Kubernetes Nginx Ingress Controller

Installation Kubernetes Nginx Ingress Controller

Zur einfachen Installation des Kubernetes Nginx Ingress Controllers solltest Du Helm verwenden. Helm bezeichnet sich selbst als Paketmanager für Kubernetes-Anwendungen. Neben der Installation bietet Helm auch einfache Updates seiner Anwendungen. Wie auch bei kubectl brauchst Du nur die K8s-Config, um direkt loszulegen:

```
$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
$ helm repo update
$ helm install my-ingress ingress-nginx/ingress-nginx
```

NOTES: The ingress-nginx controller has been installed. It may take a few minutes for the LoadBalancer IP to be available. You can watch the status by running 'kubectl –namespace default get services -o wide -w my-ingress-ingress-nginx-controller'

An example Ingress that makes use of the controller:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example
  namespace: foo
spec:
  ingressClassName: nginx
  rules:
    - host: www.example.com
      http:
        paths:
          - pathType: Prefix
            backend:
              service:
                name: exampleService
                port:
                  number: 80
            path: /
  # This section is only required if TLS is to be enabled for the Ingress
  tls:
    - hosts:
      - www.example.com
      secretName: example-tls
```

If TLS is enabled for the Ingress, a Secret containing the certificate and key must also be provided:

```
k create secret tls stirlingpdf-tls --key
```

```
/home/marko/Downloads/pdftools.tuxnet.lan.key --cert
/home/marko/Downloads/pdftools.tuxnet.lan.crt -n stirlingpdf-testing
```

or

```
  apiVersion: v1
  kind: Secret
  metadata:
    name: example-tls
    namespace: foo
  data:
    tls.crt: <base64 encoded cert>
    tls.key: <base64 encoded key>
  type: kubernetes.io/tls
```

Mit diesen Befehlen startet Helm alle nötigen Komponenten im default Namespace und gibt diesen das Label my-ingress. Für den Nginx Ingress Controller wird ein deployment, ein replicaset und ein pod erstellt. Alle http/s-Anfragen müssen an diesen pod weitergeleitet werden, damit dieser anhand von vHosts und URI-Pfaden die Anfragen sortieren kann. Dafür wurde ein service vom Typ loadbalancer erstellt, welcher auf eine öffentliche IP lauscht und den ankommenden Traffic auf den Ports 443 und 80 an unseren pod weiterleitet. Ein ähnliches Konstrukt wird auch für das default-backend angelegt, auf welche ich hier aber nicht näher eingehe. Damit Du den Überblick nicht verlierst, kannst Du Dir alle beteiligten Komponenten mit kubectl anzeigen lassen:

```
    $ kubectl get all -l app.kubernetes.io/instance=my-ingress
```

Beispielanwendungen: Apache und Nginx