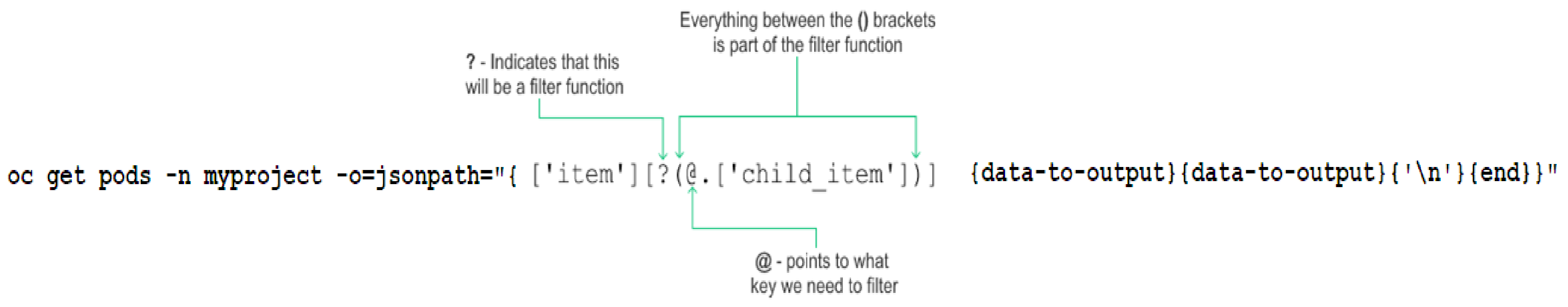
MINISHIFT/OPENSHIFT DOCUMENT

```
https://docs.okd.io/3.11/install/example_inventories.html

# prerequisites

      ❖  disable windows hypervisor and virtual compute platform features from program fatures on/off when using virtualbox
         hypervisor
      ❖  download the minishift release from github and extract in c:\soft\minishift folder

# set the various config parameters:

      https://docs.okd.io/3.11/minishift/command-ref/minishift_config.html

  c:\soft\minishift> minishift config set vm-driver virtualbox

  c:\soft\minishift> minishift config set disk-size 10GB

  c:\soft\minishift> minishift config set memory 6GB

  c:\soft\minishift> minishift config set cpus 4

  c:\soft\minishift> minishift config set skip-check-openshift-release true

# start the cluster which initiates, validates and creates the minishift cluster

  c:\soft\minishift> minishift start

# get the oc path environment and run the below output commands

  c:\soft\minishift> minishift oc-env

    SET PATH=C:\Users\atlantis\.minishift\cache\oc\v3.11.0\windows;%PATH%
    REM Run this command to configure your shell:
    REM     @FOR /f "tokens=*" %i IN ('minishift oc-env') DO @call %i

# login to the minishift cluster

  c:\soft\minishift> minishift console
     url: it opens the url  https://192.168.99.101:8443/console in the browser.

       user id: developer password: developer
       user id: admin  password: admin

# get the login command to log in to the cluster from the oc

  go to the menu written developer in the top right corner and click on
  "copy the login command" the clipboard has the following command

"oc login https://192.168.99.101:8443 --token=GVW14WaeEkTzwehltGEYCrDdGoP03TRgxUyxGVY-am0"

# now login to the cluster using the copied command

  c:\soft\minishift> oc login https://192.168.99.101:8443 --token=GVW14WaeEkTzwehltGEYCrDdGoP03TRgxUyxGVY-am0

# configure the docker env

  c:\soft\minishift> minishift docker-env

    SET DOCKER_TLS_VERIFY=1
    SET DOCKER_HOST=tcp://192.168.99.101:2376
    SET DOCKER_CERT_PATH=C:\Users\atlantis\.minishift\certs
    REM Run this command to configure your shell:
    REM     @FOR /f "tokens=*" %i IN ('minishift docker-env') DO @call %i

# now we can use the docker commands on the local machine
# compile and deploy a java microservice to okd

  https://openliberty.io/guides/okd.html#what-is-origin-community-distribution-of-kubernetes-okd


# create a service account

   c:\soft\minishift> oc create sa <sa-name>

# get service account

  c:\soft\minishift> oc get sa

# create a group ( login as admin )

  c:\soft\minishift> oc adm groups new mygroup

# assign a role to  a group

  c:\soft\minishift> oc policy add-role-to-group edit mygroup

# add a user named melvin to a group named mygroup

  c:\soft\minishift> oc adm groups add-users mygroup melvin

# get the groups

  c:\soft\minishift> oc get groups

# add cluster level role to a user

  c:\soft\minishift> oc adm policy add-cluster-role-to-user cluster-admin melvin

# create a secret from a string literal
```

```
  c:\soft\minishift> oc create secret generic mysecret --from-literal key1=secret1 --from-literal key2=secret2 -n myproj

# create password file for users with htpasswd

  c:\soft\minishift> htpasswd -c users.txt melvin

# create a secret from a htpasswd generated file

  c:\soft\minishift> oc create secret generic mysecret --from-file htpasswd=users.txt -n myproj

# add labels to nodes

  c:\soft\minishift> oc label node hostname env=production

# expose a service

  c:\soft\minishift> oc expose service servcie_name --port 80

