EOEPCA
Earth Observation Exploitation Platform Common Architecture
Operator Training

20th September 2022
Welcome

Agenda

- **Overview (20 mins)** - Richard
- **Building Blocks**
  - *Demonstration and description*
  - **Processing (45 mins)** – Fabrice/Blasco
  - **Resource Catalogue (30 mins)** - Angelos
  - **Data Access (45 mins)** - Fabian
  - **Workspace (20 mins)** - Bernhard
  - **User Management (20 mins)** - Alvaro
- **Deployment (30 mins)** - Richard
- **Q&A (15+ mins)**
Overview

Demonstration

Deployment

Q&A
Exploitation Platform

**EARTH OBSERVATION**
Data analysis

*PAST: performed by downloading the data*

**Exploitation Platform**
“Bring the user to the data”
A collaborative, virtual work environment providing access to EO data, algorithms, tools and ICT resources

**Analysis Execution**
Benefits from scalable, robust infrastructure – pay-per-use

**Cloud Infrastructure**
Increased user base

**Data**
Reaches a wider audience
Platform Ecosystem – Network of Resources

Platforms
- Virtual work environment
- Access data
- Develop algorithms
- Conduct analysis
- Share value-adding outcomes
- Collaborative communities

Platform Ecosystem
- Data sources
- Analysis tooling
- Cloud processing

Interoperation
Users of one platform may consume the services of another directly platform-to-platform.

Wouldn't it be nice if ... I didn't need to be an ICT wizard or instrument expert to integrate different data into my research or application?
Aspiration – Platform Interoperability

Multi-platform Workflow
Processing at Platform A
Combines processing at Platform B, with data from Platform C

DISCOVER: data + applications

EXECUTE: workflow
Workflow results

Open Interfaces
Standardisation through open interfaces seeks to reduce the friction between inter-platform points of contact

DISCOVER: data

Platform A

Platform B

Platform C
The goal of the Common Architecture is to define and agree a re-usable exploitation platform architecture by identifying a set of common building blocks that provide their services through open interfaces.

To encourage federation of EPs through an open consensus-based architecture for EPs in the Network of Resources.

To provide an open-source Reference Implementation of the architecture.
Use Cases

Consumer
- Discover/visualise
- Discover/visualise
- Discover/visualise
- Analysis
- Execute
- Annotate
- Federated Identity

Expert
- Data
- Applications & Workflows
- Value-added Products
- Interactive Environment
- Systematic & Bulk Processing
- Machine Learning
- Interoperate

Platform
- Publish
- Build
- Create
- Develop
- Train
- Portable Application
- Network of Resources

All rights reserved © 2022, Telespazio
Building Blocks

Resource Management
- Resource Catalogue
- Data Access
- Workspace

Processing & Chaining
- Application Deployment & Execution Service (ADES)
- Processor Development Environment
- Execution Management Service

User Management
- Login Service
  - Authorization Server
- Policy Decision Point
- Resource Guard
  - Policy Enforcement
- User Profile
- License Manager
- Accounting & Billing
Discovery – Resource Catalogue

Based on py WCS

Interfaces
- OGC CSW 2.0.2/3.0
- OGC API Records
- STAC
- OpenSearch with EO, Geo, Time

Data Model
- ISO 19115-1/2

Harvesting
- Push from Registrar/Harvester (Data Access)

Federation
- Via OGC CSW
Retrieval – Data Access

Based on EOxServer

Interfaces
- OGC WMS 1.1-1.3
- OGC WMTS 1.0
- OGC WCS 2.0
- OpenSearch with EO, Geo, Time

Harvester
- Currently integrated with CREODIAS

Registrar
- STAC Items from harvester
- Populates Resource Catalogue and EOX View Server
Processing - ADES

Based on ZOO project

ADES

Deployment & execution of user-defined processing

- OGC WPS 2.0
- Draft API Processes
- + deploy/undeploy
- STAC abstraction
- Calrissian
  CWL runner for Kubernetes

Application Package

- Metadata descriptor (CWL)
- Container image
- OGC Best Practice for EO Application Packages
Processor Development Environment (PDE)

