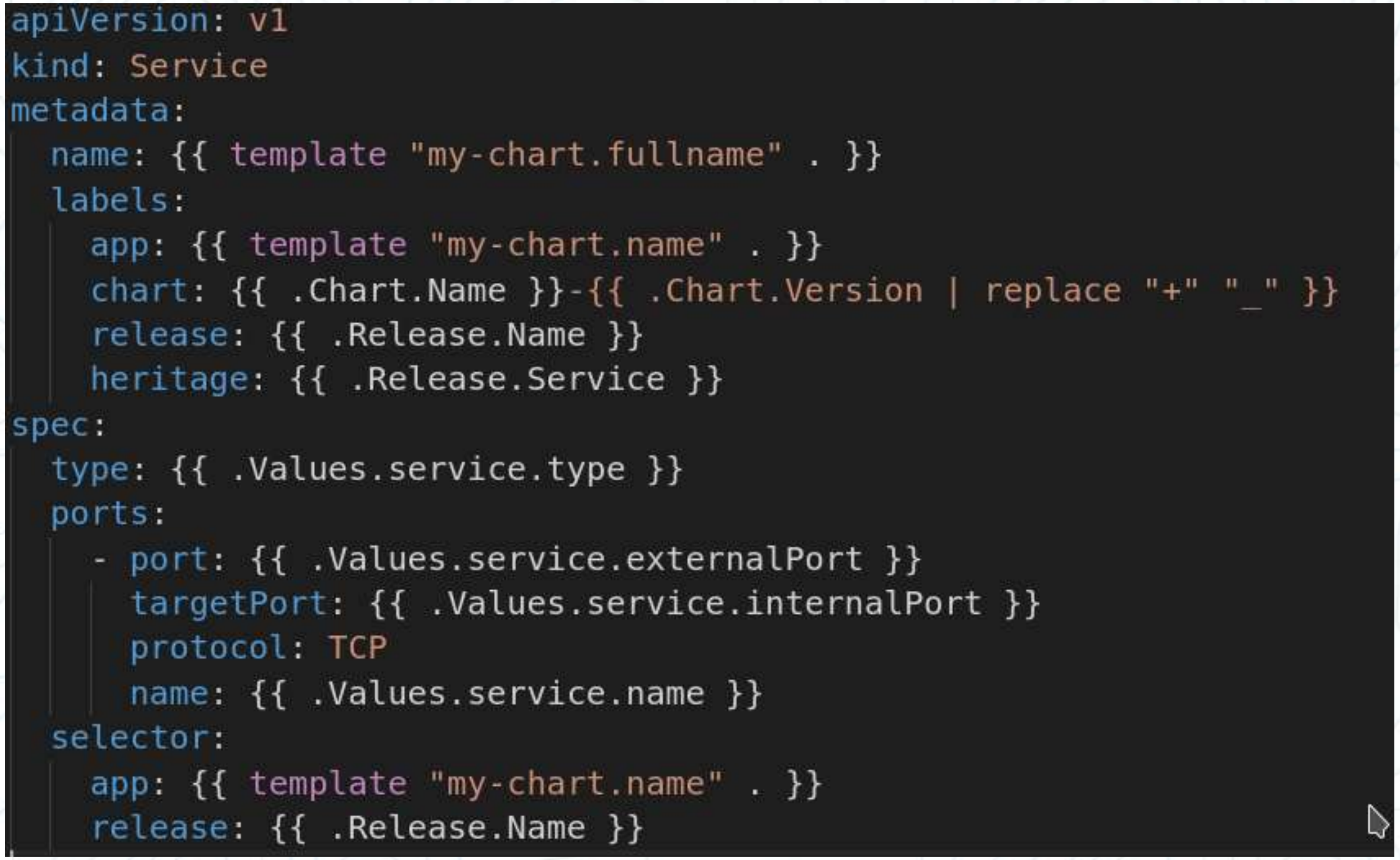
Templates

- YAML files for whatever manifests you need to create
- will be rendered as part of the release

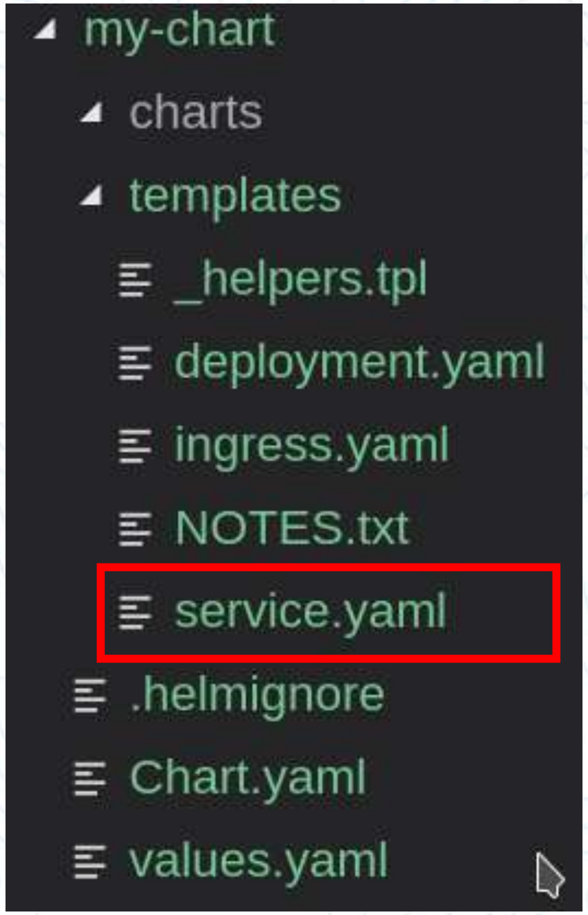


```
my-chart
  charts
  templates
    _helpers.tpl
    deployment.yaml
    ingress.yaml
    NOTES.txt
    service.yaml
  .helmignore
  Chart.yaml
  values.yaml
```

