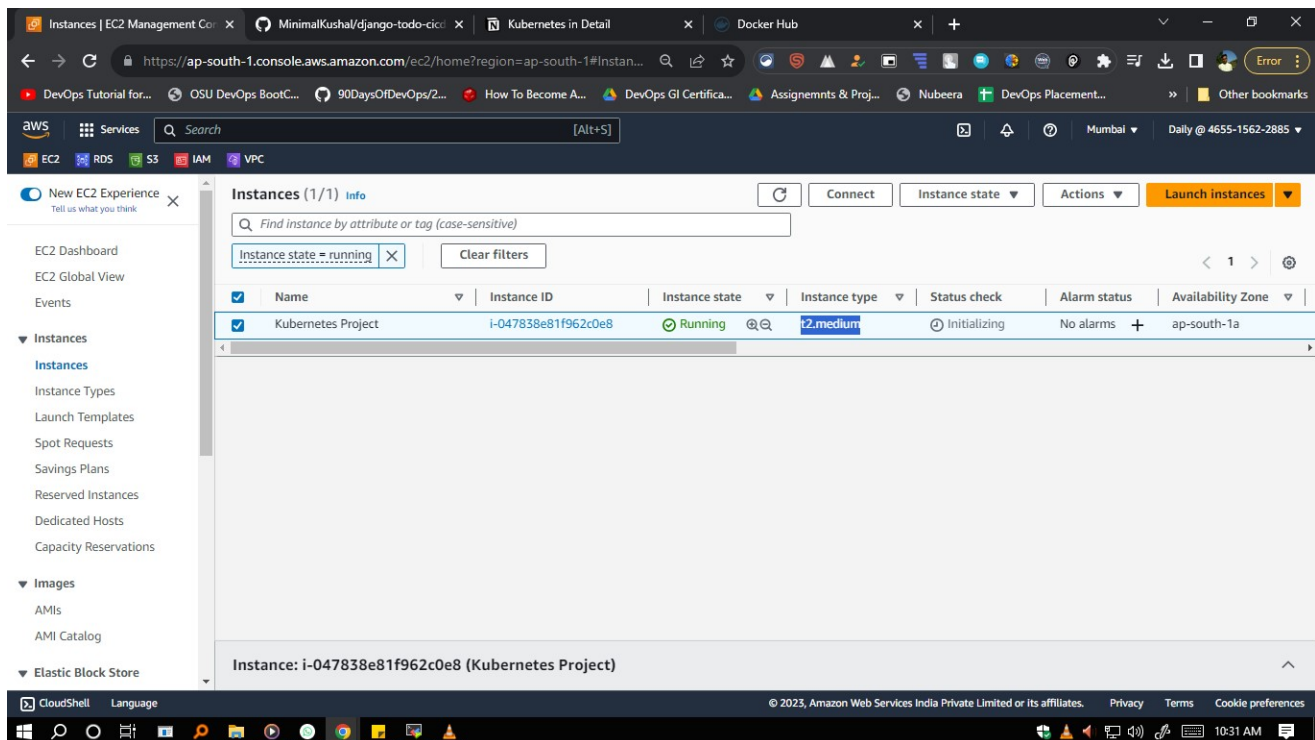


# Kubernetes Minikube Assignment

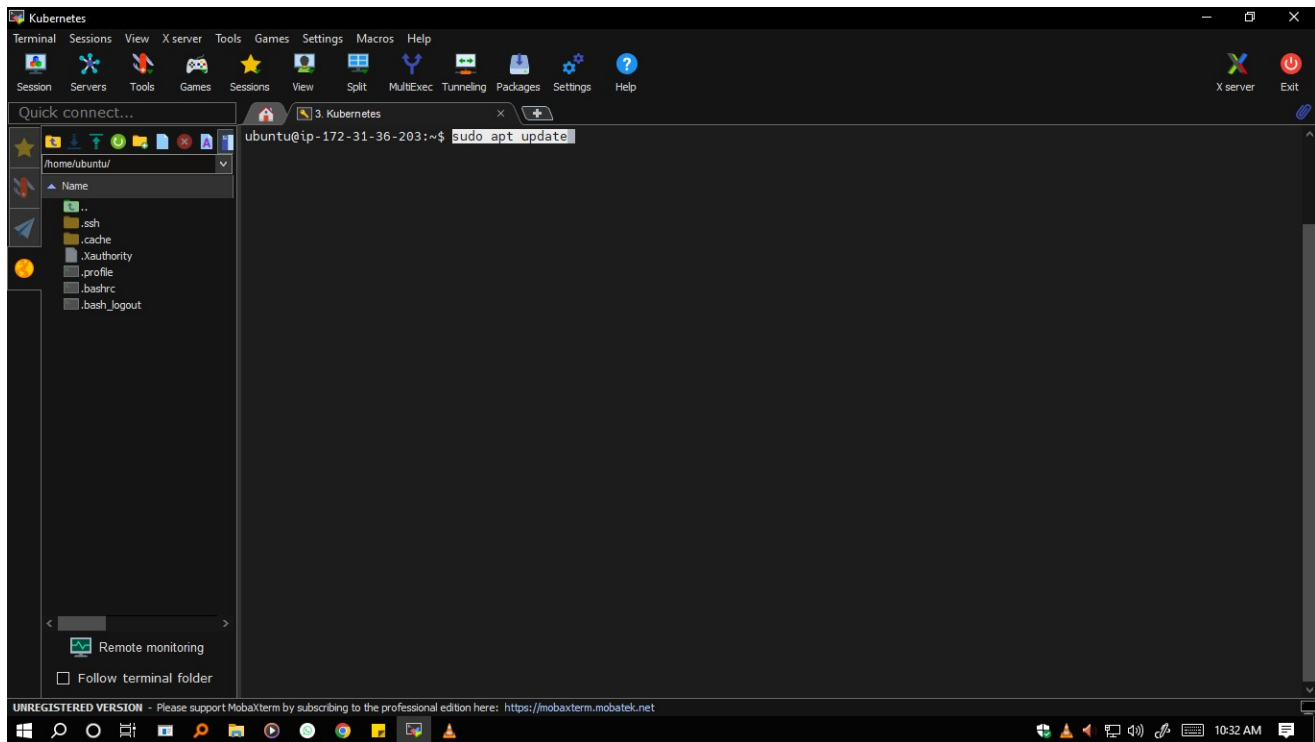
By Kushal Chauhan

In this assignment I will be using Minikube Cluster to show Pod, Deployment and Service Creation, Replication and Auto healing for Kubernetes Cluster.

1. First we need to create a T2 Medium EC2 instance (I am using Ubuntu AMI) as this is the minimum requirement for Minikube to run.



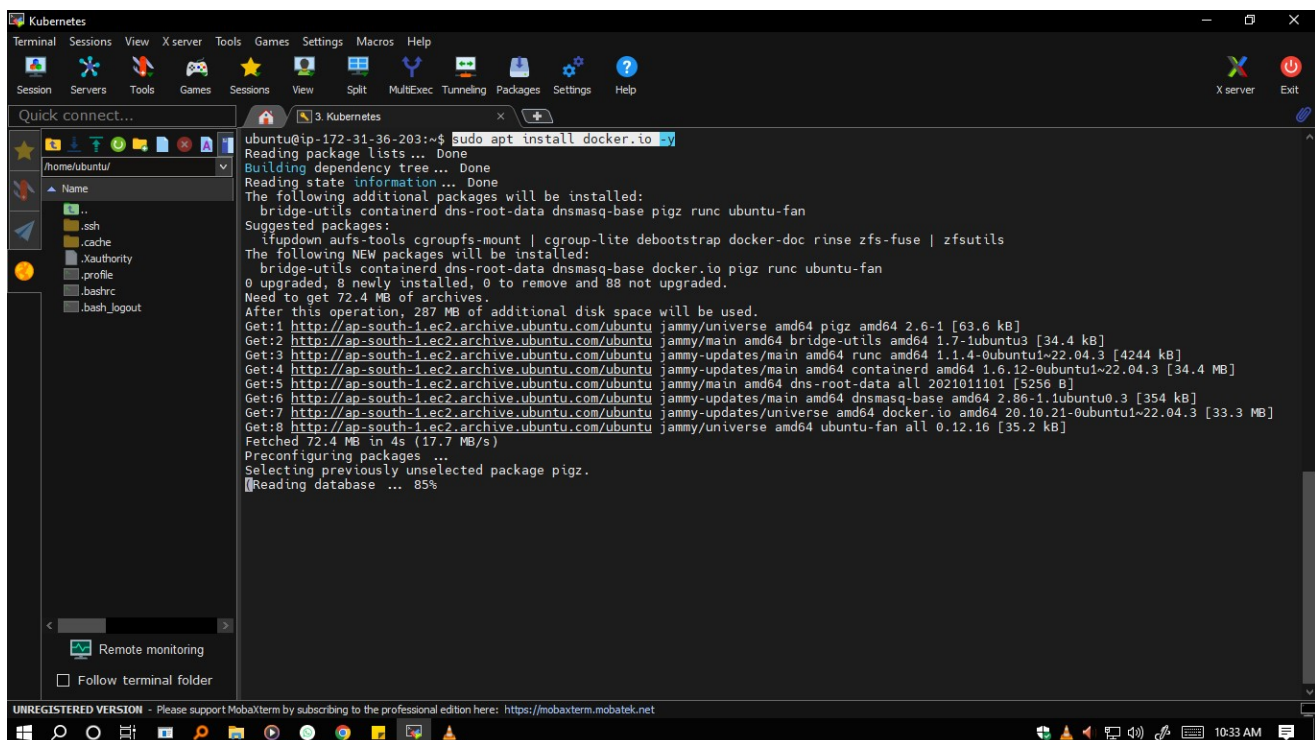
2. Then we need to update the package list using “sudo apt update” command.



The screenshot shows the MobaXterm application window. The terminal pane displays the command `sudo apt update` being executed. The left sidebar shows the file explorer with the `/home/ubuntu/` directory selected. The top menu bar includes options like Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, and Help. The bottom status bar indicates the application is an "UNREGISTERED VERSION" and shows the system time as 10:32 AM.

```
ubuntu@ip-172-31-36-203:~$ sudo apt update
```

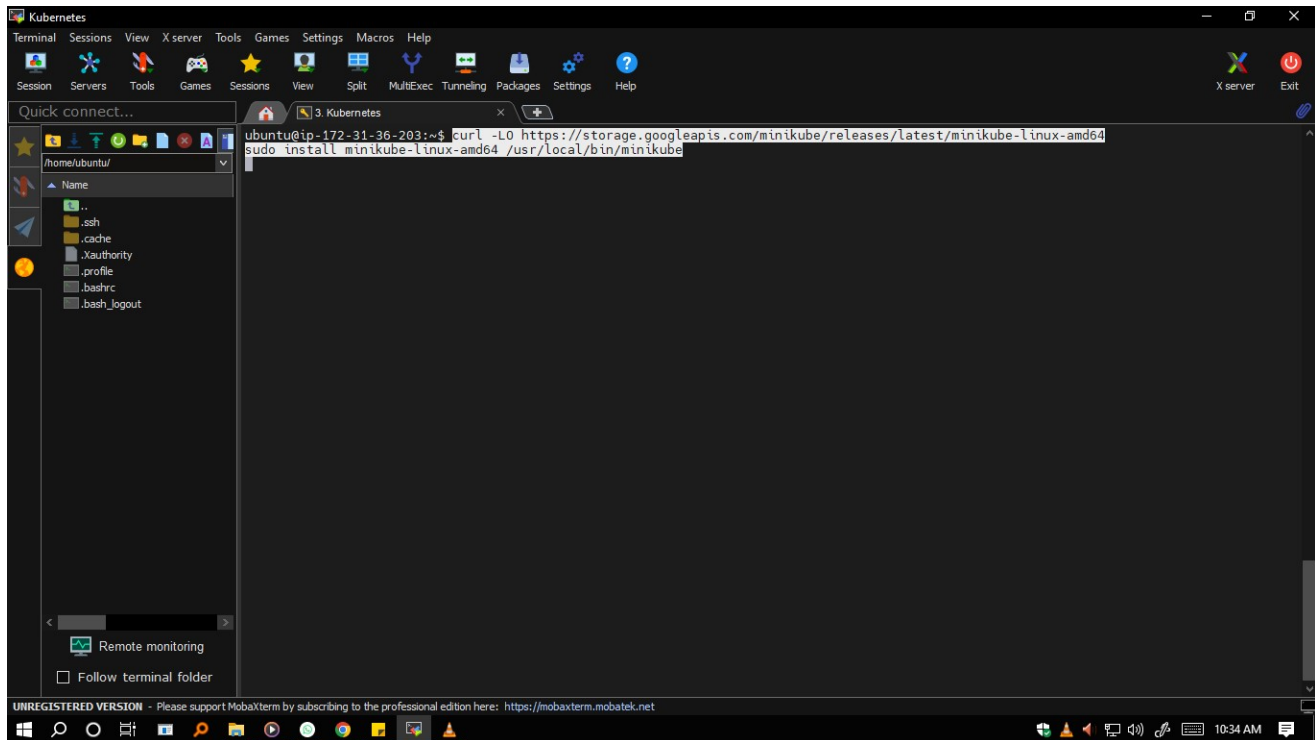
3. Then we will install Docker in our Ubuntu EC2 instance.