- Integrated web tooling
  - Interactive analysis
  - Develop, test and package applications
- JupyterHub
  - Login integrates with platform authentication
- Spawns JupyterLab instance for user
- Replicate the conditions an application experiences when running in the ADES on a platform
- NEXT STEP – Integrate with User Workspace for Application publishing
Resources - Workspace

Workspace
- Centralised management of user’s owned resources:
  - Processing outputs
  - Application Packages
  - Uploaded products
- Can also be used as a Group/Project Workspace
- Dedicated Resource Catalogue
- Dedicated Data Access
- S3 bucket integration

Workspace API
- REST API
- Admin: create and manage workspaces
- User: register resources
User Identity & Authorization

**Federation of user requests amongst platforms**

**User Identity**
- OpenID Connect (OIDC)
- External identity
  - GitHub
  - COIH

**Access Management**
- User Managed Access (UMA)
- Policy-based resource protection

**Resource Management**
- Resource API ~ registration
- Policy API ~ access rules
Platform – Reference Implementation

- Kubernetes ‘abstract’ infrastructure
- Containerised components
- Helm charts for Kubernetes
- Deployed to CREODIAS
In Summary

- Exploitation Platform blueprint
- Reference implementation building blocks
- Embraces the ‘apps-to-the-data’ architecture
- Open interface standards
- Encourage platform interoperability
Overview

Demonstration
- Processing
- Resource Catalogue
- Data Access
- Workspace
- User Management

Deployment

Q&A
Jupyter Notebook Demo

• https://github.com/EOEPCA/demo
Deployment Overview

Deployment Guide
• [https://deployment-guide.docs.eoepca.org/](https://deployment-guide.docs.eoepca.org/)
• Versions match EOEPCA releases – v1.0, v1.1 and ‘latest’ development tip
• Describes:
  - **Cluster** ~ Kubernetes requirements and prerequisites
  - **EOEPCA** ~ Deployment of EOEPCA building blocks using helm
  - **Quickstart** ~ Scripted example deployments using bash and helm

Prerequisites
• Kubernetes cluster
• helm ~ for deployment of building blocks
• kubectl ~ for Kubernetes interactions

Additional for Scripted Deployment
• minikube ~ target Kubernetes cluster
• docker ~ for minikube’s docker driver
Kubernetes

Kubernetes v1.22.x (ref. eoepca v1.1)
- Rancher Kubernetes Engine (RKE) and minikube
- Helm charts provided for each building block:
  - EOEPCA: https://eoepca.github.io/helm-charts/
  - Data Access BB: https://charts-public.hub.eox.at/

Development Cluster (@CREODIAS)
- Rancher Kubernetes Engine (RKE) v1.3.4
- Kubernetes version v1.22.5
- 1 Master node (2 vCPU, 8 GB RAM)
- 5 Worker nodes (4 vCPU, 16 GB RAM)
- 1 NFS server (2 vCPU, 8 GB RAM)

Example Deployment
- Single node for minikube
- 8 vCPU, 32 GB RAM

Supporting Services
- Nginx Ingress Controller
  - Helm: https://kubernetes.github.io/ingress-nginx
- Cert Manager ~ for Ingress TLS with letscrypt
  - Helm: https://charts.jetstack.io
- NFS Storage Provisioner
  - Helm: https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner
- Harbor ~ container registry for Workspace
  - Helm: https://helm.goharbor.io

Other Services
- Sealed Secrets ~ for secret management in git
  - Helm: https://bitnami-labs.github.io/sealed-secrets
- MinIO ~ S3-compatible object storage
  - Helm: https://charts.bitnami.com/bitnami
RKE Example

```yaml
cluster_name: demo
kubernetes_version: "v1.22.5-rancher1-1"

nodes:
  - address: 192.168.108.9
    user: eouser
    role:
      - controlplane
      - etcd
  - address: 192.168.108.7
    user: eouser
    role:
      - worker
  - address: 192.168.108.12
    user: eouser
    role:
      - worker
  - address: 192.168.108.3
    user: eouser
    role:
      - worker
  - address: 192.168.108.14
    user: eouser
    role:
      - worker
  - address: 192.168.108.18
    user: eouser
    role:
      - worker

ingress:
  provider: none

bastion_host:
  address: 185.52.195.34
  user: eouser

private_registries:
  - user: eopca
    password: XXXXXXXXXXXX
```

