**KUBERNETES INGRESS**

**What is Ingress?**

It’s probably worth a quick introduction to clear things up. Traditionally, you would create a Load Balancer service for each public system you want to expose. This can get rather expensive. Ingress gives you a way to route requests to services based on the request host or path, centralizing a number of services into a single entry point.

**Ingress Resources**

Ingress is split up into two main pieces. The first is an Ingress resource, which defines how you want requests routed to the backing services.

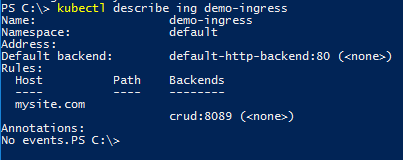
For example, a definition that defines an Ingress to handle requests for **mysite.com** and routes them to the Kubernetes services named website and forums respectively would look like:

|  |
| --- |
| **apiVersion: extensions/v1beta1 kind: Ingress metadata:  name: demo-ingress spec:  rules:  - host: mysite.com 🡪** hostname, for accessing our appn.  **http: 🡪** http/https **paths:  - backend:  serviceName: crud 🡪** service name  **servicePort: 8089 🡪** service port |

**Ingress template🡪ingress.yaml**

The above ingress template will launch an ingress in Kubernetes with name “demo-ingress”.

|  |
| --- |
| **kubectl create –f ingress.yaml** |



**Ingress Controller**

Here is where things seem to get confusing, though. Ingress on its own does not really do anything. *You need something to listen to the Kubernetes API for Ingress resources and then handle requests that match them*. This is where the second piece to the puzzle comes in — the Ingress Controller.

Ingress Controllers can technically be any system capable of reverse proxying, but the most common is Nginx. A full example Nginx Ingress Controller (and LoadBalancer service) is as follows. Please note that if you are not on a provider that supports LoadBalancer services (i.e. bare-metal), you can create a NodePort Service instead and point to your nodes with an alternative solution that fills that role, a reverse proxy capable of routing requests to the exposed NodePort for the Ingress Controller on each of your nodes.

|  |
| --- |
| **kind: Service**  **apiVersion: v1**  **metadata:**  **name: ingress-nginx**  **spec:**  **type: NodePort 🡪** expose the service as NodePort  **selector:**  **app: ingress-nginx**  **ports:**  **- name: http**  **port: 80**  **targetPort: http**  **- name: https**  **port: 443**  **targetPort: https**  **---**  **kind: Deployment**  **apiVersion: extensions/v1beta1**  **metadata:**  **name: ingress-nginx**  **spec:**  **replicas: 1**  **template:**  **metadata:**  **labels:**  **app: ingress-nginx**  **spec:**  **terminationGracePeriodSeconds: 60**  **containers:**  **- image: gcr.io/google\_containers/nginx-ingress-controller:0.8.3**  **name: ingress-nginx**  **imagePullPolicy: Always**  **ports:**  **- name: http**  **containerPort: 80**  **protocol: TCP**  **- name: https**  **containerPort: 443**  **protocol: TCP**  **livenessProbe:**  **httpGet:**  **path: /healthz**  **port: 10254**  **scheme: HTTP**  **initialDelaySeconds: 30**  **timeoutSeconds: 5**  **env:**  **- name: POD\_NAME**  **valueFrom:**  **fieldRef:**  **fieldPath: metadata.name**  **- name: POD\_NAMESPACE**  **valueFrom:**  **fieldRef:**  **fieldPath: metadata.namespace**  **args:**  **- /nginx-ingress-controller**  **- --default-backend-service=$(POD\_NAMESPACE)/nginx-default-backend** |

**Nginx controller template 🡪 ingress\_controller.yaml**

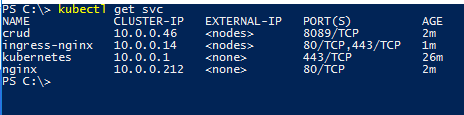
The abovetemplate will launch an Ingress controller deployment and service.

Deploy Nginx controller:

|  |
| --- |
| **kubectl create -f ingress\_controller.yaml** |

Check the Kubernetes services and access the Ingress controller using the NodePort.

|  |
| --- |
| **kubectl get svc** |



**Deploying Application**

Now we need to have an application which the ingress will connect to.

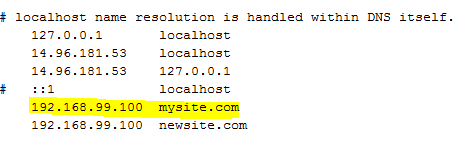
Deploy an application in Kubernetes with service name as **crud** which runs in port 8089.

|  |
| --- |
| **kubectl run crud --image techmid/crudoperations --port=8089** |

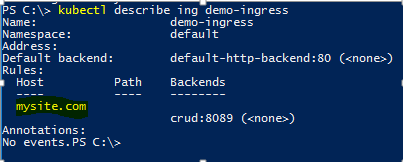
Expose the application as NodePort so that we can access the application directly using the NodePort

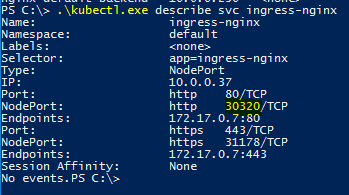
|  |
| --- |
| **kubectl expose deployment/crud --port=8089 --target-port=8089 --type=NodePort** |

Now in the /etc/hosts file in Windows/Linux machine point the hostname in ingress to your ip address of the machine.



So, if we need to access our crud application via ingress gateway use the hostname as in ingress and for port use ingress controller port. Here, the hostname is **mysite.com**





|  |
| --- |
| **http://<host-name>:<nodePort>** |

Eg: **http://mysite.com:30320/insert**