The screenshot shows the MobaXterm application window with the terminal pane displaying the command `sudo apt install docker.io`. The output shows the package list being updated and the installation of Docker. The left sidebar shows the file explorer with the `/home/ubuntu/` directory selected. The top menu bar includes options like Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, and Help. The bottom status bar indicates the application is an "UNREGISTERED VERSION" and shows the system time as 10:33 AM.

```
ubuntu@ip-172-31-36-203:~$ sudo apt install docker.io
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  lftpdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 88 not upgraded.
Need to get 72.4 MB of archives.
After this operation, 287 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.4-0ubuntu1~22.04.3 [4244 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.6.12-0ubuntu1~22.04.3 [34.4 MB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 dns-root-data all 2021011101 [5256 B]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.86-1.1ubuntu0.3 [354 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 20.10.21-0ubuntu1~22.04.3 [33.3 MB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 72.4 MB in 4s (17.7 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
Reading database ... 85%
```

4. After that we will install Minikube.

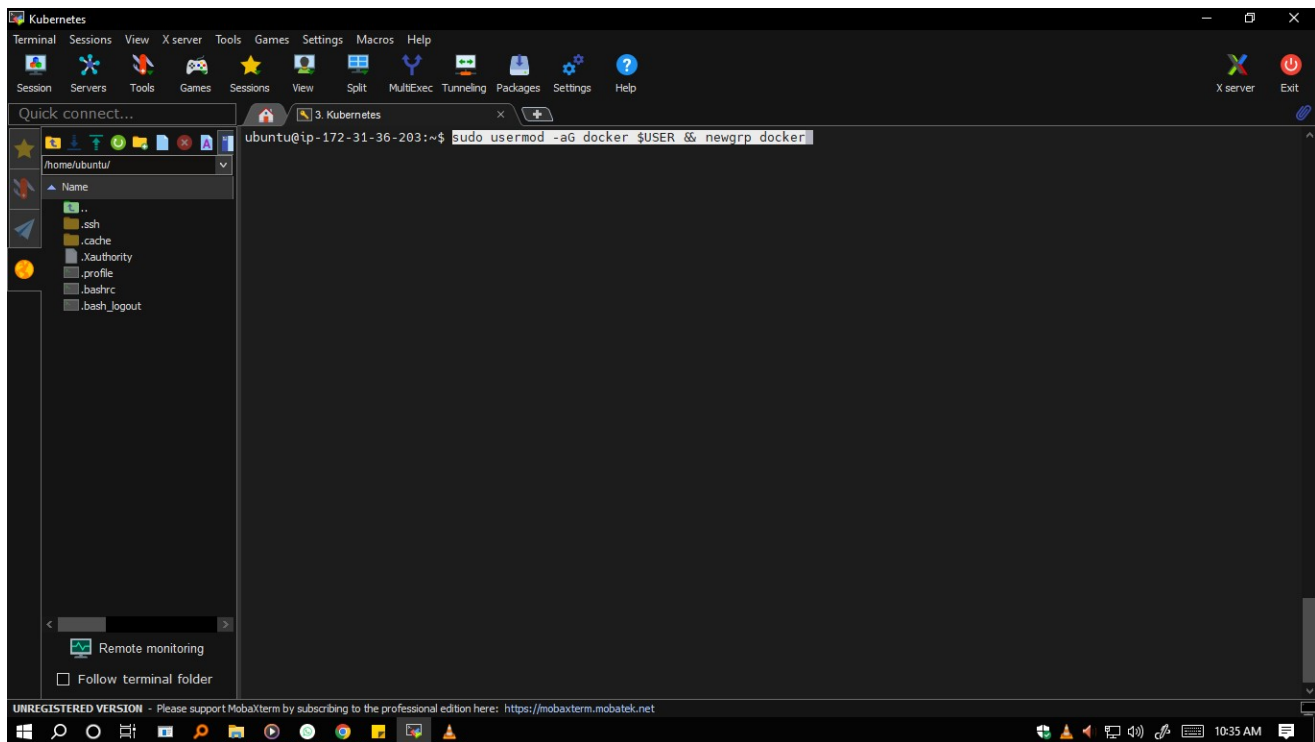


The screenshot shows a MobaXterm window titled 'Kubernetes'. The terminal is connected to an Ubuntu machine with IP 172-31-36-203. The user is at the root prompt. The commands entered are:

```
ubuntu@ip-172-31-36-203:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
ubuntu@ip-172-31-36-203:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

The left sidebar shows the file explorer for the home directory of the user 'ubuntu', listing files like .ssh, .cache, .Xauthority, .profile, .bashrc, and .bash\_logout. The bottom status bar indicates 'UNREGISTERED VERSION' and provides a link to the professional edition.

5. Add current user to Docker Group.

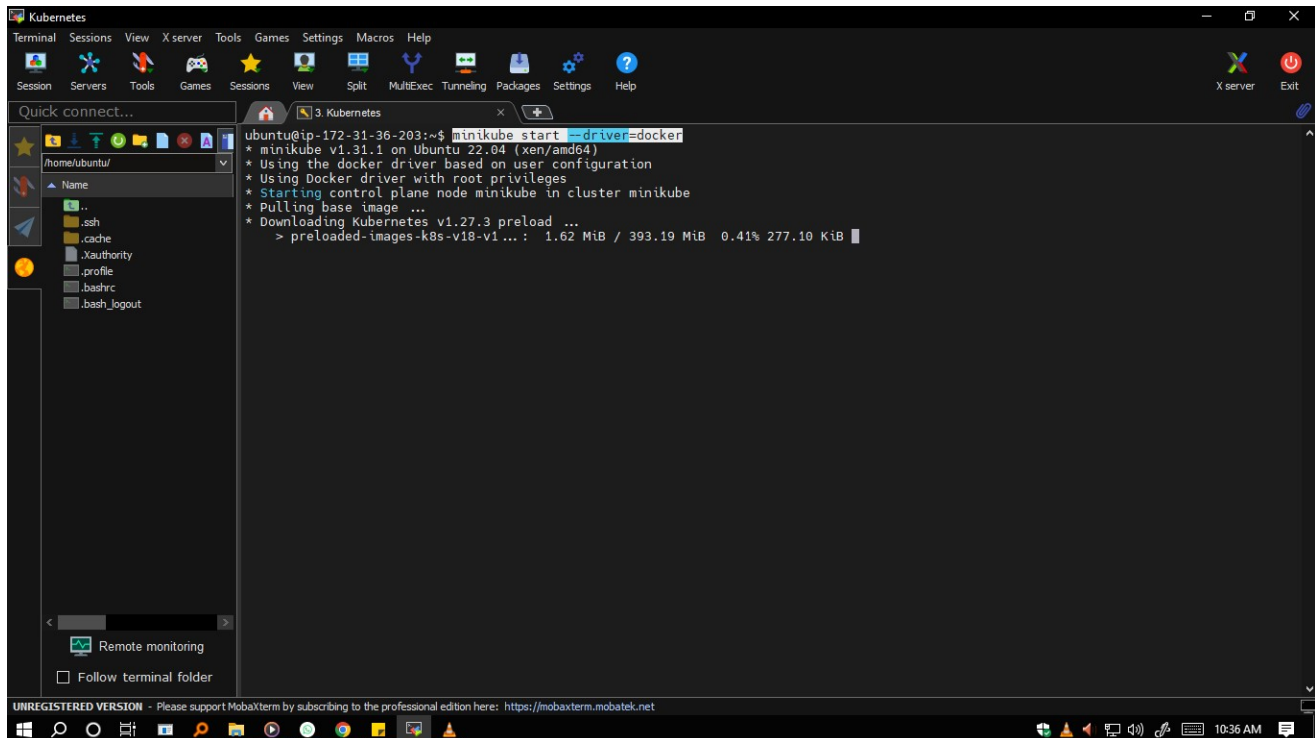


The screenshot shows the same MobaXterm window. The terminal now shows the command to add the user to the docker group:

```
ubuntu@ip-172-31-36-203:~$ sudo usermod -ag docker $USER && newgrp docker
```

The interface remains the same as in the previous screenshot, with the file explorer on the left and the status bar at the bottom.

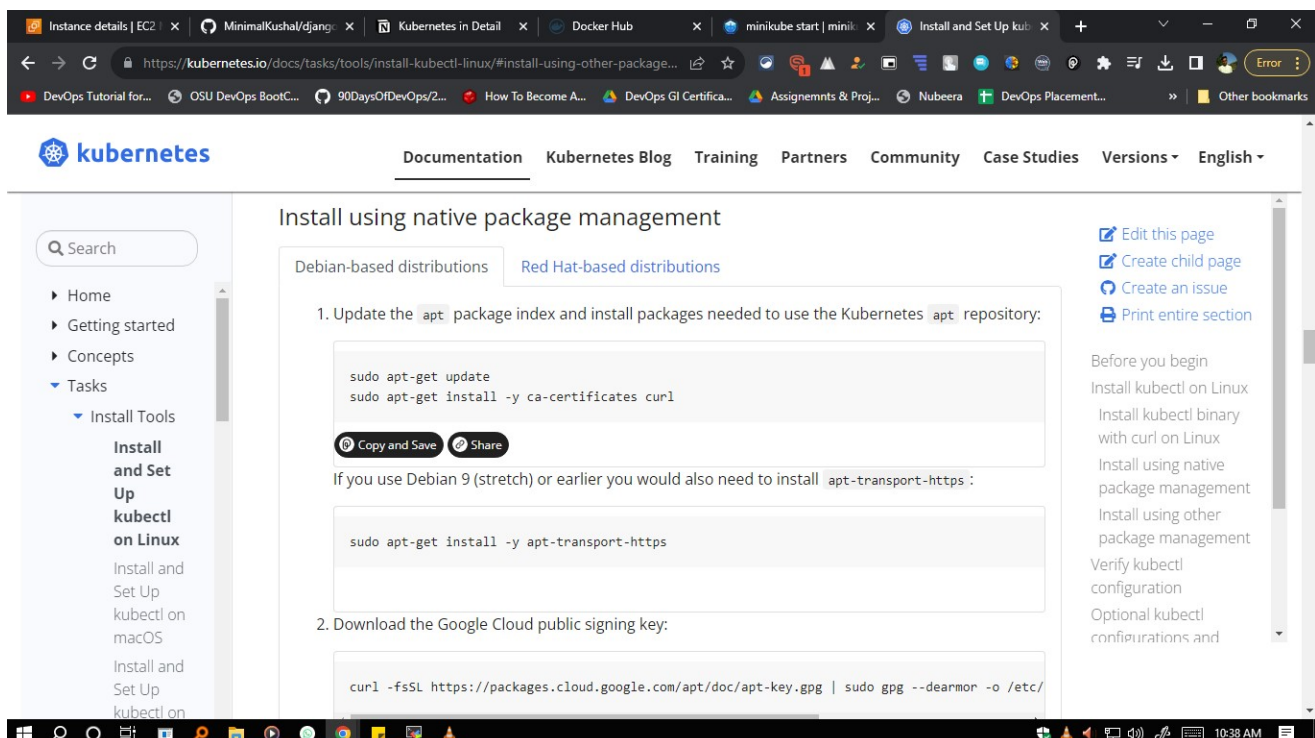
6. Start the Minikube Kubernetes Cluster using “minikube start –driver=docker” command.



The screenshot shows a MobaXterm terminal window with the title 'Kubernetes'. The terminal displays the command `minikube start --driver=docker` and its output. The output indicates that minikube v1.31.1 is running on Ubuntu 22.04 (xen/amd64), using the docker driver based on user configuration. It shows the process of starting the control plane node, pulling the base image, and downloading the Kubernetes v1.27.3 preload. The progress bar for downloading the preload images is visible, showing 1.62 MiB / 393.19 MiB at 0.41%.

