OpenShift Developer Distance Learning Program

Homework Assignment

Silke Roth

Why did I choose serverless?

Minimizes resource usage – only deployed if required, application will be idle as long as it is not required

And:
New for me during this training
Curiosity - Will this be as easy as described?

Scenario – Serverless application

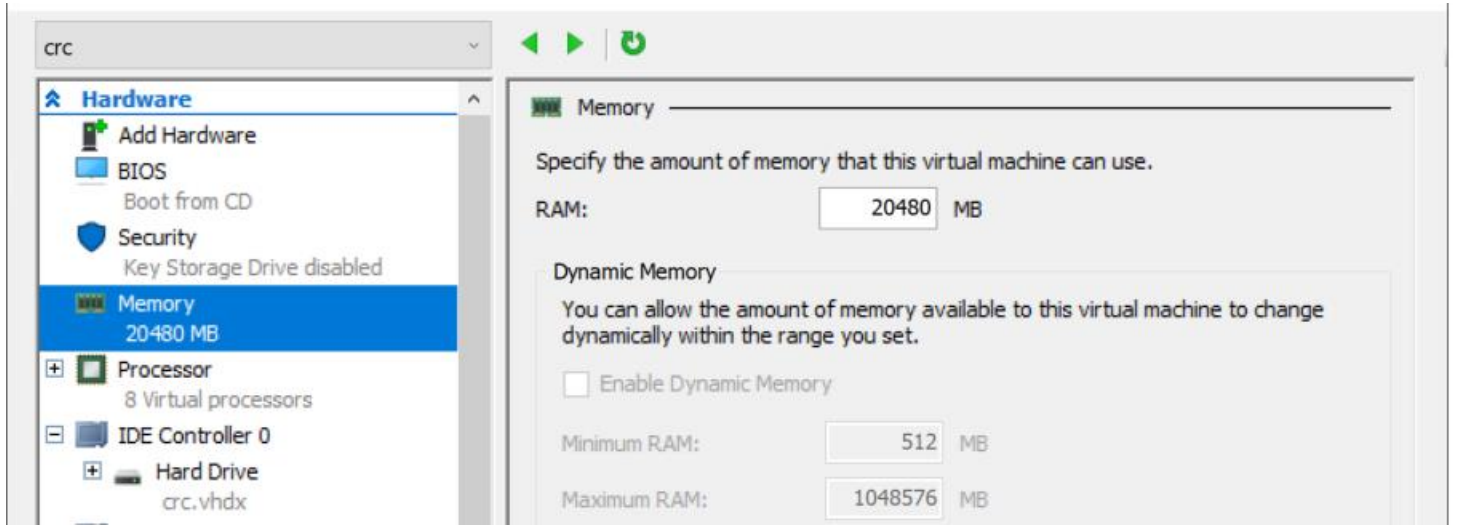
This scenario will demonstrate a simple serverless application with the following steps

- Create an instance of CodeReady Workspace on Windows
- Prepare the environment
- Create a serverless application, based sample nginx application on https://github.com/sclorg/nginx-ex.git
- Test and monitor the application

Setting up CodeReady Workspace

Prerequisites in my Environment:

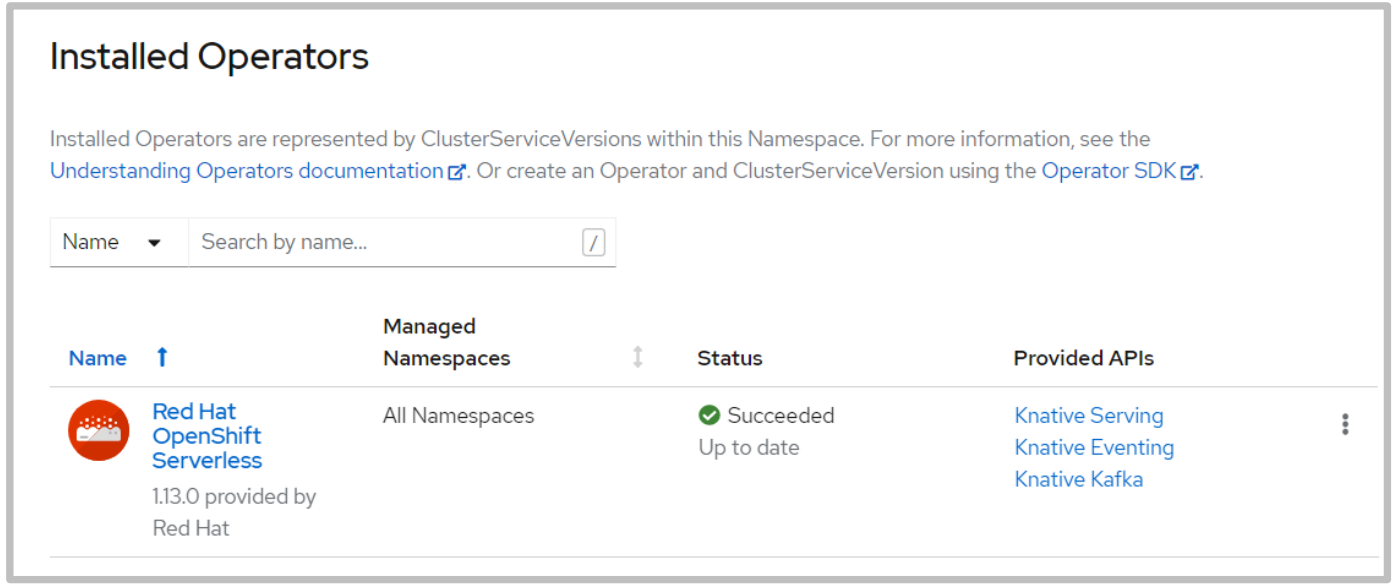
- Windows Server 2019 with Hyper-V role installed, virtual switch with the name **crc**
- Local user account ***openshift***, member of group **Administrators**
- By default the resources for the CodeReady Workspace will be installed in the user profile directory – 60 to 100 GB free space required
- Executing **crc.exe setup** creates virutal machine **crc** in Hyper-V with 4 Core and 9 GB RAM
  - To improve performance I have configured the VM to 8 Core and 20 GB RAM



First Steps

Prepare the environment

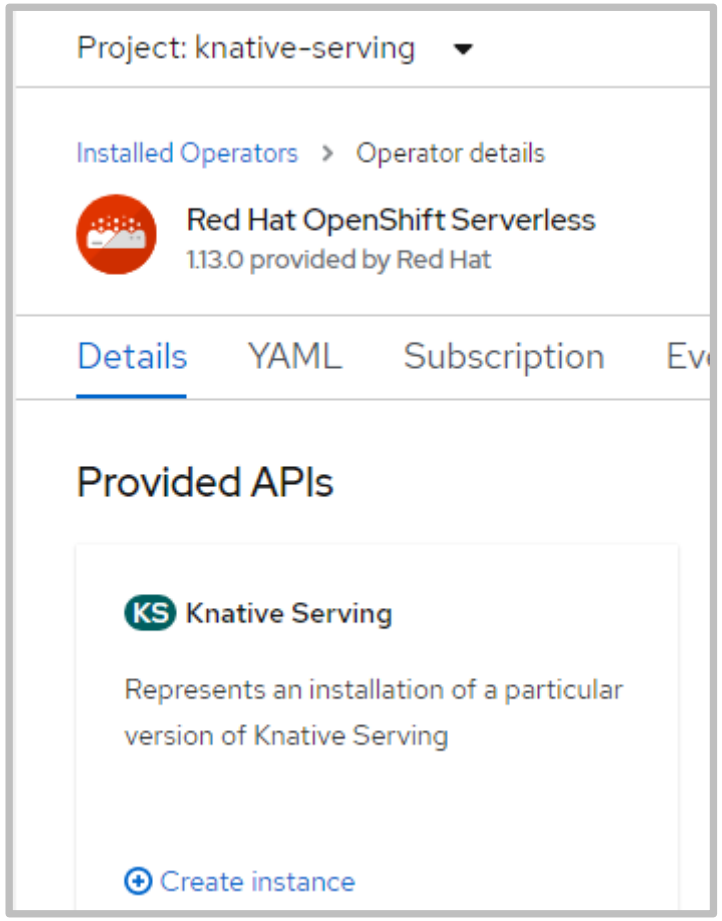
- Access open shift console using chrome browser https://console-openshift-console.apps-crc.testing
- Logon with Administrator role
- Creation of the project **serverless-demo**
- Assign permissions to developer user
- Installation of Operator Red Hat OpenShift Serverless
  - All Namespaces

