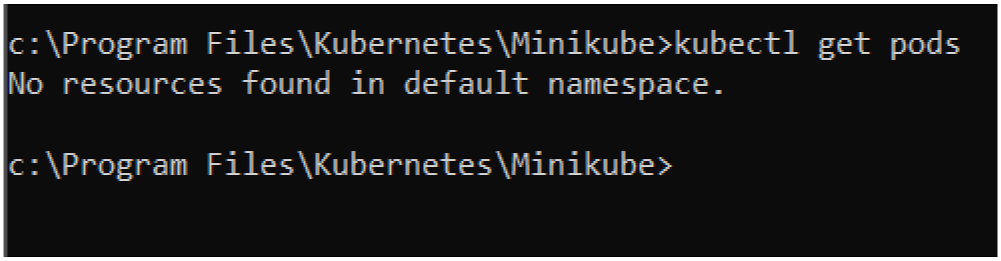
Replication controller

A Replication Controller ensures that a specified number of pod replicas are running at any one time. In other words, a Replication Controller makes sure that a pod or a homogeneous set of pods is always up and available.

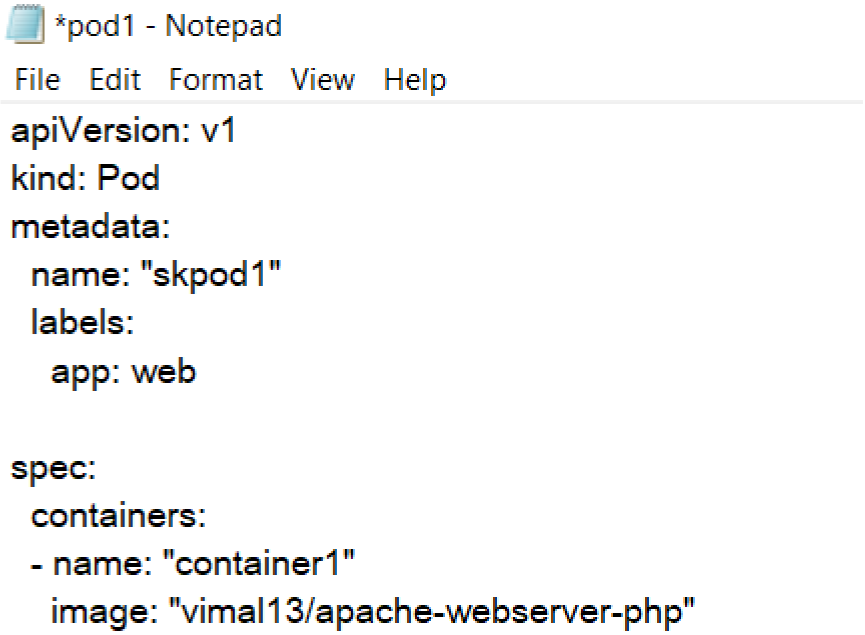
For doing this demo, first we delete all existing pods.



```
c:\Program Files\Kubernetes\Minikube>kubectl get pods
No resources found in default namespace.

c:\Program Files\Kubernetes\Minikube>
```

