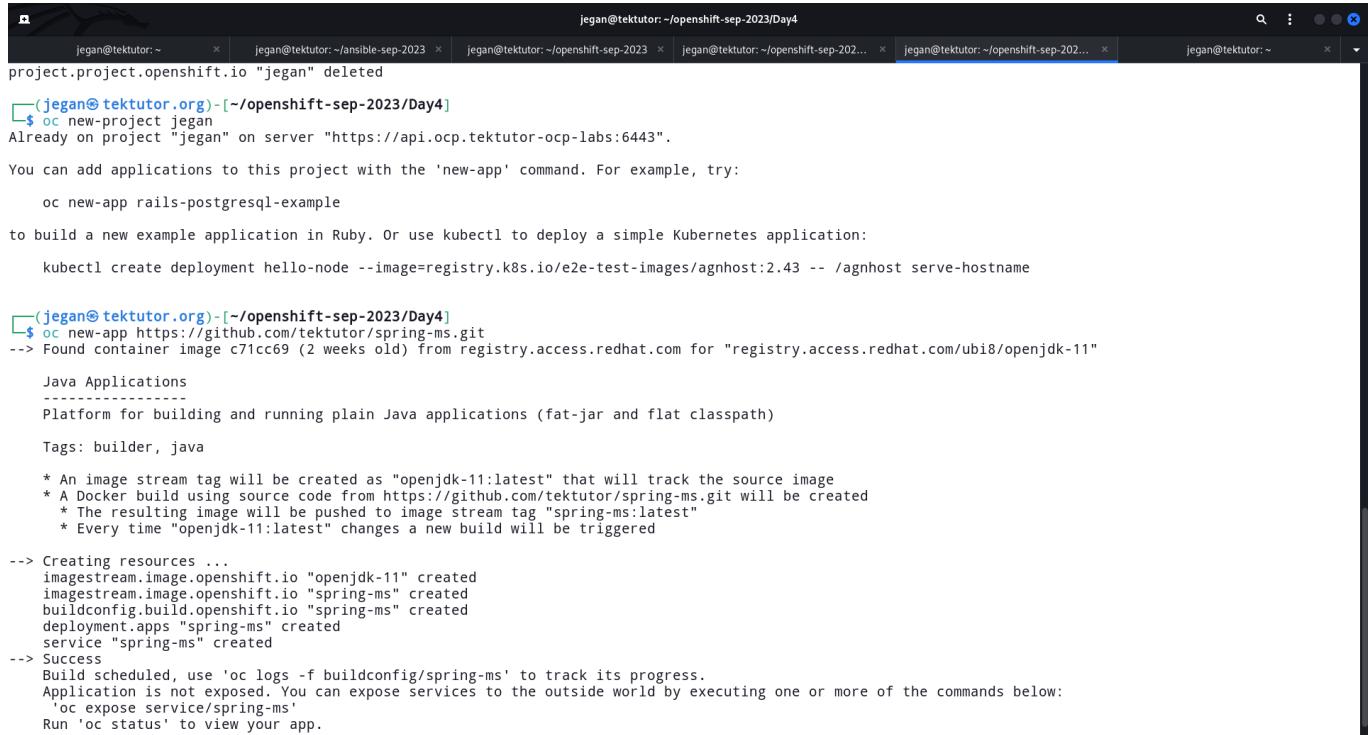


Day 4

Lab - Deploying an application using GitHub Repo source code using Docker strategy

```
oc project jegan  
oc new-app https://github.com/tektutor/spring-ms.git  
oc expose service/spring-ms
```

Expected output



The screenshot shows a terminal window with multiple tabs open, all belonging to the user 'jegan' on a host named 'tektutor'. The active tab is titled '(jegan@tektutor.org) - [~/openshift-sep-2023/Day4]'. The user has run several commands to create a new project and application:

```
jegan@tektutor:~ x jegan@tektutor:~/ansible-sep-2023 x jegan@tektutor:~/openshift-sep-2023 x jegan@tektutor:~/openshift-sep-202... x jegan@tektutor:~/openshift-sep-202... x jegan@tektutor:~  
project.project.openshift.io "jegan" deleted  
[jegan@tektutor.org] - [~/openshift-sep-2023/Day4]  
$ oc new-project jegan  
Already on project "jegan" on server "https://api.ocp.tektutor-ocp-labs:6443".  
You can add applications to this project with the 'new-app' command. For example, try:  
oc new-app rails-postgresql-example  
to build a new example application in Ruby. Or use kubectl to deploy a simple Kubernetes application:  
kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-images/agnhost:2.43 -- /agnhost serve-hostname  
  
[jegan@tektutor.org] - [~/openshift-sep-2023/Day4]  
$ oc new-app https://github.com/tektutor/spring-ms.git  
--> Found container image c71cc69 (2 weeks old) from registry.access.redhat.com for "registry.access.redhat.com/ubi8/openjdk-11"  
Java Applications  
-----  
Platform for building and running plain Java applications (fat-jar and flat classpath)  
Tags: builder, java  
* An image stream tag will be created as "openjdk-11:latest" that will track the source image  
* A Docker build using source code from https://github.com/tektutor/spring-ms.git will be created  
* The resulting image will be pushed to image stream tag "spring-ms:latest"  
* Every time "openjdk-11:latest" changes a new build will be triggered  
--> Creating resources ...  
imagestream.image.openshift.io "openjdk-11" created  
imagestream.image.openshift.io "spring-ms" created  
buildconfig.build.openshift.io "spring-ms" created  
deployment.apps "spring-ms" created  
service "spring-ms" created  
--> Success  
Build scheduled, use 'oc logs -f buildconfig/spring-ms' to track its progress.  
Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:  
'oc expose service/spring-ms'  
Run 'oc status' to view your app.
```

To check the build logs in command line, you can try this

```
oc logs -f build/spring-ms-1
```