Template Example



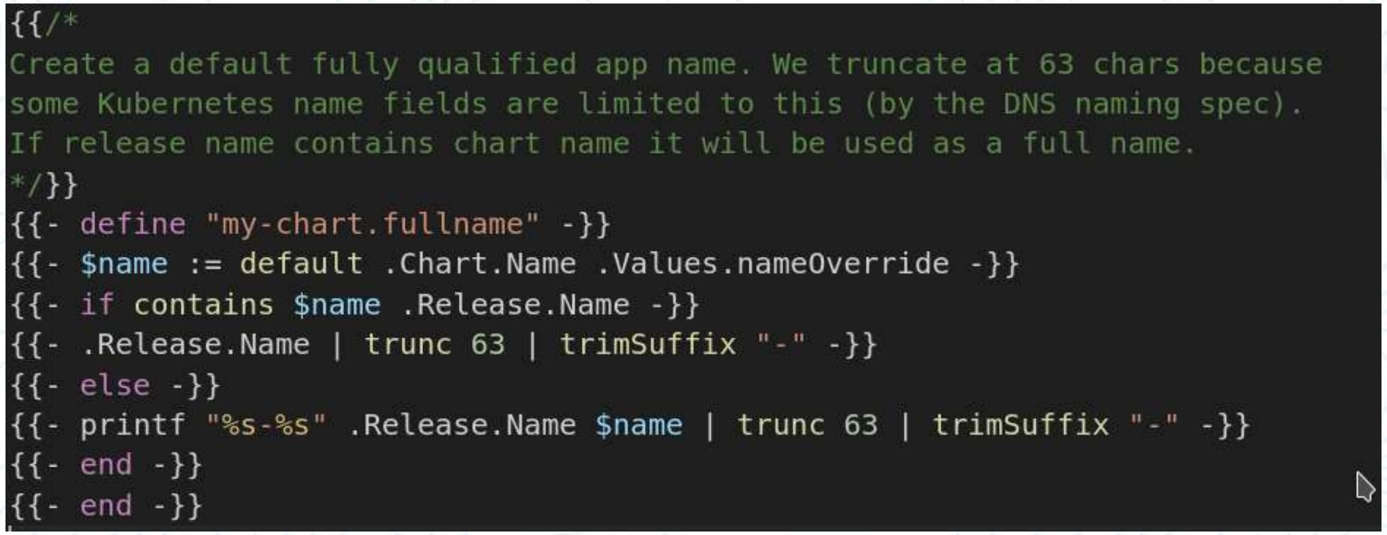
```
apiVersion: v1
kind: Service
metadata:
  name: {{ template "my-chart.fullname" . }}
  labels:
    app: {{ template "my-chart.name" . }}
    chart: {{ .Chart.Name }}-{{ .Chart.Version | replace "+" "_" }}
    release: {{ .Release.Name }}
    heritage: {{ .Release.Service }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.externalPort }}
      targetPort: {{ .Values.service.internalPort }}
      protocol: TCP
      name: {{ .Values.service.name }}
  selector:
    app: {{ template "my-chart.name" . }}
    release: {{ .Release.Name }}
```



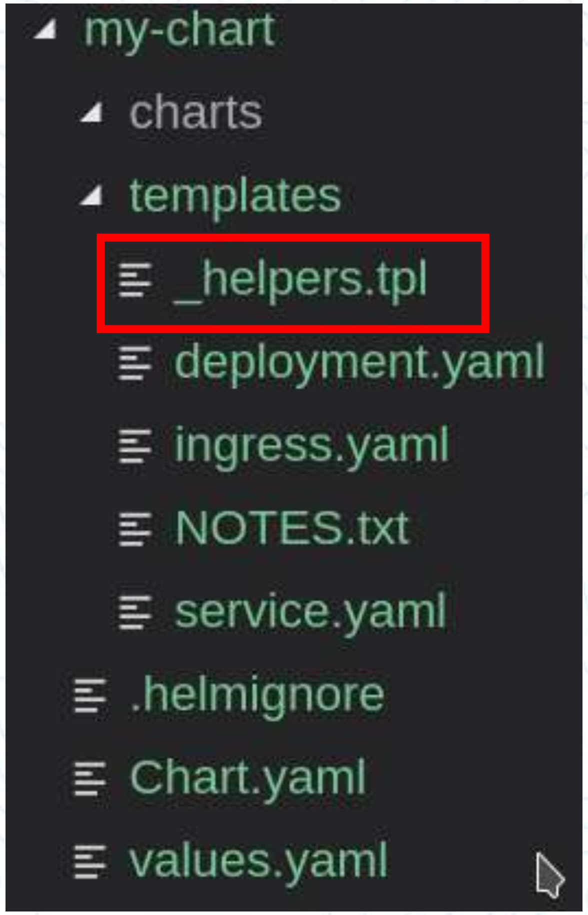
```
my-chart
  charts
  templates
    _helpers.tpl
    deployment.yaml
    ingress.yaml
    NOTES.txt
    service.yaml
  .helmignore
  Chart.yaml
  values.yaml
```

Helpers

- files beginning with “_” will NOT be rendered as part of the release
- contain helper templates used in multiple places within the chart
- By convention end with “.tpl”