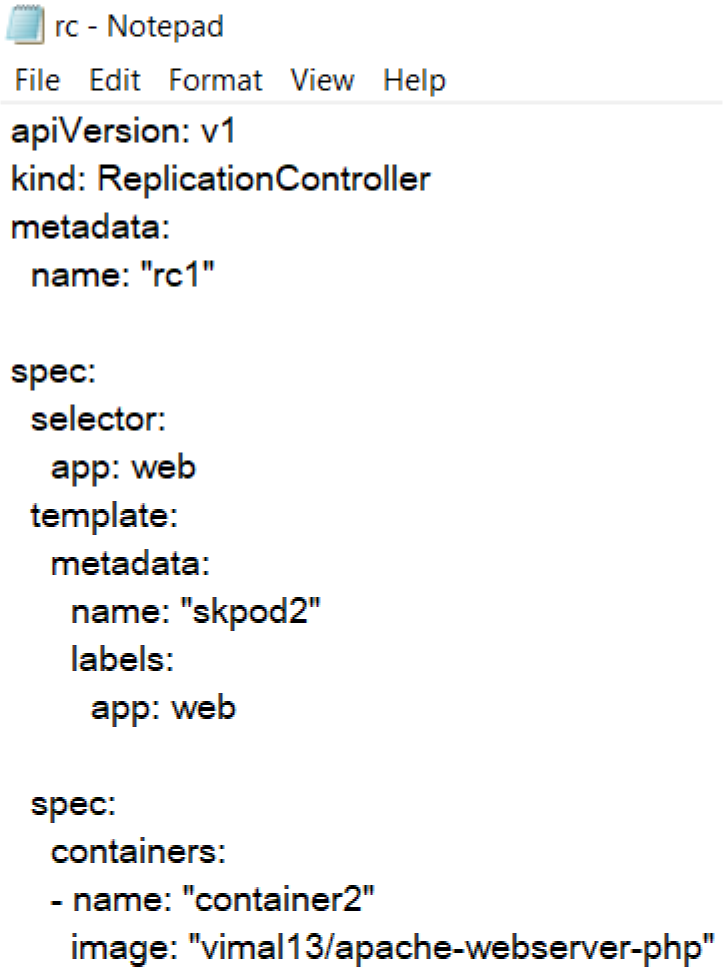
This is the yaml file, in which I have already written the code to launch pods. I have one more command in the 5th and 6th line.

This line means that I have added a label which will be used when I want to create a replica of the existing pod.



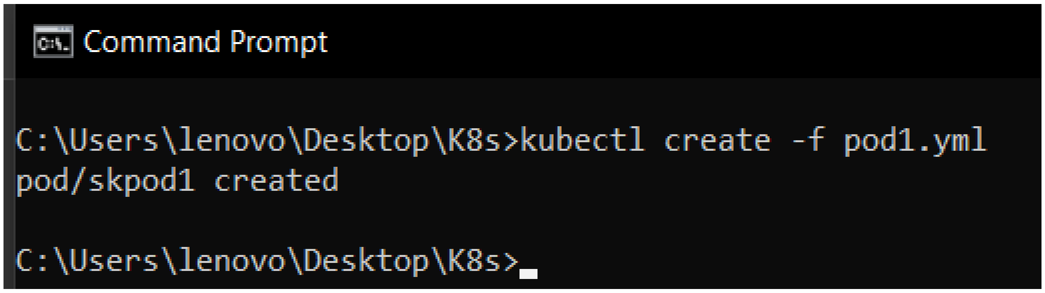
```
apiVersion: v1
kind: Pod
metadata:
  name: "skpod1"
  labels:
    app: web

spec:
  containers:
  - name: "container1"
    image: "vimal13/apache-webserver-php"
```

After that I created a file that would control the replication. I named it rc.yml



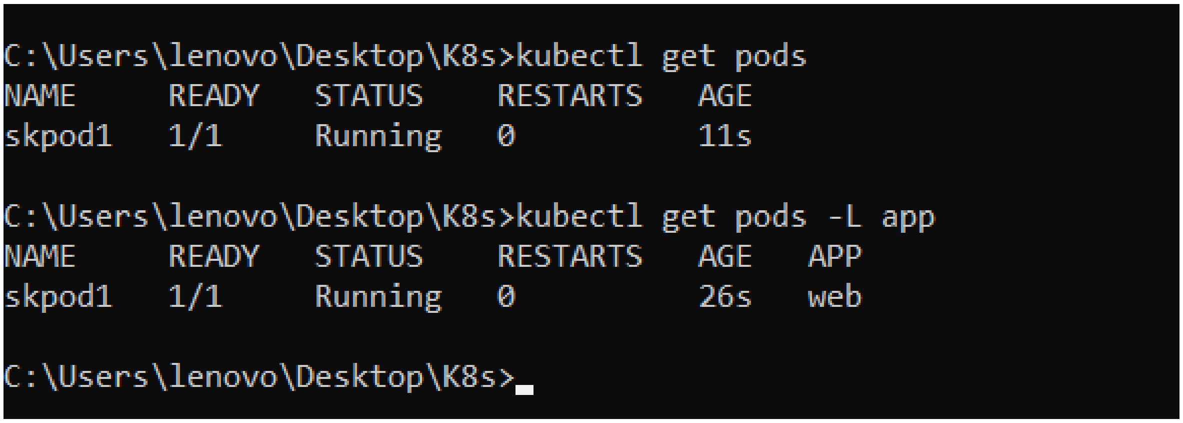
```
apiVersion: v1
kind: ReplicationController
metadata:
  name: "rc1"

spec:
  selector:
    app: web
  template:
    metadata:
      name: "skpod2"
      labels:
        app: web

  spec:
    containers:
    - name: "container2"
      image: "vimal13/apache-webserver-php"
```

