



DECEMBER 26, 2023

Install Notes

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



NAME : CHENG-ZE WU

Table of Contents

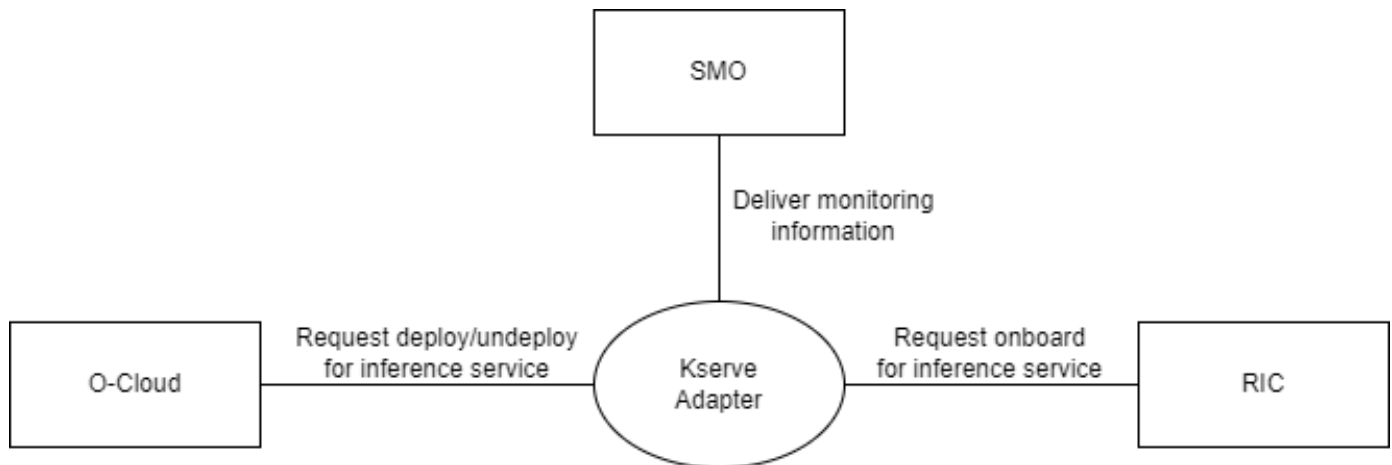
1. Hardware requirements	2
2. Context Diagram.....	2
3. Sequence Diagram	3
4. Install Kserve/golang/chartmuseum on the Near-RT RIC.....	3
4-1. Install Kserve.....	3
4-2. Install golang	3
4-2-1. method 1 : Use ubuntu's package repository to install golang.	3
4-2-2. method 2 : Installing Go via the Latest Binary Release.	4
4-3. Install chartmuseum	4
5. Run ricdms on the Near-RT RIC.....	6
5-1. Install go-swagger first.....	6
5-2. Run ricdms	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