# expose an app : get the service for the app and then use the service name to expose the app

  c:\soft\minishift> oc get svc

  c:\soft\minishift> oc expose svc/name

# expose deployment in minishift

  c:\soft\minishift> oc expose deployment/hello-limit --port 80 --target-port 8080


# scale replicaset

  c:\soft\minishift> oc scale --replicas 3 deployment/hello-limit

# autoscale a deployment

  c:\soft\minishift> oc autoscale dc/hello --min 1 --max 10 --cpu-percent 80

# get all the configured clusters

  c:\soft\minishift> oc config get-clusters

# view the combined configuration

  c:\soft\minishift> oc config view

# use the different commands in oc config <sub commands>

    current-context Displays the current-context
    delete-cluster  Delete the specified cluster from the kubeconfig
    delete-context  Delete the specified context from the kubeconfig
    get-clusters    Display clusters defined in the kubeconfig
    get-contexts    Describe one or many contexts
    rename-context  Renames a context from the kubeconfig file.
    set             Sets an individual value in a kubeconfig file
    set-cluster     Sets a cluster entry in kubeconfig
    set-context     Sets a context entry in kubeconfig
    set-credentials Sets a user entry in kubeconfig
    unset           Unsets an individual value in a kubeconfig file
    use-context     Sets the current-context in a kubeconfig file
    view            Display merged kubeconfig settings or a specified kubeconfig file

# get pod spec in yaml format

  c:\soft\minishift> oc get pods -n default

  c:\soft\minishift> oc get pod docker-registry-1-bdwls -o yaml -n default

# get api resources

  c:\soft\minishift> oc api-resources

# get all the objects in the default namespace and store the yaml output

  c:\soft\minishift> oc get deploy,sts,svc,configmap,secret -n default -o yaml
                     --export > default.yaml

# bash script to export yaml to sub folders

  for n in $(kubectl get -o=name
             pvc,configmap,serviceaccount,secret,ingress,service,
             deployment,statefulset,hpa,job,cronjob )
  do
    mkdir -p $(dirname $n)
    kubectl get -o=yaml --export $n > $n.yaml
  done

# another bash script to export yaml to a single folder

  for n in $(kubectl get -o=name
             pvc,configmap,ingress,service,secret,deployment,
             statefulset,hpa,job,cronjob | grep -v 'secret/default-token')
  do
    kubectl get -o=yaml --export $n > $(dirname $n)_$(basename $n).yaml
  done
# stop the cluster

  c:\soft\minishift> minishift stop

# delete the cluster

  c:\soft\minishift> minishift delete
```

```
# delete the c:\users\atlantis\.minishift folder

-------------------------------------------------------------------------------------

# oc project commands

  # current project

    c:\soft\minishift> oc project

  # list projects

    c:\soft\minishift> oc get project

  # switch to a project named melvin

    c:\soft\minishift> oc project melvin

  # view the cluster config

    c:\soft\minishift> oc config view

  # evicting pods

    oc get pod -n studytonight | grep Evicted | awk '{print $1}' | xargs kubectl
    delete pod -n studytonight

  # get pods identified by a specific label

    kubectl get pods --all-namespaces -o=jsonpath="{..image}" -l app=nginx

  # get secret value from a secret object

       kubectl get secret <my_secret_name> -o 'go-template={{index .data
       "<key_name>"}}' | base64 -d

       ex:
        kubectl get secret my-secret -o 'go-template={{index .data "username"}}' |
        base64 -d
```

