# Solution M7: Templating Tools and Package Management

For this challenge we will assume that we are working as the **root** user on the control plane node of a three node **Kubernetes** cluster. The pod network plugin is of no importance, so it can be any of the available. We will execute all tasks in order but keep in mind that every other task expects that the artefacts from the previous have been deleted

The original files are in the **in/** folders and the resulting ones in the **out/** folders (where applicable)

# Task 1

**Challenge:**

Using the files in **task1** folder create a template using the **sed**-based approach. Parametrize the **number of replicas** and the **service port**

**Solution:**

First, we should choose a placeholder structure. For example, capital letters surrounded with % symbol

This way, we can prepare the solution manifest (**out/homework.yaml**) with the following content *(just the two changed sections are shown)*:

...

spec:

  replicas: %REPLICA-COUNT%

...

spec:

  ports:

  - port: 5000

    nodePort: %SERVICE-PORT%

...

Then we can deploy the manifest with

**sed 's/%REPLICA-COUNT%/3/ ; s/%SERVICE-PORT%/30001/' homework.yaml | kubectl apply -f -**

Next, we can check all the created artefacts with

**kubectl get all -n homework**

Open a browser tab and navigate to http://<cluster-ip>:30001

Our application should be there and working 😊

Clean up with

**kubectl delete ns homework**

# Task 2

**Challenge:**

Using the files in **task2** folder create a template using **kustomize** with two variants – **test** and **production** with difference in the **service port** and **number of replicas**

**Solution:**

Prepare the folder structure first

**mkdir out/base**

**mkdir -p out/overlays/{production,test}**

Then copy the source file

**cp in/homework.yaml out/base/homework.yaml**

Create the **base** customization file (**out/base/kustomization.yaml**) with the following content

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

resources:

  - homework.yaml

Test it that it is working

**kustomize build out/base/**

Create the **production** customization file (**out/overlays/production/kustomization.yaml**) with the following content

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

namePrefix: prod-

commonLabels:

  variant: prod

resources:

- ../../base

patchesStrategicMerge:

- custom-rep.yaml

- custom-svc.yaml

Create the **test** customization file (**out/overlays/test/kustomization.yaml**) with the following content

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

namePrefix: test-

commonLabels:

  variant: test

resources:

- ../../base

patchesStrategicMerge:

- custom-rep.yaml

- custom-svc.yaml

Then, prepare the replica count customization (**out/overlays/production/custom-rep.yaml**) for production

apiVersion: apps/v1

kind: Deployment

metadata:

  name: homework

  namespace: homework

spec:

  replicas: 3

And one for test (**out/overlays/test/custom-rep.yaml**) with similar content

apiVersion: apps/v1

kind: Deployment

metadata:

  name: homework

  namespace: homework

spec:

  replicas: 1

Now, we must prepare a pair of files for the service node port value

For the production environment (**out/overlays/production/custom-svc.yaml**) we can create the following

apiVersion: v1

kind: Service

metadata:

  name: homework-svc

  namespace: homework

spec:

  ports:

  - port: 5000

    nodePort: 30001

    protocol: TCP

And then, for test (**out/overlays/test/custom-svc.yaml**) the following one

apiVersion: v1

kind: Service

metadata:

  name: homework-svc

  namespace: homework

spec:

  ports:

  - port: 5000

    nodePort: 32001

    protocol: TCP

Now, test both variants

**kustomize build out/overlays/production/**

**kustomize build out/overlays/test/**

Everything seems to be fine

We can start both variants with

**kubectl apply -k out/overlays/production/**

**kubectl apply -k out/overlays/test/**

We can check the resources that were created with

**kubectl get all -n homework**

Everything seems to be fine. Let’s check the details of the namespace

**kubectl get ns homework --show-labels**

We can see that the label **variant=test** appears here

In the current situation, this is clearly something which we can safely ignore (normally, the namespace won’t be part of the manifest)

Open two browser tabs and navigate to

* http://<cluster-ip>:30001 for production
* http://<cluster-ip>:32001 for test

Our application should be there and working 😊

Clean up by deleting the whole namespace

**kubectl delete ns homework**

# Task 3

**Challenge:**

Create a **Helm** chart that spins a **NGINX**-based deployment with **3 replicas** by default. It must mount a default **index.html** (with a **text** and a **picture**) page from a **ConfigMap**. The web server should be exposed via **NodePort** service on port **31000** by default. At least the **text of the default page**, **number of replica**s, and **service port** should be parametrized

**Solution:**

First, prepare the directory structure

**mkdir -p hwchart/templates**

Next create the chart manifest (**hwchart/Chart.yaml**) with the following content

apiVersion: v2

name: hwchart

description: Homework chart based on NGINX

type: application

version: 0.1.0

appVersion: "1.21.5"

Then the parameters (**hwchart/values.yaml**) file with the following content

replicaCount: 3

htmlText: "Kubernetes + Helm = Awesome!"

nodePort: 31000

Next, we can prepare the service manifest (**hwchart/templates/service.yaml**) with the following content

apiVersion: v1

kind: Service

metadata:

  name: {{ .Release.Name }}-service

spec:

  type: NodePort

  ports:

  - port: 80

    nodePort: {{ .Values.nodePort }}

    protocol: TCP

  selector:

    app: {{ .Release.Name }}

As a next step, we can prepare the configuration map (**hwchart/templates/cm.yaml**) manifest

apiVersion: v1

kind: ConfigMap

metadata:

  name: {{ .Release.Name }}-cm

data:

  index.html: <h1>{{ .Values.htmlText }}</h1><br /><img src="image.png" />

binaryData:

  image.png: iVBORw0K ... <stripped>

Please note that the **image.png** data is a **base64** encoded contents of the original file

And finally, we must prepare the deployment manifest (**hwchart/templates/deployment.yaml**)

apiVersion: apps/v1

kind: Deployment

metadata:

  name: {{ .Release.Name }}-deployment

spec:

  replicas: {{ .Values.replicaCount }}

  selector:

    matchLabels:

      app: {{ .Release.Name }}

  template:

    metadata:

      labels:

        app: {{ .Release.Name }}

    spec:

      containers:

      - name: main

        image: nginx:1.21.5

        ports:

        - containerPort: 80

        volumeMounts:

        - name: html

          mountPath: /usr/share/nginx/html

      volumes:

      - name: html

        configMap:

          name: {{ .Release.Name }}-cm

Finally, we can deploy the chart with (for example with one replica instead of three)

**helm install homework hwchart --set replicaCount=1**

Check the deployed resources

**kubectl get pods,svc,cm,deploy**

Open a browser tab and navigate to http://<cluster-ip>:31000

Our app should be there and working 😊

Clean up with

**helm uninstall homework**