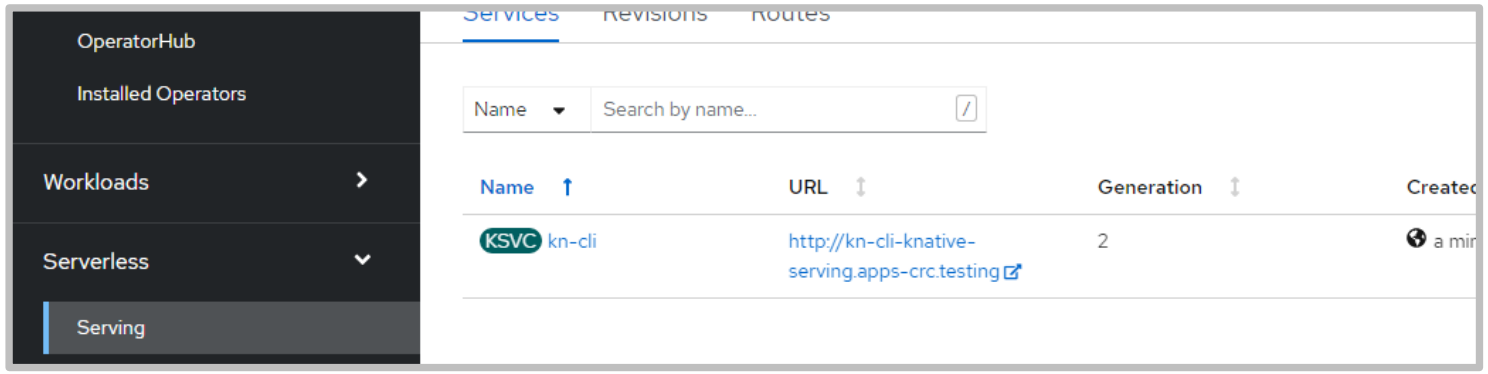


First Steps

Prepare the environment

- Next Knative Serving must be created
- Change to project knative-serving
- Under Knative Serving click **Create instance**
- The new instance can be found under Serverless Serving



