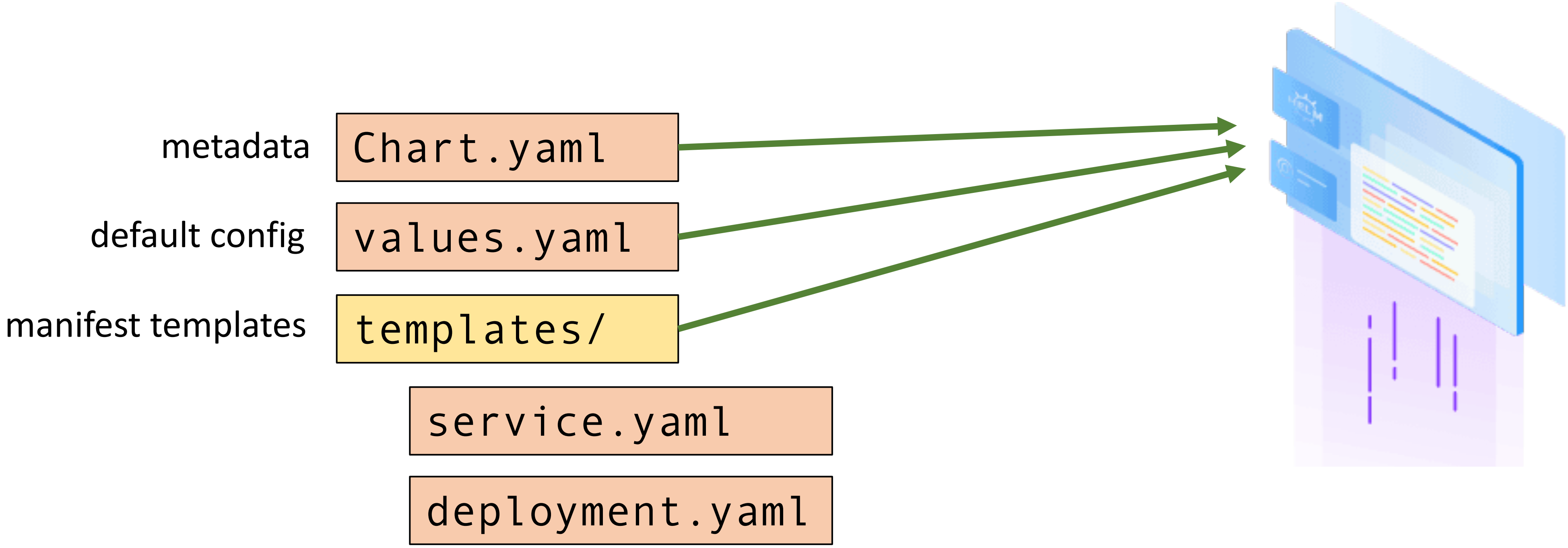
Helm Charts

Helm packages are referred to as **charts** – deployable units for Kubernetes-bound applications.

Charts are comprised of a collection of files (mostly YAML) at well-known locations.



