How to ingress in a next layer Kubernetes Cluster

For publishing web services in the current version (v1) of our next layer Kubernetes Cluster it is necessary to setup an ingress controller inside of the cluster. It is up to the customer which software he uses for this purpose. If you need further support from our engineers we recommend using Traefik.

Concept

The concept of using an ingress controller in a next layer managed Kubernetes Cluster is pretty simple. Just deploy the software by using your preferred way (yaml deployment files / helm / etc) and create it's services as type `NodePort`. As soon as the service is deployed inside of the cluster, you will get random generated or manually defined ports which we can use to route your traffic from our load balancer into your cluster.

These ports will be used by us to balance the traffic to your Kubernetes nodes and further on into your cluster. The installed **CNI** will ensure that the traffic is routed to the node where it belongs to.

Step by step (example)

1. Install helm on your local machine or bastion host.
   Helm | Installing Helm

2. Download and extract our provided traefik installation example, change into that directory and run the helm command to deploy traefik. You can modify the file called `values.yml` to customize the deployment so it fit's your needs.

   ```
   helm install traefik traefik/traefik ––namespace=traefik ––values=values.yml
   ```

3. After running this command, you should be able to see a running pod and some services in the `traefik` namespace. If you take a closer look on the output of `kubectl get services –n traefik` you will see at least one service of type `NodePort` and it's ports.

   These ports are the information we need to balance traffic into your cluster. Please just

provide us this output and add the information which public port should route to which nodeport.

```
PORT(S)
80:32123TCP,443:32763/TCP
```

The description which will be provided to us could look like this:

```
PORT(S)
1.2.3.4:80 -> 32123
1.2.3.4:443 -> 32763
```

4. Now you can add Traefik IngressRoutes and define, which domain should be proxied to which kubernetes service.

As an example:

```
---
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: mywebservice-web
  namespace: mywebservice
spec:
  entryPoints:
    - web
  routes:
  - match: Host(`mywebservice.nextlayer.at`)
    kind: Rule
    services:
    - name: mywebservice-web
      kind: Service
      namespace: mywebservice
      port: 80
```

And for tls connections:

```
---
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: mywebservice-web-tls
  namespace: mywebservice
spec:
  entryPoints:
    - websecure
  routes:
  - match: Host(`mywebservice.nextlayer.at`)
    kind: Rule
    services:
    - name: mywebservice-web
      kind: Service
      namespace: mywebservice
      port: 80
  tls:
    certResolver: leresolver
    domains:
    - main: mywebservice.nextlayer.at
```

In this example we are using the `leresolver` which is a preconfigured certificate resolver you can find in our config example. It will try to create let's encrypt ssl certificates by using http acme verification.