**Microservice Kubernetes Sample with Apache**

This demo uses [Kubernetes](https://kubernetes.io/) as Docker environment. Kubernetes also support service discovery and load balancing. An Apache httpd as a reverse proxy routes the calls to the services.

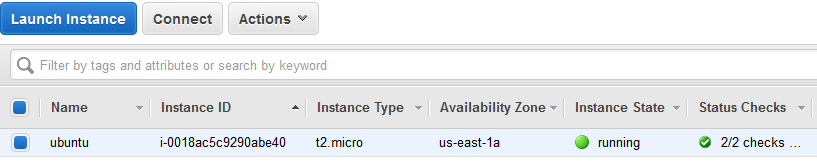
This project creates a complete micro service demo system in Docker containers. The services are implemented in Java using Spring and Spring Cloud.

It uses three microservices:

* Order to process orders.
* Customer to handle customer data.
* Catalog to handle the items in the catalog.

**Prerequisite**

1. For image creation we are using Ubuntu machine 16.04
2. Launch Ubuntu machine in AWS



1. Docker ,java & maven installation in Ubuntu machine
   1. Docker Installation

|  |
| --- |
| # apt install docker.io |

* 1. Java Installtion

|  |
| --- |
| # sudo apt install openjdk-8\* |

* 1. Maven Installation

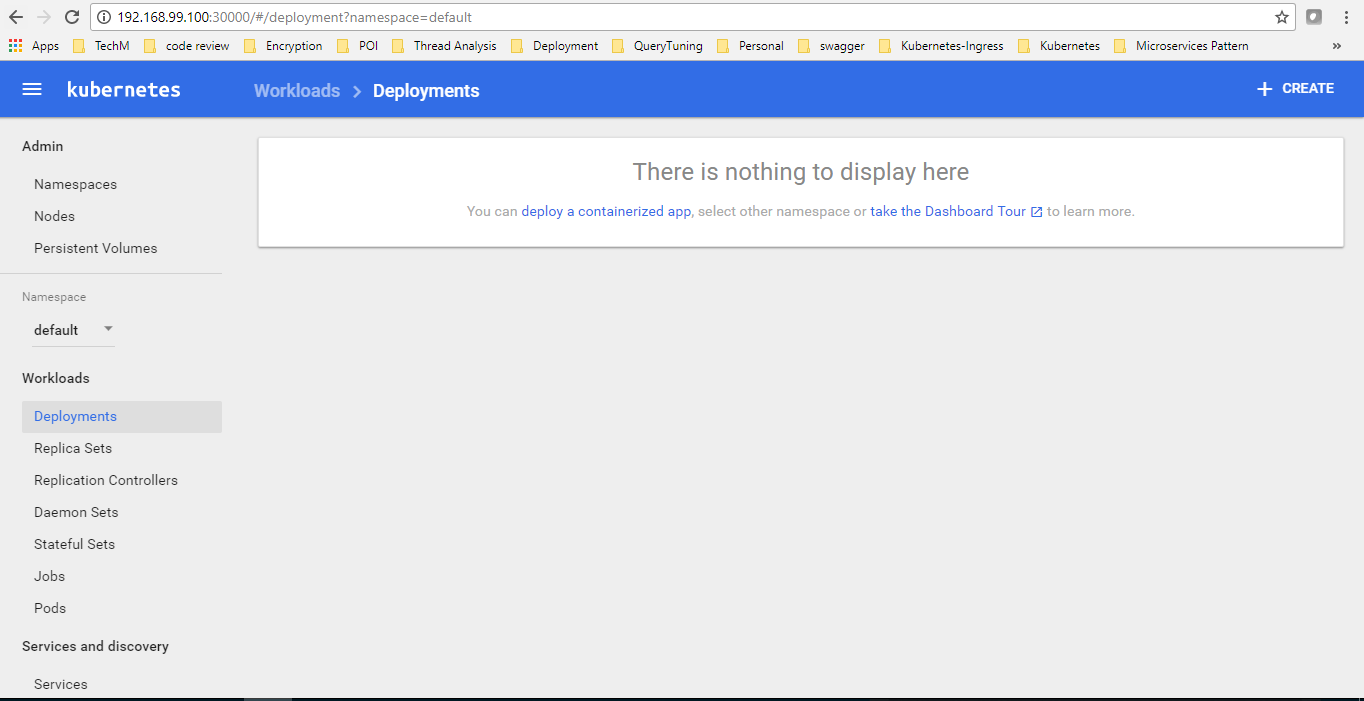
|  |
| --- |
| # sudo apt install maven |

