Master Class

Al application evaluation on Intel® hardware through Red Hat® OpenShift® Data Science platform using the Intel® Developer Cloud for the Edge

Authors
Meghana Rao – AI Technology Evangelist, Network and Edge Group, Intel Corporation
Divya Prakash – Software Research Engineer, Network and Edge Group, Intel Corporation
Puneeth Rai – Cloud Software Development Engineer, Network and Edge Group, Intel Corporation

Table of Contents
Part 1: The AI Training to Edge Inference Continuum ................................................................. 3
    Learning Objectives ............................................................................................................. 3
    The AI Lifecycle .................................................................................................................. 3
    Business challenges in the AI training to edge inference continuum ............................ 4
    Solution .................................................................................................................................. 5
        Red Hat® OpenShift® Data Science Platform ............................................................... 5
        Intel® Developer Cloud for the Edge – Container Playground ...................................... 5
Part 2: User Journey - Integrating Red Hat® OpenShift® Data Science with the Intel®
Developer Cloud for the Edge Container Playground .......................................................... 6
    Learning Objectives ............................................................................................................. 6
    User journey ....................................................................................................................... 7
    Pre-Requisites .................................................................................................................... 9
        Resources used in this masterclass ............................................................................. 9
    Steps to Complete Before we Begin .............................................................................. 9
        Upload files to your personal AWS® S3 account ....................................................... 10
Part 3: AI solution inference on Intel® hardware through RHODS ................................................................. 11

Learning Objectives.................................................................................................................................................. 11

Step 1: Start a notebook server from RHODS developer sandbox ............................................................... 11

Step 2: Choose the container deployment size ............................................................................................. 13

Step 3: Add environment variable to access the Intel® Developer Cloud for the Edge Container Playground .................................................................................................................................................. 13

Step 4: Start the Jupyter® Notebook Server ................................................................................................. 15

Step 5: Import the Intel® Developer Cloud for the Edge Jupyter® notebook from the Git repository .................................................................................................................................................. 16

Step 6: Evaluate and benchmark AI inference solution on Intel® hardware ........................................ 17

Try it yourself ......................................................................................................................................................... 24

Learn more about the Intel® Developer Cloud for the Edge Container Playground ........................................... 25

Conclusion ............................................................................................................................................................ 25

Authors ................................................................................................................................................................. 26

Notices and Disclaimers .................................................................................................................................. 27
Part 1: The AI Training to Edge Inference Continuum

Learning Objectives
- Identify the stages in the AI lifecycle
- Identify business challenges in implementing the AI lifecycle
- Contextualize the importance of Red Hat* OpenShift* Data Science and Intel® Developer Cloud for the Edge Container Playground platforms

The AI Lifecycle
The AI lifecycle within any organization can be broadly categorized into eight stages.

Figure 1: The AI Life Cycle

1. **The challenge** – Organizations brainstorm potential opportunities and rank the business value of each.
2. **Approach** – Requires a thorough assessment of the potential solutions. Example – the choice of ML/DL models, identifying whether to build a solution from scratch or start with a foundational framework and apply transfer learning, the metrics of business analytics that would be necessary to extract insights from the solution etc. The chosen solution approach should provide organizations with the right balance between value and simplicity to deliver the best ROI.
3. **Values** – Beyond developmental considerations, a thorough assessment of the legal, social and ethical issues must be studied and applied to ensure that the solution is equitable.
4. **People** – Organizations require internal buy-in to the AI approach of solving problems and building the right cross-functional team to develop solutions.
5. **Technology** – The technological assessment involves evaluation of different computing choices for both training and inference. Based on the identified approach in Step 2, the choice of technology could differ based on needs for training and ultimate deployment.
6. **Data** - One of the biggest barriers to developing an AI solution is the right quality and quantity of data. The process of sourcing, cleaning, labeling, ingesting data into the model and storage is a complex process that is often overlooked. It is also important to emphasize the ethical considerations which were highlighted in Step 3 during the data preparation process.
7. **The AI Model Training** – This stage involves ingesting the data from Step 6, identifying the right statistical or neural network for the task and either training the model from scratch or applying the process of transfer learning to tune hyper-parameters until a desired accuracy (established by the organization/industry) is reached.