USING JSONPATH WITH OC/KUBECTL



Everything between the () brackets is part of the filter function
? - Indicates that this will be a filter function
@ - points to what key we need to filter

```
oc get pods -n myproject -o=jsonpath="{ ['item'][?(@.['child_item'])] {data-to-output}{data-to-output}{'\n'}{end}}"
```

JSONPATH FILTER	YOU GET...
`['item'][?(@.['child_item'])]`	All items with the specified child_item.
`['item'][?(@.['child_item'] == 'a_string')]`	All items with the specified child_item is equal to a_string.
`['item'][?(@.['child_item'] > 10)]`	All items where child_item is greater than 10.
`['item'].[?(@.['child_item_date'] > '2018-01-01')]`	All items where child_item_date is greater than 2018-01-01.
`['item'].[?(@.['child_item'] > 10)].['another_child_item']`	The another_child_item where an items child_item is greater than 10.
`['item'].[?(@.['child_item'])].['another_child_item']`	All another_child_items that have an item that the specified child_item.

KUBECTL JSONPATH Function	Description	Example	Result
Text	the plain text	`kind is {.kind}`	kind is List
@	the current object	`{@}`	the same as input
. or []	child operator	`{.kind}` or `{['kind']}`	List
..	recursive descent	`{..name}`	127.0.0.1 127.0.0.2 myself e2e
*	wildcard. Get all objects	`{.items[*].metadata.name}`	[127.0.0.1 127.0.0.2]
[start:end :step]	subscript operator	`{.users[0].name}`	myself
[,]	union operator	`{.items[*]['metadata.name', 'status.capacity']}`	127.0.0.1 127.0.0.2 map[cpu:4] map[cpu:8]
?()	filter	`{.users[?(@.name=="e2e")].user.password}`	secret
range, end	iterate list	`{range .items[*]}[{.metadata.name}, {.status.capacity}] {end}`	[127.0.0.1, map[cpu:4]] [127.0.0.2, map[cpu:8]]
""	quote interpreted string	`{range .items[*]}{.metadata.name}{"\t"}{end}`	127.0.0.1 127.0.0.2

```
# using jsonpath to get pod names from all namespaces

c:\soft\minishift> oc get pods -A -o=jsonpath="{range.items[*]}{.metadata.namespace},{.metadata.name}{'\n'}{end}"

# using jsonpath to get pod names from a specified namespaces

c:\soft\minishift> oc get pods -n myproject -o=jsonpath="{range .items[*]}{.metadata.namespace},{.metadata.name}{'\n'}{end}"


# get pod name

c:\soft\minishift> oc get pods -o jsonpath="{range.items[?(@status.phase)]}{.metadata.name}{'\n'}{end}"

oc get pods -o jsonpath="{range.items[?(@status.phase)]}{.metadata.name}::{.status.phase}{'\n'}{end}"

# get pods in a ns which are running

c:\soft\minishift> oc get pods --namespace myproject -o
jsonpath="{range.items[?(@.status.phase=='Running')]}{.metadata.name}::{.status.phase}{'\n'}{end}"

# get pods in a ns which have failed

c:\soft\minishift> oc get pods --namespace myproject -o
jsonpath="{range.items[?(@.status.phase=='Failed')]}{.metadata.name}::{.status.phase}{'\n'}{end}"
```

SOURCE TO IMAGE TO GIT PULL, BUILD, CONTAINERIZE, DEPLOY A SPRING BOOT APP TO MINISHIFT/ OPENSHIFT PLATFORM

project in the laptop: c:\soft\minishift-examples\demo
project workspace: c:\soft\minishift-examples\demo-ws

git repo for building and deploying a spring boot app using the openshift s2i

https:

https://github.com/messages-one/minishift-examples.git

```
echo "# minishift-examples" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/messages-one/minishift-examples.git
git push -u origin master

git remote add origin https://github.com/messages-one/minishift-examples.git
git branch -M main
git push -u origin master
```

ssh:

git@github.com:messages-one/minishift-examples.git

```
echo "# minishift-examples" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:messages-one/minishift-examples.git
git push -u origin master
git remote add origin git@github.com:messages-one/minishift-examples.git
git branch -M main
git push -u origin master
```