* 1. Using WinSCP copy source code from your machine to Ubuntu environment

1. Install [minikube](https://github.com/kubernetes/minikube/releases) . Minikube is a Kubernetes environment in a virtual machine that is easy to use and install. It is not meant for production but to test Kubernetes or for developer environments.
2. Install [kubectl](https://kubernetes.io/docs/tasks/kubectl/install/). This is the command line interface for Kubernetes.

**How to run**

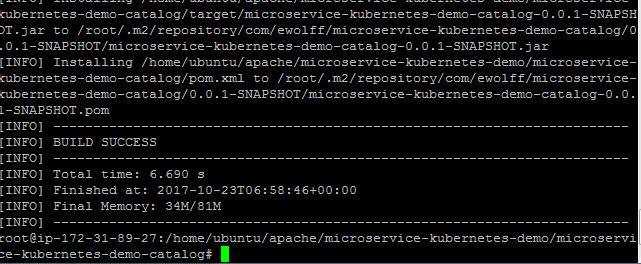
* 1. **Installation (refer :- documents for minikube installation on windows )**



## Build the Docker images

1. **Building .jar**
   1. Navigate to the catalogue service and build the code





Once build successful, jar file will be generated we need to convert .jar to docker images using dockerfile in source directory .

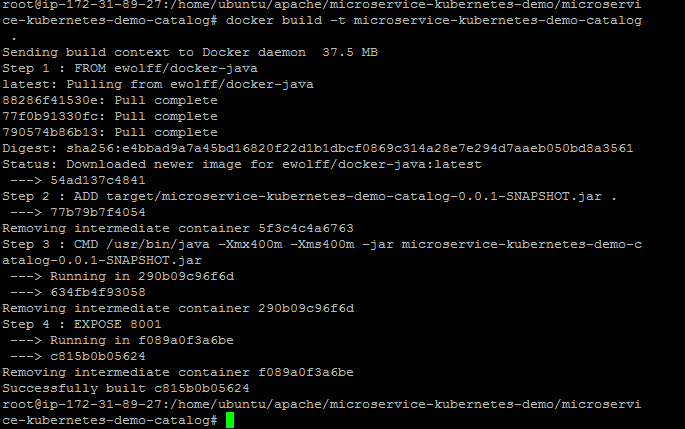
Likewise we need to generate .jar for all other services and apache

1. **Building docker Images**

2.1 .Building image

|  |
| --- |
| # docker build -t microservice-kubernetes-demo-catalog . |

Once you run this command , it will read the docker file and build docker image.