8. **Model Deployment** – Once the trained model is ready, it is ready to go into production and deployed in a real-world scenario where real-time data is fed to the pre-trained model to make a prediction. For better manageability and scalability, organizations are considering containerizing the developed solution so it can be deployed seamlessly with an orchestration engine such as OpenShift* Kubernetes* platform.

---

**Business challenges in the AI training to edge inference continuum**

The lifecycle shown in the previous section is clearly a continuum and iterative until desired results are achieved. However, the needs of compute at various stages in the lifecycle are different.

For AI model training, the dataset is acquired and processed (cleaned, labelled, split into training and validation data sets etc.) ahead of time. Data pre-processing and model training can require significant compute, parallelism and performance, especially when the data sets are large, and the models are complex. Questions that should be addressed are:

- Should businesses invest in on-premise hardware or move to the CSPs like AWS*, Azure*, Google Cloud Platform* for more compute intensive tasks?
- Would transfer learning be a better choice than training models from scratch?

Once trained, AI models that are deployed at the Edge – the point closest to where real-time data is being generated – need to be optimized for power, memory and latency without compromising on model accuracy. Organizations will need to further optimize their AI solutions for these constraints and evaluate them on dedicated processors designed for edge use cases. If needs between training and inference at the edge are so different, questions that should be addressed include:

- How can businesses reduce Total Cost of Ownership (TCO) by making the optimal choice of deployment hardware for best results? How do you address the lack of access to a variety of edge hardware to test your applications?
- What toolkits are available for optimizing solutions through software?
- How do businesses improve productivity by choosing the right platforms for development through deployment?
- How can the hardware-software choice paralysis be avoided while ensuring the necessary support for the tools used?
- How do you address the issue of manageability across the cloud to edge continuum?
Solution

Red Hat® OpenShift® Data Science (RHODS) platform and Intel® Developer Cloud for the Edge provide a solution to these questions in the AI training to the edge inference continuum!

The integration of RHODS and the Intel® Developer Cloud for the Edge provides a strong combination of managed cloud platforms built on trusted open source technologies, Integrated Development Environments (IDEs), SDKs and development tools while directly allowing developers to test solutions on some of the most advanced Edge Inference hardware from Intel®. The result – a seamless development and evaluation experience that provides ease of use, improved productivity, is performant and reduces upfront cost investments through flexibility and choice of Intel® hardware for solution evaluation.

Red Hat® OpenShift® Data Science Platform

Red Hat® OpenShift® Data Science is a managed public cloud service that enables development, training, inferencing, optimizing and testing Machine Learning and Deep Learning models in a fully managed environment. With support for Jupyter* and popular frameworks like Tensorflow* and PyTorch* together with operators for the Intel® Distribution of OpenVINO™ Toolkit on a trusted Kubernetes* infrastructure, developers can accelerate the data science process and seamlessly deploy their containers into production.

Learn more about the Red Hat® OpenShift Data Science platform

Intel® Developer Cloud for the Edge – Container Playground

Intel® Developer Cloud for the Edge Container Playground plays a critical role in complementing RHODS offerings by providing access to an extensive suite of Intel® Edge hardware – spanning the latest Intel® Xeon™ and Intel® Core® families of processors and accelerators – for evaluation of AI inferencing solutions. The Container Playground is built on Red Hat® OpenShift® and provides a cloud-desktop environment to develop cloud-native solutions by leveraging robust workload orchestration provided by the OpenShift® Platform. It is also backed by the improved security and configuration best practices offered by OpenShift® protecting application workloads at runtime.

The Container Playground supports Bring-Your-Own-Container, Bring-Your-Own-Source and Bring-Your-Own-Data pathways to build containers in a Kubernetes* environment. Users can import pre-built containers from popular container registries like AWS®, Azure® Red Hat® Quay.io, RHODS etc.; source code hosted on Git* repos or data from cloud storage into the local file system to build containers and evaluate them on Intel® hardware to obtain hyper-customized telemetry data, thus enhancing confidence on ultimate AI solution deployment performance.

The workflow below demonstrates how the Container Playground works in a browser environment.
In this master class, we explore the power of integrating RHODS with the Intel® Developer Cloud for the Edge Container Playground and show how developers can use these two platforms together to seamlessly develop AI solutions while benchmarking them on Intel® edge hardware.