# create a project

```
  c:\soft\minishift> oc new-project minishift-demo-project
```

# get the oc client and extract to c:\soft\minishift folder

https://access.redhat.com/downloads/content/290/ver=4.10/rhel---8/4.10.14/x86_64/product-software

# get docker client from

https://download.docker.com/win/static/stable/x86_64/

# copy the docker.exe in c:\soft\minishift folder

# get the docker env details from minishift

```
  c:\soft\minishift> minishift docker-env
```

# execute the output of the above command one by one

# login to the registry.redhat.io

https://access.redhat.com/RegistryAuthentication#creating-registry-service-accounts-6

redhat developer account:

user name: messages.one@outlook.com
password: discovery

# creating registry service account
https://access.redhat.com/RegistryAuthentication#creating-registry- serviceaccounts-6

# login to the registry.redhat.io from docker

```
  c:\soft\minishift> docker login https://registry.redhat.io

    user name: messages.one@outlook.com
    password: aprilJones@67
```

# pull the jdk11 s2i image: check this page:

https://docs.openshift.com/online/pro/using_images/s2i_images/java.html

```
  c:\soft\minishift> docker pull registry.redhat.io/ubi8/openjdk-11
```

# pull the latest openjdk-17 s2i image from registry.access.redhat.com
use the same credentials as above.

list of downloadable container images for minishift/openshift:

https://catalog.redhat.com/software/containers/explore

```
  c:\soft\minishift> docker pull registry.access.redhat.com/ubi8/openjdk-17:1.12-
                     1.1651233093


# create a new app and begin the build process with jdk-11

    c:\soft\minishift> oc new-app registry.redhat.io/ubi8/openjdk-
    11~https://github.com/messages-one/minishift-examples.git --name=minishift-demo

# to use the jdk-17 s2i

    c:\soft\minishift> oc new-app registry.access.redhat.com/ubi8/openjdk-
    17~https://github.com/messages-one/minishift-examples.git --name=minishift-demo

# check the compiler logs if a build fails

    c:\soft\minishift> oc logs -f bc/minishift-demo

# restart the build

    c:\soft\minishift> oc start-build minishift-demo

# when the build is successful we get a docker image in the logs

    172.30.1.1:5000/demo-minishift-s2i/minishift-demo:latest

# check that the image exists

    c:\soft\minishift> docker images

REPOSITORY                                          TAG

172.30.1.1:5000/demo-minishift-s2i/minishift-demo   latest
registry.access.redhat.com/ubi8/openjdk-17          1.12-1.1651233093        registry.redhat.io/ubi8/openjdk-11
latest

# get pods

    c:\soft\minishift> oc get pods

# delete multiple pods

    c:\soft\minishift> oc delete pods minishift-demo-1-build minishift-demo-2-build
                       minishift-demo-3-build

------------------------------------------------------------------------------------

# enable admin addon. this plugin helps to login to Minishift as cluster admin.

    c:\soft\minishift> minishift addons apply admin-user

#  grant role cluster-admin to user admin.

    c:\soft\minishift> oc login -u system:admin
    c:\soft\minishift> oc adm policy add-cluster-role-to-user cluster-admin admin
    c:\soft\minishift> oc login -u admin -p admin

#  The image used for building runnable Java apps (openjdk18-openshift) is not
#  available by default on Minishift.
#  We can import it manually from RedHat registry using oc import-image command or
#  just enable and apply plugin xpaas.

    c:\soft\minishift> minishift addons apply xpaas

# login to the minishift console as admin

       C:\soft\minishift> minishift console

       user name: admin  password: admin


# select the project demo-minishift-s2i
# go the application menu on the left

  Select the services -> minishift-demo -> create a route -> copy the url

   Ex: http://minishift-demo-minishift-demo-project.192.168.99.101.nip.io/hello


# your application is accessible from this url

------------------------------------------------------------------------------------
```

