

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

Containers and Kubernetes

LAB - 1

Kubernetes, Docker Registry and downloading lab files

Step 1: Kubernetes and your Docker Registry

There should be a registry running that we will push our images to:

```
echo $REGISTRY_HOST
```

Step 2: Download lab files and code

For convenience all files, scripts, etc that we will be using for the remainder of this course have been packaged up into an archive.

```
cd $HOME && curl -s http://$WORKSHOP_NAMESPACE-files/_static/lab-files.tar.gz | tar -xvz
```

1.3. Build Docker image for your frontend application

Step 1: Analyze Dockerfile for your frontend application

```
cd $HOME/gowebapp/gowebapp
```

Let's inspect the `Dockerfile`. You can do so by opening it in your favorite command line editor, use the built-in editor, or we have embedded the file directly in the lab instructions here for easy viewing. This Dockerfile contains the instructions to tell the Docker build system how to create an image for our webapp.

```
[-] $ echo $REGISTRY_HOST
registry-kubeacademy-w52-s186.acad-kube-prd1.labs.kube.academy
[-] $ cd $HOME && curl -s http://$WORKSHOP_NAMESPACE-files/_static/lab-files.tar.gz | tar -xvz
./
./gowebapp/
./gowebapp/gowebapp-mysql/
./gowebapp/gowebapp-mysql/Dockerfile
./gowebapp/gowebapp-mysql/gowebapp.sql
./gowebapp/gowebapp-mysql/gowebapp-mysql-deployment.yaml
./gowebapp/gowebapp-mysql/gowebapp-mysql-service.yaml
./gowebapp/gowebapp/
./gowebapp/gowebapp/gowebapp-service.yaml
./gowebapp/gowebapp/gowebapp-deployment.yaml
./gowebapp/gowebapp/Dockerfile
./gowebapp/gowebapp/code/
./gowebapp/gowebapp/code/static/
./gowebapp/gowebapp/code/static/js/
./gowebapp/gowebapp/code/static/js/bootstrap.min.js
./gowebapp/gowebapp/code/static/js/jquery1.11.0.min.js
./gowebapp/gowebapp/code/static/js/underscore.min.js
./gowebapp/gowebapp/code/static/js/custom-elements.min.js
./gowebapp/gowebapp/code/static/js/global.js
./gowebapp/gowebapp/code/static/js/underscore-min.js
./gowebapp/gowebapp/code/static/js/cldr-icons.min.js
./gowebapp/gowebapp/code/static/fonts/
./gowebapp/gowebapp/code/static/fonts/glyphicons-halflings-regular.ttf
./gowebapp/gowebapp/code/static/fonts/glyphicons-halflings-regular.woff
./gowebapp/gowebapp/code/static/fonts/glyphicons-halflings-regular.eot
./gowebapp/gowebapp/code/static/fonts/glyphicons-halflings-regular.woff2
./gowebapp/gowebapp/code/static/fonts/glyphicons-halflings-regular.svg
./gowebapp/gowebapp/code/static/css/
./gowebapp/gowebapp/code/static/css/cldr-ui.min.css
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

Building different docker images for application

The screenshot displays the KubeAcademy interface for Lab 01: Containerize Applications. The left sidebar shows the course structure with 'Current lesson: Lab 01: Containerize Applications' selected. The main content area is divided into two sections: 'Step 2: Build gowebapp Docker image locally' and '1.4. Build Docker image for your backend application'. The 'Step 2' section includes a code block for the command `cd $HOME/gowebapp/gowebapp` and a text instruction: 'Build the gowebapp image locally. Make sure to include "*" at the end. Make sure the build runs to completion without errors. You should get a success message.' Below this, a code block shows the command `docker build -t gowebapp:v1 .`. The '1.4. Build Docker image for your backend application' section includes a code block for `cd $HOME/gowebapp/gowebapp-mysql` and a text instruction: 'Let's inspect the Dockerfile. You can do so by opening it in your favorite command line editor, use the built-in editor, or we have embedded the file directly in the lab instructions here for easy viewing. This Dockerfile contains the instructions to tell the Docker build engine how to create an image for the backend MySQL database.' A 'Dockerfile' link is provided. The right side of the interface features a 'Terminal' window showing the execution of the `docker build` command. The terminal output shows the build process, including the transfer of the Dockerfile, the build context, and the final image creation. The terminal output is as follows:

```
~/gowebapp/gowebapp/code/gowebapp
~/gowebapp/gowebapp/config/
~/gowebapp/gowebapp/config/config.json
[~] $
[~] $
[~] $ cd $HOME/gowebapp/gowebapp
~/gowebapp/gowebapp $
~/gowebapp/gowebapp $
~/gowebapp/gowebapp $ docker build -t gowebapp:v1 .
[+] Building 2.6s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 204B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:impish
=> [1/4] FROM docker.io/library/ubuntu:impishsha256:ff46b78279f207db3b8e57e20dee7cecef3567d09489369d88591f150f9c8154
=> resolve docker.io/library/ubuntu:impishsha256:ff46b78279f207db3b8e57e20dee7cecef3567d09489369d88591f150f9c8154
=> sha256:58398b9058c55fa969c8ab5db6a11ff52686a9b6738884a9113d164dc567dfac 1.46kB / 1.46kB
=> sha256:54b8fda3c9de3a40f0891ce20de55fc92e825315021192957f1bfe3fc807d7d 30.38MB / 30.38MB
=> sha256:ff46b78279f207db3b8e57e20dee7cecef3567d09489369d88591f150f9c8154 1.42kB / 1.42kB
=> sha256:240a11a56faca317f3355b143b5cc2791ba60ad0edd393b3ae3c85c29cad1a6c 529B / 529B
=> extracting sha256:54b8fda3c9de3a40f0891ce20de55fc92e825315021192957f1bfe3fc807d7d
=> [internal] load build context
=> transferring context: 14.74MB
=> [2/4] COPY ./code /opt/gowebapp
=> [3/4] COPY ./config /opt/gowebapp/config
=> [4/4] WORKDIR /opt/gowebapp/
=> exporting to image
=> exporting layers
=> writing image sha256:47f02f4bf94ba3a241ff66b77e8f1613b4aa5fb5356e3ae2a10ceebd73f9b15b
=> naming to docker.io/library/gowebapp:v1
[~/gowebapp/gowebapp] $
```

Creating docker network and launching its containers

7:33:38 PM ↑ 0.0 b ↓ 3.0 k

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 51

Microservices Course

Testing the application container mysql and trying sql commands

The screenshot displays the KubeAcademy website interface for the course 'Lab 01: Containerize Applications'. The page includes a navigation bar with 'Courses', 'Learning Paths', and 'Instructors'. The main content area shows the current lesson, 'Lab 01: Containerize Applications', with a 'Give Feedback' button and a 'Mark as Complete' button. The terminal window is open, showing the following commands and output:

```
docker exec -it gowebapp-mysql mysql -u root -pmysqlpassword gowebapp
mysql>
mysql> sleep 20
mysql>
mysql> docker run -p 8880:8880 --net gowebapp -d --name gowebapp --hostname gowebapp gowebapp:v1
294487aaf864bf67b285d6cc6f7b6cc593f157cb3ca8783ce7b2657637553
mysql>
```

The terminal output shows the MySQL server starting up and listening on port 3306. The user 'root' is connected to the 'mysql' database. The terminal also shows the command to create a new database 'gowebapp' and the command to run the 'gowebapp' container.

Step 3: Test the application locally

Now that we've launched the application containers, let's try to test the web application locally.

You should be able to access the application by running the following command in the terminal and then clicking the URL it produces.

```
echo "http://$SESSION_NAME-gowebapp-docker.$INGRESS_DOMAIN"
```

Create an account and login. Write something on your Notepad and save it. This will verify that the application is working and properly integrates with the backend database container.

Step 4: Inspect the MySQL database

Let's connect to the backend MySQL database container and run some queries to ensure that application persistence is working properly:

```
docker exec -it gowebapp-mysql mysql -u root -pmysqlpassword gowebapp
```

Once connected, run some simple SQL commands to inspect the database tables and persistence:

```
#Simple SQL to navigate
SHOW DATABASES;
USE gowebapp;
SHOW TABLES;
SELECT * FROM table_name;
exit;
```

Step 5: Cleanup application containers

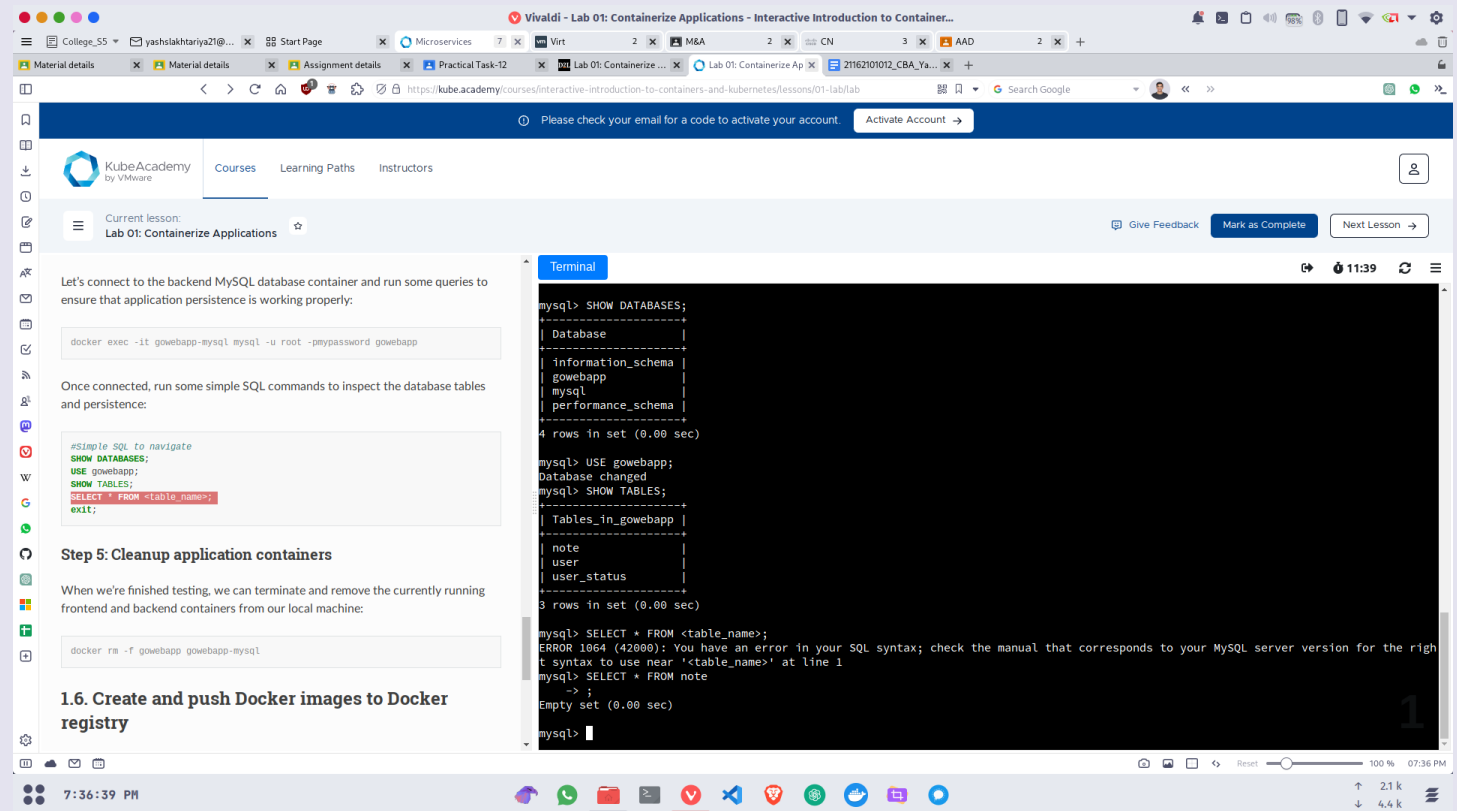
When we're finished testing, we can terminate and remove the currently running frontend and backend containers from our local machine:

```
docker rm -f gowebapp gowebapp-mysql
```

1.6. Create and push Docker images to Docker registry

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

Removing and cleaning up files



The screenshot shows a web browser window with the KubeAcademy course page for 'Lab 01: Containerize Applications'. The page includes instructions for connecting to a MySQL database container and a terminal window showing SQL commands and their output.

Current lesson: Lab 01: Containerize Applications

Let's connect to the backend MySQL database container and run some queries to ensure that application persistence is working properly:

```
docker exec -it gowebapp-mysql mysql -u root -ppassword gowebapp
```

Once connected, run some simple SQL commands to inspect the database tables and persistence:

```
#Simple SQL to navigate
SHOW DATABASES;
USE gowebapp;
SHOW TABLES;
SELECT * FROM <table_name>;
exit;
```

Step 5: Cleanup application containers

When we're finished testing, we can terminate and remove the currently running frontend and backend containers from our local machine:

```
docker rm -f gowebapp gowebapp-mysql
```

1.6. Create and push Docker images to Docker registry

Terminal

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| gowebapp |
| mysql |
| performance_schema |
+-----+
4 rows in set (0.00 sec)

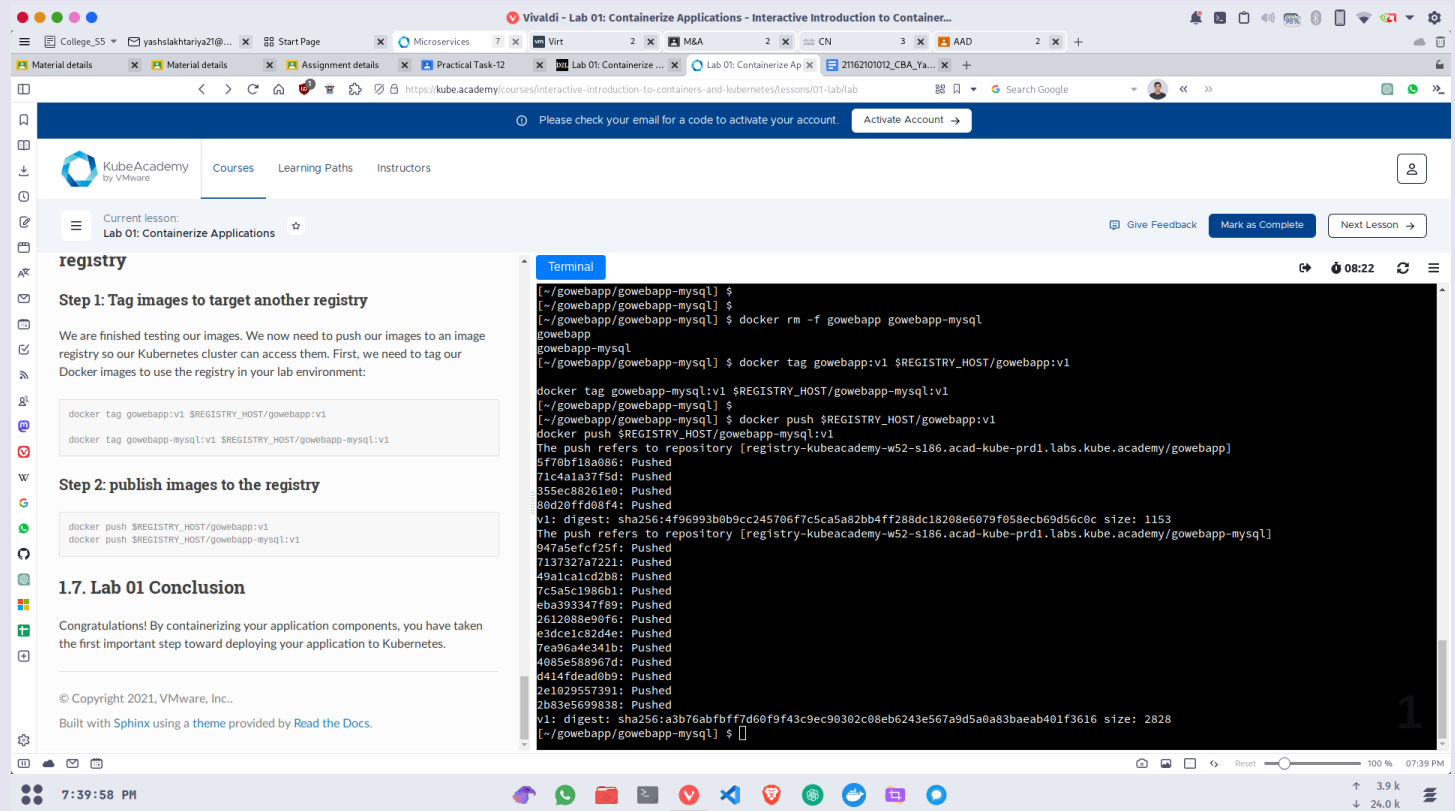
mysql> USE gowebapp;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_gowebapp |
+-----+
| note |
| user |
| user_status |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM <table_name>;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '<table_name>' at line 1
mysql> SELECT * FROM note
-> ;
Empty set (0.00 sec)

mysql>
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

Assigning tags and publishing to registry



registry

Step 1: Tag images to target another registry

We are finished testing our images. We now need to push our images to an image registry so our Kubernetes cluster can access them. First, we need to tag our Docker images to use the registry in your lab environment:

```
docker tag gowebapp:v1 $REGISTRY_HOST/gowebapp:v1
docker tag gowebapp-mysql:v1 $REGISTRY_HOST/gowebapp-mysql:v1
```

Step 2: publish images to the registry

