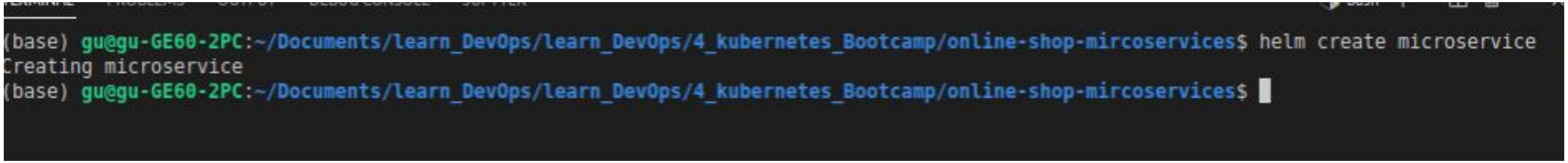
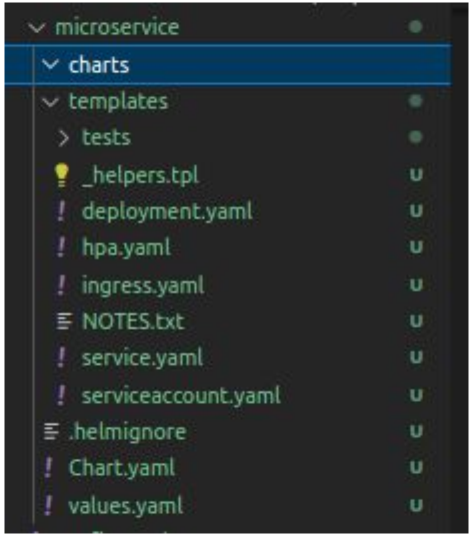
helm chart: reuseable k8s configuration
two ways creating helm chart

- helm chart for each microservice, when configuration are very different
- 1 shared helm chart for all microservice
- combine of the both options

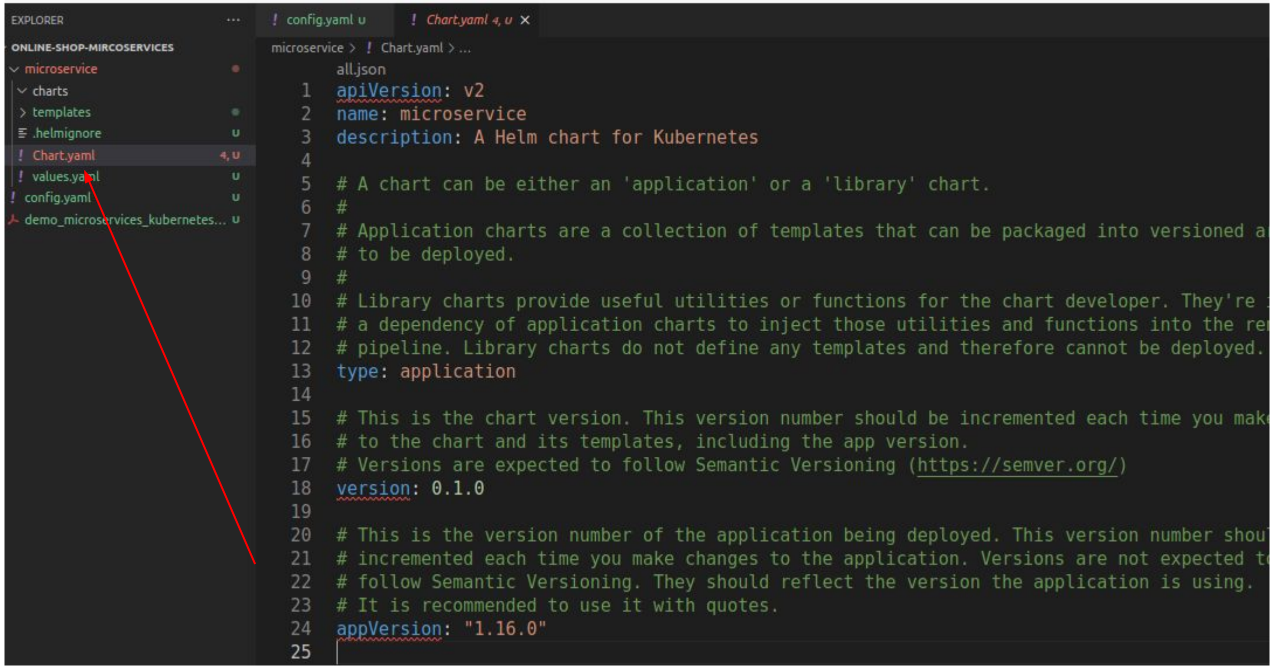


```
(base) gu@gu-GE60-2PC:~/Documents/learn_DevOps/learn_DevOps/4_kubernetes_Bootcamp/online-shop-mircoservices$ helm create microservice
Creating microservice
(base) gu@gu-GE60-2PC:~/Documents/learn_DevOps/learn_DevOps/4_kubernetes_Bootcamp/online-shop-mircoservices$
```

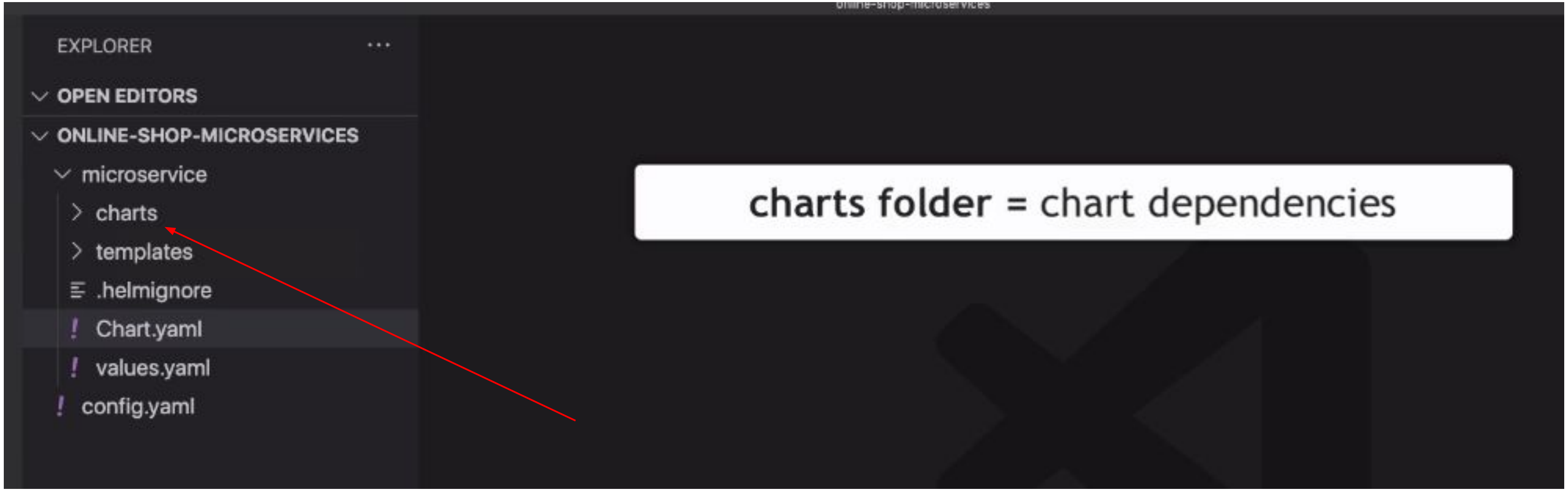
1. create a helm chart with name micrioservice



```
microservice
  charts
  templates
    tests
    _helpers.tpl            U
    deployment.yaml         U
    hpa.yaml                U
    ingress.yaml            U
    NOTES.txt               U
    service.yaml            U
    serviceaccount.yaml     U
  .helmignore               U
  Chart.yaml                U
  values.yaml               U
```



```yaml
apiVersion: v2
name: microservice
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 can be packaged into versioned a
# to be deployed.
#
# Library charts provide useful utilities or functions for the chart developer. They're i
# a dependency of application charts to inject those utilities and functions into the re
# pipeline. Library charts do not define any templates and therefore cannot be deployed.
type: application

# This is the chart version. This version number should be incremented each time you make
# to the chart and its templates, including the app version.
# Versions are expected to follow Semantic Versioning (https://semver.org/)
version: 0.1.0

# This is the version number of the application being deployed. This version number shou
# incremented each time you make changes to the application. Versions are not expected to
# follow Semantic Versioning. They should reflect the version the application is using.
# It is recommended to use it with quotes.
appVersion: "1.16.0"
```