```
ubuntu@ip-172-31-36-203:~$ minikube start --driver=docker
* minikube v1.31.1 on Ubuntu 22.04 (xen/amd64)
* Using the docker driver based on user configuration
* Using Docker driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.27.3 preload ...
  > preloaded-images-k8s-v18-v1 ... : 1.62 MiB / 393.19 MiB  0.41% 277.10 KiB
```

7. Then head over to Kubectl installation page to get all commands regarding the installation of Kubectl.



The screenshot shows the Kubernetes documentation page for installing kubectl on Linux. The page is titled 'Install using native package management' and is divided into two sections: 'Debian-based distributions' and 'Red Hat-based distributions'. The 'Debian-based distributions' section contains the following steps:

- Update the `apt` package index and install packages needed to use the Kubernetes `apt` repository:

```
sudo apt-get update
sudo apt-get install -y ca-certificates curl
```

Buttons: Copy and Save, Share

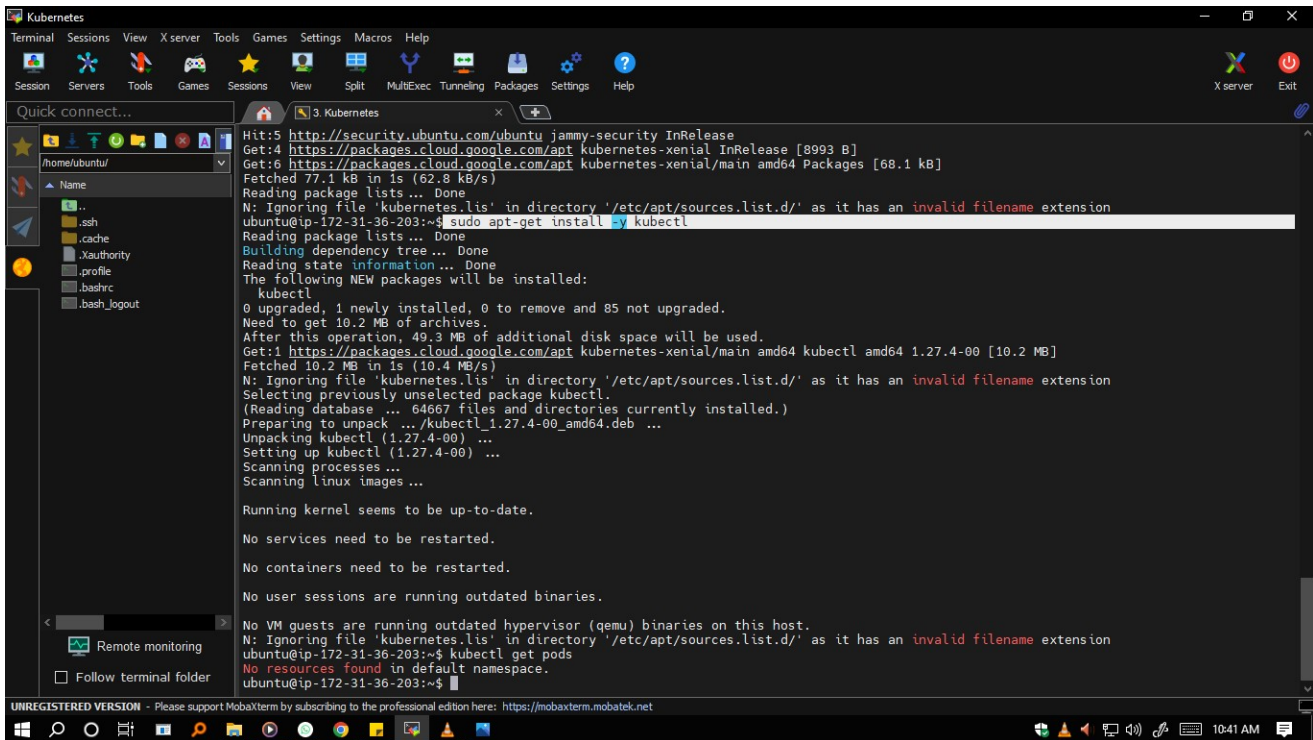
If you use Debian 9 (stretch) or earlier you would also need to install `apt-transport-https`:

```
sudo apt-get install -y apt-transport-https
```
- Download the Google Cloud public signing key:

```
curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /etc/
```

The right sidebar contains links for 'Edit this page', 'Create child page', 'Create an issue', and 'Print entire section'. Below these links is a 'Before you begin' section with links to 'Install kubectl on Linux', 'Install kubectl binary with curl on Linux', 'Install using native package management', 'Install using other package management', 'Verify kubectl configuration', 'Optional kubectl configurations and', and 'Other bookmarks'.

8. After following all the commands on the page we have successfully installed Kubectl. We have confirmed it by using “kubectl get pods” command which is used to display the created pod.



```
Kubernetes
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help

Quick connect...
/home/ubuntu/
Name
ssh
.ssh
.cache
.xauthority
.profile
.bashrc
.bash_logout

Remote monitoring
Follow terminal folder

Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:4 https://packages.cloud.google.com/apt/kubernetes-xenial InRelease [8993 B]
Get:6 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 Packages [68.1 kB]
Fetched 77.1 kB in 1s (62.8 kB/s)
Reading package lists... Done
N: Ignoring file 'kubernetes.lis' in directory '/etc/apt/sources.list.d/' as it has an invalid filename extension
ubuntu@ip-172-31-36-203:~$ sudo apt-get install kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  kubectl
0 upgraded, 1 newly installed, 0 to remove and 85 not upgraded.
Need to get 10.2 MB of archives.
After this operation, 49.3 MB of additional disk space will be used.
Get:1 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 kubectl amd64 1.27.4-00 [10.2 MB]
Fetched 10.2 MB in 1s (10.4 MB/s)
N: Ignoring file 'kubernetes.lis' in directory '/etc/apt/sources.list.d/' as it has an invalid filename extension
Selecting previously unselected package kubectl.
(Reading database ... 64667 files and directories currently installed.)
Preparing to unpack .../kubectl_1.27.4-00_amd64.deb ...
Unpacking kubectl (1.27.4-00) ...
Setting up kubectl (1.27.4-00) ...
Scanning processes ...
Scanning linux images ...

Running kernel seems to be up-to-date.

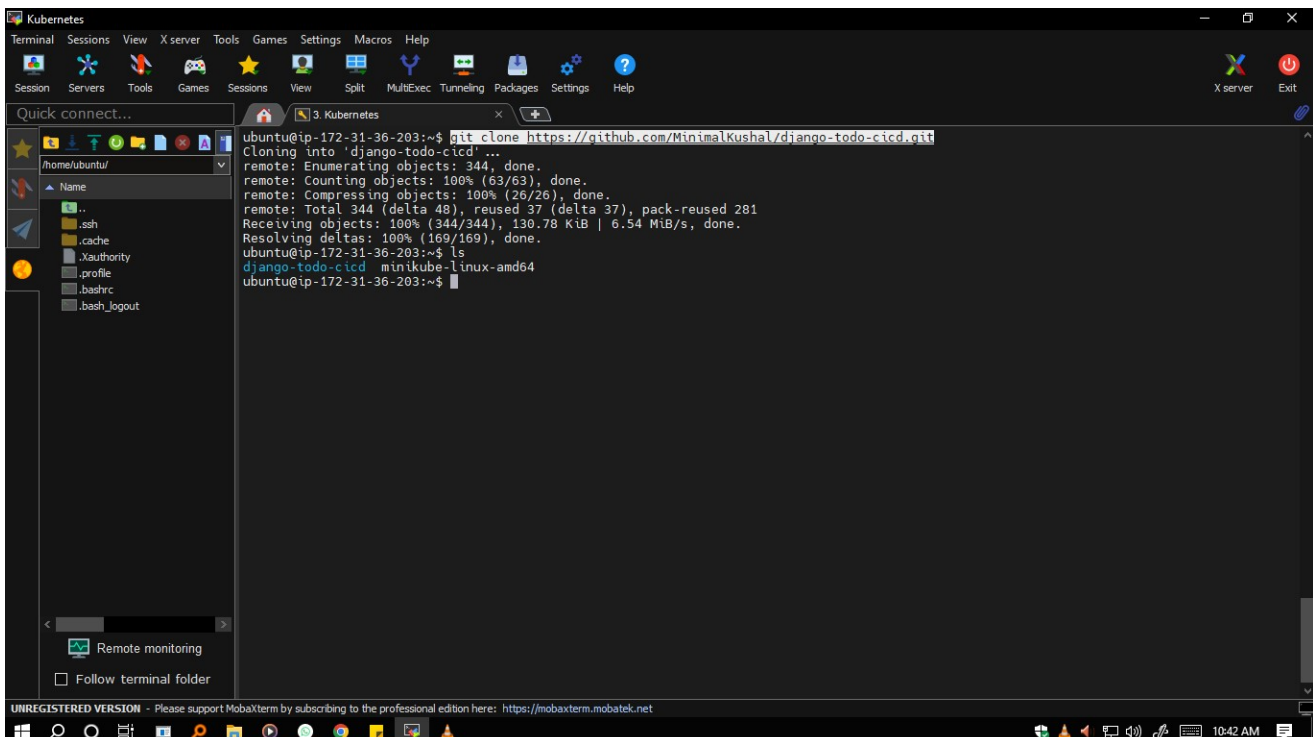
No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
N: Ignoring file 'kubernetes.lis' in directory '/etc/apt/sources.list.d/' as it has an invalid filename extension
ubuntu@ip-172-31-36-203:~$ kubectl get pods
No resources found in default namespace.
ubuntu@ip-172-31-36-203:~$
```

9. After successfully installing all the required packages, we clone the GitHub repo to our local machine.



```
Kubernetes
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help

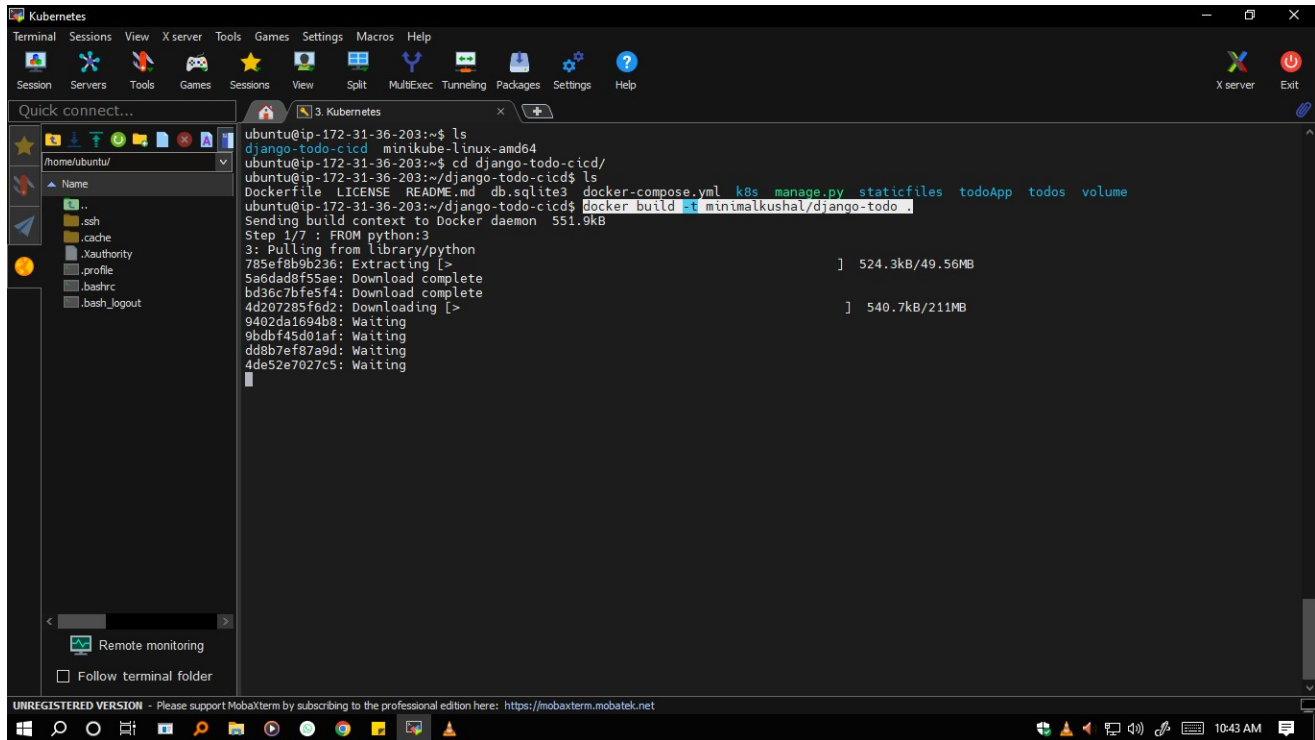
Quick connect...
/home/ubuntu/
Name
ssh
.ssh
.cache
.xauthority
.profile
.bashrc
.bash_logout