```
docker push $REGISTRY_HOST/gowebapp:v1
docker push $REGISTRY_HOST/gowebapp-mysql:v1
```

1.7. Lab 01 Conclusion

Congratulations! By containerizing your application components, you have taken the first important step toward deploying your application to Kubernetes.

© Copyright 2021, VMware, Inc..
Built with Sphinx using a theme provided by Read the Docs.

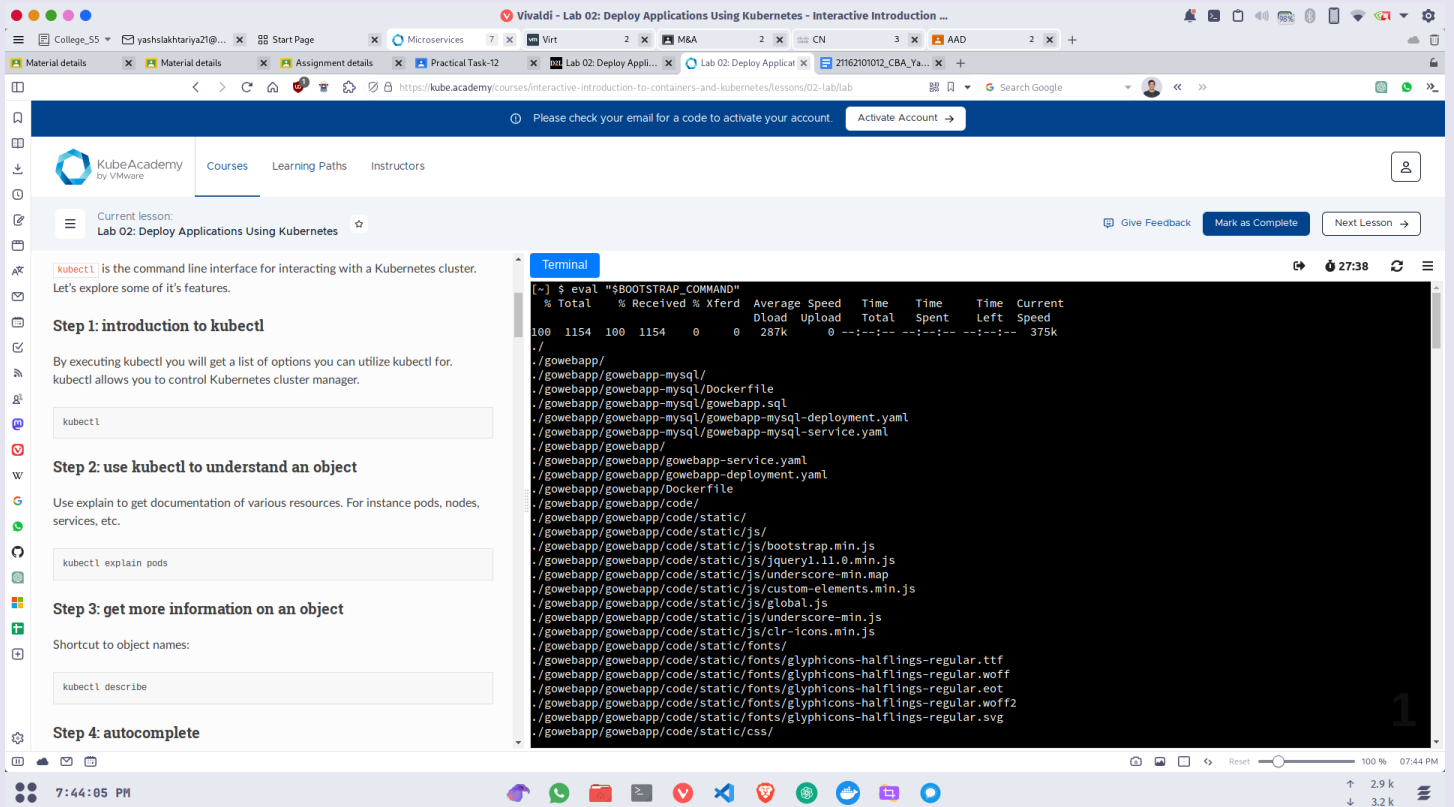
Terminal

```
[~/gowebapp/gowebapp-mysql] $
[~/gowebapp/gowebapp-mysql] $
[~/gowebapp/gowebapp-mysql] $ docker rm -f gowebapp gowebapp-mysql
gowebapp
gowebapp-mysql
[~/gowebapp/gowebapp-mysql] $ docker tag gowebapp:v1 $REGISTRY_HOST/gowebapp:v1
[~/gowebapp/gowebapp-mysql] $ docker tag gowebapp-mysql:v1 $REGISTRY_HOST/gowebapp-mysql:v1
[~/gowebapp/gowebapp-mysql] $ docker push $REGISTRY_HOST/gowebapp:v1
docker push $REGISTRY_HOST/gowebapp-mysql:v1
The push refers to repository [registry-kubeacademy-w52-s186.acad-kube-prd1.labs.kube.academy/gowebapp]
5f70bf18a086: Pushed
71c4a1a37f5d: Pushed
355ec88261e9: Pushed
80d20ff0d08f4: Pushed
v1: digest: sha256:4f96993b0b9cc245706f7c5ca5a82bb4ff288dc18208e6079f058ecb69d56c0c size: 1153
The push refers to repository [registry-kubeacademy-w52-s186.acad-kube-prd1.labs.kube.academy/gowebapp-mysql]
947a5efcf25f: Pushed
7137327a7221: Pushed
49a1ca1cd2b8: Pushed
7c5a5c1986b1: Pushed
eba393347f69: Pushed
2612888e90f6: Pushed
e3dce1c2824e: Pushed
7ea96a4e341b: Pushed
4085e588967d: Pushed
d414fdead0b9: Pushed
2e1029557391: Pushed
2b83e5699838: Pushed
v1: digest: sha256:a3b76abfbff7d60f9f43c9ec90302c08eb6243e567a9d5a0a83baeab401f3616 size: 2828
[~/gowebapp/gowebapp-mysql] $
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

