Lab 4.1 - Installing a Chart With Custom Values

Let’s combine our knowledge of Helm chart templating and chart repositories to install a third-party application. For this, we will use the **`bitnami/wordpress`** chart from before to install a new WordPress site.

Note: Before going any further, you must make sure your Kubernetes cluster has some default storage class defined as well as cluster DNS.

Step 1

You can check for if you have a default storage class using the following command:

```
$ kubectl get storageclass
No resources found in default namespace.
```

Step 2

This indicates you do not have any storage providers. If running on MicroK8s, this is easily fixed by running the following command:

```
$ microk8s enable storage
Enabling default storage class
deployment.apps/hostpath-provisioner created
storageclass.storage.k8s.io/microk8s-hostpath created
serviceaccount/microk8s-hostpath created
clusterrole.rbac.authorization.k8s.io/microk8s-hostpath created
clusterrolebinding.rbac.authorization.k8s.io/microk8s-hostpath created
Storage will be available soon
```

Step 3

After some time, you should see the **`microk8s-hostpath`** storage class available:

```
$ kubectl get storageclass
NAME                          PROVISIONER            RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
microk8s-hostpath (default)   microk8s.io/hostpath   Delete
Immediate           false                  41s
```

For more information about Kubernetes storage, please see https://kubernetes.io/docs/concepts/storage/storage-classes/.

Step 4

To see if DNS is enabled in your cluster, check for the **`kube-dns`** service in the **`kube-system`** namespace:

```
$ kubectl get service -n kube-system
No resources found in kube-system namespace.
```

Step 5

If running on MicroK8s, you can enable DNS with the following command:

```
$ microk8s enable dns
Enabling DNS
Applying manifest
serviceaccount/coredns created
configmap/coredns created
deployment.apps/coredns created
service/kube-dns created
clusterrole.rbac.authorization.k8s.io/coredns created
clusterrolebinding.rbac.authorization.k8s.io/coredns created
Restarting kubelet
DNS is enabled
```

Step 6

After some time, you should see a **`kube-dns`** service in the **`kube-system`** namespace:

```
$ kubectl get service -n kube-system
NAME       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)
```

```
AGE
kube-dns    ClusterIP   10.152.183.10   <none>
53/UDP,53/TCP,9153/TCP   71s
```

You are now ready to move on.

Let’s say we want to expose traffic to the WordPress site on the host machine at port 30001. To do so, we will need to use a **`NodePort`** service type, which is a type of Kubernetes service which listens on the same port on each node it is running on.

Upon inspecting the **`values.yaml`** file in the **`bitnami/wordpress`** chart, we see that we can modify the service type using the **`service.type`** value, and the port using the **`service.nodePorts.http`** value.

Lastly, let's use an exact version of the Helm chart and container image so we don't run into any unexpected issues when Bitnami updates this chart. We will use the chart version 9.2.5.

Step 7

To install the **`bitnami/wordpress`** chart with our custom version and values, run the following:

```
$ helm install wordpress bitnami/wordpress \
     --version 9.2.5 \
      --set service.type=NodePort \
      --set service.nodePorts.http=30001
NAME: wordpress
LAST DEPLOYED: Tue May 19 05:51:08 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
** Please be patient while the chart is being deployed **

To access your WordPress site from outside the cluster follow the
steps below:

1. Get the WordPress URL by running these commands:

   export NODE_PORT=$(kubectl get --namespace default -o
jsonpath="{.spec.ports[0].nodePort}" services wordpress)
   export NODE_IP=$(kubectl get nodes --namespace default -o
jsonpath="{.items[0].status.addresses[0].address}")
   echo "WordPress URL: http://$NODE_IP:$NODE_PORT/"
   echo "WordPress Admin URL: http://$NODE_IP:$NODE_PORT/admin"

2. Open a browser and access WordPress using the obtained URL.

3. Login with the following credentials below to see your blog:

  echo Username: user
  echo Password: $(kubectl get secret --namespace default wordpress -o
jsonpath="{.data.wordpress-password}" | base64 --decode)
```