Expected output

```
jegan@tekututor: ~/openshift-sep-2023/Day4
jegan@tekututor:~ x jegan@tekututor:~/ansible-sep-2023 x jegan@tekututor:~/openshift-sep-2023 x jegan@tekututor:~/openshift-sep-202... x jegan@tekututor:~/openshift-sep-202... x jegan@tekututor:~ x

$ oc get builds
NAME        TYPE      FROM          STATUS     STARTED      DURATION
spring-ms-1   Docker   Git@82552fb  Running    56 seconds ago

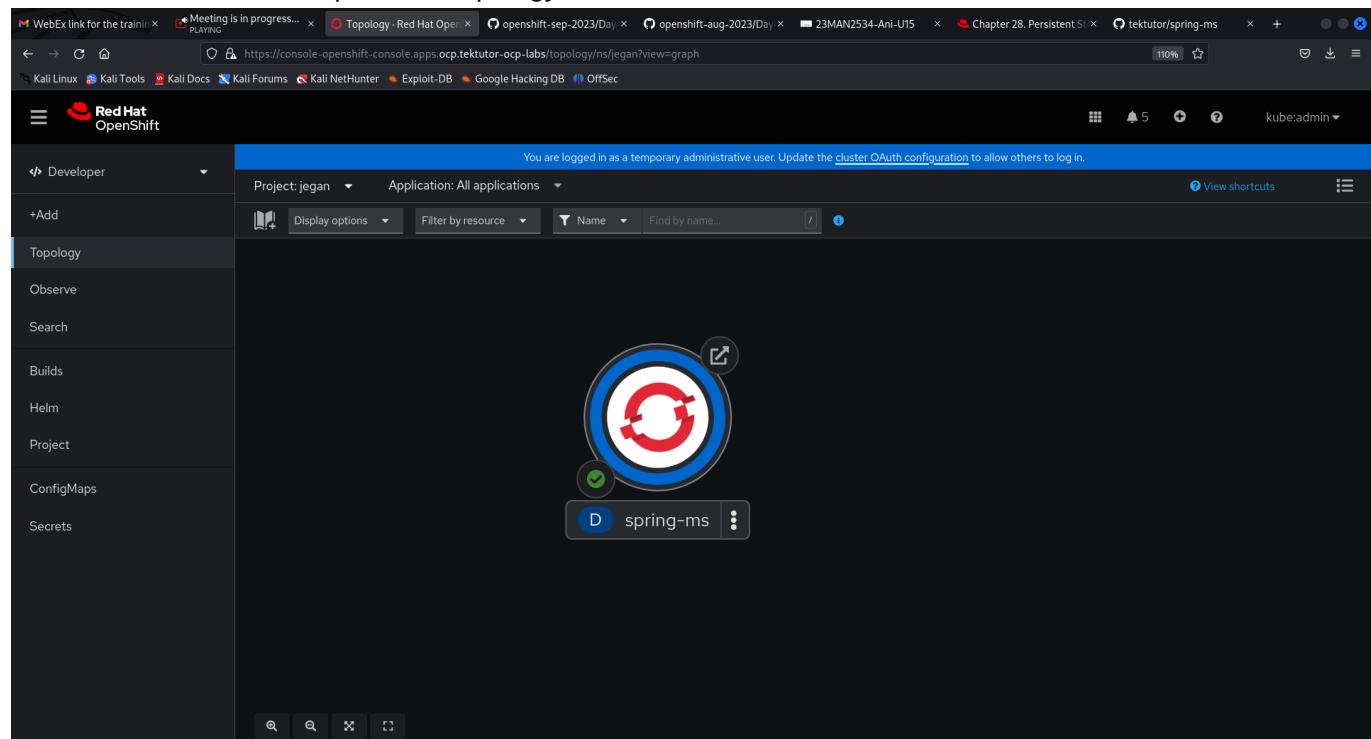
(jegan@tekututor.org)-[~/openshift-sep-2023/Day4]
$ oc logs -f build/spring-ms-1
Cloning "https://github.com/tekututor/spring-ms.git" ...
Commit: 82552fb8a8eb3a7cc7e8165b8878dc5e47e50db3 (Renamed deploy.yml to deploy.yaml)
Author: Jeganathan Swaminathan <mail2jegan@gmail.com>
Date: Wed Feb 15 15:11:17 2023 +0530
Replaced Dockerfile FROM image registry.access.redhat.com/ubi8/openjdk-11
time="2023-09-14T06:16:55Z" level=info msg="Not using native diff for overlay, this may cause degraded performance for building images: kernel has CONFIG_OVERLAY_FS_REDIRECT_DIR enabled"
I0914 06:16:55.838227       defaults.go:112] Defaulting to storage driver "overlay" with options [mountopt=metacopy-on].
Caching blobs under "/var/cache/blobs".

Pulling image docker.io/maven:3.6.3-jdk-11 ...
Trying to pull docker.io/library/maven:3.6.3-jdk-11...
Getting image source signatures
Copying blob sha256:6c215442f70bd949a6f2e8092549943905e2d4f9c87a4f532d7740ae8647d33a
Copying blob sha256:5d6f1e8117dbb1c6a57603cb4f321a861a08105a81bcc6b01b0ec2b78c8523a5
Copying blob sha256:234b20d0479d7f16d7ee8d04e4ffdac57d7d14313faf59d332f18b2e9418743
Copying blob sha256:48c2faf66abc3dce9f54d6722ff592fce6dd4fb58a0d0b72282936c6598a3b3
Copying blob sha256:d7eb6c022a4e6128219b32a8e07c8c22c89624ff440ebac1506121794bc15cc
Copying blob sha256:004feef8d7f37f5f5e2a1a649fa7edd7f713d1300532fd0909bb39cd48437d7
Copying blob sha256:355e8215390faee903502a9fddfc65cd823f1606f053376ba2575adce66974a1
Copying blob sha256:c5eb43522f68d7e2347e19ad70dadcf1594d25b792ede0464c2936ff902c4c6
Copying blob sha256:4fee0489a65b64056f81358639bfe85fd87776630830fd02ce8c15e34928bf9c
Copying blob sha256:413646e6fa5d7bcd9722d3e400fc080a77deb505baed79afa5fedea23583af25
Copying config sha256:e23b595c92ada5c9f20a7d547ed980a445f644eb1cbde7cfb27478fa38c4691
Writing manifest to image destination
Storing signatures

(jegan@tekututor.org)-[~/openshift-sep-2023/Day4]
$ oc expose service/spring-ms
route.route.openshift.io/spring-ms exposed

(jegan@tekututor.org)-[~/openshift-sep-2023/Day4]
$
```

You can check the Developer view Topology



The screenshot shows the Red Hat OpenShift developer interface. On the left, a sidebar lists various navigation options: Developer, +Add, Topology (which is selected), Observe, Search, Builds, Helm, Project, ConfigMaps, and Secrets. The main content area is titled 'Topology' and shows a single application named 'spring-ms'. The application icon is a circle with a red arrow and a blue border. Below the icon, there's a green checkmark and a blue button labeled 'D'. At the bottom of the screen, there are search and filter tools.

Project: jegan Application: All applications

Display options Filter by resource Name Find by name... View shortcuts

spring-ms

Hello Microservice !

Lab - Starting a build from command-line using buildconfig

```
oc get buildconfigs  
oc get buildconfig  
oc get bc
```

Start a build from buildconfig

```
oc start-build bc/spring-ms
oc logs -f bc/spring-ms
```

Expected output

The screenshot shows a terminal window with multiple tabs, all titled 'jegan@tektutor: ~'. The terminal content is as follows:

```
(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc get buildconfigs
NAME      TYPE    FROM      LATEST
spring-ms  Docker  Git       1

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc get builds
NAME      TYPE    FROM      STATUS     STARTED      DURATION
spring-ms-1 Docker  Git@82552fb  Complete   19 minutes ago  1m58s

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc start-build buildconfig/spring-ms
build.build.openshift.io/spring-ms-2 started

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc get builds
NAME      TYPE    FROM      STATUS     STARTED      DURATION
spring-ms-1 Docker  Git@82552fb  Complete   19 minutes ago  1m58s
spring-ms-2 Docker  Git       Running    4 seconds ago

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc logs -f bc/spring-ms
Cloning "https://github.com/tektutor/spring-ms.git" ...
  Commit: 82552fb8a8eb3a7cc7e8165b8878dc5e47e50db3 (Renamed deploy.yml to deploy.yaml)
  Author: Jeganathan Swaminathan <mail12jegan@gmail.com>
  Date:  Wed Feb 15 15:11:17 2023 +0530
Replaced Dockerfile FROM image registry.access.redhat.com/ubi8/openjdk-11
time="2023-09-14T06:36:46Z" level=info msg="Not using native diff for overlay, this may cause degraded performance for building images: kernel has CONFIG_OVERLAY_FS_REDIRECT_DIR enabled"
I0914 06:36:46.326305      1 defaults.go:112] Defaulting to storage driver "overlay" with options [mountopt=metacopy=on].
Caching blobs under "/var/cache/blobs".

Pulling image docker.io/maven:3.6.3-jdk-11 ...
```

Create a route for spring-ms application deployment

```
oc get svc
oc expose svc/spring-ms
```

Lab - Deploying our custom application into OpenShift using source strategy

```
oc delete project/jegan
oc new-project jegan
oc new-app registry.access.redhat.com/ubi8/openjdk-
11~https://github.com/tektutor/spring-ms.git --strategy=source
```

Expected output

```
jegan@tektutor: ~/openshift-sep-2023/Day4
jegan@tektutor: ~/ansible-sep-2023 x jegan@tektutor: ~/openshift-sep-2023 x jegan@tektutor: ~/openshift-sep-202... x jegan@tektutor: ~/openshift-sep-202... x jegan@tektutor: ~
$ oc new-project jegan
Already on project "jegan" on server "https://api.ocp.tektutor-ocp-labs:6443".

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

  oc new-app rails-postgresql-example

to build a new example application in Ruby. Or use kubectl to deploy a simple Kubernetes application:

  kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-images/agnhost:2.43 -- /agnhost serve-hostname

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc new-app registry.access.redhat.com/ubi8/openjdk-11-https://github.com/tektutor/spring-ms.git --strategy=source
--> Found container image c71cc69 (2 weeks old) from registry.access.redhat.com for "registry.access.redhat.com/ubi8/openjdk-11"

Java Applications
-----
Platform for building and running plain Java applications (fat-jar and flat classpath)

Tags: builder, java

* An image stream tag will be created as "openjdk-11:latest" that will track the source image
* A source build using source code from https://github.com/tektutor/spring-ms.git will be created
* The resulting image will be pushed to image stream tag "spring-ms:latest"
* Every time "openjdk-11:latest" changes a new build will be triggered

--> Creating resources ...
imagestream.image.openshift.io "openjdk-11" created
imagestream.image.openshift.io "spring-ms" created
buildconfig.build.openshift.io "spring-ms" created
deployment.apps "spring-ms" created
service "spring-ms" created
--> Success
Build scheduled, use 'oc logs -f buildconfig/spring-ms' to track its progress.
Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
  'oc expose service/spring-ms'
Run 'oc status' to view your app.

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
$
```

