Install Notes

OSC AI/ML Framework (Release H)
Implementation Notes for Kserve Adapter
Table of Contents

1. Hardware requirements .................................................................................................................. 2
2. Context Diagram ............................................................................................................................. 2
3. Sequence Diagram .......................................................................................................................... 3
4. Install Kserve/golang/chartmuseum on the Near-RT RIC ............................................................. 3
   4-1. Install Kserve .............................................................................................................................. 3
   4-2. Install golang .............................................................................................................................. 3
         4-2-1. method 1 : Use ubuntu’s package repository to install golang .............................................. 3
         4-2-2. method 2 : Installing Go via the Latest Binary Release ...................................................... 4
   4-3. Install chartmuseum ................................................................................................................. 4
5. Run rcmdms on the Near-RT RIC .................................................................................................. 6
   5-1. Install go-swagger first .............................................................................................................. 6
   5-2. Run rcmdms ............................................................................................................................... 7
6. Run kserve-adapter on the Near-RT RIC ....................................................................................... 8
   6-1. Install Kserve-adapter ............................................................................................................... 8
   6-2. Onboard sample-xapp descriptor and schema processing ....................................................... 9
   6-3. Generating and upload helm package ....................................................................................... 10
7. Reference ........................................................................................................................................ 14
1. Hardware requirements

Near-RT RIC configuration:

1. Hardware:
   - RAM: 8G RAM
   - CPU: 6 core
   - Disk: 40G Storage

2. Installation Environment:
   - Host: Windows 10
   - Hypervisor: VMware Workstation 16 Player
   - VM: Ubuntu 20.04 LTS (Focal Fossa)
   - Kubernetes version: 1.18.0

2. Context Diagram

- Cloud: To deploy/undeploy kserve inference service, RICDMS requests to deploy/undeploy to Kserve Adapter.
- RIC: Kserve Adapter requests onboarding of inference service to xapp-onboarder before deploy inference service.
- SMO: Kserve Adapter monitors and manages kserve inference service. Kserve Adapter will deliver monitoring information to SMO, and SMO can retrieve inference.

Source: [https://wiki.o-ran-sc.org/display/AIMLFEW/Kserve+Adapter](https://wiki.o-ran-sc.org/display/AIMLFEW/Kserve+Adapter)
3. Sequence Diagram

Source: [https://wiki.o-ran-sc.org/display/AIMLFEW/Kserve+Adapter](https://wiki.o-ran-sc.org/display/AIMLFEW/Kserve+Adapter)

4. Install Kserve/golang/chartmuseum on the Near-RT RIC

4-1. Install Kserve

1. `git clone https://gerrit.o-ran-sc.org/r/aiml-fw/aimlfw-dep`
2. `cd /aimlfw-dep/bin/`
3. `./install_kserve.sh`

4-2. Install golang

We need to install ricdms first.

1. `git clone https://gerrit.o-ran-sc.org/r/ric-plt/ricdms`

4-2-1. method 1 : Use ubuntu’s package repository to install golang.

1. `cd ricdms/`
2. `apt install golang-go`
4-2-2. method 2 : Installing Go via the Latest Binary Release.

Uninstall the existing Go package.

1. `sudo rm -rvf /usr/local/go`

Download specific binary release for your system.

1. `sudo wget https://go.dev/dl/go1.21.0.linux-amd64.tar.gz`

Extract package.

1. `sudo tar -xvf go1.21.0.linux-amd64.tar.gz -C /usr/local`

Setup Go Environment.

1. `export GOROOT=/usr/local/go`
2. `export GOPATH=$HOME/go`
3. `export PATH=$GOPATH/bin:$GOROOT/bin:$PATH`

- GOROOT : is the location where the Go package is installed on your system.
- GOPATH : is the work directory of your go project.

1. `source ~/.bashrc`

Verify Installations.

1. `go version`


4-3. Install chartmuseum

Add helm repository.
1. cd
2. helm repo add chartmuseum https://chartmuseum.github.io/charts

Pull the latest version.