online-shop-microservices

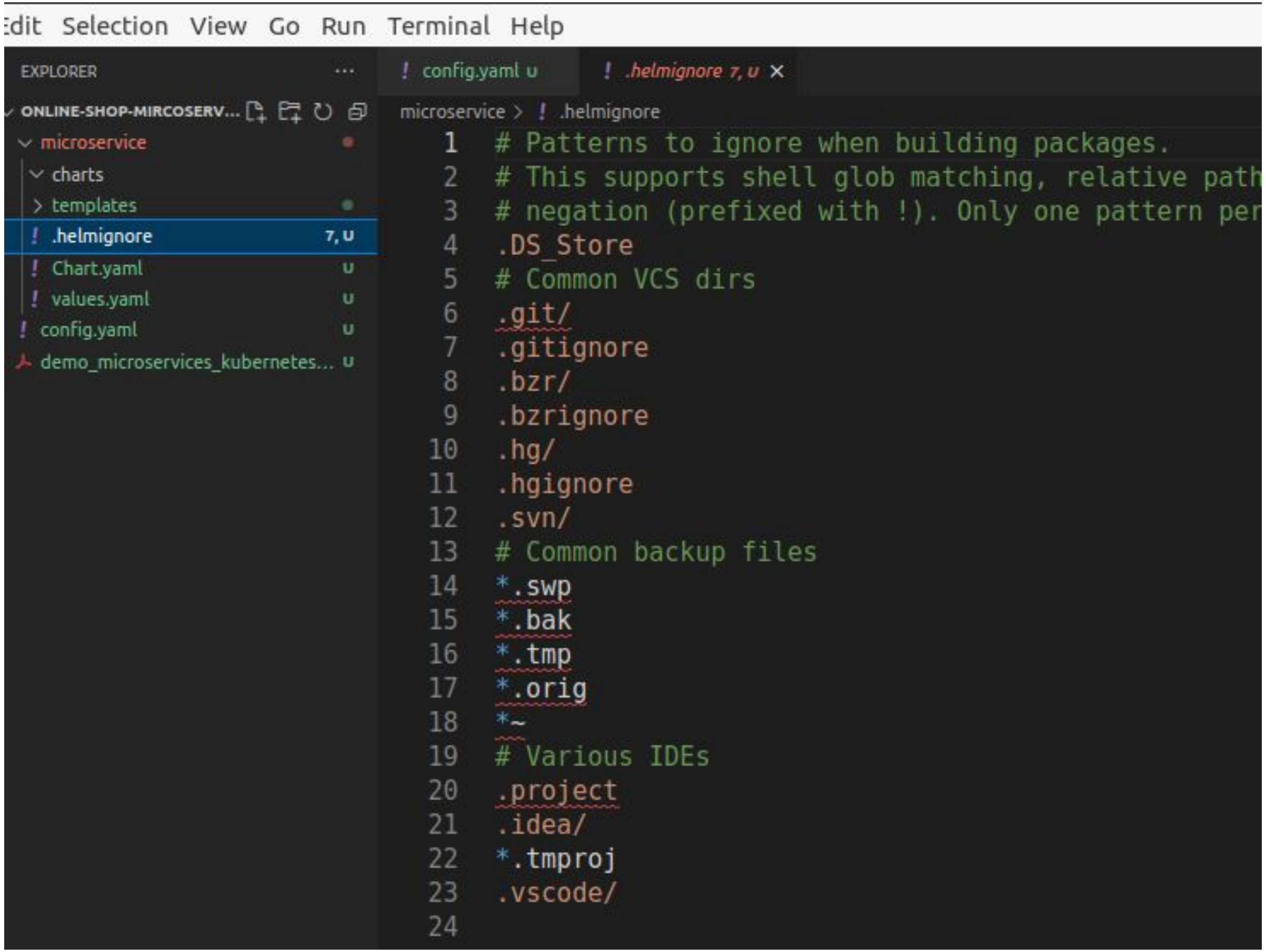
EXPLORER

OPEN EDITORS

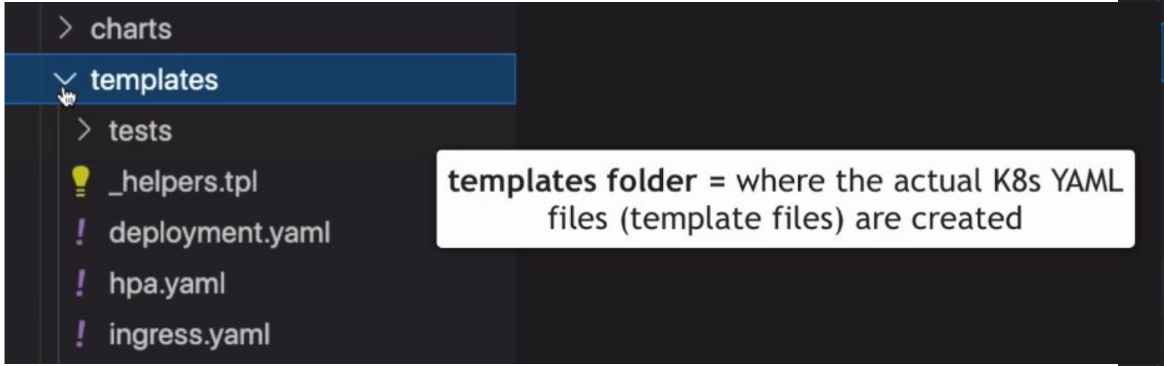
ONLINE-SHOP-MICROSERVICES

```
microservice
  charts
  templates
  .helmignore
  Chart.yaml
  values.yaml
config.yaml
```

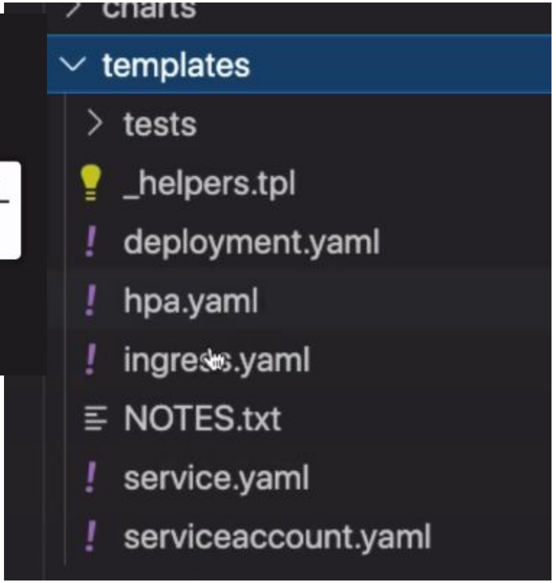
charts folder = chart dependencies



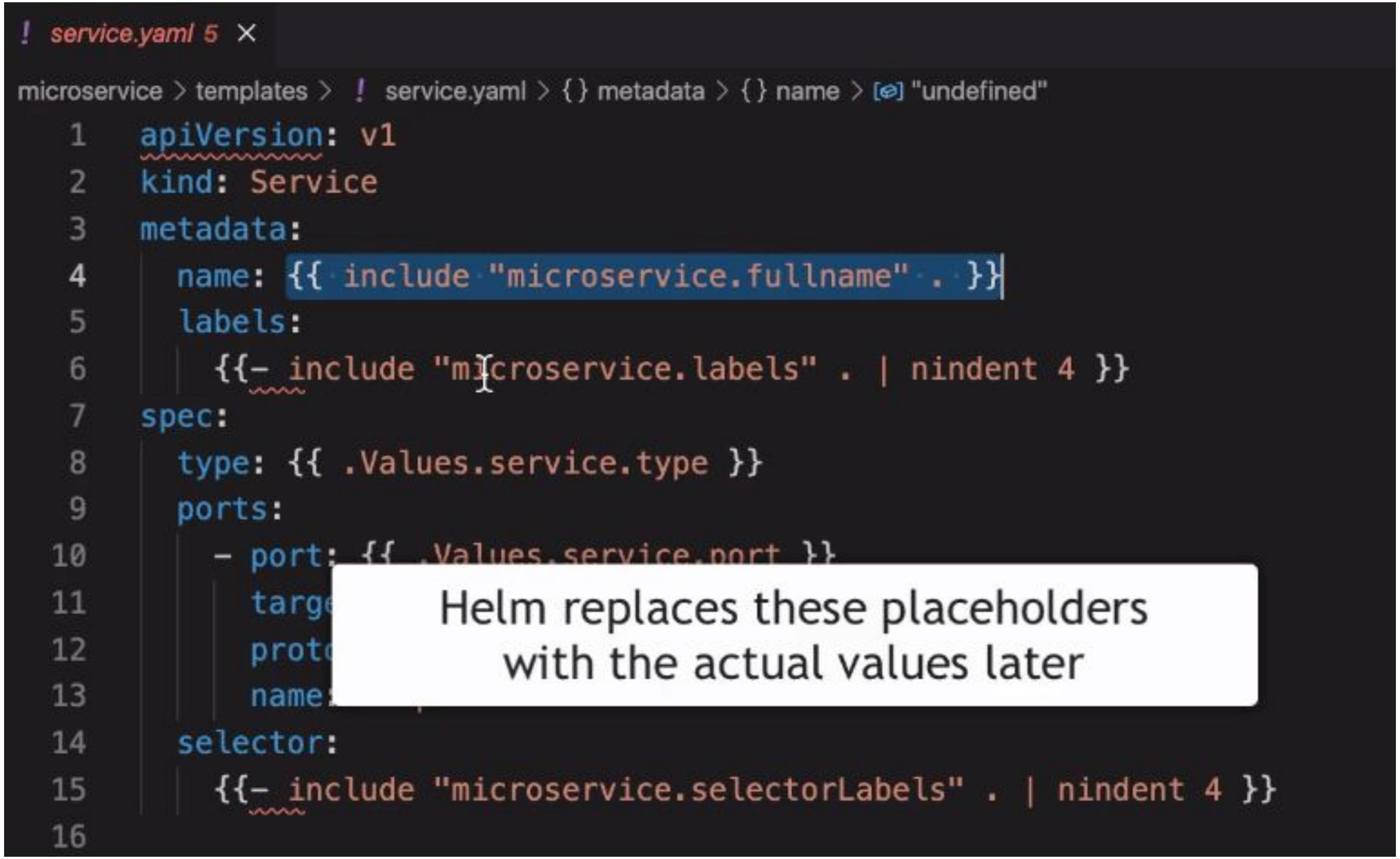
```
# Patterns to ignore when building packages.
# This supports shell glob matching, relative path
# negation (prefixed with !). Only one pattern per
.DS_Store
# Common VCS dirs
.git/
.gitignore
.bzr/
.bzrignore
.hg/
.hgignore
.svn/
# Common backup files
*.swp
*.bak
*.tmp
*.orig
*~
# Various IDEs
.project
.idea/
*.tmproj
.vscode/

```



templates folder = where the actual K8s YAML files (template files) are created

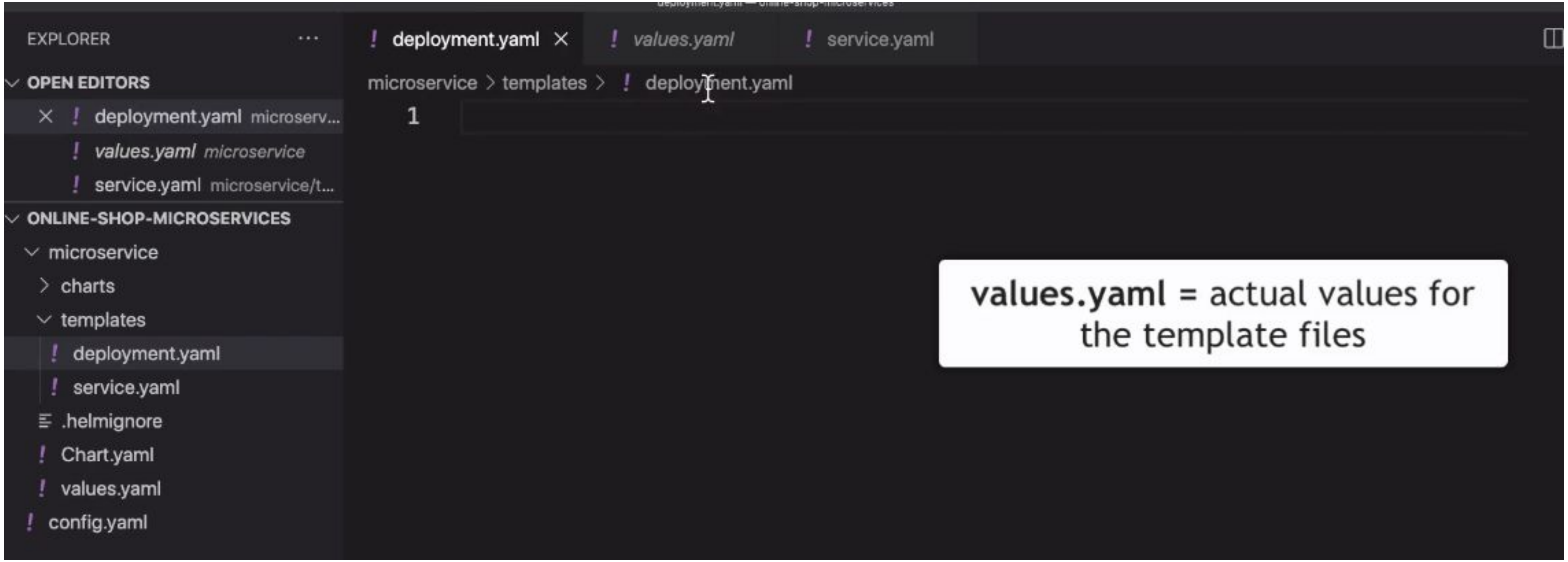


create blueprint of yaml file


! service.yaml 5 ×
microservice > templates > ! service.yaml > {} metadata > {} name > "undefined"

```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ include "microservice.fullname" . }}
  labels:
    {{- include "microservice.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targe
      proto
      name:
  selector:
    {{- include "microservice.selectorLabels" . | nindent 4 }}