metadata `Chart.yaml`

default config `values.yaml`

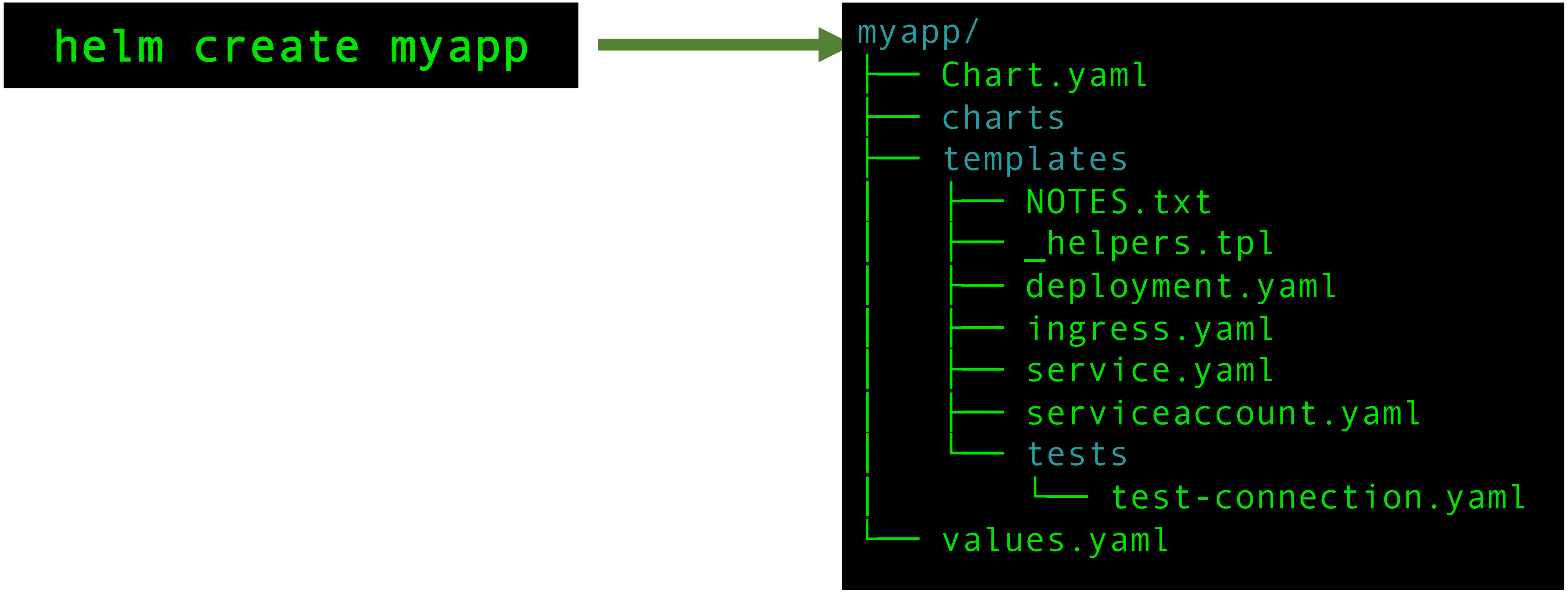
manifest templates `templates/`

`service.yaml`

`deployment.yaml`

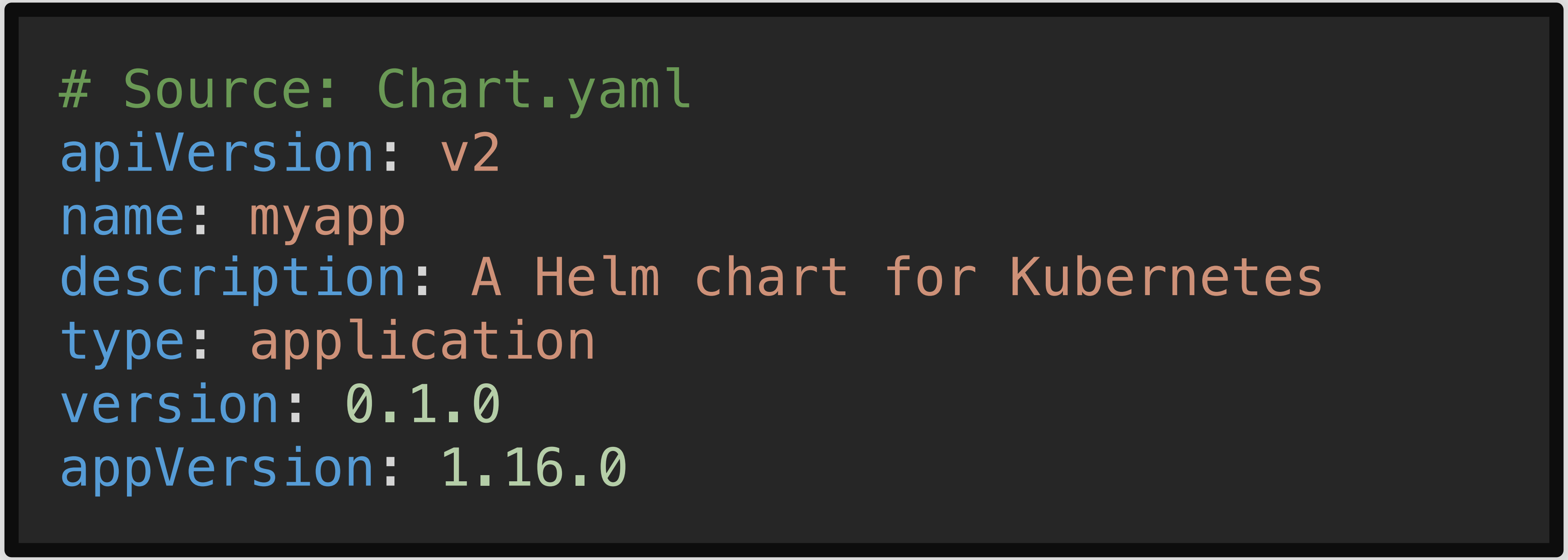
Creating a Chart

Helm provides a command to scaffold out a typical chart layout:

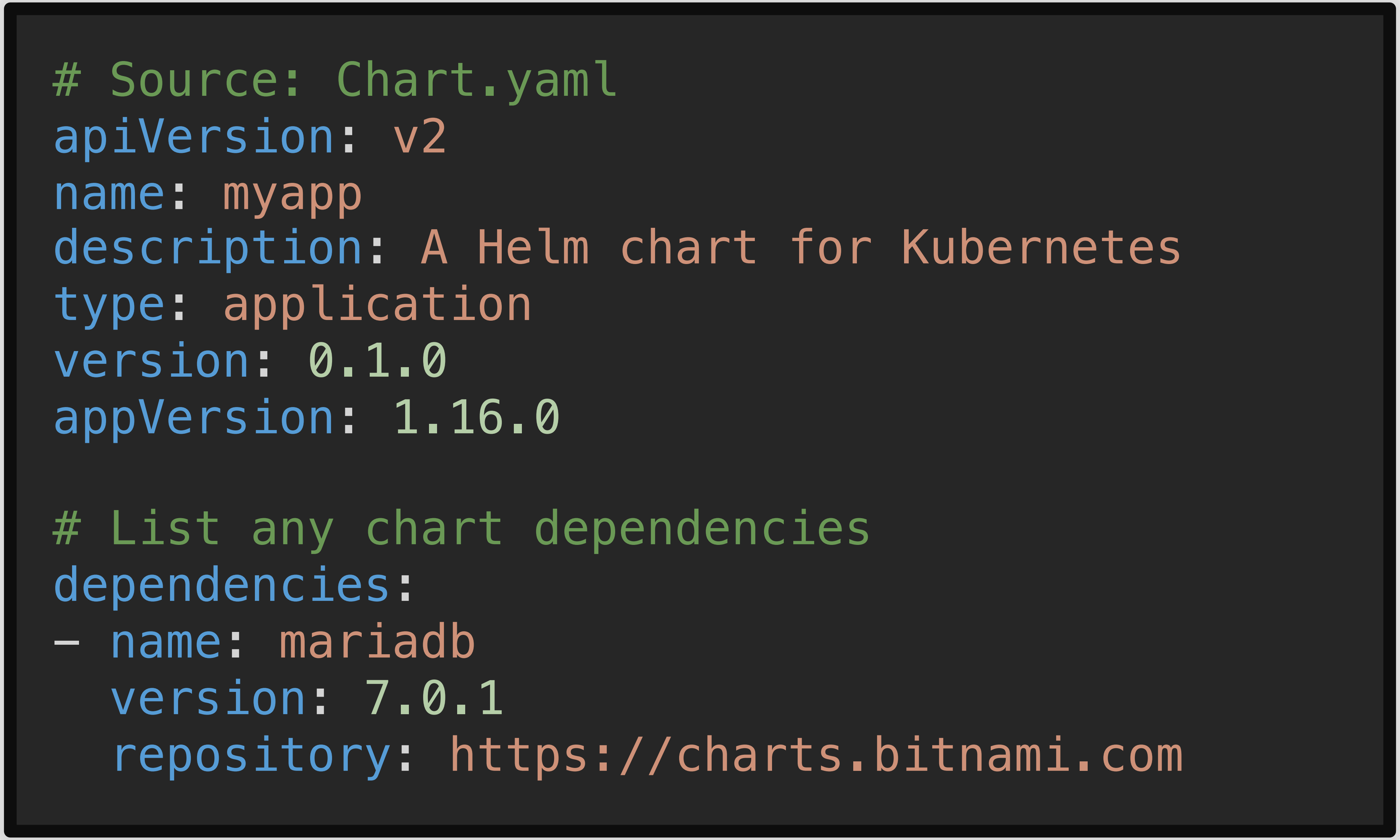


```
helm create myapp
```

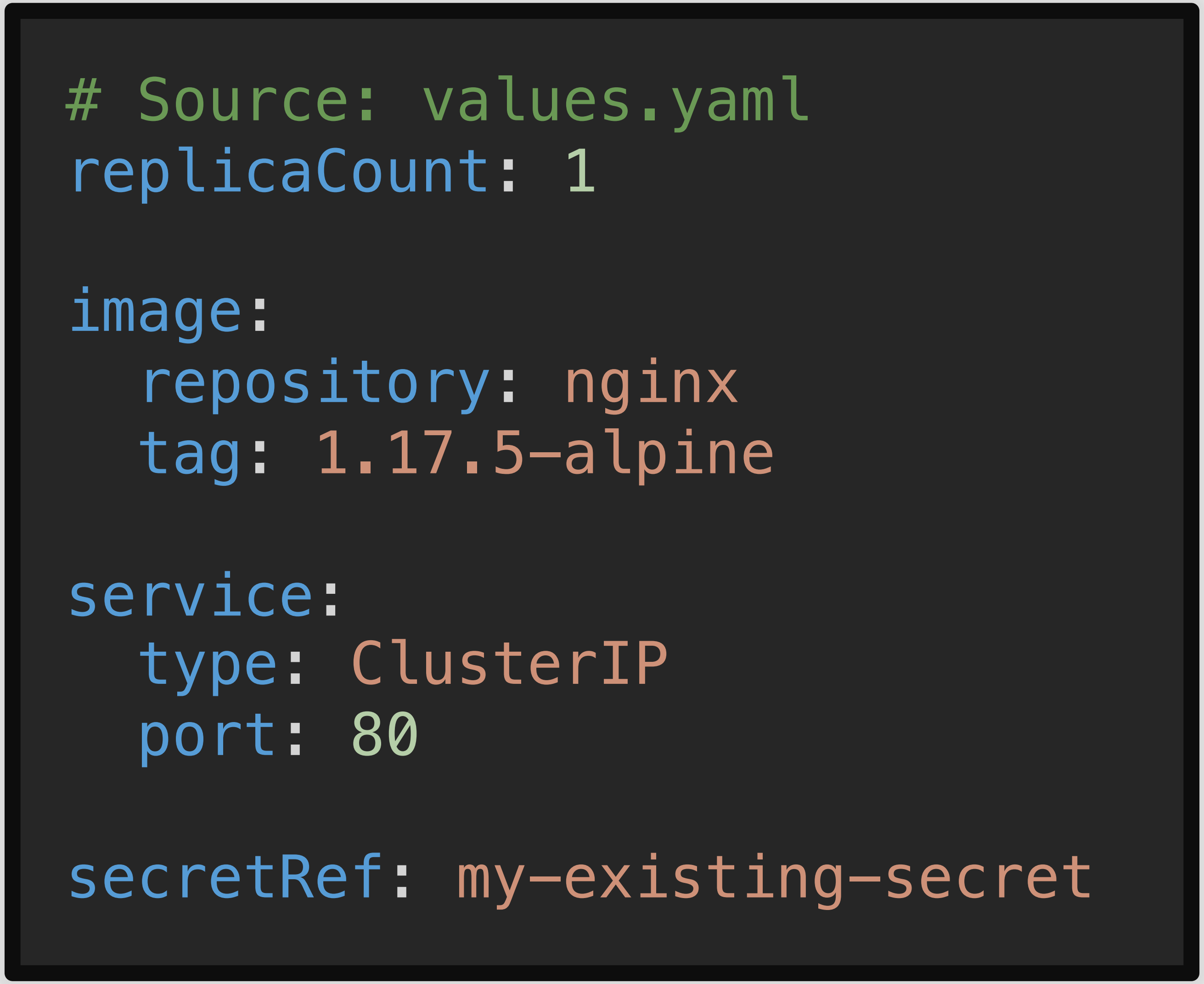
```
myapp/
├── Chart.yaml
├── charts
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── deployment.yaml
│   ├── ingress.yaml
│   ├── service.yaml
│   ├── serviceaccount.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```



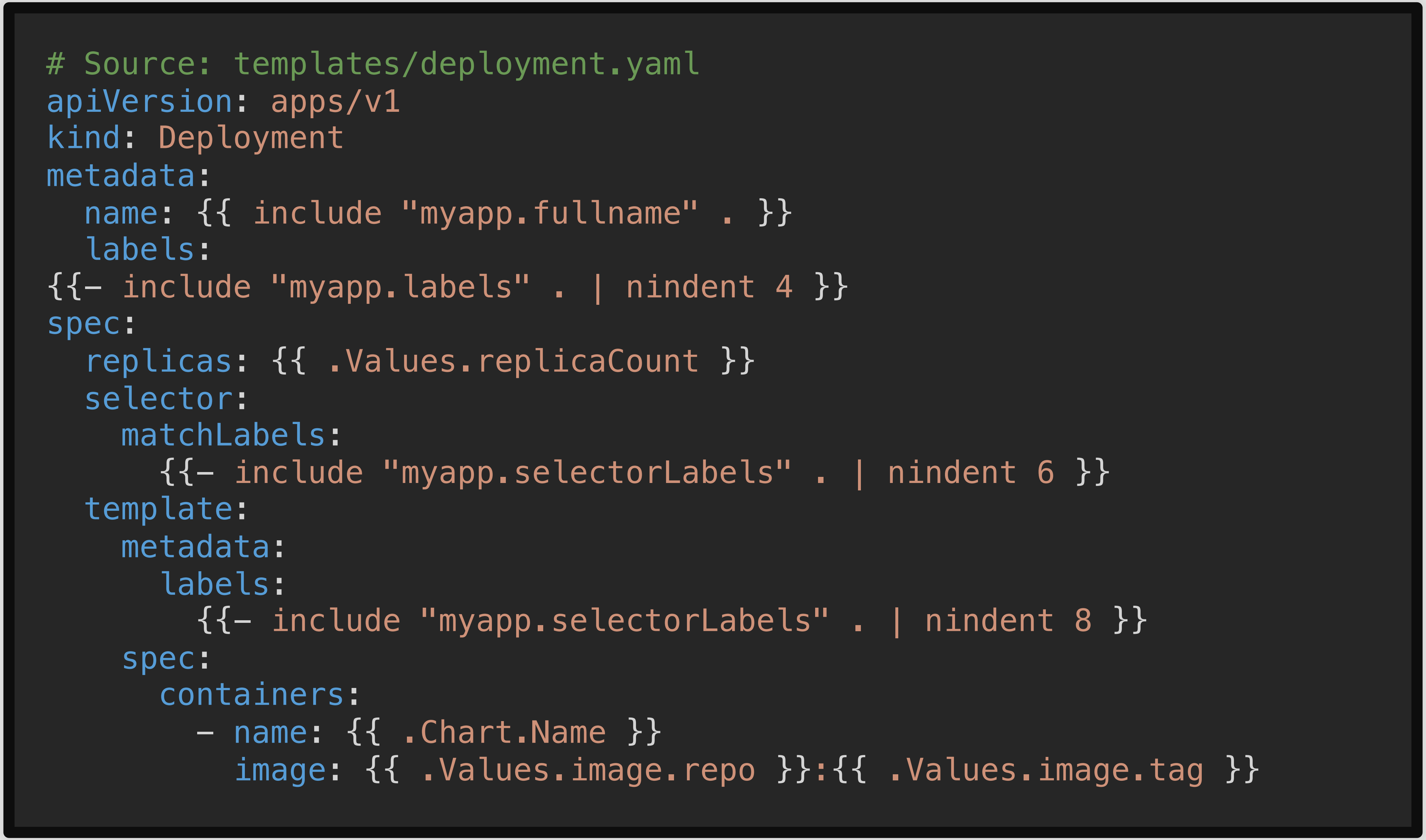
```yaml
# Source: Chart.yaml
apiVersion: v2
name: myapp
description: A Helm chart for Kubernetes
type: application
version: 0.1.0
appVersion: 1.16.0
```



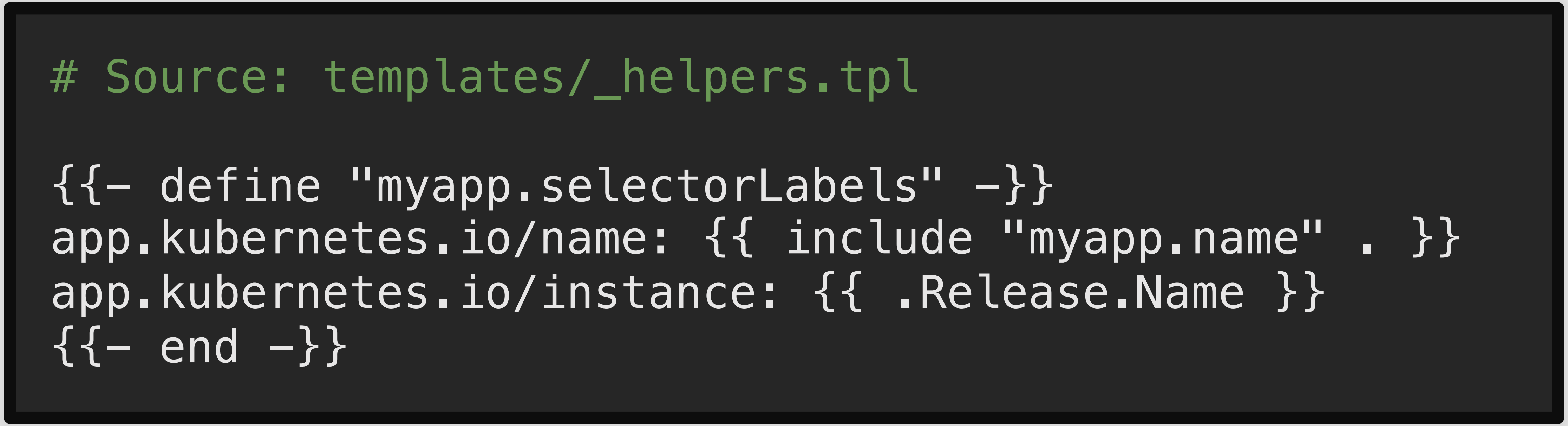
```yaml
# Source: Chart.yaml
apiVersion: v2
name: myapp
description: A Helm chart for Kubernetes
type: application
version: 0.1.0
appVersion: 1.16.0