**Part 2: User Journey - Integrating Red Hat* OpenShift* Data Science with the Intel® Developer Cloud for the Edge Container Playground**

**Learning Objectives**

- Comprehend the user journey from RHODS to Intel® Developer Cloud for the Edge Container Playground using a kidney tumor segmentation model example
- Set up accounts to access RHODS and Intel® Developer Cloud for the Edge
- Access the pre-requisite resources to integrate RHODS and Intel® Developer Cloud for the Edge
User journey
This tutorial demonstrates a Kidney Tumor Segmentation model example using the KiTS19* dataset. There are two components to the user journey for this AI training to inference solution.

1. **AI model training** uses Red Hat® OpenShift® Data Science platform. Developers spin up a Jupyter® notebook with the relevant Tensorflow® or PyTorch® kernel, clone the training sample notebook (ipynb file) from the source code repository and using the dataset of choice, train the model and upload the trained model to a storage facility of your choice. Learn more about how to use the Red Hat® OpenShift® Data Science through the “Launch Red Hat OpenShift Data Science” Learning Path.

   For the chosen example in this tutorial, we will assume developers have completed this part of the journey and uploaded the trained PyTorch® kidney segmentation model to the AWS® S3 bucket. For convenience, we provide pre-trained models for this exercise in the form of OpenVINO™ Intermediate Representation (IR) files. See pre-requisites section for instructions.

2. **AI model inference** on different Intel® hardware leverages the Intel® Developer Cloud for the Edge Container Playground. The inference shows kidney tumor segmentation using PyTorch® Lightning and the OpenVINO™ Toolkit and performs benchmarks on different Intel® processors available through the Container Playground. For convenience, we will provide a Jupyter® notebook showing the steps needed to invoke the Container Playground from within RHODS and to evaluate your solution on Intel® hardware. See pre-requisites section for instructions.

   The tutorial aims to show how RHODS and Intel® Developer Cloud for the Edge provide a complementary value proposition to organizations and developers in the process of accelerating their data science workflow while providing the ability to evaluate solutions on Intel® Edge Processors and accelerators before deployment into production. Below workflow shows the user journey and the handshake between the RHODS platform and the Container Playground platform to accelerate your AI lifecycle process.
Through this master class, you will learn how to access Intel® edge hardware from within RHODS using Intel® Developer Cloud for the Edge Container Playground ReST APIs and test your solutions without having to navigate away from the RHODS platform. Once completed, you will be able to use any of the OpenVINO™ notebooks from RHODS and test them on Intel® hardware through the Container Playground.

Let’s get started.
Pre-Requisites

Resources used in this masterclass

The solution we will demonstrate in this master class will leverage open source resources listed below:

1. A sample Jupyter notebook (made available under the Apache 2.0 License) that demonstrates the Intel® Developer Cloud for the Edge Container Playground ReST APIs which we will use from the Red Hat® OpenShift® Data Science platform. This notebook leverages:
2. The following...
   a. The Intel® Developer Cloud for the Edge Container Playground as a Python library – Available for download from PyPi.
   b. Pre-trained models for the Intel® OpenVINO™ Toolkit sample application for kidney segmentation with PyTorch* Lightning. These files are made available in the OpenVINO™ Intermediate Representation format - unet_kits19.bin, unet_kits19.xml and unet_kits19.mapping. Alternatively, you can also obtain these files by enabling the OpenVINO operator through RHODS. Learn more by launching the "How to get started with Intel OpenVINO" learning path.
   c. Startup scripts – run.sh and benchmark.sh (made available under the Apache 2.0 License) facilitate evaluation and benchmarking of the AI model on Intel® hardware. The benchmark.sh script leverages the OpenVINO™ Benchmark App.

Steps to Complete Before we Begin

Create accounts on RHODS and the Intel® Developer Cloud for the Edge

   a. Create an account on RHODS to access JupyterLab*-as-a-service and train an AI model.
b. [Create account on the Intel® Developer Cloud for the Edge](https://www.intel.com/content/www/us/en/developer/tools/devcloud/edge/overview.html) to access the Container Playground from inside RHODS and evaluate solutions on Intel® hardware and benchmark using the OpenVINO™ Toolkit.

