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

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

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 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
```

```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
```