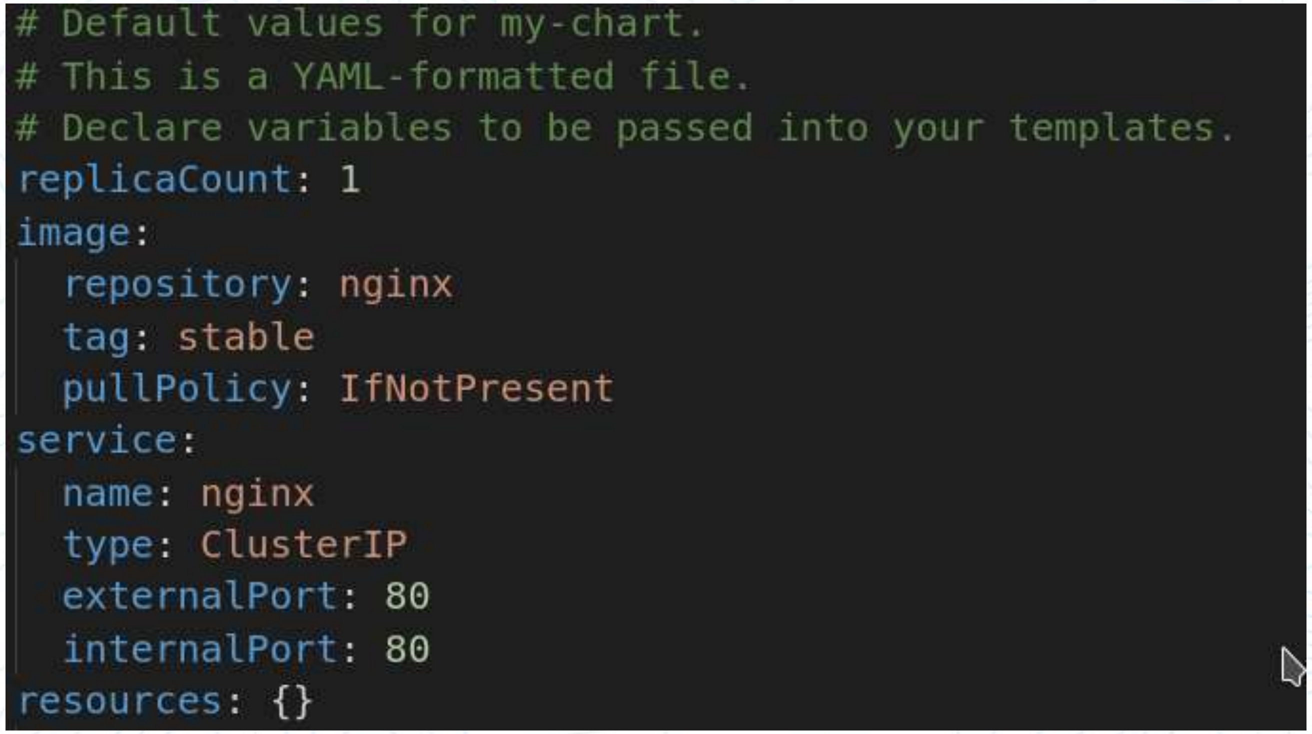
```
{{/*
Create a default fully qualified app name. We truncate at 63 chars because
some Kubernetes name fields are limited to this (by the DNS naming spec).
If release name contains chart name it will be used as a full name.
*/}}
{{- define "my-chart.fullname" -}}
{{- $name := default .Chart.Name .Values.nameOverride -}}
{{- if contains $name .Release.Name -}}
{{- .Release.Name | trunc 63 | trimSuffix "-" -}}
{{- else -}}
{{- printf "%s-%s" .Release.Name $name | trunc 63 | trimSuffix "-" -}}
{{- end -}}
{{- end -}}
```



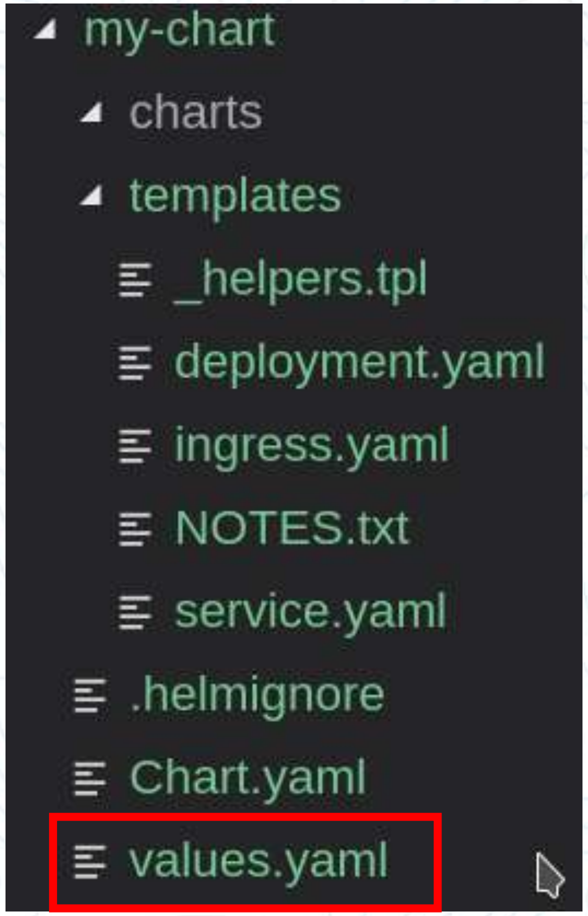
```
my-chart
  charts
  templates
    _helpers.tpl
    deployment.yaml
    ingress.yaml
    NOTES.txt
    service.yaml
  .helmignore
  Chart.yaml
  values.yaml
```

values.yaml

- contains default values for the chart



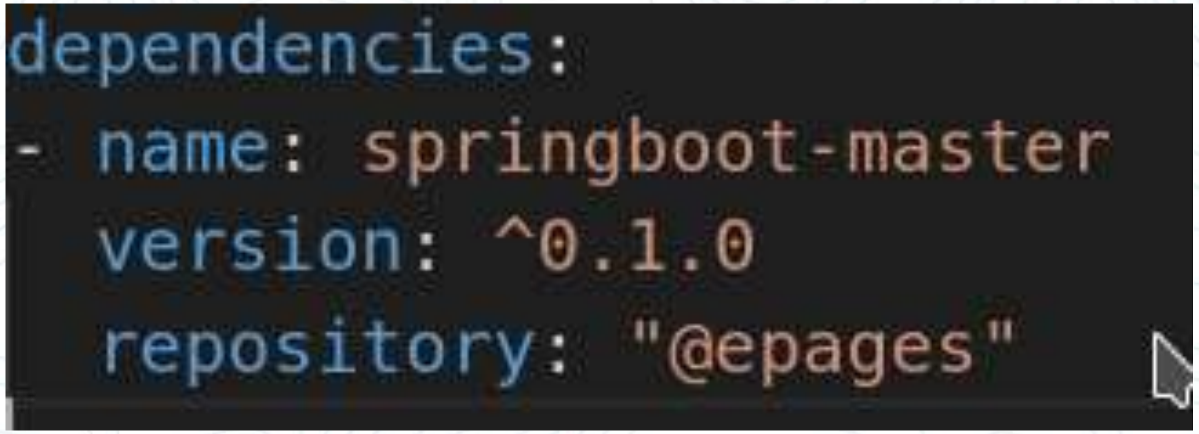
```yaml
# Default values for my-chart.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.
replicaCount: 1
image:
  repository: nginx
  tag: stable
  pullPolicy: IfNotPresent
service:
  name: nginx
  type: ClusterIP
  externalPort: 80
  internalPort: 80
resources: {}
```



```
my-chart
  charts
  templates
    _helpers.tpl
    deployment.yaml
    ingress.yaml
    NOTES.txt
    service.yaml
  .helmignore
  Chart.yaml
  values.yaml
```

requirements.yaml

- lists dependencies for this chart



```
dependencies:
- name: springboot-master
  version: ^0.1.0
  repository: "@epages"
```