1. helm fetch chartmuseum/chartmuseum

Unzip the file.

1. tar xzvf chartmuseum-3.10.1.tgz

Before Install the chartmuseum, revise the configuration.

1. vim chartmuseum/values.yaml

```yaml
# disable all routes prefixed with /api
DISABLE_API: false
# allow chart versions to be re-uploaded
ALLOW_OVERRIDE: true
```

Install chartmuseum.

1. helm upgrade --install chartmuseum ./chartmuseum

Check complete install or not.

1. kubectl get pod -A

Result:

```
<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cert-manager</td>
<td>1/1</td>
<td>Running 0</td>
<td>0</td>
<td>43h</td>
</tr>
<tr>
<td></td>
<td>cert-manager-cainjector</td>
<td>1/1</td>
<td>Running 0</td>
<td>0</td>
<td>43h</td>
</tr>
<tr>
<td></td>
<td>cert-manager-webbook</td>
<td>1/1</td>
<td>Running 0</td>
<td>0</td>
<td>43h</td>
</tr>
<tr>
<td></td>
<td>chartmuseum</td>
<td>1/1</td>
<td>Running 0</td>
<td>0</td>
<td>15h</td>
</tr>
<tr>
<td></td>
<td>istio-system</td>
<td>1/1</td>
<td>Running 0</td>
<td>0</td>
<td>43h</td>
</tr>
<tr>
<td></td>
<td>knative-serving</td>
<td>1/1</td>
<td>Running 0</td>
<td>0</td>
<td>43h</td>
</tr>
<tr>
<td></td>
<td>webhook-7ff5646cc5-g6js2</td>
<td>2/2</td>
<td>Running 0</td>
<td>0</td>
<td>43h</td>
</tr>
</tbody>
</table>
```
Check chart museum IP.

1. `kubectl get svc -A`

| default | chartmuseum | ClusterIP | 10.98.150.17 | <none> | 8080/TCP |

Check it has a chart or not.

1. `curl http://10.98.150.17:8080/api/charts`

Because I already complete the upload chart, if you have upload not yet, you will see this result `{}`.

Reference: [https://blog.csdn.net/lishuailing123/article/details/133313094](https://blog.csdn.net/lishuailing123/article/details/133313094)

5. Run rcdms on the Near-RT RIC

5-1. Install go-swagger first

1. `cd rcdms`
2. `git clone https://github.com/go-swagger/go-swagger`
3. `cd go-swagger`
4. `go install ./cmd/swagger`

To verify that go-swagger has been installed, type “swagger” and press ENTER. That should give below output.

1. **Please** specify one command of: diff, expand, flatten, generate, init, mixin, serve, validate or version

If you received the information like this “swagger: command not found”, Use this command to solve it.

1. `echo 'export PATH=$(GO_PATH-~/go)/bin:$PATH'` >> ~/.bashrc
2. `source ~/.bashrc`

Because you're missing $GOPATH/bin in your path reason why it cannot find it. Use this command to check again.

1. `swagger version`
5-2. Run ricdms

Build the file.

1. cd ricdms/
2. make build

If you receive error about “go: updates to go.mod needed; to update it:go mod tidy”, use this command to resolve.

1. go mod tidy

Reason:

It is usually because the version of the dependent package used in the project is inconsistent with the version recorded in the go.mod file. This problem can be solved by running “go mod tidy”.

Check the ricdms file is create or not.

1. ls

Use this command to build image.

