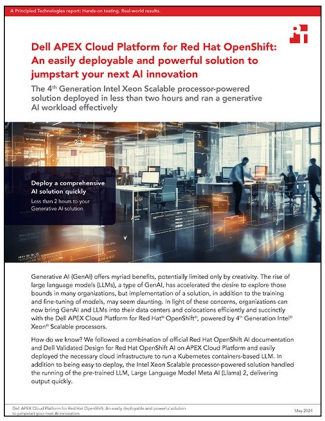

The science behind the report:

Dell APEX Cloud Platform for Red Hat OpenShift: An easily deployable and powerful solution to jumpstart your next AI innovation

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Dell APEX Cloud Platform for Red Hat OpenShift: An easily deployable and powerful solution to jumpstart your next AI innovation.

We concluded our hands-on testing on April 11, 2024. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on March 3, 2024 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to http://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 2: The steps we completed to deploy our Dell APEX Cloud Platform for Red Hat OpenShift solution and Llama 2 and the time we needed to complete each step. Source: Principled Technologies.

Step	Time (h:mm:ss)
Create an admin login for OpenShift	0:03:37
Install OpenShift AI	0:03:42
Create Data Science Project	0:01:32
Install both KServe and OpenShift Serverless	0:02:12
Download Llama 2 model	0:45:31
Upload Llama 2 model to cloud storage	0:19:49
Create a workspace with Llama 2 runtime	0:14:34
Deploy Redis	0:13:02
Create index and populate it	0:01:20
Create Gradio deployment	0:05:52
Total	**1:51:11**

System configuration information

Our solution consisted of four Dell APEX MC-760 servers configured as a Red Hat OpenShift cluster. We configured all four as worker servers. Three of the four servers operated as management servers as well.

How we tested

Completing the initial configuration and installing OpenShift AI

Following the Red Hat guide for installing OpenShift AI, we added our user logins to the Dell APEX cluster configured with OpenShift. We used Oauth authentication, but you can also follow the process in this document: https://access.redhat.com/documentation/en-us/openshift_container_platform/4.15/html/authentication_and_authorization/configuring-internal-oauth).

Then, we used the official Red Hat OpenShift documentation to install the prerequisites and OpenShift AI (refer to the OpenShift documentation here: https://access.redhat.com/documentation/en-us/red_hat_openshift_ai_self-managed/2.8/html/serving_models/serving-large-models_serving-large-models). We made better success with the automated installation for the version of OpenShift AI we used, but Red Hat recommends using a manual installation for the Dell APEX platform. This is because Dell installs the ServiceMeshControlPlane and KNativeServing resources on APEX with OpenShift. We then followed the steps in https://access.redhat.com/documentation/en-us/red_hat_openshift_ai_self-managed/2.8/html/installing_and_uninstalling_openshift_ai_self-managed/enabling-gpu-support_install) to enable GPU support in our installation before continuing to the next segment.

Adding a model

We shifted to using vLLM for our model serving runtime because we encountered difficuty with the provided OpenShift model serving runtimes. Following the instructions here in the document https://access.redhat.com/documentation/en-us/red_hat_openshift_ai_self-managed/2-latest/html/serving_models/serving-large-models_serving-large-models#adding-a-custom-model-serving-runtime-for-the-single-model-serving-platform_serving-large-models, we added vLLM to OpenShift AI. After our description of how we tested, we provide the yaml files that we used.

After we added vLLM as a model serving runtime, we downloaded the Llama 2 model from Hugging Face, which you can find at https://huggingface.co/meta-llama/Llama-2-13b-chat-hf, and uploaded the model to our local S3 storage. Doing so allowed us to enable it for use by our digital assistant. You can find those steps at https://access.redhat.com/documentation/en-us/red_hat_openshift_ai_self-managed/2-latest/html/serving_models/serving-large-models_serving-large-models#deploying-models-on-the-single-model-serving-platform_serving-large-models.

Installing Redis

To install a Redis Enterprise database for our digital assistant, we followed Dell documentation (available either in the Dell Validated Document or their Github page https://github.com/DellBizApps/dell-digital-assistant/blob/main/03-redis/README.md). To summarize the documentation, we installed the Redis Enterprise Operator from the Operator Hub and added the users and permissions that allow Redis to change our OpenShift deployment. Then we could create the Redis database that we needed to serve as our document repository. After the database was running, we used the Dell steps at https://github.com/DellBizApps/dell-digital-assistant/blob/main/04-data-processing/notebooks/01-Redis-Ingest-HTML-from-web.ipynb to add 32 web pages to our database to serve as reference for the LLM when answering our questions.

Installing Gradio

To install Gradio, we used the Dell methodology as a base (for more info refer to https://github.com/DellBizApps/dell-digital-assistant/tree/main/05-UI/Gradio/caikit-rag-redis), but we needed a different set of files to make the deployment work with our changes to the OpenShift AI deployment. To install Gradio, we modified the yaml files according to the Dell recommendations and deployed them. This gave us a text GUI that enabled us to talk to the LLM backend, using the Redis database as a document repository that the LLM could reference for custom information.

