LAB1

A.1 Prerequisites for working with Kubernetes

Install minikube on linux virtual fedora:

Requirement:
https://minikube.sigs.k8s.io/docs/drivers/none/
Installation guide:
https://minikube.sigs.k8s.io/docs/start/

WARNING:
Use miniklube as **root** user

```
minikube start --force --driver=docker
```

Set kubectl cli alias
https://minikube.sigs.k8s.io/docs/handbook/kubectl/

```
cp /etc/bashrc /etc/bashrc.bk
echo "alias kubectl=\"minikube kubectl --\"" >> /etc/bashrc
source /etc/bashrc
```

Start kubernetes in docker with NON user root
as root:

```
adduser developer
```

Create a password for developer

```
passwd developer
```

Add developer user to wheel and docker groups

```
usermod -aG wheel developer
usermod -aG docker developer
```

Login as developer

```
su - developer
```

Start minikube

```
minikube start --driver=docker
```

Set kubectl cli alias
https://minikube.sigs.k8s.io/docs/handbook/kubectl/

```
echo "alias kubectl=\"minikube kubectl --\"" >> ~/.bashrc
```

Install kubernetes in you own operating system only if necessary
https://minikube.sigs.k8s.io/docs/start/

LAB2

Working with pod

CREATING A NAMESPACE FROM A YAML FILE
First, create a **custom-namespace.yaml** file with the following listing's contents (you'll find the file in the book's code archive).

```
apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace
```

Creating a simple YAML descriptor for a pod nxinx-pod.yaml

```
kind: Pod
apiVersion: v1
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.11
    ports:
    - containerPort: 80
      protocol: TCP
```

Using kubectl create to create the pod

To create the pod from your YAML file, use the kubectl create command:

```
$ kubectl create -f nginx-pod.yaml -n custom-namespace
```

Use the following command to see the full descriptor of the pod and take a little time to inspect the output

```
$ kubectl get po nginx -o yaml -n custom-namespace
```

Let's list pods to see their statuses:

```
$ kubectl get po -n custom-namespace
NAME                      READY   STATUS    RESTARTS   AGE
nginx                     1/1     Running   0          2m14s
```

Set up the following alias:

```
cp /etc/bashrc /etc/bashrc.bk
echo "alias kcd=\"kubectl config set-context $(kubectl config
current-context) --namespace \"" >> /etc/bashrc
source /etc/bashrc
```

Set the custom-namespace as default

```
$  kcd custom-namespace
Context "minikube" modified.
$ kubectl get po
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          84s
```

FORWARDING A LOCAL NETWORK PORT TO A PORT IN THE POD

The following command will forward your machine's local port 8080 to port 80 of your nginix pod:

```
$ kubectl get po
NAME                      READY   STATUS    RESTARTS   AGE
nginx                     1/1     Running   0          171m
$ kubectl port-forward nginx 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

Open a new shell to get the log

```
$ kubectl logs po/nginx -f
```

The port forwarder is running and you can now connect to your pod through the local port.

```
$ curl localhost:8080

<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
```

```
<p>If you see this page, the nginx web server is successfully
installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

See the log:

```
$ kubectl logs po/nginx -f
127.0.0.1 - - [29/Nov/2020:21:46:19 +0000] "GET / HTTP/1.1" 200
425 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 11_0_0)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198
Safari/537.36" "-"
127.0.0.1 - -
[...]
```

Deleting pods by deleting the whole namespace

```
$ kcd default
Context "minikube" modified.
```

Stop the port forwarder typing Ctrl+c

Delete your namespace `"custom-namespace"`

```
$ kubectl delete ns custom-namespace
namespace "custom-namespace" deleted


$ kubectl get ns
NAME              STATUS   AGE
default           Active   57d
kube-node-lease   Active   57d
kube-public       Active   57d
kube-system       Active   57d
```

LAB3

Creating a ReplicationController

create a YAML file called **kubia-rc.yaml** for your ReplicationController

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: kubia
spec:
  replicas: 3
  selector:
    app: kubia
  template:
    metadata:
      labels:
        app: kubia
    spec:
      containers:
      - name: kubia
        image: luksa/kubia
        ports:
        - containerPort: 8080
```

Use the kubectl create command to create your ReplicationController

```
$ kubectl create -f kubia-rc.yaml
replicationcontroller/kubia created
$ kubectl get po
NAME          READY   STATUS    RESTARTS   AGE
kubia-9cbh4   1/1     Running   0          116s
kubia-d7c8m   1/1     Running   0          116s
kubia-hwmt8   1/1     Running   0          116s
```

Try to delete a pod to see how the ReplicationController spins up a new one immediately

```
$ kubectl delete pod kubia-9cbh4
pod "kubia-9cbh4" deleted

$ kubectl get pod
NAME          READY   STATUS        RESTARTS   AGE
kubia-9cbh4   1/1     Terminating   0          13m
kubia-9gjdj   1/1     Running       0          27s
kubia-d7c8m   1/1     Running       0          13m
kubia-hwmt8   1/1     Running       0          13m
```

Now, let’s see what information the kubectl get command shows for ReplicationControllers

```
$ kubectl get rc
NAME    DESIRED   CURRENT   READY   AGE
kubia   3         3         3       15m
```

You can see additional information about your ReplicationController with the kubectl describe command, as shown in the following listing.

