Contents

Lab: Module 5 - Working with Kubernetes MiniKube

Duration: 40 minutes

Exercise 1: Setup and Configure MiniKube on a Virtual Machine

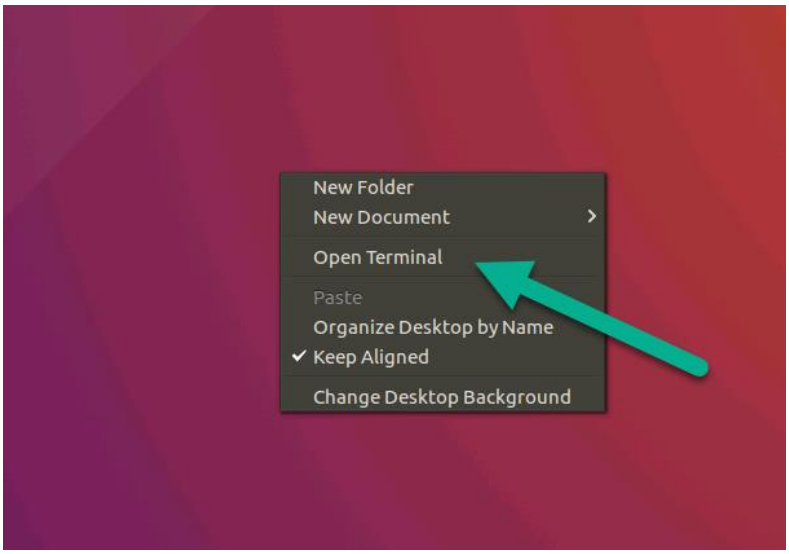
In this exercise you will setup and configure MiniKube on an Ubuntu VM. MiniKube is a tool that makes it easy to run Kubernetes locally and runs a single-node Kubernetes cluster.

Note that the LOD machine you are using supports nested virtualization. If you want to use your enterprise or MyVisualStudio (MSDN) account in the future to provision a VM, you must use a Standard_D2s_v3 or higher on Azure. The Azure Passes given during this workshop do not provide nested virtualization VMs in Azure.

Tasks

1. **Install Minikube**

    1. Sign into your LOD Ubuntu VM in the same manner you have in previous labs.
    2. Open a terminal by right clicking anywhere on the Desktop.



    3. Go into root and type in your password when prompted. The Password for the VM is: P@ssw0rd123!

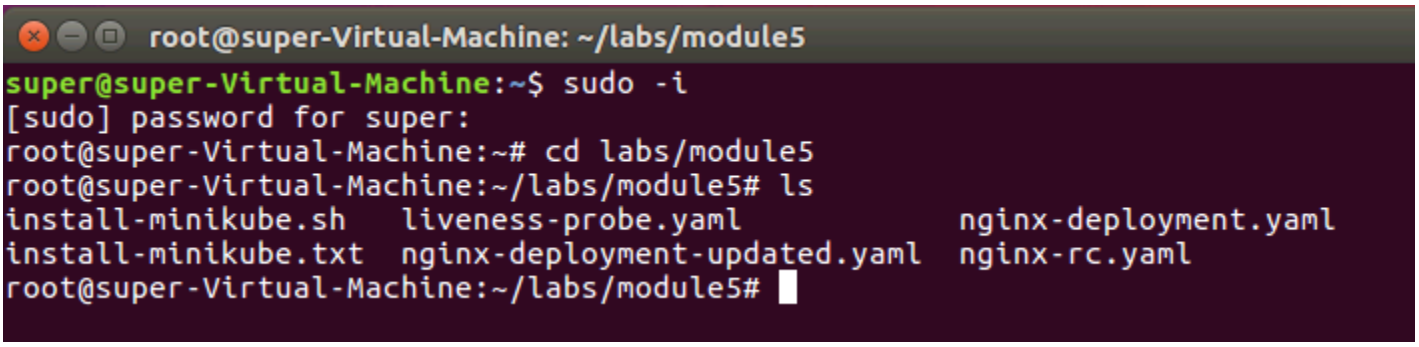
    ```
    sudo -i
    ```

    4. Run the following commands:

```
rm /var/lib/apt/lists/lock
rm /var/cache/apt/archives/lock
rm /var/lib/dpkg/lock
```

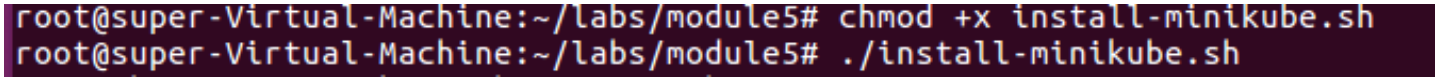
5. Navigate to the lab files by running:

```
cd labs/module5
```



