Multi-Tier Application Deployment

Kubernetes

WordPress

Step 1: Creating a deployment YAML for MySQL and WordPress without actually applying it to the cluster.

```
root@master:~# kubectl create deployment mysql --image=docker.io/mysql --dry-run=client -o yaml > mysql.yaml
root@master:~# kubectl create deployment wordpress --image=docker.io/wordpress --dry-run=client -o yaml > wp.yaml
root@master:~#
```

Step 2: Edit both YAML file and add enviroment variables.

```
root@master:~# vi mysql.yaml
root@master:~# vi wp.yaml
root@master:~#
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: mysql
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: mysql
    spec:
      containers:
      - image: docker.io/mysql
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: ajinkya
        - name: MYSQL_DATABASE
          value: K8s
        resources: {}
status: {}
~
```

- YAML file of the mysql

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: wordpress
  name: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: wordpress
    spec:
      containers:
      - image: docker.io/wordpress
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: mysql
        - name: WORDPRESS_DB_NAME
          value: K8s
        - name: WORDPRESS_DB_PASSWORD
          value: ajinkya
        - name: WORDPRESS_DB_USER
          value: root
        resources: {}
status: {}
~
```

- YAML file of the WordPress

Step 3: Apply the yaml file of the mysql and wordpress

```
root@master:~# kubectl apply -f mysql.yaml
deployment.apps/mysql created
root@master:~# kubectl apply -f wp.yaml
deployment.apps/wordpress created
root@master:~#
```

Step 4: Fetching the details of pods using the following command.

Kubectl get pods

```
root@master:~# kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
mysql-7d7cfc644-lcsd7        1/1     Running   0          48s
wordpress-68f6bc44cc-4cwdw   1/1     Running   0          42s
root@master:~# kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE   IP          NODE     NOMINATED NODE   READINESS GATES
mysql-7d7cfc644-lcsd7        1/1     Running   0          54s   10.36.0.1   worker   <none>           <none>
wordpress-68f6bc44cc-4cwdw   1/1     Running   0          48s   10.36.0.2   worker   <none>           <none>
root@master:~#
```

Step 5: Creating the service and give the port number to mysql and wordpress.

```
root@master:~# kubectl expose deployment mysql --port=3306
service/mysql exposed
root@master:~# kubectl expose deployment wordpress --port=80 --type=NodePort
service/wordpress exposed
root@master:~#
```

Step 6: Fetch the service of wordpress and mysql

```
root@master:~# kubectl get svc
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        2d5h
mysql        ClusterIP   10.107.124.116   <none>        3306/TCP       9m13s
wordpress    NodePort    10.111.104.147   <none>        80:31837/TCP   8m43s
root@master:~#
```

Step 7: Fetch the deployments

