**Lab for Kubernetes**

# What you need to install

**1.** [**kubectl**](<https://kubernetes.io/docs/tasks/tools/install-kubectl/>)

**2.** [**VirtualBox**](<https://www.virtualbox.org/wiki/Downloads>)

**3.** [**minikube**](<https://github.com/kubernetes/minikube/releases>)

**4.** [**Docker**](<https://docs.docker.com/install/>)

**5.** [**helm**](<https://docs.helm.sh/using_helm/#quickstart-guide>)

# Installing Kubectl

Install Kubectl: <https://kubernetes.io/docs/tasks/tools/install-kubectl/>

# Installing VirtualBox

Install Kubectl: <https://kubernetes.io/docs/tasks/tools/install-kubectl/>

# Installing Minikube

* <https://github.com/kubernetes/minikube/releases>

(if you get permission denied, retry each command (&& is the separator) with sudo

* <https://kubernetes.io/docs/setup/minikube/#installation>
* If you get error: start.go:305] Error restarting cluster: restarting kube-proxy: waiting for kube-proxy to be up for configmap update: timed out waiting for the condition
  + $ minikube delete
  + Always start the minikube with this command:

$ minikube start --vm-driver=virtualbox

* + $ minikube dashboard

# Edit minikube CPUs/memory:

$ minikube stop

$ VBoxManage list vms ## usually the name will be “minikube”

$ VBoxManage showvminfo minikube ## verify the number of CPUs/memory before the change

$ VBoxManage modifyvm minikube --cpus 4

$ VBoxManage modifyvm minikube --memory 4096

$ VBoxManage showvminfo minikube ## verify the number of CPUs/memory after the change

$ minikube start

$ kubectl describe nodes ## verify that the change took place for the minikube itself as well

# Installing Docker

Install Docker: https://docs.docker.com/install/

# Installing Helm

Install helm: <https://docs.helm.sh/using_helm/#installing-helm>

Run: helm init To configure it to use the kubernetes cluster

# Workflow

**Prerequisite**

1. Download InsightEdge zip: https://www.gigaspaces.com/product/insightedge-platform

**Get Started**

1. minikube start
2. minikube dashboard

**First use of Kubernetes + insightEdge:**

1. Create an empty folder:
2. helm repo add gigaspaces https://resources.gigaspaces.com/helm-charts
3. helm repo update
4. export GS\_VERSION=14.0
5. Run: helm install gigaspaces/insightedge --version $GS\_VERSION --name demo
6. Run: helm status demo
7. Run minikube ip
8. To view the deployed pu go to the Rest API - http://<minikube ip>:30890
9. To delete: helm del --purge demo

**Configuring the Kubernetes Service Accounts**

In Kubernetes clusters with RBAC enabled, users can configure Kubernetes RBAC roles and service accounts used by the various Spark jobs on Kubernetes components to access the Kubernetes API server.

Spark on Kubernetes supports specifying a custom service account for use by the Driver Pod via the configuration property that is passed as part of the submit command. To create a custom service account, run the following kubectl command:

kubectl create serviceaccount spark

After the custom service account is created, you need to grant a service account role. To grant a service account a Role, a RoleBinding is needed. To create a RoleBinding or ClusterRoleBinding, use the kubectl create rolebinding (or clusterrolebinding for ClusterRoleBinding) command. For example, the following command creates an edit ClusterRole in the default namespace and grants it to the spark service account you created above.

kubectl create clusterrolebinding spark-role --clusterrole=edit --serviceaccount=default:spark --namespace=default

After the service account has been created and configured, you can apply it in the Spark submit:

--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark

**Running the spark Job**

The insightedge-submit script is located in the InsightEdge home directory, in insightedge/bin. This script is similar to the spark-submit command used by Spark users to submit Spark jobs. The following examples run both a pure Spark example and an InsightEdge example by calling this script.

### **SaveRDD Example**

Run the following InsightEdge submit script for the SaveRDD example, which generates “N” products, converts them to RDD, and saves them to the data grid. This example has the following configuration:

* The --master has the prefix k8s://<Kubernetes Master URL>:<port>.
* The spark.insightedge.space.lookup.locator is set with the headless service of the Manager Pod (<release name>-insightedge-manager-hs).
* The example lookup is the default Space called demo.
* In Kubernetes clusters with RBAC enabled, the service account must be set (e.g. serviceAccountName=spark).
* The spark.kubernetes.container.image is set with the desired Docker image (This is usually of the form gigaspaces/insightedge-enterprise:1.0.0).

./insightedge-submit \  
--master k8s://https://192.168.99.100:8443 \  
--deploy-mode cluster \  
--name i9e-saveRdd \  
--class org.insightedge.examples.basic.SaveRdd \  
--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \  
--conf spark.kubernetes.container.image=gigaspaces/insightedge-enterprise:14.0 \  
--conf spark.insightedge.space.manager=demo \

--conf spark.insightedge.space.name=demo \  
local:///opt/gigaspaces/insightedge/examples/jars/insightedge-examples.jar

Use the Zeppelin UI to query the number of objects in the demo data grid. The output should show 100,000 objects of type org.insightedge.examples.basic.Product.

Port 930990 is the minikube zeppelin port