Once the configuration is in place then the cluster creation can be initiated...

```
$ rke up
```

Limited local deployment can be made using a suitable local single-node kubernetes deployment using - for example using `minikube`...

```
minikube -p eopca start --cpus max --memory max --kubernetes-version v1.22.5
minikube profile eopca
```
Storage & Persistence

Component persistence
- Persistent Volume Claims
- Dynamic Claims via Storage Class
- Configured via helm release values

ReadWriteMany Storage
- Volume can be mounted as read-write by many nodes
- Required by some BBs – e.g. ADES
- Dynamically provisioned via Storage Class

Options
- NFS
- Clustered: e.g. Longhorn, GlusterFS
- Minikube ‘standard’ storage class (host)

EOEPCA Development Team Approach
- NFS Server – outside Kubernetes cluster
- NFS Subdirectory External Provisioner
  - Supports ReadWriteMany storage
- Pre-defined Persistent Volumes & Claims – per domain:
  - Resource Management: eoepca-resman-pv/pvc
  - Processing: eoepca-proc-pv/pvc
  - User Management: eoepca-userman-pv/pvc
- Storage Classes (for dynamic provisioning)
  - managed-nfs-storage (reclaimPolicy: Delete)
  - managed-nfs-storage-retain (reclaimPolicy: Retain)
Helm Repositories

Building Blocks

- **EOEPCA**
  - [https://eoepca.github.io/helm-charts](https://eoepca.github.io/helm-charts)
  - ADES
  - Bucket Operator
  - Login Service
  - PDE
  - PDP
  - Resource Catalogue
  - Resource Guard
  - User Profile
  - Workspace API

- **EOX**
  - [https://charts-public.hub.eox.at/](https://charts-public.hub.eox.at/)
  - vs (Data Access)

Others

- **Jetstack**
  - [https://charts.jetstack.io/index.yaml](https://charts.jetstack.io/index.yaml)
  - Cert Manager

- **NGINX**
  - [https://kubernetes.github.io/ingress-nginx/](https://kubernetes.github.io/ingress-nginx/)
  - NGINX Ingress Controller

- **Bitnami**
  - [https://charts.bitnami.com/bitnami](https://charts.bitnami.com/bitnami)
  - Minio

- **Bitnami Labs**
  - [https://bitnami-labs.github.io/sealed-secrets](https://bitnami-labs.github.io/sealed-secrets)
  - Sealed Secrets

- **Harbor**
  - [https://helm.goharbor.io/](https://helm.goharbor.io/)
  - Harbor
Login Service Deployment

Provides the platform Authorization Server for authenticated user identity and request authorization

Helm Chart
Installed via ‘login-service’ helm chart in the EOEPCA chart repository

```
helm install -repo https://eoepca.github.io/helm-charts
    --version 1.1.5
    --values login-service-values.yaml
login-service
login-service
```

Values
Many values can be specified.

Typically, at minimum...

- **Public hostname of Authorization Server**
  e.g. `auth.myplatform.org`

- **Public IP address**
  e.g. of `LoadBalancer`, `reverse-proxy` etc.

- **Initial password for the admin user**
  >=6 chars, >=1 each of uppercase, lowercase, digit, special

- **Volume claim for persistence**
  The name of an existing claim is expected

- **TLS configuration**
  incl. certificate provider – e.g. for `letsencrypt`

- Plus, ‘front-end’ customizations via ConfigMap

Reference
User Profile Deployment

Provides access to the user’s ‘account’ in the platform

**Helm Chart**

Installed via ‘user-profile’ helm chart in the EOEPCA chart repository

```
helm install -repo https://eoepca.github.io/helm-charts
  --version 1.1.3
  --values user-profile-values.yaml
user-profile
user-profile
```

**Values**

Typical values to specify…

- **Public hostname of Authorization Server**
  e.g. `auth.myplatform.org`

- **Public IP address**
  e.g. of `LoadBalancer`, `reverse-proxy` etc.