```

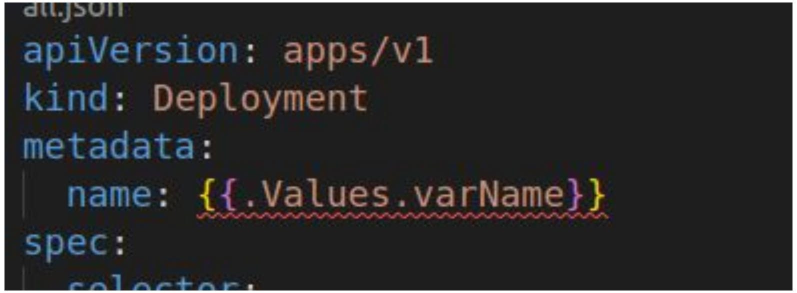
Helm replaces these placeholders with the actual values later



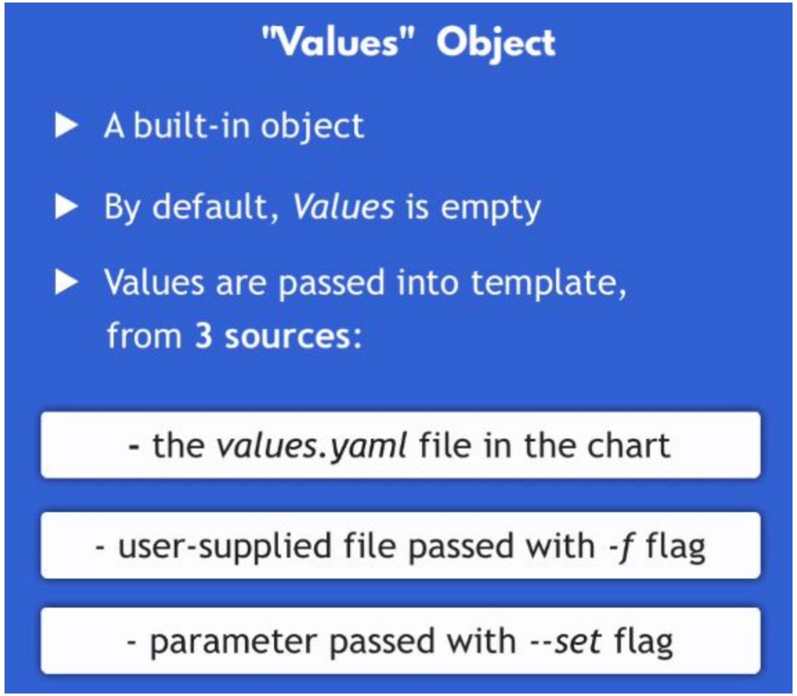
values.yaml = actual values for the template files

Create Microservices Helm Chart

Create Basic Template File



```
all.json
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{.Values.varName}}
spec:
  selector:
```



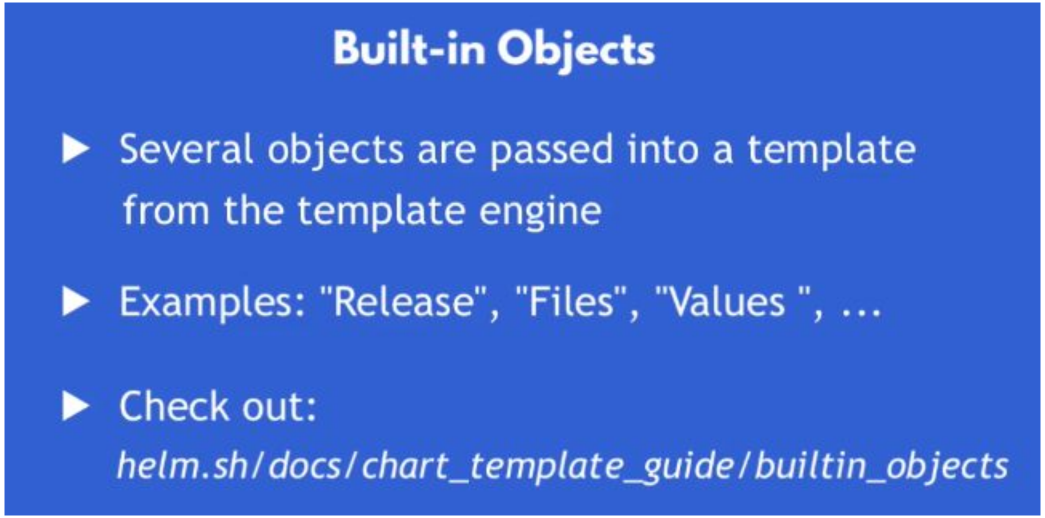
"Values" Object

- A built-in object
- By default, *Values* is empty
- Values are passed into template, from 3 sources:

- the *values.yaml* file in the chart

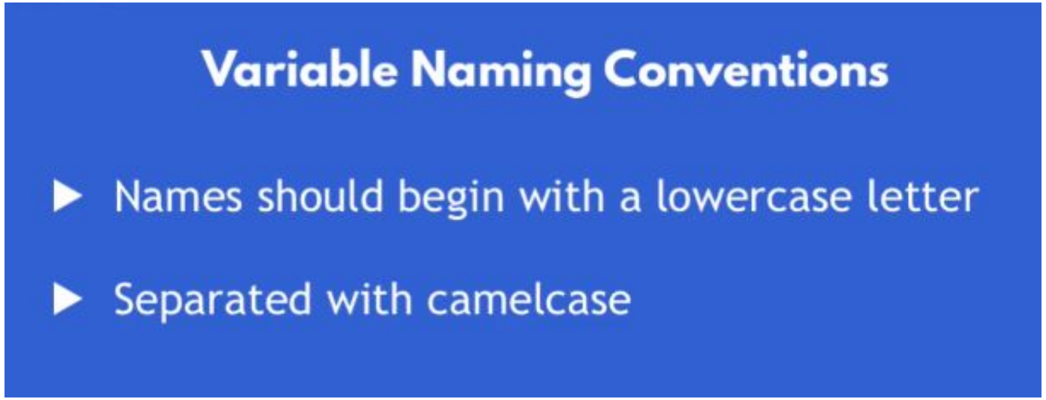
- user-supplied file passed with *-f* flag

- parameter passed with --*set* flag



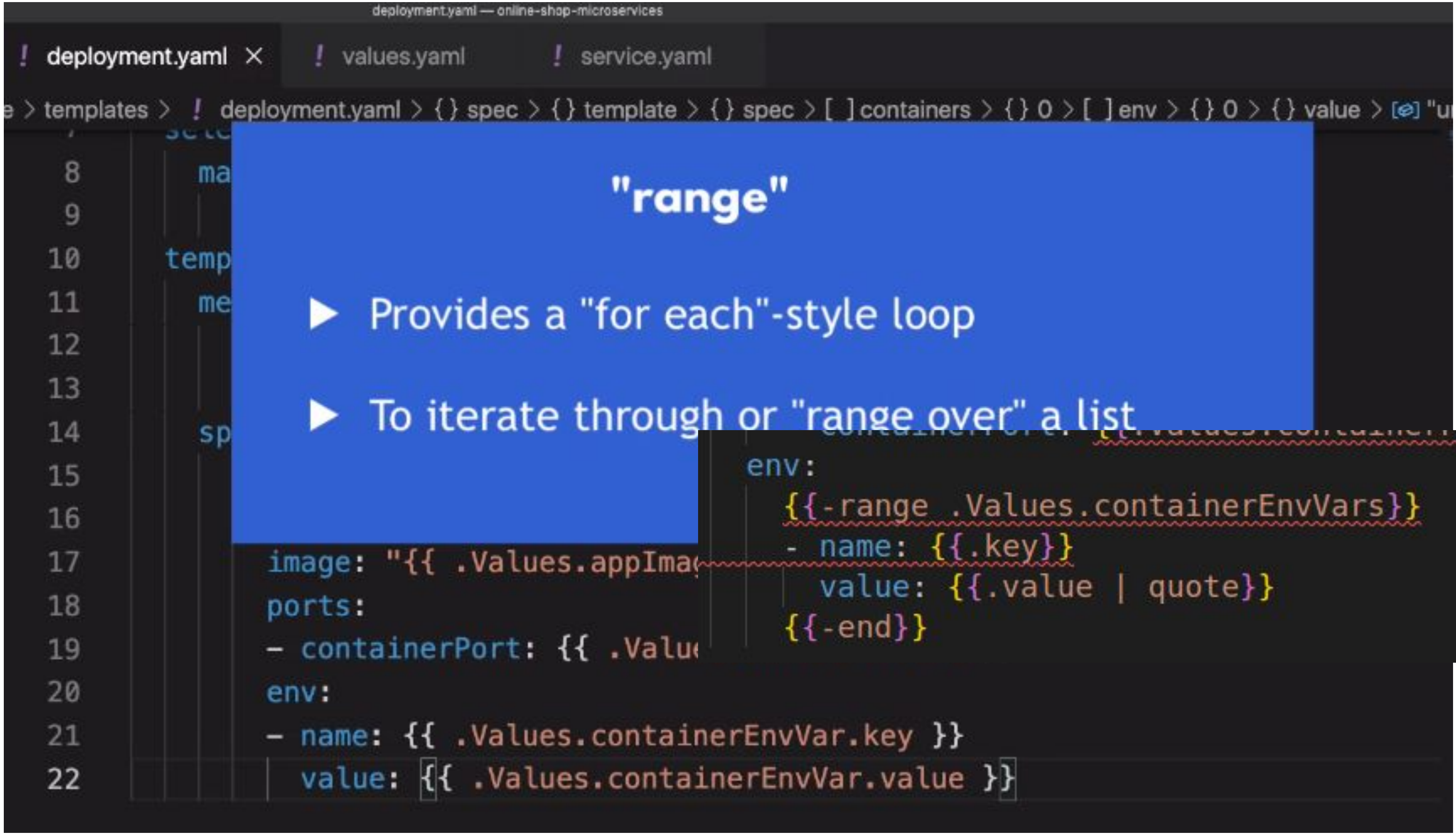
Built-in Objects

- Several objects are passed into a template from the template engine
- Examples: "Release", "Files", "Values ", ...
- Check out: *helm.sh/docs/chart_template_guide/builtin_objects*