SIMPLE EXAMPLE PROJECT

```
# create a new project

  c:\soft\minishift> oc new-project melvin

    Now using project "melvin" on server "https://192.168.99.101:8443".

    You can add applications to this project with the 'new-app' command.
    For example, try:

        oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git

        to build a new example application in Ruby.

c:\soft\minishift> oc new-app openshift/hello-openshift
```

```
  --> Found Docker image 7af3297 (4 years old) from Docker Hub for
      "openshift/hello-openshift"

      * An image stream tag will be created as "hello-openshift:latest" that will
        track this image
      * This image will be deployed in deployment config "hello-openshift"
      * Ports 8080/tcp, 8888/tcp will be load balanced by service "hello-openshift"
      * Other containers can access this service through the hostname "hello-
        openshift"

  --> Creating resources ...
      imagestream.image.openshift.io "hello-openshift" created
      deploymentconfig.apps.openshift.io "hello-openshift" created
      service "hello-openshift" created
  --> Success
      Application is not exposed. You can expose services to the outside world by
      executing one or more of the commands below:
       'oc expose svc/hello-openshift'
      Run 'oc status' to view your app.

# create an ingress object ingress.yaml

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hello-openshift
spec:
  rules:
  - host: hello-openshift.yourcluster.example.com   # change the host name. yourcluster.example.com is the cluster name given at
the time of creation
    http:
      paths:
      - backend:
          # Forward to a Service called 'hello-openshift'
          service:
            name: hello-openshift
            port:
              number: 8080
        path: /
        pathType: Exact
# apply the ingress object. it also creates a route which is a wildcard domain

  c:\soft\minishift> oc apply -f ingress.yaml

# get the ingress object

  c:\soft\minishift> oc get ingress

# get the route

  c:\soft\minishift> oc get route

# access the app

  c:\soft\minishift> curl hello-openshift.apps.ocp1.example.com

# delete the route

  c:\soft\minishift> oc delete route hello-openshift-5cbw4

# delete the ingress object in this project

  c:\soft\minishift> oc delete ingress --all

-----------------------------------------------------------------------------------

c:\soft\minishift> minishift start

       The server is accessible via web console at:
       https://192.168.99.101:8443/console

       You are logged in as:
              User:     developer
              Password: <any value>

       To login as administrator:
              oc login -u system:admin

-----------------------------------------------------------------------------------
```

WORKING WITH PV/PVC

```
# ssh into the docker container hosting the minishift cluster

c:\soft\minishift>  minishift ssh

[docker@minishift ~]$ sudo -i

[root@minishift ~]#

[root@minishift ~]# mkdir -p /mnt/sda1/var/lib/minishift/openshift.local.volumes/pv

[root@minishift ~]# mkdir
                    /mnt/sda1/var/lib/minishift/openshift.local.volumes/pv/registry

[root@minishift ~]# chmod 777 -R
                    /mnt/sda1/var/lib/minishift/openshift.local.volumes/pv
```

```
[root@minishift ~]#   exit

[docker@minishift ~]$ exit

c:\soft\minishift>
```

# create a pv spec in c:\soft\minishift\minishift-demo-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: minishift-demo-pv
 labels:
  minishift-demo-storage: "1"
spec:
 storageClassName: local-storage
 capacity:
  storage: 1Gi
 accessModes:
  - ReadWriteOnce
 storageClassName: local-storage
 hostPath:
  path: /mnt/sda1/var/lib/minishift/openshift.local.volumes/pv/registry
```

```
c:\soft\minishift> oc create -f minishift-demo-pv.yaml
```

# create a pvc spec in c:\soft\minishift\minishift-demo-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: minishift-demo-pvc
  namespace: minishift-demo-project
  resourceVersion: '259804'
spec:
  volumeName: minishift-demo-pv
  storageClassName: local-storage
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      minishift-demo-storage: "1"
```

```
c:\soft\minishift> oc create -f minishift-demo-pvc
```

# use the pvc in a pod c:\soft\minishift\pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: minishift-demo
spec:
  volumes:
    - name: minishift-storage
      persistentVolumeClaim:
        claimName: minishift-demo-pvc
  containers:
    - name: minishift-demo
      image: 172.30.1.1:5000/minishift-demo-project/minishift-demo
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: minishift-storage
```

```
c:\soft\minishift> oc create -f pod.yaml
```

USING PV/PVC FOR GCP FILESTORE NFS SERVICE WITH GKE

https://cloud.google.com/filestore/docs/accessing-fileshares

GKE VERSION UPGRADE CONTROL / DATA PLANE

https://www.youtube.com/watch?v=ajbC1yTW2x0

Gke version upgrade happens in two steps

1. upgrade the control plane

    For zero downtime always create a regional gke cluster gke does a rolling update

the old version node is drained and cordoned to ensure no pods are running.
The node is deleted and a new node is created with the new version

these steps are repeated for all the nodes until the control plane is updated

this process can be automated by enabling the automatic node upgrades.
If this option is not selected gke will still alert when a new version is available

Always ensure that you have a replic for your pods as standalone pods won't be scheduled

2. using multiple node pools to update the cluster

Here we create a fresh node pool instead of updating the old node pool and then migrate workload to the new node pool one node at a time

This is a manual process

Assume that the gke cluster has 3 nodes in a node pool named default-pool