6. You are now to install Minikube. Run the following two commands (the first command gives execute permission to your script, the second script runs it):

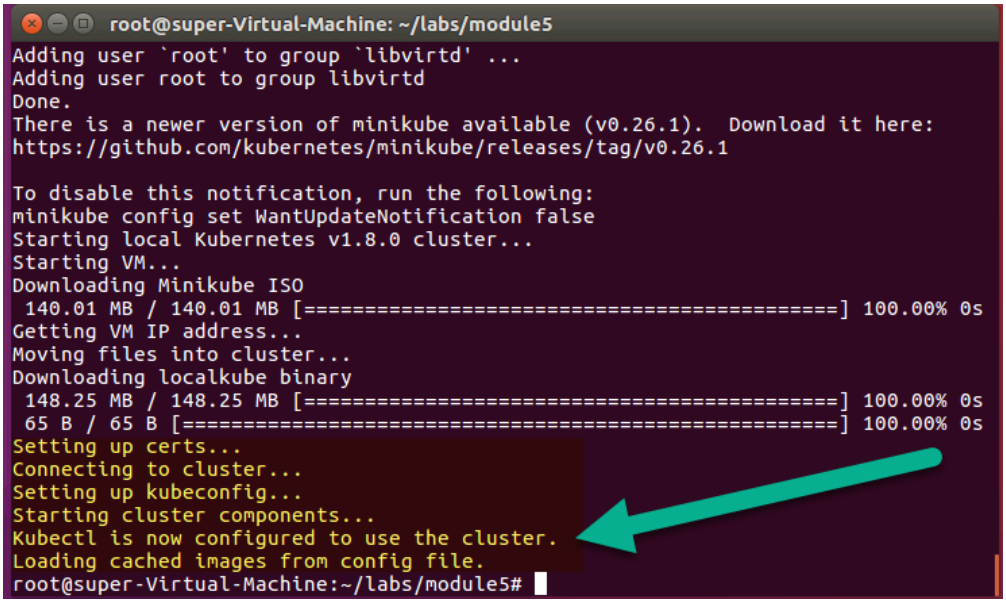
```
chmod +x install-minikube.sh
./install-minikube.sh
```



7. Read over the contents of the bash file you just ran while it is installing (the install will take ~3-5 minutes):

```
sudo -i

sudo apt-get -y update
sudo apt-get -y upgrade

#Install kubectl (latest)
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/sta

#Make the kubectl binary executable.
chmod +x ./kubectl

#Move the binary in to your PATH.
sudo mv ./kubectl /usr/local/bin/kubectl

#Install minikube. Make sure to check for latest version (current version is 0.24.1)
curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.24.1/minikube-linux-amd64 && chmod +x minikube && sudo mv minikube /

#Install kvm2
curl -LO https://storage.googleapis.com/minikube/releases/latest/docker-machine-driver-kvm2 && chmod +x docker-machine-driver-kvm2 && sudo

#Install Install libvirt and qemu-kvm and libvirt-bin
sudo apt install -y qemu-kvm libvirt-bin

#Add group libvirtd
sudo addgroup libvirtd
#Add current user to libvirtd group
sudo adduser $USER libvirtd

#Run minikube
minikube start --vm-driver kvm2
```

After minikube installation is completed, the last command in the text file will start the minikube with a single node Kubernetes cluster.



8. You can always find out more about minikube commands using the help switch, try running it now (do not worry about the screen color in the screenshots being different from your LOD Ubuntu VM!):

