

Task 1: Ubuntu VM creation, docker apache container

- Create an Ubuntu VM on a local machine.
- make sure you are able to ping the VM IP from the laptop
- Install docker on the VM
- Run apache2 container
- Open browser in laptop and access apache2 container test page which was started in the previous step.

Checking connectivity between VM and host:

[illegible]

Checking the docker status:

```

gitcommit: de40add
ox@ox:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2021-03-29 10:07:47 EEST; 3h 45min ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 433336 (dockerd)
      Tasks: 31
     Memory: 256.0M
    CGroup: /system.slice/docker.service
            └─433336 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
              └─437321 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 5000 -container-ip 19
Mar 29 13:18:29 ox dockerd[433336]: time="2021-03-29T13:18:29.019791915+03:00" level=info msg="ignoring"
Mar 29 13:18:29 ox dockerd[433336]: time="2021-03-29T13:18:29.330237677+03:00" level=info msg="ignoring"
Mar 29 13:18:29 ox dockerd[433336]: time="2021-03-29T13:18:29.788918002+03:00" level=info msg="ignoring"
Mar 29 13:18:30 ox dockerd[433336]: time="2021-03-29T13:18:30.270768768+03:00" level=info msg="ignoring"
Mar 29 13:18:30 ox dockerd[433336]: time="2021-03-29T13:18:30.562071031+03:00" level=info msg="ignoring"
Mar 29 13:18:30 ox dockerd[433336]: time="2021-03-29T13:18:30.854119562+03:00" level=info msg="ignoring"
Mar 29 13:18:31 ox dockerd[433336]: time="2021-03-29T13:18:31.138527124+03:00" level=info msg="ignoring"
Mar 29 13:18:31 ox dockerd[433336]: time="2021-03-29T13:18:31.605253250+03:00" level=info msg="ignoring"
Mar 29 13:18:31 ox dockerd[433336]: time="2021-03-29T13:18:31.914473238+03:00" level=info msg="ignoring"
Mar 29 13:18:32 ox dockerd[433336]: time="2021-03-29T13:18:32.230132215+03:00" level=info msg="ignoring"
lines 1-22/22 (END)