```
$ kubectl get nodes

$ gcloud container node-pools create pool-two

$ kubectl get nodes
```

At this point in time the pods are still running on the old node pool

Now lets move the workload one node at a time. Cordon the node so that no new pods are scheduled on that node

```
$ kubectl cordon <node-name>

$ kubectl drain <node-name> --force
```

ensure that the pods are scheduled on the new node
Repeat the steps for all the nodes

```
// finally remove the old node pool

$ gcloud container node-pools delete default-pool
```

If for some reason the upgrade fails then

Uncordon the old node

Mark node <node-name> as schedulable.

```
$ kubectl uncordon <node-name>
```

Kubectl command reference:

https://jamesdefabia.github.io/docs/user-guide/kubectl/kubectl/

```
kubectl get secret my-secret -o 'go-template={{index .data "username"}}' | base64 -d
```

```
kubectl rollout status deployment/<deployment-name>
```

This will run in foreground, it waits and displays the status, and exits when rollout is complete on success or failure. If you're writing a shell script, then check the return code right after the command, something like this.

```
kubectl rollout status deployment/<deployment-name>

if [[ "$?" -ne 0 ]] then
    echo "deployment failed!"
    exit 1
fi
```

To even further automate your script:

```
deployment_name=$(kubectl get deployment -n <your namespace> | awk '!/NAME/{print $1}')
kubectl rollout status deployment/"${deployment_name}" -n <your namespace>
if [[ "$?" -ne 0 ]] then
    echo "deployment failed!"
    #exit 1
else
    echo "deployment succeeded"
fi
```

ISTIO MINISHIFT ADDON AND DEPLOY A SAMPLE APP

https://github.com/VeerMuchandi/istio-on-openshift/blob/master/DeployingIstioWithMinishift.md

```
# if the profile exists due to a failed installation then delete the profile

      c:\soft\minishift> minishift delete profile servicemesh

      $ rm -rf ~/.minishift/profiles/servicemesh


# create a minishift profile

      c:\soft\minishift> minishift profile set servicemesh

      c:\soft\minishift> minishift config set memory 8GB

      c:\soft\minishift> minishift config set cpus 4

      c:\soft\minishift> minishift config set image-caching true

      c:\soft\minishift> minishift config set openshift-version v3.10.0

      c:\soft\minishift> minishift addon enable admin-user

      c:\soft\minishift> minishift addon enable anyuid

# start minishift

      c:\soft\minishift> minishift start

      c:\soft\minishift> oc login -u system:admin

      c:\soft\minishift> git clone https://github.com/minishift/minishift-addons

      c:\soft\minishift> oc new-project myproject

      c:\soft\minishift> oc project myproject

      c:\soft\minishift> minishift addon install C:\soft\minishift\minishift-
                          addons\add-ons\istio

      c:\soft\minishift> minishift addon enable istio

      c:\soft\minishift> minishift addon apply istio

      c:\soft\minishift> oc get pods -w -n istio-system --as system:admin

# verify istio installation

      c:\soft\minishift> oc project istio-system

      c:\soft\minishift> oc get sa

NAME                                     SECRETS   AGE
builder                                  2         7h
default                                  2         7h
deployer                                 2         7h
elasticsearch                            2         7h
grafana                                  2         7h
istio-citadel-service-account            2         7h
istio-egressgateway-service-account      2         7h
istio-galley-service-account             2         7h
istio-ingressgateway-service-account     2         7h
istio-mixer-service-account              2         7h
istio-pilot-service-account              2         7h
istio-sidecar-injector-service-account   2         7h
jaeger                                   2         7h
kiali-service-account                    2         7h
openshift-ansible                        2         7h
prometheus                               2         7h