Checking the build log

```
oc logs -f bc/spring-ms
```

Expected output

```
jegan@tekututor: ~/openshift-sep-2023/Day4
[jegan@tekututor: ~] jegan@tekututor: ~/ansible-sep-2023 [jegan@tekututor: ~/openshift-sep-2023 [jegan@tekututor: ~/openshift-sep-202... [jegan@tekututor: ~/openshift-sep-202... [jegan@tekututor: ~]
Platform for building and running plain Java applications (fat-jar and flat classpath)

Tags: builder, java

* An image stream tag will be created as "openjdk-11:latest" that will track the source image
* A source build using source code from https://github.com/tektutor/spring-ms.git will be created
  * The resulting image will be pushed to image stream tag "spring-ms:latest"
  * Every time "openjdk-11:latest" changes a new build will be triggered

--> Creating resources ...
imagestream.image.openshift.io "openjdk-11" created
imagestream.image.openshift.io "spring-ms" created
buildconfig.build.openshift.io "spring-ms" created
deployment.apps "spring-ms" created
service "spring-ms" created
--> Success
Build scheduled, use 'oc logs -f buildconfig/spring-ms' to track its progress.
Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
  'oc expose service/spring-ms'
Run 'oc status' to view your app.

[jegan@tekututor.org]-[~/openshift-sep-2023/Day4]
$ oc logs -f bc/spring-ms
Cloning "https://github.com/tektutor/spring-ms.git" ...
  Commit: 82552fb8a8eb3a7cc7e8165b8878dc5e47e50db3 (Renamed deploy.yml to deploy.yaml)
  Author: Jeganathan Swaminathan <mail2jegan@gmail.com>
  Date:  Wed Feb 15 15:11:17 2023 +0530
time="2023-09-14T07:15:18Z" level=info msg="Not using native diff for overlay, this may cause degraded performance for building images: kernel has CONFIG_OVERLAY_FS_REDIRECT_DIR enabled"
I0914 07:15:18.977200 [1 defaults.go:112] Defaulting to storage driver "overlay" with options [mountopt=metacopy=on].
Caching blobs under "/var/cache/blobs".
Trying to pull registry.access.redhat.com/ubi8/openjdk-11:sha256:08362e3aa3675784930149da5ddf5bec416f8130ff508da0460cbccf51416b9e...
Getting image source signatures
Copying blob sha256:b4ef52c030245341405883294991f0c0a1de14addadb69a7a08413c9fb1bf701
Copying blob sha256:0c10cd59e10eb07960a86667f3bbc3d156c315246f8cd60742882b3383e61b59
Copying config sha256:c71cc69728232de4fae5d418c4b328738aa47992e17f5fd9ce58f29424806d
Writing manifest to image destination
Storing signatures
Generating dockerfile with builder image registry.access.redhat.com/ubi8/openjdk-11:sha256:08362e3aa3675784930149da5ddf5bec416f8130ff508da0460cbccf51416b9e
Adding transient rw bind mount for /run/secrets/rhsm
STEP 1/9: FROM registry.access.redhat.com/ubi8/openjdk-11:sha256:08362e3aa3675784930149da5ddf5bec416f8130ff508da0460cbccf51416b9e
STEP 2/9: LABEL "io.openshift.s2i.destination"="/tmp" "io.openshift.build.image"="registry.access.redhat.com/ubi8/openjdk-11:sha256:08362e3aa3675784930149da5ddf5bec416f8130ff508da0460cbccf51416b9e
```

```
jegan@tektutor:~/openshift-sep-2023/Day4
[jean@tektutor:~] [jegan@tektutor:~/ansible-sep-2023] [jegan@tektutor:~/openshift-sep-2023] [jegan@tektutor:~/openshift-sep-202...] [jegan@tektutor:~/openshift-sep-202...] [jegan@tektutor:~]
[INFO] Downloaded from central: https://repo1.maven.org/maven2/org/eclipse/sisu/org.eclipse.sisu.inject/0.3.4/org.eclipse.sisu.inject-0.3.4.jar (379 kB at 315 kB/s)
[INFO] Downloaded from central: https://repo1.maven.org/maven2/org/codehaus/plexus/plexus-classworlds/2.6.0/plexus-classworlds-2.6.0.jar (53 kB at 38 kB/s)
[INFO] Downloaded from central: https://repo1.maven.org/maven2/org/codehaus/plexus/plexus-utils/3.2.1/plexus-utils-3.2.1.jar (262 kB at 180 kB/s)
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:24 min
[INFO] Finished at: 2023-09-14T07:17:57Z
[INFO] -----
[WARNING] The requested profile "openshift" could not be activated because it does not exist.
INFO Copying deployments from target to /deployments...
'./tmp/src/target/spring-hello-1.0.jar' -> './deployments/spring-hello-1.0.jar'
INFO Cleaning up source directory ('./tmp/src')
STEP 9/9: CMD /usr/local/bin/run
COMMIT temp.builder.openshift.io/jegan/spring-ms-1:0cde66f9
Getting image source signatures
Copying blob sha256:ad5362eb8973710afa7450bf83d23979133ed21195b84118618dedc15e41c96
Copying blob sha256:6541aa1c8d041f0b1979591c4762fa962e0d2649f68e2a579c6ef2059ce8b806
Copying blob sha256:dcf8d29bf1c92962c9b88708fb9f9632593fafba36ccf6d22f8290a7bcab8921
Copying config sha256:c2502d3ce1c44a1d81a5ee7e57a07416616b2f79436bdcf8652be6488eb2e616
Writing manifest to image destination
Storing signatures
--> c2502d3ce1c
Successfully tagged temp.builder.openshift.io/jegan/spring-ms-1:0cde66f9
c2502d3ce1c44a1d81a5ee7e57a07416616b2f79436bdcf8652be6488eb2e616

Pushing image image-registry.openshift-image-registry.svc:5000/jegan/spring-ms:latest ...
Getting image source signatures
Copying blob sha256:dcf8d29bf1c92962c9b88708fb9f9632593fafba36ccf6d22f8290a7bcab8921
Copying blob sha256:b4ef52c030245341405883294991f0c0a1de14addad69a7a08413c9fb1bf701
Copying blob sha256:0c10cd59e10eb07960a86667f3bbc3d156c315246f8cd60742882b3383e61b59
Copying config sha256:c2502d3ce1c44a1d81a5ee7e57a07416616b2f79436bdcf8652be6488eb2e616
Writing manifest to image destination
Storing signatures
Successfully pushed image-registry.openshift-image-registry.svc:5000/jegan/spring-ms:sha256:40b1675706a45479b0891ca201ba7ef9a7e68abe9c3c35c08f2b0a695b50f21c
Push successful

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
```

Let's create a route for the spring-ms service

```
oc get svc
oc expose svc/spring-ms
oc get routes
```