LAB - 2

Trying kubectl command



The screenshot shows a web browser window with the URL <https://kube.academy/courses/interactive-introduction-to-containers-and-kubernetes/lessons/02-lab/lab>. The page is titled "Lab 02: Deploy Applications Using Kubernetes" and is part of a course by KubeAcademy by VMware. The sidebar on the left contains navigation links for "Courses", "Learning Paths", and "Instructors". The main content area is divided into four steps:

- Step 1: introduction to kubectl**
By executing kubectl you will get a list of options you can utilize kubectl for. kubectl allows you to control Kubernetes cluster manager.
- Step 2: use kubectl to understand an object**
Use explain to get documentation of various resources. For instance pods, nodes, services, etc.
- Step 3: get more information on an object**
Shortcut to object names:
- Step 4: autocomplete**

A terminal window is open on the right side of the page, showing the output of the `eval "$BOOTSTRAP_COMMAND"` command. The output is a table with columns: % Total, % Received, % Xferd, Average Speed, Time, Time, Time, Current, Dload, Upload, Total, Spent, Left, Speed. The table shows a download progress of 100% for a file of size 1154 bytes. The file is located at `./gwebapp/gowebapp/code/static/css/`.

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current	Dload	Upload	Total	Spent	Left	Speed
100	1154	100	1154	0	0	287k	0	--:--:--	--:--:--	--:--:--	--:--:--	--:--:--	375k

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

Using kubectl for describing and explaining its parts

2.1. Getting Started with kubectl

kubectl is the command line interface for interacting with a Kubernetes cluster. Let's explore some of its features.

Step 1: introduction to kubectl

By executing kubectl you will get a list of options you can utilize kubectl for. kubectl allows you to control Kubernetes cluster manager.

Step 2: use kubectl to understand an object

Use explain to get documentation of various resources. For instance pods, nodes, services, etc.

Step 3: get more information on an object

Shortcut to object names:

Terminal

```
Settings Commands:
label             Update the labels on a resource
annotate          Update the annotations on a resource
completion        Output shell completion code for the specified shell (bash, zsh, fish, or powershell)

Other Commands:
api-resources      Print the supported API resources on the server
api-versions       Print the supported API versions on the server, in the form of "group/version"
config            Modify kubeconfig files
plugin            Provides utilities for interacting with plugins
version           Print the client and server version information

Usage:
kubectl [flags] [options]

Use "kubectl <command> --help" for more information about a given command.
Use "kubectl options" for a list of global command-line options (applies to all commands).

[1] $ kubectl explain pods
KIND:      Pod
VERSION:   v1

DESCRIPTION:
Pod is a collection of containers that can run on a host. This resource is
created by clients and scheduled onto hosts.

FIELDS:
apiVersion    <string>
APIVersion defines the versioned schema of this representation of an object.
Servers should convert recognized schemas to the latest internal value, and
may reject unrecognized values. More info:
https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources
```

Step 2: use kubectl to understand an object

Use explain to get documentation of various resources. For instance pods, nodes, services, etc.

Step 3: get more information on an object

Shortcut to object names:

Step 4: autocomplete

Use TAB to autocomplete:

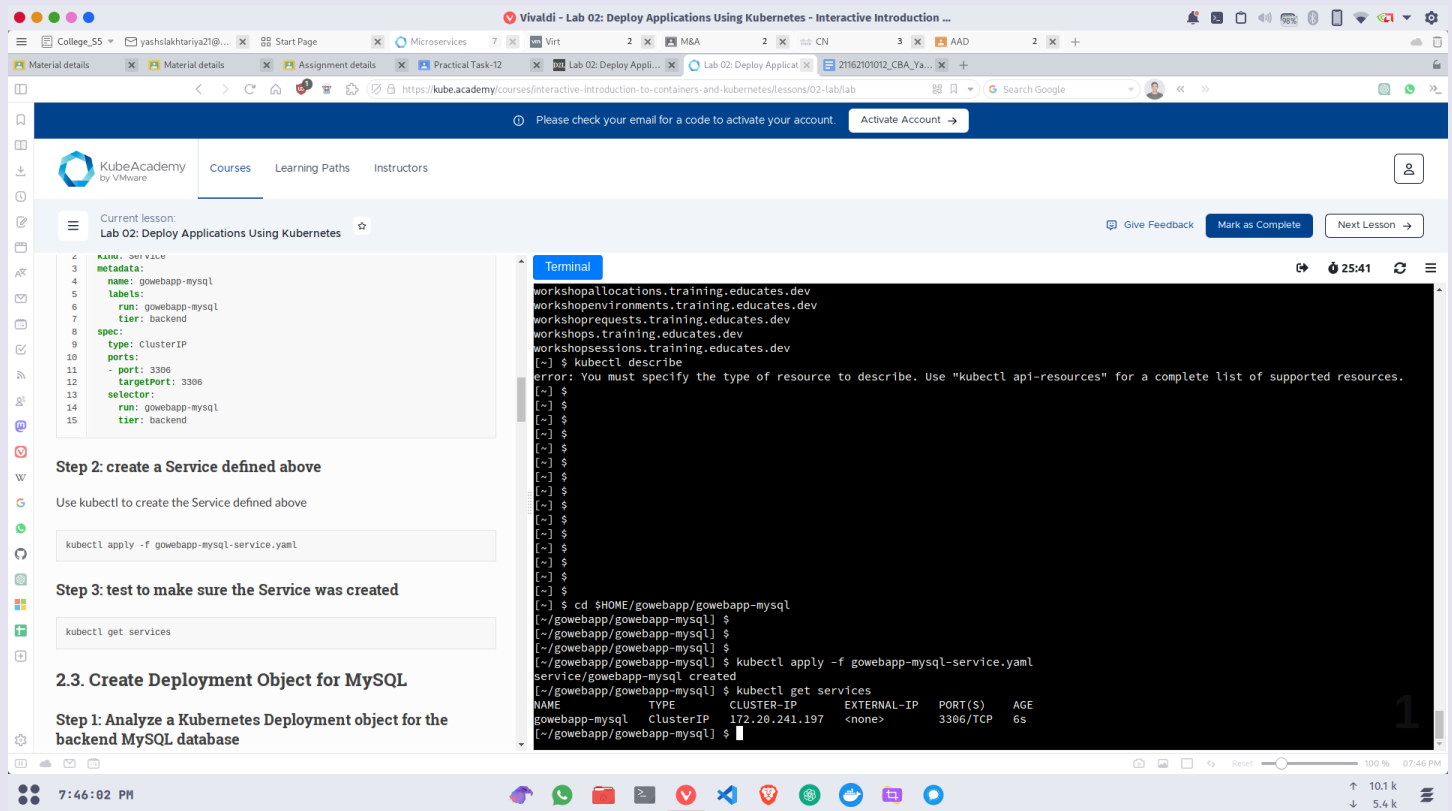
2.2. Create Service Object for MySQL

Step 1: Analyze a Kubernetes Service object for the backend MySQL database

Terminal

```
[-] $ kubectl describe
error: You must specify the type of resource to describe. Use "kubectl api-resources" for a complete list of supported resources.
[-] $ kubectl describe
Display all 132 possibilities? (y or n)
admissionreports.kyverno.io
agents.clusters.tmc.cloud.vmware.com
apiservices.apiregistration.k8s.io
apps.kappctrl.k14s.io
backgroundscanreports.kyverno.io
certificaterquests.cert-manager.io
certificates.cert-manager.io
certificates.secretgen.k14s.io
certificatesigningrequests.certificates.k8s.io
challenges.acme.cert-manager.io
cleanuppolicies.kyverno.io
clusteradmissionreports.kyverno.io
clusterbackgroundscanreports.kyverno.io
clustercleanuppolicies.kyverno.io
clusterissuers.cert-manager.io
clusterpolicies.intents.tmc.cloud.vmware.com
clusterpolicies.kyverno.io
clusterpolicyreports.wgpolicyk8s.io
clusterrolebindings.rbac.authorization.k8s.io
clusterroles.rbac.authorization.k8s.io
cninodes.vpcresources.k8s.aws
componentstatuses
configmaps
contourconfigurations.projectcontour.io
contourdeployments.projectcontour.io
controllerrevisions.apps
```


Creating and checking the service via yaml file of container



Creating and checking deployments and pods

Vivaldi - Lab 02: Deployment Applications Using Kubernetes - Interactive Introduction ...

College_55 | yashlakhtariya21@... | Start Page | Microservices | Virt | M&A | CN | AAD

Material details | Material details | Assignment details | Practical Task-12 | Lab 02: Deploy Appli... | Lab 02: Deploy Appli... | 21162701012_CBA_Ya... | +

https://kubernetes.io/docs/tutorials/kubernetes-basics/lab02-lablab

Please check your email for a code to activate your account. Activate Account →

KubeAcademy
by VMware

Courses Learning Paths Instructors

Current lesson:
Lab 02: Deploy Applications Using Kubernetes ☆

Give Feedback Mark as Complete Next Lesson →

Step 3: test to make sure the Deployment was created

kubectl get deployments

We can also look at the pods that were created by the deployment by having kubectl get us all pods. Note that the **STATUS** field returned may take some time to get to the value we're looking for which is **running**.

kubectl get pods

2.4. Create Service Object for frontend application: gowebapp

Step 1: Analyze a Kubernetes Service object for the frontend gowebapp

cd \$HOME/gowebapp/gowebapp

Let's inspect the **gowebapp-service.yaml** file. You can do so by opening it in your favorite command line editor, use the built-in editor, or we have embedded this file directly in the lab instructions here for easy viewing. This Kubernetes configuration file contains the definition of what the desired state should be for our Kubernetes Service object for the frontend gowebapp.

Terminal

```
[~] $  
[~] $  
[~] $  
[~] $  
[~] $  
[~] $  
[~] $  
[~] $  
[~] $  
[~] $ cd $HOME/gowebapp/gowebapp-mysql  
[~/gowebapp/gowebapp-mysql] $  
[~/gowebapp/gowebapp-mysql] $  
[~/gowebapp/gowebapp-mysql] $  
[~/gowebapp/gowebapp-mysql] $ kubectl apply -f gowebapp-mysql-service.yaml  
service/gowebapp-mysql created  
[~/gowebapp/gowebapp-mysql] $ kubectl get services  
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE  
gowebapp-mysql ClusterIP   172.20.241.197 <none>        3306/TCP    6s  
[~/gowebapp/gowebapp-mysql] $  
[~/gowebapp/gowebapp-mysql] $  
[~/gowebapp/gowebapp-mysql] $ sed -i s/"image: gowebapp"/"image: $REGISTRY_HOST/gowebapp/g_gowebapp-mysql-deployment.yaml  
[~/gowebapp/gowebapp-mysql] $  
[~/gowebapp/gowebapp-mysql] $ kubectl apply -f gowebapp-mysql-deployment.yaml  
deployment.apps/gowebapp-mysql created  
[~/gowebapp/gowebapp-mysql] $  
[~/gowebapp/gowebapp-mysql] $ kubectl get deployments  
NAME      READY   UP-TO-DATE   AVAILABLE   AGE  
gowebapp-mysql 1/1     1            1           7s  
[~/gowebapp/gowebapp-mysql] $
```

7:47:45 PM | 6.0 k / 11.3 k

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 51

Microservices Course

Repeating the process for another application

Step 1: Analyze a Kubernetes Deployment object for the frontend gowebapp

cd \$HOME/gowebapp/gowebapp

Let's inspect the `gowebapp-deployment.yaml` file. You can do so by opening it in your favorite command line editor, use the built-in editor, or we have embedded the file directly in the lab instructions here for easy viewing. This Kubernetes configuration file contains the definition of what the desired state should be for our Kubernetes Deployment object for the frontend gowebapp.