Remote monitoring
Follow terminal folder

ubuntu@ip-172-31-36-203:~$ git clone https://github.com/MinimaKushal/django-todo-cicd.git
Cloning into 'django-todo-cicd' ...
remote: Enumerating objects: 344, done.
remote: Counting objects: 100% (63/63), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 344 (delta 48), reused 37 (delta 37), pack-reused 281
Receiving objects: 100% (344/344), 130.78 KiB | 6.54 MiB/s, done.
Resolving deltas: 100% (169/169), done.
ubuntu@ip-172-31-36-203:~$ ls
django-todo-cicd minikube-Linux-amd64
ubuntu@ip-172-31-36-203:~$
```



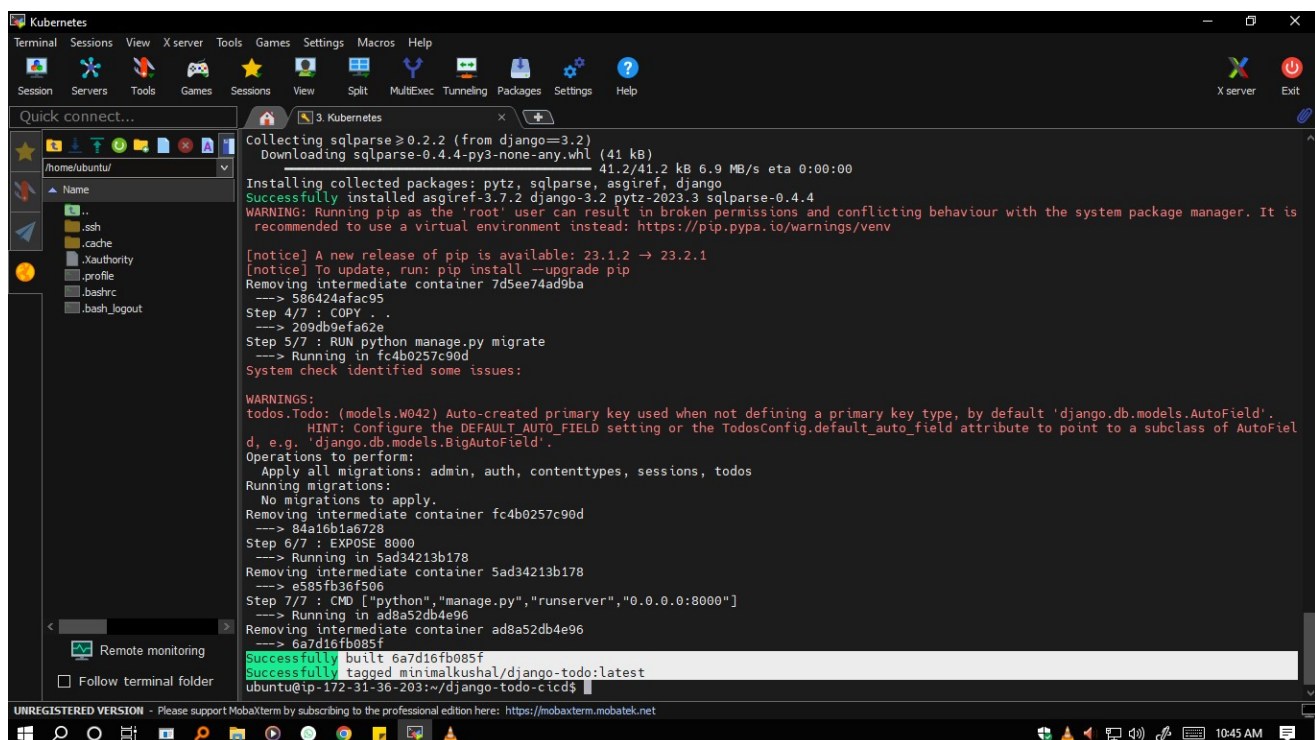
10. Then we go Inside the directory which we cloned using “cd” command. And then we build the Docker image using the Dockerfile provided.



The screenshot shows a MobaXterm terminal window with the title 'Kubernetes'. The terminal output shows the user navigating to the directory `~/django-todo-cicd` and running `docker build -t minikushal/django-todo`. The build process starts by sending the build context to the Docker daemon (551.9kB). It then pulls the Python base image `python:3` (524.3kB/49.56MB) and the Django image `django:3.2` (540.7kB/211MB). The build is currently waiting for the Django image to be fully downloaded.

```
ubuntu@ip-172-31-36-203:~$ ls
django-todo-cicd minikube-linux-amd64
ubuntu@ip-172-31-36-203:~$ cd django-todo-cicd/
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ ls
Dockerfile LICENSE README.md db.sqlite3 docker-compose.yml k8s manage.py staticfiles todoApp todos volume
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ docker build -t minikushal/django-todo
Sending build context to Docker daemon 551.9kB
Step 1/7 : FROM python:3
3: Pulling from library/python
785ef8b9b236: Extracting [>] 524.3kB/49.56MB
5a6dad8f55ae: Download complete
bd36c7bfe5f4: Download complete
4d207285f6d2: Downloading [>] 540.7kB/211MB
9402da1694b8: Waiting
9bdbf45d01af: Waiting
dd8b7ef87a9d: Waiting
4de52e7027c5: Waiting
```

11. Our Docker images has successfully been created.



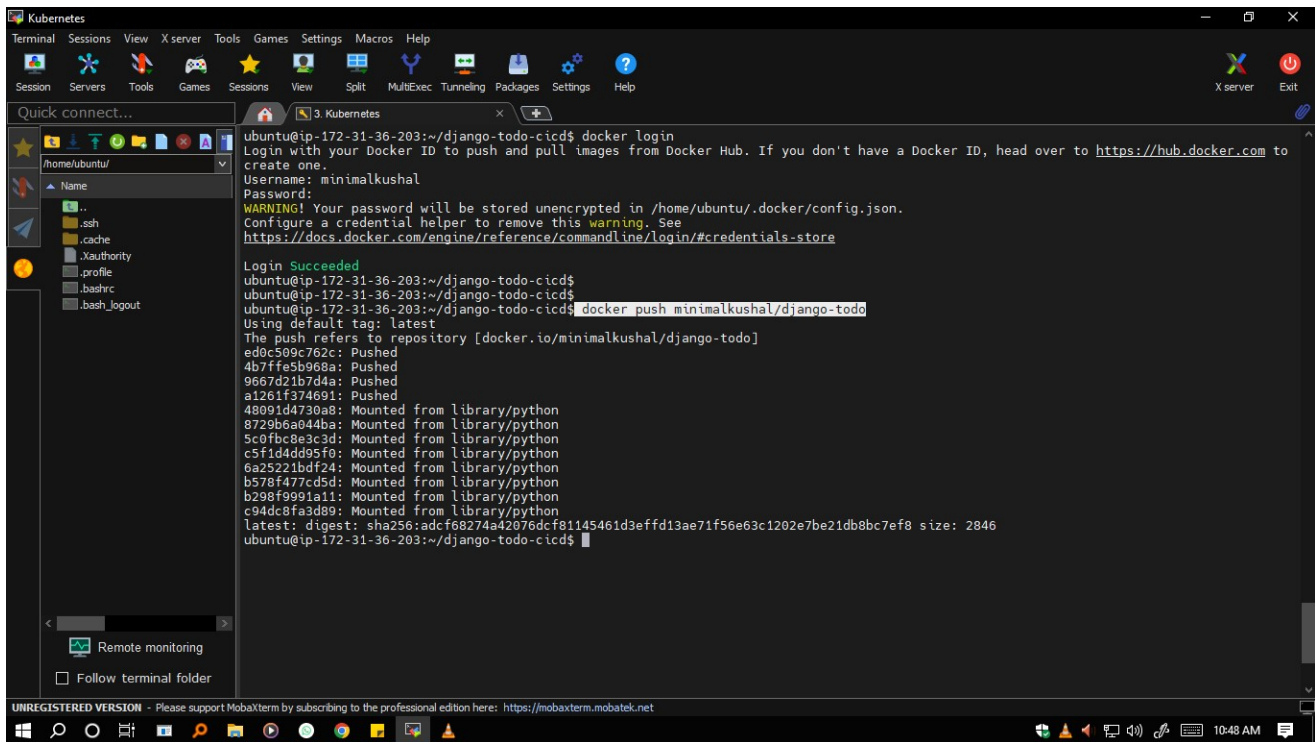
The screenshot shows the continuation of the Docker build process. The image `minikushal/django-todo:latest` is successfully built and tagged. The terminal then shows the user running `docker-compose up`, which starts the application. The output shows the installation of dependencies (41 kB), the creation of the database, and the application starting. The user then runs `docker-compose down` to stop the containers. The terminal output shows the removal of intermediate containers and the successful removal of the application container.

```
Collecting sqlparse>=0.2.2 (from django==3.2)
Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
Installing collected packages: pytz, sqlparse, asgiref, django
Successfully installed asgiref-3.7.2 django-3.2 pytz-2023.3 sqlparse-0.4.4
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is
recommended to use a virtual environment instead: https://pip.pya.io/warnings/venv

[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: pip install --upgrade pip
Removing intermediate container 7d5ee74ad9ba
--> 586424afac95
Step 4/7 : COPY . .
--> 209db9efa62e
Step 5/7 : RUN python manage.py migrate
--> Running in fc4b0257c90d
System check identified some issues:

WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.
HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.default_auto_field attribute to point to a subclass of AutoFiel
d, e.g. 'django.db.models.BigAutoField'.
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, todos
Running migrations:
  No migrations to apply.
Removing intermediate container fc4b0257c90d
--> 84a16b1a6728
Step 6/7 : EXPOSE 8000
--> Running in 5ad34213b178
Removing intermediate container 5ad34213b178
--> e585fb36f506
Step 7/7 : CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
--> Running in ad8a52db4e96
Removing intermediate container ad8a52db4e96
--> 6a7d16fb085f
Successfully built 6a7d16fb085f
Successfully tagged minikushal/django-todo:latest
ubuntu@ip-172-31-36-203:~/django-todo-cicd$
```

## 12. Then login in and push the created image to Docker Hub.

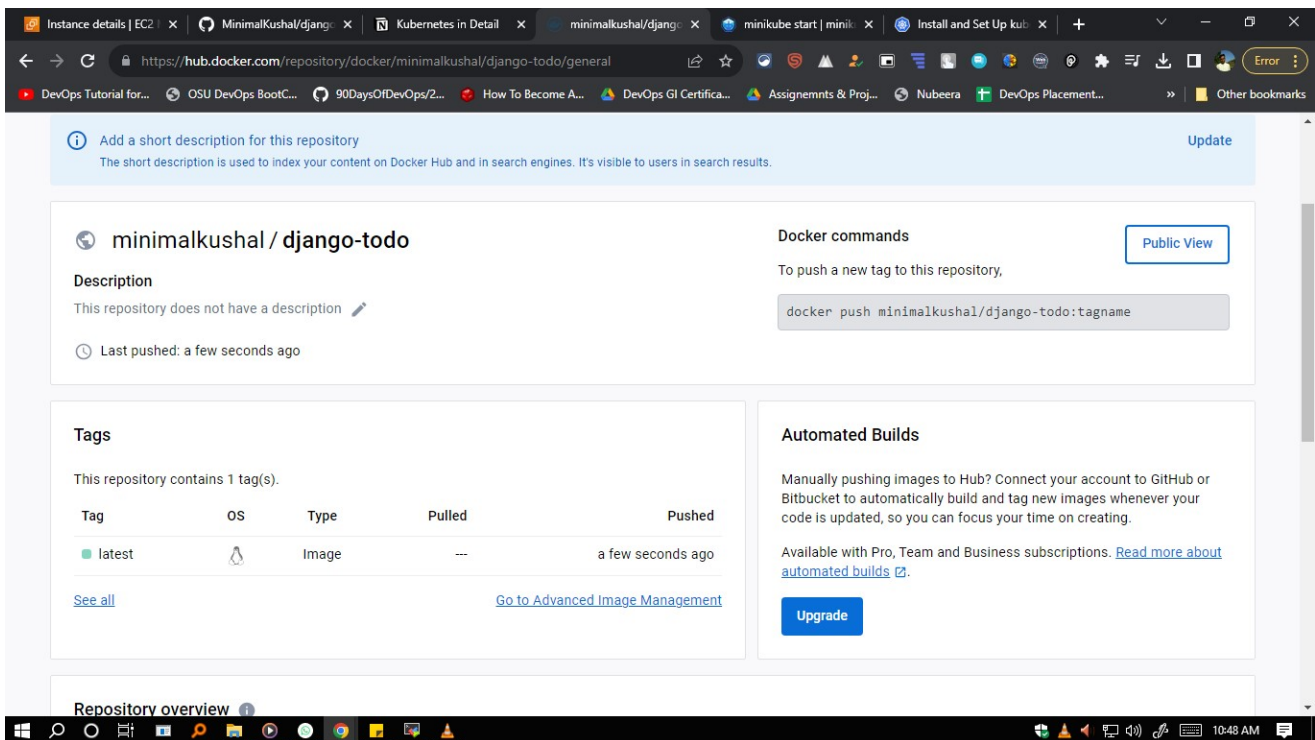


The screenshot shows a terminal window with the following commands and output:

```
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: minimalkushal
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-172-31-36-203:~/django-todo-cicd$
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ docker push minimalkushal/django-todo
Using default tag: latest
The push refers to repository [docker.io/minimalkushal/django-todo]
ed0c509c762c: Pushed
4b7ffe5b968a: Pushed
9667d21b7d4a: Pushed
a1261f374691: Pushed
48091d4730a8: Mounted from library/python
8729b6a044ba: Mounted from library/python
5c0fbc8e3c3d: Mounted from library/python
c5f1d4d95f0: Mounted from library/python
6a25221bdf24: Mounted from library/python
b578f477cd5d: Mounted from library/python
b298f9991a11: Mounted from library/python
c94dc8fa3d89: Mounted from library/python
latest: digest: sha256:adc68274a42076dcf81145461d3effd13ae71f56e63c1202e7be21db8bc7ef8 size: 2846
ubuntu@ip-172-31-36-203:~/django-todo-cicd$
```

## 13. We can see our Docker images has been pushed to Docker Hub.



The screenshot shows the Docker Hub repository page for `minimalkushal / django-todo`. The page includes a description, tags, and automated builds section.

**minimalkushal / django-todo**

**Description**  
This repository does not have a description  
Last pushed: a few seconds ago

**Tags**  
This repository contains 1 tag(s).

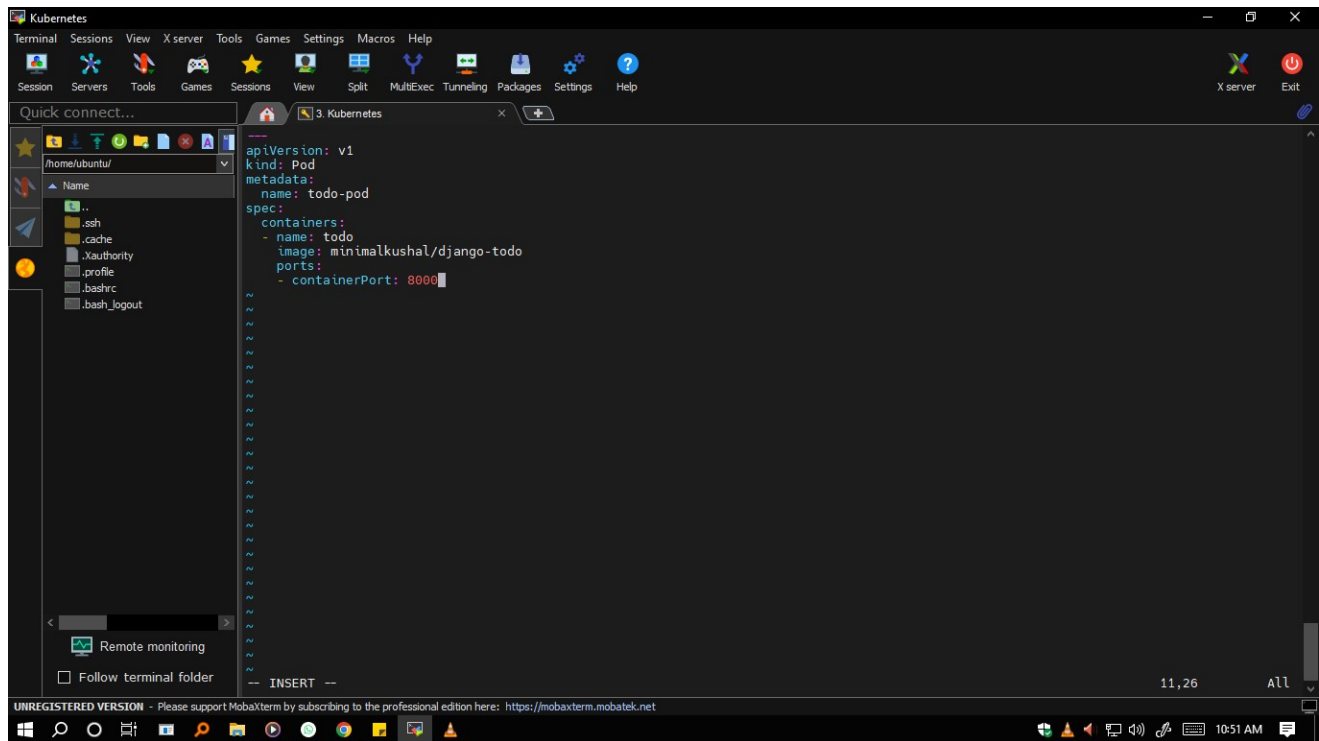
Tag	OS	Type	Pulled	Pushed
latest		Image	---	a few seconds ago

[See all](#) [Go to Advanced Image Management](#)

**Docker commands**  
To push a new tag to this repository,  
`docker push minimalkushal/django-todo:tagname`

**Automated Builds**  
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.  
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)  
[Upgrade](#)

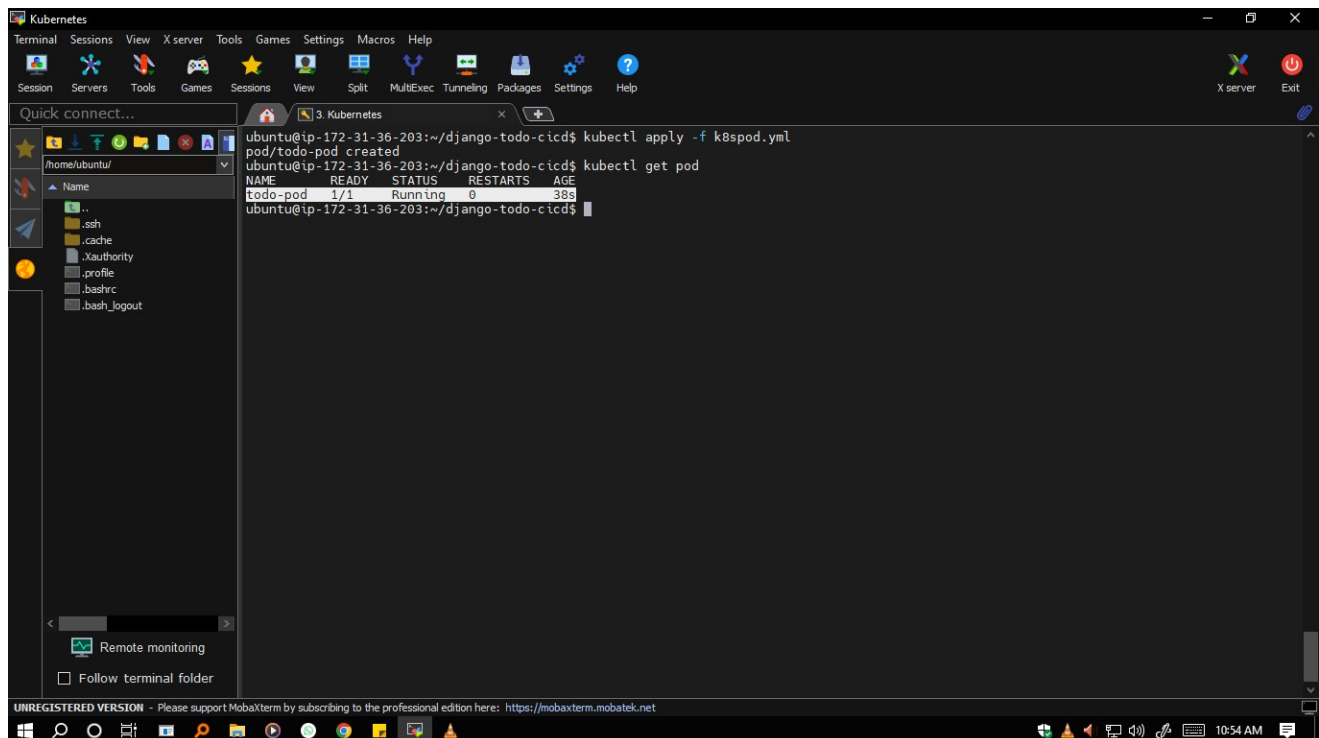
14. Then we create a pod.yml file and write a Kubernetes Manifest file to create a pod.



The screenshot shows the MobaXterm application window with a terminal session. The terminal displays the contents of a Kubernetes manifest file named `k8spod.yml`. The manifest defines a pod named `todo-pod` using the `minimalkushal/django-todo` image, with a container port of `8000`. The left sidebar shows the file explorer with the `todo-pod` directory selected. The bottom status bar indicates the application is an "UNREGISTERED VERSION" and the time is 10:51 AM.

```
apiVersion: v1
kind: Pod
metadata:
  name: todo-pod
spec:
  containers:
    - name: todo
      image: minimalkushal/django-todo
      ports:
        - containerPort: 8000
```

15. Then we apply the manifest file and we get a pod.

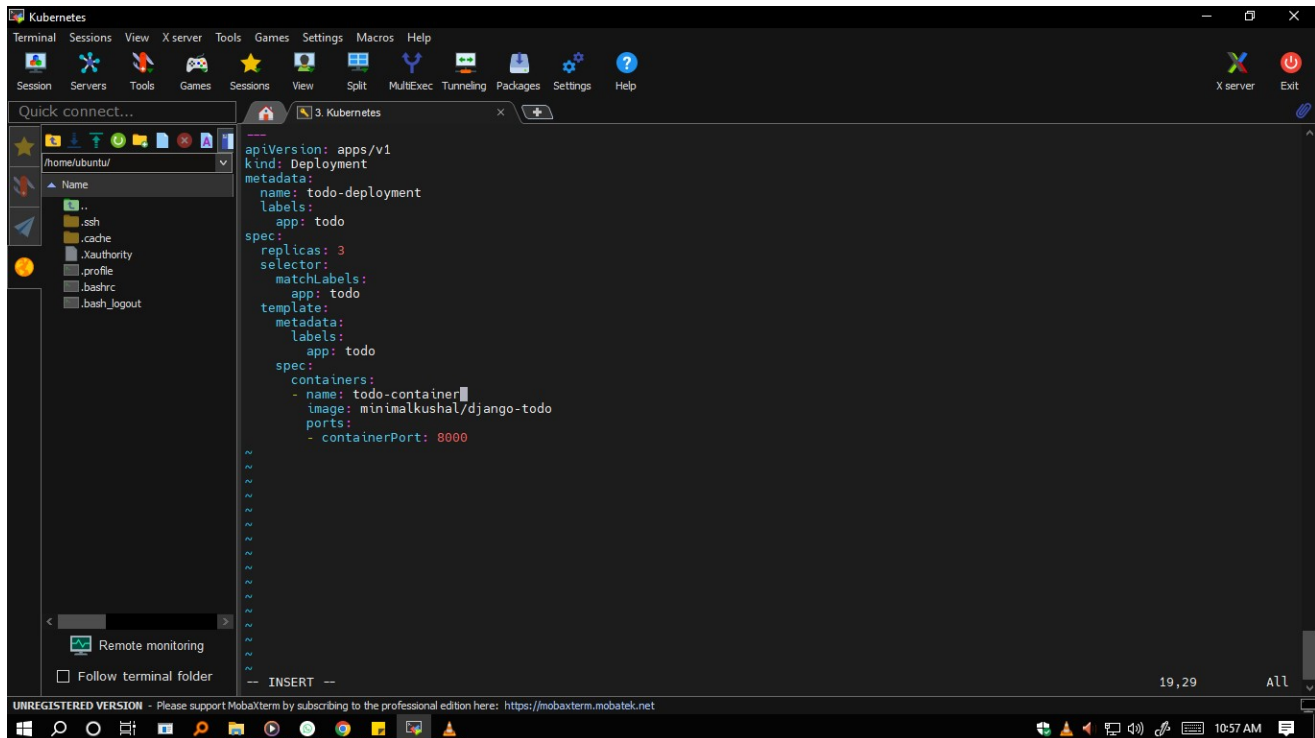


The screenshot shows the MobaXterm application window with a terminal session. The terminal displays the command `kubectl apply -f k8spod.yml` being executed, followed by the output `pod/todo-pod created`. The user then runs `kubectl get pod`, which displays a table of pods. The table shows one pod named `todo-pod` in a `Running` state. The left sidebar shows the file explorer with the `todo-pod` directory selected. The bottom status bar indicates the application is an "UNREGISTERED VERSION" and the time is 10:54 AM.