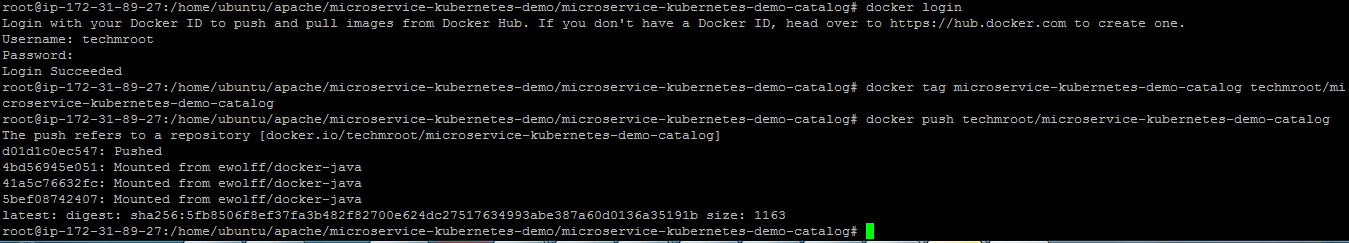
2.2. How to check

|  |
| --- |
| # docker images |



2.3. Pushing image to docker hub

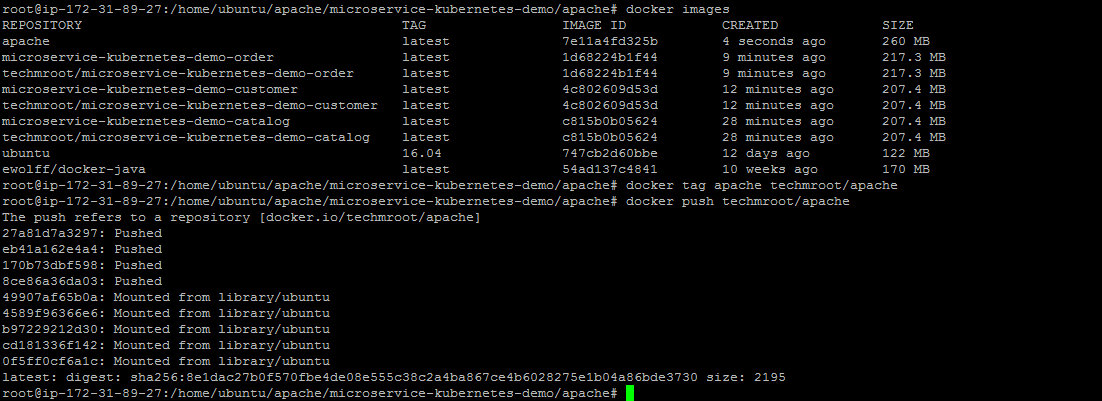
|  |
| --- |
| Docker Login  #docker login  Tag the image  # docker tag microservice-kubernetes-demo-catalog techmroot/microservice-kubernetes-demo-catalog  Pushing the image  # docker push techmroot/microservice-kubernetes-demo-catalog |



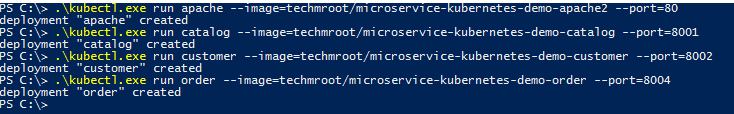
Follow the same procedure to build images of other services