# List any chart dependencies
dependencies:
- name: mariadb
  version: 7.0.1
  repository: https://charts.bitnami.com
```



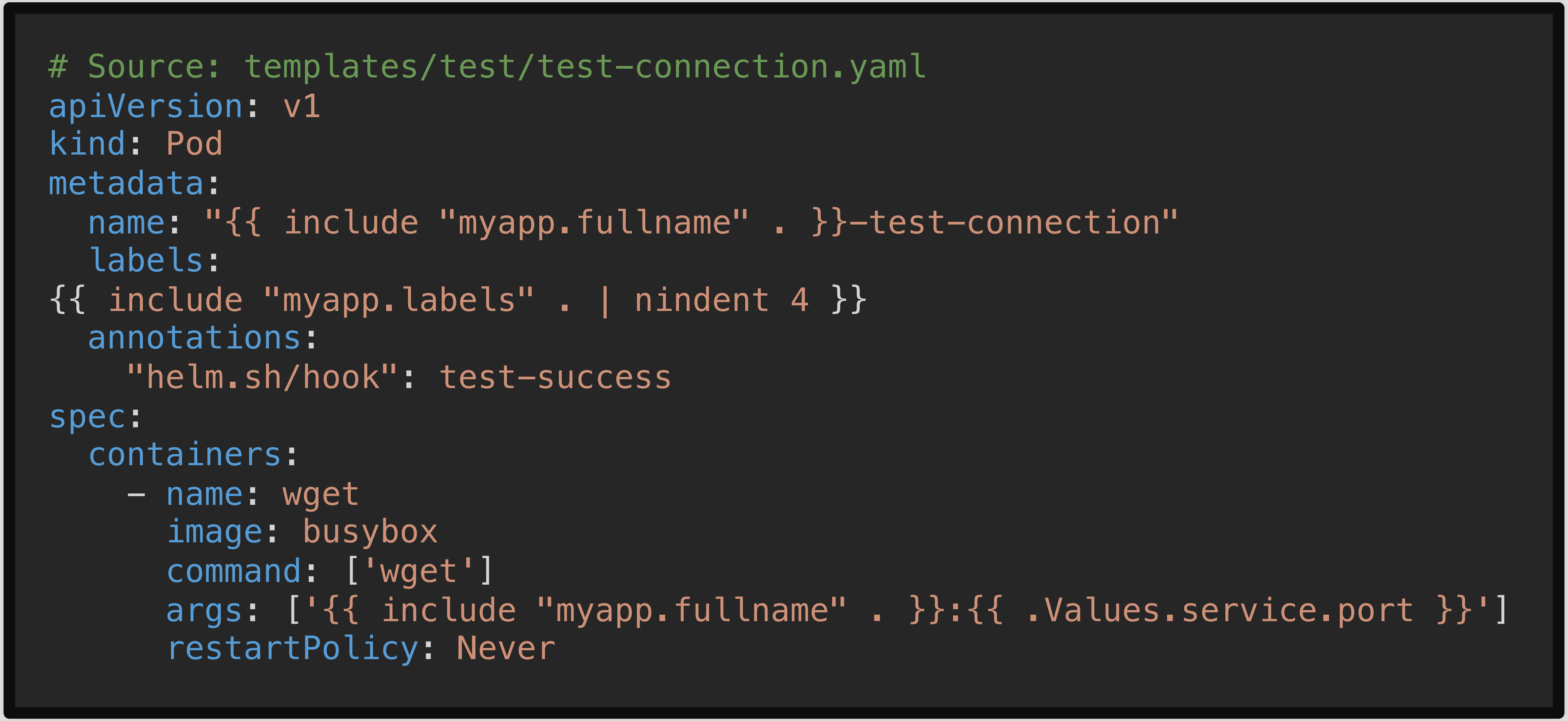
```yaml
# Source: values.yaml
replicaCount: 1