![Intel® Developer Cloud for the Edge](intel.com/content/www/us/en/developer/tools/devcloud/edge/overview.html)

**Upload files to your personal AWS* S3 account**

It is assumed that users have completed the training process and have pre-trained model files ready for evaluation. To evaluate trained AI models for inference performance on Intel® hardware, these files should first be stored in cloud storage, such as Amazon* S3. The Intel® Developer Cloud for the Edge ReST APIs used within RHODS pull these trained models from the S3 bucket into the Container Playground filesystem using the [Cloud Connector capability](https://www.intel.com/content/www/us/en/developer/tools/devcloud/edge/overview.html) and makes them available for evaluation on Intel® processors.

Pre-requisites listed above provide a starting point to understand this workflow. To proceed, create an AWS* S3 account and upload files listed in steps 2b and 2c. You will need to have your AWS S3 region, bucket name, access and secret keys handy to successfully complete the exercise.
Part 3: AI solution inference on Intel® hardware through RHODS

Learning Objectives

- Access Intel® Developer Cloud for the Edge Container Playground from the RHODS developer sandbox
- Use Intel® Developer Cloud for the Edge ReST APIs to build containers, evaluate and benchmark on Intel® processors within RHODS

The Intel® Developer Cloud for the Edge Container Playground, available as a Python* library can be easily leveraged from IDEs like Jupyter* notebook. This feature makes it extremely simple and easy to access Intel® hardware, build containers, evaluate and benchmark them through the Container Playground, without having to navigate away from RHODS.

Step 1: Start a notebook server from RHODS developer sandbox

a. Login to Red Hat OpenShift Data Science using credentials created in Part 2.

b. Log into the DevSandbox.
c. Launch the Jupyter notebook server by clicking on “Launch Application”.

d. For the inference workflow, choose either a Standard Data Science Python 3.8 kernel or the OpenVINO™ Toolkit 2022.1 Python 3.8.6 kernel.
Step 2: Choose the container deployment size
For this tutorial, a default small container size suffices. If your needs exceed the default size, choose a larger container size accordingly.

Deployment size

Step 3: Add environment variable to access the Intel® Developer Cloud for the Edge Container Playground

a. Login to the Intel® Developer Cloud for the Edge using account credentials created in Chapter 2. Copy the API token from the user profile.
b. In the RHODS platform, create an environment variable `DEVCL OUD_TOKEN` and paste the token copied from Step 3a. The token is specific to each session.

Environment variables

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVCL OUD_TOKEN</td>
<td>[Token value]</td>
</tr>
</tbody>
</table>

- **Variable value:** [Token value]
- **Secret:** Yes
Step 4: Start the Jupyter* Notebook Server

Start server  
Cancel

Start server in current tab

The OAuth authentication takes a few minutes to complete and launches a fresh Jupyter* notebook server with the requested resources.

Open Jupyter* in a new tab.
Step 5: Import the Intel® Developer Cloud for the Edge Jupyter® notebook from the Git repository

The devcloud.ipynb Jupyter® notebook shows how to use the ReST APIs with the kidney segmentation sample application and access Intel® hardware from within RHODS.

a. Navigate to the Intel® Developer Cloud for the Edge samples Git repo and copy the Git URL to access the training-to-inference Jupyter® notebook and required script files.

b. Enter this URL in the Git Clone tab within the RHODS platform.
c. Open the devcloud.ipynb notebook from the training-to-inference folder.

![Image of devcloud.ipynb notebook]

**Step 6: Evaluate and benchmark AI inference solution on Intel® hardware**

The steps below will demonstrate the Intel® Developer Cloud for the Edge ReST APIs in action. Without having to navigate away from the RHODS platform, users can now access Intel® hardware for inference performance evaluation and benchmarking. While this master class walks through the APIs in the Jupyter® notebook, users starting from a fresh notebook can also copy and paste the code from this tutorial to complete this exercise.