Using the pod1.yml file, I launched a pod.



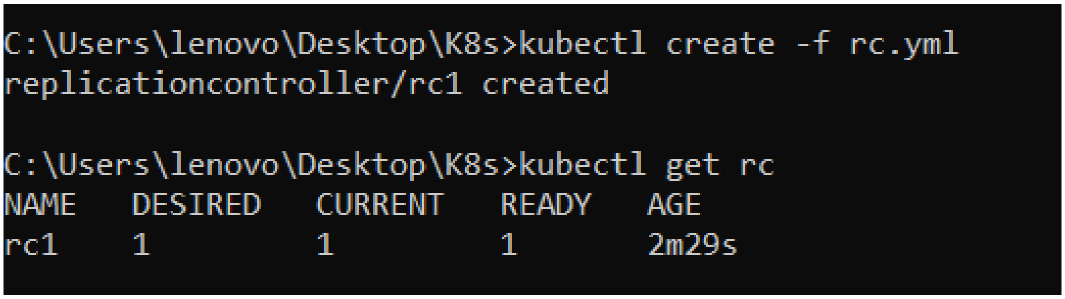
```
C:\Users\lenovo\Desktop\K8s>kubectl create -f pod1.yml
pod/skpod1 created

C:\Users\lenovo\Desktop\K8s>
```

After launching the pod, with the help of command **kubectl get pods –L app**, I check if the pod has been launched with the label name or not.



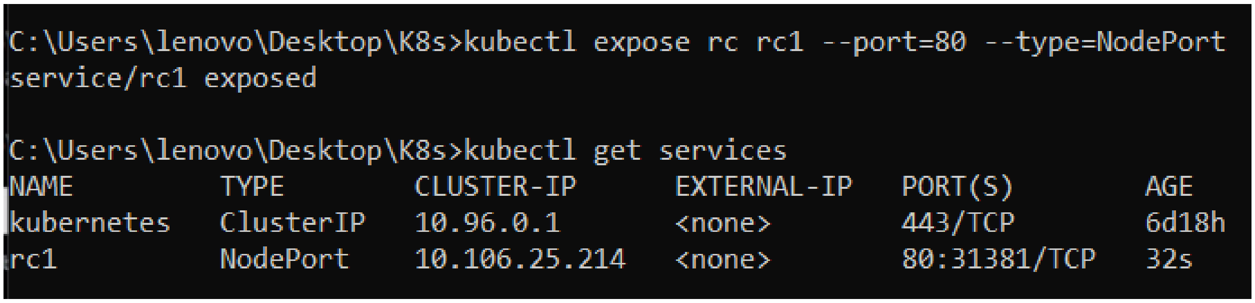
```
C:\Users\lenovo\Desktop\K8s>kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
skpod1   1/1     Running   0          11s

C:\Users\lenovo\Desktop\K8s>kubectl get pods -L app
NAME     READY   STATUS    RESTARTS   AGE   APP
skpod1   1/1     Running   0          26s   web

C:\Users\lenovo\Desktop\K8s>
```

Now using rc.yml file, I create the replication.