```
root@master:~# kubectl get deployment
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mysql       1/1     1            1           13m
wordpress   1/1     1            1           13m
root@master:~#
```

Step 8: Describe the service. Mysql and WordPress.

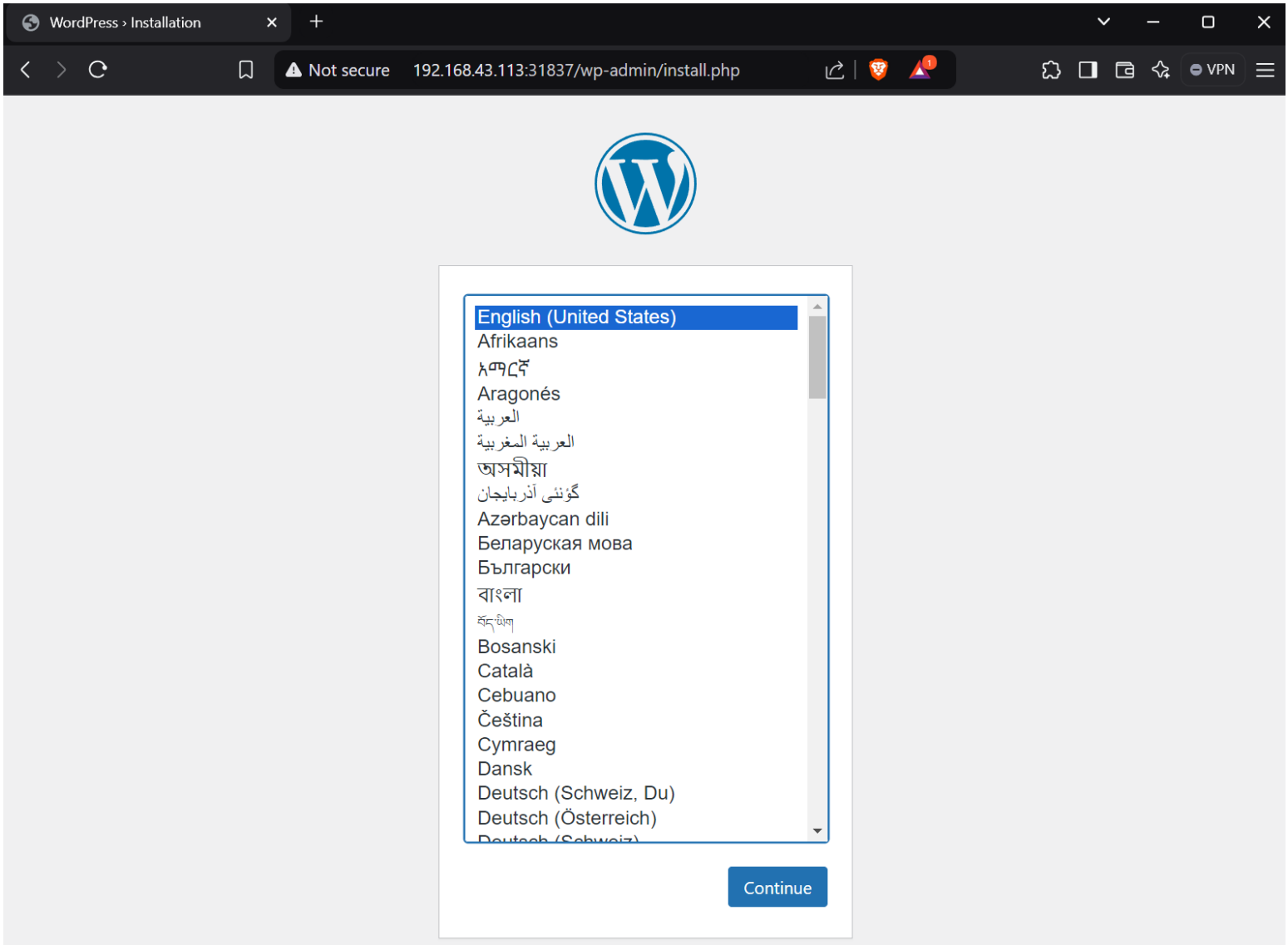
- Msql