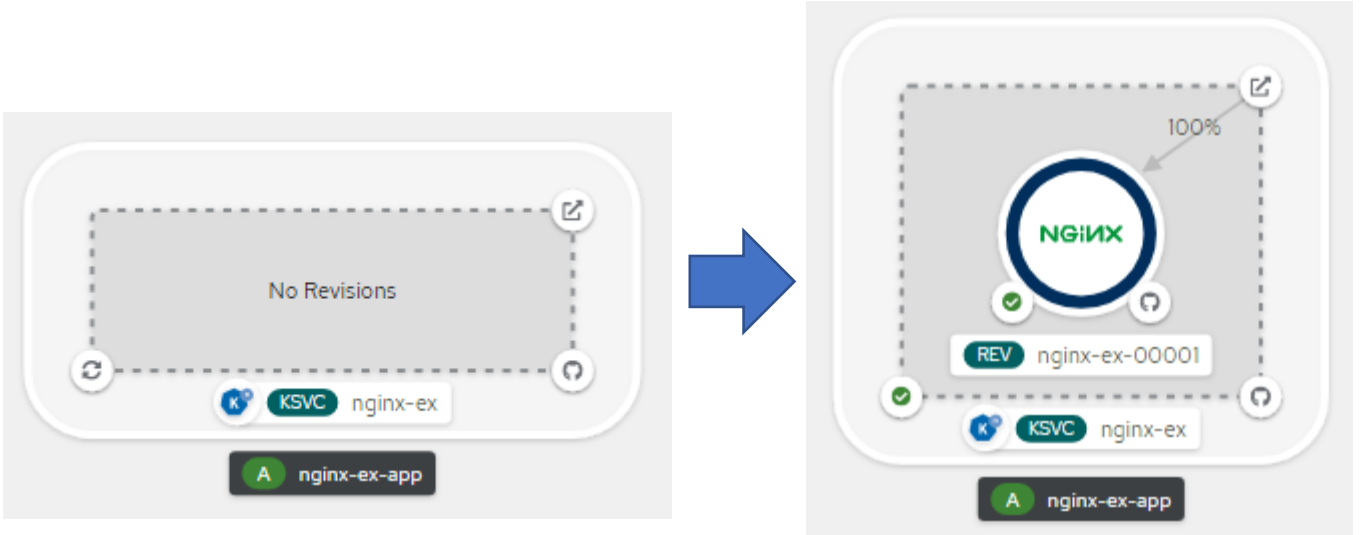


Prepare the Application Soure

- Access open shift console using chrome browser https://console-openshift-console.apps-crc.testing
- Use developer role, ensure that the project **serverless-demo** is selected
- Navigate to **Topology** and click **Samples**
- Select Nginx
- Note the URL of the Sample repository
- Create a copy to your Git account
- → this will be helpful if want to do changes later

Create the serverless Application

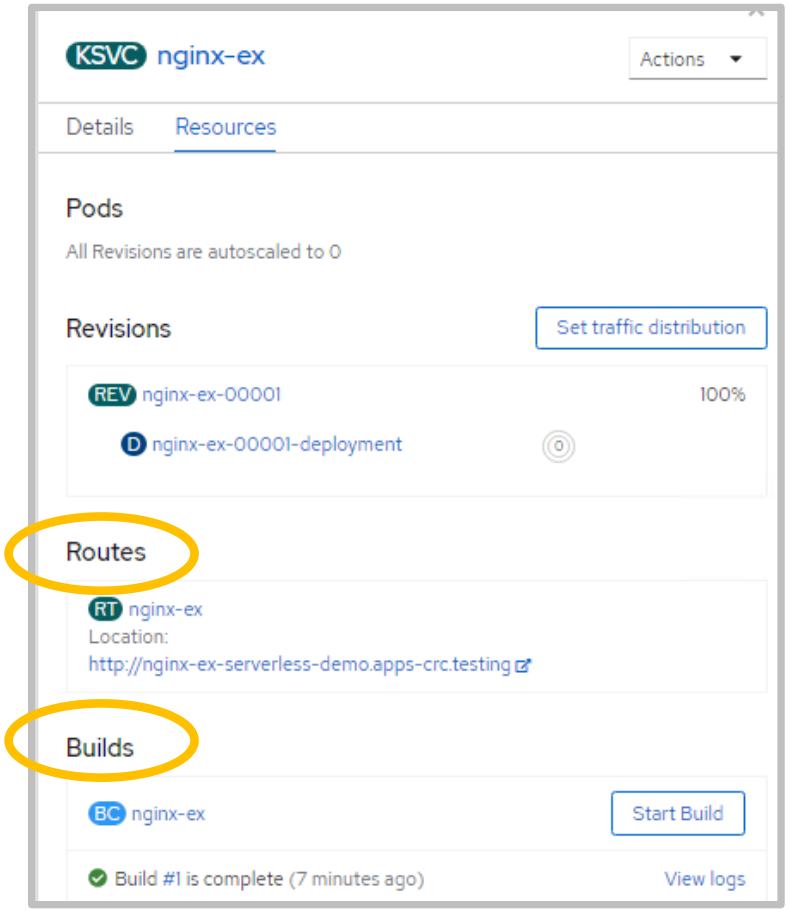
- Access open shift console using chrome browser https://console-openshift-console.apps-crc.testing
- Click **Add** and Select **From Git**
- Enter your Git URL and select Nginx als Builder Image
- Application name: **nginx-ex-app** as per default
- Name: **nginx-ex**
- Select **Knative Service** as resource type
- Check **Create Route to the Application**
- Click **Create**
- The provisioning starts and finishes after short time

