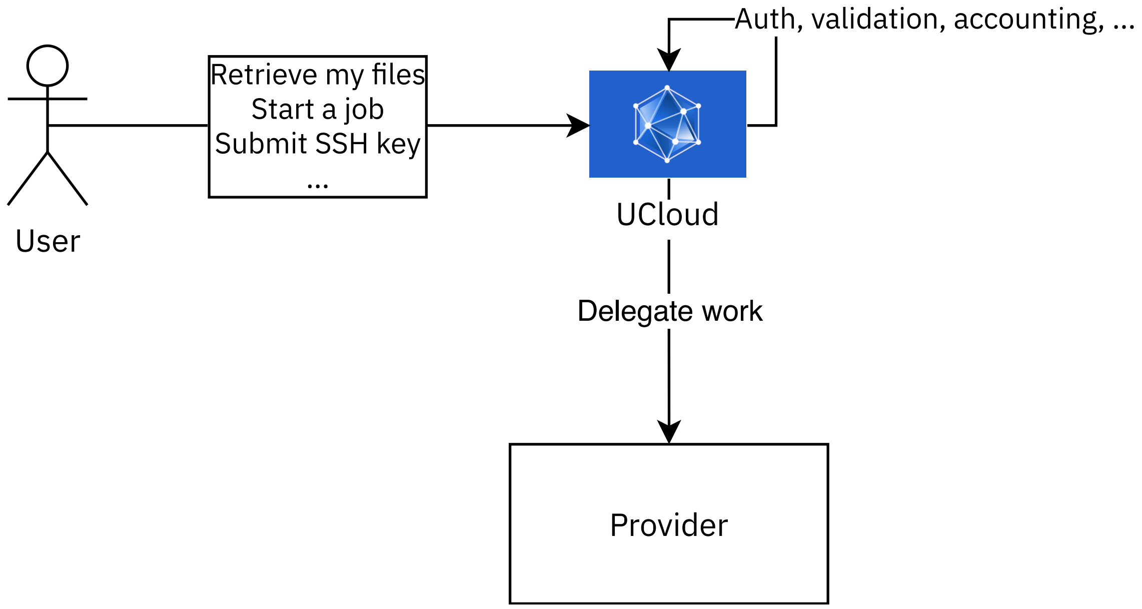
UCloud

Provider Integration

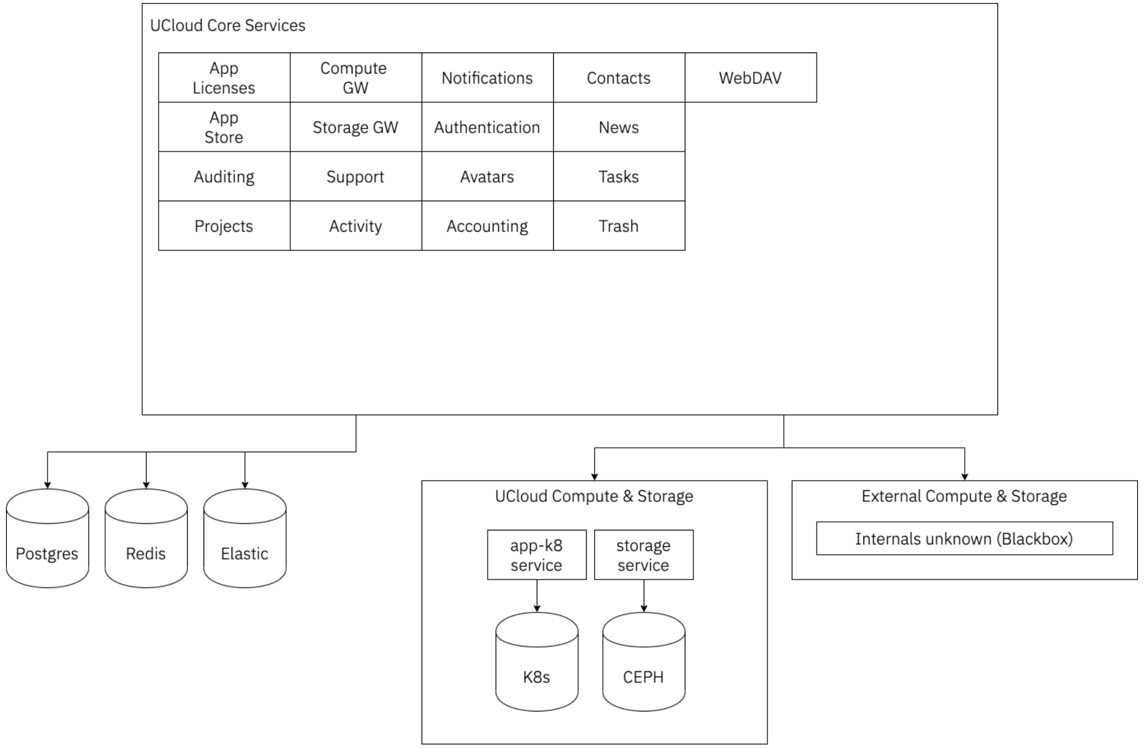
Providers

A provider of UCloud is an entity who provides one or more resources to the users of UCloud.



Providers

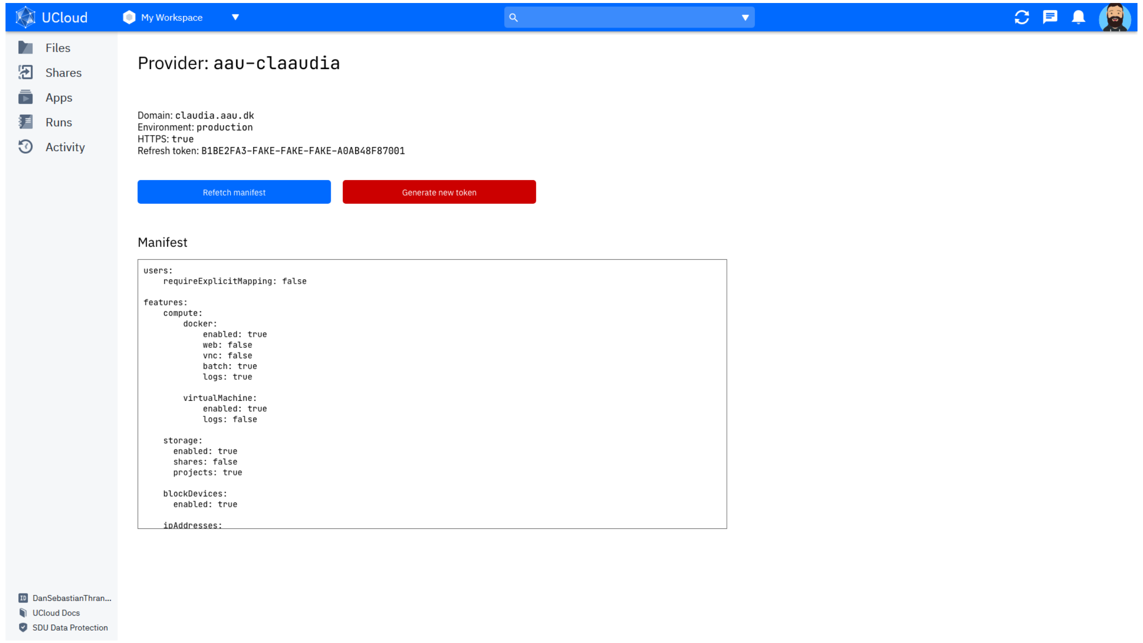
A provider of UCloud is an entity who provides one or more resources to the users of UCloud.



Definition

A UCloud provider is identified by two pieces of information:

1. Metadata: Statically defines to UCloud who they are, and how to get more information.
2. Manifest: Provides information about what features the provider supports.



Provider: aau-claaudia

Domain: claudia.aau.dk
Environment: production
HTTPS: true
Refresh token: B1BE2FA3-FAKE-FAKE-FAKE-A0AB48F87001

Manifest

```
users:
    requireExplicitMapping: false

features:
    compute:
        docker:
            enabled: true
            web: false
            vnc: false
            batch: true
            logs: true

        virtualMachine:
            enabled: true
            logs: false

    storage:
      enabled: true
      shares: false
      projects: true

    blockDevices:
      enabled: true

    ipAddresses:
```

Metadata

```yaml
id: ucloud
domain: cloud.sdu.dk
https: true
env: production
```

- Provides a unique and immutable identifier for the provider
- Defines how to contact the provider

Manifest

```
features:
    compute:
        docker:
            enabled: true
            web: false
            vnc: false
            batch: true
            logs: true
        virtualMachine:
            enabled: true
            logs: false
    storage:
      enabled: true
      shares: false
      projects: true
    blockDevices:
      enabled: true
    ipAddresses:
      enabled: true
```