[gowebapp-deployment.yaml](#)

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: gowebapp
5    labels:
6      run: gowebapp
7      tier: frontend
8  spec:
9    replicas: 2
10   selector:
11     matchLabels:
12       run: gowebapp
13       tier: frontend
14   template:
15     metadata:
16       labels:
17         run: gowebapp
```

Terminal

```
~/gowebapp/gowebapp-mysql $ kubectl apply -f gowebapp-mysql-service.yaml
service/gowebapp-mysql created
~/gowebapp/gowebapp-mysql $ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
gowebapp-mysql ClusterIP    172.20.241.197 <none>        3306/TCP    6s
~/gowebapp/gowebapp-mysql $ kubectl apply -f gowebapp-mysql-deployment.yaml
deployment.apps/gowebapp-mysql created
~/gowebapp/gowebapp-mysql $ kubectl get deployments
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
gowebapp-mysql 1/1     1             1           7s
~/gowebapp/gowebapp-mysql $ cd $HOME/gowebapp/gowebapp
~/gowebapp/gowebapp $ kubectl apply -f gowebapp-service.yaml
service/gowebapp created
~/gowebapp/gowebapp $ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
gowebapp      NodePort     172.20.151.77 <none>        8080:30341/TCP 6s
gowebapp-mysql ClusterIP    172.20.241.197 <none>        3306/TCP    2m17s
```

Step 2: create the Deployment defined above

Use kubectl to create the Deployment defined above

```
kubectl apply -f gowebapp-mysql-deployment.yaml
```

Step 3: test to make sure the Deployment was created

```
kubectl get deployments
```

We can also look at the pods that were created by the deployment by having kubectl get us all pods. Note that the `STATUS` field returned may take some time to get to a value we're looking for which is `Running`.

```
kubectl get pods
```

2.4. Create Service Object for frontend application: gowebapp

Step 1: Analyze a Kubernetes Service object for the frontend gowebapp

```
~/gowebapp/gowebapp-mysql $ kubectl apply -f gowebapp-mysql-deployment.yaml
deployment.apps/gowebapp-mysql created
~/gowebapp/gowebapp-mysql $ kubectl get deployments
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
gowebapp-mysql 1/1     1             1           7s
~/gowebapp/gowebapp-mysql $ cd $HOME/gowebapp/gowebapp
~/gowebapp/gowebapp $ kubectl apply -f gowebapp-service.yaml
service/gowebapp created
~/gowebapp/gowebapp $ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
gowebapp      NodePort     172.20.151.77 <none>        8080:30341/TCP 6s
gowebapp-mysql ClusterIP    172.20.241.197 <none>        3306/TCP    2m17s
~/gowebapp/gowebapp $ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
gowebapp-mysql-7946c97b64-ng7qm    1/1     Running   0           3m45s
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

Vivaldi - Lab 02: Deploy Applications Using Kubernetes - Interactive Introduction ...

College_S5 yashlakhtariya21@... Start Page Microservices 7 Lab 02: Deploy Appl... 2 M&A 2 CN 3 AAD 2 +

Material details Material details Assignment details Practical Task-12 Lab 02: Deploy Appl... 21162101012_CBA_Ya... +

https://kubernetes.io/docs/concepts/cluster-administration/permissions/ Search Google

Please check your email for a code to activate your account. [Activate Account](#)

KubeAcademy by VMware

Courses Learning Paths Instructors

Current lesson: Lab 02: Deploy Applications Using Kubernetes [Give Feedback](#) [Mark as Complete](#) [Next Lesson](#)

Use kubectl to create the service defined above

```
kubectl apply -f gowebapp-deployment.yaml
```

Step 3: test to make sure the Deployment was created

```
kubectl get deployment
```

We can also look at the pods that were created by the deployment by having kubectl get us all pods. This time we should see the database pod as well as the two gowebapp pods that were created. Note that the **STATUS** field returned may take some time to get to a the value we're looking for which is **Running**.

```
kubectl get pods
```

2.6. Test Your Application

Step 1: Access gowebapp through the Service

You should be able to access the application by running the following command in the terminal and then clicking the URL it produces.

```
echo "http://$SESSION_NAME-gowebapp-k8s.$INGRESS_DOMAIN"
```

Terminal

```
[~/gowebapp/gowebapp] $  
[~/gowebapp/gowebapp] $ kubectl get pods  
NAME READY STATUS RESTARTS AGE  
gowebapp-mysql-7946c97b64-ng7qm 1/1 Running 0 3m45s  
[~/gowebapp/gowebapp] $  
[~/gowebapp/gowebapp] $  
[~/gowebapp/gowebapp] $ kubectl apply -f gowebapp-service.yaml  
service/gowebapp unchanged  
[~/gowebapp/gowebapp] $ kubectl get services  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
gowebapp NodePort 172.20.151.77 <none> 8080:30341/TCP 3m56s  
gowebapp-mysql ClusterIP 172.20.241.197 <none> 3306/TCP 6m7s  
[~/gowebapp/gowebapp] $  
[~/gowebapp/gowebapp] $ sed -i s/"image: gowebapp"/"image: $REGISTRY_HOST/gowebapp"/g gowebapp-deployment.yaml  
[~/gowebapp/gowebapp] $ kubectl apply -f gowebapp-deployment.yaml  
deployment.apps/gowebapp created  
[~/gowebapp/gowebapp] $  
[~/gowebapp/gowebapp] $ kubectl get deployment  
NAME READY UP-TO-DATE AVAILABLE AGE  
gowebapp 2/2 2 2 6s  
gowebapp-mysql 1/1 1 1 4m47s  
[~/gowebapp/gowebapp] $  
[~/gowebapp/gowebapp] $ kubectl get pods  
NAME READY STATUS RESTARTS AGE  
gowebapp-7cf5c6cf49-4q892 1/1 Running 0 10s  
gowebapp-7cf5c6cf49-qh6z8 1/1 Running 0 10s  
gowebapp-mysql-7946c97b64-ng7qm 1/1 Running 0 4m52s  
[~/gowebapp/gowebapp] $
```

7:52:30 PM

0.0 b
3.4 k

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

Accessing the application through the service

2.0. Test Your Application

Step 1: Access gowebapp through the Service

You should be able to access the application by running the following command in the terminal and then clicking the URL it produces.

```
echo "http://$SESSION_NAME-gowebapp-k8s.$INGRESS_DOMAIN"
```

You can now sign up for an account, login and use the Notepad.