```

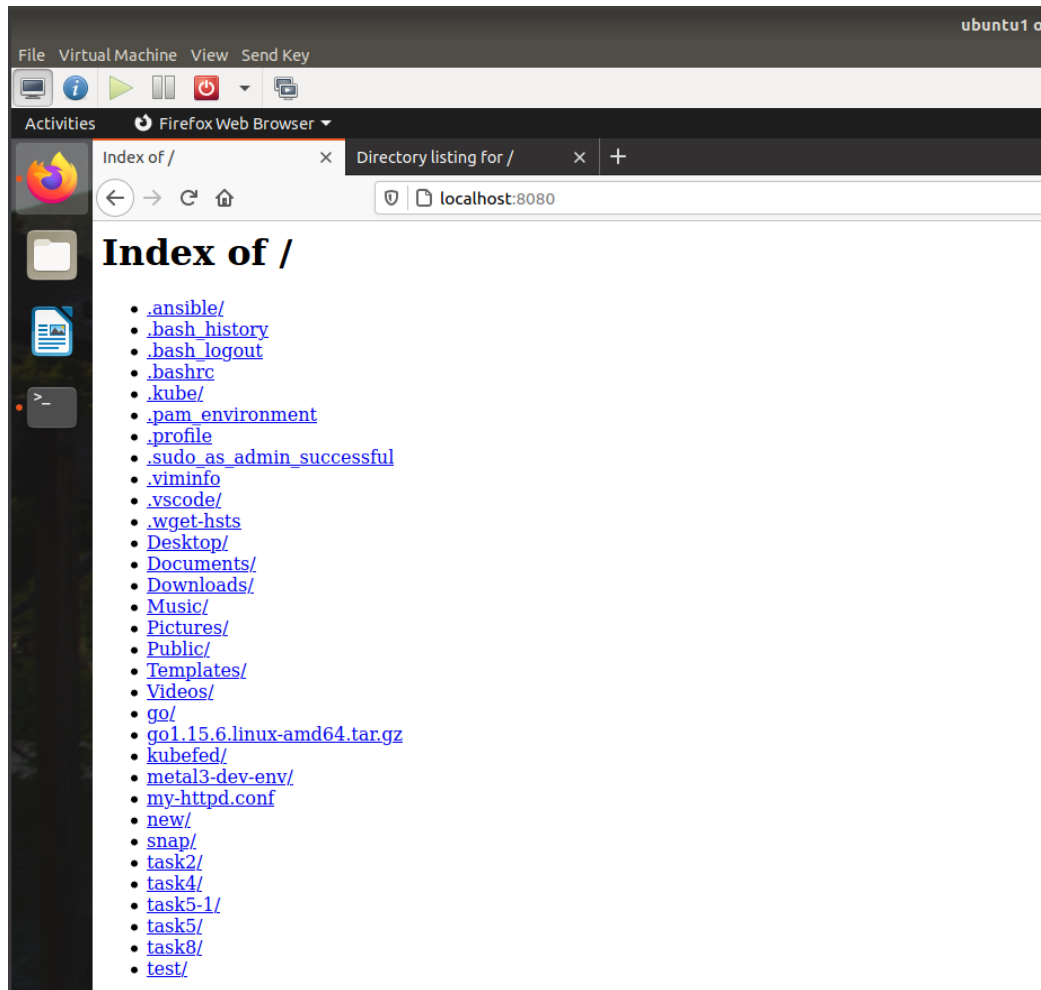
```
Running container $ docker run -dit --name my-apache-app -p 8080:80 -v
"$PWD": /usr/local/apache2/htdocs/ httpd:2.4
```

```

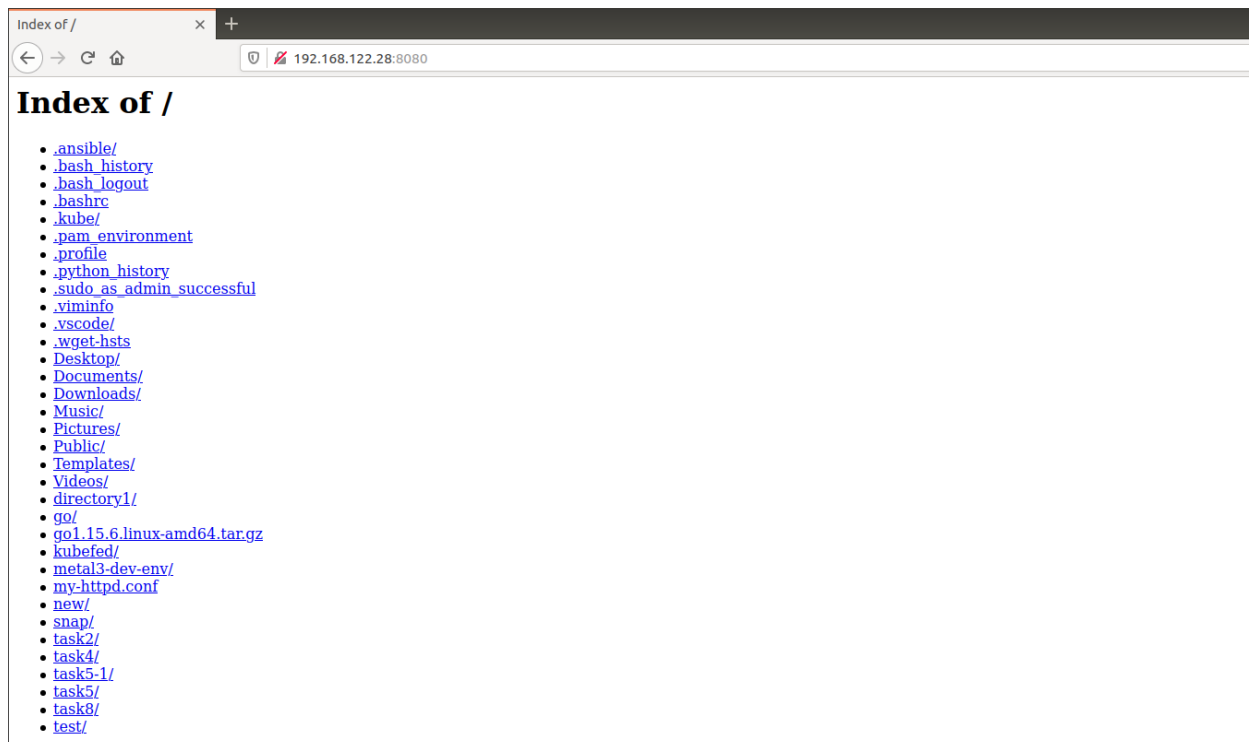
ox@ox:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
MES
cce5ab5adc39   httpd:2.4     "httpd-foreground"      3 minutes ago Up 3 minutes  0.0.0.0:8080->80/tcp
-apache-app