- Manifest is collected by UCloud by pulling from the provider
- All features are opt-in
- UCloud will not present the option to the user if it is not supported by the provider

Integration

Providers will have two choices when it comes to integrating with UCloud:

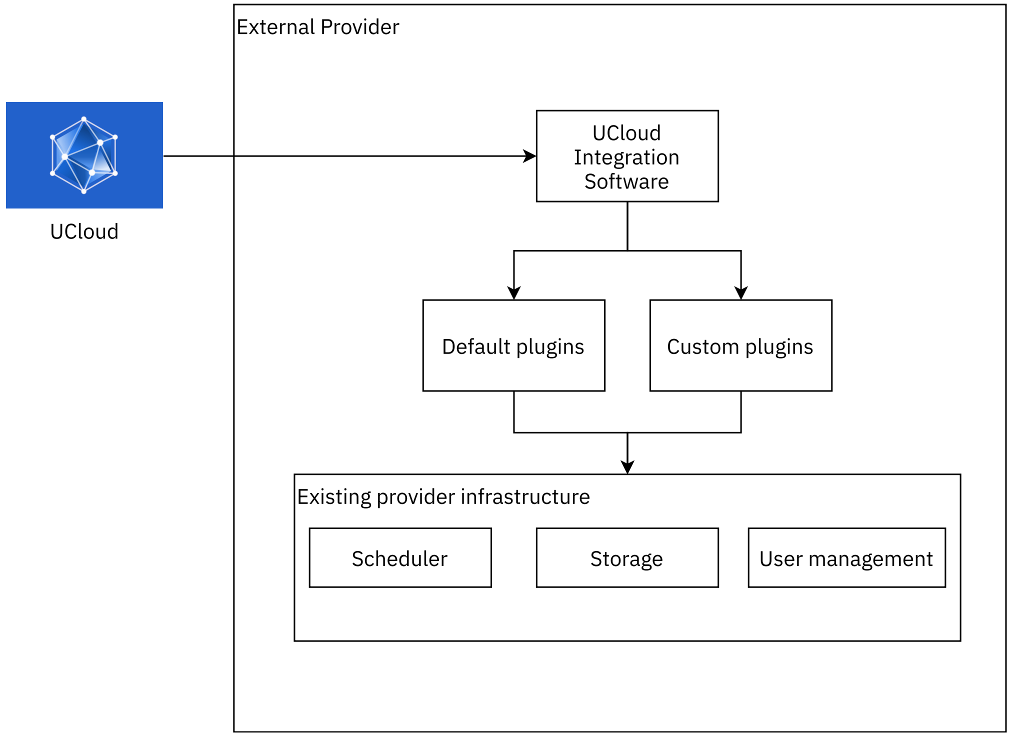
1) The UCloud integration software

- A single piece of software to install
- Plugin based architecture
- Gives you a very large degree of control

2) Implement low-level APIs

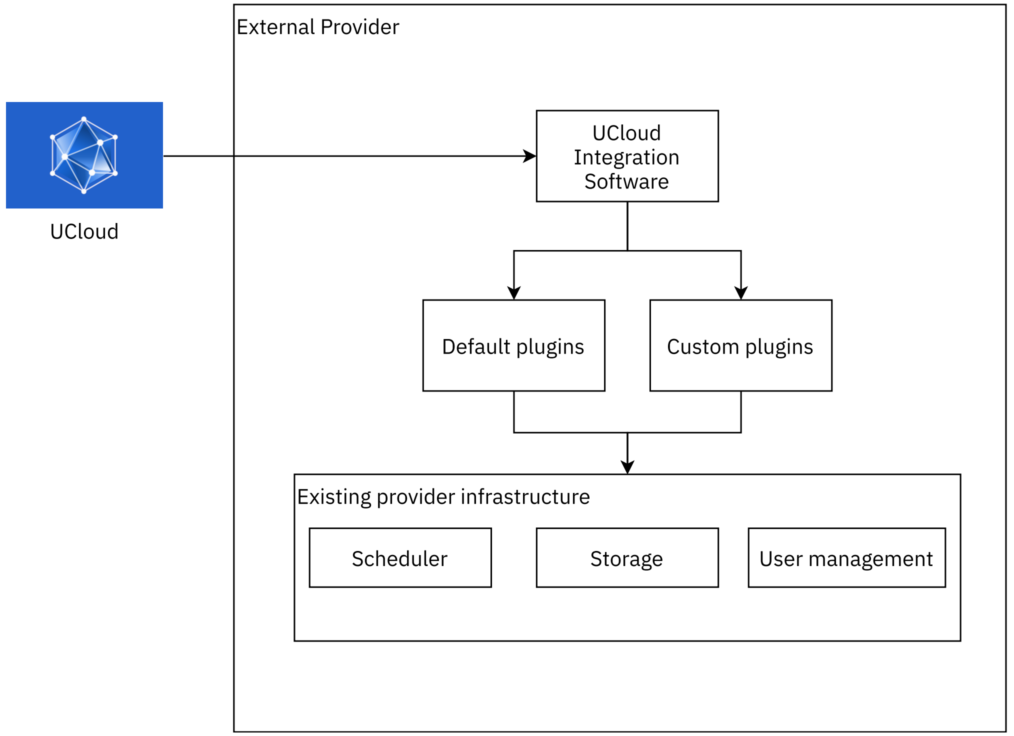
- Gives you full control
- HTTP and WebSockets
- This is how the integration software will be implemented