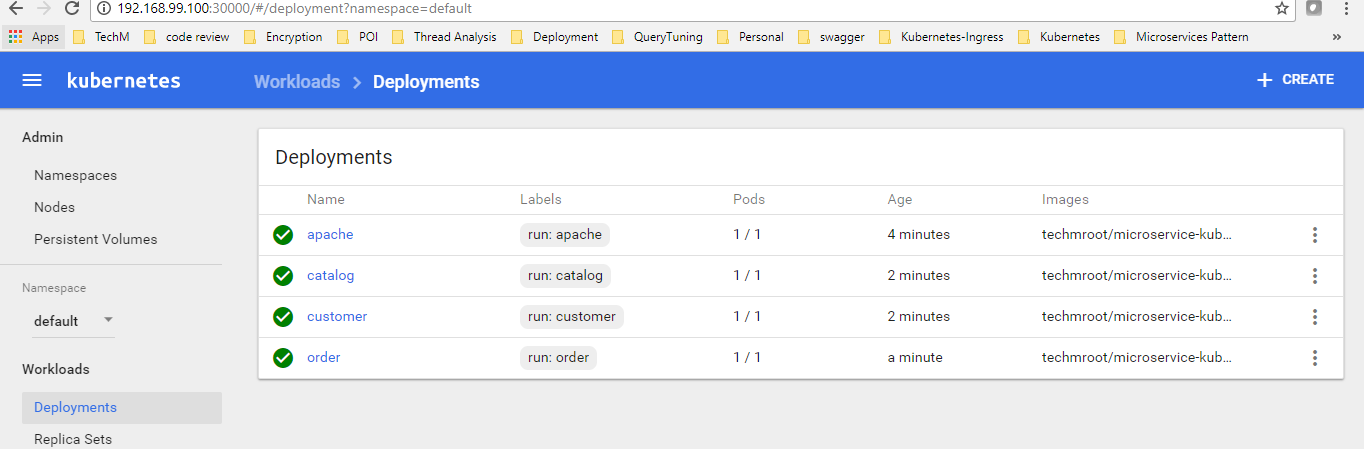
Same as well for **apache**



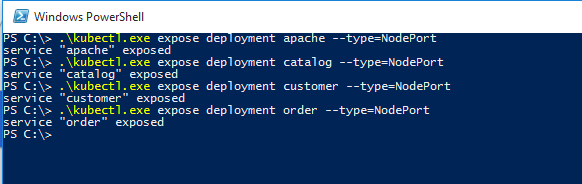


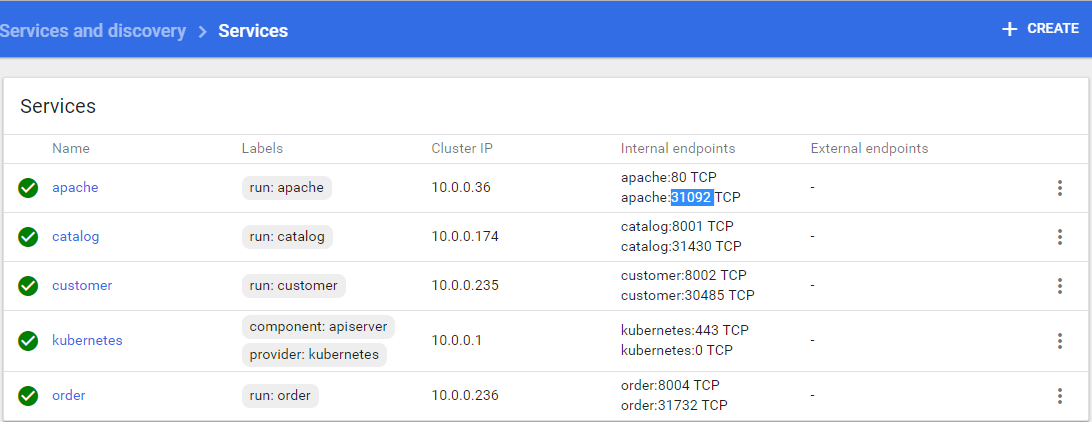
1. **Deploying and exposing docker images on minikue**
   1. Deploying docker images