How to organize Charts

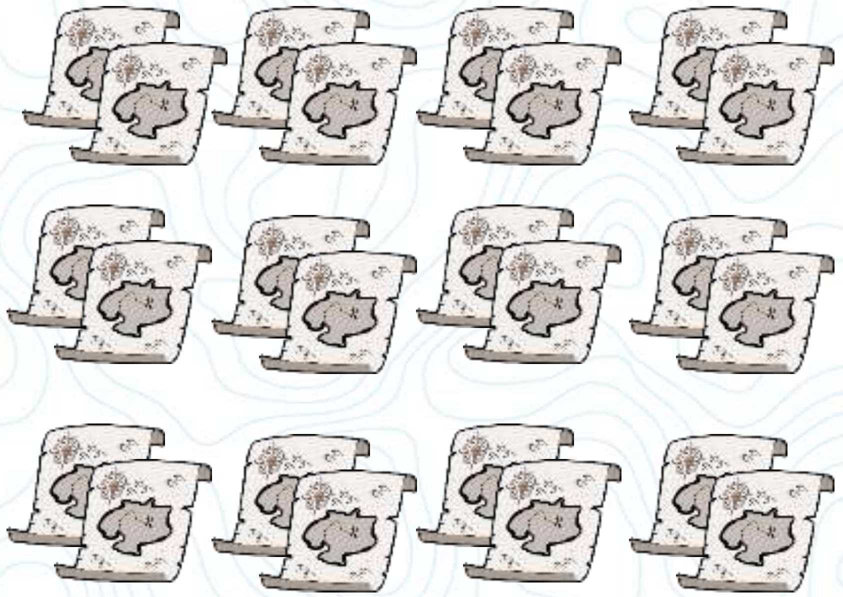


Chart per service

One Chart to Rule Them All

Chart per Service

Pro

- flexible
- low complexity of charts
- simple versioning

Contra

- huge amount of duplication
- difficult to keep consistent
- hard to introduce global changes

One Chart to Rule Them All

Pro

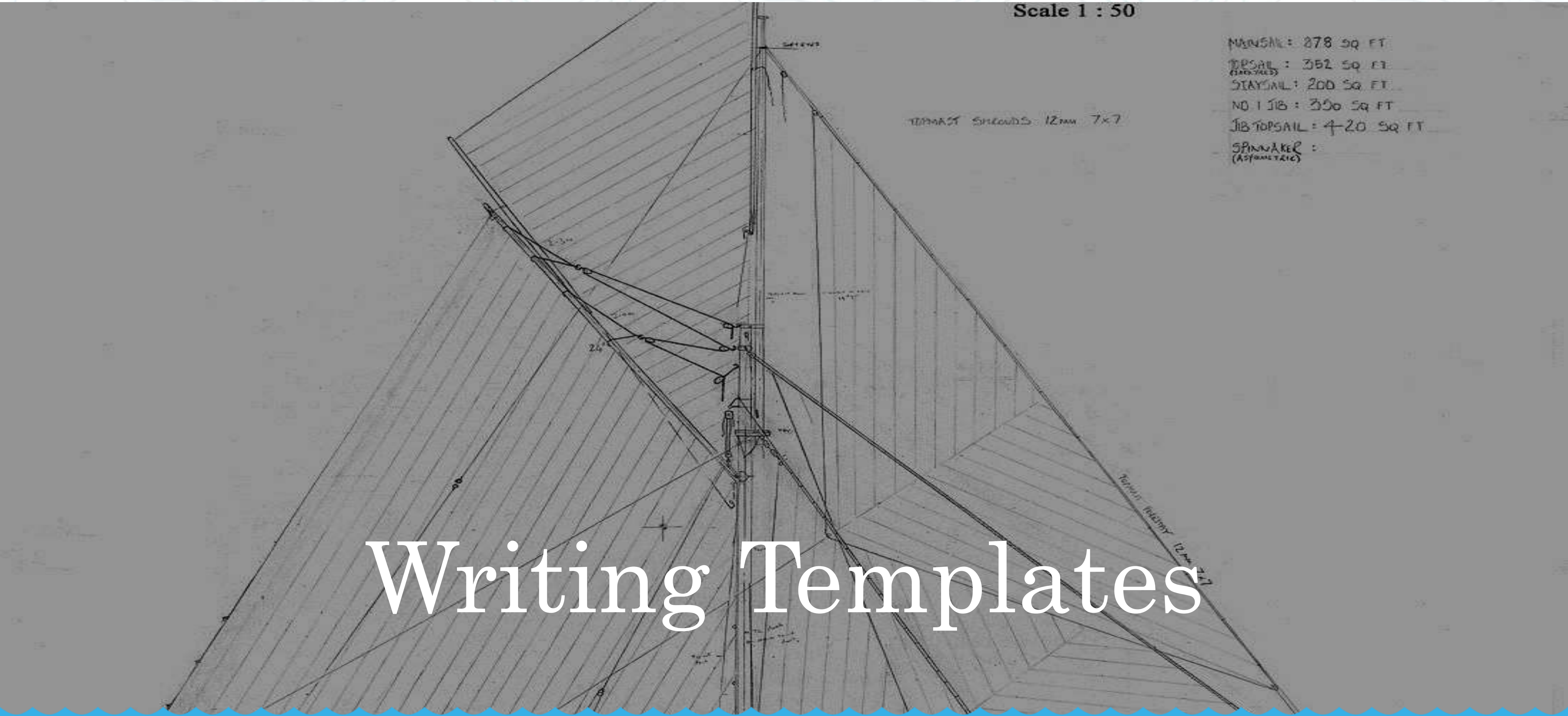
- avoids unnecessary duplication
- consistency
- easy to introduce global changes