   minikube --help

```
root@minikube-vm:~# minikube --help
Minikube is a CLI tool that provisions and manages single-node Kubernetes clusters optimized for development workflows.

Usage:
  minikube [command]

Available Commands:
  addons           Modify minikube's kubernetes addons
  cache            Add or delete an image from the local cache.
  completion       Outputs minikube shell completion for the given shell (bash or zsh)
  config           Modify minikube config
  dashboard        Opens/displays the kubernetes dashboard URL for your local cluster
  delete           Deletes a local kubernetes cluster
  docker-env       Sets up docker env variables; similar to '$(docker-machine env)'
  get-k8s-versions Gets the list of Kubernetes versions available for minikube when using the localkube bootstrapper
  ip               Retrieves the IP address of the running cluster
  logs             Gets the logs of the running localkube instance, used for debugging minikube, not user code
  mount            Mounts the specified directory into minikube
  profile          Profile sets the current minikube profile
  service          Gets the kubernetes URL(s) for the specified service in your local cluster
  ssh              Log into or run a command on a machine with SSH; similar to 'docker-machine ssh'
  ssh-key          Retrieve the ssh identity key path of the specified cluster
  start            Starts a local kubernetes cluster
  status           Gets the status of a local kubernetes cluster
  stop             Stops a running local kubernetes cluster
  update-check     Print current and latest version number
  update-context   Verify the IP address of the running cluster in kubeconfig.
  version          Print the version of minikube

Flags:
      --alsologtostderr                  log to standard error as well as files
  -b, --bootstrapper string              The name of the cluster bootstrapper that will set up the kubernetes cluster. (default "localkube")
      --log_backtrace_at traceLocation   when logging hits line file:N, emit a stack trace (default :0)
      --log_dir string                   If non-empty, write log files in this directory
      --loglevel int                     Log level (0 = DEBUG, 5 = FATAL) (default 1)
      --logtostderr                      log to standard error instead of files
  -p, --profile string                   The name of the minikube VM being used.
        This can be modified to allow for multiple minikube instances to be run independently (default "minikube")
      --stderrthreshold severity         logs at or above this threshold go to stderr (default 2)
  -v, --v Level                          log level for V logs
      --vmodule moduleSpec               comma-separated list of pattern=N settings for file-filtered logging

Use "minikube [command] --help" for more information about a command.
```

9. You are now going to create a simple deployment based on nginx container image and expose it using a service.

10. Create a deployment and name it “nginx”

    kubectl run nginx --image=nginx --port=80

11. Expose the deployment using a service

    kubectl expose deployment nginx --type=NodePort

12. Check that your deployment as well as your service are ready

    kubectl get deployment

```
root@super-Virtual-Machine:~/labs/module5# kubectl get deployment
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx     1         1         1            1           28m
```

    kubectl get service

```
root@super-Virtual-Machine:~/labs/module5# kubectl get service
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        42m
nginx        NodePort    10.100.106.12   <none>        80:30891/TCP   28m
```

13. Access the nginx default web page using the curl command.

    curl $(minikube service nginx --url)

```
root@minikube-vm:~# curl $(minikube service nginx --url)
<!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>
```

14. Clean up the deployment and service with the following commands

    kubectl delete service nginx

    kubectl delete deployment nginx

Exercise 2: Working with Replica Sets, Deployments and Health Probes

Task

1. Working with the health probe

In this task you will create a new pod and enable a health probe. To test the probe, pod will run a single container that is going to explicitly fail the health probe request after every 5 probes.

1. You should still be in the /labs/module5 folder, if not, navigate to there.
2. Create the pod using the yaml file:

   kubectl apply -f liveness-probe.yaml
3. Check the status of the newly created pod. It may take few seconds for the container to be up and running.

   kubectl get pods

Notice the STATUS column shows Running and RESTARTS column have the value zero. That's expected because container is just started, and the health probe has not failed yet.

```
NAME              READY     STATUS    RESTARTS   AGE
liveness-probe    1/1       Running   0          3s
```

4. After 3-4 minutes if you view the status of pods again you should see the RESTARTS column with the value 1 (or higher depending on how long you have waited to check the status of the pod)

```
NAME              READY     STATUS    RESTARTS   AGE
liveness-probe    1/1       Running   1          2m
```

If you wait for few more minutes and check the status of pod again you should see the value of RESTARTS column changes to a higher number.