image:
  repository: nginx
  tag: 1.17.5-alpine

service:
  type: ClusterIP
  port: 80

secretRef: my-existing-secret
```



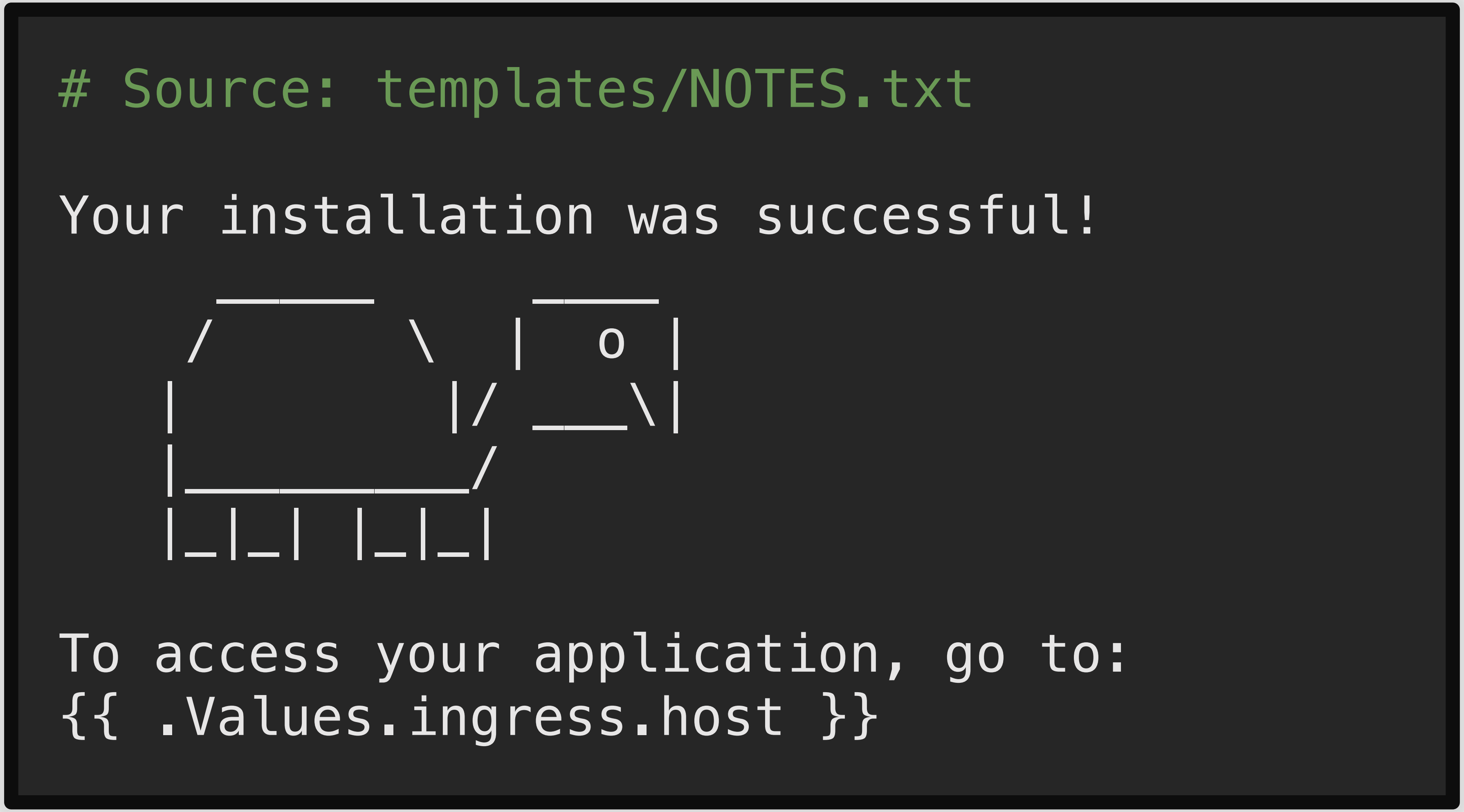
```
# Source: templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "myapp.fullname" . }}
  labels:
{{- include "myapp.labels" . | nindent 4 }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      {{- include "myapp.selectorLabels" . | nindent 6 }}
  template:
    metadata:
      labels:
        {{- include "myapp.selectorLabels" . | nindent 8 }}
    spec:
      containers:
        - name: {{ .Chart.Name }}
          image: {{ .Values.image.repo }}:{{ .Values.image.tag }}
```



```
# Source: templates/_helpers.tpl

{{- define "myapp.selectorLabels" -}}
app.kubernetes.io/name: {{ include "myapp.name" . }}
app.kubernetes.io/instance: {{ .Release.Name }}
{{- end -}}
```



```yaml
# Source: templates/test/test-connection.yaml
apiVersion: v1
kind: Pod
metadata:
  name: "{{ include "myapp.fullname" . }}-test-connection"
  labels:
{{ include "myapp.labels" . | nindent 4 }}
  annotations:
    "helm.sh/hook": test-success
spec:
  containers:
    - name: wget
      image: busybox
      command: ['wget']
      args: ['{{ include "myapp.fullname" . }}:{{ .Values.service.port }}']
      restartPolicy: Never
```



```
# Source: templates/NOTES.txt

Your installation was successful!