**a. Install the Developer Cloud for the Edge Container Playground library**

Install the Intel® Developer Cloud for the Edge Python® library from PyPi®. Users can then use the ReST APIs to access Intel® hardware for solution evaluation and benchmarking using the Container Playground directly through RHODS. Refer to the API documentation on PyPi® for any updates.

```
%pip install devcloud
from devcloud import Devcloud

Successfully installed devcloud-0.0.2 importlib-resources-5.10.0 typer-0.7.0
```
b. Connect to the Intel® Developer Cloud Container Playground

Devcloud.connect()

When the connection is complete, you will see the list of projects that you have created in the Container Playground. If this is the first time you have accessed the Container Playground, this list will be empty.

<table>
<thead>
<tr>
<th>Sr.No.</th>
<th>project_id</th>
<th>Name</th>
<th>Create Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>536245</td>
<td>ovms test</td>
<td>1668538307729</td>
</tr>
<tr>
<td>2</td>
<td>536231</td>
<td>nginx</td>
<td>1668537774098</td>
</tr>
</tbody>
</table>


c. Transfer the trained model from the AWS* S3 bucket into the Container Playground filesystem

The Container Playground supports the Cloud-Connector feature to enable file sharing between the Container Playground file system and cloud storage such as AWS* S3 bucket. This step demonstrates the API to transfer the pre-trained model in the form of OpenVINO™ Intermediate Representation files (unet_kits19.xml, unet_kits.bin and unet_kits19.mapping) and the startup script (run.sh) from the AWS* S3 bucket into the Container Playground file system. Set AWS_REGION_NAME to the region of your S3 bucket. Set AWS_BUCKET_NAME to point to the bucket containing the required files. You will need to have handy, the AWS Access and Secret Keys for your account to complete this step.

AWS_REGION_NAME = "user-region"
AWS_BUCKET_NAME = "user-bucket"
Devcloud.transfer(AWS_REGION_NAME, AWS_BUCKET_NAME, ["unet_kits19.mapping", "unet_kits19.xml", "unet_kits19.bin", "run.sh"])

```python
AWS_REGION_NAME = "user-region"
AWS_BUCKET_NAME = "user-bucket"
Devcloud.transfer(AWS_REGION_NAME, AWS_BUCKET_NAME, ["unet_kits19.mapping", "unet_kits19.xml", "unet_kits19.bin", "run.sh"])
```
d. **List all Intel® Processors available for testing on the Container Playground**

The "Devcloud.availableHardware()" API is used to list all the Intel® edge processors available through the Container Playground. The list shows the processor name, the processor ID, lists GPUs (where available) and the memory capacity. Users can choose from one or more Intel® hardware from the below list to run inference on using the processor ID.

```java
Devcloud.availableHardware()
```

The sample output is shown here:

<table>
<thead>
<tr>
<th></th>
<th>Processor Name</th>
<th>GPU</th>
<th>Memory Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intel® Xeon® D-2712T Processor</td>
<td>None</td>
<td>64GB</td>
</tr>
<tr>
<td>2</td>
<td>Intel® Xeon® Platinum-8357B Processor</td>
<td>None</td>
<td>64GB</td>
</tr>
<tr>
<td>3</td>
<td>Intel® Gen13 Core™ 17-11700E Processor</td>
<td>Intel® UHD Graphics 77</td>
<td>32GB</td>
</tr>
<tr>
<td>4</td>
<td>Intel® Gen13 Core™ 19-11980E Processor</td>
<td>Intel® UHD Graphics 77</td>
<td>32GB</td>
</tr>
<tr>
<td>5</td>
<td>Intel® Gen11 Core™ 17-11850E Processor</td>
<td>Intel® Iris® Xe Graphics</td>
<td>16 GB</td>
</tr>
<tr>
<td>6</td>
<td>Intel® Gen9 Core™ 17-9750H Processor</td>
<td>Intel® UHD Graphics 630</td>
<td>64 GB</td>
</tr>
<tr>
<td>7</td>
<td>Intel® Gen11 Core™ 17-11700 Processor</td>
<td>Intel® UHD Graphics 750</td>
<td>32 GB</td>
</tr>
<tr>
<td>8</td>
<td>Intel® Gen8 Core™ 13-8109U Processor</td>
<td>Intel® Iris® Plus Graphics 655</td>
<td>16 GB</td>
</tr>
<tr>
<td>9</td>
<td>Intel® Gen8 Core™ 13-8149U Processor</td>
<td>Intel® UHD Graphics 620</td>
<td>16 GB</td>
</tr>
<tr>
<td>10</td>
<td>Intel® Gen8 Core™ 15-8855UE Processor</td>
<td>Intel® UHD Graphics 620</td>
<td>16 GB</td>
</tr>
<tr>
<td>11</td>
<td>Intel® Gen10 Core™ 17-10700 Processor</td>
<td>Intel® UHD Graphics 630</td>
<td>32 GB</td>
</tr>
<tr>
<td>12</td>
<td>Intel® Xeon® D-2790MT Processor</td>
<td>None</td>
<td>64GB</td>
</tr>
<tr>
<td>13</td>
<td>Intel® Xeon® W-1290T Processor</td>
<td>None</td>
<td>16 GB</td>
</tr>
<tr>
<td>14</td>
<td>11th Generation Intel® Core™ 17 Processor</td>
<td>Intel® UHD Graphics 750</td>
<td>32GB</td>
</tr>
<tr>
<td>15</td>
<td>12th Generation Intel® Core™ 17 Processor</td>
<td>Intel® UHD Graphics 770</td>
<td>32GB</td>
</tr>
<tr>
<td>16</td>
<td>Intel® Xeon® W-1390 Processor</td>
<td>None</td>
<td>32GB</td>
</tr>
<tr>
<td>17</td>
<td>Intel® Xeon® Gold-6221U Processor</td>
<td>None</td>
<td>64 GB</td>
</tr>
<tr>
<td>18</td>
<td>12th Generation Intel® Core™ 19 Processor</td>
<td>Intel® UHD Graphics 770</td>
<td>32GB</td>
</tr>
<tr>
<td>19</td>
<td>11th Generation Intel® Core™ 19 Processor</td>
<td>Intel® UHD Graphics 750</td>
<td>32GB</td>
</tr>
</tbody>
</table>

**Check out the complete list of processors currently on the Container Playground.**

Newer processors are added to the Container Playground at regular intervals. Learn more about the types of [use cases supported by each category of processors](#).
e. **Create a container**
Create a container with a base image URL and assign it to a project.

```python
PROJECT_NAME="kidney-segmentation"
CONTAINER_NAME="openvino-kidney-segmentation"
Devcloud.createContainer(PROJECT_NAME, CONTAINER_NAME, "docker.io/openvino/openvino_tensorflow_ubuntu20_runtime")
```

f. **Configure the container**
Developers have the flexibility to configure containers to include additional parameters like port, enabling routes, adding entry points, output mount points etc.

For this example, we set the entry point to the run.sh script and the output mount point to the Developer Cloud for the Edge file system from where you can access the output files (“/mount_folder”).

```python
Devcloud.configureContainer(PROJECT_NAME, CONTAINER_NAME, [], [], "/s3/run.sh", "/mount_folder", 
[('/s3/' + AWS_BUCKET_NAME, "/s3")], ")
```

Learn more about [configuring containers](#).

g. **Choose Intel® Processor to launch the container and run inference on**
Based on the guidance in Step 6d, identify the processor that is optimal for your use case and choose the device ID corresponding to the chosen processor.

In this example, we chose device ID 18 which corresponds to 11th Generation Intel® Xeon® W-1390 Processor.

```python
Devcloud.launch(PROJECT_NAME, 18)
```
h. **Query device status**

You can query the container status at any point through the Jupyter notebook on RHODS without having to navigate to the Container Playground interface. The container is put in the queue for execution on the chosen edge node. The container goes from "Waiting" into "Deployed" and then "Running" state when the selected edge node becomes available for execution.

**Note** It takes approximately 2 minutes for the workload to be deployed on the edge node. Query the status of the container and ensure it is in "Terminated" state to read the throughput and latency for this inference.

```
Devcloud.getStatus(projectName=PROJECT_NAME, output="wide")
```

Sample output demonstrating the FPS and Throughput for this workload on the chosen Intel® processor:

```
<table>
<thead>
<tr>
<th>project</th>
<th>target</th>
<th>status</th>
<th>execution time</th>
<th>create time</th>
<th>fps</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>kidney-segmentation</td>
<td>Xeon Gold-6212U</td>
<td>TERMINATED</td>
<td>206</td>
<td>2023-08-01 16:31:06</td>
<td>28.18</td>
<td>2.45</td>
</tr>
</tbody>
</table>
```

i. **View inference results on the chosen Intel® processor**

When inference execution completes, several inference images are generated that show the segmentation of the kidney tumor. You can view any of these images in the notebook by providing the output file path. The code below shows the default output file path from the Container Playground filesystem for this example.