5. Behind the scenes, every time a container fails the health probe it will be restarted again. To get bit more information about the failing health probe run the following command:

kubectl describe po liveness-probe

This describes the pod in detail along with the events that are happening including the failed health probes

```
Events:
  Type     Reason                 Age               From               Message
  ----     ------                 ----              ----               -------
  Normal   Scheduled              5m                default-scheduler  Successfully assigned liveness-probe to minikube
  Normal   SuccessfulMountVolume  5m                kubelet, minikube  MountVolume.SetUp succeeded for volume "default-token-nmg4m"
  Normal   Pulled                 1m (x3 over 5m)   kubelet, minikube  Successfully pulled image "rbinrais/healthcheck"
  Normal   Created                1m (x3 over 5m)   kubelet, minikube  Created container
  Normal   Started                1m (x3 over 5m)   kubelet, minikube  Started container
  Warning  Unhealthy              32s (x9 over 4m)  kubelet, minikube  Liveness probe failed: HTTP probe failed with statuscode: 500
  Normal   Pulling                2s (x4 over 5m)   kubelet, minikube  pulling image "rbinrais/healthcheck"
  Normal   Killing                2s (x3 over 3m)   kubelet, minikube  Killing container with id docker://liveness-probe:Container failed
```

6. Eventually after failing the health probes multiple times in a short interval container will be put under "CrashLoopBackOff" status.

```
NAME              READY     STATUS             RESTARTS   AGE
liveness-probe    0/1       CrashLoopBackOff   6          12m
```

7. You can view the logs from the container that is terminated by using the command:

kubectl logs liveness-probe --previous

The sample docker container application is basic so very limited information is available in logs but typically for production ready applications its recommended to write more detailed messages to the logs.

8. Finally, remove the pod

kubectl delete pod liveness-probe

2. Working with Replica Set

1. In this task you will first create a replica with predefined labels assigned to pods. Later you will change the labels for a pod and observer behavior of replica set.

   A **Replica Set** ensures how many replicas of a pod should be running. It can be considered as a replacement of replication controller. The key difference between the replica set and the replication controller is, the replication controller only supports equality-based selector whereas the replica set supports set-based selector.

   The nginx-rc.yaml file is available inside the /labs/module5 subfolder and contains definition of replica set. If you review the content of file you will notice that it will maintain 3 pods with each running nginx container. Pods are also labeled app=webapp.

   To create the replica set and pods run the following command in the **labs/module5 directory**.

   kubectl create -f nginx-rc.yaml

2. Let's look at the pods along with their labels.

   kubectl get pods --show-labels

```
NAME                    READY     STATUS    RESTARTS   AGE       LABELS
nginx-replica-2ggkk     1/1       Running   0          2m        app=webapp
nginx-replica-kqdmn     1/1       Running   0          2m        app=webapp
nginx-replica-s5r8j     1/1       Running   0          2m        app=webapp
```

   You can also list all the replica sets that are available by using the command:

   kubectl get replicaset

```
NAME              DESIRED   CURRENT   READY     AGE
nginx-replica     3         3         3         36m
```

3. Notice we have three pods running. If you delete one of them, replica set will ensure that total pods count remain three and it will do that by creating a new pod.

   First delete one of the pods (get name from this command: kubectl get pods --show-labels)

   kubectl delete pod <<pod-name>>

Now, check the pods again. Notice you still have three pods running and one of them is terminating.

kubectl get pods --show-labels

```
NAME                    READY     STATUS        RESTARTS   AGE       LABELS
nginx-replica-2ggkk     0/1       Terminating   0          10m       app=webapp
nginx-replica-f7xsm     1/1       Running       0          3s        app=webapp
nginx-replica-kqdmn     1/1       Running       0          10m       app=webapp
nginx-replica-s5r8j     1/1       Running       0          10m       app=webapp
```

4. Another factor that plays an important role in determining pods relationship with replica set is the labels. Currently app=webapp is the selector used by replica set to determine the pods under its watch. If you change the label of a pod from app=webapp to say app=debugging then replica set will effectively remove it from its watch and create another pod with the label app=webapp. For replica set its job is to maintain the total count of pods to three as per the definition provided in the yaml file.

   kubectl label pod <<pod-name>> app=debugging --overwrite=true

5. View the pods again and notice that there are four pods running. Replica set created an additional pod immediately after it noticed pod count was less than three.

   kubectl get pods --show-labels

```
NAME                    READY     STATUS    RESTARTS   AGE       LABELS
nginx-replica-f7xsm     1/1       Running   0          16m       app=debugging
nginx-replica-kqdmn     1/1       Running   0          27m       app=webapp
nginx-replica-s5r8j     1/1       Running   0          27m       app=webapp
nginx-replica-v4vcb     1/1       Running   0          1m        app=webapp
```

6. Replica set is essentially using selector (defined in the yaml file) to which pods to observe. In this case its label *app* matching value *webapp*. You can also get all the pods with app=webapp label using the following command.