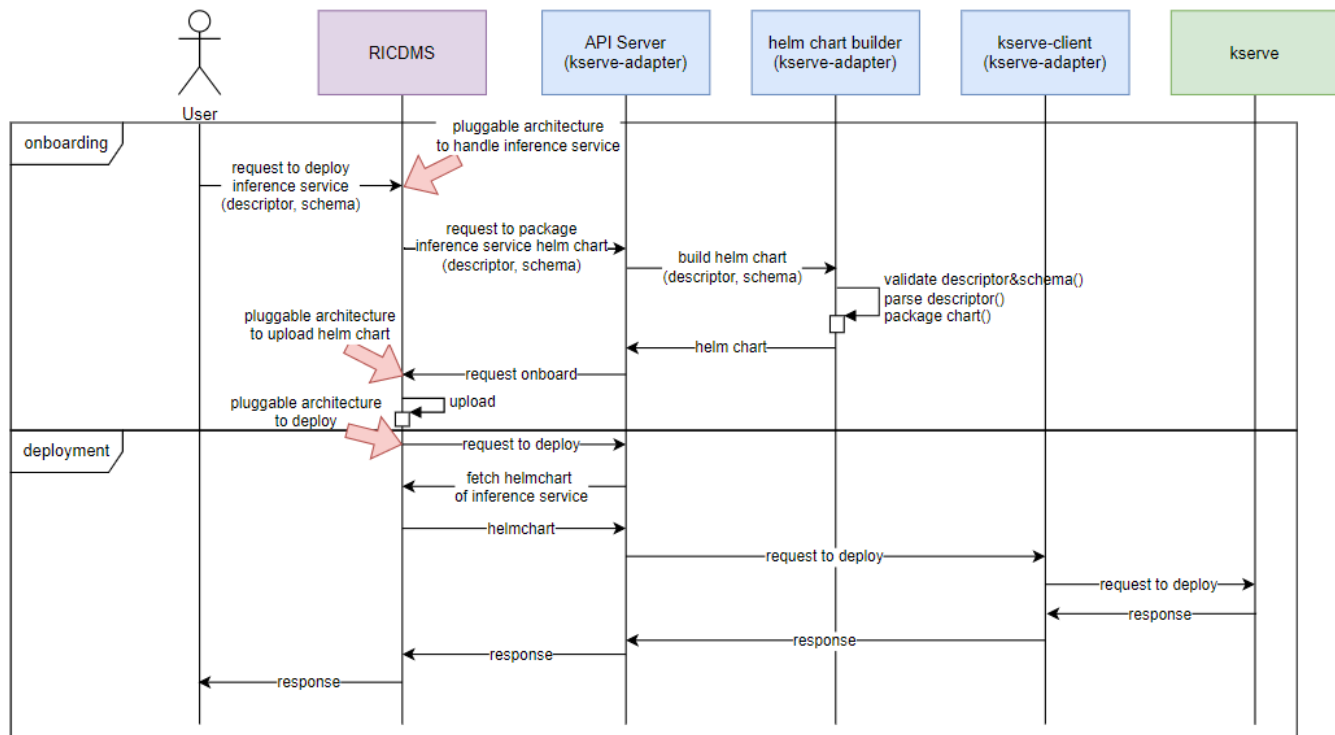
2. Context Diagram



- 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>

3. Sequence Diagram



Source: <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
```

```
root@h-near-rt-ric:~/ricdms# go version
go version go1.21.0 linux/amd64
```

Reference: <https://jinxankit.medium.com/upgrade-your-go-golang-version-to-1-21-latest-a-step-by-step-guide-1d72294453f8>

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`

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

Install chartmuseum.

1. `helm upgrade --install chartmuseum ./chartmuseum`

Check complete install or not.

1. `kubectl get pod -A`

Result :

```
root@h-near-rt-ric:~# kubectl get pod -A
NAMESPACE      NAME                                                    READY   STATUS    RESTARTS   AGE
cert-manager   cert-manager-5696d644bb-g2vmz                         1/1     Running   0           43h
cert-manager   cert-manager-cainjector-567957b749-k6fj9              1/1     Running   0           43h
cert-manager   cert-manager-webhook-74bcb7875d-axhcw                 1/1     Running   0           43h
default        chartmuseum-76ccb74dd-wwwvp                           1/1     Running   0           15h
istio-system   istio-ingressgateway-67d48bfc54-wpbvq                 1/1     Running   0           43h
istio-system   istiod-666444b865-hqcbm                               1/1     Running   0           43h
knative-serving activator-8484954469-skxpx                             1/1     Running   0           43h
knative-serving autoscaler-77b4bfb877-7sqhf                            1/1     Running   0           43h
knative-serving controller-c7c78dd7c-65rzq              1/1     Running   0           43h
knative-serving istio-webhook-5b8bc67c99-8gxks          1/1     Running   0           43h
knative-serving networking-istio-5966f45c8d-dcpvr      1/1     Running   0           43h
knative-serving webhook-7ff5646cc5-q6js2              1/1     Running   0           43h
kserve         kserve-controller-manager-0                            2/2     Running   0           43h
```

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
```

```
root@h-near-rt-ric:~# curl http://10.98.150.17:8080/api/charts
{"inference-service":{"name":"inference-service","version":"1.0.0","description":"Inference Service Helm Chart","apiVersion":"v1","appVersion":"1.0","urls":["charts/inference-service-1.0.0.tgz"],"created":"2023-10-27T18:33:00.457986206Z","digest":"68942acd1f04cba845228c7dbd3cca60031d03924fd8be728e30453870e576aa"}}root@h-near-rt-ric:~#
```

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>

5. Run ricdms on the Near-RT RIC

5-1. Install go-swagger first

```
1. cd ricdms
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="${GOPATH-~/go}/bin:$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`