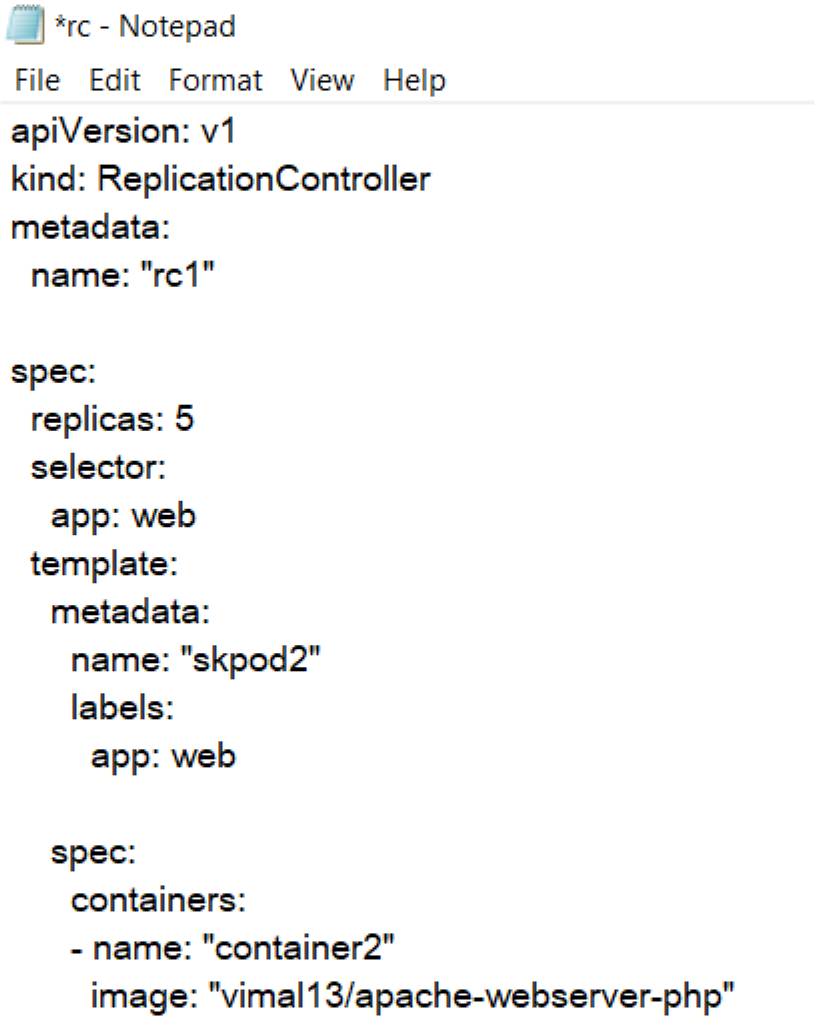
```
C:\Users\lenovo\Desktop\K8s>kubectl create -f rc.yml
replicationcontroller/rc1 created

C:\Users\lenovo\Desktop\K8s>kubectl get rc
NAME   DESIRED   CURRENT   READY   AGE
rc1    1         1         1       2m29s
```

Later, I exposed it to port 80 and check the services



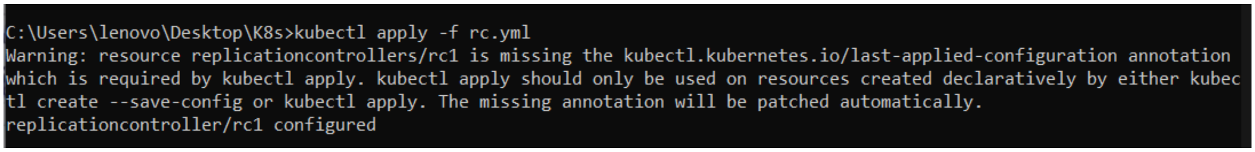
```
C:\Users\lenovo\Desktop\K8s>kubectl expose rc rc1 --port=80 --type=NodePort
service/rc1 exposed

C:\Users\lenovo\Desktop\K8s>kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        6d18h
rc1          NodePort    10.106.25.214   <none>        80:31381/TCP   32s
```

Now, I want 5 more such pods to get launched. For that, I make changes in line 6 of rc.yml. I set the value of replicas 5

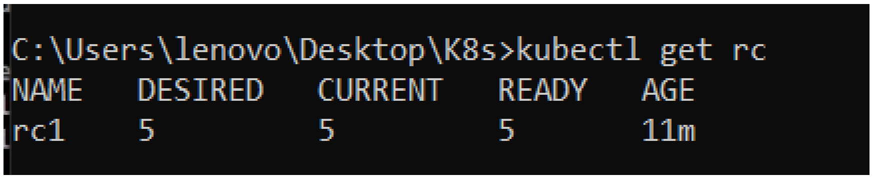


```
apiVersion: v1
kind: ReplicationController
metadata:
  name: "rc1"

spec:
  replicas: 5
  selector:
    app: web
  template:
    metadata:
      name: "skpod2"
      labels:
        app: web

    spec:
      containers:
      - name: "container2"
        image: "vimal13/apache-webserver-php"
```

I use **kubectl apply –f rc.yml** command to apply the changes.