```

Access from VM:



Access from host machine:



Task 2: Container RUN_COUNT

- Create a container by using new Dockerfile. The container should run a shell script which creates a test file and write this string in file "RUN_COUNT = 1" in it and the container should exit. When you run the container next time, it should read the file which was created on the previous run. Increment the read count value and write the updated value to the file. e.g the second time you run the container, the value in the file should be "RUN_COUNT = 2". And similarly, next time it should be "RUN_COUNT = 3"

Dockerfile

```
FROM ubuntu:18.04
```

```
WORKDIR /opt
```

```
CMD /opt/script.sh
```

script.sh

```
#!/bin/bash
```

```
if [ ! -f /opt/test.txt ];  
then  
    touch /opt/test.txt  
    echo "RUN_COUNT = 1" > /opt/test.txt  
else  
    value=`awk '{print $3}' /opt/test.txt`  
    i=$((value+1))  
    echo "RUN_COUNT = $i" > /opt/test.txt  
fi
```

```
ox@ox:~/task2/opt$ vi Dockerfile  
ox@ox:~/task2/opt$ ls  
Dockerfile  script.sh  
ox@ox:~/task2/opt$ docker build -t task2:latest .  
Sending build context to Docker daemon 3.072kB  
Step 1/3 : FROM ubuntu:18.04  
18.04: Pulling from library/ubuntu  
6e0aa5e7af40: Pull complete  
d47239a868b3: Pull complete  
49cbb10cca85: Pull complete  
Digest: sha256:122f506735a26c0a1aff2363335412cfc4f84de38326356d31ee00c2cbe52171  
Status: Downloaded newer image for ubuntu:18.04  
--> 3339fde08fc3  
Step 2/3 : WORKDIR /opt  
--> Running in e7c9cc45b00e  
Removing intermediate container e7c9cc45b00e  
--> 5c802cada013  
Step 3/3 : CMD /opt/script.sh  
--> Running in d4a1ebee2c83  
Removing intermediate container d4a1ebee2c83  
--> 0a9154e39fa8  
Successfully built 0a9154e39fa8  
Successfully tagged task2:latest  
ox@ox:~/task2/opt$ ls  
Dockerfile  script.sh  
ox@ox:~/task2/opt$ mv Dockerfile ..  
ox@ox:~/task2/opt$ ls  
script.sh  
ox@ox:~/task2/opt$ cd ..  
ox@ox:~/task2$ ls  
Dockerfile  opt  
ox@ox:~/task2$ docker run -it -v ${PWD}/opt:/opt --name mycontianer task2:latest  
ox@ox:~/task2$ ls  
Dockerfile  opt  
ox@ox:~/task2$ ls opt  
script.sh  test.txt  
ox@ox:~/task2$ cat opt/test.txt  
RUN_COUNT = 1  
ox@ox:~/task2$ docker start mycontianer  
mycontianer  
ox@ox:~/task2$ cat opt/test.txt  
RUN_COUNT = 2  
ox@ox:~/task2$ docker start mycontianer  
mycontianer  
ox@ox:~/task2$ cat opt/test.txt  
RUN_COUNT = 3  
ox@ox:~/task2$
```

Task 3: Create Kubernetes cluster in Ubuntu VM (Using kind cluster)

- Create a kind cluster in the Ubuntu VM by following the instructions here.
<https://kind.sigs.k8s.io/docs/user/quick-start/>
- Once cluster is setup, run "kubectl get nodes" and verify how many kubernetes nodes exist in the cluster
- destroy the cluster
- create the kind cluster again with 4 worker nodes.

```
ebaroks@elxa6t56s73:~$ kind create cluster
Creating cluster "kind" ...
✓ Ensuring node image (kindest/node:v1.20.2)
✓ Preparing nodes
✓ Writing configuration
✓ Starting control-plane
✓ Installing CNI
✓ Installing StorageClass
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Thanks for using kind!
ebaroks@elxa6t56s73:~$ kubectl get nodes
NAME                 STATUS    ROLES                  AGE   VERSION
kind-control-plane   Ready     control-plane,master   84s   v1.20.2
ebaroks@elxa6t56s73:~$ kind delete cluster
Deleting cluster "kind" ...
ebaroks@elxa6t56s73:~$
```