     _____     ____
    /      \  |  o |
   |        |/ ___\|
   |_________/
   |_|_| |_|_|

To access your application, go to:
{{ .Values.ingress.host }}
```

Templates

Files under the **templates/** directory are treated as dynamic YAML templates using the Go template language with some added functionality.

YAML templating prevents config duplication, and allows you to install the same chart in dev, staging, and production environments.

Values

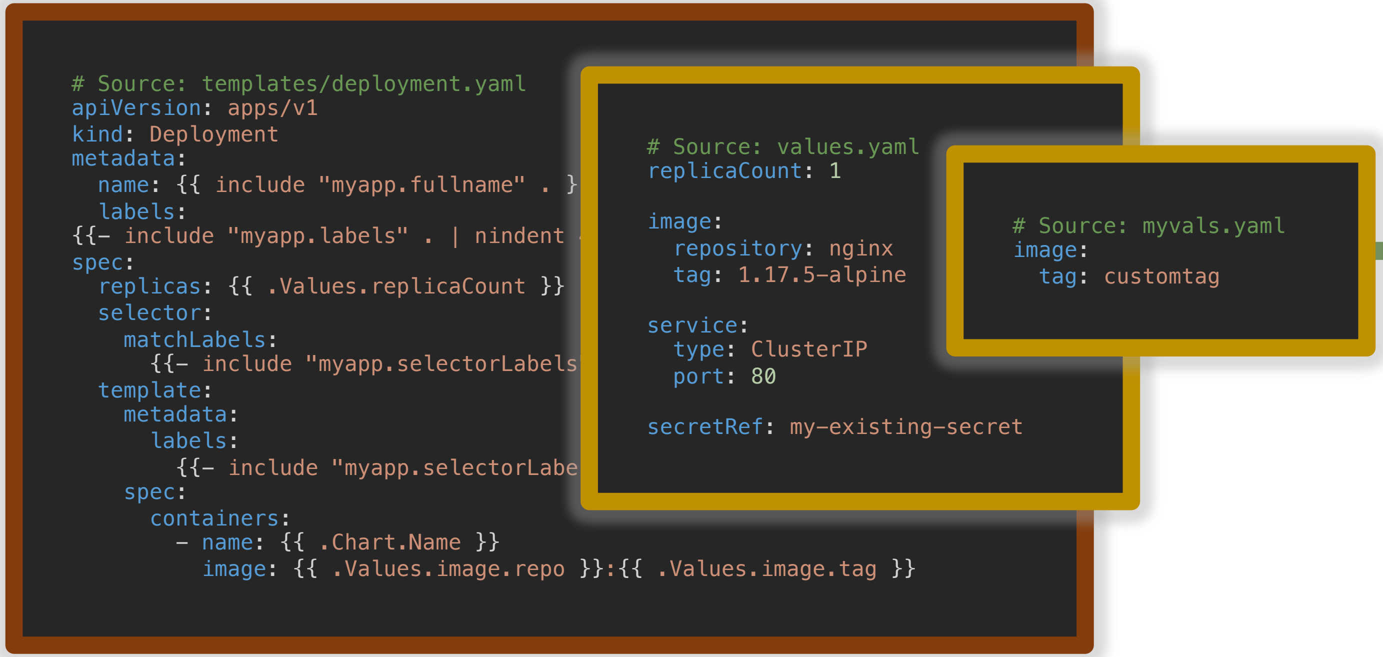
Values files contain a collection of key-values that represent the configuration settings for a chart.

values.yaml is the default, baseline values file.

Override default configuration of the chart by using additional values files or individual key-value pairs.

Templates + Values = 🚀

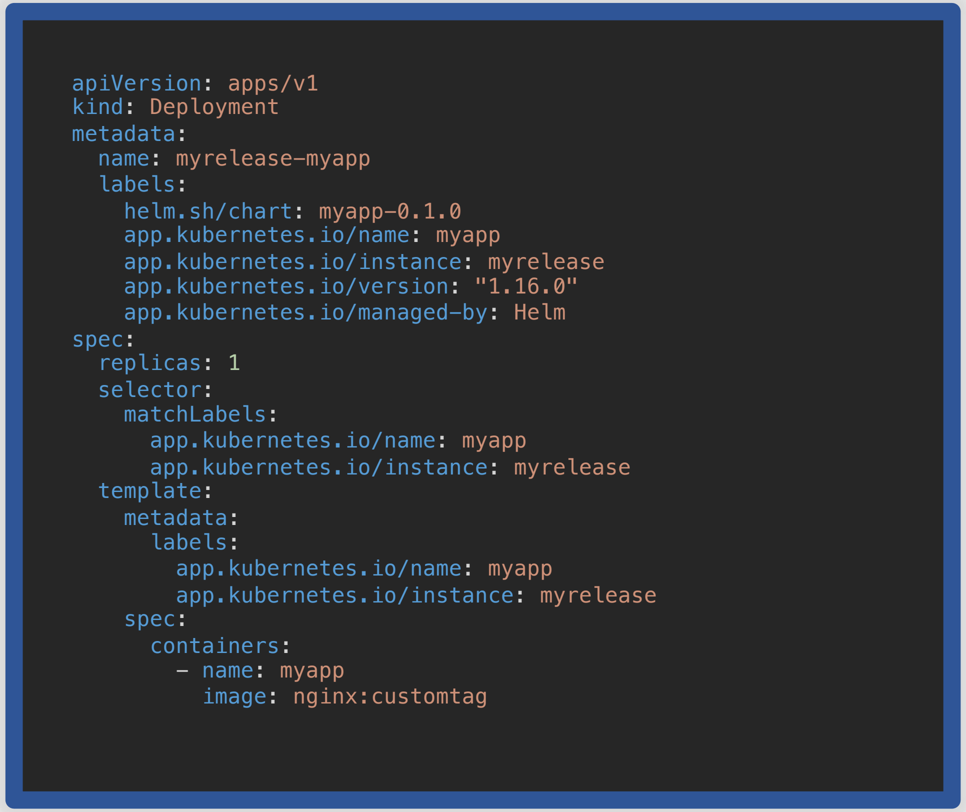
The templates are rendered at install time against provided values, resulting in static, valid Kubernetes YAML.



```yaml
# Source: templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "myapp.fullname" . }
  labels:
{{- include "myapp.labels" . | nindent
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      {{- include "myapp.selectorLabels
  template:
    metadata:
      labels:
        {{- include "myapp.selectorLabe
    spec:
      containers:
        - name: {{ .Chart.Name }}
          image: {{ .Values.image.repo }}:{{ .Values.image.tag }}
```

```yaml
# Source: values.yaml
replicaCount: 1

image:
  repository: nginx
  tag: 1.17.5-alpine

service:
  type: ClusterIP
  port: 80

secretRef: my-existing-secret
```

```yaml
# Source: myvals.yaml
image:
  tag: customtag
```



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myrelease-myapp
  labels:
    helm.sh/chart: myapp-0.1.0
    app.kubernetes.io/name: myapp
    app.kubernetes.io/instance: myrelease
    app.kubernetes.io/version: "1.16.0"
    app.kubernetes.io/managed-by: Helm
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: myapp
      app.kubernetes.io/instance: myrelease
  template:
    metadata:
      labels:
        app.kubernetes.io/name: myapp
        app.kubernetes.io/instance: myrelease
    spec:
      containers:
        - name: myapp
          image: nginx:customtag
```

Installing a Chart

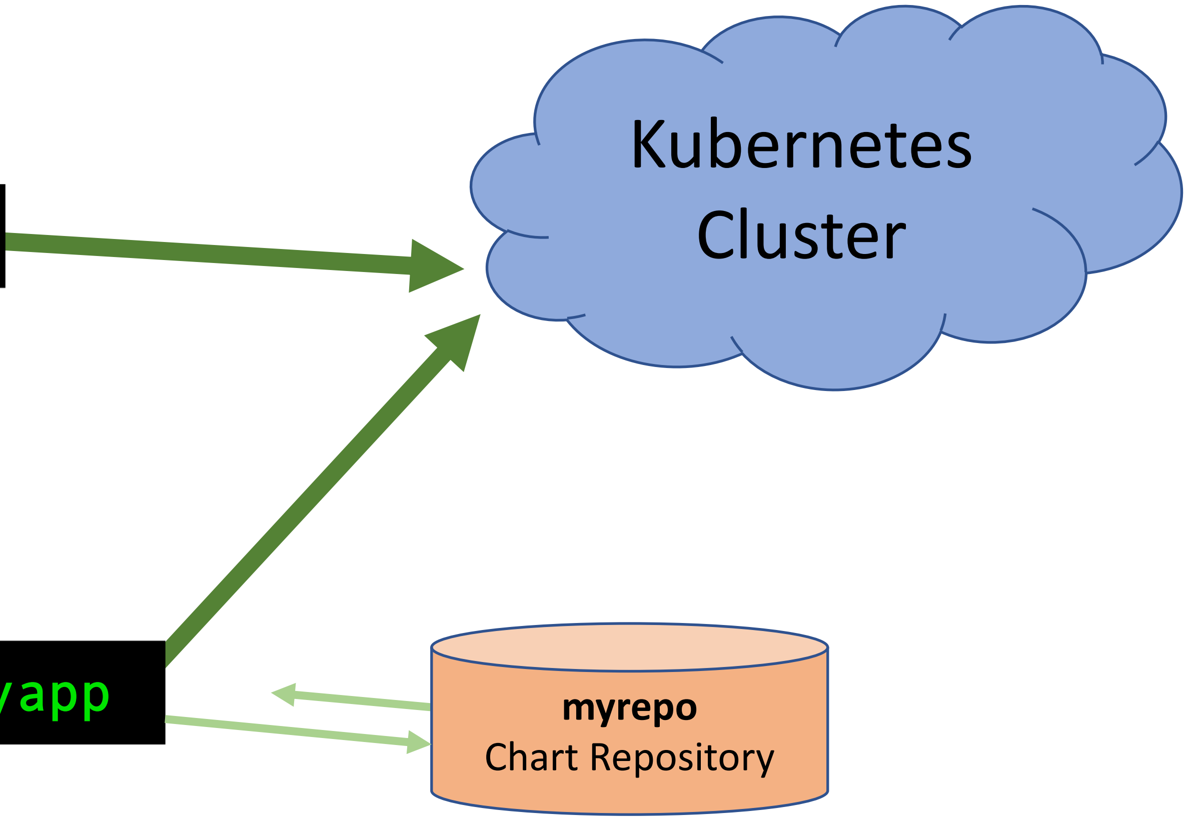
In an environment where you are authenticated against a running Kubernetes cluster, use Helm to install a chart from a chart directory, or from a remote *chart repository*.