- **Volume claim for persistence**
  *The name of an existing claim is expected*

**Reference**

PDP Deployment

Provides the platform policy database and associated service for access policy decision requests

Helm Chart
Installed via ‘pdp-engine’ helm chart in the EOEPCA chart repository

```bash
helm install -repo https://eoepca.github.io/helm-charts
--version 1.1.3
--values pdp-values.yaml
pdp
pdp-engine
```

Values

Typical values to specify…

- **Public hostname of Authorization Server**
  e.g. auth.myplatform.org
- **Public IP address**
  e.g. of LoadBalancer, reverse-proxy etc.
- **Volume claim for persistence**
  *The name of an existing claim is expected*

Reference

Resource Protection (access authorisation)

Multiple instances - deployed ‘in front of’ each resource server (e.g. ADES etc.) to apply protection that integrates with Login Service, PDP.

Injects itself into the `auth_request` path of the Nginx Ingress Controller

**Helm Chart**
Installed via ‘resource-guard’ helm chart in the EOEPCA chart repository

```
helm install -repo
  https://eoepca.github.io/helm-charts
  --version 1.0.7
  --values myservice-guard-values.yaml
  myservice-guard
  resource-guard
```

**Values**
Wraps the charts for the PEP and the UMA User Agent:

Typically, at minimum…
- **Public hostname and IP of Authorization Server**
  e.g. auth.myplatform.org + LoadBalancer/Proxy IP
- **Authorization Server Client Credentials**
  Supplied via a Kubernetes Secret
- **Volume claim for PEP persistence**
  Can be created ‘on-demand’ by the chart
- **Ingress configuration**
  For the services at the protected resource server
- **TLS configuration**
  incl. certificate provider – e.g. for letsencrypt

**Reference**
ADES Deployment

Provides a platform-hosted execution engine through which users can initiate parameterised processing jobs using applications made available within the platform.

Helm Chart
Installed via ‘ades’ helm chart in the EOEPCA chart repository

```
helm install -repo
https://eoepca.github.io/helm-charts
--version 1.1.10
--values ades-values.yaml
ades
ades
```

Values
Many values can be specified.

Typically, at minimum...
- **StorageClass for dynamic persistence**
  Must be `ReadWriteMany`
  - **StageIn/out configuration**
    CWL to override the chart built-in
  - **S3 Object Store integration**
    To persist stage-out of results
  - **Workspace integration**
    For registration of results during stage-out
  - **PEP integration**
    For dynamic registration of resources

Reference
PDE Deployment

The Processor Development Environment (PDE) provides a web-based application that allows the user to perform platform-hosted interactive analysis and application development.

Helm Chart
Installed via 'pde' helm chart in the EOEPCA chart repository

code

```
helm install -repo
https://eoepca.github.io/helm-charts
--version 1.1.12
--values pde-values.yaml
pde
jupyterhub
```

Values

Typical values to specify...

- **Public service URL**
  e.g. pde.myplatform.org

- **StorageClass for dynamic persistence**

- **JupyterHub -> login-service integration for authentication**
  **OAuth service URLs**

- **Initial admin password**

Reference

Resource Catalogue Deployment

Provides a standards-based EO metadata catalogue that includes support for OGC CSW / API Records, STAC and OpenSearch

Helm Chart
Installed via ‘rm-resource-catalogue’ helm chart in the EOEPCA chart repository

```
helm install -repo https://eoepca.github.io/helm-charts
   --version 1.1.0
   --values resource-catalogue-values.yaml
resource-catalogue
rm-resource-catalogue
```

Values
Many values can be specified.

Typically, at minimum...
- **Public service URL**
  e.g. catalogue.myplatform.org
- **StorageClass for dynamic persistence**
- **Metadata describing the Catalogue instance**
  E.g. for Inspire compliance
- **Tuning configuration for PostgreSQL**
  Optional – for advanced tweaking

Reference
Data Access Deployment

Provides standards-based services for access to platform hosted data - OGC WMS/WMTS (visualisation), OGC WCS(retrieval), plus data harvesting and registration

Helm Chart
Installed via ‘vs’ helm chart in the EOX chart repository

```
helm install -repo https://charts-public.hub.eox.at/
  --version 2.1.4
  --values data-access-values.yaml
data-access
vs
```

Values
Many values can be specified.

Typically, at minimum...