```
root@master:~# kubectl describe svc mysql
Name:              mysql
Namespace:         default
Labels:            app=mysql
Annotations:       <none>
Selector:          app=mysql
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.107.124.116
IPs:               10.107.124.116
Port:              <unset>  3306/TCP
TargetPort:        3306/TCP
Endpoints:         10.36.0.1:3306
Session Affinity:  None
Events:            <none>
root@master:~#
```

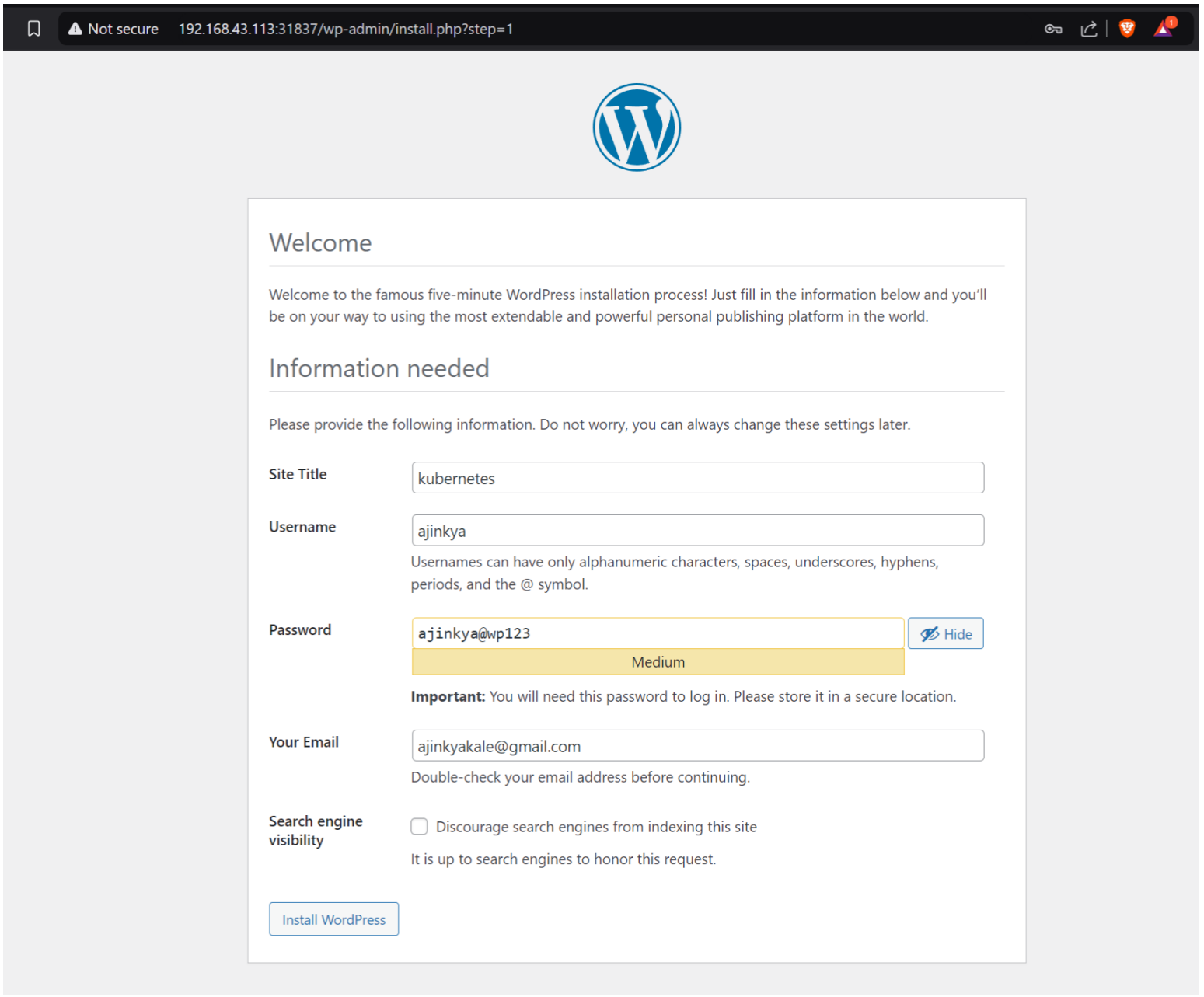
- WordPress

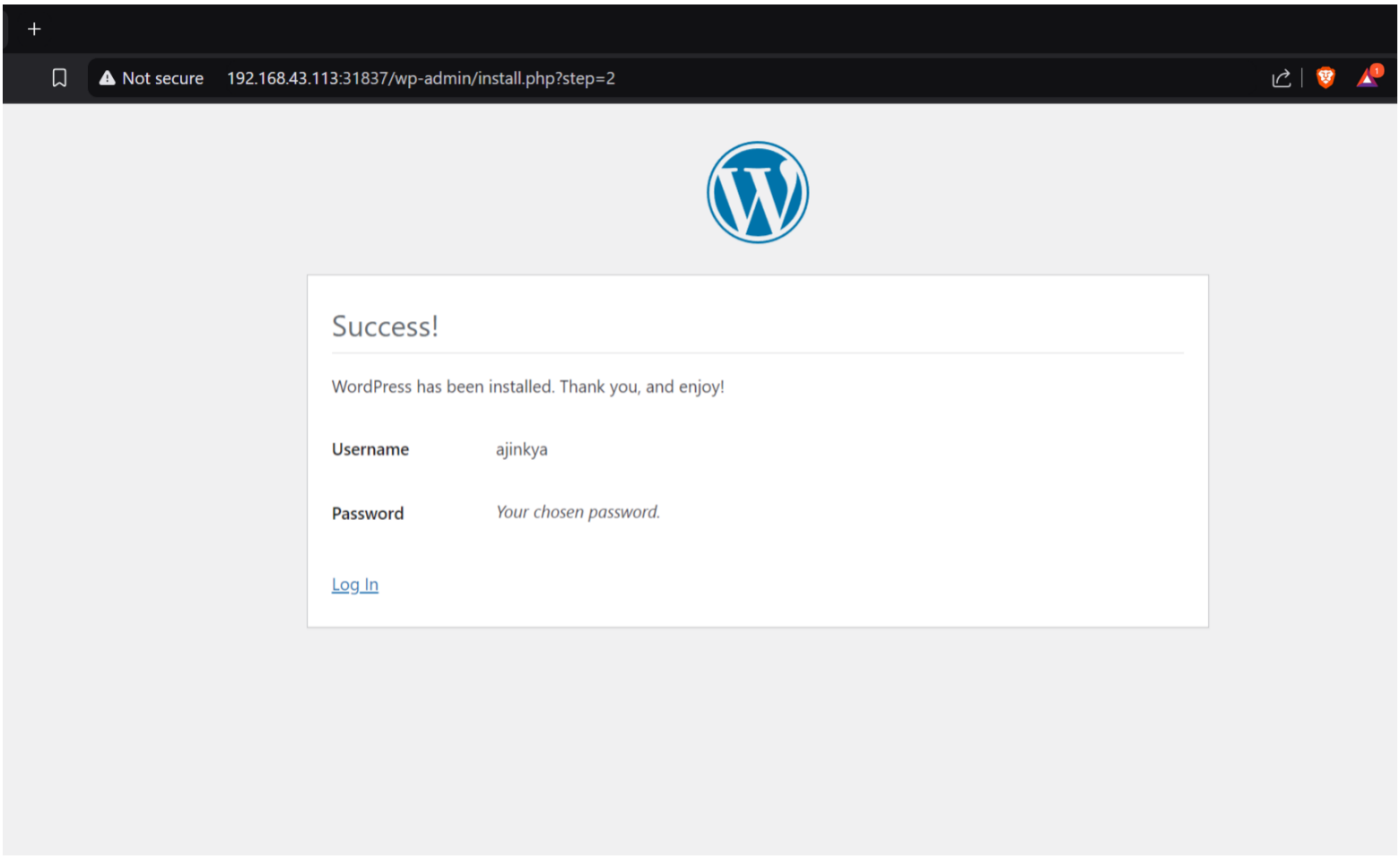
```
root@master:~# kubectl describe svc wordpress
Name:                     wordpress
Namespace:                default
Labels:                   app=wordpress