1. From a chart directory:

```
helm install myrelease ./myapp
```

2. From a remote chart repository:

```
helm install myrelease myrepo/myapp
```



Kubernetes Cluster

myrepo
Chart Repository

Using Custom Values

Pass along any number of values files or individual key-value pairs in order to override chart defaults, overlayed from left to right

1. Using a values file:

```
helm install myrelease ./myapp -f custom.yaml
```

2. Using individual key-value pair:

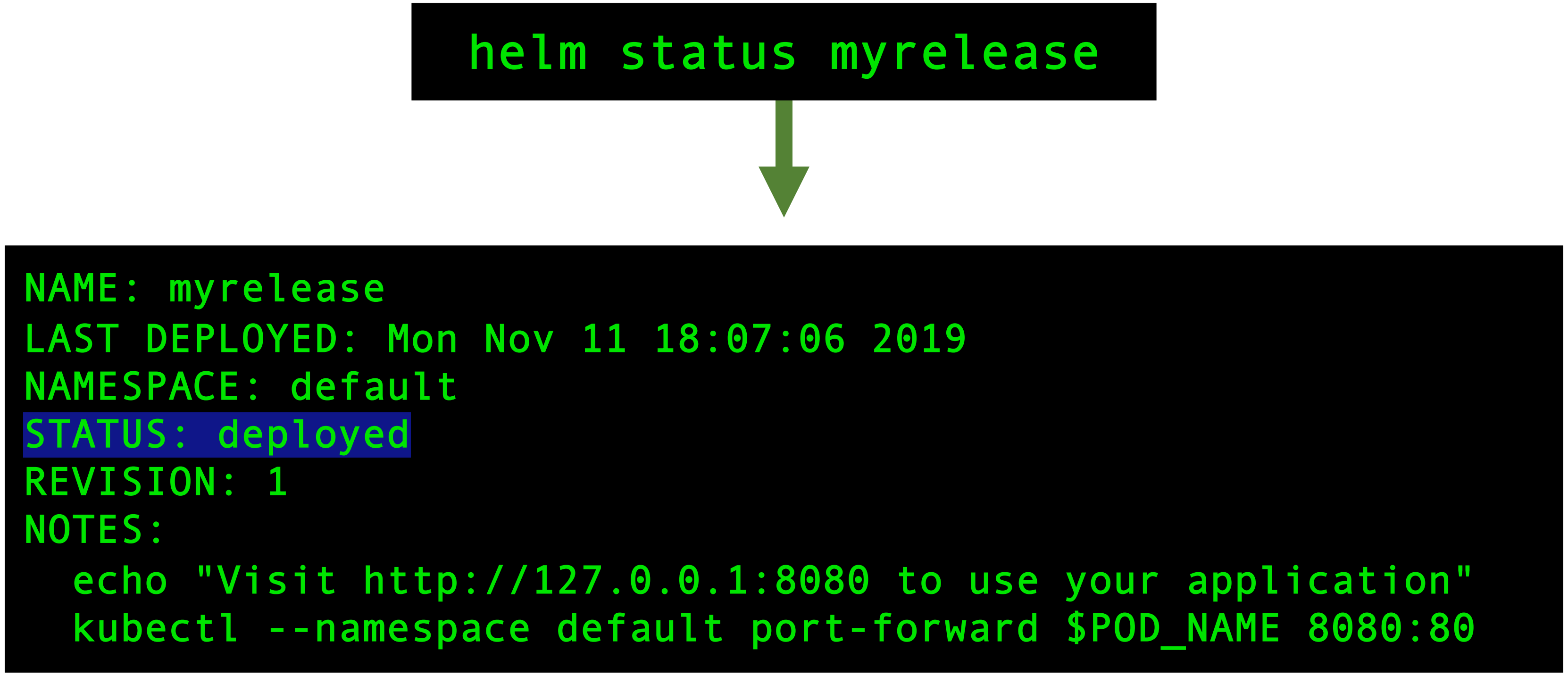
```
helm install myrelease ./myapp --set image.tag=master
```

3. Advanced usage:

```
helm install myrelease ./myapp \
  -f staging.yaml \
  -f us-east-1.yaml \
  --set tracing.enabled=true
```

Check release status

Determine the status of an individual release, check if the installation of your chart was successful.

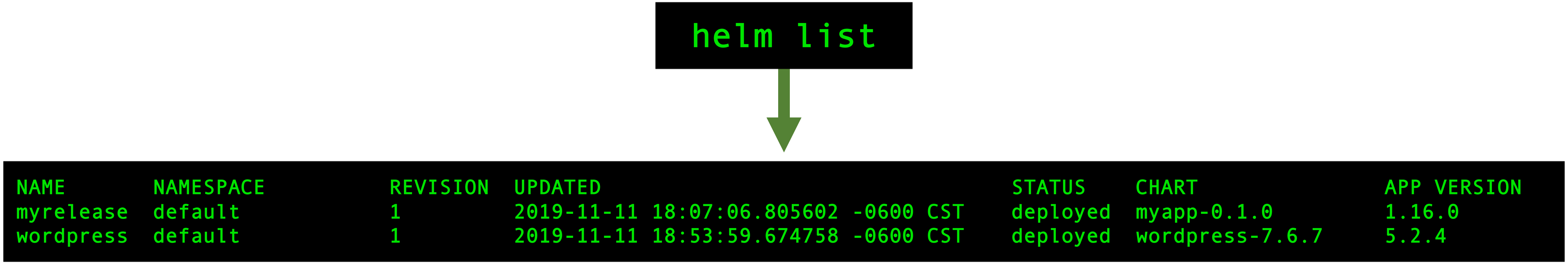


```
helm status myrelease
```

```
NAME: myrelease
LAST DEPLOYED: Mon Nov 11 18:07:06 2019
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:80
```

See what’s running

Helm has the ability to track all applications that have been installed in the cluster using Helm.