**Note** on accessing the output file path through the Container Playground interface:
Login to the Container Playground. Go to My Library -> Filesystem -> Project Name -> Hardware name as show =n below and copy the file path from the Inference folder.
Provide this path as input to the "Devcloud.getFilesPreview" API

```python
%pip install Pillow
import IPython.display
import PIL.Image
from PIL import Image
from io import StringIO

image_bytes = Devcloud.getFilesPreview(PROJECT_NAME, path="output/inference/24_inference.jpeg")
img = PIL.Image.open(image_bytes)
IPython.display.display(img)
```

The image output demonstrates the kidney tumor segmentation.

![Kidney Tumor Segmentation Image](image.png)

j. **Benchmark the container on Intel® Processor with the OpenVINO™ toolkit**

   Benchmark the AI model using the benchmark app of the OpenVINO™ toolkit hosted on DockerHub. When prompted, you will need to provide the AWS Access and Secret Keys to transfer the required pre-trained model files into the Intel® Developer Cloud for the Edge Container Playground file system.

   ```python
   BENCHMARK_PROJECT = "kidney-segmentation-benchmark"
   BENCHMARK_CONTAINER = "container-kidney-segmentation-benchmark"
   
   Devcloud.transfer(AWS_REGION_NAME,AWS_BUCKET_NAME, ["unet_kits19.mapping","unet_kits19.xml ","unet_kits19.bin","run.sh", "benchmark_run.sh")
   
   Devcloud.createContainer(BENCHMARK_PROJECT, BENCHMARK_CONTAINER, "docker.io/openvino/ubuntu20_dev")
   ```

   This step successfully creates a container and assigns it to the designated project.
k. **Configure the container and launch**
Provide the entry point and the output mount point to the container.
```
Devcloud.configureContainer(BENCHMARK_PROJECT, BENCHMARK_CONTAINER, [], [], 
"/s3/benchmark_run.sh", "/mount_folder", 
["/s3/" + AWS_BUCKET_NAME, "/s3"], ")
```
```
Devcloud.launch(BENCHMARK_PROJECT, 18)
```

l. **View benchmark output**
Prove status to ensure project status is "Terminated"
```
Devcloud.getStatus(projectName= BENCHMARK_PROJECT, output="wide")
```

Copy the benchmark_report.csv file path from the Container Playground filesystem

Provide this path as the input to “Devcloud.GetFilesPreview” API to view detailed output of the inference to view detailed benchmarking results.
```
res = Devcloud.getFilesPreview(BENCHMARK_PROJECT, path="benchmark_report.csv")
```
```
import csv
decoded_content = res.read().decode('utf-8')

my_list = list(cr)
for row in my_list:
    print(row)
```
A snapshot of the benchmarking report is shown below:

```json
['Command line parameters']
['-e', '/s3/unet_kits19.xml']
[-report_folder', '/mount_folder']
[-report_type', 'detailed_counts']
[]
['Configuration setup']
['topology', 'Model1']
['target device', 'CPU']
['API', 'async']
['inference_only', 'True']
['precision', 'UNSPECIFIED']
['batch size', '1']
['number of iterations', 'None']
['number of parallel infer requests', '12']
['duration (ms)', '60000']
['number of CPU streams', '12']
[]
['Execution results']
['read network time (ms)', '23.72']
['load network time (ms)', '171.89']
['create infer requests time (ms)', '0.03']
['first inference time (ms)', '115.56']
['total execution time (ms)', '60842.71']
['total number of iterations', '1224']
['latency (ms)', '591.31']
['avg latency', '505.97']
['min latency', '342.39']
['max latency', '876.91']
['throughput', '20.00']
[]
```