Annotations:              <none>
Selector:                 app=wordpress
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.111.104.147
IPs:                      10.111.104.147
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31837/TCP
Endpoints:                10.36.0.2:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
root@master:~#
```

Step 9: After entring the the system’s IP with correct port number i.e NodePort described in the service.

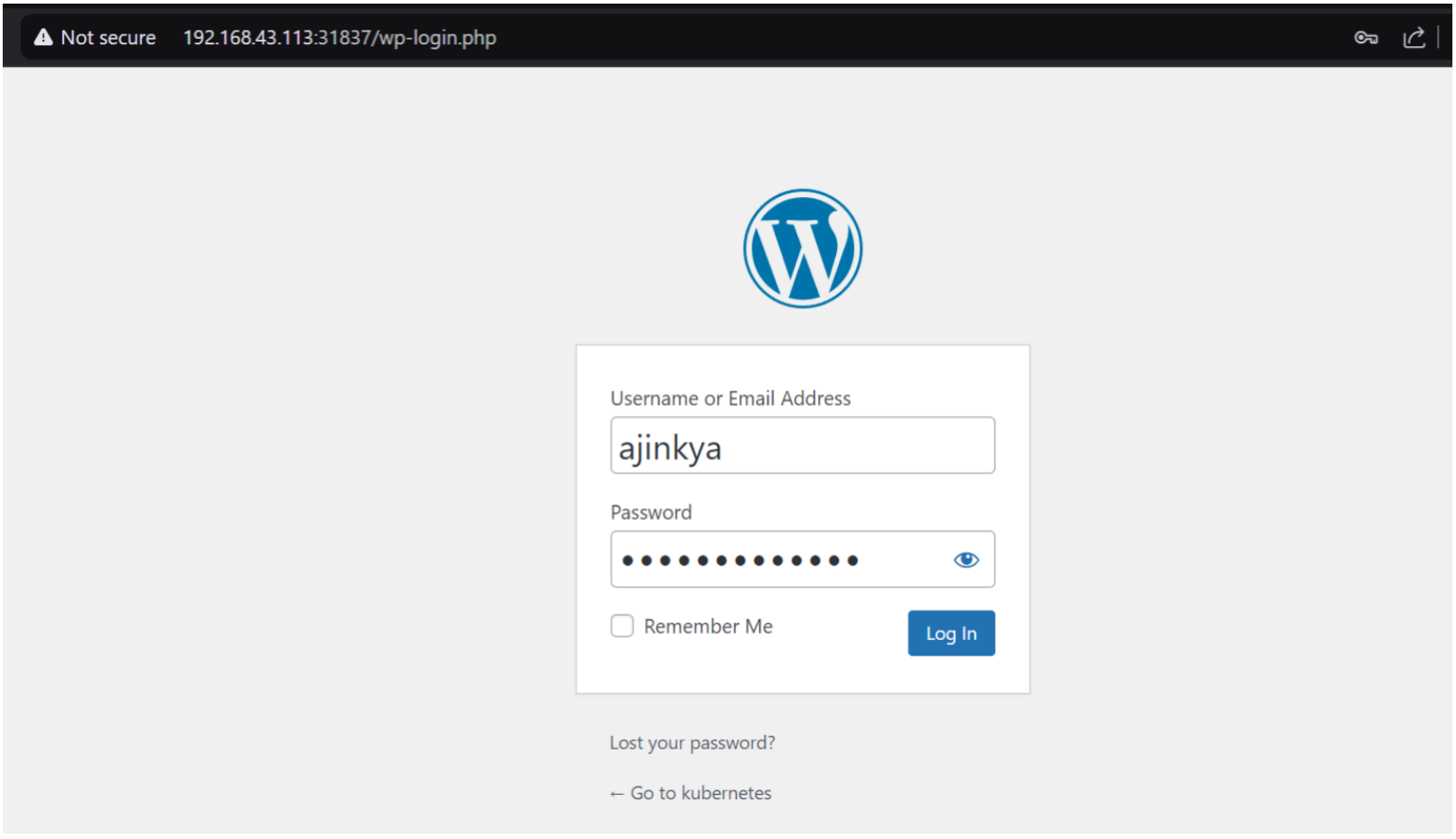


- Enter the correct credential

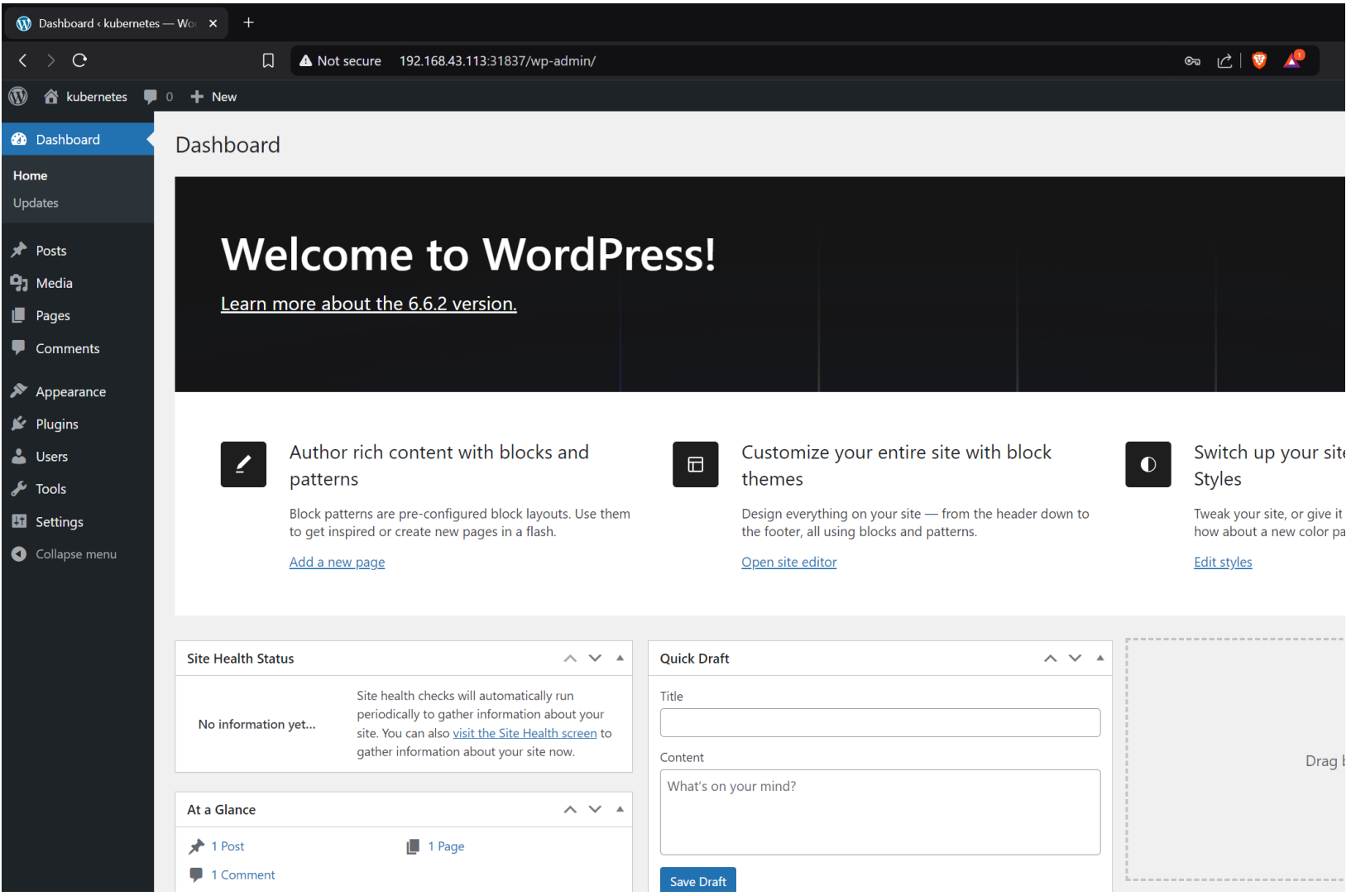




- Entering the correct login credentails.



- Welcome page of WordPress.



Ajinkya Kale