Config for multiple nodes creation:

```
# this config file contains all config fields with comments
# NOTE: this is not a particularly useful config file
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
# patch the generated kubeadm config with some extra settings
kubeadmConfigPatches:
- |
  apiVersion: kubelet.config.k8s.io/v1beta1
  kind: KubeletConfiguration
  evictionHard:
    nodefs.available: "0%"
# patch it further using a JSON 6902 patch
kubeadmConfigPatchesJSON6902:
- group: kubeadm.k8s.io
  version: v1beta2
  kind: ClusterConfiguration
  patch: |
    - op: add
      path: /apiServer/certSANs/-
      value: my-hostname
# 1 control plane node and 3 workers
nodes:
# the control plane node config
- role: control-plane
# the three workers
- role: worker
- role: worker
- role: worker
```

```

ebaroks@elxa6t56s73:~$ kind create cluster --config task1.yaml
Creating cluster "kind" ...
  ✓ Ensuring node image (kindest/node:v1.20.2)
  ✓ Preparing nodes
  ✓ Writing configuration
  ✓ Starting control-plane
  ✓ Installing CNI
  ✓ Installing StorageClass
  ✓ Joining worker nodes
Set kubectrl context to "kind-kind"
You can now use your cluster with:

kubectrl cluster-info --context kind-kind

Thanks for using kind!
ebaroks@elxa6t56s73:~$ kubectrl get nodes
NAME                 STATUS    ROLES                  AGE   VERSION
kind-control-plane   NotReady  control-plane,master   46s   v1.20.2
kind-worker          NotReady  <none>                 10s   v1.20.2
kind-worker2         NotReady  <none>                 10s   v1.20.2
kind-worker3         NotReady  <none>                 11s   v1.20.2
ebaroks@elxa6t56s73:~$ kubectrl get nodes
NAME                 STATUS    ROLES                  AGE   VERSION
kind-control-plane   Ready     control-plane,master   81s   v1.20.2
kind-worker          Ready     <none>                 45s   v1.20.2
kind-worker2         Ready     <none>                 45s   v1.20.2
kind-worker3         NotReady  <none>                 46s   v1.20.2
ebaroks@elxa6t56s73:~$ kubectrl get nodes
NAME                 STATUS    ROLES                  AGE   VERSION
kind-control-plane   Ready     control-plane,master   94s   v1.20.2
kind-worker          Ready     <none>                 58s   v1.20.2
kind-worker2         Ready     <none>                 58s   v1.20.2
kind-worker3         Ready     <none>                 59s   v1.20.2
ebaroks@elxa6t56s73:~$

```

Task 4: Golang for reading YAML file

- Write a program in golang to read the following YAML file
<https://stackoverflow.com/questions/30947534/how-to-read-a-yaml-file>

Text

this:

is:

- my
- test
- yaml file

- Modify it the YAML file so that it looks like below

Text

this:

is:

my:

- test
- yaml file

```

task4.go
package main

import (
    "fmt"
    "io/ioutil"
    "log"

    "gopkg.in/yaml.v2"
)

type Input struct {
    This struct {
        Is []string `yaml:"is"`
    } `yaml:"this"`
}

type Output struct {
    This struct {
        Is struct {
            My []string `yaml:"my"`
        } `yaml:"is"`
    } `yaml:"this"`
}

func getInput() Input {
    var c Input
    yamlFile, err := ioutil.ReadFile("task4.yaml")
    if err != nil {
        log.Printf("yamlFile.Get err  %#v ", err)
    }
    err = yaml.Unmarshal(yamlFile, &c)
    if err != nil {
        log.Fatalf("Unmarshal: %v", err)
    }
    return c
}

func getOutput(c Input) Output {
    var o Output = Output{}
    o.This.Is.My = c.This.Is
    o.This.Is.My = []string{c.This.Is[1], c.This.Is[2]}
    return o
}

func main() {
    c := getInput()
    o := getOutput(c)
    output, err := yaml.Marshal(o)
    if err != nil {
        log.Printf("yamlFile.Get err  %#v ", err)
    }
    fmt.Printf("%+v\n", string(output))
}

```

```
task4.go
9
10 type Input struct {
11     This struct {
12         Is []string `yaml:"is"`
13     } `yaml:"this"`
14 }
15
16 type Output struct {
17     This struct {
18         Is struct {
19             My []string `yaml:"my"`
20         } `yaml:"is"`
21     } `yaml:"this"`
22 }
23
24 func getInput() Input {
25     var c Input
26     yamlFile, err := ioutil.ReadFile("task4.yaml")
27     if err != nil {
28         log.Printf("yamlFile.Get err %#v ", err)
29     }
30     err = yaml.Unmarshal(yamlFile, &c)
31     if err != nil {
32         log.Fatalf("Unmarshal: %v", err)
33     }
34     return c
35 }

task4.yaml
1 ---
2 this:
3   is:
4     - my
5     - test
6     - yaml file
7
8
9
10 # this:
11 # is:
12 # my:
13 #   - test
14 #   - yaml file

Terminal
ox@ox:~/task4$ go run task4.go
this:
  is:
    my:
      - my
      - test
      - yaml file

ox@ox:~/task4$ go run task4.go
this:
  is:
    my:
      - test
      - yaml file

ox@ox:~/task4$
```