Expected output

```
jegan@tektutor:~/openshift-sep-2023/Day4
[jean@tektutor:~] [jegan@tektutor:~/ansible-sep-2023] [jegan@tektutor:~/openshift-sep-2023] [jegan@tektutor:~/openshift-sep-202...] [jegan@tektutor:~/openshift-sep-202...] [jegan@tektutor:~]
[WARNING] The requested profile "openshift" could not be activated because it does not exist.
INFO Copying deployments from target to /deployments...
'./tmp/src/target/spring-hello-1.0.jar' -> './deployments/spring-hello-1.0.jar'
INFO Cleaning up source directory ('./tmp/src')
STEP 9/9: CMD /usr/local/bin/run
COMMIT temp.builder.openshift.io/jegan/spring-ms-1:0cde66f9
Getting image source signatures
Copying blob sha256:ad5362eb8973710afa7450bf83d23979133ed21195b84118618dedc15e41c96
Copying blob sha256:6541aa1c8d041f0b1979591c4762fa962e0d2649f68e2a579c6ef2059ce8b806
Copying blob sha256:dcf8d29bf1c92962c9b88708fb9f9632593fafba36ccf6d22f8290a7bcab8921
Copying config sha256:c2502d3ce1c44a1d81a5ee7e57a07416616b2f79436bdcf8652be6488eb2e616
Writing manifest to image destination
Storing signatures
--> c2502d3ce1c
Successfully tagged temp.builder.openshift.io/jegan/spring-ms-1:0cde66f9
c2502d3ce1c44a1d81a5ee7e57a07416616b2f79436bdcf8652be6488eb2e616

Pushing image image-registry.openshift-image-registry.svc:5000/jegan/spring-ms:latest ...
Getting image source signatures
Copying blob sha256:dcf8d29bf1c92962c9b88708fb9f9632593fafba36ccf6d22f8290a7bcab8921
Copying blob sha256:b4ef52c030245341405883294991f0c0a1de14addad69a7a08413c9fb1bf701
Copying blob sha256:0c10cd59e10eb07960a86667f3bbc3d156c315246f8cd60742882b3383e61b59
Copying config sha256:c2502d3ce1c44a1d81a5ee7e57a07416616b2f79436bdcf8652be6488eb2e616
Writing manifest to image destination
Storing signatures
Successfully pushed image-registry.openshift-image-registry.svc:5000/jegan/spring-ms:sha256:40b1675706a45479b0891ca201ba7ef9a7e68abe9c3c35c08f2b0a695b50f21c
Push successful

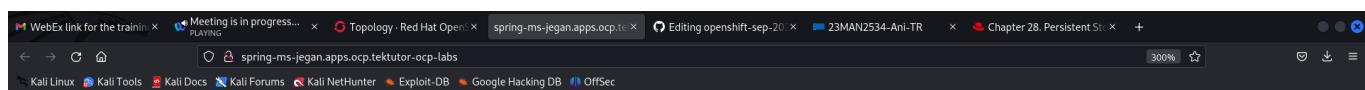
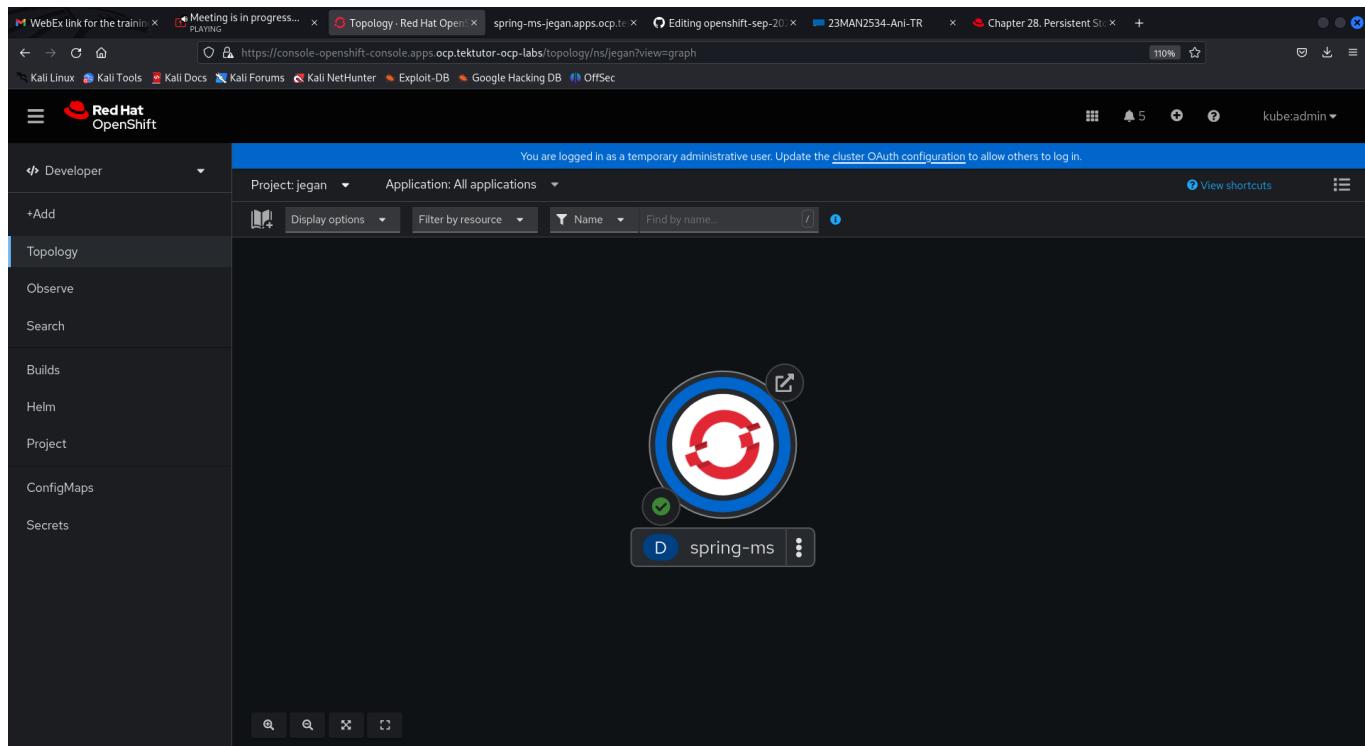
(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc get svc
oc expose svc/spring-ms
oc get routes

NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
spring-ms   ClusterIP   172.30.248.193   <none>        8080/TCP,8443/TCP,8778/TCP   4m49s
route.route.openshift.io/spring-ms exposed

NAME      HOST/PORT      PATH      SERVICES      PORT      TERMINATION      WILDCARD
spring-ms   spring-ms-jegan.apps.ocp.tektutor-ocp-labs      spring-ms      8080-tcp      None

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4]
```

Now you may access the route from Developer context Topology from your web browser on the CentOS Lab machine.



Hello Microservice !

Lab - Deploying custom appling using Docker Hub custom image

```
oc delete project/jegan
oc new-project jegan
oc new-app tektutor/spring-ms:1.0
```

Expected output

```
jegan@tektutor: ~ | jegan@tektutor: ~/ansible-sep-2023 | jegan@tektutor: ~/openshift-sep-2023 | jegan@tektutor: ~/openshift-sep-202... | jegan@tektutor: ~/openshift-sep-202... | jegan@tektutor: ~
(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc delete project/jegan
project.project.openshift.io "jegan" deleted

(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc new-project jegan
Already on project "jegan" on server "https://api.ocp.tektutor-ocp-labs:6443".

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

  oc new-app rails-postgresql-example

to build a new example application in Ruby. Or use kubectl to deploy a simple Kubernetes application:

  kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-images/agnhost:2.43 -- /agnhost serve-hostname

(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc new-app tektutor/spring-ms:1.0
--> Found container image 9175b94 (13 months old) from Docker Hub for "tektutor/spring-ms:1.0"

* An image stream tag will be created as "spring-ms:1.0" that will track this image

--> Creating resources ...
imagestream.image.openshift.io "spring-ms" created
deployment.apps "spring-ms" created
service "spring-ms" 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 service/spring-ms'
Run 'oc status' to view your app.

(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
$
```

Let us check the deployment status

```
oc status
oc get svc
oc expose svc/spring-ms
oc get route
```

Expected output

```
jegan@tektutor: ~ | jegan@tektutor: ~/ansible-sep-2023 | jegan@tektutor: ~/openshift-sep-2023 | jegan@tektutor: ~/openshift-sep-202... | jegan@tektutor: ~/openshift-sep-202... | jegan@tektutor: ~
(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
service "spring-ms" 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 service/spring-ms'
Run 'oc status' to view your app.

(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc status
In project jegan on server https://api.ocp.tektutor-ocp-labs:6443

svc/spring-ms - 172.30.42.250:8080
  deployment/spring-ms deploys istag/spring-ms:1.0
    deployment #2 running for 49 seconds - 1 pod
    deployment #1 deployed 52 seconds ago

1 info identified, use 'oc status --suggest' to see details.

(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc get svc
NAME        TYPE      CLUSTER-IP      EXTERNAL-IP     PORT(S)      AGE
spring-ms   ClusterIP  172.30.42.250  <none>        8080/TCP    67s

(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc expose service/spring-ms
route.route.openshift.io/spring-ms exposed

(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
$ oc get route
NAME      HOST/PORT      PATH      SERVICES      PORT      TERMINATION      WILDCARD
spring-ms  spring-ms-jegan.apps.ocp.tektutor-ocp-labs  spring-ms  8080-tcp  None

(jegan@ tektutor.org)-[~/openshift-sep-2023/Day4]
$
```

Now you may access the application using its route url from the OpenShift webconsole

The screenshot shows the Red Hat OpenShift webconsole interface. The left sidebar has a 'Topology' section selected. The main area shows a single application named 'spring-ms' with a blue circular icon containing a red circular arrow. Below the icon is a button labeled 'D spring-ms'. The URL in the browser bar is <https://console.openshift-console.apps.ocp.tektutor-ocp-labs/topology/ns/jegan?view=graph>.

Greetings from Spring Boot!

