WP4 : Current ideas for Common Portal Architecture

10th September, 2019

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Summary

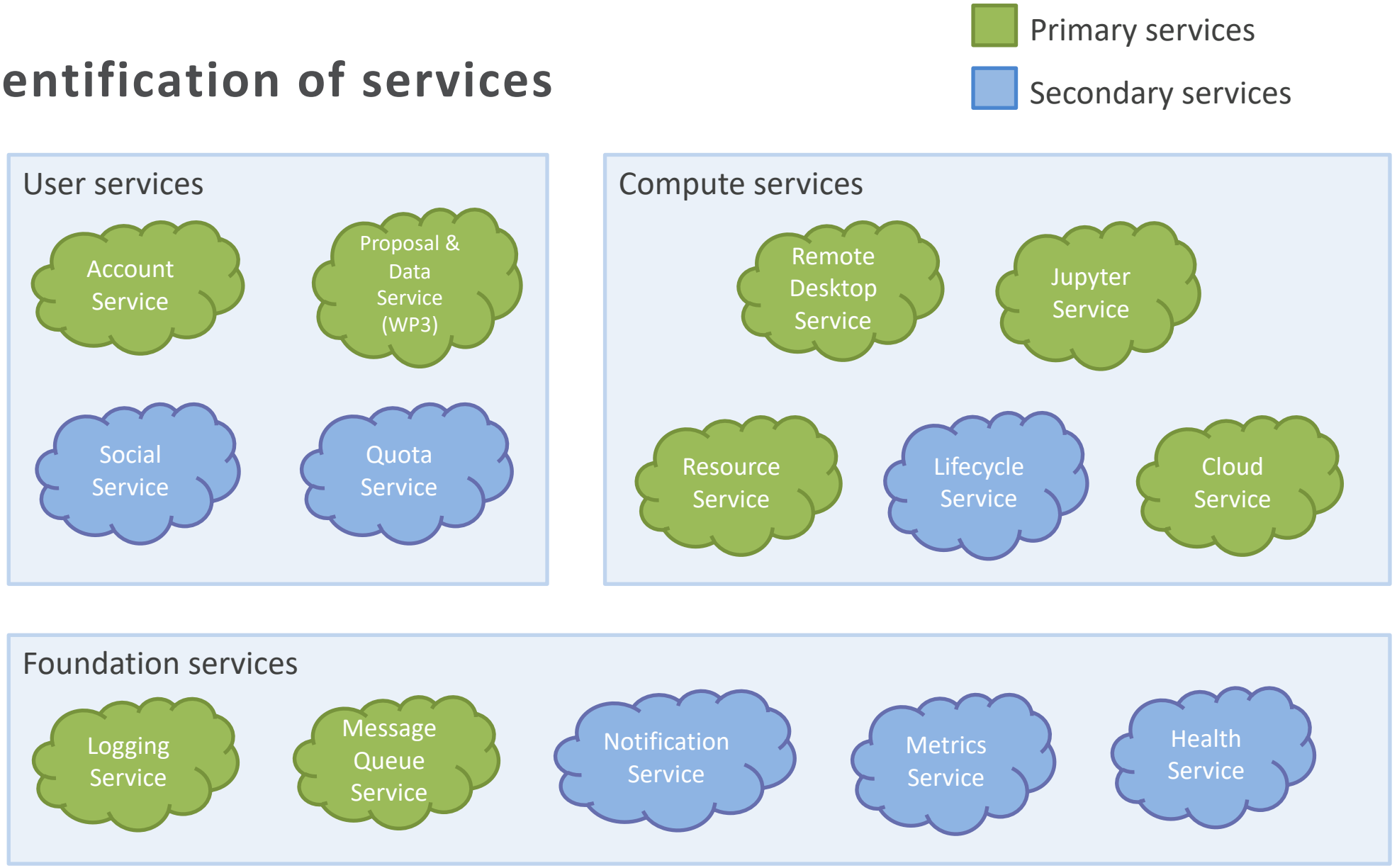
- **Current state built from discussion between ILL (Jamie, Stuart, William) and ESRF (Aidan)**

- **Build from experiences of current implementations**
  - VISA (ILL)
  - CalipsoPlus (ESRF, ALBA, PSI, …)

- **Aim to have a more flexible/decoupled system**
  - Move away from a monolithic application structure
  - Easy to have multiple developers developing separate functionalities (shared responsibilities)
  - Enable site-specific implementations
  - Less dependence on a single language (although better to have common one)
  - Well-defined API for each module/service

- **Microservice Architecture seems well adapted to these requirements**

Identification of services



Foundation services

Name	Description
Logging Service	Centralised logging
Message Queue	Decoupling of services Publisher-Subscriber actions
Notification Service	General push notifications to users Centralised email service
Metrics Service	Usage stats Reporting Most popular analysis environments (RD/Jupyter) Number and percentage of hardware resources used (current and for specific date range)
Health Service	Verify health of all other services

User services

Name	Description
Account Service	Token validation (WP6) Account information (user attributes) Role management Usage abuse/blacklist
Social Service	Chat Help/support Screen sharing
Quota Service	Handle per-user quota
Proposal & Data Service (WP3: FAIR data API)	Find proposals Get data paths Unarchive data

Compute services

Name	Description
Jupyter Service	Give access to Jupyter notebooks and kernels - *Initially investigated integration of JupyterHub into portal architecture however API does not facilitate this.* - *Current idea is to spawn Jupyter Notebook Servers directly on k8s/Slurm (same as JupyterHub, but we handle the resource allocation)*
Remote Desktop Service	Protocol for remote desktop access Web-socket management
Resource Service	Manages list of available analysis environment images Manages resource flavours (memory and CPU) Create resources (image + memory + CPU) Associated proposals with resource
Cloud Service	Interface with backend cloud (k8s, OpenStack) - Create/delete resource/container - Poll states or resources - Obtain IP:PORT of a container Handle queuing of resource requests Report memory and CPU usage
Lifecycle Service	Handles lifecycle of analysis environments Notify users when environment is ready or will expire

Current status and ideas

- **Refining architecture services**
  - Improving the definition of functionalities of each one
- **Start defining APIs for *Primary Services***
- **Node seems to be a common language between ILL and ESRF**
  - How is this for other sites?
- **Current architecture is Work in Progress**
  - Changes are likely!
  - Working document available in confluence: https://confluence.panosc.eu/display/wp4/Common+Portal+Architecture
  - All feedback/suggestions are very welcome