Task 5: Use Golang Cobra to read input parameters

- Modify the program in Task 4 to take two parameters as an input, which are described below. Use Golang Cobra framework to read input parameters:

Param1 : full path for input yaml file

Param2 : username

- This time, it should read the yaml file modified by the program in the last step and modify "my:" to give the String username "username:". Check the example below

Text

```
---
this:
  is:
    username:
      - test
      - yaml file
```

MAIN.GO

```
package main

import (
    "task5_yaml/cmd"
)

func main() {
    cmd.Execute()
}
```

ROOT.GO

```
package cmd

import (
    "fmt"
    "log"
    "regexp"
    "github.com/spf13/cobra"
    "gopkg.in/yaml.v2"
)

var (
    // Used for flags.
    rootCmd = &cobra.Command{
        Use: "task5",
        Short: "task5 application",
        Args: cobra.ExactArgs(2),
        Run: func(cmd *cobra.Command, args []string) {
            validateUsername(args[1])
            validateYamlFile(args[0])
            C := getInput(args[0], args[1])
            output, err := yaml.Marshal(C)
            if err != nil {
                log.Printf("yamlFile.Get err  %#v ", err)
            }
            //C.This.Is = map[string][]string{args[1]: C.This.Is["username"]}
            //fmt.Println(C)
            fmt.Printf("%+v\n", string(output))
        },
    }
)

func validateUsername(username string) {
    r, _ := regexp.Compile("^[a-zA-z][a-zA-z_0-9]{0,10}")
    if !r.MatchString(username) {
        log.Fatalf("Provide a valid username")
    }
    if len(username) > 10 || len(username) < 4 {
        log.Fatalf("Provide at least 4 characters, or your username is too long")
    }
}

func validateYamlFile(filename string) {
    r, _ := regexp.Compile("^(.*)\\.yaml$")
    if !r.MatchString(filename) {
        log.Fatalf("Provide a yaml file")
    }
}

func Execute() error {
    return rootCmd.Execute()
}
```

TASK5.GO

```
package cmd
```

```
import (
    "io/ioutil"
```



```

        "log"
        "gopkg.in/yaml.v2"
    )

    type Input struct {
        This struct {
            Is map[string][]string `yaml:"is"`
        } `yaml:"this"`
    }

    func getInput(filename string, username string) Input {
        var c Input
        yamlFile, err := ioutil.ReadFile(filename)
        if err != nil {
            log.Printf("yamlFile.Get err  %#v ", err)
        }
        err = yaml.Unmarshal(yamlFile, &c)
        if err != nil {
            log.Fatalf("Unmarshal: %v", err)
        }
        c.This.Is[username] = c.This.Is["my"]
        delete(c.This.Is, "my")
        return c
    }

```

Task 6: Packaging and running program in kubernetes

- Create a container to package the binary built in Task 5. make sure the container does not exit after the binary has exited and hang indefinitely until the container is killed.
- Push the container to any publicly accessible docker registry
- write manifests so that the container is run on the kind cluster created in Task 3
- You should be able to provide the input YAML file using a config map.
- Check the output of the program by going to the container running in Kubernetes and verify the file is modified.

Dockerfile

```
FROM golang:1.16-alpine AS build
```

```
WORKDIR /app/
```

```
#COPY main.go go.* /app/
```

```
COPY . /app
```

```
RUN CGO_ENABLED=0 go build
```

```
FROM alpine
```

```
WORKDIR /app/
```

```
COPY script.sh /app
```

```
COPY --from=build /app/task5_yaml /app/task5_yaml
```

```
ENTRYPOINT ["/app/script.sh"]
```

script.sh

```
#!/bin/sh
```

```
./task5_yaml $*
```

```
while true; do sleep 1000; done
```