```
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl apply -f k8spod.yml
pod/todo-pod created
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
todo-pod      1/1     Running   0           38s
ubuntu@ip-172-31-36-203:~/django-todo-cicd$
```



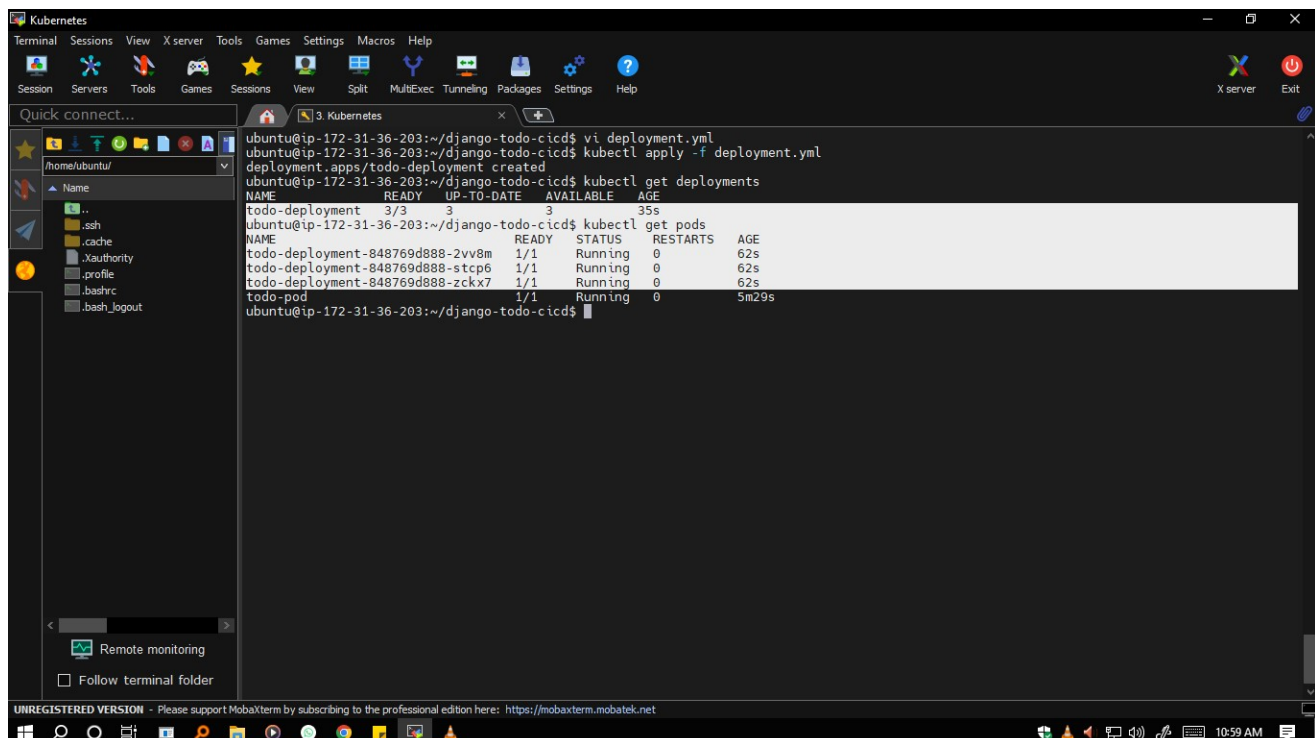
16. For managing Deployment, creating replicas and Auto-healing we will create a Deployment manifest file.



The screenshot shows the MobaXterm interface with a terminal window displaying a Kubernetes Deployment manifest file. The manifest is for a deployment named 'todo-deployment' using the 'apps/v1' API version. It specifies 3 replicas and a container named 'todo-container' using the 'minimalkushal/django-todo' image, with port 8000 exposed. The interface includes a sidebar with file explorer, a top menu bar, and a status bar at the bottom.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: todo-deployment
  labels:
    app: todo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: todo
  template:
    metadata:
      labels:
        app: todo
    spec:
      containers:
        - name: todo-container
          image: minimalkushal/django-todo
          ports:
            - containerPort: 8000
```

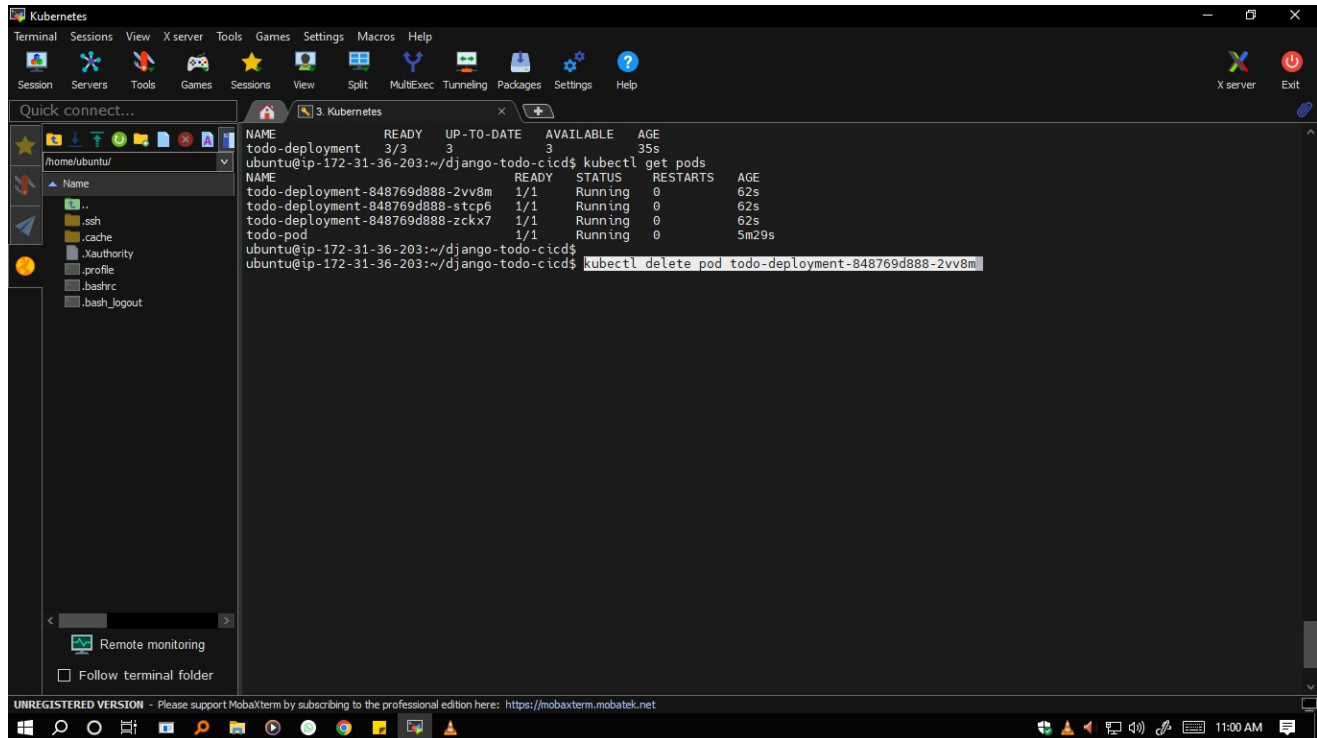
17. After applying the Deployment manifest file we can see the Deployment Object has been created.



The screenshot shows the MobaXterm interface with a terminal window displaying the commands used to apply the deployment manifest and the resulting deployment status. The commands executed are 'vi deployment.yml', 'kubectl apply -f deployment.yml', and 'kubectl get deployments'. The output shows the deployment 'todo-deployment' created successfully with 3 replicas. The interface includes a sidebar with file explorer, a top menu bar, and a status bar at the bottom.

```
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ vi deployment.yml
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl apply -f deployment.yml
deployment.apps/todo-deployment created
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl get deployments
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
todo-deployment      3/3     3             3           35s
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl get pods
NAME                 STATUS    RESTARTS   AGE
todo-deployment-848769d888-2vv8m  1/1      Running    0           62s
todo-deployment-848769d888-stcp6  1/1      Running    0           62s
todo-deployment-848769d888-zckx7  1/1      Running    0           62s
todo-pod              1/1      Running    0           5m29s
ubuntu@ip-172-31-36-203:~/django-todo-cicd$
```

18. If we try to delete a pod which is created using the Deployment we will see it heals itself which is called Auto Healing in Kubernetes.



The screenshot shows a MobaXterm terminal window titled 'Kubernetes'. The terminal displays the output of 'kubectl get pods', showing a table of pods. The 'todo-pod' is highlighted. Below the table, the command 'kubectl delete pod todo-deployment-848769d888-2vv8m' is entered and highlighted.

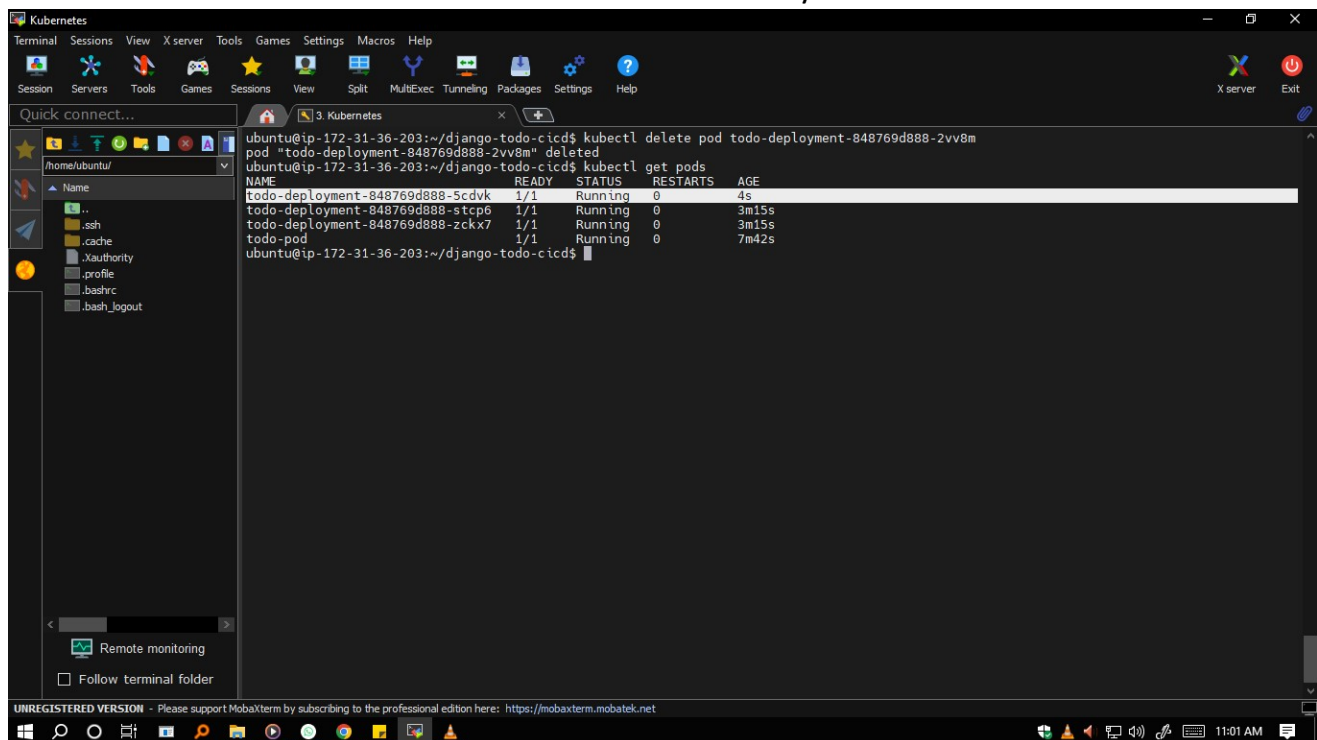
NAME	READY	UP-T0-DATE	AVAILABLE	AGE
todo-deployment	3/3	3	3	35s