Lab - Node Affinity Required criteria

When creating the pod, we have added some criteria for the default-scheduler to follow. When the criteria is required, the scheduler will look for nodes that has a label matching "ssd" type disk, in case it isn't able to find a node that has label disk=ssd then the Pod will not be deployed.

```
cd ~/openshift-sep-2023
git pull
cd Day4/node-affinity
oc apply -f pod-with-node-affinity-required.yml
```

```
oc get po  
oc describe pod/hello
```

Expected output

```
jegan@tektutor:~/openshift-sep-2023/Day4/node-affinity
jegan@tektutor:~ x jegan@tektutor:~/ansible-sep-2023 x jegan@tektutor:~/openshift-sep-2023 x jegan@tektutor:~/openshift-sep-202... x jegan@tektutor:~/openshift-sep-202... x jegan@tektutor:~ x
(jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc apply -f pod-with-node-affinity-required.yml
Warning: would violate PodSecurity "restricted:v1.24": allowPrivilegeEscalation != false (container "hello" must set securityContext.allowPrivilegeEscalation=false), unrestricted capabilities (container "hello" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "hello" must set securityContext.runAsNonRoot=true), seccompProfile (pod or container "hello" must set securityContext.seccompProfile.type to "RuntimeDefault" or "Localhost")
pod/hello created

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc get node -l disk=ssd
No resources found

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc get po -w
NAME READY STATUS RESTARTS AGE
hello 0/1 Pending 0 16s
^C

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc describe pod/hello
Name: hello
Namespace: jegan
Priority: 0
Service Account: default
Node: <none>
Labels: run=hello
Annotations: openshift.io/scc: anyuid
Status: Pending
IP:
IPs:
Containers:
hello:
  Image: tektutor/spring-tektutor-helloms:latest
  Port: <none>
  .....
```

```
jegan@tektutor:~/openshift-sep-2023/Day4/node-affinity
jegan@tektutor:~ x jegan@tektutor:~/ansible-sep-2023 x jegan@tektutor:~/openshift-sep-2023 x jegan@tektutor:~/openshift-sep-202... x jegan@tektutor:~/openshift-sep-202... x jegan@tektutor:~ x
IP:
IPs:
Containers:
hello:
  Image: tektutor/spring-tektutor-helloms:latest
  Port: <none>
  Host Port: <none>
  Environment: <none>
  Mounts:
    /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-ffldt (ro)
Conditions:
Type Status
PodScheduled False
Volumes:
kube-api-access-ffldt:
  Type: Projected (a volume that contains injected data from multiple sources)
  TokenExpirationSeconds: 3607
  ConfigMapName: kube-root-ca.crt
  ConfigMapOptional: <nil>
  DownwardAPI: true
  ConfigMapName: openshift-service-ca.crt
  ConfigMapOptional: <nil>
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
Type Reason Age From Message
---- ---- -- -
Warning FailedScheduling 16s default-scheduler 0/5 nodes are available: 5 node(s) didn't match Pod's node affinity/selector. preemption: 0/5 nodes are available: 5 Preemption is not helpful for scheduling..
```

Now let's add the label to worker-1 node and see the Pod getting deployed into worker-1.

```
oc label node/worker-1.ocp.tektutor-ocp-labs disk=ssd  
oc get po -w  
oc get po -o wide  
oc get nodes -l disk=ssd
```

Expected output

```
jegan@tektutor: ~ | jegan@tektutor: ~/ansible-sep-2023 | jegan@tektutor: ~/openshift-sep-2023 | jegan@tektutor: ~/openshift-sep-202... | jegan@tektutor: ~/openshift-sep-202... | jegan@tektutor: ~
jegan@tektutor: ~
master-1.ocp.tektutor-ocp-labs Ready control-plane,master,worker 7d8h v1.26.7+0ef5eae
master-2.ocp.tektutor-ocp-labs Ready control-plane,master,worker 7d8h v1.26.7+0ef5eae
master-3.ocp.tektutor-ocp-labs Ready control-plane,master,worker 7d8h v1.26.7+0ef5eae
worker-1.ocp.tektutor-ocp-labs Ready worker 7d8h v1.26.7+0ef5eae
worker-2.ocp.tektutor-ocp-labs Ready worker 7d8h v1.26.7+0ef5eae

[jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc get po
NAME READY STATUS RESTARTS AGE
hello 0/1 Pending 0 5m35s

[jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc label node/worker-1.ocp.tektutor-ocp-labs disk:ssd
node/worker-1.ocp.tektutor-ocp-labs labeled

[jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc get po -w
NAME READY STATUS RESTARTS AGE
hello 0/1 ContainerCreating 0 5m51s
hello 1/1 Running 0 6m9s
^C

[jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc get po -o wide
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
hello 1/1 Running 0 6m16s 10.128.2.16 worker-1.ocp.tektutor-ocp-labs <none> <none>

[jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc get nodes -l disk:ssd
NAME STATUS ROLES AGE VERSION
worker-1.ocp.tektutor-ocp-labs Ready worker 7d8h v1.26.7+0ef5eae

[jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$
```

Lab - Pod manifest has a preferred criteria i.e node has label disk=ssd

How the node affinity works in case of Preferred ?

- The scheduler will search for nodes that has label disk=ssd, if it is able to find a node that has the label then the Pod will be scheduled there
- If the scheduler isn't able to find such a node, the scheduler will deploy it on any node as per scheduler's choice

```
cd ~/openshift-sep-2023
git pull
oc delete pod/hello
cd Day4/node-affinity
oc apply -f pod-with-node-affinity-preferred.yml
oc get nodes -l disk:ssd
oc get po
```

Expected output

```
(jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc delete pod/hello
pod "hello" deleted

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc apply -f pod-with-node-affinity-preferred.yml
Warning: would violate PodSecurity "restricted:v1.24": allowPrivilegeEscalation != false (container "hello" must set securityContext.allowPrivilegeEscalation=false), unrestricted capabilities (container "hello" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "hello" must set securityContext.runAsNonRoot=true), seccompProfile (pod or container "hello" must set securityContext.seccompProfile.type to "RuntimeDefault" or "Localhost")
pod/hello created

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc get nodes -l disk=ssd
No resources found

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$ oc get po
NAME      READY   STATUS    RESTARTS   AGE
hello     1/1     Running   0          13s

(jegan@tektutor.org)-[~/openshift-sep-2023/Day4/node-affinity]
$
```

Info - Docker Network Model

Info - OpenShift/Kubernetes Network Model

Post Test Link

<https://app.mymapit.in/code4/tiny/yTefEw>

Feedback link :

<https://tcheck.co/HgqK27>

Installing OpenShift on your laptop

<https://developers.redhat.com/products/openshift-local/overview>

You need to login to your RedHat account to download crc compressed file and pullsecret files.

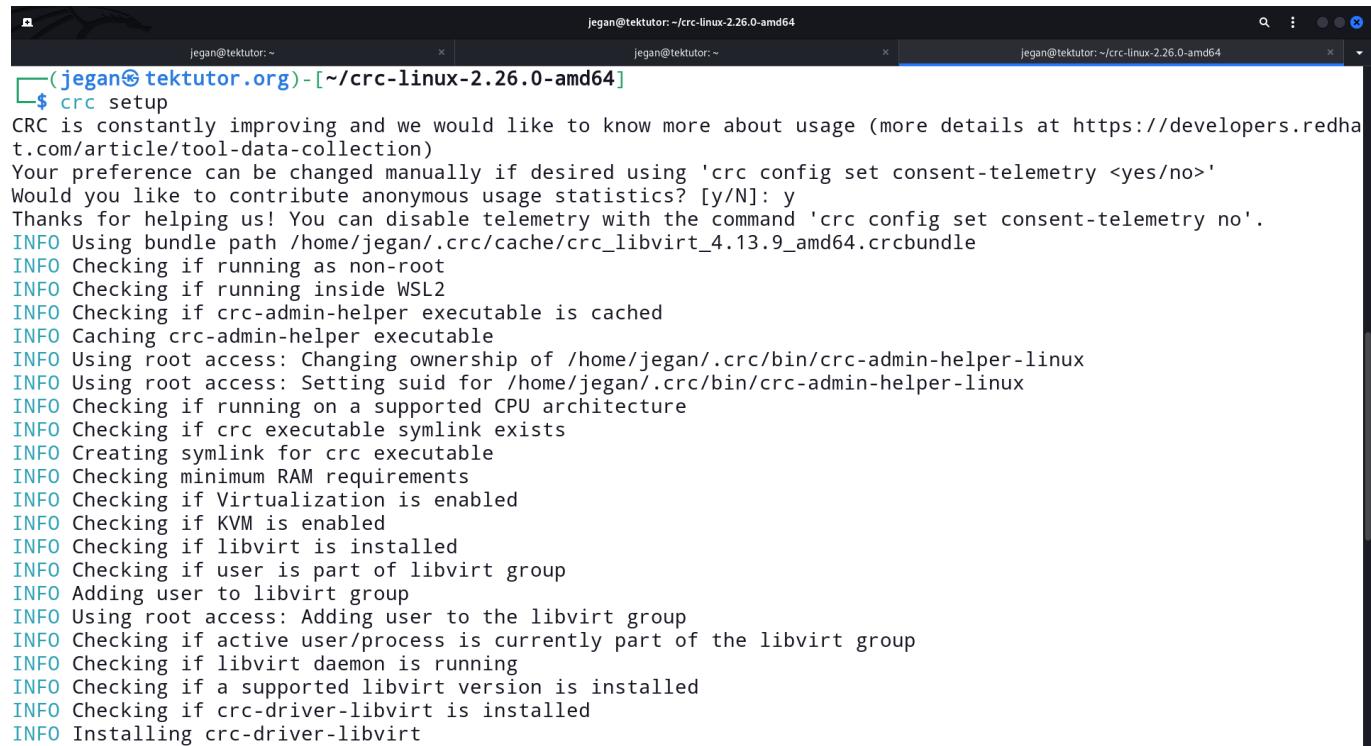
You need to extract it

```
cd ~/Downloads
mv crc-linux-amd64.tar.xz ..
cd ..
tar xvf crc-linux-amd64.tar.xz
cd crc-linux-amd64.tar.xz
sudo cp crc /usr/bin
```