```
root@h-near-rt-ric:~/ricdms# ls
Dockerfile  LICENSE.txt  README.md  cmd      container-tag.yaml  dms-entrypoint.sh  go-swagger  go.sum  pkg      ricdms
INFO.yaml  Makefile    api        config   deployment          docs               go.mod      go1.21.0.linux-amd64.tar.gz  releases
```

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`

```
log-level: debug
onboard-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"
getXappHealth-url: "http://127.0.0.1:9191/ric/v1/health/alive/%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`

```
root@h-near-rt-ric:~/ricdms# ./ricdms
{"ts":1698337800559,"crit":"INFO","id":"ricdms","mdc":{},"msg":"Logger is initialized with config file : /root/ricdms/config/config-test.yaml"}
{"ts":1698337800568,"crit":"INFO","id":"ricdms","mdc":{},"msg":"Starting server at : 0.0.0.0:8000"}
2023/10/26 16:30:00 Serving r i c d m s at http://[:]:8000
```

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.

```
1. {  
2.   "xapp_name": "sample-xapp",  
3.   "xapp_type": "inferenceservice",  
4.   "version": "2.2.0",  
5.   "sa_name": "default",  
6.   "inferenceservice": {  
7.     "engine": "tensorflow",  
8.     "storage_uri": "http://192.168.8.44:32002/model/qaetest6/1/Model.zip",  
9.     "runtime_version": "2.5.1",  
10.    "api_version": "serving.kubeflow.org/v1beta1",  
11.    "min_replicas": 1,  
12.    "max_replicas": 1  
13.  }  
14. }
```

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
```

Result :

```
root@h-near-rt-ric:~/kserve-adapter# 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=8080 go run cmd/kserve-adapter/main.go
[DEBUG][kserve-adapter]2023/10/27 06:38:44 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve client.go (*Client).Init : 93 [IN]
[DEBUG][kserve-adapter]2023/10/27 06:38:44 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve client.go inferenceServiceGetter : 70 [out of cluster]
[DEBUG][kserve-adapter]2023/10/27 06:38:44 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve client.go (*Client).Init : 101 [OUT]
[DEBUG][kserve-adapter]2023/10/27 06:38:44 main main.go main : 47 [IN]
[DEBUG][kserve-adapter]2023/10/27 06:38:44 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api_rest_server.go RunWebServer : 96 [IN]
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
- using env: export GIN_MODE=release
- using code: gin.SetMode(gin.ReleaseMode)

[GIN-debug] POST    /v1/ips                --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/deployment.Command.Deploy-fm (3 handlers)
[GIN-debug] DELETE /v1/ips                --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/deployment.Command.Delete-fm (3 handlers)
[GIN-debug] PUT     /v1/ips                --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/deployment.Command.Update-fm (3 handlers)
[GIN-debug] GET     /v1/healthcheck       --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/healthcheck.Command.Ping-fm (3 handlers)
[GIN-debug] GET     /v1/ips/revision      --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/revision.Command.Get-fm (3 handlers)
[GIN-debug] GET     /v1/ips/status        --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/status.Command.Get-fm (3 handlers)
[GIN-debug] GET     /v1/ips/info          --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/info.Command.Get-fm (3 handlers)
[GIN-debug] POST   /v1/ips/preparation   --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/preparation.Command.Preparation-fm (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :10000
```

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 :

```
root@h-near-rt-ric:~# curl http://127.0.0.1:8080/api/charts
{"inference-service":[{"name":"inference-service","version":"1.0.0","description":"Inference Service Helm Chart","apiVersion":"v1","appVersion":"1.0","urls":["charts/inference-service-1.0.0.tgz"],"created":"2023-10-27T18:33:00.457986206Z","digest":"68942acd1f04cba845228c7dbd3cca60031d03924fd8be728e30453870e576aa"}]}root@h-near-rt-ric:~#
```

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

Bad Result :

```
root@h-near-rt-ric:~# 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.

```
root@h-near-rt-ric:~/ricdms# ./ricdms
{"ts":1698337800559,"crit":"INFO","id":"ricdms","mdc":{},"msg":"Logger is initialized with config file : /root/ricdms/config/config-test.yaml"}
{"ts":1698337800568,"crit":"INFO","id":"ricdms","mdc":{},"msg":"Starting server at : 0.0.0.0:8000"}
2023/10/26 16:30:00 Serving r i c d m s at http://[:]:8000
```