Step 8

After a couple minutes, after the container images have downloaded, and WordPress is connected successfully to its MariaDB database (defined as a dependency chart), you should see two pods running:

```
$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
wordpress-5c8cc7769-897rh   1/1     Running   0          110s
wordpress-mariadb-0         1/1     Running   0          110s
```

Step 9

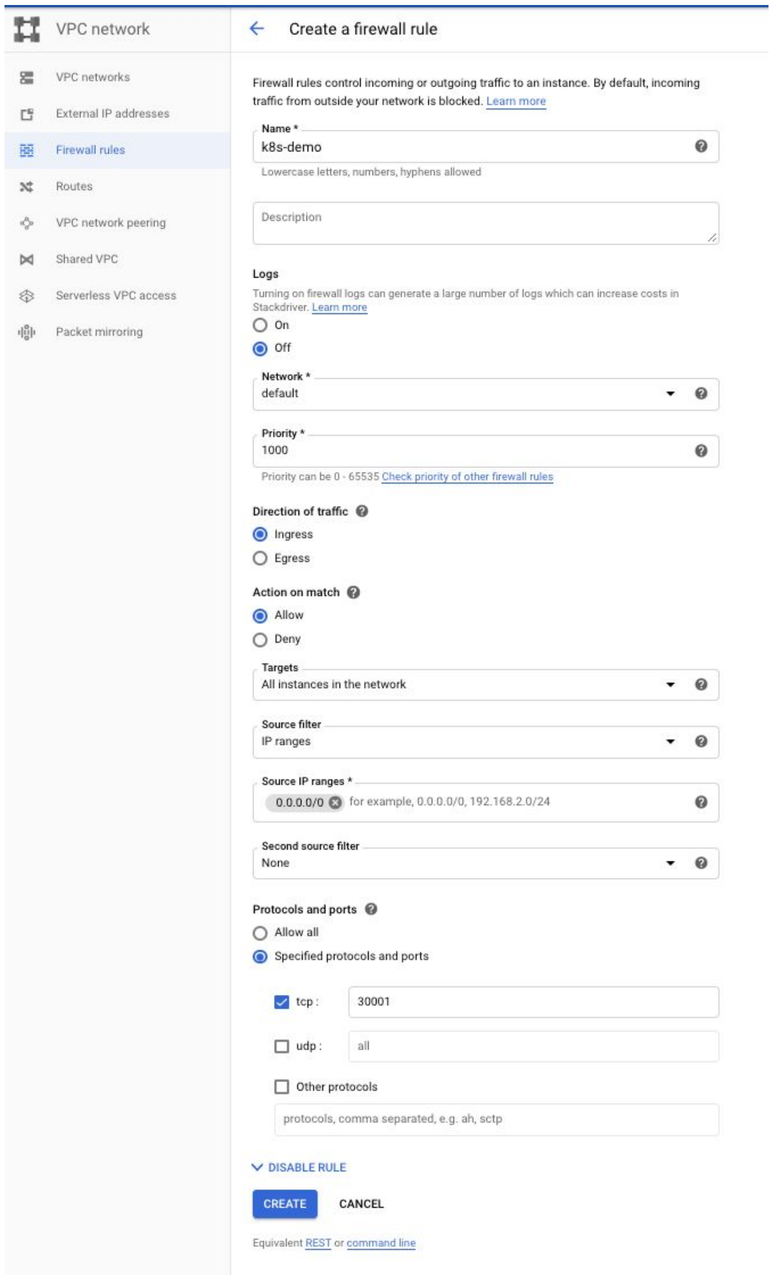
Additionally, you should see the NodePort service running on port 30001:

```
$ kubectl get service
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kubernetes          ClusterIP   10.152.183.1     <none>        443/TCP
14d
wordpress           NodePort    10.152.183.210   <none>
80:30001/TCP,443:31265/TCP   2m20s
wordpress-mariadb   ClusterIP   10.152.183.178   <none>
3306/TCP                     2m20s
```

Identify the public IP on the server you have been running these commands on. Then, in your web browser, navigate to http://<ip>:30001.