 c:\soft\minishift> oc get pods

NAME                              READY     STATUS    RESTARTS   AGE
istio-ca-2617747623-0ch0b         1/1       Running   0          15s
istio-egress-2389443630-l8706     1/1       Running   0          16s
istio-ingress-355016184-nd4gp     1/1       Running   0          16s
istio-mixer-3229407178-v3q3m      2/2       Running   0          19s
istio-pilot-589912157-7x7p7       1/1       Running   0          17s

c:\soft\minishift> oc get crd

NAME                                                          AGE
adapters.config.istio.io                                      7h
apikeys.config.istio.io                                       7h
attributemanifests.config.istio.io                            7h
authorizations.config.istio.io                                7h
bypasses.config.istio.io                                      7h
checknothings.config.istio.io                                 7h
circonuses.config.istio.io                                    7h
deniers.config.istio.io                                       7h
destinationrules.networking.istio.io                          7h
edges.config.istio.io                                         7h
envoyfilters.networking.istio.io                              7h
```

```
fluentds.config.istio.io                                      7h
gateways.networking.istio.io                                  7h
handlers.config.istio.io                                      7h
httpapispecbindings.config.istio.io                           7h
httpapispecs.config.istio.io                                  7h
installations.istio.openshift.com                             7h
instances.config.istio.io                                     7h
kubernetesenvs.config.istio.io                                7h
kuberneteses.config.istio.io                                  7h
listcheckers.config.istio.io                                  7h
listentries.config.istio.io                                   7h
logentries.config.istio.io                                    7h
memquotas.config.istio.io                                     7h
meshpolicies.authentication.istio.io                          7h
metrics.config.istio.io                                       7h
noops.config.istio.io                                         7h
opas.config.istio.io                                          7h
openshiftwebconsoleconfigs.webconsole.operator.openshift.io   7h
policies.authentication.istio.io                              7h
prometheuses.config.istio.io                                  7h
quotas.config.istio.io                                        7h
quotaspecbindings.config.istio.io                             7h
quotaspecs.config.istio.io                                    7h
rbacconfigs.rbac.istio.io                                     7h
rbacs.config.istio.io                                         7h
redisquotas.config.istio.io                                   7h
reportnothings.config.istio.io                                7h
rules.config.istio.io                                         7h
servicecontrolreports.config.istio.io                         7h
servicecontrols.config.istio.io                               7h
serviceentries.networking.istio.io                            7h
servicerolebindings.rbac.istio.io                             7h
serviceroles.rbac.istio.io                                    7h
signalfxs.config.istio.io                                     7h
solarwindses.config.istio.io                                  7h
stackdrivers.config.istio.io                                  7h
statsds.config.istio.io                                       7h
stdios.config.istio.io                                        7h
templates.config.istio.io                                     7h
tracespans.config.istio.io                                    7h
virtualservices.networking.istio.io                           7h

c:\soft\minishift> oc get attributemanifests

NAME         AGE
istioproxy   7h
kubernetes   7h


c:\soft\minishift> oc get metrics

NAME              AGE
requestcount      7h
requestduration   7h
requestsize       7h
responsesize      7h
tcpbytereceived   7h
tcpbytesent       7h


c:\soft\minishift> oc get prometheuses

NAME      AGE
handler   7h


c:\soft\minishift> oc get rules

NAME                     AGE
kubeattrgenrulerule      7h
promhttp                 7h
promtcp                  7h
stdio                    7h
stdiotcp                 7h
tcpkubeattrgenrulerule   7h

c:\soft\minishift> oc get logentries

NAME           AGE
accesslog      7h
tcpaccesslog   7h


c:\soft\minishift> oc get stdios

NAME      AGE
handler   7h

c:\soft\minishift> oc get deployments

NAME                        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
grafana                     1         1         1            1           7h
istio-citadel               1         1         1            1           7h
istio-egressgateway         1         1         1            1           7h
istio-galley                1         1         1            1           7h
istio-ingressgateway        1         1         1            1           7h
istio-pilot                 1         1         1            1           7h
istio-policy                1         1         1            1           7h
istio-sidecar-injector      1         1         1            1           7h
istio-statsd-prom-bridge    1         1         1            1           7h
istio-telemetry             1         1         1            1           7h
jaeger-collector            1         1         1            1           7h
```

```
jaeger-query               1         1         1            1           7h
kiali                      1         1         1            1           7h
prometheus                 1         1         1            1           7h
Note the services running here.
```