  
Minikube UI

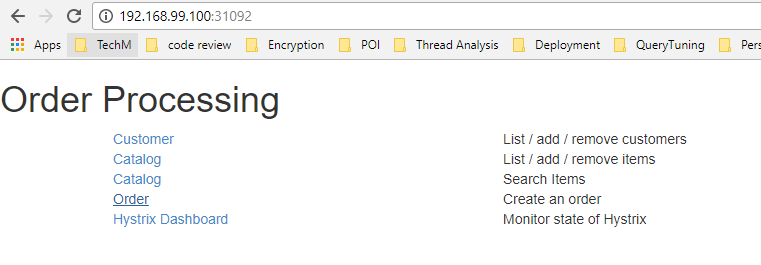


* 1. Exposing the deployed services

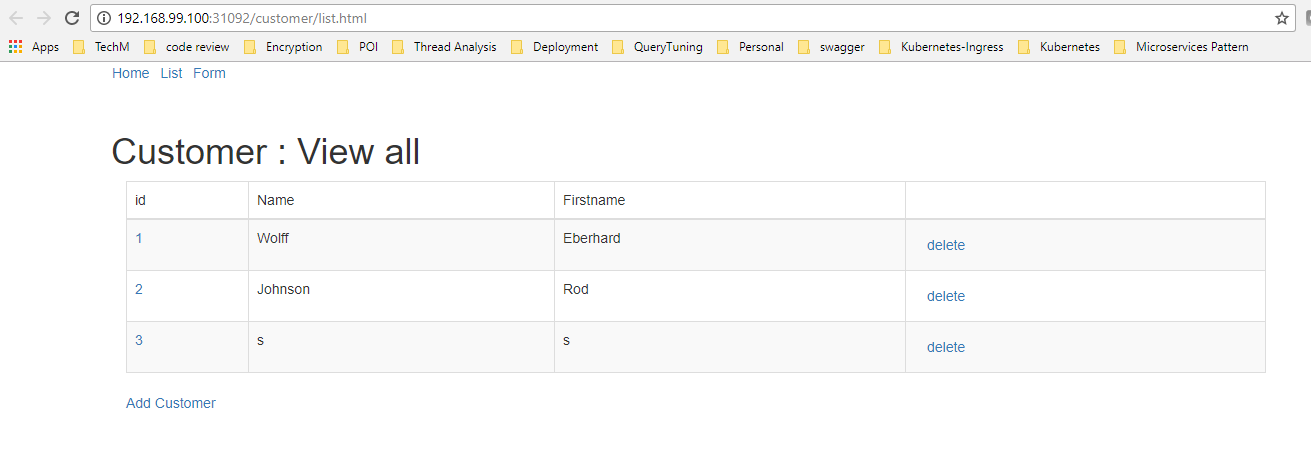


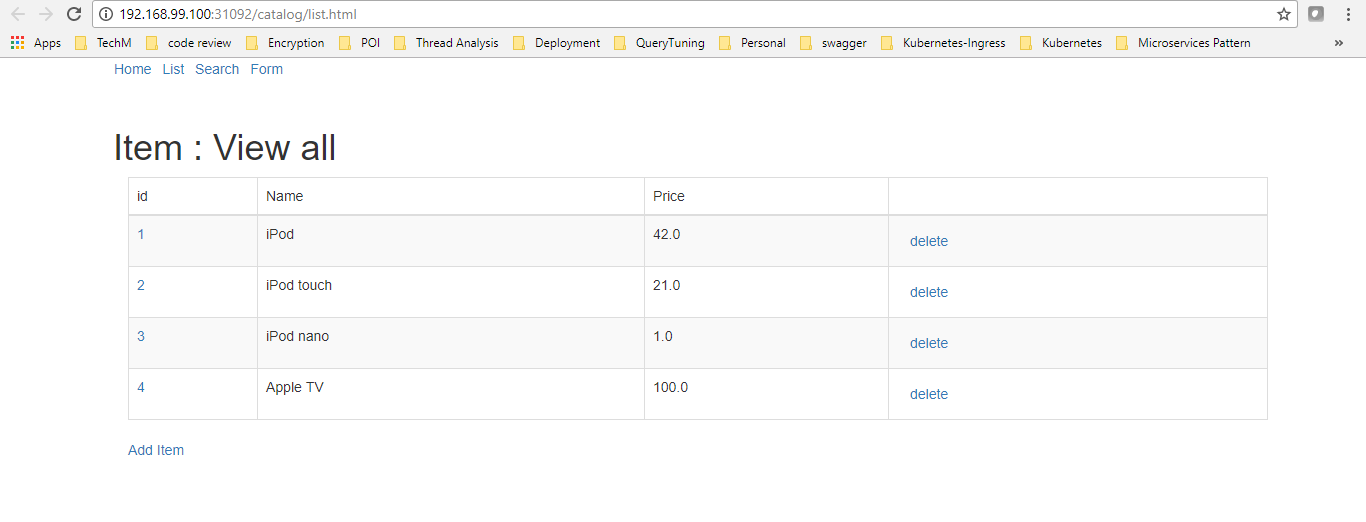


Accessing the application using minikube ip and loadbalancer(apache) nodeport



We can access customer ,catalogue and order service by clicking option in home page ,loadbalancer will redirect to the specified service without changing the port





**Remarks on the code**

* 1. **Apache HTTP Load Balancer**

Apache HTTP is used to provide the web page of the demo at port 8080. It also forwards HTTP requests to the microservices. This is not really necessary as each service has its own port on the Minikube host but it provides a single point of entry for the whole system. Apache HTTP is configured as a reverse proxy for this. Load balancing is left to Kubernetes.

To configure this Apache HTTP needs to get all registered services from Kubernetes. It just uses DNS for that.

Please refer to the subdirectory [**microservice-kubernetes-demo/apache**](https://github.com/ewolff/microservice-kubernetes/blob/master/microservice-kubernetes-demo/apache) to see how this works.

|  |
| --- |
| 000-default.conf -- [**microservice-kubernetes-demo/apache**](https://github.com/ewolff/microservice-kubernetes/blob/master/microservice-kubernetes-demo/apache)**/000-default.conf** |
| <VirtualHost \*:80>  DocumentRoot /var/www/html  # This should be secured!  <Location "/balancer-manager">  SetHandler balancer-manager  </Location>  ErrorLog ${APACHE\_LOG\_DIR}/error.log  CustomLog ${APACHE\_LOG\_DIR}/access.log combined  ProxyPreserveHost On  //**need to mention corresponding service port number as in application.prop -----src/main/resource/ application.prop**  ProxyPass /order http://order:**8004**/  ProxyPassReverse /order http://order:**8004**/  ProxyPass /catalog http://catalog:**8001**/  ProxyPassReverse /catalog http://catalog:**8001**/  ProxyPass /customer http://customer:**8002**/  ProxyPassReverse /customer http://customer:**8002**/  </VirtualHost> |