Warning

Note: Your browser may cache an old instance of the gowebapp application from previous labs. When the webpage loads, look at the top right. If you see 'Logout', click it. You can then proceed with creating a new test account.

2.7. Lab 02 Conclusion

Congratulations! You have successfully deployed your applications using Kubernetes!

© Copyright 2021, VMware, Inc.
Built with Sphinx using a theme provided by Read the Docs.

Terminal

```
~/gowebapp/gowebapp] $  
~/gowebapp/gowebapp] $ kubectl apply -f gowebapp-service.yaml  
service/gowebapp unchanged  
~/gowebapp/gowebapp] $ kubectl get services  
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE  
gowebapp  NodePort  172.20.151.77  <none>         8080:30341/TCP   3m56s  
gowebapp-mysql  ClusterIP  172.20.241.197  <none>         3306/TCP         6m7s  
~/gowebapp/gowebapp] $  
~/gowebapp/gowebapp] $ sed -i s/"image: gowebapp"/"image: $REGISTRY_HOST/gowebapp"/g gowebapp-deployment.yaml  
~/gowebapp/gowebapp] $ kubectl apply -f gowebapp-deployment.yaml  
deployment.apps/gowebapp created  
~/gowebapp/gowebapp] $  
~/gowebapp/gowebapp] $ kubectl get deployment  
NAME      READY  UP-TO-DATE  AVAILABLE  AGE  
gowebapp  2/2    2            2          6s  
gowebapp-mysql  1/1    1            1          4m47s  
~/gowebapp/gowebapp] $  
~/gowebapp/gowebapp] $ kubectl get pods  
NAME                                READY  STATUS   RESTARTS  AGE  
gowebapp-7cfc6cf49-4q892            1/1    Running  0          10s  
gowebapp-7cfc6cf49-qh6z8            1/1    Running  0          10s  
gowebapp-mysql-7946c97b64-ng7qm     1/1    Running  0          4m52s  
~/gowebapp/gowebapp] $  
~/gowebapp/gowebapp] $ echo "http://$SESSION_NAME-gowebapp-k8s.$INGRESS_DOMAIN"  
http://kubeademy-w51-s179-gowebapp-k8s.acad-kube-prd1.labs.kube.academy  
~/gowebapp/gowebapp] $  
~/gowebapp/gowebapp] $  
~/gowebapp/gowebapp] $
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

Course completion

The screenshot shows a web browser window with the title "Vivaldi - Homepage - Containers and Kubernetes". The address bar shows the URL "https://itacademy.brightspace.com/d2l/home/6722". The page content includes a header "Containers and Kubernetes" and a "Visual Table of Contents" section. The table lists four modules, all with a completion status of 100%.

Module	Completion Status
Welcome Modern Apps Learning Path	100% 1 of 1 Topics Completed
Module 1. Containers 101	100% 4 of 4 Topics Completed
Module 2. Why Kubernetes	100% 6 of 6 Topics Completed
Module 03. Kubernetes Fundamentals	100% 1 of 1 Topics Completed

The "Welcome" section on the right contains a message: "Relaunch the Welcome Window to review the information." with a button labeled "Relaunch the Welcome Window".

The "Announcements" section features a post titled "Updated Datasheet, Program Guide, and FAQ Now Available" by "Anonymous User" dated "Sep 19, 2023 8:22 AM". The post mentions that "IT Academy has updated three of our main resources:" and lists a "Datasheet" which includes updates to course pathways.

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 51
Microservices Course

KubeAcademy
by VMware

Courses

Learning Paths

Instructors

Please check your email for a code to activate your account.

Activate Account →

YASH LAKHTARIYA

Home

My Account

My Favorites

Give Feedback

Log Out

Interactive Introduction to Containers and Kubernetes

Beginner

4 Lessons

1h 36m

Ready to dive a little deeper into the world of Kubernetes? Get up to speed on the first principles of a cloud native infrastructure, then learn how to containerize and deploy a Kubernetes application in our lab environment.

Restart Course →

Add to Favorites

Share:

Lesson 01: Introduction to Containers

Covers concepts of containers that are necessary to use Kubernetes. Why containers exist, comparisons to virtual machines, and the necessary commands to build, run, and debug containers are all covered.

Video

11m

Lab 01: Containerize Applications

This lab puts into practice the concepts introduced by using a two tier web application. You will containerize the applications, run and verify the stack, and then tag and push the built images to a registry so they can be used in the next lab with Kubernetes.

Lab

30m

Lesson 02: Kubernetes Fundamentals

Kubernetes is a large and complex platform. Why Kubernetes exists and principles of its design are carefully introduced. You will learn the fundamental concepts needed to deploy and run an application in Kubernetes.

Video

25m

Lab 02: Deploy Applications Using Kubernetes

This lab builds on the previous and puts into practice the Kubernetes concepts you've learned. The two tier application that was containerized in the first lab will be deployed and ran in Kubernetes.

Lab

30m

INSTRUCTOR

Ryan Schneider

Lead Technical Trainer at VMware

Restart Course →

KubeAcademy

Unlock your full potential with Kubernetes courses designed by experts.

© 2023 VMware, Inc.

LEARN

Courses

Learning Paths

ENGAGE

Instructors

Give Feedback

MORE

Contact

Terms

Privacy

Your California Privacy Rights

Cookie Settings