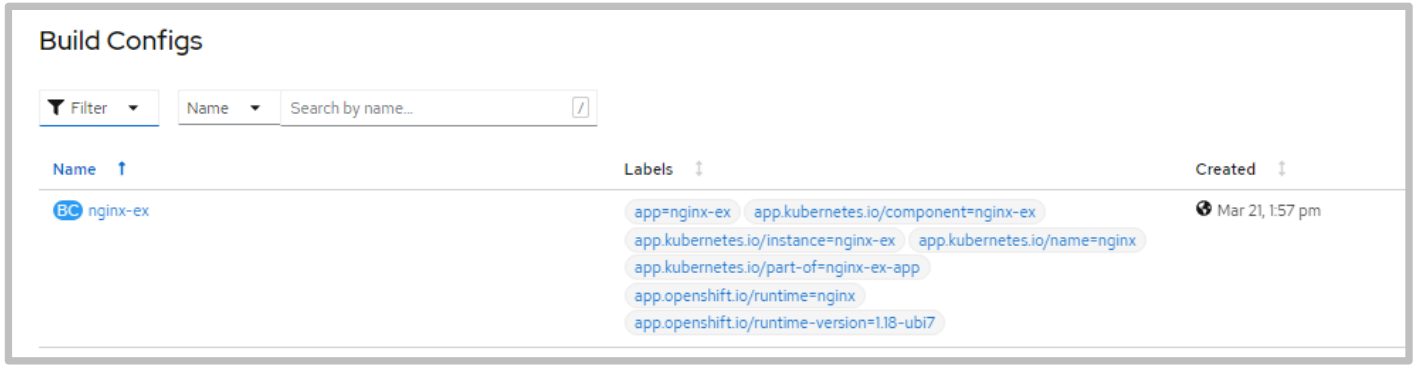


Test and monitor the Application

Configured Resources

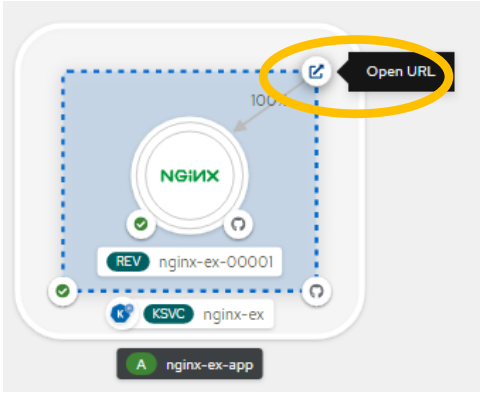
- Route and build have been created
- Currently no pods are running