```
c:\soft\minishift> oc get svc

NAME                       TYPE           CLUSTER-IP       EXTERNAL-IP                   PORT(S)
AGE
elasticsearch              ClusterIP      172.30.221.120   <none>                        9200/TCP
7h
elasticsearch-cluster      ClusterIP      172.30.146.4     <none>                        9300/TCP
7h
grafana                    ClusterIP      172.30.98.124    <none>                        3000/TCP
7h
istio-citadel              ClusterIP      172.30.7.128     <none>                        8060/TCP,9093/TCP
7h
istio-egressgateway        ClusterIP      172.30.42.76     <none>                        80/TCP,443/TCP
7h
istio-galley               ClusterIP      172.30.40.24     <none>                        443/TCP,9093/TCP
7h
istio-ingressgateway       LoadBalancer   172.30.57.84     172.29.203.39,172.29.203.39
80:31380/TCP,443:31390/TCP,31400:31400/TCP,15011:30316/TCP,8060:32290/TCP,853:31213/TCP,15030:30194/TCP,15031:31527/TCP   7h
istio-pilot                ClusterIP      172.30.7.142     <none>                        15010/TCP,15011/TCP,8080/TCP,9093/TCP
7h
istio-policy               ClusterIP      172.30.57.36     <none>                        9091/TCP,15004/TCP,9093/TCP
7h
istio-sidecar-injector     ClusterIP      172.30.76.218    <none>                        443/TCP
7h
istio-statsd-prom-bridge   ClusterIP      172.30.56.73     <none>                        9102/TCP,9125/UDP
7h
istio-telemetry            ClusterIP      172.30.16.103    <none>                        9091/TCP,15004/TCP,9093/TCP,42422/TCP
7h
jaeger-collector           ClusterIP      172.30.21.135    <none>                        14267/TCP,14268/TCP,9411/TCP
7h
jaeger-query               LoadBalancer   172.30.102.230   172.29.59.125,172.29.59.125   80:30224/TCP
7h
kiali                      ClusterIP      172.30.178.25    <none>                        20001/TCP
7h
prometheus                 ClusterIP      172.30.63.80     <none>                        9090/TCP
7h
tracing                    LoadBalancer   172.30.226.196   172.29.56.4,172.29.56.4       80:31411/TCP
7h
zipkin                     ClusterIP      172.30.218.223   <none>                        9411/TCP
```

```
c:\soft\minishift> oc get route

NAME                   HOST/PORT                                                 PATH      SERVICES               PORT
TERMINATION   WILDCARD
grafana                grafana-istio-system.192.168.64.72.nip.io                           grafana                http
None
istio-ingressgateway   istio-ingressgateway-istio-system.192.168.64.72.nip.io              istio-ingressgateway   http2
None
jaeger-query           jaeger-query-istio-system.192.168.64.72.nip.io                      jaeger-query           jaeger-query
edge          None
kiali                  kiali-istio-system.192.168.64.72.nip.io                             kiali                  http-kiali
reencrypt     None
prometheus             prometheus-istio-system.192.168.64.72.nip.io                        prometheus             http-prometheus
None
tracing                tracing-istio-system.192.168.64.72.nip.io                           tracing                tracing
edge          None
```

# deploy the sample bookinfo app that comes with istio

https://istio.io/latest/docs/examples/bookinfo/

# Add a namespace label to instruct Istio to automatically inject Envoy sidecar proxies when you deploy your application in the namespace

```
c:\soft\minishift> oc label namespace myproject istio-injection=enabled
```

# these commands are meant for minishift/openshift clusters

```
$ oc adm policy add-scc-to-group anyuid system:serviceaccounts:istio-system

$ oc -n istio-system expose svc/istio-ingressgateway --port=http2
```

# debugging istio elastic search pod failure with creash back off status

# look into the logs of the pod in minishift console.
# It displays that the container failed because the memory mapped file size is too less

Go to the windows console

```
c:\soft\minishift> minishift ssh

    $ sudo  sysctl -w vm.max_map_count=262144
```

Now check the pod status. Elastic search should come up successful and so will jaeger containers.

# deploy the sample app. Check the below link

https://istio.io/latest/docs/setup/getting-started/#bookinfo

```
c:\soft\minishift> oc apply -f C:\soft\minishift\istio-1.13.4/
                   samples/bookinfo/platform/kube/bookinfo.yaml
```

XXX