Contra

- high complexity of templates
- tight coupling
- less flexible

Hybrid Approach

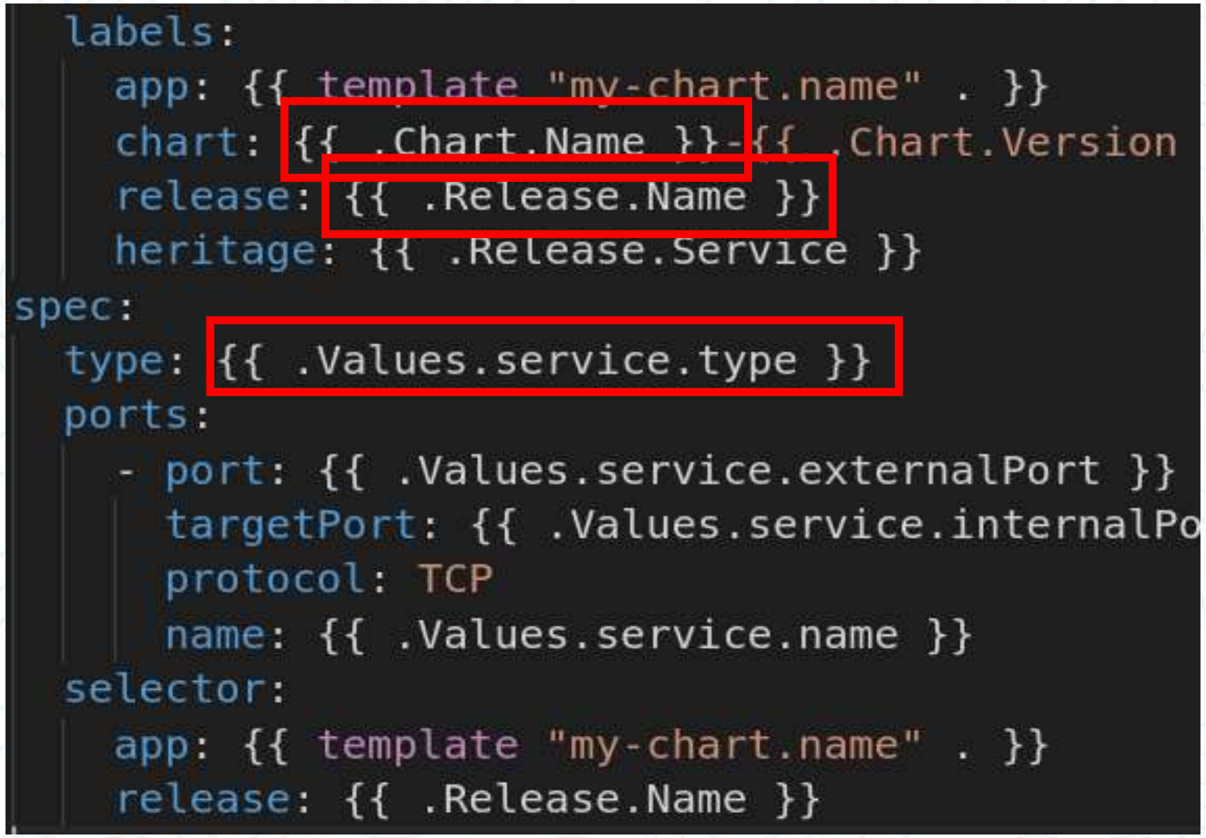
- one “master” chart per service group
- one dependent chart per service, containing
  - values
  - specific additions



Writing Templates

Built-in Objects

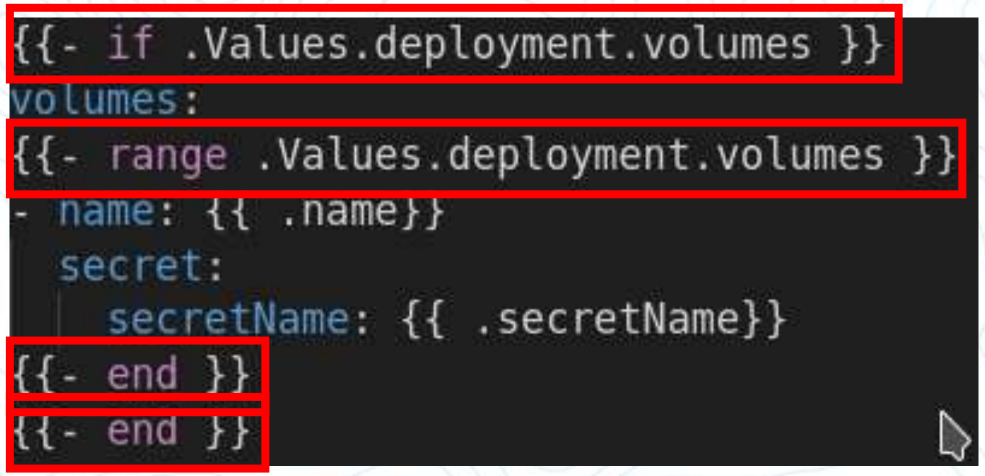
- provide access to specific sets of values
- the main ones are:
  - Chart
  - Release
  - Values