Terminal 2 : kserve_adapter run main.go.

```
root@h-near-rt-ric:~/kserve-adapter# 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
[DEBUG][kserve-adapter]2023/10/27 06:38:44 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve client.go (*Client).Init : 93 [IN]
[DEBUG][kserve-adapter]2023/10/27 06:38:44 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve client.go inferenceServiceGetter : 70 [out of cluster]
[DEBUG][kserve-adapter]2023/10/27 06:38:44 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve client.go (*Client).Init : 101 [OUT]
[DEBUG][kserve-adapter]2023/10/27 06:38:44 main main.go main : 47 [IN]
[DEBUG][kserve-adapter]2023/10/27 06:38:44 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/rest_server.go RunWebServer : 96 [IN]
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
- using env: export GIN_MODE=release
- using code: gin.SetMode(gin.ReleaseMode)

[GIN-debug] POST    /v1/ips          --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/deployment.Command.Deploy-fm (3 handlers)
[GIN-debug] DELETE  /v1/ips          --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/deployment.Command.Delete-fm (3 handlers)
[GIN-debug] PUT     /v1/ips          --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/deployment.Command.Update-fm (3 handlers)
[GIN-debug] GET     /v1/healthcheck --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/healthcheck.Command.Ping-fm (3 handlers)
[GIN-debug] GET     /v1/ips/revision --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/revision.Command.Get-fm (3 handlers)
[GIN-debug] GET     /v1/ips/status   --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/status.Command.Get-fm (3 handlers)
[GIN-debug] GET     /v1/ips/info     --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/info.Command.Get-fm (3 handlers)
[GIN-debug] POST   /v1/ips/preparation --> gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/preparation.Command.Preparation-fm (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :10000
```

Terminal 3 : chartmuseum port forwarding.

```
root@h-near-rt-ric:~/ricdms# kubectl port-forward svc/chartmuseum 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::]:8080 -> 8080
```

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

1. `curl --request POST --url 'http://127.0.0.1:10000/v1/ips/preparation?configfile=pkg/helm/data/sample_config.js on&schemafile=pkg/helm/data/sample_schema.json'`

Terminal 1 : Recieve request to onboard.

```
root@h-near-rt-ric:~/ricdms# ./ricdms
{"ts":1698431550435,"crit":"INFO","id":"ricdms","mdc":{},"msg":"Logger is initialized with config file : /root/ricdms/config/config-test.yaml"}
{"ts":1698431550444,"crit":"INFO","id":"ricdms","mdc":{},"msg":"Starting server at : 0.0.0.0:8000"}
2023/10/27 18:32:30 Serving r i c d m s at http://[:]:8000
{"ts":1698431580450,"crit":"DEBUG","id":"ricdms","mdc":{},"msg":"==> invoked custom onboarding"}
{"ts":1698431580450,"crit":"DEBUG","id":"ricdms","mdc":{},"msg":"onboarder received req to onboard"}
```

Terminal 2 : Onboard chart to ricdms

```
[DEBUG][kserve-adapter]2023/10/27 18:33:00 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go buildURLforChartOnboard : 58 [IN]
[DEBUG][kserve-adapter]2023/10/27 18:33:00 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go buildURLforChartOnboard : 67 [OUT]
[DEBUG][kserve-adapter]2023/10/27 18:33:00 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go onboardChartToRicdms : 215 [IN]
[DEBUG][kserve-adapter]2023/10/27 18:33:00 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go onboardChartToRicdms : 249 [OUT]
[DEBUG][kserve-adapter]2023/10/27 18:33:00 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms.Executor client.go OnboardHelmChart : 70 [OUT]
[DEBUG][kserve-adapter]2023/10/27 18:33:00 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/controller/v1/adapter.Executor controller.go Preperation : 183 [OUT]
[DEBUG][kserve-adapter]2023/10/27 18:33:00 gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/preparation.Executor preparation.go Preperation : 69 [OUT]
[GIN] 2023/10/27 - 18:33:00 | 201 | 112.672071ms | 127.0.0.1 | POST | "/v1/ips/preparation?configfile=pkg/helm/data/sample_config.js on&schemafile=pkg/helm/data/sample_schema.json"
```

Terminal 3 : Handling connection for 8080.

```
root@h-near-rt-ric:~/ricdms# kubectl port-forward svc/chartmuseum 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
```

Terminal 4 : Check uploaded charts.