NAME	READY	STATUS	RESTARTS	AGE
todo-deployment-848769d888-2vv8m	1/1	Running	0	62s
todo-deployment-848769d888-stcp6	1/1	Running	0	62s
todo-deployment-848769d888-zckx7	1/1	Running	0	62s
todo-pod	1/1	Running	0	5m29s

```
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl delete pod todo-deployment-848769d888-2vv8m
```

19. As we deleted a pod from the Deployment, now we see another new pod has been created automatically.

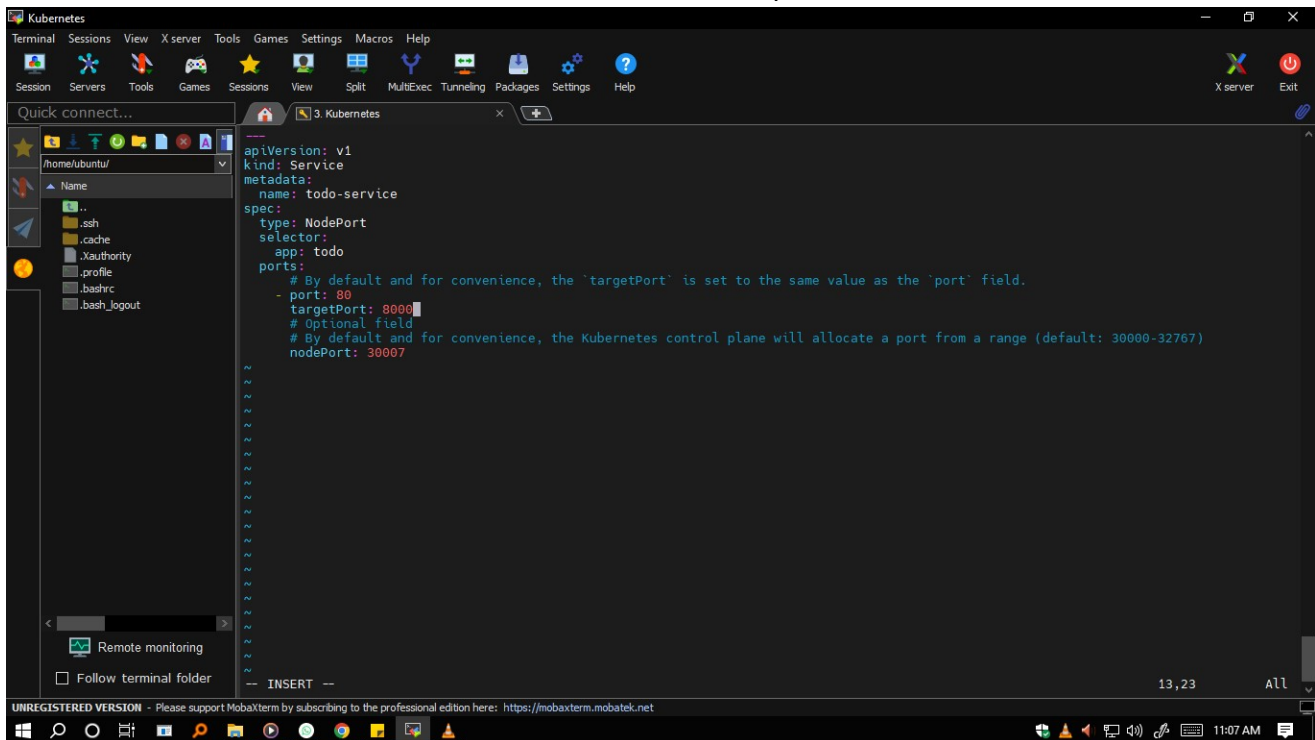


The screenshot shows the same MobaXterm terminal window. The command 'kubectl delete pod todo-deployment-848769d888-2vv8m' has been executed, and the output shows 'pod "todo-deployment-848769d888-2vv8m" deleted'. Below this, the command 'kubectl get pods' is entered, and the output shows a table of pods. The 'todo-pod' is highlighted, showing it has been replaced by a new pod with the same name.

```
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl delete pod todo-deployment-848769d888-2vv8m
pod "todo-deployment-848769d888-2vv8m" deleted
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
todo-deployment-848769d888-5cdvk	1/1	Running	0	4s
todo-deployment-848769d888-stcp6	1/1	Running	0	3m15s
todo-deployment-848769d888-zckx7	1/1	Running	0	3m15s
todo-pod	1/1	Running	0	7m42s

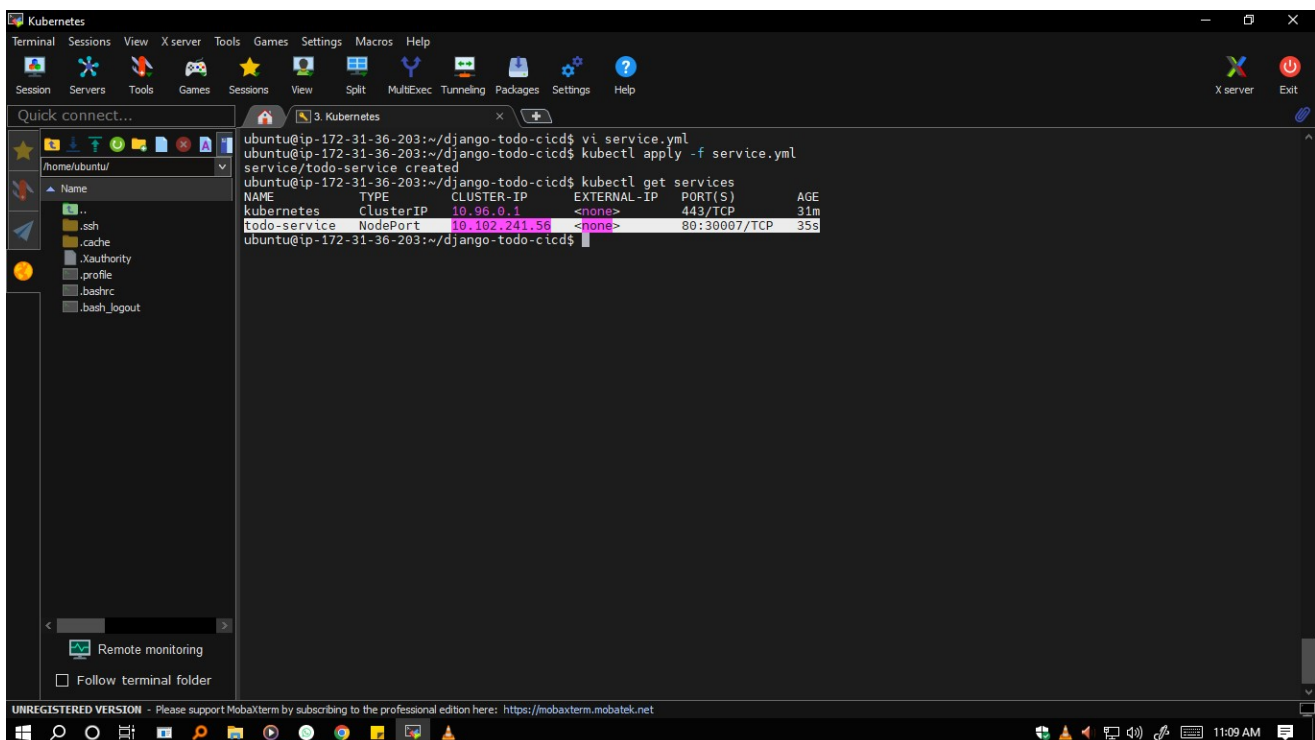
20. To expose the ToDo app to public we need to create a Service Manifest and apply it.  
(But as we are using Minikube we cannot access it from the browser so we will be using the “curl” command)



The screenshot shows a MobaXterm terminal window with a file explorer on the left. The terminal is running a command to create a Service Manifest for the 'todo' app. The output shows the manifest details, including the port configuration.

```
apiVersion: v1
kind: Service
metadata:
  name: todo-service
spec:
  type: NodePort
  selector:
    app: todo
  ports:
    # By default and for convenience, the 'targetPort' is set to the same value as the 'port' field.
    - port: 80
      targetPort: 8000
      # Optional field
      # By default and for convenience, the Kubernetes control plane will allocate a port from a range (default: 30000-32767)
      nodePort: 30007
```

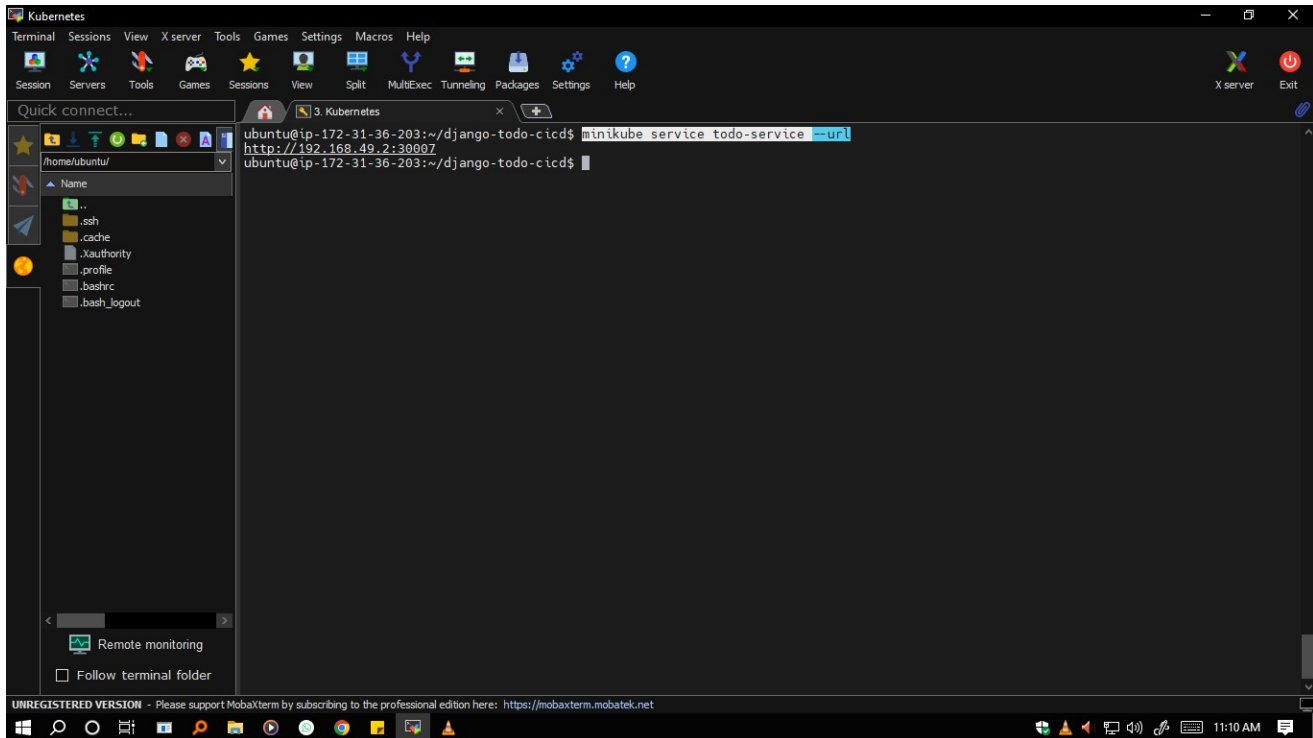
21. After applying the Service manifest file we get a Cluster IP but we will not be able to access it from outside the cluster.



The screenshot shows a MobaXterm terminal window with a file explorer on the left. The terminal is running a command to apply the Service Manifest. The output shows the service status, including the Cluster IP and the NodePort.

```
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ vi service.yml
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl apply -f service.yml
service/todo-service created
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes    ClusterIP     10.96.0.1     <none>          443/TCP          31m
todo-service   NodePort      10.102.241.56 <none>          80:30007/TCP     35s
```

22. We use the command “minikube service <Service-Name> --url” to get the URL using which we can try to access our app using curl command.



