Terradue

Advancing Earth Science

Processing and Chaining Overview

EOEPCA Operator Training Preparation

TERRADUE
Advancing Earth Science

ADES

- The **Application Deployment and Execution Service (ADES)** is responsible for the execution of the processing services through an **OGC Processes REST API**.
- The ADES software uses **ZOO-Project** as the main framework for exposing the **OGC compliant** rest api interface.
- It is designed to perform the processing and chaining functions on a **Kubernetes** cluster.
- The processing service specifications are grouped into an **Application Package**

TERRAQUE
Advancing Earth Science

OGC API Processes

Resource	Method	Description
/	GET	landing page of this API
/conformance	GET	Lists all requirements classes specified in the standard (e.g., OGC API - Processes Part 1: Core) that the server conforms to
/processes	GET	Lists all available processes this server offers.
/processes/{id}	GET	Describes a process.
/processes/{id}/execution	POST	Submits a new job.
/jobs	GET	Lists all running and finished jobs.
/jobs/{jobID}	GET	Shows the status of a job.
/jobs/{jobID}/results	GET	Lists available results of a job. In case of a failure, lists exceptions instead.

ADES API

Resource	Method	Description
/processes	POST	Deploys process
/processes/{id}	DELETE	Undeploys process

Application Package

- See OGC presentation about the Best Practices for EO Application Packages

OGC Publishes Best Practice for Earth Observation Application Packages

Contact:
info@ogc.org

Release Date:
25 January 2022

The document outlines the implementation, packaging, and deployment of cross-cloud EO Applications - A step forward for greater efficiency and bringing the 'user to the data.'

BP available at: https://docs.ogc.org/bp/20-089r1.html

Where is the ADES used?

- ESA's Charter Processing Environment "Charter Mapper", operated by Terradue:
  - Deployed on DataSpace and Scaleway
  - Processing services:
    - LIST Hasard Flood Delineation
    - NHAZCA Change Detection Analysis (IRIS)
    - DLR InSAR Browse
    - Sinergise S2Cloudless
    - Terradue with 10+ processing services
- Terradue mCube for the Asian Development Bank (ADB)
  - Deployed on Scaleway
  - Processing services:
    - Planetek Water Quality and Crop Monitoring processing services

Where is the ADES used?

- Geohazards Exploitation Platform (GEP):
  - Data ingestion and calibration (same model as Charter Mapper)
  - Near future ADES supporting the SNAPPING Interferogram generation
- Urban TEP
  - align to use the ADES as an additional processing backend
- H2020 Reliance
  - ADES deployed on the Poznan Supercomputing and Networking Center (PSNC) supporting the Geohazards community (S1 SLC InSAR)
- SpaceApps as EOEPCA Operators
  - Development environment
  - Staging environment @CreoDIAS

Terradue as Operators

- Deploy all Building Blocks @CreoDIAS
- Manage several deployment on different providers:
  - Use a single kubernetes cluster with node pools on different providers
    - AWS, Scaleway, PSNC, CreoDIAS
    - Coming months: Mundi, EODC, ONDA, Sobloo and CloudSigma
    - Deploy the services on one or more providers
- Users (service providers) from the H2020 NextOcean project:
  - Plymouth Marine Laboratory (PML)
    - Fisheries Monitoring & Surveillance service
    - Monitoring of Aquaculture Structures service

Helm Chart Configuration

- Helm Chart is available here: https://github.com/EOEPCA/helm-charts/tree/main/charts/ades

- Resource Manager

Parameter	Description	Default
workflowExecutor.useResourceManager	Enables resource manager	false
workflowExecutor.resourceManagerWorkspacePrefix	Resource manager workspace prefix	rm-user
workflowExecutor.resourceManagerEndpoint	Resource manager endpoint	https://resourcemanager-api.com

Helm Chart Configuration

- Application version

```
image:
 repository: eoepca/proc-ades
 pullPolicy: IfNotPresent
 tag: "2.0.0"
 proxyRepository: eoepca/kubectl-proxy
 proxyTag: "0.9.0"
```

- Ingress

```
ingress:
 enabled: true
 annotations: {}
 hosts:
   - host: ades.eoepca.com
     paths:
       - path: /
         pathType: ImplementationSpecific
 tls: []
```

TERRAQUE
Advancing Earth Science

Helm Chart Configuration

- Persistence
  - The Storage Class must support **ReadWriteMany**
  - Possible options (among other): **glusterfs-storage** or **longhorn**

```
workflowExecutor:
 processingStorageClass: glusterfs-storage
```