Custom yaml files

We used the following yaml files in our deployment.

vLLM yaml file

```
apiVersion: serving.kserve.io/v1alpha1
kind: ServingRuntime
labels:
  opendatahub.io/dashboard: "true"
metadata:
  annotations:
    openshift.io/display-name: vLLM
    opendatahub.io/recommended-accelerators: '["nvidia.com/gpu"]'
  name: vllm
spec:
  builtInAdapter:
    modelLoadingTimeoutMillis: 90000
  containers:
    - args:
        - --model
        - /mnt/models/
        - --download-dir
        - /models-cache
        - --port
        - "8080"
      image: quay.io/rh-aiservices-bu/vllm-openai-ubi9:0.4.0
      name: kserve-container
      ports:
        - containerPort: 8080
          name: http1
          protocol: TCP
  multiModel: false
  supportedModelFormats:
    - autoSelect: true
      name: pytorch
```

Gradio yaml files

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: glrv
  labels:
    app: glrv
spec:
  replicas: 0
  selector:
    matchLabels:
      app: glrv
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: glrv
    spec:
      restartPolicy: Always
      schedulerName: default-scheduler
      affinity: {}
      terminationGracePeriodSeconds: 120
      securityContext: {}
      containers:
        - resources:
            limits:
              cpu: '16'
              memory: 64Gi
            requests:
              cpu: '12'
```

```
            memory: 48Gi
          readinessProbe:
            httpGet:
              path: /queue/status
              port: http
              scheme: HTTP
            timeoutSeconds: 5
            periodSeconds: 30
            successThreshold: 1
            failureThreshold: 3
          terminationMessagePath: /dev/termination-log
          name: server
          livenessProbe:
            httpGet:
              path: /queue/status
              port: http
              scheme: HTTP
            timeoutSeconds: 8
            periodSeconds: 100
            successThreshold: 1
            failureThreshold: 3
          env:
            - name: APP_TITLE
              value: 'Talk with your documentation'
            - name: INFERENCE_SERVER_URL
              value: 'https://llama2-model-testing.apps.acp1.tme.apexeng.labs.dell/v1'
            - name: DB_CONNECTION_STRING
              value: 'redis://default:Password1!@172.30.18.162:16707'
            - name: DB_COLLECTION_NAME
              value: 'dellwebdocs'
            - name: MAX_NEW_TOKENS
              value: '512'
            - name: TOP_P
              value: '0.95'
            - name: TYPICAL_P
              value: '0.95'
            - name: TEMPERATURE
              value: '0.01'
            - name: REPETITION_PENALTY
              value: '1.03'
          securityContext:
            capabilities:
              drop:
                - ALL
            runAsNonRoot: true
            allowPrivilegeEscalation: false
            seccompProfile:
              type: RuntimeDefault
          ports:
            - name: http
              containerPort: 7860
              protocol: TCP
          imagePullPolicy: IfNotPresent
          startupProbe:
            httpGet:
              path: /queue/status
              port: http
              scheme: HTTP
            timeoutSeconds: 1
            periodSeconds: 30
            successThreshold: 1
            failureThreshold: 24
          terminationMessagePolicy: File
          image: 'quay.io/dellbizapps/ai/glrv:0.1'
      dnsPolicy: ClusterFirst
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 25%
      maxSurge: 1
  revisionHistoryLimit: 10
  progressDeadlineSeconds: 600
```

route.yaml

```
kind: Route
apiVersion: route.openshift.io/v1
metadata:
  name: glrv
  labels:
    app: glrv
spec:
  to:
    kind: Service
    name: glrv
    weight: 100
  port:
    targetPort: http
  tls:
    termination: edge
  wildcardPolicy: None
```

svc.yaml

```
kind: Service
apiVersion: v1
metadata:
  name: glrv
  labels:
    app: glrv
spec:
  clusterIP: None
  ipFamilies:
    - IPv4
  ports:
    - name: http
      protocol: TCP
      port: 7860
      targetPort: http
  type: ClusterIP
  ipFamilyPolicy: SingleStack
  sessionAffinity: None
  selector:
    app: glrv
```

redis_schema.yaml

```
text:
- name: source
  no_index: false
  no_stem: false
  sortable: false
  weight: 1
  withsuffixtrie: false
- name: content
  no_index: false
  no_stem: false
  sortable: false
  weight: 1
  withsuffixtrie: false
vector:
- algorithm: FLAT
  block_size: 1000
  datatype: FLOAT32
  dims: 768
  distance_metric: COSINE
  initial_cap: 20000
  name: content_vector
```

Read the report at https://facts.pt/u1GfRQh

This project was commissioned by Dell Technologies.

Principled Technologies®

Facts matter.®

Principled Technologies is a registered trademark of Principled Technologies, Inc.
All other product names are the trademarks of their respective owners.

DISCLAIMER OF WARRANTIES; LIMITATION OF LIABILITY:
Principled Technologies, Inc. has made reasonable efforts to ensure the accuracy and validity of its testing, however, Principled Technologies, Inc. specifically disclaims any warranty, expressed or implied, relating to the test results and analysis, their accuracy, completeness or quality, including any implied warranty of fitness for any particular purpose. All persons or entities relying on the results of any testing do so at their own risk, and agree that Principled Technologies, Inc., its employees and its subcontractors shall have no liability whatsoever from any claim of loss or damage on account of any alleged error or defect in any testing procedure or result.

In no event shall Principled Technologies, Inc. be liable for indirect, special, incidental, or consequential damages in connection with its testing, even if advised of the possibility of such damages. In no event shall Principled Technologies, Inc.'s liability, including for direct damages, exceed the amounts paid in connection with Principled Technologies, Inc.'s testing. Customer's sole and exclusive remedies are as set forth herein.