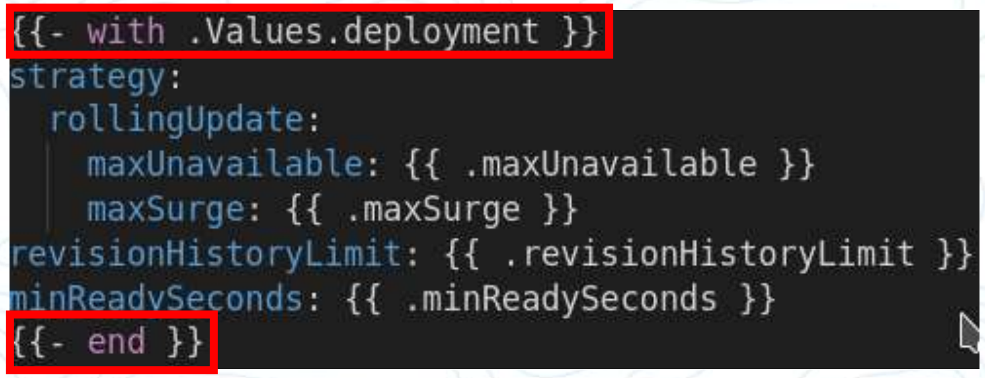
```
    labels:
      app: {{ template "my-chart.name" . }}
      chart: {{ .Chart.Name }}-{{ .Chart.Version
      release: {{ .Release.Name }}
      heritage: {{ .Release.Service }}
spec:
    type: {{ .Values.service.type }}
    ports:
      - port: {{ .Values.service.externalPort }}
        targetPort: {{ .Values.service.internalPo
        protocol: TCP
        name: {{ .Values.service.name }}
    selector:
      app: {{ template "my-chart.name" . }}
      release: {{ .Release.Name }}
```

Control Structures

- Go templates provide typical control structures:
  - `if` / `else`
  - `range` (loop)
  - `with` (scoping)



```
{{- if .Values.deployment.volumes }}
volumes:
{{- range .Values.deployment.volumes }}
- name: {{ .name}}
  secret:
    secretName: {{ .secretName}}
{{- end }}
{{- end }}
```



```
{{- with .Values.deployment }}
strategy:
  rollingUpdate:
    maxUnavailable: {{ .maxUnavailable }}
    maxSurge: {{ .maxSurge }}
revisionHistoryLimit: {{ .revisionHistoryLimit }}
minReadySeconds: {{ .minReadySeconds }}
{{- end }}
```

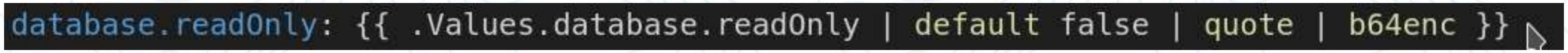
Functions

- Go templates provide some basic functions
- Sprig template library provide a lot of additions

- Examples:
  - default
  - quote
  - b64enc
  - sha256sum
  - trim
  - …

Pipelines

- most functions (and expressions) can be pipelined
- provides the well-known benefits of composability



```
database.readOnly: {{ .Values.database.readOnly | default false | quote | b64enc }}
```

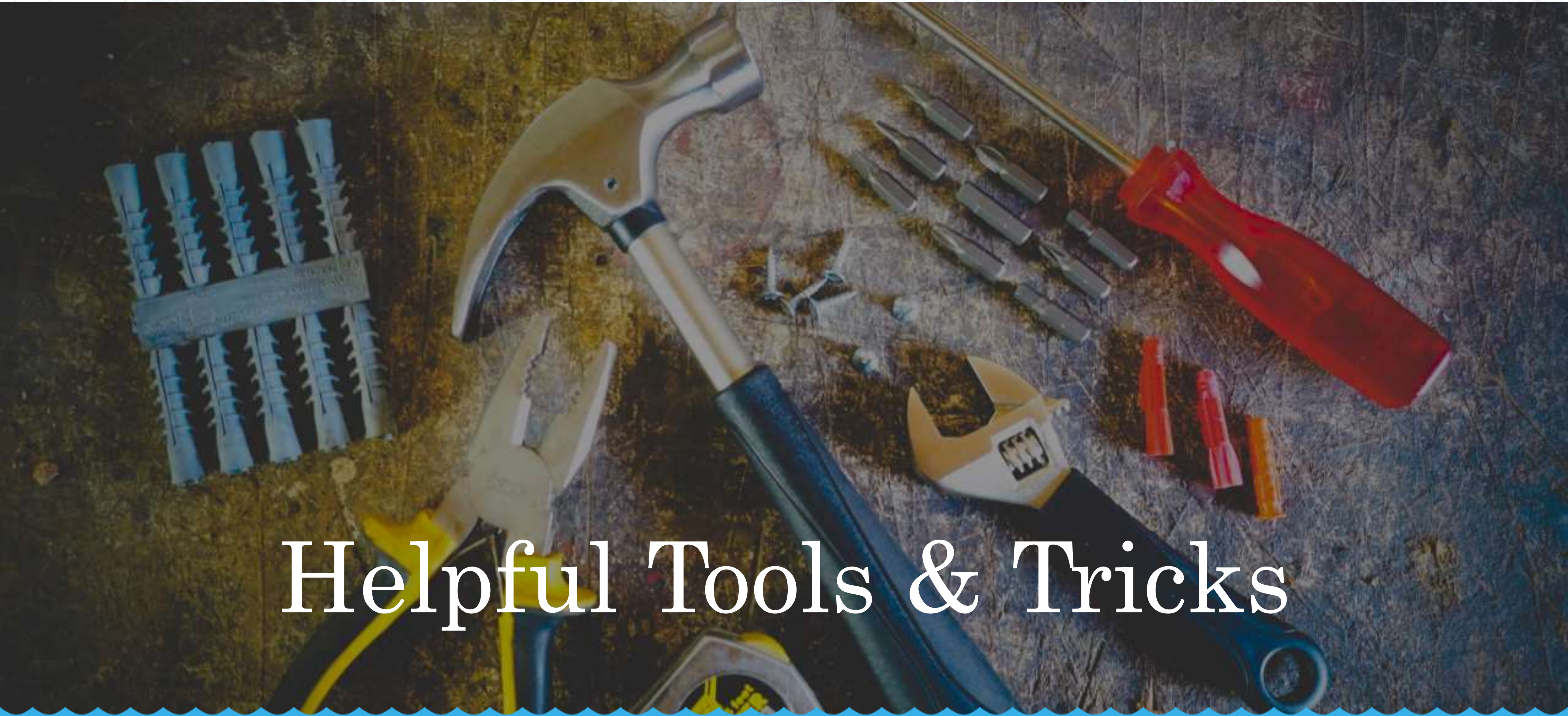
Checking the Results

`helm lint`

- examines a chart for possible issues
- emits ERROR messages when the chart will fail
- emits WARNING when conventions & best practices are violated