- **Public service URL**
  e.g. data-access.myplatform.org
- **StorageClass for dynamic persistence**
- **Metadata describing the service instance**
- **Data Specification**
  Layers, data collections and product types
- **Harvester configuration**
  Defining search endpoints and query params
- **Volume claims for persistence**
  To re-use existing volume claims (database, redis)
- **Object storage details**
  For data and cache components

Reference
[https://vs.pages.eox.at/documentation/operator/main/configuration.html#helm-configuration-variables](https://vs.pages.eox.at/documentation/operator/main/configuration.html#helm-configuration-variables)
Workspace API Deployment

Provides protected user resource management that includes dedicated storage and services for resource discovery and access.

Helm Chart
Installed via ‘rm-workspace-api’ helm chart in the EOEPCA chart repository

```
helm install -repo
https://eoepca.github.io/helm-charts
--version 1.1.11
--values workspace-api-values.yaml
workspace-api
rm-workspace-api
```

Values
Many values can be specified.

Typically, at minimum...
- **Public service URL**
  e.g. workspace-api.myplatform.org
- **Workspace naming prefix**
  Acts as a namespace, to avoid name clashes
- **Object storage access details**
  Url, credentials, etc.
- **Container registry access details**
  Url, credentials, etc.
- **Workspace helm charts – ConfigMap...**
  - template-hr-resource-catalogue.yaml
  - template-hr-vs.yaml
  - template-hr-resource-guard.yaml
- **Optional installation of flux**

Reference
Bucket Operator Deployment

Provides a Kubernetes operator through which object storage ‘buckets’ are created for users

Helm Chart
Installed via ‘rm-bucket-operator’ helm chart in the EOEPACA chart repository

Values

Typical values to specify…
- **OpenStack details…**
  member role ID, service project ID, …
- **OpenStack credentials (secret)…**
  username, password, domain

Reference

https://github.com/EOEPACA/rm-bucket-operator#readme

```
helm install -repo https://eoepca.github.io/helm-charts
  --version 0.9.9
  --values bucket-operator-values.yaml
  bucket-operator
  rm-bucket-operator
```
Container Registry Deployment

Harbor container registry to support the development and deployment of ADES application packages

Helm Chart
Installed via ‘harbor’ helm chart in the Harbor chart repository

```
helm install -repo 
  https://helm.goharbor.io 
  --version 1.7.3     
  --values harbor-values.yaml 
  harbor 
  harbor
```

Values
Many values can be specified.

Typically, at minimum...

- **Public service URL**
  e.g. harbor.myplatform.org
- **StorageClass** for dynamic persistence
- **Initial admin password**

Reference

https://goharbor.io/docs/2.4.0/install-config/harbor-ha-helm/
Quickstart Deployments

Scripted Deployment
• Full system deployment via helm + bash scripts

```bash
git clone https://github.com/EOEPCA/deployment-guide
cd deployment-guide
./deploy/eoepca/eoepca.sh
```

• Configured via cmdline args and environment variables
• By default:
  - Creates a minikube cluster + support services
  - Deploys EOEPCA building blocks into cluster
• Environment variables:
  - Control cluster creation + service selection
    Ref. variables REQUIRE_<cluster-component>
  - Control EOEPCA building block selection
    Ref. variables REQUIRE_<eoepca-component>
  - Configure common aspects of helm chart values
• https://deployment-guide.docs.eoepca.org/current/quickstart/scripted-deployment/#environment-variables

Customised Deployments
• Simple: ./deploy/simple/simple
• Processing: ./deploy/processing/processing
• Data Access: ./deploy/data-access/data-access
• CREODIAS: ./deploy/creodias/creodias

How to use these?
• Instantiate a local cluster in minikube
  Provides an environment for experimentation
• Deploy to existing cluster
  - Disable minikube: REQUIRE_MINIKUBE=false
  - Ensure KUBECONFIG is set
• Inspect deployments with helm
  - helm get values <name>
    See configured helm chart values
  - helm get manifest <name>
    See resultant Kubernetes yaml
• Inspect the scripts to observe examples
• Customise your own deployment
  By adapting the examples
THANK YOU
FOR YOUR ATTENTION

telespazio.com