Now you can start the Code Ready Container Setup

```
crc setup
```

Expected output



The screenshot shows a terminal window with three tabs, all titled 'jegan@tektutor: ~'. The central tab is active and displays the output of the 'crc setup' command. The output is as follows:

```
jegan@tektutor: ~
```

```
(jegan@tektutor.org)-[~/crc-linux-2.26.0-amd64]
```

```
$ crc setup
```

```
CRC is constantly improving and we would like to know more about usage (more details at https://developers.redhat.com/article/tool-data-collection)
```

```
Your preference can be changed manually if desired using 'crc config set consent-telemetry <yes/no>'
```

```
Would you like to contribute anonymous usage statistics? [y/N]: y
```

```
Thanks for helping us! You can disable telemetry with the command 'crc config set consent-telemetry no'.
```

```
INFO Using bundle path /home/jegan/.crc/cache/crc_libvirt_4.13.9_amd64.crcbundle
```

```
INFO Checking if running as non-root
```

```
INFO Checking if running inside WSL2
```

```
INFO Checking if crc-admin-helper executable is cached
```

```
INFO Caching crc-admin-helper executable
```

```
INFO Using root access: Changing ownership of /home/jegan/.crc/bin/crc-admin-helper-linux
```

```
INFO Using root access: Setting uid for /home/jegan/.crc/bin/crc-admin-helper-linux
```

```
INFO Checking if running on a supported CPU architecture
```

```
INFO Checking if crc executable symlink exists
```

```
INFO Creating symlink for crc executable
```

```
INFO Checking minimum RAM requirements
```

```
INFO Checking if Virtualization is enabled
```

```
INFO Checking if KVM is enabled
```

```
INFO Checking if libvirt is installed
```

```
INFO Checking if user is part of libvirt group
```

```
INFO Adding user to libvirt group
```

```
INFO Using root access: Adding user to the libvirt group
```

```
INFO Checking if active user/process is currently part of the libvirt group
```

```
INFO Checking if libvirt daemon is running
```

```
INFO Checking if a supported libvirt version is installed
```

```
INFO Checking if crc-driver-libvirt is installed
```

```
INFO Installing crc-driver-libvirt
```

```
jegan@tektutor: ~/crc-linux-2.26.0-amd64
jegan@tektutor: ~
jegan@tektutor: ~

INFO Installing crc-driver-libvirt
INFO Checking crc daemon systemd service
INFO Setting up crc daemon systemd service
INFO Checking crc daemon systemd socket units
INFO Setting up crc daemon systemd socket units
INFO Checking if systemd-networkd is running
INFO Checking if NetworkManager is installed
INFO Checking if NetworkManager service is running
INFO Checking if /etc/NetworkManager/conf.d/crc-nm-dnsmasq.conf exists
INFO Writing Network Manager config for crc
INFO Using root access: Writing NetworkManager configuration to /etc/NetworkManager/conf.d/crc-nm-dnsmasq.conf
INFO Using root access: Changing permissions for /etc/NetworkManager/conf.d/crc-nm-dnsmasq.conf to 644
INFO Using root access: Executing systemctl daemon-reload command
INFO Using root access: Executing systemctl reload NetworkManager
INFO Checking if /etc/NetworkManager/dnsmasq.d/crc.conf exists
INFO Writing dnsmasq config for crc
INFO Using root access: Writing NetworkManager configuration to /etc/NetworkManager/dnsmasq.d/crc.conf
INFO Using root access: Changing permissions for /etc/NetworkManager/dnsmasq.d/crc.conf to 644
INFO Using root access: Executing systemctl daemon-reload command
INFO Using root access: Executing systemctl reload NetworkManager
INFO Checking if libvirt 'crc' network is available
INFO Setting up libvirt 'crc' network
INFO Checking if libvirt 'crc' network is active
INFO Starting libvirt 'crc' network
INFO Checking if CRC bundle is extracted in '$HOME/.crc'
INFO Checking if /home/jegan/.crc/cache/crc_libvirt_4.13.9_amd64.crcbundle exists
INFO Getting bundle for the CRC executable
INFO Downloading bundle: /home/jegan/.crc/cache/crc_libvirt_4.13.9_amd64.crcbundle...
210.64 MiB / 3.94 GiB [--->] 5.22% 1.27 MiB/s
```

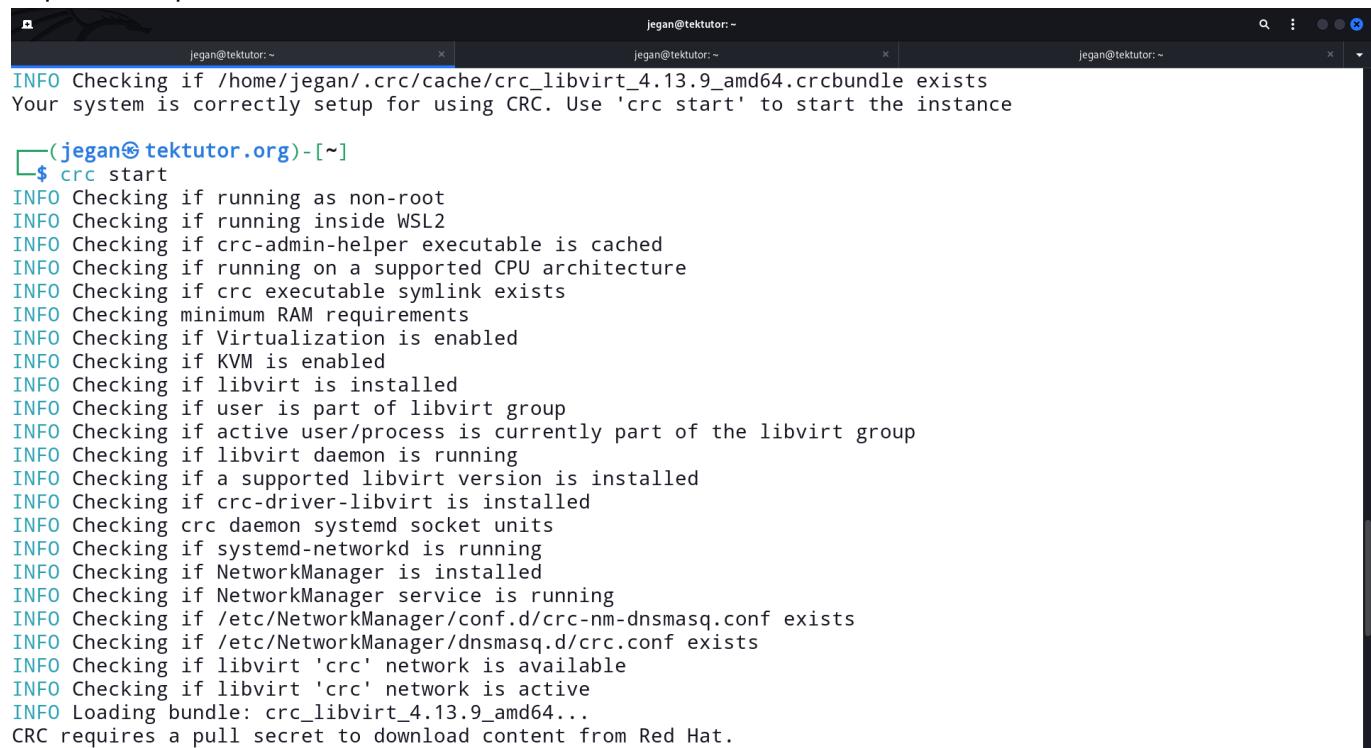
```
jegan@tektutor:~/crc-linux-2.26.0-amd64
```

```
INFO Writing Network Manager config for crc
INFO Using root access: Writing NetworkManager configuration to /etc/NetworkManager/conf.d/crc-nm-dnsmasq.conf
INFO Using root access: Changing permissions for /etc/NetworkManager/conf.d/crc-nm-dnsmasq.conf to 644
INFO Using root access: Executing systemctl daemon-reload command
INFO Using root access: Executing systemctl reload NetworkManager
INFO Checking if /etc/NetworkManager/dnsmasq.d/crc.conf exists
INFO Writing dnsmasq config for crc
INFO Using root access: Writing NetworkManager configuration to /etc/NetworkManager/dnsmasq.d/crc.conf
INFO Using root access: Changing permissions for /etc/NetworkManager/dnsmasq.d/crc.conf to 644
INFO Using root access: Executing systemctl daemon-reload command
INFO Using root access: Executing systemctl reload NetworkManager
INFO Checking if libvirt 'crc' network is available
INFO Setting up libvirt 'crc' network
INFO Checking if libvirt 'crc' network is active
INFO Starting libvirt 'crc' network
INFO Checking if CRC bundle is extracted in '$HOME/.crc'
INFO Checking if /home/jegan/.crc/cache/crc_libvirt_4.13.9_amd64.crcbundle exists
INFO Getting bundle for the CRC executable
INFO Downloading bundle: /home/jegan/.crc/cache/crc_libvirt_4.13.9_amd64.crcbundle...
906.42 MiB / 3.94 GiB [----->] 930.00 MiB / 3.94 GiB 23.04
1.05 GiB / 3.94 GiB [----->] 26.76% 560.11 KiB/s
1.14 GiB / 3.94 GiB [----->] 28.83% 657.29 K
3.94 GiB / 3.94 GiB [----->] 100.00% 603.89 KiB/s
INFO Uncompressing /home/jegan/.crc/cache/crc_libvirt_4.13.9_amd64.crcbundle
crc.qcow2: 15.08 GiB / 15.08 GiB [----->] 100.00%
oc: 141.85 MiB / 141.85 MiB [----->] 100.00%
Your system is correctly setup for using CRC. Use 'crc start' to start the instance
```