`helm template`

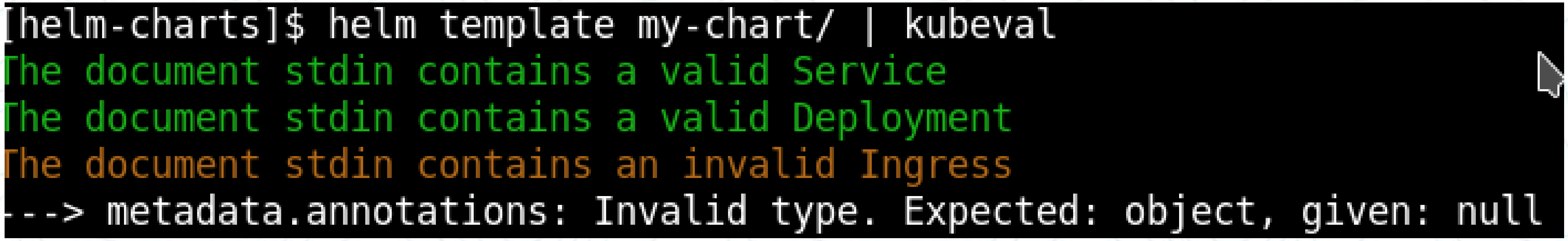
- locally render templates
- does not require Tiller



Helpful Tools & Tricks

kubeval

- `kubeval` validates manifests against specific API versions
- mostly provides detailed validation errors
- pass a manifest file name or pipe to stdin:



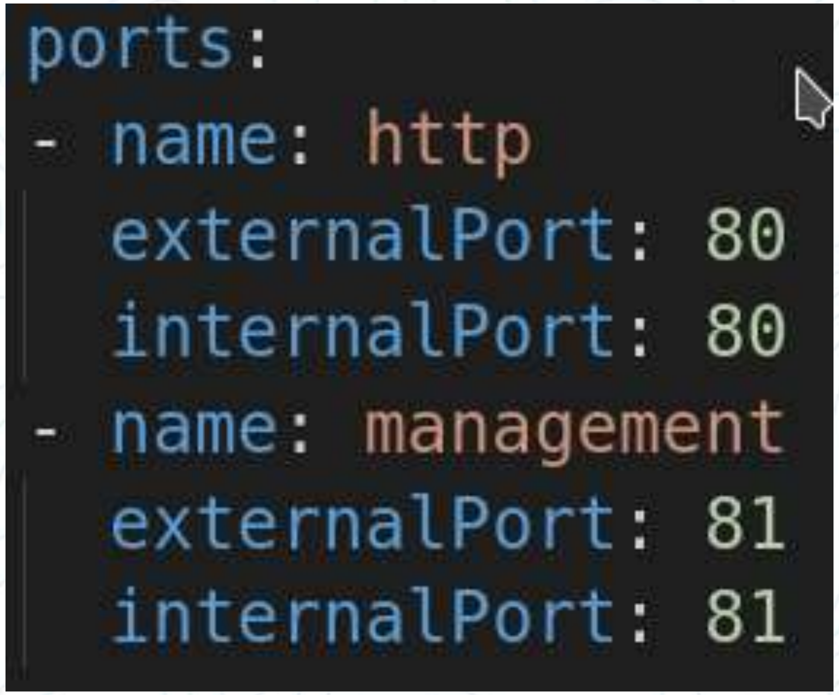
```
[helm-charts]$ helm template my-chart/ | kubeval
The document stdin contains a valid Service
The document stdin contains a valid Deployment
The document stdin contains an invalid Ingress
---> metadata.annotations: Invalid type. Expected: object, given: null
```

kubetest

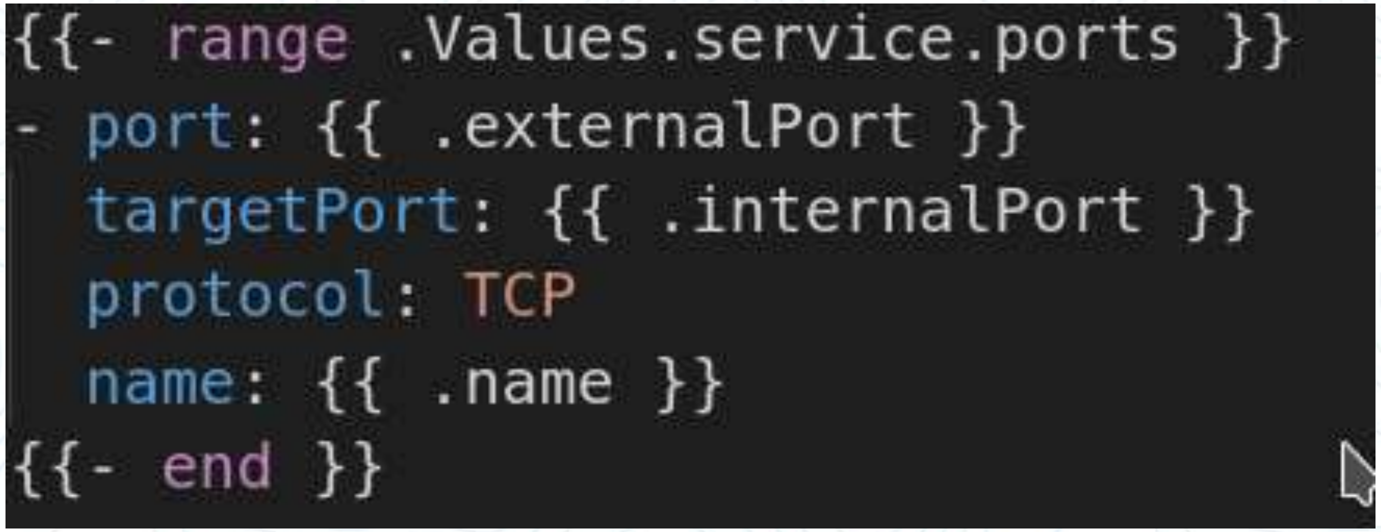
- `kubetest` let‘s you write automatic tests to check the generated manifests
- tests are written in Skylark, a lightweight Python dialect
- can be used to verify some invariants, e.g.
  - existence of specific labels or annotations
  - minimum number of replicas
  - etc.