1. make image

Step 27/30 : COPY dms-entrypoint.sh /opt/dms/
  ---> 78b16d72f923
Step 28/30 : ARG default_config=/opt/dms/config.yaml
  ---> Running in f60b51d6f374
Removing intermediate container f60b51d6f374
  ---> 5114a19071c0
Step 29/30 : ENV RIC_DMS_CONFIG_FILE=${default_config}
  ---> Running in 3b8aadb516ee
Removing intermediate container 3b8aadb516ee
  ---> ae01960ad587
Step 30/30 : ENTRYPOINT ["/opt/dms/dms-entrypoint.sh"]
  ---> Running in 1ccc87d2c82a
Removing intermediate container 1ccc87d2c82a
  ---> 38248310171e
Successfully built 38248310171e
Successfully tagged ric-dms:v1.0
Revise the “config-test.yaml” customOnboard-url port to “8080” which is “37019” and add new line below download-chart as download-charts-url-format: “http://127.0.0.1:8080/charts/%s-%s.tgz”.

1. cd config/
2. vim config-test.yaml

```yaml
log-level: debug
onborder-url: "http://127.0.0.1:9191"
mock-server: "127.0.0.1:9191"
getcharts-url: "http://127.0.0.1:9191/helmrepo/api/charts"
#download-charts-url-format: "http://127.0.0.1:9191/helmrepo/charts/%s-%s.tgz"
download-charts-url-format: "http://127.0.0.1:8080/charts/%s-%s.tgz"
getcharts-by-name-url: "http://127.0.0.1:9191/helmrepo/api/charts/%s"
getcharts-by-name-and-version-url: "http://127.0.0.1:9191/helmrepo/api/charts/%s/%s"
getcharts-by-name-and-version-url: "http://127.0.0.1:9191/helmrepo/api/charts/%s/%s"
customOnboard-url: "http://127.0.0.1:8080/api/charts"
```

Run ricdms.

1. export RIC_DMS_CONFIG_FILE=$(pwd)/config/config-test.yaml
2. ./ricdms

6. Run kserve-adapter on the Near-RT RIC

Open “new terminal” to install kserve-adapter.

6-1. Install Kserve-adapter

1. cd
2. git clone "https://gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter"

Check the go version is up then go1.16.0 version. If not, move to “4-2-2.” to upgrade the go version.

1. go version

Build Kserve-adapter.

1. cd kserve-adapter/
2. go get ./cmd/kserve-adapter
3. go build -o kserve-adapter cmd/kserve-adapter/main.go
6-2. Onboard sample-xapp descriptor and schema processing

Create namespace.

1. kubectl create ns ricips

Configuration setting.

1. export PATH=$PATH:/usr/local/go/bin/

Update <Model URL> in "sample_config.json" file.

1. cd kserve-adapter/pkg/helm/data/
2. vim sample_config.json

Revise the <Model URL>, this URL can obtain from your AI/ML dashboard where you complete to training.

```json
{
  "xapp_name": "sample-xapp",
  "xapp_type": "inferenceservice",
  "version": "2.2.0",
  "sa_name": "default",
  "inferenceservice": {
    "engine": "tensorflow",
    "runtime_version": "2.5.1",
    "api_version": "serving.kubeflow.org/v1beta1",
    "min_replicas": 1,
    "max_replicas": 1
  }
}
```

After revise, use this command to setting “KUBECONFIG”, “API_SERVER_PORT”, “CHART_WORKSPACE_PATH”, “RIC_DMS_IP” and “RIC_DMS_PORT” to run main.go.

1. cd ../../
2. KUBECONFIG=/root/.kube/config API_SERVER_PORT=10000
   CHART_WORKSPACE_PATH="/root/kserve-adapter/pkg/helm/data" RIC_DMS_IP=127.0.0.1
   RIC_DMS_PORT=8000
   go run cmd/kserve-adapter/main.go
6-3. Generating and upload helm package

Before upload helm package, you need to prepare preparatory work. Open "new terminal" and set up port forwarding.

1. `kubectl port-forward svc/chartmuseum 8080:8080`

Open "new terminal" to keep processing. Add helm repository.

1. `helm repo add localhost http://127.0.0.1:8080`

Check you can visit or not.

1. `curl http://127.0.0.1:8080/api/charts`

Result:

Because I already complete the upload chart processing, if you have upload not yet, you will see this result `{}.