Note: If you have trouble connecting, make sure to add a firewall rule allowing inbound access to port 30001. For source, use either 0.0.0.0/0 (open to internet), or <your_ip>/32 (just your IP *address).*

In Google Cloud, this looks like the following:



Create a firewall rule

Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. Learn more

Name *: k8s-demo
Lowercase letters, numbers, hyphens allowed

Description

Logs
Turning on firewall logs can generate a large number of logs which can increase costs in Stackdriver. Learn more
On [✓ Off]

Network *: default

Priority *: 1000
Priority can be 0 - 65535 Check priority of other firewall rules

Direction of traffic
[✓ Ingress] Egress

Action on match
[✓ Allow] Deny

Targets: All instances in the network

Source filter: IP ranges

Source IP ranges *: 0.0.0.0/0 for example, 0.0.0.0/0, 192.168.2.0/24

Second source filter: None

Protocols and ports
Allow all [✓ Specified protocols and ports]

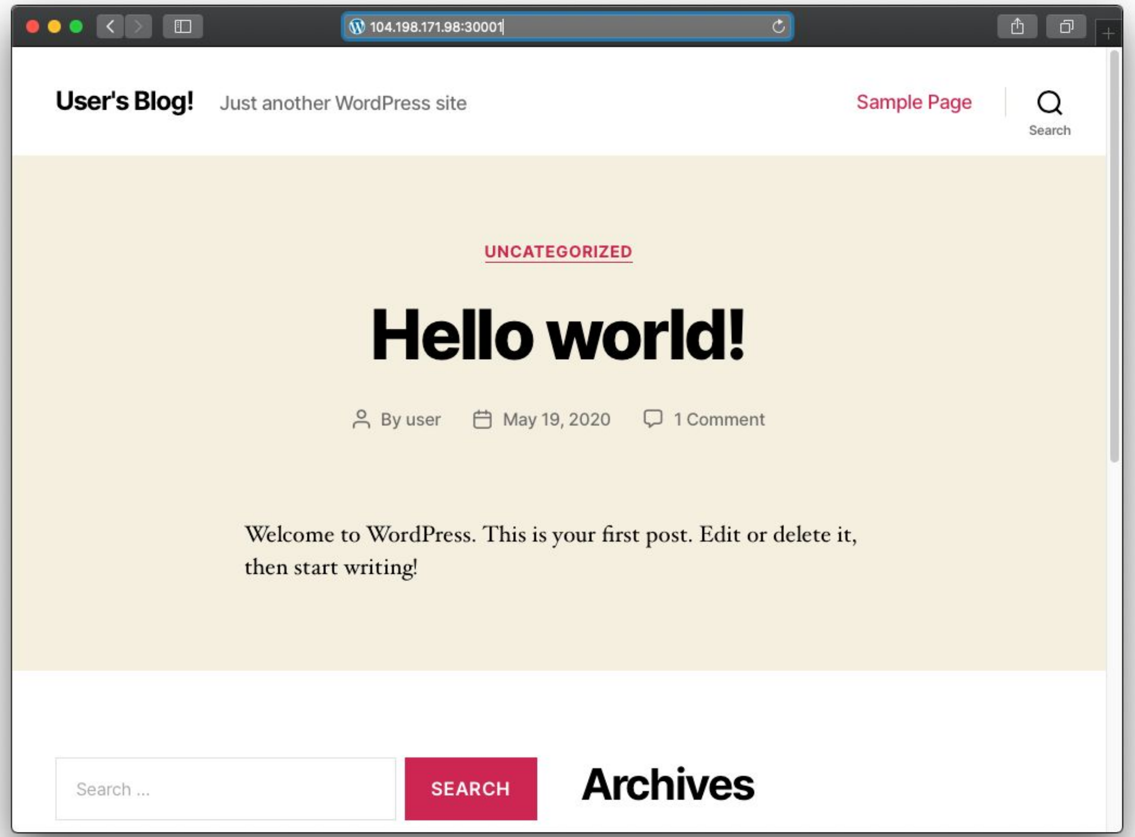
[✓] tcp : 30001
[ ] udp : all
[ ] Other protocols
protocols, comma separated, e.g. ah, sctp