Tips & Tricks (1)

If you want to override value lists…



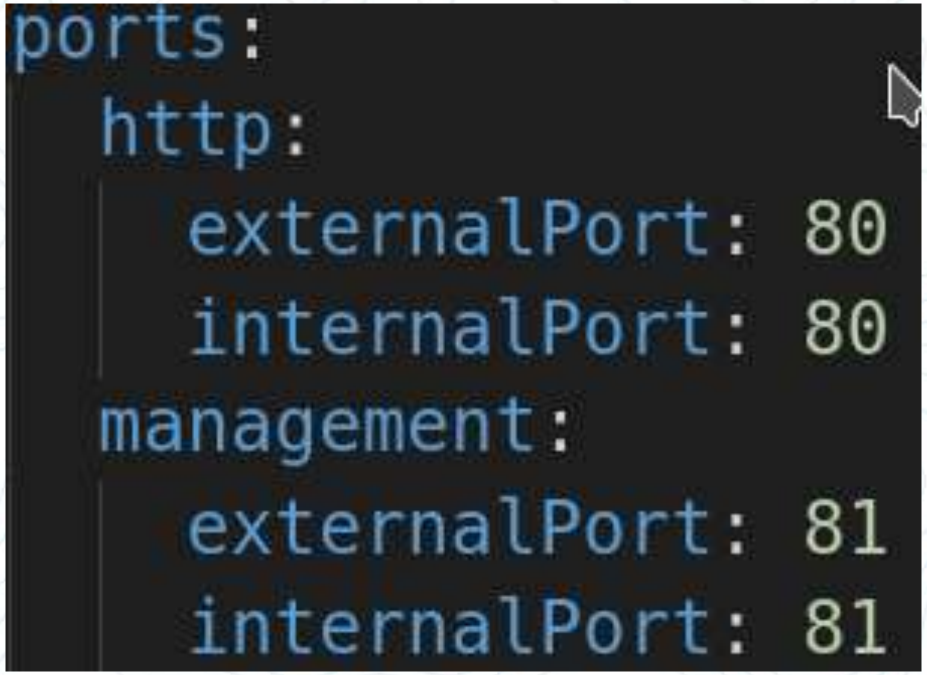
```
ports:
- name: http
  externalPort: 80
  internalPort: 80
- name: management
  externalPort: 81
  internalPort: 81
```



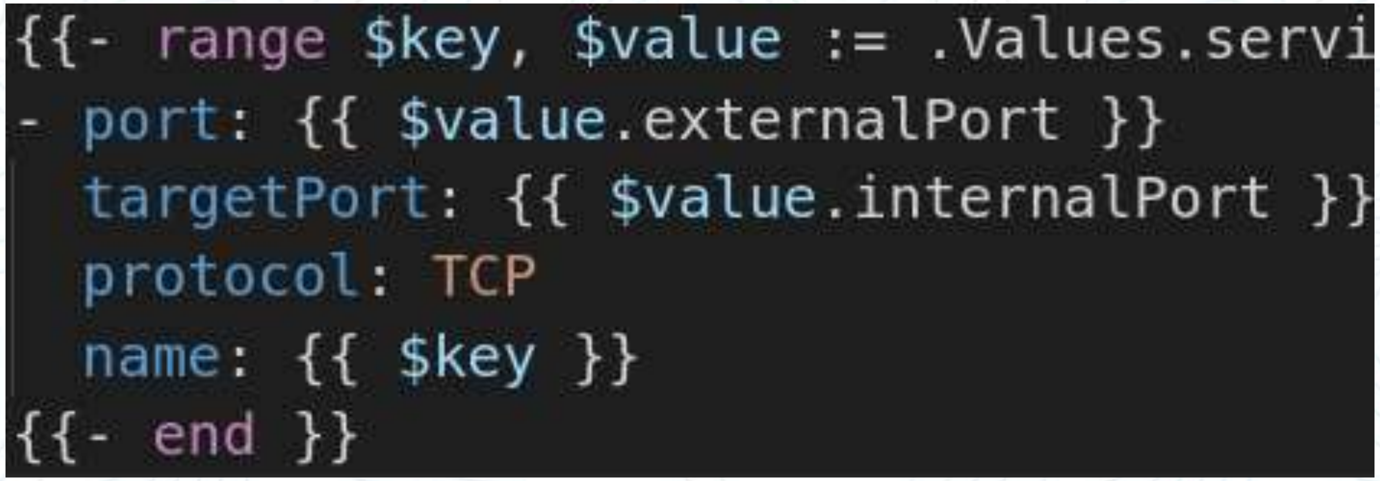
```
{{- range .Values.service.ports }}
- port: {{ .externalPort }}
  targetPort: {{ .internalPort }}
  protocol: TCP
  name: {{ .name }}
{{- end }}
```

Tips & Tricks (2)

…use hashes instead



```
ports:
  http:
    externalPort: 80
    internalPort: 80
  management:
    externalPort: 81
    internalPort: 81
```



```
{{- range $key, $value := .Values.servi
- port: {{ $value.externalPort }}
  targetPort: {{ $value.internalPort }}
  protocol: TCP
  name: {{ $key }}
{{- end }}
```

This way, you can override them indiviually, instead of only whole lists

Questions?

Resources

- Helm docs https://docs.helm.sh/
- Michael Goodness: One Chart to Rule Them All (YouTube) http://bit.ly/2DoERqM
- kubeval https://github.com/garethr/kubeval
- kubetest https://github.com/garethr/kubetest
- Artifactory Helm integration https://jfrog.com/integration/kubernetes-helm/
- ChartMuseum https://github.com/kubernetes-helm/chartmuseum

Thank you!

d.jablonski@epages.com

@djablonski