Bad Result:

```
root@near-rt-ric:~/.kserve-adapter# curl http://127.0.0.1:8080/api/charts
curl: (7) Failed to connect to 127.0.0.1 port 8080: Connection refused
```

Now, check if all terminal is running.
Terminal 1: Run ricdms.

Terminal 2: kserve_adapter run main.go.

Terminal 3: chartmuseum port forwarding.

Terminal 4: Upload helm package. Use this command to upload helm package.


Terminal 1: Receive request to onboard.

Terminal 2: Onboard chart to ricdms
Terminal 3: Handling connection for 8080.

Terminal 4: Check uploaded charts.

1. `curl http://127.0.0.1:8080/api/charts`

Result:

6-4. Deploy the model

1. `curl --request POST --url 'http://127.0.0.1:10000/v1/ips?name=inference-service&version=1.0.0'`

Terminal 1: Response

Terminal 2: Response
Check deployment.

1. `kubectl get InferenceService -n ricips`

Result:

<table>
<thead>
<tr>
<th>NAME</th>
<th>URL</th>
<th>READY</th>
<th>PREV</th>
<th>LATEST</th>
<th>PREROLLEDOUTREVISION</th>
<th>LATESTREVISION</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample-xapp</td>
<td><a href="http://sample-xapp.ricips.example.com">http://sample-xapp.ricips.example.com</a></td>
<td>True</td>
<td>100</td>
<td>100</td>
<td>000001</td>
<td>26s</td>
<td></td>
</tr>
<tr>
<td>sample-xapp.predictor-default</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6-5. Perform predictions

Use below command to obtain Ingress port for Kserve.

1. `kubectl get svc istio-ingressgateway -n istio-system`

Duplicate the below content and revise IP and Port.

```bash
1. model_name=sample-xapp
2. curl -v -H "Host: $model_name.ricips.example.com" http://"IP of where Kserve is deployed":"ingress port for Kserve"/v1/models/$model_name:predict -d @./input_qoe.json
```

“IP of where Kserve is deployed”: 10.0.10.217 (Please revise to your IP.)
“Ingress port for kserve”: 31011 (Please revise to your IP.)

Create sample data to prediction.

```json
1. {"signature_name": "serving_default", "instances": [[[2.56, 2.56],
2. [2.56, 2.56],
3. [2.56, 2.56],
4. [2.56, 2.56],
5. [2.56, 2.56],
6. [2.56, 2.56],
7. [2.56, 2.56],
8. [2.56, 2.56],
9. [2.56, 2.56]]]}
```
6-6. Result

Use this command to trigger prediction.

1. `source predict_inference.sh`

Result:

```
root@near-rt-ric:-# source predict_inference.sh
* Trying 10.0.10.217:31011...
* TCP_NODELAY set
* Connected to 10.0.10.217 (10.0.10.217) port 31011 (#0)
> POST /v1/models/sample-xapp:predict HTTP/1.1
> Host: sample-xapp.ricips.example.com
> User-Agent: curl/7.68.0
> Accept: */*
> Content-Length: 248
> Content-Type: application/x-www-form-urlencoded
> upload completely sent off: 248 out of 248 bytes
> Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< content-length: 53
< content-type: application/json
< date: Sat, 28 Oct 2023 14:52:29 GMT
< x-envoy-upstream-service-time: 15
< server: istio-envoy
{
  "predictions": [[2.5540688, 2.56379425]]
} Connection #0 to host 10.0.10.217 left intact
```

7. Reference

- [https://stackoverflow.com/questions/60382748/go-swagger-command-not-found](https://stackoverflow.com/questions/60382748/go-swagger-command-not-found)
- [https://wiki.o-ran-sc.org/download/attachments/81297504/kserve_adapter_demo.mp4?api=v2](https://wiki.o-ran-sc.org/download/attachments/81297504/kserve_adapter_demo.mp4?api=v2)
- [https://adamtheautomator.com/install-go-on-ubuntu/](https://adamtheautomator.com/install-go-on-ubuntu/)