```
$ kubectl describe rc kubia
Name:         kubia
Namespace:    default
Selector:     app=kubia
Labels:       app=kubia
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=kubia
  Containers:
   kubia:
    Image:        luksa/kubia
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age   From                    Message
  ----    ------            ----  ----                    -------
  Normal  SuccessfulCreate  17m   replication-controller  Created
pod: kubia-hwmt8
  Normal  SuccessfulCreate  17m   replication-controller  Created
pod: kubia-9cbh4
  Normal  SuccessfulCreate  17m   replication-controller  Created
pod: kubia-d7c8m
  Normal  SuccessfulCreate  5m1s  replication-controller  Created
pod: kubia-9gjdj
```

Your ReplicationController has been keeping three instances of your pod running. Try to scale that number up to 10 now manually.

```
$ kubectl scale rc kubia --replicas=10

$ kubectl get po
NAME          READY   STATUS              RESTARTS   AGE
kubia-9gjdj   1/1     Running             0          8m45s
kubia-bdt9h   0/1     ContainerCreating   0          7s
```

```
kubia-bmz6g   0/1     ContainerCreating   0          7s
kubia-d7c8m   1/1     Running             0          21m
kubia-hwmt8   1/1     Running             0          21m
kubia-pbm7g   1/1     Running             0          7s
kubia-pp2lc   0/1     ContainerCreating   0          7s
kubia-tvt92   1/1     Running             0          7s
kubia-wzxnq   0/1     ContainerCreating   0          7s
kubia-zgh9r   0/1     ContainerCreating   0          7s
```

Scaling your ReplicationController by editing it's definition

```
$ kubectl edit rc kubia
replicationcontroller/kubia edited
```

Set the value of
Spec:
  Replicas: 3

```
$ kubectl get po
NAME          READY   STATUS    RESTARTS   AGE
kubia-9gjdj   1/1     Running   0          14m
kubia-d7c8m   1/1     Running   0          27m
kubia-hwmt8   1/1     Running   0          27m
```

Delete your ReplicationControllet without deleting associated pod (--cascade=false)

```
$ kubectl delete rc kubia --cascade=false
replicationcontroller "kubia" deleted

$ kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
kubia-9gjdj   1/1     Running   0          18m
kubia-d7c8m   1/1     Running   0          30m
kubia-hwmt8   1/1     Running   0          30m
```

You'll rewrite your ReplicationController into a ReplicaSet by creating a new file called **kubia-replicaset.yaml** with the contents in the following listing.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: kubia
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kubia
  template:
    metadata:
      labels:
        app: kubia
    spec:
      containers:
      - name: kubia
        image: luksa/kubia
```

```
$ kubectl create -f kubia-replicaset.yaml
replicaset.apps/kubia created
```

```
$ kubectl describe rs kubia
Name:         kubia
Namespace:    default
Selector:     app=kubia
Labels:       <none>
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=kubia
  Containers:
   kubia:
    Image:        luksa/kubia
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:           <none>
```

Try to delete an other pod:

```
$ kubectl delete pod kubia-9gjdj
pod "kubia-9gjdj" deleted
```

```
$ kubectl get po
NAME          READY   STATUS        RESTARTS   AGE
kubia-9gjdj   1/1     Terminating   0          24m
kubia-d7c8m   1/1     Running       0          37m
kubia-dv7h9   1/1     Running       0          5s
kubia-hwmt8   1/1     Running       0          37m
```

Clean all.

LAB 4

If not already active create a new ReplicaSet

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: kubia
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kubia
  template:
    metadata:
      labels:
        app: kubia
    spec:
      containers:
      - name: kubia
        image: luksa/kubia

$ kubectl create -f kubia-replicaset.yaml
replicaset.apps/kubia created
```

Create a file called **kubia-svc.yaml** with the following listing's contents.

```
apiVersion: v1
kind: Service
metadata:
  name: kubia
spec:
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: kubia
```

After posting the YAML, you can list all Service resources in your namespace and see that an internal cluster IP has been assigned to your service:

```
$ kubectl get service
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
kubia        ClusterIP   10.102.158.76   <none>        80/TCP    16s
```

REMOTELY EXECUTING COMMANDS IN RUNNING CONTAINERS

be sure to replace the pod name and the service IP with your own:

```
$ kubectl get pod
NAME           READY   STATUS    RESTARTS   AGE
kubia-b2v5k    1/1     Running   0          57m
kubia-hst84    1/1     Running   0          57m
kubia-l55vf    1/1     Running   0          57m

$ kubectl exec kubia-hst84 -- curl -s http://10.102.158.76
You've hit kubia-l55vf
```

Using DNS
You can use the kubectl exec command to run bash (or any other shell) inside a pod's container.

```
$ kubectl get po
NAME           READY   STATUS    RESTARTS   AGE
kubia-b2v5k    1/1     Running   0          74m
kubia-hst84    1/1     Running   0          74m
kubia-l55vf    1/1     Running   0          74m

kubectl exec -it kubia-b2v5k bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in
a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@kubia-b2v5k:/# curl http://kubia.default.svc.cluster.local
You've hit kubia-l55vf
root@kubia-b2v5k:/# curl http://kubia.default
You've hit kubia-hst84
```

Look at the /etc/resolv.conf file in the container and you'll understand:

```
root@kubia-b2v5k:/# cat /etc/resolv.conf
nameserver 10.96.0.10
```