   kubectl get pods --show-labels -l app=webapp

7. Finally remove the replica set using the following command.

   kubectl delete replicaset nginx-replica

8. As part of the deletion process replica set will remove all the pods that it had created. You can see that by listing the pods and looking at the STATUS column which shows Terminating.

   kubectl get pods

```
NAME                    READY     STATUS        RESTARTS   AGE
nginx-replica-f7xsm     1/1       Running       0          29m
nginx-replica-kqdmn     0/1       Terminating   0          40m
nginx-replica-s5r8j     0/1       Terminating   0          40m
nginx-replica-v4vcb     0/1       Terminating   0          14m
```

Eventually pods will be removed. However, if you list the pods again the pod with label app=debugging is still Running.

kubectl get pods --show-labels

```
NAME                    READY     STATUS    RESTARTS   AGE       LABELS
nginx-replica-f7xsm     1/1       Running   0          34m       app=debugging
```

Since you have change the label this pod is no longer manage by the replica set. In cases like these you can bulk remove pods based on labels.

kubectl delete pods -l app=debugging

3. Working with Deployments

1. In this task you will begin by performing a deployment based on specific version of the nginx container image (v 1.7.9). Later you will leverage a RollingUpdate strategy for deployment to update pods running nginx container image from v1.7.9 to container image 1.8.

   The nginx-deployment.yaml file is available inside the /labs/module5 subfolder and contains definition of deployment. If you review the content of file you will notice that it will maintain 2 pods with each running nginx container image v1.7.9. Pods are also labeled app=nginx.

   Run the following command from the labs/module5 directory:

   kubectl create -f nginx-deployment.yaml

2. Notice the deployment status by running the command:

   kubectl get deployment

```
NAME                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment        2         2         2            2           3m
```

3. If you list the pods you should see the out similar to following:

   kubectl get pods --show-labels

```
NAME                                    READY     STATUS    RESTARTS   AGE       LABELS
nginx-deployment-6d8f46cfb7-qhnpz       1/1       Running   0          5m        app=nginx,pod-template-hash=2849027963
nginx-deployment-6d8f46cfb7-vb45t       1/1       Running   0          5m        app=nginx,pod-template-hash=2849027963
```

Notice the LABELS column and presence of pod-template-hash label. This label is used by the deployment during the update process.

4. You are now going to update the deployment. You are going to update nginx container image from v 1.7.9 to v1.8. Before you do that first check the existing definition of the deployment:

   kubectl describe deployment nginx-deployment

```
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Sat, 20 Jan 2018 17:13:59 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision=1
Selector:               app=nginx
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.7.9
    Port:         80/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
```

   Notice the line *Image: nginx:1.7.9* which confirms that the current deployment is using 1.7.9 version of nginx image.

5. Perform the update using the command below.

   kubectl apply -f nginx-deployment-updated.yaml

   If you review the content of nginx-deployment-updated.yaml file and compare it with original nginx-deployment-updated.yaml the only difference is the image tag which is changed from 1.7.9 to 1.8.

6. If you immediately (after step 5) run the command to list all the pods you should see output like following:

   kubectl get pods --show-labels

```
NAME                                READY     STATUS        RESTARTS   AGE       LABELS
nginx-deployment-6d8f46cfb7-qhnpz   0/1       Terminating   0          14m       app=nginx,pod-template-hash=2849027963
nginx-deployment-6d8f46cfb7-vb45t   0/1       Terminating   0          14m       app=nginx,pod-template-hash=2849027963
nginx-deployment-784794c74c-h7sv6   1/1       Running       0          7s        app=nginx,pod-template-hash=3403507307
nginx-deployment-784794c74c-pbsqm   1/1       Running       0          9s        app=nginx,pod-template-hash=3403507307
```

Notice that the deployment strategy of rolling update ensures that the old pods (nginx v 1.7.9) are terminated only after new pods (nginx image v 1.8) are in a running state. Also notice that the label pod-template-hash values are different for old and new pods. This is because the pod definition (due to change of image tag) is not same for both deployments.

7. You can also look at the new deployment details and make sure that correct nginx image (v 1.8) is used.

   kubectl describe deployment nginx-deployment

```
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Sat, 20 Jan 2018 17:13:59 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision=2
                        kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1beta1",
nginx-deployment","namespace":"default"},"spec":{"replicas":2,"se...
Selector:               app=nginx
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.8
    Port:         80/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-deployment-784794c74c (2/2 replicas created)
```