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

Result :

```
root@h-near-rt-ric:~# curl http://127.0.0.1:8080/api/charts
{"inference-service":{"name":"inference-service","version":"1.0.0","description":"Inference Service Helm Chart","apiVersion":"v1","appVersion":"1.0","urls":["charts/inference-service-1.0.0.tgz"],"created":"2023-10-27T18:33:00.457986206Z","digest":"68942acd1f04cba845228c7dbd3cca60031d03924fd8be728e30453870e576aa"}}
root@h-near-rt-ric:~#
```

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

```
root@h-near-rt-ric:~/ricdms# ./ricdms
{"ts":1698492897173,"crit":"INFO","id":"ricdms","mdc":{},"msg":"Logger is initialized with config file : /root/ricdms/config/config-test.yaml"}
{"ts":1698492897183,"crit":"INFO","id":"ricdms","mdc":{},"msg":"Starting server at : 0.0.0.0:8000"}
2023/10/28 11:34:57 Serving r i c d m s at http://[::]:8000
{"ts":1698492903321,"crit":"DEBUG","id":"ricdms","mdc":{},"msg":"==> Download helm chart"}
{"ts":1698492903321,"crit":"DEBUG","id":"ricdms","mdc":{},"msg":"DownloadCharts: invoked"}
{"ts":1698492903321,"crit":"DEBUG","id":"ricdms","mdc":{},"msg":"Download Charts invoked"}
```

Terminal 2 : Response

```
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/deployment.Execut
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/controller/v1/adapter.Ex
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms.Executor c
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go b
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go b
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go g
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go g
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go g
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go g
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go U
[ERROR] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go U
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms utils.go U
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/ricdms.Executor c
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/controller/v1/adapter ut
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/controller/v1/adapter ut
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/controller/v1/adapter ut
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve client.go
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve utils.go c
[DEBUG] [kserve-adapter] 2023/10/28 11:35:03 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve utils.go c
[DEBUG] [kserve-adapter] 2023/10/28 11:35:04 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/client/kserve client.go
[DEBUG] [kserve-adapter] 2023/10/28 11:35:04 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/controller/v1/adapter.Ex
[DEBUG] [kserve-adapter] 2023/10/28 11:35:04 Gerrit.o-ran-sc.org/r/aiml-fw/aihp/ips/kserve-adapter/pkg/api/v1/deployment.Execut
[GIN] 2023/10/28 - 11:35:04 | 201 | 1.279970557s | 127.0.0.1 | POST | "/v1/ips?name=inference-service&version=1.0.0"
```


Check deployment.

```
1. kubectl get InferenceService -n ricips
```

Result :

```
root@h-near-rt-ric:~/ricdms# kubectl get InferenceService -n ricips
NAME      URL                                     READY  PREV  LATEST  PREVROLLEDOUTREVISION  LATESTREADYREVISION  AGE
sample-xapp  http://sample-xapp.ricips.example.com  True   100   15021:32563/TCP,80:31011/TCP,443:32349/TCP,15012:32093/TCP,15443:31520/TCP  sample-xapp-predictor-default-00001  26m
```

6-5. Perform predictions

Use below command to obtain Ingress port for Kserve.

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

```
root@h-near-rt-ric:~# kubectl get svc istio-ingressgateway -n istio-system
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                                     AGE
istio-ingressgateway  LoadBalancer  10.107.192.239  <pending>     15021:32563/TCP,80:31011/TCP,443:32349/TCP,15012:32093/TCP,15443:31520/TCP  46h
```

```
1. cd
2. vim predict_inference.sh
```

Duplicate the below content and revise IP and Port.

```
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.

```
1. vim input_qoe.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@h-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, 2.56370425]
]
}
* Connection #0 to host 10.0.10.217 left intact
```

7. Reference

- <https://stackoverflow.com/questions/60382748/go-swagger-command-not-found>
- <https://shashankvivek-7.medium.com/go-swagger-a-go-web-framework-worth-learning-af0e9ed75343>
- https://wiki.o-ran-sc.org/download/attachments/81297504/kserve_adapter_demo.mp4?api=v2
- <https://adamtheautomator.com/install-go-on-ubuntu/>
- <https://jinxankit.medium.com/upgrade-your-go-golang-version-to-1-21-latest-a-step-by-step-guide-1d72294453f8>
- <https://docs.o-ran-sc.org/projects/o-ran-sc-aiml-fw-aimlfw-dep/en/latest/installation-guide.html>