The integration software



- UCloud Integration Software
  - Service running at the provider
  - Single point-of-contact for UCloud
  - Delegates work to plugins
- Plugins
  - Invoked by the integration software
  - Runs adjacent to integration software
  - Interface not yet decided

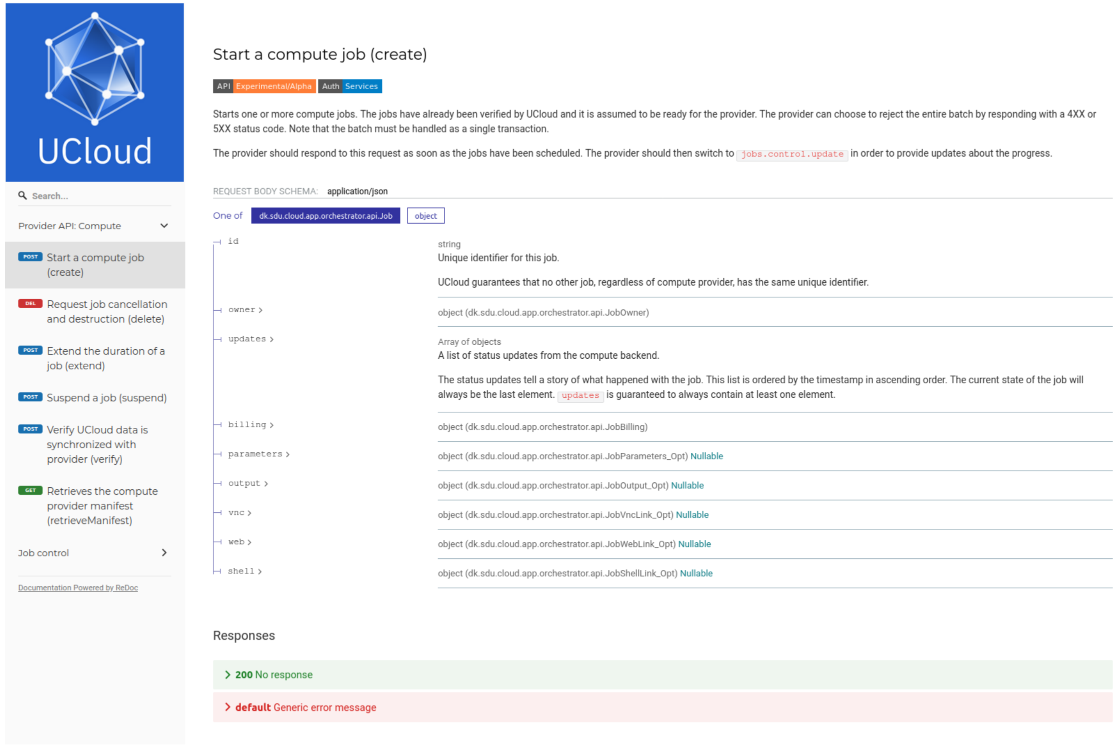
The integration software



- Default plugins
  - Maintained and provided by UCloud dev team
  - Provides integrations for common provider architectures
  - Examples:
    - Slurm
    - Storage
    - SSH keys
    - ...
- Custom plugins
  - Developed and maintained by external provider
  - Provides flexibility for the provider

Low-level API

- All between UCloud and the provider is done via HTTP and WebSockets
- Authentication uses JWT tokens in both directions
- This is all based on our existing stack



UCloud
Search...
Provider API: Compute
POST Start a compute job (create)
DEL Request job cancellation and destruction (delete)
POST Extend the duration of a job (extend)
POST Suspend a job (suspend)
POST Verify UCloud data is synchronized with provider (verify)
GET Retrieves the compute provider manifest (retrieveManifest)
Job control
Documentation Powered by ReDoc

Start a compute job (create)

API Experimental/Alpha Auth Services

Starts one or more compute jobs. The jobs have already been verified by UCloud and it is assumed to be ready for the provider. The provider can choose to reject the entire batch by responding with a 4XX or 5XX status code. Note that the batch must be handled as a single transaction.