**Try it yourself**

Now that you have learned how to use the Intel® Developer Cloud for the Edge APIs to evaluate a sample AI solution on one of the available Intel® edge nodes, try:

- Picking a different Intel® processor to evaluate and benchmark results on and compare results
- Bring your own pre-trained models into RHODS and build and evaluate it as a container solution on Intel® hardware
- Launch OpenVINO™ notebooks from RHODS and test on Intel® hardware using the ReST APIs learned in this class. Learn more by launching the “How to Get Started with Intel OpenVINO” learning path
Learn more about the Intel® Developer Cloud for the Edge Container Playground

The Container Playground Powered by Red Hat* OpenShift* offers many more workflows for cloud-native solution development. Explore. Check out the Developer Guide for a detailed walk-through to learn how to:

- Evaluate No-Code/Low-Code solutions from the Intel® Developer Cloud for the Edge Marketplace
- Bring-Your-Own-Containers from popular registries for evaluation on Intel® hardware
- Evaluate containers using HELM charts and Docker Compose
- Bring-Your-Own-Code from Git repositories and build containers
- Bring-Your-Own-Data and models through the Cloud Connector and build containers
- Configure multi-container solutions
- Develop containers in JupyterLab

Try out these workflows on the Intel® Developer Cloud for the Edge Container Playground today!

Conclusion

The AI solution development journey is fraught with many business and developmental challenges. The integrated solution demonstrated in this master class aims at explaining and overcoming these challenges using the cumulative offerings of the Red Hat* OpenShift* Data Science platform and the Intel® Developer Cloud for the Edge Container Playground. Developers can now leverage Intel® hardware, right from within the RHODS platform and complete the journey of data processing, model training and evaluation of containerized AI solutions on real-world, cutting-edge Intel® processors thereby accelerating the data science development and deployment life cycle.
Authors

Meghana Rao (AI Technology Evangelist)

Meghana Rao is a passionate technology evangelist with more than 18 years of experience in the semiconductor industry during which time, she has held diverse roles in software development and ecosystem enabling. She started her career at Intel in 2011 as a lead technology advocate on developer programs in the AI/IoT domains. In her current role, she educates customers, students and developers worldwide on Intel's IoT, Network and Edge AI portfolio of products. She is also a speaker at major conferences and an author.

Meghana has a Bachelor's degree in Computer Science and Engineering from Bangalore University and a Master's degree in Engineering and Technology Management from Portland State University. Before joining Intel, she was a Senior Software Engineer at Infineon Technologies India Pvt. Ltd where she was responsible for embedded software development for VoIP and Image/video products.

Divya Prakash (Product Manager)

Divya brings 16 of years of industry experience spanning development of embedded Linux drivers, leading the automotive vehicle blade orchestration team. She is currently the Product Manager of the Intel® Developer Cloud for the Edge.

An engineer at heart with a product mindset, she brings a unique perspective to the technology landscape. With deep understanding of containerization and orchestration, she has successfully led cross-functional teams in delivering innovative products that leverage the power of Kubernetes at the edge.
Puneeth Rai (Cloud Software Development Engineer)

Puneeth is a lead cloud stack engineer at Intel with over 11 years of experience in the cloud and telecommunication industry. In the past 1 year at Intel, he is working on Intel® Developer Cloud for the Edge platform helping customers with deploying their solution on the cloud and finding the right Intel hardware for their workload.

Puneeth has a Bachelor’s degree in Telecommunication Engineering from Visvesvaraya Technological University and a Master’s degree in Data Science and Engineering from Birla Institute Of Technology And Science, Pilani. Prior to joining Intel, he was an Architect at Mavenir working on cloud native solution for 5G and 4G related technologies along with VoIP solutions. His hobbies include farming, travelling and photography.

Notices and Disclaimers

Intel® technologies may require enabled hardware, software, or service activation.

No product or component can be absolutely secure.

Performance varies by use, configuration and other factors.

Your costs and results may vary.

Intel is committed to respecting human rights and avoiding complicity in human rights abuses. See Intel’s Global Human Rights Principles. Intel® products and software are intended only to be used in applications that do not cause or contribute to a violation of an internationally recognized human right.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein. No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document. Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Users of the solutions herein are solely responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.