Variable Naming Conventions

- Names should begin with a lowercase letter
- Separated with camelcase


deployment.yaml — online-shop-microservices

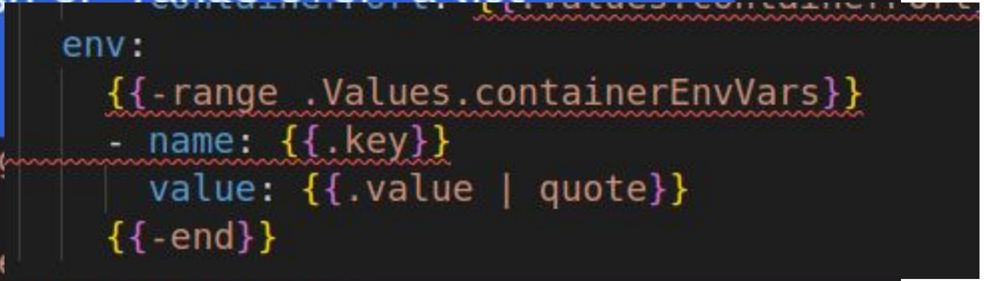
deployment.yaml × | values.yaml | service.yaml

e > templates > deployment.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > [ ] env > {} 0 > {} value > "u

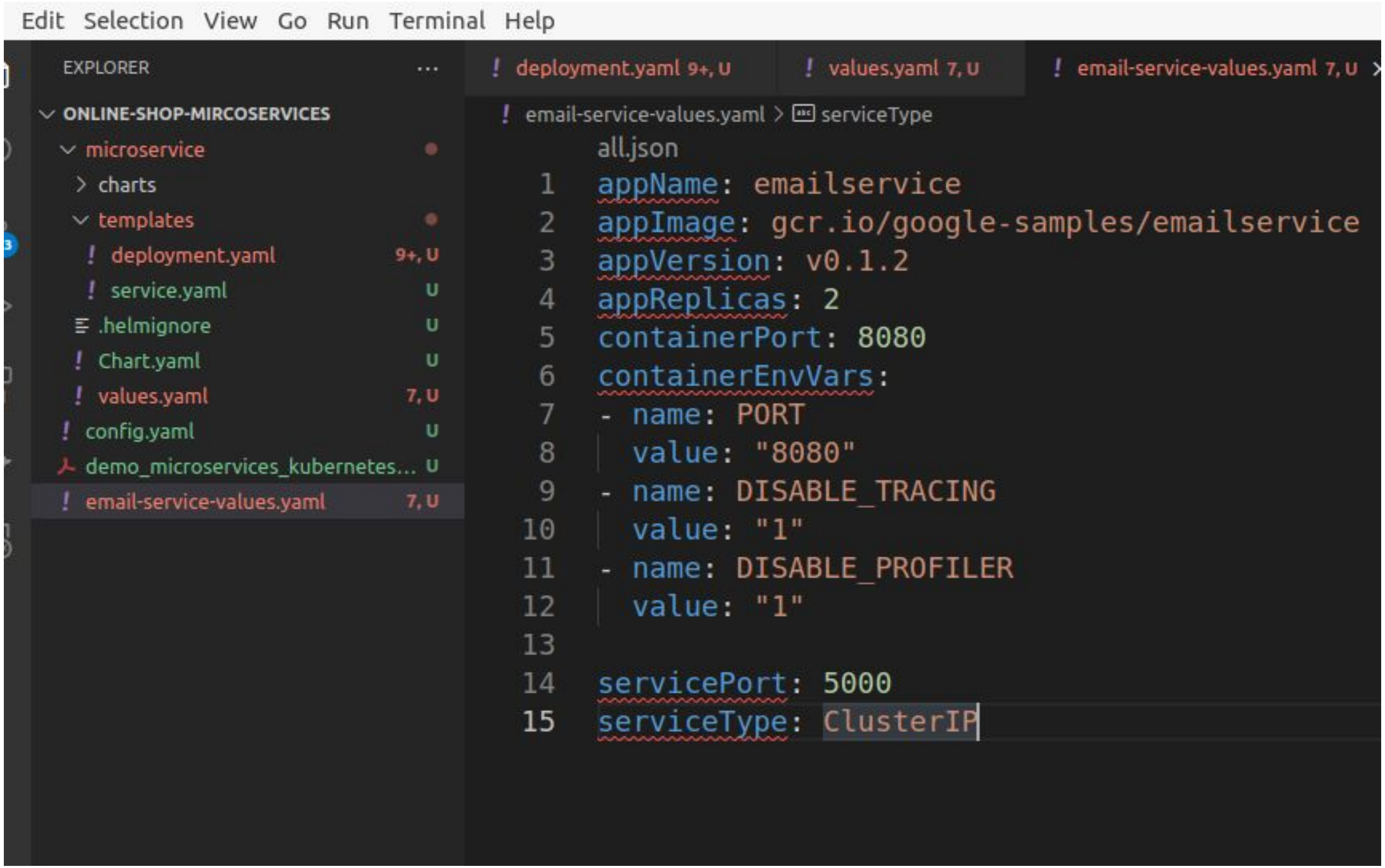
```
    ma

  temp
    me

    sp


        image: "{{ .Values.appImag
        ports:
        - containerPort: {{ .Valu
        env:
        - name: {{ .Values.containerEnvVar.key }}
          value: {{ .Values.containerEnvVar.value }}
```

"range"

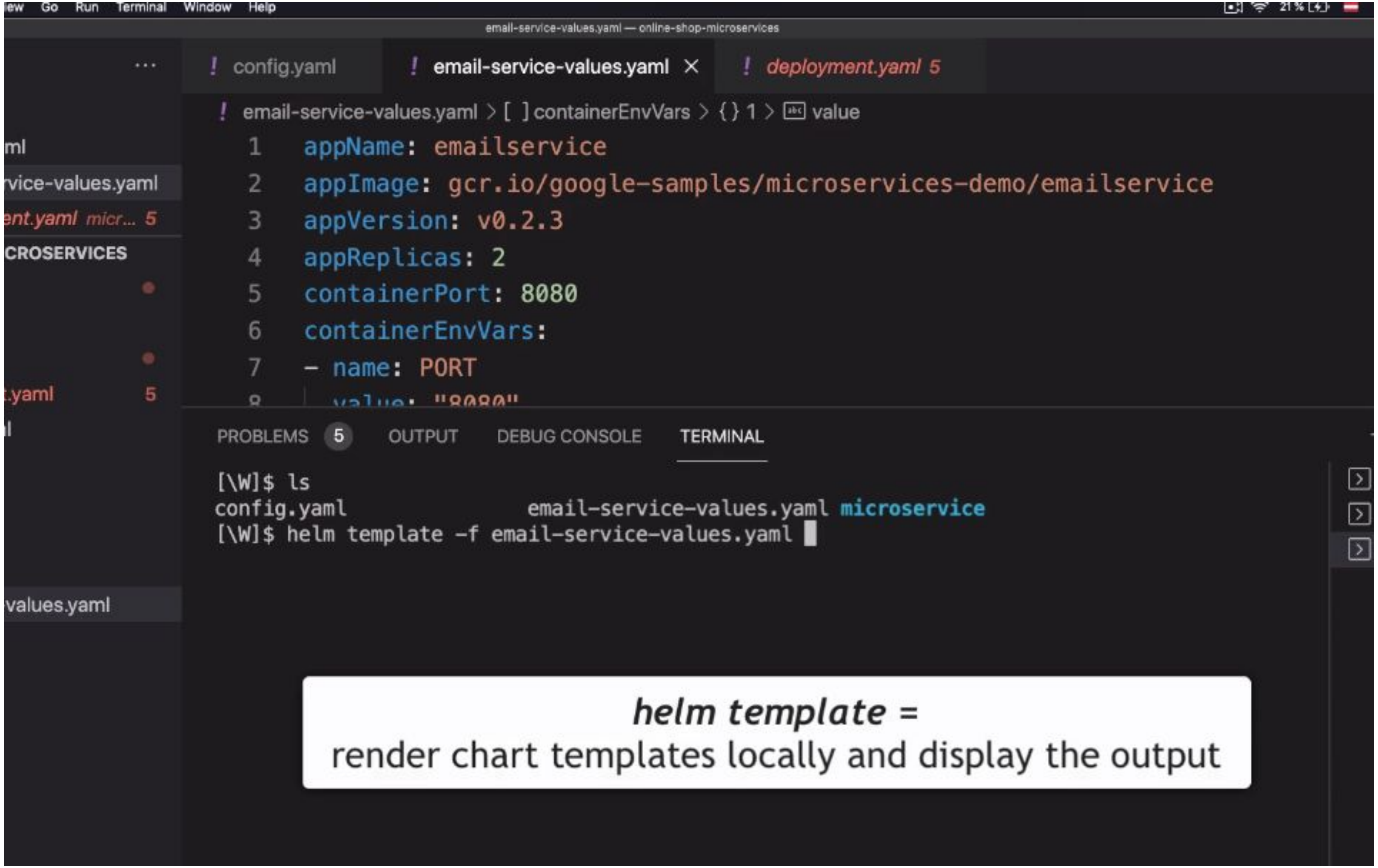
- Provides a "for each"-style loop
- To iterate through or "range over" a list



```
env:
  {{-range .Values.containerEnvVars}}
  - name: {{.key}}
    value: {{.value | quote}}
  {{-end}}
```


Edit Selection View Go Run Terminal Help
EXPLORER
ONLINE-SHOP-MIRCOSERVICES
microservice
charts
templates
deployment.yaml 9+, U
service.yaml U
.helmignore U
Chart.yaml U
values.yaml 7, U
config.yaml U
demo_microservices_kubernetes... U
email-service-values.yaml 7, U
deployment.yaml 9+, U
values.yaml 7, U
email-service-values.yaml 7, U
email-service-values.yaml > serviceType
all.json