manifest.yaml

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: my-container-task6
```

```
spec:
```

```
  hostNetwork: true
```

```
  containers:
```

```
    - name: task5image
```

```
      image: ebaroks/imagetask6
```

```
      args: ["/tmp/task5.yaml", 'username']
```

```
      volumeMounts:
```

```
        - name: configvolume
```

```
          mountPath: /tmp/task5.yaml
```

```
          subPath: task5.yaml
```

```
      env:
```

```
        - name: CONFIG_USERNAME
```

```
          valueFrom:
```

```
            secretKeyRef:
```

```
              name: task7-secret
```

```
              key: username
```

```
        - name: CONFIG_PASSWORD
```

```
          valueFrom:
```

```
            secretKeyRef:
```

```
              name: task7-secret
```

```
              key: password
```

```
  volumes:
```

```
    - name: configvolume
```

```
      configMap:
```

```
        name: task6
```

configmap.yaml

```
apiVersion: v1
```

```
kind: ConfigMap
```

```
metadata:
```

```
  name: task6
```

```
data:
```

```
  task5.yaml: |
```

```
  this:
```

```
    is:
```

```
      my:
```

```
        - test
```

```
        - yaml file
```

Running the container with a binary inside:

```
the connection to the server localhost:8080 was refused - did you specify the right host?
ox@ox:~/new$ kind create cluster
Creating cluster "kind" ...
 ✓ Ensuring node image (kindest/node:v1.20.2) 📦
 ✓ Preparing nodes 📶
 ✓ Writing configuration 📄
 ✓ Starting control-plane 🚦
 ✓ Installing CNI 🌐
 ✓ Installing StorageClass 💾
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Not sure what to do next? 😊 Check out https://kind.sigs.k8s.io/docs/user/quick-start/
ox@ox:~/new$ kubectl apply -f configmap.yaml
configmap/task6 created
ox@ox:~/new$ kubectl apply -f secret.yaml
secret/task7-secret created
ox@ox:~/new$ kubectl apply -f manifest.yaml
pod/my-container-task6 created
ox@ox:~/new$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
544b633c585c   kindest/node:v1.20.2               "/usr/local/bin/entr..." About a minute ago Up A
ox@ox:~/new$ kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
my-container-task6 0/1     ContainerCreating 0          19s
ox@ox:~/new$ kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
my-container-task6 0/1     ContainerCreating 0          21s
ox@ox:~/new$ kubectl get pod -w
NAME          READY   STATUS    RESTARTS   AGE
my-container-task6 1/1     Running   0          24s
^Cox@ox:~/new$
```

Description of the running container

```
Image ID:      docker.io/ebarkos/imagetask6@sha256:839b6c6dfdcdd801ed46fc17537c5af8be33f9fa550f2c327e0825ef3aed0d34
Port:          <none>
Host Port:     <none>
Command:       /app/script.sh
Args:          /tmp/task5.yaml
              username
State:         Running
Started:       Sat, 10 Apr 2021 23:46:51 +0300
Ready:         True
Restart Count: 0
Environment:   CONFIG_USERNAME: <set to the key 'username' in secret 'task7-secret'> Optional: false
              CONFIG_PASSWORD: <set to the key 'password' in secret 'task7-secret'> Optional: false
Mounts:        /tmp/task5.yaml from configvolume (rw,path="/task5.yaml")
              /var/run/secrets/kubernetes.io/serviceaccount from default-token-qk2kd (ro)
Conditions:
  Type             Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  configvolume:
    Type:          ConfigMap (a volume populated by a ConfigMap)
    Name:          task6
    Optional:      false
  default-token-qk2kd:
    Type:          Secret (a volume populated by a Secret)
    SecretName:    default-token-qk2kd
    Optional:      false
QoS Class:        BestEffort
Node-Selectors:   <none>
Tolerations:      node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason             Age    From          Message
  ---    -
Warning  FailedScheduling   2m16s (x3 over 2m26s)  default-scheduler  0/1 nodes are available: 1 node(s) had taint {node.kubernetes.io/not-ready: },
Normal   Scheduled          2m12s  default-scheduler  Successfully assigned default/my-container-task6 to kind-control-plane
Normal   Pulling            2m11s  kubelet          Pulling image "ebarkos/imagetask6"
Normal   Pulled             2m4s   kubelet          Successfully pulled image "ebarkos/imagetask6" in 6.806755467s
Normal   Created            2m4s   kubelet          Created container task5image
Normal   Started            2m4s   kubelet          Started container task5image
ox@ox:~/new$
```

Task 7: Using Kubernetes secrets

- Now modify the manifest in Task 6 and provide the input file using a secret this time.

`secret.yaml`