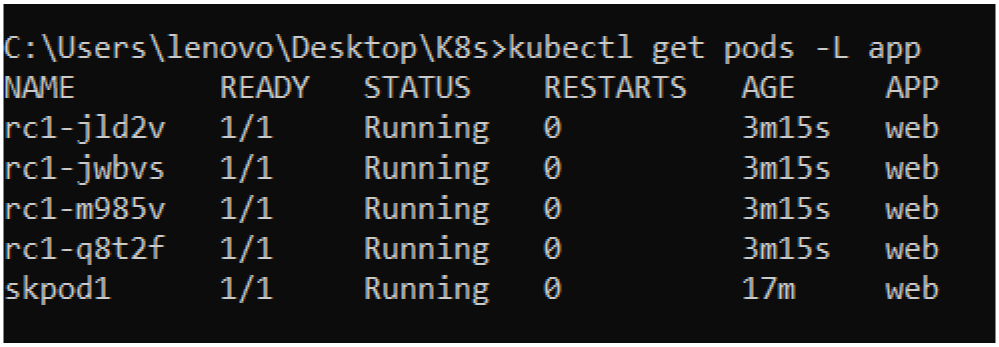


```
C:\Users\lenovo\Desktop\K8s>kubectl apply -f rc.yml
Warning: resource replicationcontrollers/rc1 is missing the kubectl.kubernetes.io/last-applied-configuration annotation
which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubec
tl create --save-config or kubectl apply. The missing annotation will be patched automatically.
replicationcontroller/rc1 configured
```



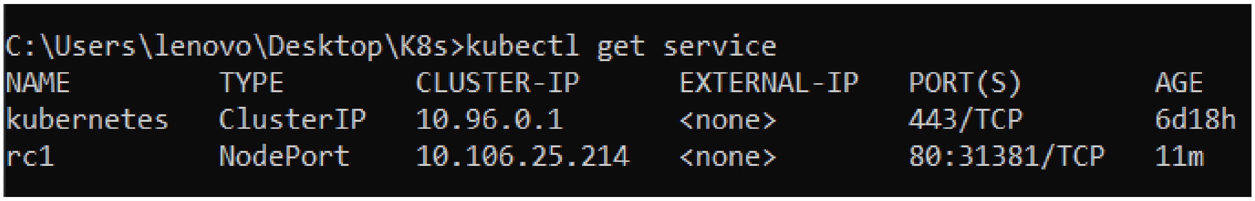
```
C:\Users\lenovo\Desktop\K8s>kubectl get rc
NAME   DESIRED   CURRENT   READY   AGE
rc1    5         5         5       11m
```

With the help of **kubectl get pods –L app**, I can see the details of each pod that is launched.



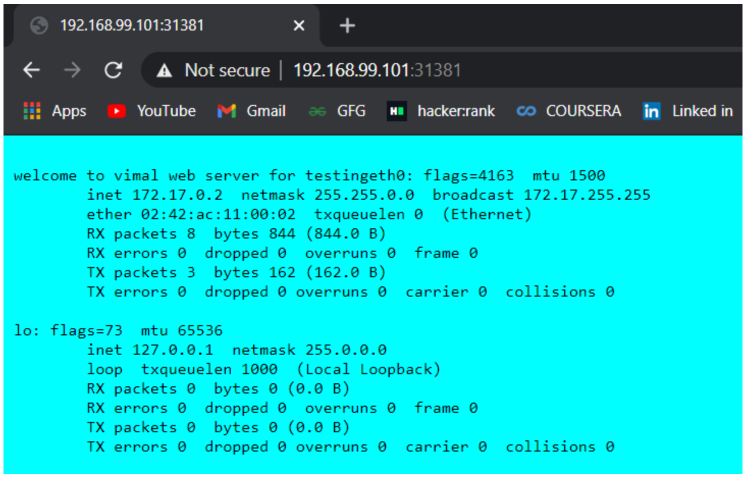
```
C:\Users\lenovo\Desktop\K8s>kubectl get pods -L app
NAME          READY   STATUS    RESTARTS   AGE     APP
rc1-jld2v     1/1     Running   0          3m15s   web
rc1-jwbvs     1/1     Running   0          3m15s   web
rc1-m985v     1/1     Running   0          3m15s   web
rc1-q8t2f     1/1     Running   0          3m15s   web
skpod1        1/1     Running   0          17m     web
```