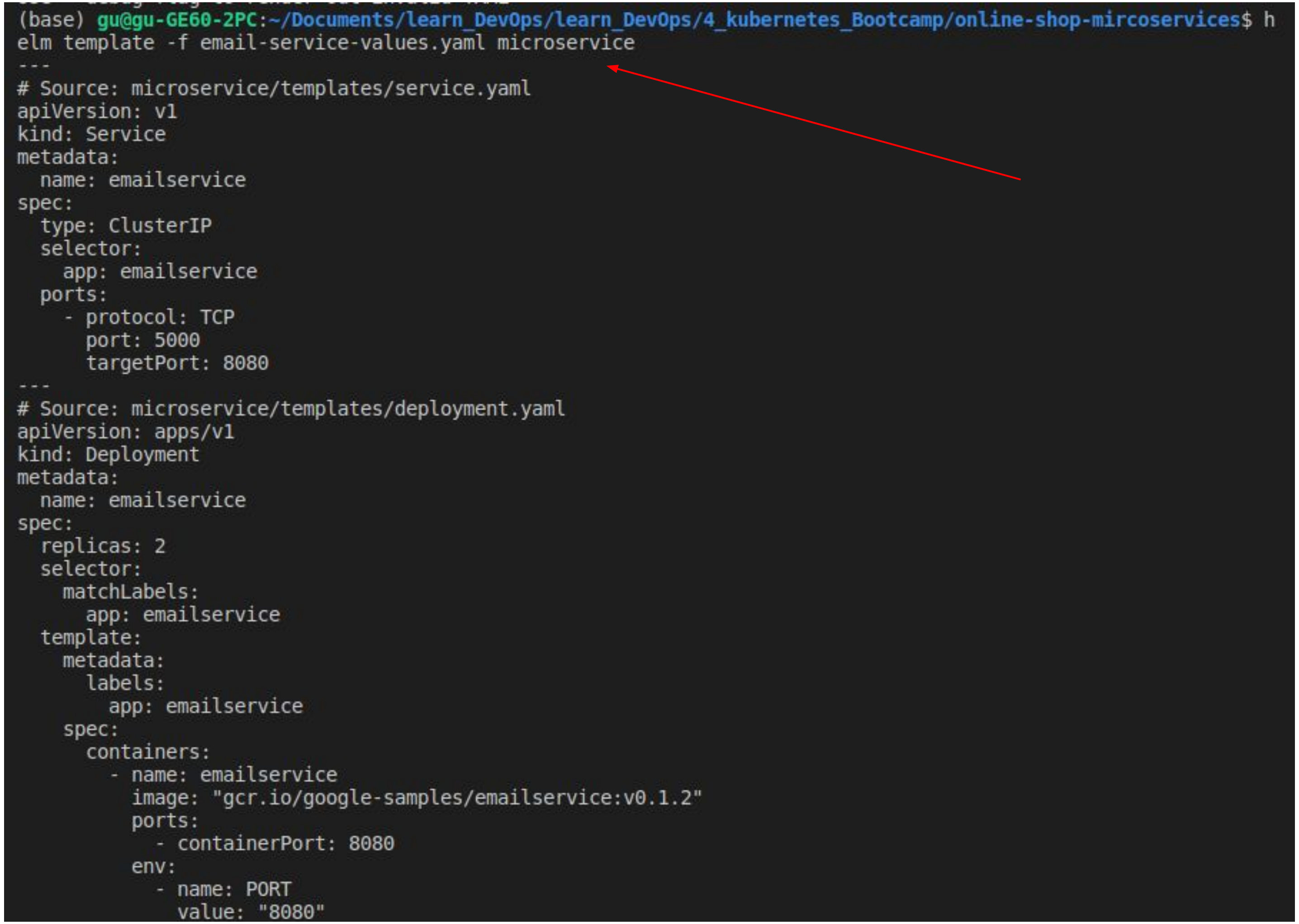
```yaml
appName: emailservice
appImage: gcr.io/google-samples/emailservice
appVersion: v0.1.2
appReplicas: 2
containerPort: 8080
containerEnvVars:
- name: PORT
  value: "8080"
- name: DISABLE_TRACING
  value: "1"
- name: DISABLE_PROFILER
  value: "1"

servicePort: 5000
serviceType: ClusterIP
```


ew Go Run Terminal Window Help
email-service-values.yaml — online-shop-microservices
config.yaml
email-service-values.yaml
deployment.yaml 5
email-service-values.yaml > [ ] containerEnvVars > { } 1 > value
ml
vice-values.yaml
ent.yaml micr... 5
CROSERVICES
.yaml 5
values.yaml
PROBLEMS 5
OUTPUT
DEBUG CONSOLE
TERMINAL

```yaml
appName: emailservice
appImage: gcr.io/google-samples/microservices-demo/emailservice
appVersion: v0.2.3
appReplicas: 2
containerPort: 8080
containerEnvVars:
- name: PORT
  value: "8080"
```

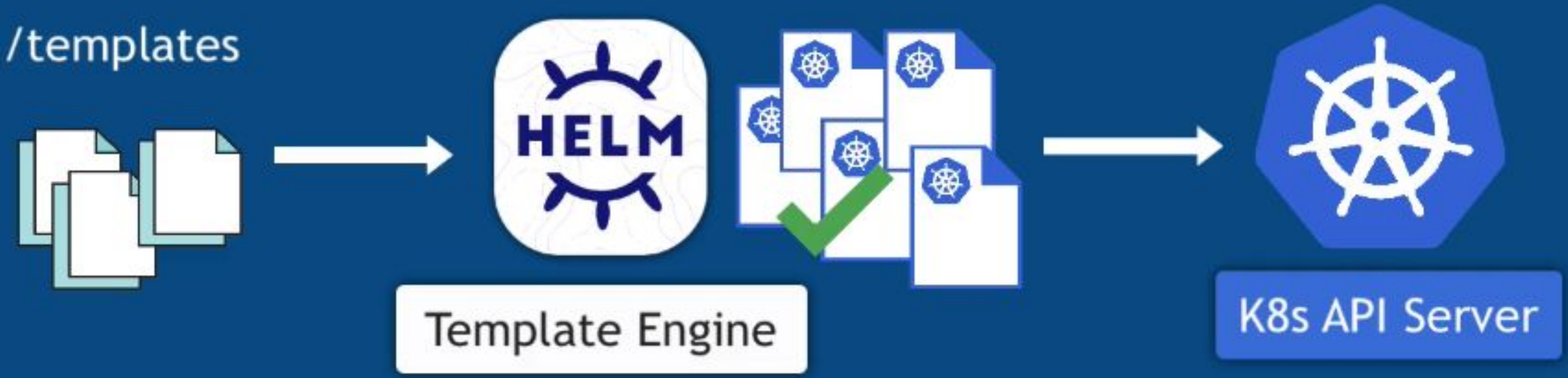
```
[\W]$ ls
config.yaml                 email-service-values.yaml microservice
[\W]$ helm template -f email-service-values.yaml
```

helm template **=**
render chart templates locally and display the output

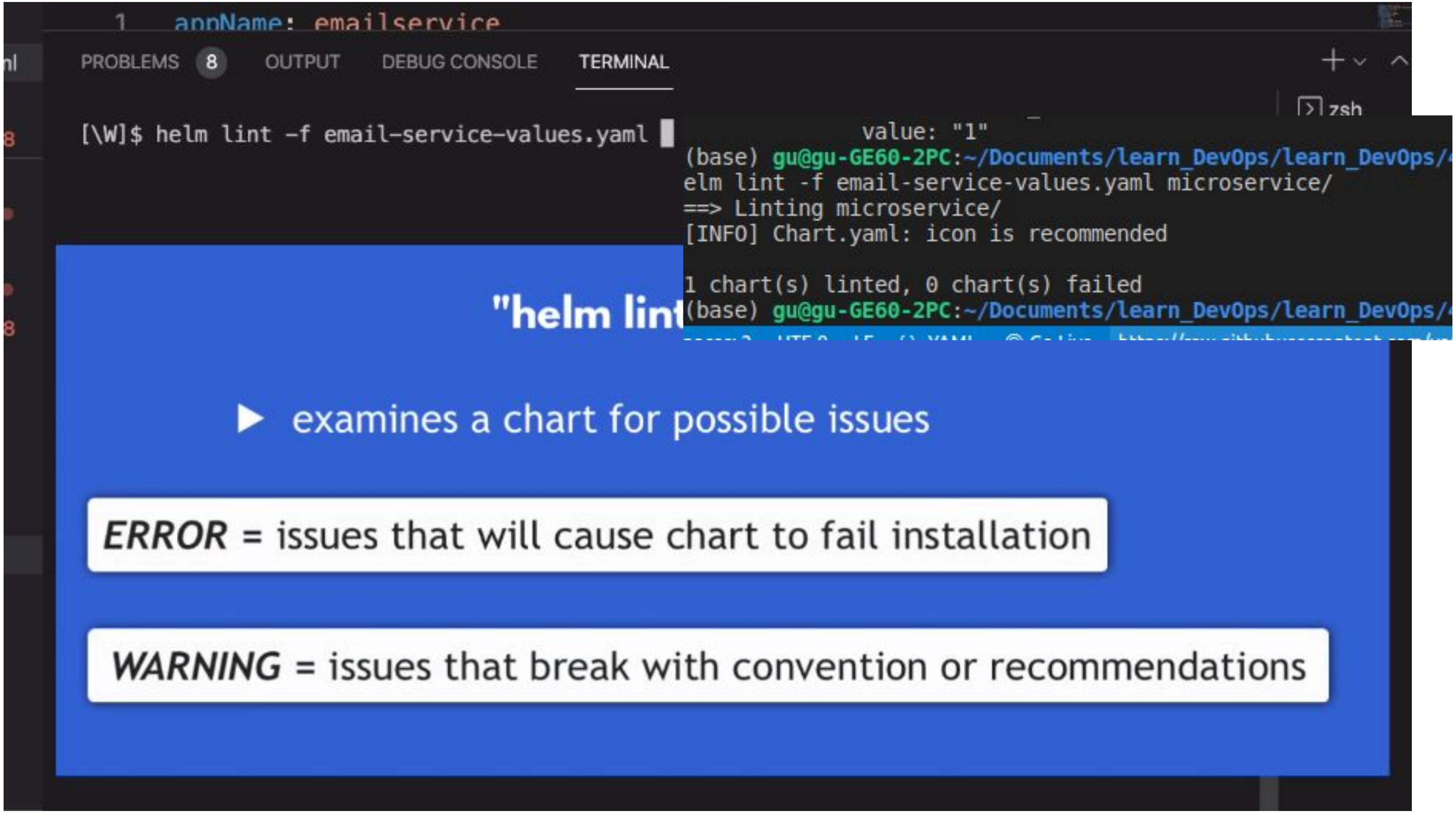


```
(base) gu@gu-GE60-2PC:~/Documents/learn_DevOps/learn_DevOps/4_kubernetes_Bootcamp/online-shop-mircoservices$ h
elm template -f email-service-values.yaml microservice
---
# Source: microservice/templates/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: emailservice
spec:
  type: ClusterIP
  selector:
    app: emailservice
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 8080
---
# Source: microservice/templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: emailservice
spec:
  replicas: 2
  selector:
    matchLabels:
      app: emailservice
  template:
    metadata:
      labels:
        app: emailservice
    spec:
      containers:
        - name: emailservice
          image: "gcr.io/google-samples/emailservice:v0.1.2"
          ports:
            - containerPort: 8080
          env:
            - name: PORT
              value: "8080"
```