```
Kubernetes
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help

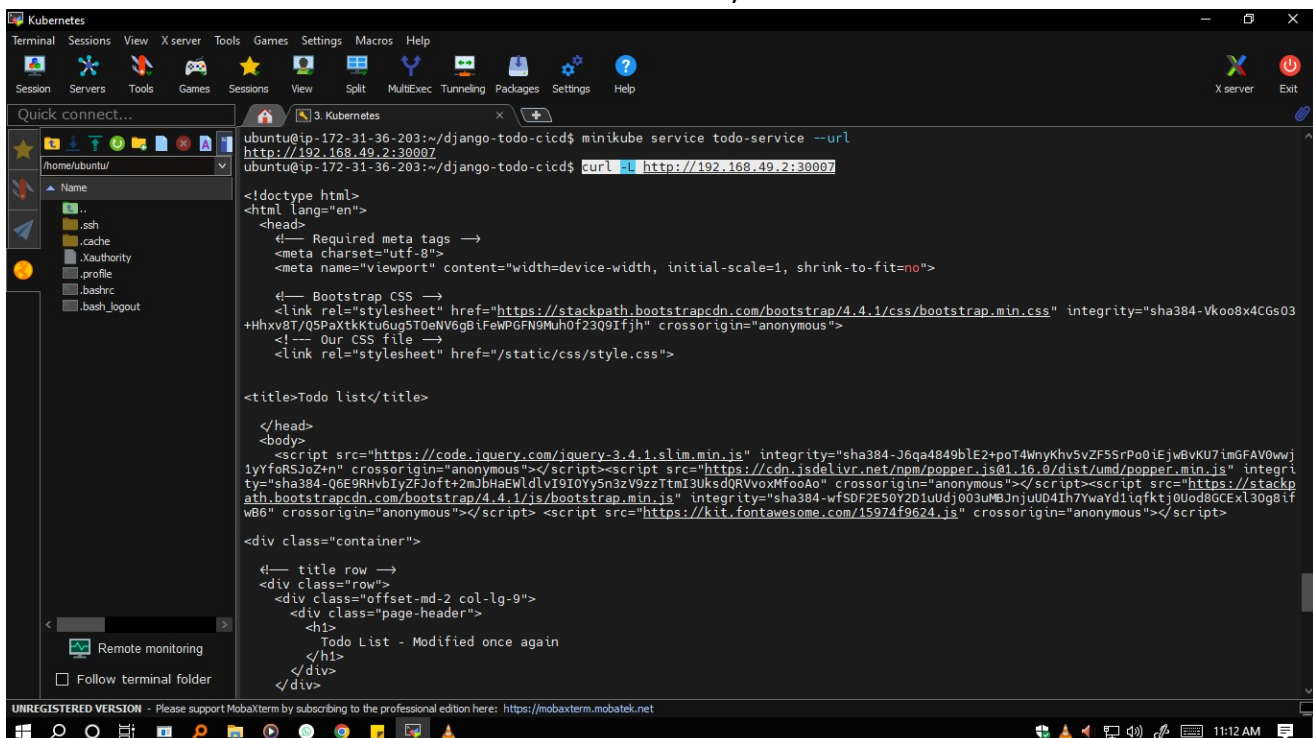
Quick connect...
/home/ubuntu/
Name
..
.ssh
.cache
.Xauthority
.profile
.bashrc
.bash_logout

Remote monitoring
Follow terminal folder

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

ubuntu@ip-172-31-36-203:~/django-todo-cicd$ minikube service todo-service --url
http://192.168.49.2:30007
ubuntu@ip-172-31-36-203:~/django-todo-cicd$
```

23. When we try to access the app using curl command we get the response. (Curl command shows only the content not the UI. If we use any other Kubernetes Cluster like MicroK8s or Kubeadm then we will be able to access our app in the browser and interact it with the UI).



```
Kubernetes
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help

Quick connect...
/home/ubuntu/
Name
..
.ssh
.cache
.Xauthority
.profile
.bashrc
.bash_logout

Remote monitoring
Follow terminal folder

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

ubuntu@ip-172-31-36-203:~/django-todo-cicd$ minikube service todo-service --url
http://192.168.49.2:30007
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ curl -I http://192.168.49.2:30007

<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGs03
+Hhvxv8T/QSPaXtkKtU6ug5T0eNV6g8iFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">
    <!-- Our CSS file -->
    <link rel="stylesheet" href="/static/css/style.css">

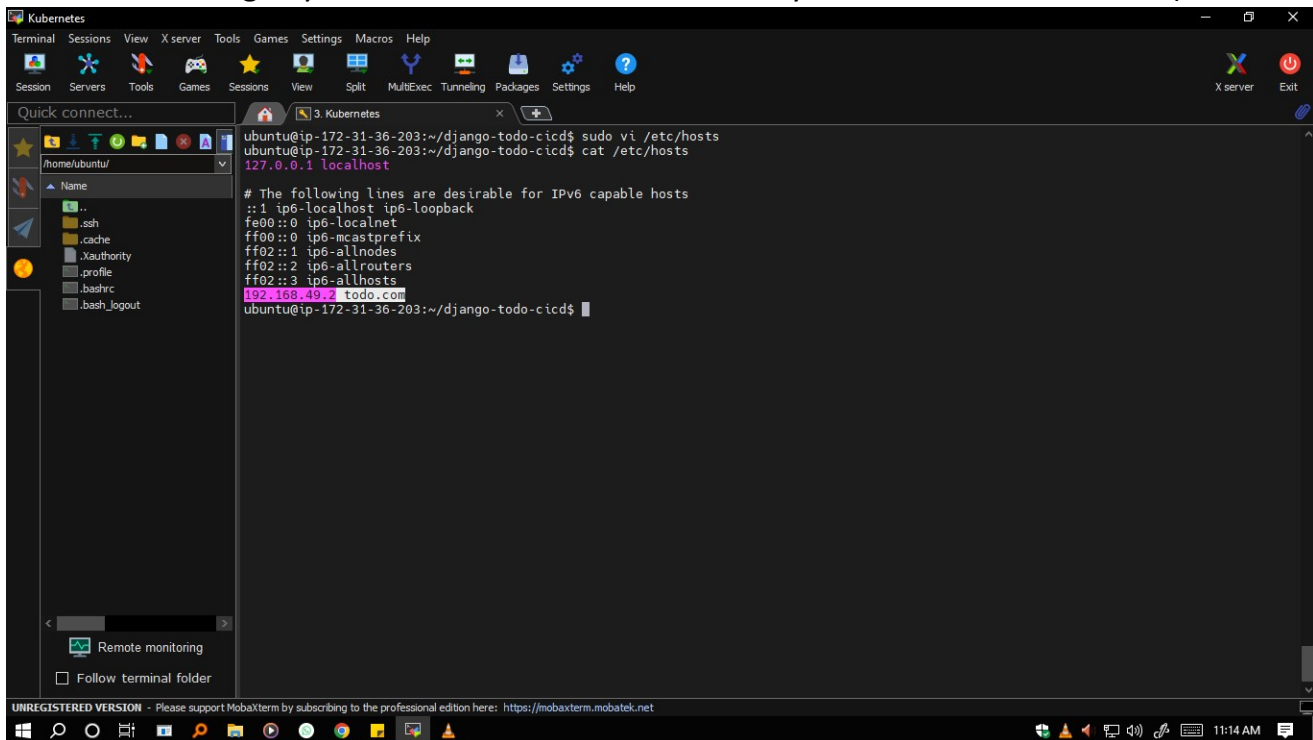
  <title>Todo list</title>

  </head>
  <body>
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SRPo0iEjwBvKU7imGFAV0w
1yYfoR5JoZ+n" crossorigin="anonymous"></script><script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integri
ty="sha384-Q6E9RHvIyZFJoft+2mJbHaEwldlV19I0Yy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script><script src="https://stackp
ath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDfzE50Y2D1uUdJ003uMBJnjuUD4Ih7YwaYdi1qfktj00od8GCExl30g8f
w86" crossorigin="anonymous"></script> <script src="https://kit.fontawesome.com/15974f9624.js" crossorigin="anonymous"></script>

    <div class="container">
      <!-- title row -->
      <div class="row">
        <div class="offset-md-2 col-lg-9">
          <div class="page-header">
            <h1>
              Todo List - Modified once again
            </h1>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```



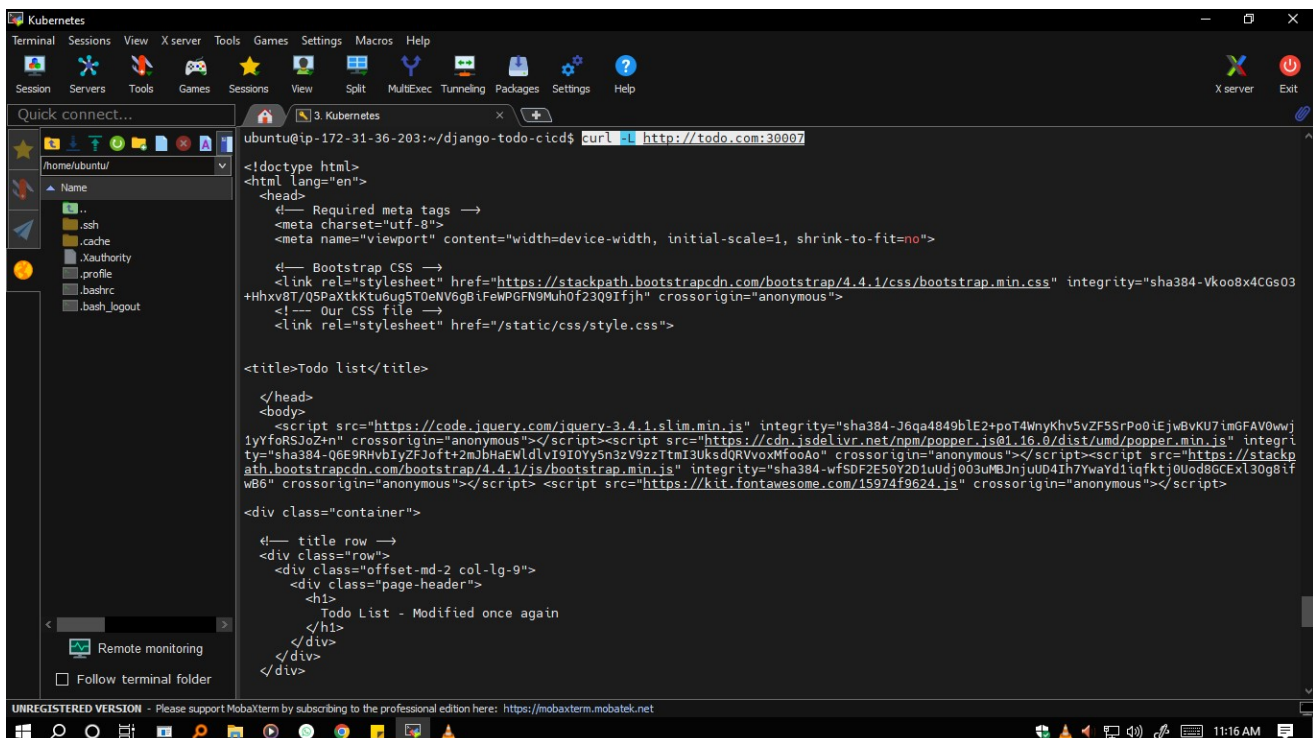
24. We setup a proxy name for our IP in the host file which can be found in “/etc/hosts” so that we don’t need to remember the IP. We can use the name directly. (This domain name as in our example is for our local use only and cannot be used to access from browser if you are using any other Kubernetes Cluster where you are able to view the UI).



```
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ sudo vi /etc/hosts
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
192.168.49.2 todo.com
ubuntu@ip-172-31-36-203:~/django-todo-cicd$
```

25. We are able to access the app using the proxy name we entered in the /etc/hosts file.



```
ubuntu@ip-172-31-36-203:~/django-todo-cicd$ curl http://todo.com:30007
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGs03
+Hhvx8T/qSPaXtkktu6ugST0eNV6gBifEPGFN9Mh0f23Q9Ijfh" crossorigin="anonymous">
    <!-- Our CSS file -->
    <link rel="stylesheet" href="/static/css/style.css">

    <title>Todo list</title>
  </head>
  <body>
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849bLE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj
1yYf0R8l0Zn" crossorigin="anonymous"></script><script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integri
ty="sha384-Q6E9RHvibiyZFJf0t+2mjbHaeWldV1910Yy5n3zV9zzTtm13UksdQ0RVoxMfooAo" crossorigin="anonymous"></script><script src="https://stackp
ath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDf2E50Y2D1uUdj003UMBjnuUD04Ih7Ywayd1iqfktj0UodBGCExl30g8if
wB6" crossorigin="anonymous"></script><script src="https://kit.fontawesome.com/15974f9624.js" crossorigin="anonymous"></script>

    <div class="container">
      <!-- title row -->
      <div class="row">
        <div class="offset-md-2 col-lg-9">
          <div class="page-header">
            <h1>
              Todo List - Modified once again
            </h1>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```