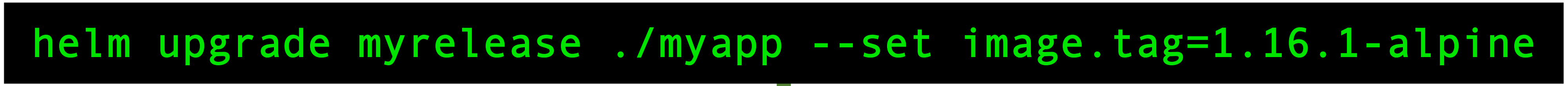


```
helm list
```

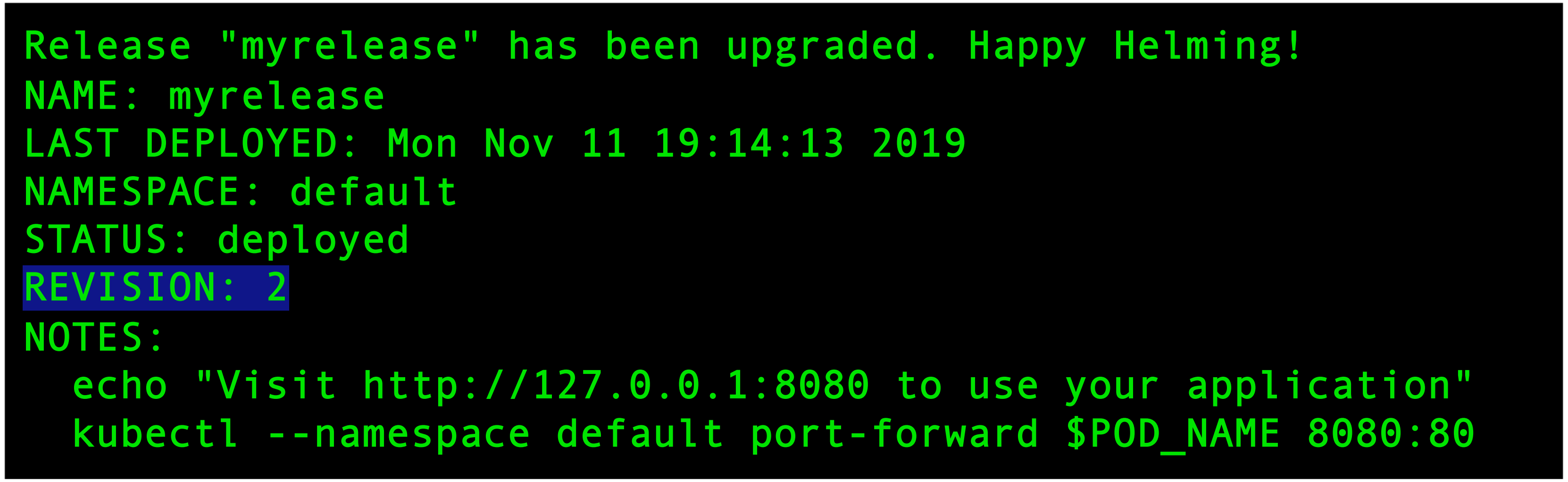
```
NAME       NAMESPACE           REVISION  UPDATED                                 STATUS    CHART               APP VERSION
myrelease  default             1         2019-11-11 18:07:06.805602 -0600 CST    deployed  myapp-0.1.0         1.16.0
wordpress  default             1         2019-11-11 18:53:59.674758 -0600 CST    deployed  wordpress-7.6.7     5.2.4
```

Upgrading a release

Create a new revision of your release by updating either the template sources or configuration values.



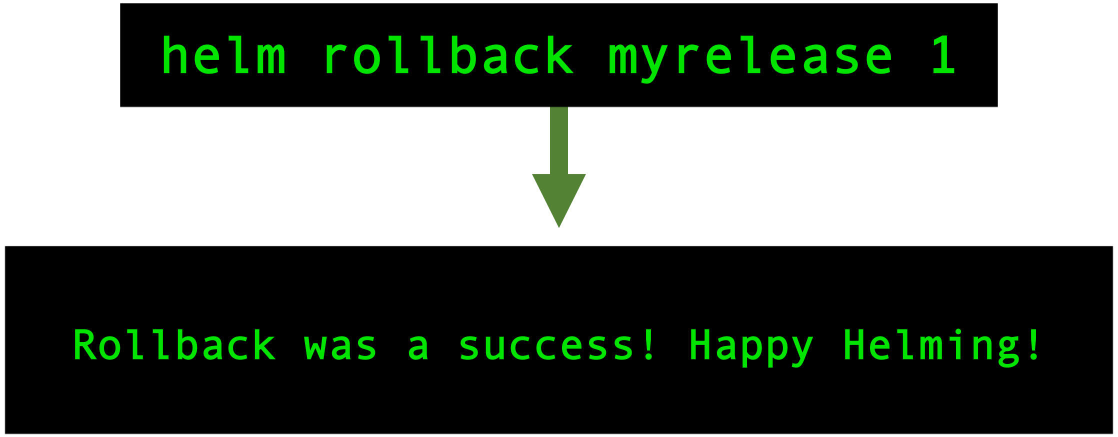
```
helm upgrade myrelease ./myapp --set image.tag=1.16.1-alpine
```



```
Release "myrelease" has been upgraded. Happy Helming!
NAME: myrelease
LAST DEPLOYED: Mon Nov 11 19:14:13 2019
NAMESPACE: default
STATUS: deployed
REVISION: 2
NOTES:
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:80
```

Rollback a release

Helm tracks every revision made on releases. When something goes wrong, revert back to a working version.

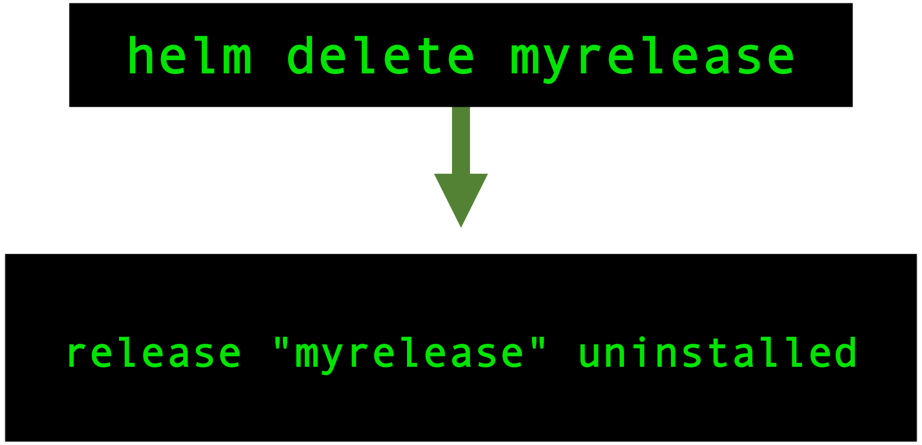


```
helm rollback myrelease 1
```

```
Rollback was a success! Happy Helming!
```

Remove a release

Remove all Kubernetes resources from the cluster that were created as part of a release.



```
helm delete myrelease
```

```
release "myrelease" uninstalled
```

Questions?