Helm Rendering Process



- Engine replaces the variables with the actual values (from the 3 different sources)



```
appName: emailservice
```

```
[\W]$ helm lint -f email-service-values.yaml
```

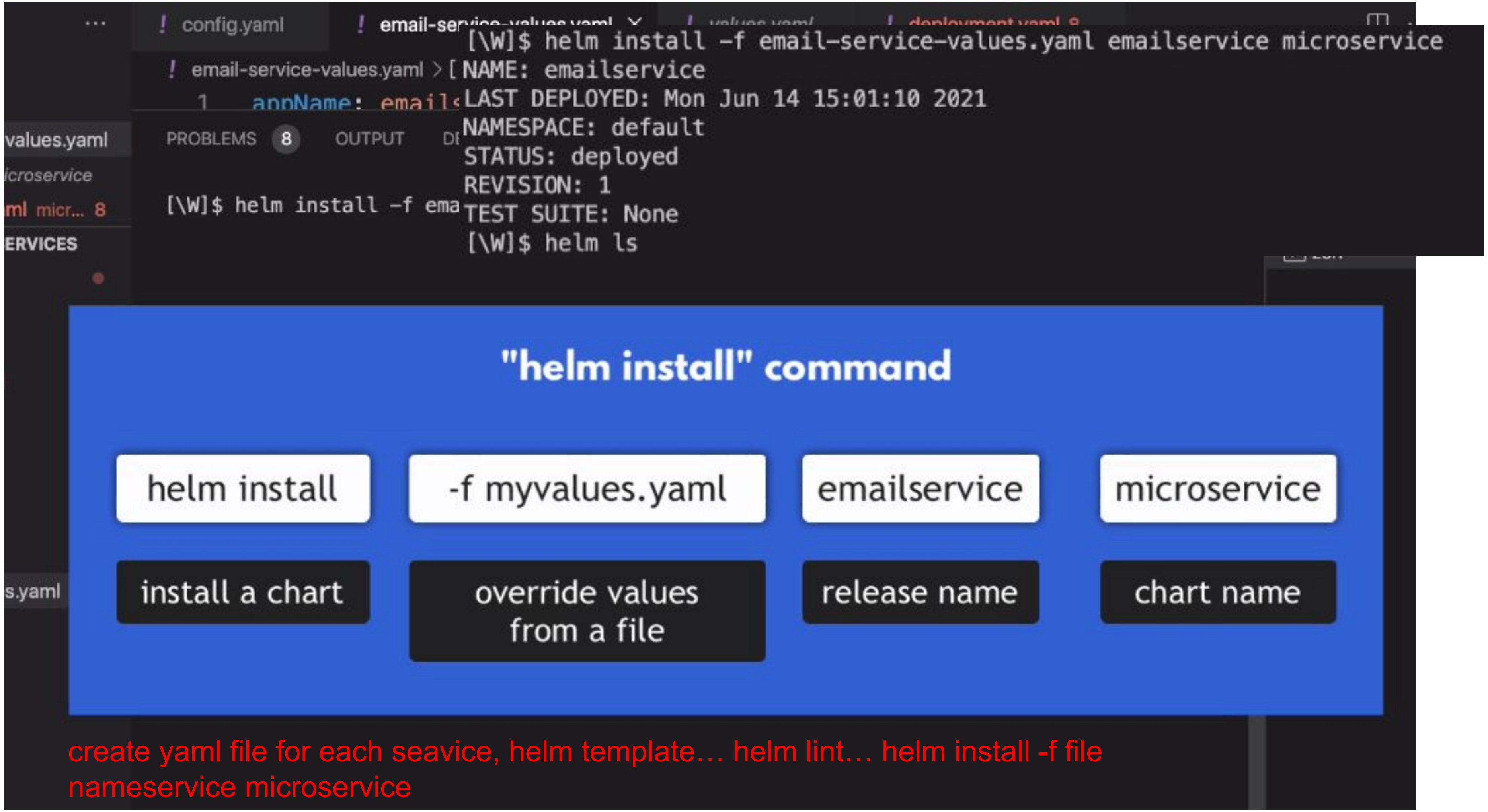
```
                  value: "1"
(base) gu@gu-GE60-2PC:~/Documents/learn_DevOps/learn_DevOps/
elm lint -f email-service-values.yaml microservice/
==> Linting microservice/
[INFO] Chart.yaml: icon is recommended

1 chart(s) linted, 0 chart(s) failed
(base) gu@gu-GE60-2PC:~/Documents/learn_DevOps/learn_DevOps/
```

"helm lint

▶ examines a chart for possible issues

ERROR = issues that will cause chart to fail installation

WARNING = issues that break with convention or recommendations



```
1    appName: email
```

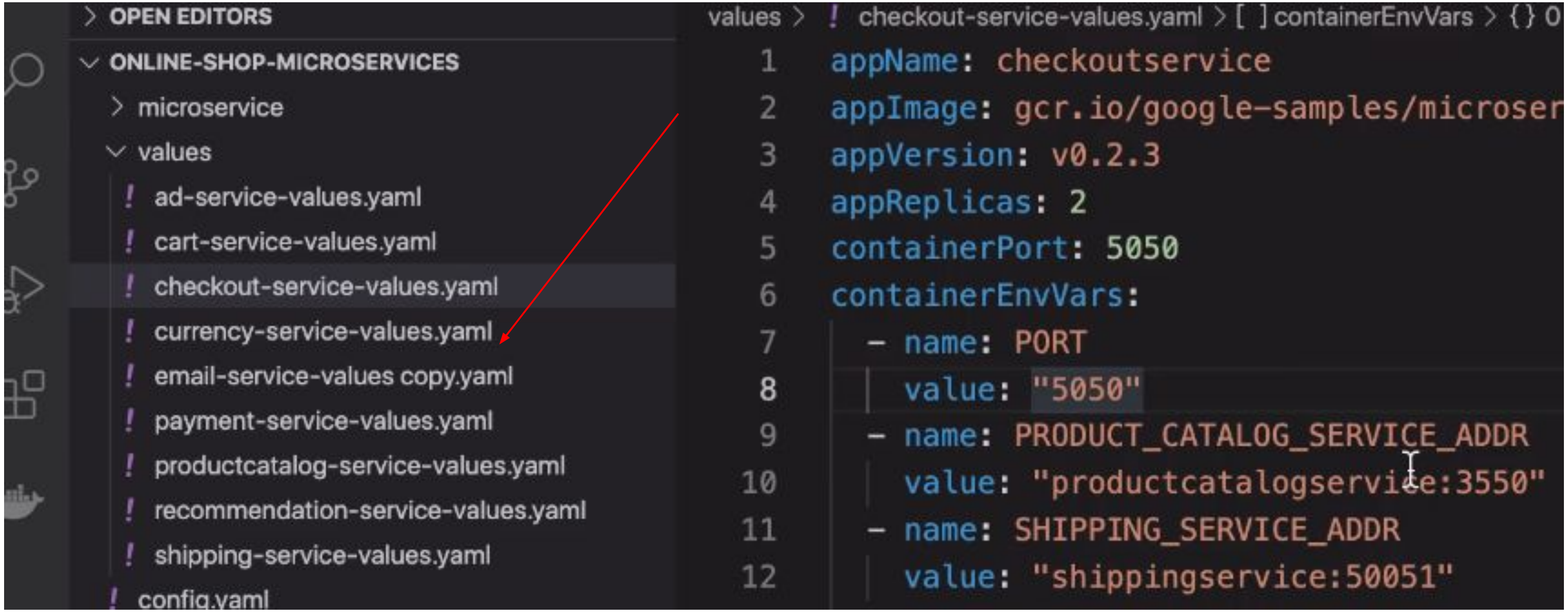
```
[\W]$ helm install -f ema
```

```
[\W]$ helm install -f email-service-values.yaml emailservice microservice
NAME: emailservice
LAST DEPLOYED: Mon Jun 14 15:01:10 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
[\W]$ helm ls
```

"helm install" command

helm install	-f myvalues.yaml	emailservice	microservice
install a chart	override values from a file	release name	chart name

create yaml file for each seavice, helm template… helm lint… helm install -f file nameservice microservice


OPEN EDITORS
ONLINE-SHOP-MICROSERVICES
microservice
values
ad-service-values.yaml
cart-service-values.yaml
checkout-service-values.yaml
currency-service-values.yaml
email-service-values copy.yaml
payment-service-values.yaml
productcatalog-service-values.yaml
recommendation-service-values.yaml
shipping-service-values.yaml
config.yaml
values > checkout-service-values.yaml > [ ] containerEnvVars > {} 0

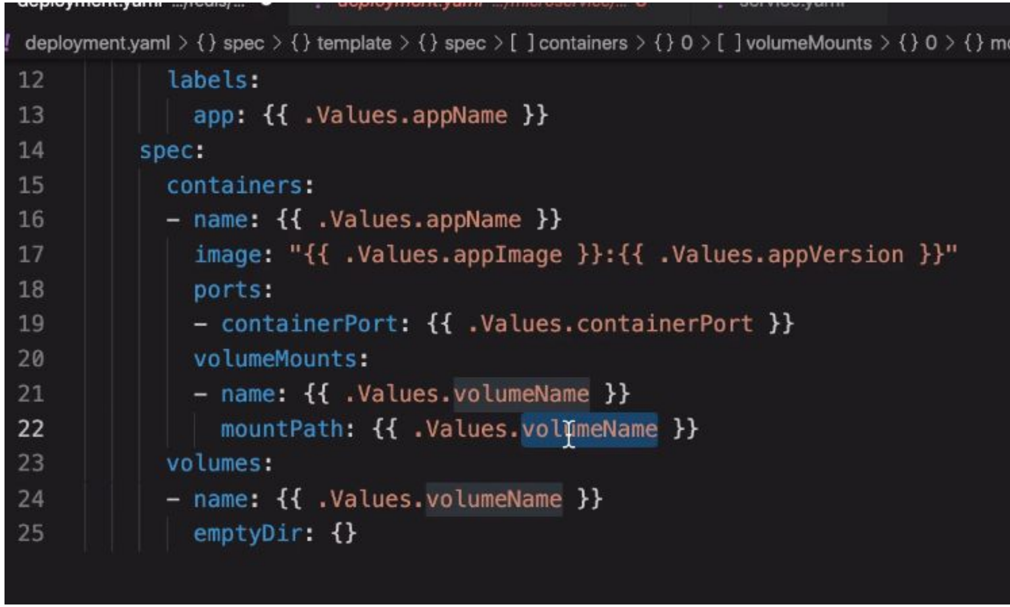
```yaml
appName: checkoutservice
appImage: gcr.io/google-samples/microser
appVersion: v0.2.3
appReplicas: 2
containerPort: 5050
containerEnvVars:
  - name: PORT
    value: "5050"
  - name: PRODUCT_CATALOG_SERVICE_ADDR
    value: "productcatalogservice:3550"
  - name: SHIPPING_SERVICE_ADDR
    value: "shippingservice:50051"
```