```
apiVersion: v1
kind: Secret
metadata:
  name: task7-secret
  type: Opaque
data:
  username: dXNlcm5hbWU=
  password: cGFzc3dvcmQ=
```

Secrets are attached to the Pod via Environment variables (CONFIG_USERNAME, CONFIG_PASSWORD).
Configmap from the task6 is attached to the Pod via Volume.

Task 8: Using kubernetes client-go library

- Write a small program that reads Kubernetes manifest files created in Task 6 and create the same container using the golang code, without using the kubectl.
- You can use the client-go library in golang for this purpose.

The following resources were used to write a program:

<https://github.com/kubernetes/client-go/>

<https://pkg.go.dev/k8s.io/api/core/v1>

<https://v1-16.docs.kubernetes.io/docs/reference/generated/kubernetes-api/v1.16/#pod-v1-core>

<https://gist.github.com/dlorenc/2ac000a25656447a0c06e79150407b4c> k8s yaml parse

<https://github.com/hossainemruz/k8s-client-go-practice/blob/56eeb12f1ce54d3fe4d428e7546d70cd5bde3b50/main.go> client go clientset

`task5.go`

```
package cmd
import (
    "bufio"
    "context"
    "fmt"
    "io"
    "log"
    "os"
    "path/filepath"
    core "k8s.io/api/core/v1"
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
    "k8s.io/apimachinery/pkg/runtime"
    "k8s.io/apimachinery/pkg/util/yaml"
    "k8s.io/client-go/kubernetes"
    "k8s.io/client-go/kubernetes/scheme"
    "k8s.io/client-go/tools/clientcmd"
    "k8s.io/client-go/util/homedir"
)

func deployManifest(filename string, username string) {
    f, err := os.Open(filename)
    if err != nil {
        log.Fatalf("yamlFile.Get err  %#v ", err)
    }
    defer f.Close()
    b := bufio.NewReader(f)
```

```

    r := yaml.NewYAMLReader(b)

    doc, err := r.Read()
    if err == io.EOF {
        log.Fatalf("Empty Yaml file: %s\n%s", filename, err)
    }
    if err != nil {
        log.Fatal(err)
    }
    d := scheme.Codecs.UniversalDeserializer()
    obj, _, err := d.Decode(doc, nil, nil)
    if err != nil {
        log.Fatalf("could not decode yaml: %s\n%s", filename, err)
    }
    fmt.Println(obj)
    fmt.Println("-----")
    clientset := createClientSet()
    deployPod(clientset, obj)
}

func createClientSet() *kubernetes.Clientset {
    // var kubeconfig *string
    // if home := homedir.HomeDir(); home != "" { // check if machine has home directory.
    //     // read kubeconfig flag. if not provided use config file $HOME/.kube/config
    //     kubeconfig = flag.String("kubeconfig", filepath.Join(home, ".kube", "config"), "(optional)
absolute path to the kubeconfig file")
    // } else {
    //     kubeconfig = flag.String("kubeconfig", "", "absolute path to the kubeconfig file")
    // }
    // flag.Parse()
    home := homedir.HomeDir()
    kubeconfig := filepath.Join(home, ".kube", "config")

    // build configuration from the config file.
    config, err := clientcmd.BuildConfigFromFlags("", kubeconfig)
    if err != nil {
        panic(err)
    }
    // create kubernetes clientset. this clientset can be used to create,delete,patch,list etc for the kubernetes
resources
    clientset, err := kubernetes.NewForConfig(config)
    if err != nil {
        panic(err)
    }
    return clientset
}

func deployPod(clientset *kubernetes.Clientset, obj runtime.Object) {
    // now create the pod in kubernetes cluster using the clientset
    podobj := obj.(*core.Pod)
    _, err := clientset.CoreV1().Pods("default").Create(context.Background(), podobj,
        metav1.CreateOptions{})
    if err != nil {
        panic(err)
    }
    fmt.Println("Pod created successfully...")
}

```

Output of the running program:

```
exit status 2
oobx@v-tasks05 ~$ go run main.go manifest.yaml username
Pod created successfully...
oobx@v-tasks05 ~$ go run main.go manifest.yaml username
6Pod(ObjectMeta:{my-container-task8      0000-01-01 00:00:00 +0000 UTC} <nil> <nil> map[] [] map[] [] []);Spec:PodSpec{Volumes:[Volume{Volume{Name:configVolume,VolumeSource:VolumeSource{HostPath:nil,EmptyDir:nil,GCEPersistentDisk:nil,AWSElasticBlockStore:nil,GlusterPod:nil,NFS:nil,Secret:nil,NFS:nil,TCS:nil,Glusterfs:nil,PersistentVolumeClaim:nil,RBD:nil,FlexVolume:nil,Cinder:nil,CephFS:nil,Flocker:nil,DownwardAPI:nil,FCInit:nil,AzureFile:nil,ConfigMap:ConfigMap{VolumeSource{LocalObjectReference{Name:task8,KItems:[KeyToPath{"key": "secret",Optional:nil},ValuePerVolume:nil,DuoByte:nil,AzureDisk:nil,PhotonPersistentDisk:nil,PortworxVolume:nil,ScaledIO:nil,Projected:nil,StorageOS:nil,CSI:nil,Ephemeral:nil,...}],Containers:[Container{ContainerName:task5Image,Image:ebarks/image:task5,Command:[/app/script.sh],Args:[tmp/task5.yaml username],WorkingDir:,Ports:[ContainerPort{}],Env:[EnvVar{EnvName:CONFIG_USERNAME,Value:ValueFrom:<EnvVarSource{FieldRef:nil,ResourceFieldRef:nil,ConfigMapKeyRef:nil,SecretKeySelector:LocalObjectReference{LocalObjectReference{Name:task7-secret,key:username,Optional:nil}},}}},EnvVarName:CONFIG_PASSWORD,Value:ValueFrom:<EnvVarSource{FieldRef:nil,ResourceFieldRef:nil,ConfigMapKeyRef:nil,SecretKeySelector:LocalObjectReference{LocalObjectReference{Name:task7-secret,key:password,Optional:nil}},}}},Resources:ResourceRequirements{Limits:ResourceList{Requests:ResourceList{{}},VolumeMounts:[VolumeMount{VolumeMount{Name:configVolume,ReadOnly:false,MountPath:/tmp/task5.yaml,SubPath:task5.yaml,MountPropagation:nil,SubPathExpr:...}],LivenessProbe:nil,ReadinessProbe:nil,Lifecycle:nil,TerminationMessagePath:ImagePullPolicy:SecurityContext:nil,Stdin:false,false,TTY:false,EnvFrom:[<EnvFromSource>,TerminationMessagePolicy:VolumeDevices:[VolumeDevice]{,StartupProbe:nil,,RestartPolicy:TerminationGracePeriodSeconds:nil,ActiveDeadlineSeconds:nil,DNSPolicy:Nodeselectormap[string]sdnsmap[string]ServiceAccountName:PodSecurityContext:Nodename:HostNetwork:true,HostIPC:false,SecurityContext:nil,ImagePullSecrets:[LocalObjectReference{HostPriority:SchedulingAffinity:nil,SchedulerName:InitContainer:nil},AutomountServiceAccountToken:nil,Tolerations:[Toleration]{,HostAliases:[HostAlias]{,PriorityClassName:HostPriority:nil,DNSConfig:nil,ShareProcessNamespace:nil,ReadinessGates:[PodReadinessGate],RuntimeClassName:nil,EnableServiceLinks:nil,PreemptionPolicy:nil,Overhead:ResourceList{,TopologySpreadConstraints:[TopologySpreadConstraint]{,EphemeralContainers:[EphemeralContainer]{,SetHostname:FQDN:nil,,Status:PodStatus{Phase:Conditions:[PodCondition{,Message:Reason:,HostIP:,PodIP:,StartTime:nil,,ContainerStatuses:[ContainerStatus]{,OSClass:,InitContainerStatuses:[ContainerStatus]{,NominatedNodeName:,PodIPs:[PodIP]{,EphemeralContainerStatuses:[ContainerStatus]{,}}}
Pod created successfully...
oobx@v-tasks05 ~$
```

Check container creation:

```

my-container-task6 1/1 Running 0 23h
ox@ox:~$ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:42999
KubeDNS is running at https://127.0.0.1:42999/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
ox@ox:~$ kubectl et pods
Error: unknown command "et" for "kubectl"

Did you mean this?
    set
    get
    edit
    cp

Run 'kubectl --help' for usage.
ox@ox:~$ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:42999
KubeDNS is running at https://127.0.0.1:42999/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
ox@ox:~$ kubectl get pods
NAME                                STATUS    RESTARTS   AGE
my-container-task6                  1/1      Running    0           23h
my-container-task8                  1/1      Running    0           23m
ox@ox:~$

```

The code written for the tasks can be found here:

https://github.com/oksanabaranova/Learning_tasks