Now you need to start the CRC as shown below

crc start

Expected output



The screenshot shows a terminal window with three tabs, all titled "jegan@tektutor:~". The central tab is active and displays the output of the "crc start" command. The output is a series of informational messages (INFO) detailing the process of starting the CRC service. It includes checks for file existence, system setup, virtualization support, and various system services like libvirt, NetworkManager, and dnsmasq. The final message indicates that CRC requires a pull secret to download content from Red Hat.

```
INFO Checking if /home/jegan/.crc/cache/crc_libvirt_4.13.9_amd64.crcbundle exists
Your system is correctly setup for using CRC. Use 'crc start' to start the instance
[jegan@tektutor.org] ~
$ crc start
INFO Checking if running as non-root
INFO Checking if running inside WSL2
INFO Checking if crc-admin-helper executable is cached
INFO Checking if running on a supported CPU architecture
INFO Checking if crc executable symlink exists
INFO Checking minimum RAM requirements
INFO Checking if Virtualization is enabled
INFO Checking if KVM is enabled
INFO Checking if libvirt is installed
INFO Checking if user is part of libvirt group
INFO Checking if active user/process is currently part of the libvirt group
INFO Checking if libvirt daemon is running
INFO Checking if a supported libvirt version is installed
INFO Checking if crc-driver-libvirt is installed
INFO Checking crc daemon systemd socket units
INFO Checking if systemd-networkd is running
INFO Checking if NetworkManager is installed
INFO Checking if NetworkManager service is running
INFO Checking if /etc/NetworkManager/conf.d/crc-nm-dnsmasq.conf exists
INFO Checking if /etc/NetworkManager/dnsmasq.d/crc.conf exists
INFO Checking if libvirt 'crc' network is available
INFO Checking if libvirt 'crc' network is active
INFO Loading bundle: crc_libvirt_4.13.9_amd64...
CRC requires a pull secret to download content from Red Hat.
```

```
jegan@tektutor:~  
INFO Loading bundle: crc_libvirt_4.13.9_amd64...  
CRC requires a pull secret to download content from Red Hat.  
You can copy it from the Pull Secret section of https://console.redhat.com/openshift/create/local.  
? Please enter the pull secret *****  
INFO Creating CRC VM for OpenShift 4.13.9...  
INFO Generating new SSH key pair...  
INFO Generating new password for the kubeadmin user  
INFO Starting CRC VM for openshift 4.13.9...  
INFO CRC instance is running with IP 192.168.130.11  
INFO CRC VM is running  
INFO Updating authorized keys...  
INFO Configuring shared directories  
INFO Check internal and public DNS query...  
INFO Check DNS query from host...  
INFO Verifying validity of the kubelet certificates...  
INFO Starting kubelet service  
INFO Waiting for kube-apiserver availability... [takes around 2min]  
INFO Adding user's pull secret to the cluster...  
INFO Updating SSH key to machine config resource...  
INFO Waiting until the user's pull secret is written to the instance disk...  
INFO Changing the password for the kubeadmin user  
INFO Updating cluster ID...  
INFO Updating root CA cert to admin-kubeconfig-client-ca configmap...  
INFO Starting openshift instance... [waiting for the cluster to stabilize]  
INFO 8 operators are progressing: dns, image-registry, ingress, kube-storage-version-migrator, network, ...  
INFO 6 operators are progressing: image-registry, ingress, kube-storage-version-migrator, network, openshift-controller-manager, ...  
INFO 5 operators are progressing: image-registry, ingress, kube-storage-version-migrator, network, service-ca  
    INFO 5 operators are progressing: image-registry, ingress, kube-storage-version-migrator, network, service-ca  
    ...  
jegan@tektutor:~
```

```
INFO 4 operators are progressing: image-registry, kube-storage-version-migrator, network, service-ca  
INFO 2 operators are progressing: image-registry, service-ca  
INFO Operator image-registry is progressing  
INFO Operator authentication is degraded  
INFO Operator authentication is not yet available  
INFO Operator authentication is not yet available  
INFO Operator authentication is not yet available  
INFO All operators are available. Ensuring stability...  
INFO Operators are stable (2/3)...  
INFO Operators are stable (3/3)...  
INFO Adding crc-admin and crc-developer contexts to kubeconfig...  
Started the OpenShift cluster.
```

The server is accessible via web console at:
<https://console-openshift-console.apps-crc.testing>

Log in as administrator:

```
Username: kubeadmin  
Password: wYPHP-uLhMY-hMgmS-YKRYh
```

Log in as user:

```
Username: developer  
Password: developer
```

Use the 'oc' command line interface:

```
$ eval $(crc oc-env)  
$ oc login -u developer https://api.crc.testing:6443
```

```
└─(jegan@tektutor.org)-[~]  
$
```

Don't forget to save the Red Hat OpenShift login credentials, in my case I saved it as shown below

```

jegan@tektutor:~ jegan@tektutor:~ jegan@tektutor:~
└$ cat openshift-crc.txt
The server is accessible via web console at:
https://console-openshift-console.apps-crc.testing

Log in as administrator:
Username: kubeadmin
Password: wYPHP-uLhMY-hMgmS-YKRYh

Log in as user:
Username: developer
Password: developer

Use the 'oc' command line interface:
$ eval $(crc oc-env)
$ oc login -u developer https://api.crc.testing:6443

(jegan@tektutor.org)-[~]
$
```

Login to Red Hat OpenShift via command-line and create a project

```
oc login -u developer https://api.crc.testing:6443
```