put in the values folder for all services

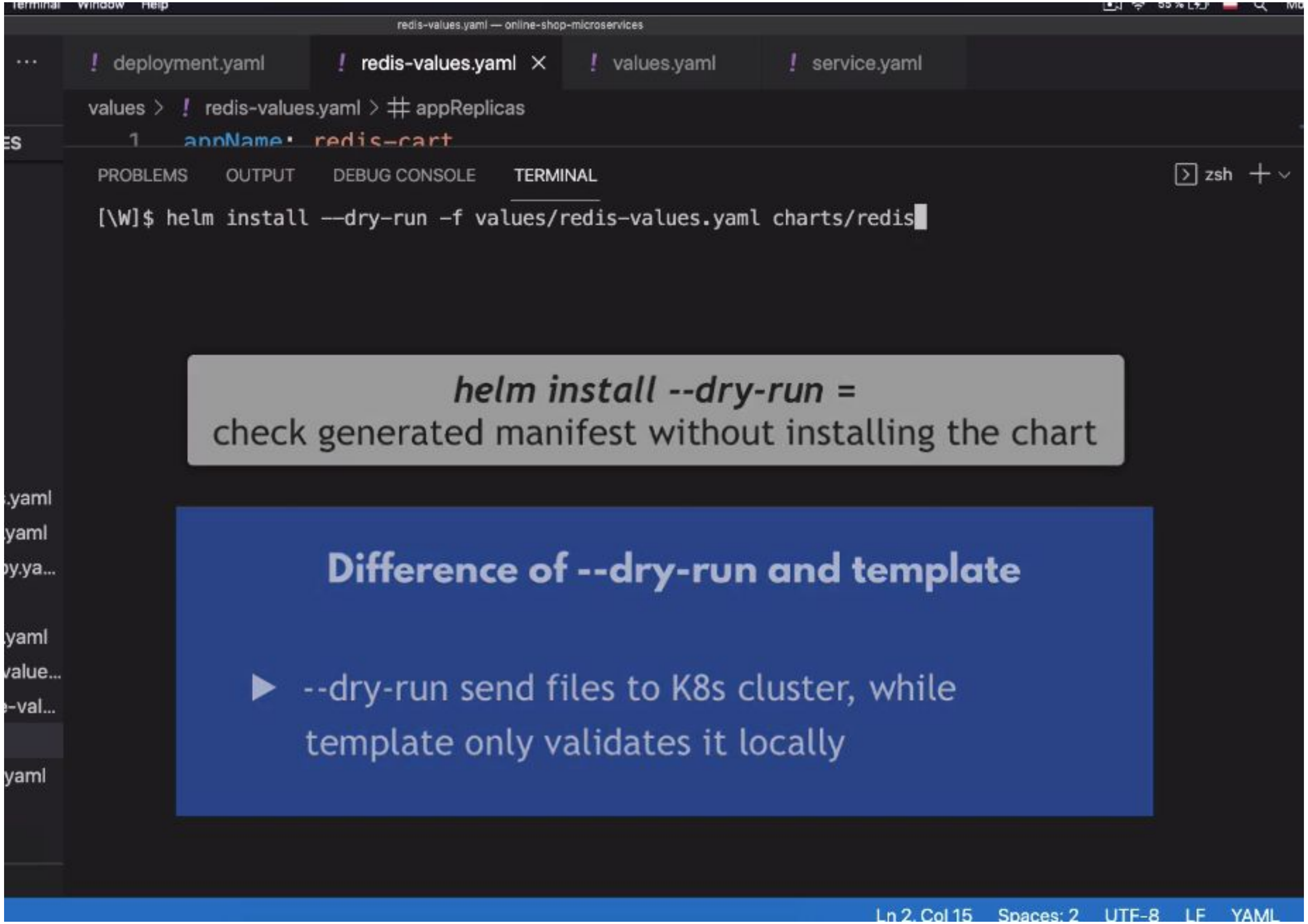
create a redis chart
put microservice and chart together into a chart folder

- helm create redis
- create
    - deployment.yaml
    - service.yaml

replace alll redis to {{.Value.appName}} ……
create a redis-value


deployment.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > [ ] volumeMounts > {} 0 > {} m

```yaml
      labels:
        app: {{ .Values.appName }}
    spec:
      containers:
      - name: {{ .Values.appName }}
        image: "{{ .Values.appImage }}:{{ .Values.appVersion }}"
        ports:
        - containerPort: {{ .Values.containerPort }}
        volumeMounts:
        - name: {{ .Values.volumeName }}
          mountPath: {{ .Values.volumeName }}
      volumes:
      - name: {{ .Values.volumeName }}
        emptyDir: {}
```



```
1  appName: redis-cart
```

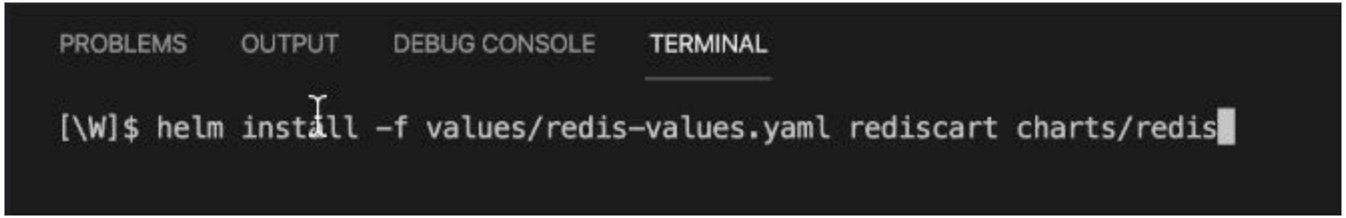
```
[\W]$ helm install --dry-run -f values/redis-values.yaml charts/redis
```

helm install --dry-run =
check generated manifest without installing the chart

Difference of --dry-run and template

- --dry-run send files to K8s cluster, while template only validates it locally

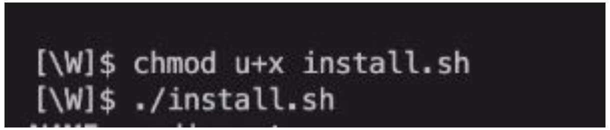
2 ways deploy k8s cluster

- helm command one by one or in script file
- 

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

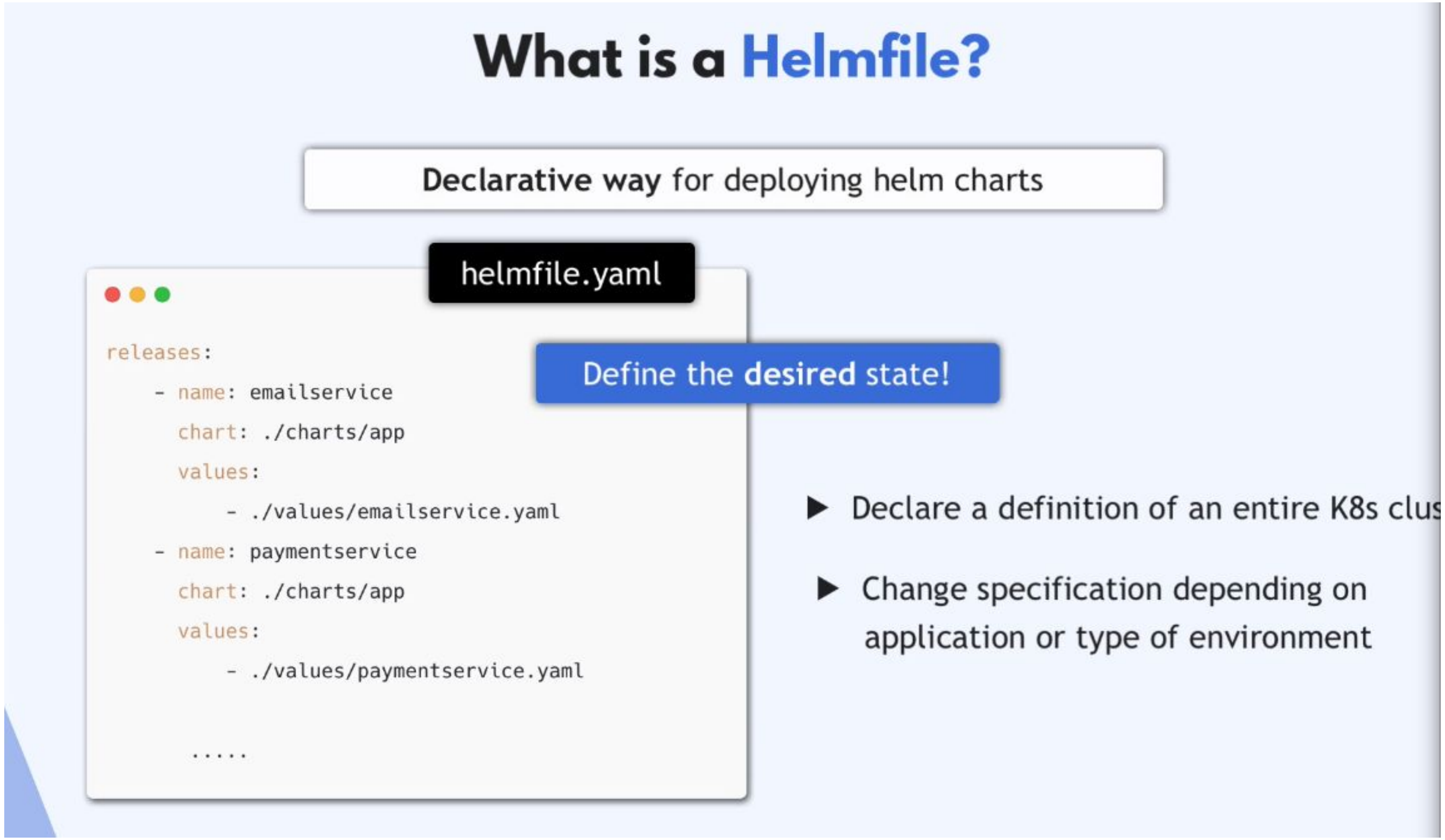
```
[\W]$ helm install -f values/redis-values.yaml rediscart charts/redis
```

change mode
./install.sh



```
[\W]$ chmod u+x install.sh
[\W]$ ./install.sh
```

2. method helmfile



What is a Helmfile?

