

HW_4_solution

I've configured ingress:

```
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: my-ingress
5    namespace: {{ .Values.namespace_name }}
6    annotations:
7      nginx.ingress.kubernetes.io/use-regex: "true"
8      nginx.ingress.kubernetes.io/rewrite-target: /$1
9  spec:
10   ingressClassName: nginx
11   rules:
12   - http:
13     paths:
14     - path: /api/resource-service/(.*)
15       pathType: Prefix
16       backend:
17         service:
18           name: resource-service
19           port:
20             number: 8080
```

And installed ingress nginx controller:

```
[robertgazizov@MacBook-Air-Robert helm-chart % kubectl get pod -n ingress-nginx
NAME                                READY   STATUS    RESTARTS   AGE
ingress-nginx-admission-create-mrcx1 0/1     Completed 0           2m39s
ingress-nginx-admission-patch-m7mzg  0/1     Completed 1           2m39s
ingress-nginx-controller-7799c6795f-628lc 1/1     Running   0           2m39s
[robertgazizov@MacBook-Air-Robert helm-chart % kubectl get ingress -n my
NAME          CLASS   HOSTS      ADDRESS      PORTS   AGE
my-ingress    nginx   *          192.168.49.2 80      119s
```

Then I opened minikube tunnel so I am able to access ingress through the localhost:

```
robertgazitov@MacBook-Air-Robert helm-chart % sudo minikube tunnel
Password:
[✓] Tunnel successfully started

📌 NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...

! The service/ingress my-ingress requires privileged ports to be exposed: [80 443]
🔑 sudo permission will be asked for it.
🔧 Starting tunnel for service my-ingress.
```

As a result, I am able to reach the resource-service with new URL structure:

The screenshot shows a REST client interface. The top bar displays a **GET** request to `http://localhost/api/resource-service/actuator/health`. Below the bar, tabs for **Params**, **Authorization**, **Headers (6)**, **Body**, **Pre-request Script**, **Tests**, and **Settings** are visible. The **Body** tab is selected, showing a **200 OK** status. The response body is displayed in **JSON** format, showing a status of `"UP"` and a list of groups: `"liveness"` and `"readiness"`.

```
{
  "status": "UP",
  "groups": [
    "liveness",
    "readiness"
  ]
}
```

and interact with API:

The screenshot shows a REST client interface. The top bar displays a **POST** request to `http://localhost/api/resource-service/resources`. Below the bar, tabs for **Params**, **Authorization**, **Headers (7)**, **Body**, **Pre-request Script**, **Tests**, and **Settings** are visible. The **Body** tab is selected, showing a **200 OK** status. The response body is displayed in **JSON** format, showing a single field: `"id": 4`.

```
{
  "id": 4
}
```