- Processing Resources

```
workflowExecutor:
     # Size of the Kubernetes Tmp Volumes
     processingVolumeTmpSize: "6Gi"
     # Size of the Kubernetes Output Volumes
     processingVolumeOutputSize: "6Gi"
     # Max ram to use for a job
     processingMaxRam: "8Gi"
     # Max number of CPU cores to use concurrently for a job
     processingMaxCores: "4"
```

Helm Chart Configuration

- cwl-wrapper
  - library used by the ADES to wrap all the phases (stage-in/application/stage-out) of the processing in a single CWL.
  - Git repos: https://github.com/EOEPCA/cwl-wrapper
- Stage-in / Stage-out

Parameter	Description
workflowExecutor.main.cwl	Main CWL workflow used by `cwl-wrapper`
workflowExecutor.stagein.cwl	Stage-in CWL workflow
workflowExecutor.stageout.cwl	Stage-out CWL workflow
workflowExecutor.rulez.cwl	Data structure for defining the CWL parameter used by `cwl-wrapper`

Helm Chart Configuration

Example of Application Package

Input of type **Directory**

Output of type **Directory**

```
- class: Workflow
  doc: Applies s expressions to EO acquisitions
  id: s-expression
  inputs:
    input_reference:
      doc: Input product reference
      label: Input product reference
      type: Directory
    s_expression:
      doc: s expression
      label: s expression
      type: string
    cbn:
      doc: Common band name
      label: Common band name
      type: string
  label: s expressions
  outputs:
  - id: wf_outputs
    outputSource:
    - step_1/results
    type: Directory
  steps:
    step_1:
      in:
        input_reference: input_reference
        s_expression: s_expression
        cbn: cbn
      out:
      - results
      run: '#clt'
```

- Git Repos: https://github.com/EOEPCA/app-s-expression/blob/main/app-s-expression.dev.0.0.2.cwl
- If an input of the application is of type **Directory**, the cwl-wrapper will prepend to the final CWL a stage-in step
- The stage-in step can be configured in the helm chart values:

```
workflowExecutor.stagein.cwl
```

Helm Chart Configuration

- stagein.cwl

Stars image

Stars copy command

Cwl Inputs

Env variables

```
cwlVersion: v1.0
id: stars
doc: "Run Stars for staging input data"
class: CommandLineTool
hints:
DockerRequirement:
    dockerPull: terradue/stars:1.0.0-beta.11
baseCommand: Stars
arguments:
- copy
- -v
- -rel
- -r
- '4'
- -o
- ./
- --harvest
inputs:
ADES_STAGEIN_AWS_SERVICEURL:
    type: string?
ADES_STAGEIN_AWS_ACCESS_KEY_ID:
    type: string?
ADES_STAGEIN_AWS_SECRET_ACCESS_KEY:
    type: string?
outputs: {}
requirements:
EnvVarRequirement:
    envDef:
    PATH: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
    AWS__ServiceURL: $(inputs.ADES_STAGEIN_AWS_SERVICEURL)
    AWS_ACCESS_KEY_ID: $(inputs.ADES_STAGEIN_AWS_ACCESS_KEY_ID)
    AWS_SECRET_ACCESS_KEY: $(inputs.ADES_STAGEIN_AWS_SECRET_ACCESS_KEY)
ResourceRequirement: {}
```

- The *Stars* **copy** command is responsible of downloading the data and converting it into a (STAC) item.
- *Stars* is an example of a stage-in/out container and it is provided as-is. Operators are invited to implement their own solutions according to their needs
- The **STAC** specification provides a common language to describe a range of geospatial information, so it can more easily be indexed and discovered.
  A 'spatiotemporal asset' is any file that represents information about the earth captured in a certain space and time.
- From the helm chart values, it is possible to set static inputs.
  For example, the access key and secret access key could be passed as static inputs to the services to be then set as environment variables in the stage in phase.

```
workflowExecutor:
  inputs:
    STAGEIN_AWS_SERVICEURL: http://data.cloudferro.com
    STAGEIN_AWS_ACCESS_KEY_ID: test
    STAGEIN_AWS_SECRET_ACCESS_KEY: test
```