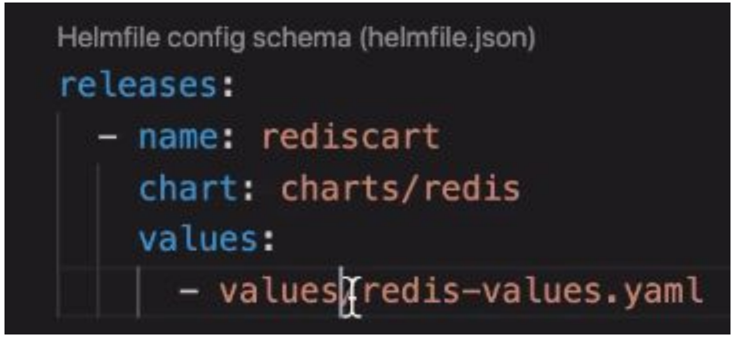
Declarative way for deploying helm charts

helmfile.yaml

```
releases:
    - name: emailservice
      chart: ./charts/app
      values:
          - ./values/emailservice.yaml
    - name: paymentservice
      chart: ./charts/app
      values:
          - ./values/paymentservice.yaml

   .....
```

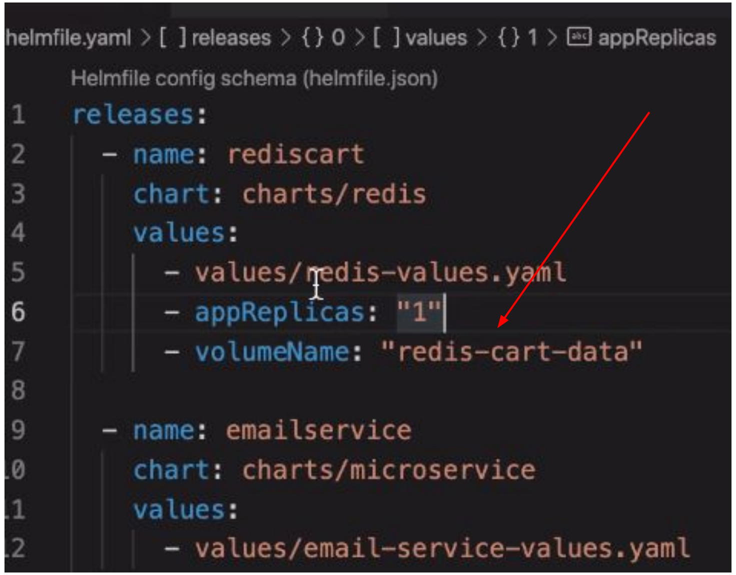
Define the **desired** state!

- Declare a definition of an entire K8s clus
- Change specification depending on application or type of environment



Helmfile config schema (helmfile.json)

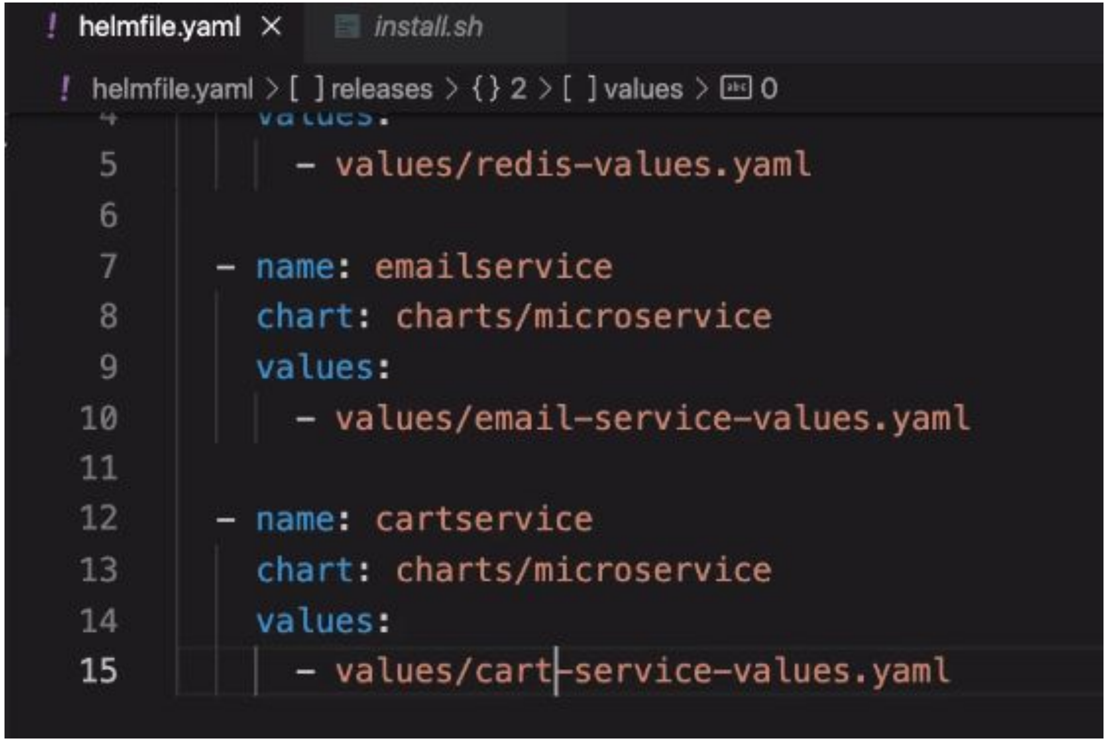
```yaml
releases:
  - name: rediscart
    chart: charts/redis
    values:
      - values/redis-values.yaml
```



helmfile.yaml > [ ] releases > { } 0 > [ ] values > { } 1 > appReplicas

Helmfile config schema (helmfile.json)

```yaml
releases:
  - name: rediscart
    chart: charts/redis
    values:
      - values/redis-values.yaml
      - appReplicas: "1"
      - volumeName: "redis-cart-data"

  - name: emailservice
    chart: charts/microservice
    values:
      - values/email-service-values.yaml
```

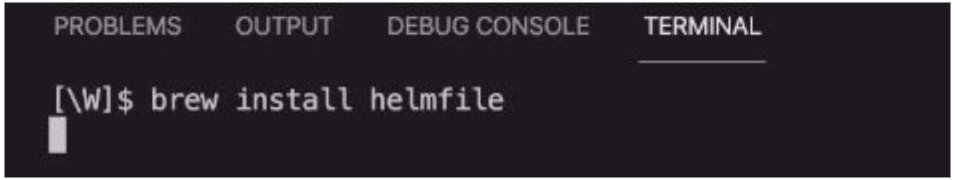


helmfile.yaml × install.sh

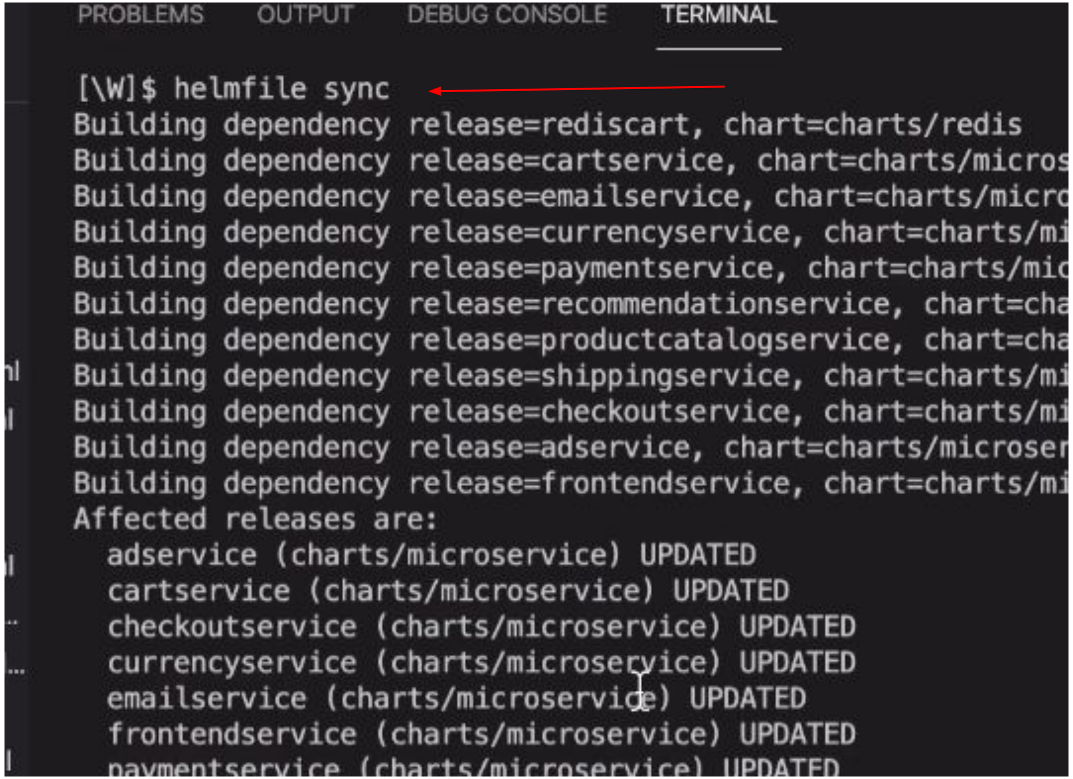
helmfile.yaml > [ ] releases > { } 2 > [ ] values > 0

```yaml
    values:
      - values/redis-values.yaml

  - name: emailservice
    chart: charts/microservice
    values:
      - values/email-service-values.yaml

  - name: cartservice
    chart: charts/microservice
    values:
      - values/cart-service-values.yaml
```

install helmfile



```
[\W]$ brew install helmfile
```



```
[\W]$ helmfile sync
Building dependency release=rediscart, chart=charts/redis
Building dependency release=cartservice, chart=charts/micros
Building dependency release=emailservice, chart=charts/micro
Building dependency release=currencyservice, chart=charts/mi
Building dependency release=paymentservice, chart=charts/mic
Building dependency release=recommendationservice, chart=cha
Building dependency release=productcatalogservice, chart=cha
Building dependency release=shippingservice, chart=charts/mi
Building dependency release=checkoutservice, chart=charts/mi
Building dependency release=adservice, chart=charts/microser
Building dependency release=frontendservice, chart=charts/mi
Affected releases are:
  adservice (charts/microservice) UPDATED
  cartservice (charts/microservice) UPDATED
  checkoutservice (charts/microservice) UPDATED
  currencyservice (charts/microservice) UPDATED
  emailservice (charts/microservice) UPDATED
  frontendservice (charts/microservice) UPDATED
  paymentservice (charts/microservice) UPDATED
```

helmfile list
check again
kubectl get pod
check again:
loadbalancer IP address

helmfile destory

host the helm charts in the git repo

- with application code or
- seperate git repo just for helm chart

How to feed into git CICD