Using kubectl get services, I get the port number where the rc is running. In this case, it is 31381

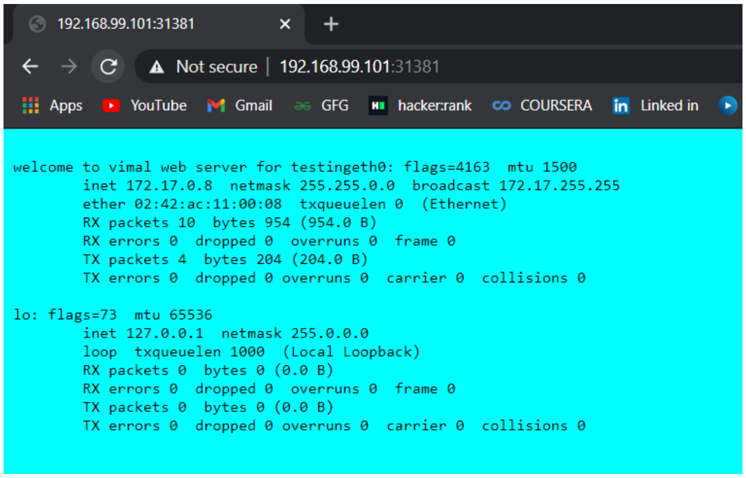


```
C:\Users\lenovo\Desktop\K8s>kubectl get service
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        6d18h
rc1          NodePort    10.106.25.214   <none>        80:31381/TCP   11m
```

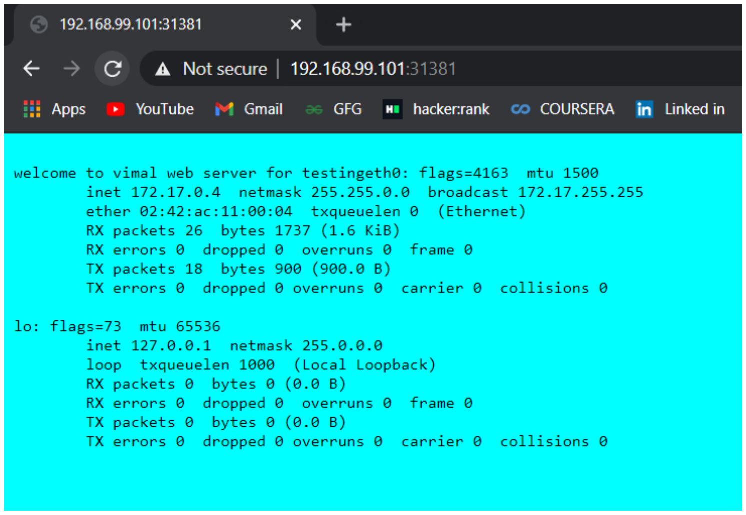
Notice that, even if the IP gets changed on refreshing the page, the contents are still available.



```
welcome to vimal web server for testingeth0: flags=4163  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 8  bytes 844 (844.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3  bytes 162 (162.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



```
welcome to vimal web server for testingeth0: flags=4163  mtu 1500
        inet 172.17.0.8  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:08  txqueuelen 0  (Ethernet)
        RX packets 10  bytes 954 (954.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 204 (204.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



```
welcome to vimal web server for testingeth0: flags=4163  mtu 1500
        inet 172.17.0.4  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:04  txqueuelen 0  (Ethernet)
        RX packets 26  bytes 1737 (1.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 900 (900.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```