Helm Chart Configuration

- Stagein step in the workflow CWL

```
steps:
   node_stage_in:
     in:
       ADES_STAGEIN_AWS_ACCESS_KEY_ID: ADES_STAGEIN_AWS_ACCESS_KEY_ID
       ADES_STAGEIN_AWS_SECRET_ACCESS_KEY: ADES_STAGEIN_AWS_SECRET_ACCESS_KEY
       ADES_STAGEIN_AWS_SERVICEURL: ADES_STAGEIN_AWS_SERVICEURL
       input: input_reference
     out:
     - input_reference_out
     run:
       arguments:
       - copy
       - -v
       - -rel
       - -r
       - '4'
       - -o
       - ./
       - --harvest
       baseCommand: Stars
       class: CommandLineTool
       cwlVersion: v1.0
       doc: Run Stars for staging input data
       hints:
         DockerRequiremeent:
           dockerPull: terradue/stars:1.0.0-beta.11
       id: stars
       inputs:
         ADES_STAGEIN_AWS_ACCESS_KEY_ID:
           type: string?
         ADES_STAGEIN_AWS_SECRET_ACCESS_KEY:
           type: string?
         ADES_STAGEIN_AWS_SERVICEURL:
           type: string?
         input:
           type: string?
       outputs:
         input_reference_out:
           outputBinding:
             glob: .
           type: Directory
       requirements:
         EnvVarRequirement:
           envDef:
             AWS_ACCESS_KEY_ID: $(inputs.ADES_STAGEIN_AWS_ACCESS_KEY_ID)
             AWS_SECRET_ACCESS_KEY:
$(inputs.ADES_STAGEIN_AWS_SECRET_ACCESS_KEY)
             AWS__ServiceURL: $(inputs.ADES_STAGEIN_AWS_SERVICEURL)
             PATH:
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
         ResourceRequirement: {}
```

- Sample inputs:

```
{
"ADES_STAGEIN_AWS_ACCESS_KEY_ID": "test",
"ADES_STAGEIN_AWS_SECRET_ACCESS_KEY": "test",
"ADES_STAGEIN_AWS_SERVICEURL": "http://data.cloudferro.com",
"cbn": "ndvi",
"input_reference":
"https://resource-catalogue.demo.eoepca.org/csw/?mode=opensearch&service=CSW&version=3.0.0&request=GetRecords&elementsetname=full&resulttype=results&typenames=csw:Record&recordids=S2B_MSIL2A_20200902T090559_N0214_R050_T34SFH_20200902T113910.SAFE",
 "s_expression": "(/ (- nir red) (+ nir red))",
"workflow": "s-expression",
"process": "wf-416fef08-32a9-11ed-a59d-6a311a077f94".
...
```

Helm Chart Configuration

- stageout.cwl

Stars image

Stars copy command

Processing result output

Stage out destination

```
cwlVersion: v1.0
baseCommand: Stars
doc: "Run Stars for staging results"
class: CommandLineTool
hints:
DockerRequirement:
    dockerPull: terradue/stars:1.0.0-beta.11
id: stars
arguments:
- copy
- -v
- -r
- '4'
- -o
- $( inputs.ADES_STAGEOUT_OUTPUT + "/" + inputs.process )
- -res
- $( inputs.process + ".res" )
- valueFrom: |
    ${
        if( !Array.isArray(inputs.wf_outputs) )
        {
            return inputs.wf_outputs.path + "/catalog.json";
        }
        var args=[];
        for (var i = 0; i < inputs.wf_outputs.length; i++)
        {
            args.push(inputs.wf_outputs[i].path + "/catalog.json");
        }
        return args;
    }
```

Inputs

Env variables

```
inputs:
ADES_STAGEOUT_AWS_PROFILE:
    type: string?
ADES_STAGEOUT_AWS_SERVICEURL:
    type: string?
ADES_STAGEOUT_AWS_ACCESS_KEY_ID:
    type: string?
ADES_STAGEOUT_AWS_SECRET_ACCESS_KEY:
    type: string?
aws_profiles_location:
    type: File?
ADES_STAGEOUT_OUTPUT:
    type: string?
ADES_STAGEOUT_AWS_REGION:
    type: string?
process:
    type: string?
outputs:
s3_catalog_output:
    outputBinding:
    outputEval: ${  return inputs.ADES_STAGEOUT_OUTPUT + "/" + inputs.process +
"/catalog.json"; }
    type: string
requirements:
InlineJavascriptRequirement: {}
EnvVarRequirement:
    envDef:
    PATH: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
    AWS__ServiceURL: $(inputs.ADES_STAGEOUT_AWS_SERVICEURL)
    AWS__SignatureVersion: "2"
    AWS_ACCESS_KEY_ID: $(inputs.ADES_STAGEOUT_AWS_ACCESS_KEY_ID)
    AWS_SECRET_ACCESS_KEY: $(inputs.ADES_STAGEOUT_AWS_SECRET_ACCESS_KEY)
  ResourceRequirement: {}
```

TERRAQUE

Advancing Earth Science

Thanks!

- Any Questions?