Expected output

```

jegan@tektutor:~ jegan@tektutor:~ jegan@tektutor:~
└(jegan@tektutor.org)-[~] $ eval $(crc oc-env)
└(jegan@tektutor.org)-[~] $ oc login -u developer https://api.crc.testing:6443
Logged into "https://api.crc.testing:6443" as "developer" using existing credentials.

You don't have any projects. You can try to create a new project, by running
  oc new-project <projectname>

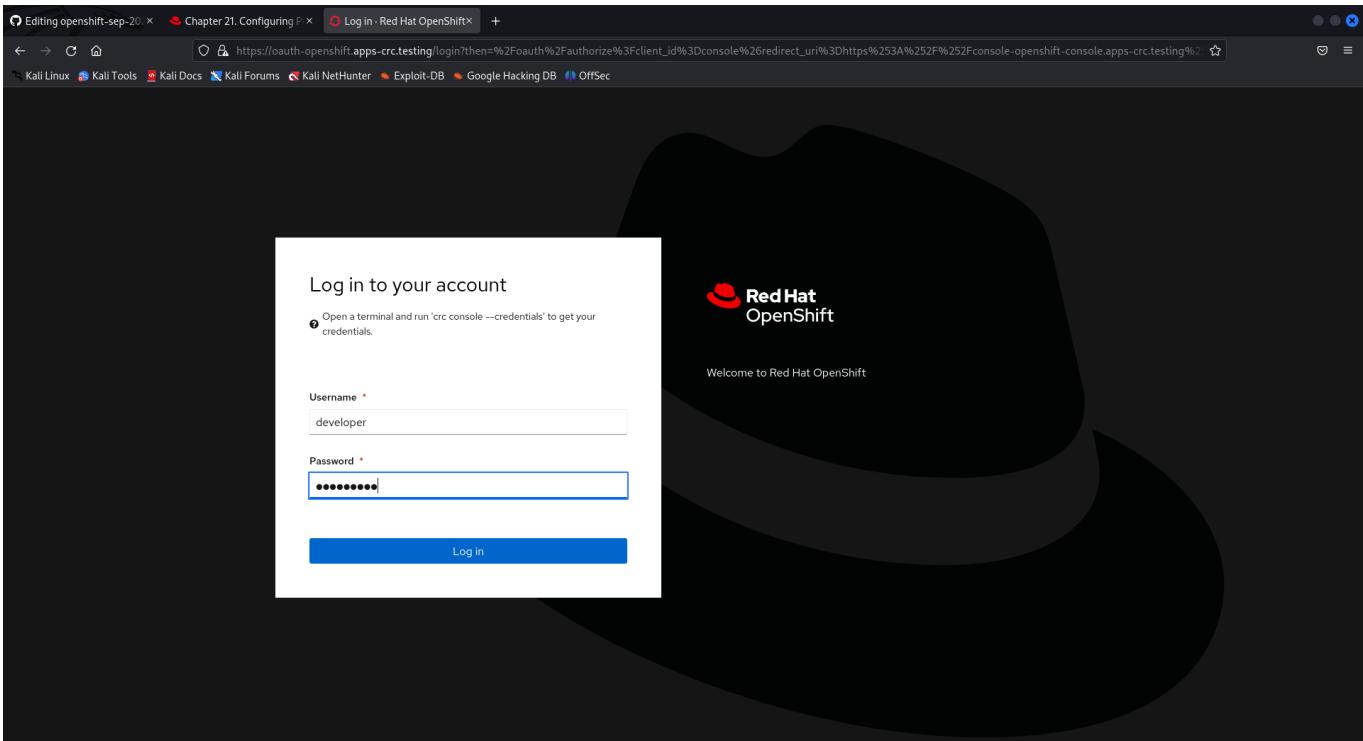
(jegan@tektutor.org)-[~] $ oc new-project jegan
Now using project "jegan" on server "https://api.crc.testing:6443".

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

to build a new example application in Ruby. Or use kubectl to deploy a simple Kubernetes application:
  kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-images/agnhost:2.43 -- /agnhost serve-
hostname

(jegan@tektutor.org)-[~] $ set -o vi
```

You can access the webconsole at the below URL in your web browser with developer as user and developer as password



Deploy an application to check if your cluster is working properly

```
oc new-app registry.access.redhat.com/ubi8/openjdk-11~https://github.com/tektutor/spring-ms.git --strategy=source
```

Expected output

```
jegan@tektutor:~ [jegan@tektutor.org] - [~]
$ oc new-app registry.access.redhat.com/ubi8/openjdk-11~https://github.com/tektutor/spring-ms.git --strategy=source
--> Found container image c71cc69 (2 weeks old) from registry.access.redhat.com for "registry.access.redhat.com/ubi8/openjdk-11"
Java Applications
-----
Platform for building and running plain Java applications (fat-jar and flat classpath)
Tags: builder, java
* An image stream tag will be created as "openjdk-11:latest" that will track the source image
* A source build using source code from https://github.com/tektutor/spring-ms.git will be created
  * The resulting image will be pushed to image stream tag "spring-ms:latest"
  * Every time "openjdk-11:latest" changes a new build will be triggered
--> Creating resources ...
imagestream.image.openshift.io "openjdk-11" created
imagestream.image.openshift.io "spring-ms" created
buildconfig.build.openshift.io "spring-ms" created
deployment.apps "spring-ms" created
service "spring-ms" created
--> Success
Build scheduled, use 'oc logs -f buildconfig/spring-ms' to track its progress.
Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
  'oc expose service/spring-ms'
Run 'oc status' to view your app.
```

Create a route to access the deployed application

```
oc expose service/spring-ms
```

Expected output

```
jegan@tektutor:~ * An image stream tag will be created as "openjdk-11:latest" that will track the source image
* A source build using source code from https://github.com/tektutor/spring-ms.git will be created
* The resulting image will be pushed to image stream tag "spring-ms:latest"
* Every time "openjdk-11:latest" changes a new build will be triggered

--> Creating resources ...
imagestream.image.openshift.io "openjdk-11" created
imagestream.image.openshift.io "spring-ms" created
buildconfig.build.openshift.io "spring-ms" created
deployment.apps "spring-ms" created
service "spring-ms" created
--> Success
Build scheduled, use 'oc logs -f buildconfig/spring-ms' to track its progress.
Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
'oc expose service/spring-ms'
Run 'oc status' to view your app.

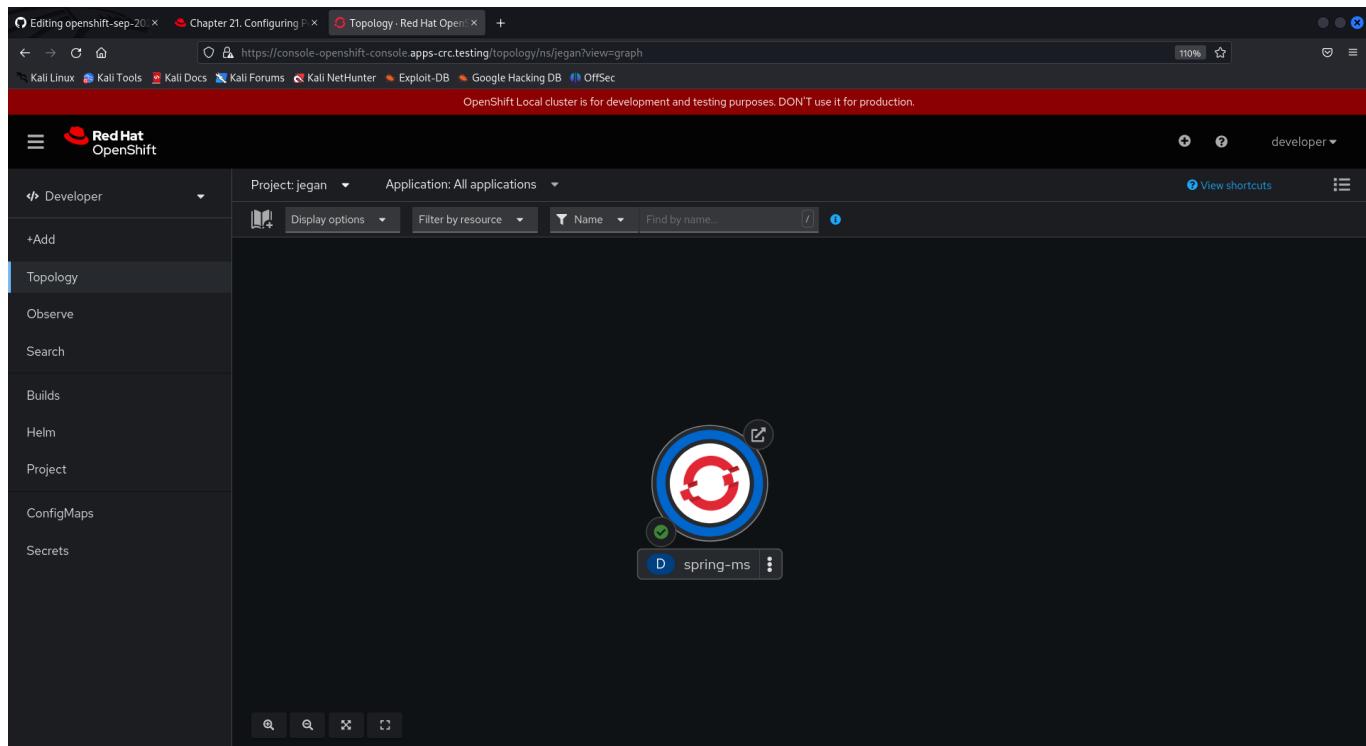
[jegan@tektutor.org]-[~]
$ oc expose service/spring-ms
route.route.openshift.io/spring-ms exposed

[jegan@tektutor.org]-[~]
$ oc get route
NAME            HOST/PORT          PATH  SERVICES    PORT   TERMINATION  WILDCARD
spring-ms       spring-ms-jegan.apps-crc.testing      spring-ms  8080-tcp
None

[jegan@tektutor.org]-[~]
$
```

Now head-over to your Red Hat OpenShift webconsole to access the application route

The screenshot shows the Red Hat OpenShift webconsole interface. On the left, a sidebar menu includes options like Developer, Topology (which is selected), Observe, Search, Builds, Helm, Project, ConfigMaps, and Secrets. The main content area displays a network topology graph for the 'spring-ms' application. In the center, there is a circular icon with a red and blue design, representing the application's status. Below this icon, the text 'D spring-ms' is visible. At the bottom of the screen, there is a navigation bar with icons for search, refresh, and other functions.



Congratulations!, your Red Hat OpenShift local setup is ready to practice ...