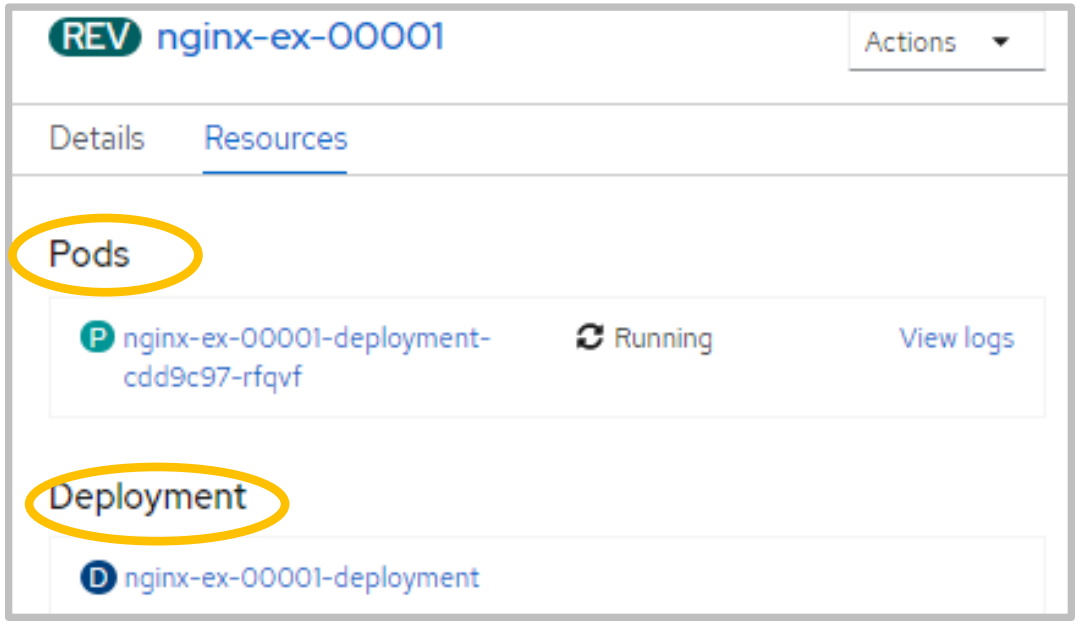


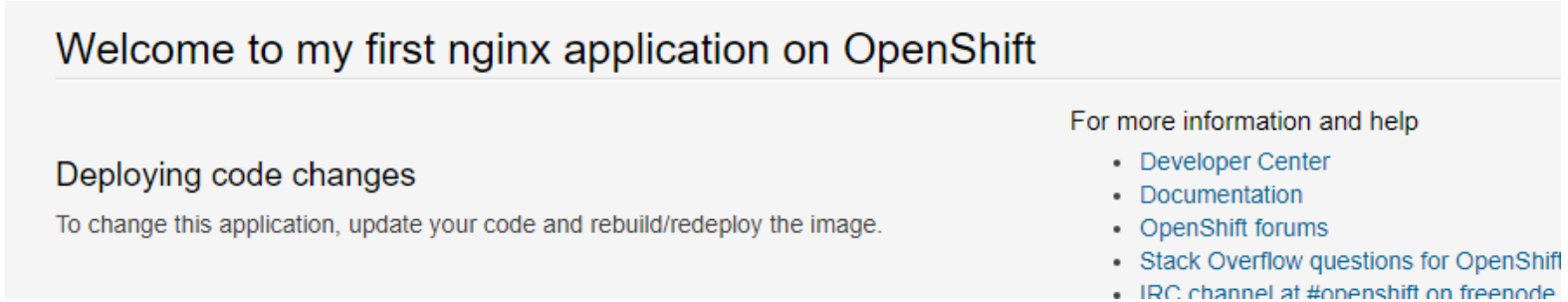
Test and monitor the application

Accessing the Application



- After Clicking **Open URL** the pod is scheduled and started immediately
- The website is reachable
- The application scales down to 0 after a short time of inactivity





Test and monitor the application

view from the command line

While the application is running

oc get po shows the running pod and the completed pod from the build process

```
>oc get po
NAME                                    READY   STATUS      RESTARTS   AGE
nginx-ex-00001-deployment-cdd9c97-7wmtb 2/2     Running     0          40s
nginx-ex-1-build                        0/1     Completed   0          59m
```

Once the application is idle the pods are scaled down, reach status terminating and are finally unscheduled.

```
>oc get po
NAME                                    READY   STATUS        RESTARTS   AGE
nginx-ex-00001-deployment-cdd9c97-7wmtb 0/2     Terminating   0          6m45s
nginx-ex-1-build                        0/1     Completed     0          65m
>oc get po
NAME              READY   STATUS      RESTARTS   AGE
nginx-ex-1-build  0/1     Completed   0          65m
```

Summery

- OpenShift Serverless is a great way to create applications as expected
- Scaling down to 0 helps saving resources and money und scaling up ensures the performance if it is required.
- It easy to handle, even with little experience in OpenShift.
- Using CodeReady workspace a good way to provide test sc in case a cloud based test environment is not feasible. However, CodeReady workspace needs some CPU, Memory and disk resources.