DISABLE RULE

CREATE CANCEL

Equivalent REST or command line

If things are working properly, you should see a fresh WordPress installation in your browser:

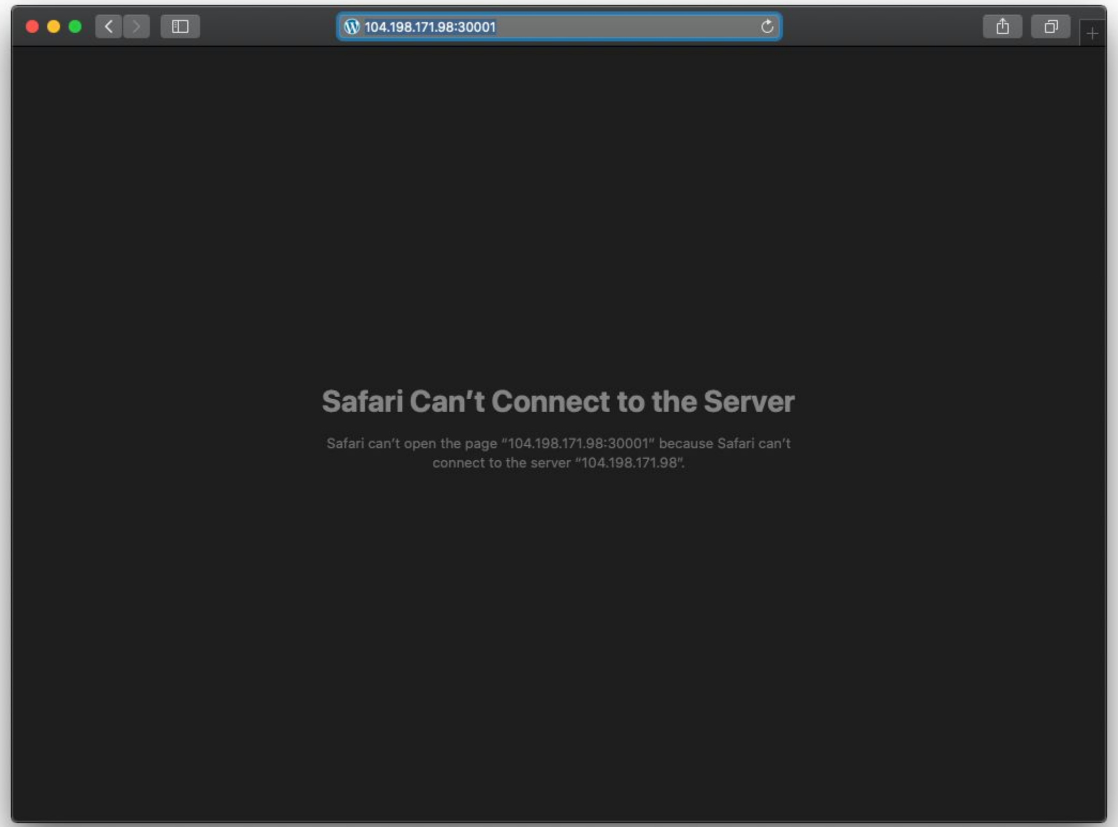


Step 10

To cleanup this install, you can run **`helm delete`**:

```
$ helm delete wordpress
release "wordpress" uninstalled
```

You'll notice if you refresh your browser after this, the site should be down:



104.198.171.98:30001

Safari Can't Connect to the Server

Safari can't open the page "104.198.171.98:30001" because Safari can't connect to the server "104.198.171.98".

Optionally, you can also remove the firewall rule you created if that step was required.