The provider should respond to this request as soon as the jobs have been scheduled. The provider should then switch to `jobs.control.update` in order to provide updates about the progress.

REQUEST BODY SCHEMA: application/json

One of dk.sdu.cloud.app.orchestrator.api.Job object

`id`	string Unique identifier for this job. UCloud guarantees that no other job, regardless of compute provider, has the same unique identifier.
`owner ›`	object (dk.sdu.cloud.app.orchestrator.api.JobOwner)
`updates ›`	Array of objects A list of status updates from the compute backend. The status updates tell a story of what happened with the job. This list is ordered by the timestamp in ascending order. The current state of the job will always be the last element. `updates` is guaranteed to always contain at least one element.
`billing ›`	object (dk.sdu.cloud.app.orchestrator.api.JobBilling)
`parameters ›`	object (dk.sdu.cloud.app.orchestrator.api.JobParameters_Opt) Nullable
`output ›`	object (dk.sdu.cloud.app.orchestrator.api.JobOutput_Opt) Nullable
`vnc ›`	object (dk.sdu.cloud.app.orchestrator.api.JobVncLink_Opt) Nullable
`web ›`	object (dk.sdu.cloud.app.orchestrator.api.JobWebLink_Opt) Nullable
`shell ›`	object (dk.sdu.cloud.app.orchestrator.api.JobShellLink_Opt) Nullable

Responses

› **200** No response

› **default** Generic error message

User Mapping

1. User authenticates with UCloud
    - This uses one of the IdPs we support (one of which is WAYF)
    - This identity is mapped to a *local UCloud user*
2. User initiates connection with provider
3. UCloud contacts the provider (using the low-level API) and asks to initiate connection procedure
    - UCloud will include a reference to the local UCloud `user X`
4. Provider redirects user to local login
5. User authenticates as `user Y` with the provider
6. Provider stores a record of the mapping between UCloud `user X` and provider `user Y`

Resources and APIs

- Files
- Compute jobs (Interactive *and* batch)
- HTTP ingress
- Projects
- Virtual machines
- Block storage
- IP addresses/Networking
- SSH keys

Call to action

We need to hear from you in order for us to design and implement the integration software!

- Which APIs are you interested in using? Did we miss any?
- What does your current stack look like?
  - Scheduler (e.g. Slurm)
  - Accounting (e.g. Slurm)
  - Storage system (e.g. BeeGFS)
  - User management (e.g. NIS, LDAP, AD)
  - Custom scripts and workflows (e.g. project and user creation)
- Preferred scripting language (for custom plugins, e.g. Python, Ruby, Bash)

Work packages

This presentation was meant as an overview of UCloud's role in project 5. The work is split into 3 WPs:

- **Work package 1:** Web portal/GUI (M1-M18, improvements/bug fixing M18-M30)
    - Covers changes needed to the UCloud interface. A lot of this already exists today.
    - M1-M18: Development
    - M18-M30: Improvements/bug fixing
- Work package 2: Integration with HPC centers
- Work package 3: Integration with national data resources

Work packages

This presentation was meant as an overview of UCloud's role in project 5. The work is split into 3 WPs:

- Work package 1: Web portal/GUI (M1-M18, improvements/bug fixing M18-M30)
- **Work package 2:** Integration with HPC centers
  - Covers the provider API and integration software discussed in these slides
  - M1-M18: Provider API design
  - M6-M18: Development of API and integration software
  - M18-M30: Improvements/bug fixing
- Work package 3: Integration with national data resources

Work packages

This presentation was meant as an overview of UCloud's role in project 5. The work is split into 3 WPs:

- Work package 1: Web portal/GUI (M1-M18, improvements/bug fixing M18-M30)
- Work package 2: Integration with HPC centers
- **Work package 3:** Integration with national data resources
  - Not discussed explicitly today
  - Extends on the provider APIs to grant access to national data resources
  - M12-M34: Design and development