PROCEEDINGS OF SCIENCE

Supporting Native Cloud Tools in the EGI Federated Cloud via the FedCloud Client

Viet Tran[a,*]

[a] *Instiute of Informatics, Slovak Academy of Sciences*
Dubravska cesta 9, Bratislava, Slovakia

E-mail: viet.tran@savba.sk

The EGI Federated Cloud (FedCloud) is a multinational cloud system that seamlessly integrates community, private, and public clouds into a scalable computing platform dedicated to research. Each cloud within this federated infrastructure configures its Cloud Management Framework (CMF) based on its preferences and constraints. The inherent heterogeneity among cloud sites can pose challenges when attempting to use native cloud tools such as Terraform at the federation level. The FedCloud client, the official client of the EGI Federated Cloud, plays a crucial role in simplifying the use of these tools. It offers the following capabilities: creating a working environment for the tools; making the tools federation-aware; and selecting suitable resources and configurations for deployments. In essence, the FedCloud client serves as a valuable bridge, simplifying the use of native cloud tools within the EGI Federated Cloud environment. Its features contribute to a more user-friendly and efficient cloud computing experience, particularly when dealing with the diverse cloud infrastructure found within the federation.

International Symposium on Grids and Clouds (ISGC2024)
24 -29 March, 2024
Academia Sinica Computing Centre (ASGC), Institute of Physics, Academia Sinica
Taipei, Taiwan

[*] Speaker

PoS(ISGC2024)010

1. Introduction

The EGI Federated Cloud [1],[2] integrates community, private, and public clouds into a scalable research computing platform by pooling resources from a heterogeneous set of cloud providers using a single authentication and authorization framework. Each participating cloud provider in the federation manages its resources using its own Cloud Management Framework (CMF) [3], which essentially acts as a control panel for their cloud resources. These CMFs must integrate with the EGI Authentication and Authorization Infrastructure (AAI) [4] so users can access services with a single identity. Currently, only OpenStack CMF [12] is supported in the federation.

However, compatibility between resource centers ends there. Each center defines project names, virtual images, and virtual machine configurations (OpenStack flavors) differently. This inherent heterogeneity can be challenging when using native cloud tools like Terraform across the entire federation.

The FedCloud client is a user-friendly Python package that simplifies interacting with OpenStack services within the EGI infrastructure. It allows users to manage access tokens, browse available services, and execute commands on OpenStack sites.

This paper will explore how the FedCloud client can be used to support native cloud tools within the EGI Federated Cloud. The client will act as an intermediary, handling tasks like generating working environments, managing authentication and authorization, and selecting resources for these native cloud tools, ensuring they function flawlessly in this federated, heterogeneous environment.

2. The FedCloud client

The FedCloud client is a user-friendly command-line tool that serves as the primary interface for interacting with EGI Cloud infrastructure. It can interact with various services across the federation, including EGI Check-in [5], GOCDB [6], Secret Management Service [7], OpenStack sites, and more. Despite its rich functionality, the client is designed to be user-friendly and programmable for automation purposes. This section will provide an overview of the client's design principles and features.

2.1 Modular design

For easy development and maintenance, the client is designed to be modular. Each module is responsible for interacting with a specific service of the EGI Cloud infrastructure. Currently, five main modules are available in the client:

- **The checkin module** handles interactions with the EGI Check-in service for authentication and authorization. The module retrieves OpenID Connect (OIDC) access tokens from various token storages like oidc-agent and mytoken. It can also extract information such as the token's validity and lifespan, and VO memberships (including entitlements) from the access tokens. Since access tokens are needed for authenticating and accessing other services, this module is called by other modules.

- **The endpoint module** interacts with the service catalog (also known as Grid Operation Configuration Database, GOCDB). It is used for searching and retrieving service endpoints via GOCDB, such as listing the web-based Horizon dashboard of all or selected OpenStack sites. The module can also directly probe OpenStack sites and obtain unscoped or scoped tokens from the OpenStack Keystone service.
- **The sites module** manages site configurations. It maintains a mapping of sites and VOs and can translate site-VO abstractions to OpenStack Keystone endpoints and project IDs. More information is provided in subsection 2.2.
- **The openstack module** provides a wrapper for the native OpenStack command-line client and offers a high-level abstraction based on site/VO information. The module can also perform federation-wide operations, such as listing all VMs in all OpenStack sites within the entire federation for a specific VO with a single command.
- **The secret module** provides access to secrets stored in the Secret Management Service.

2.2 Site and VO abstraction

When users need to perform a command on multiple OpenStack sites, they need to specify the configuration of each site. The main parameters include Keystone endpoints and project IDs. These parameters can be difficult to remember, especially with a large number of sites in the EGI Cloud infrastructure. Therefore, the FedCloud client simplifies this process by replacing them with memorable site and VO names for all standard OpenStack commands.

The **sites** module in the FedCloud client manages and translates site and VO names to Keystone endpoints and project IDs. This functionality relies on YAML configuration files where site/VO are mapped to Keystone endpoints and project IDs. The default site configuration files are centrally managed on GitHub for all users. However, users have the ability to edit the files locally and create customized versions.

This abstraction significantly improves the user-friendliness of the FedCloud client. Users can easily execute standard OpenStack commands via FedCloud client, for example: listing virtual machine (VM) images in the vo.access.egi.eu VO on the IISAS-FedCloud site:

```
fedcloud openstack image list --site IISAS-FedCloud --vo vo.access.egi.eu
```

FedCloud client also provides simple commands for listing OpenStack sites (`fedcloud site list`) and VO memberships (`fedcloud token list-vos`) to help users recall site and VO names.

2.3 JSON outputs and scripting

The FedCloud client is designed for automation, so processing its outputs in scripts is emphasized from the beginning. Most commands in the client support JSON-formatted outputs, which are much easier for machines to process than free text and table formats. When combined with JSON processing tools like `jq`, complex actions like listing all OpenStack flavors with attached GPUs on all sites in the infrastructure can be performed in just one or two lines of a shell script:

```
fedcloud openstack flavor list --long --site ALL_SITES --vo vo.access.egi.eu --json-output |
```

```
jq -r 'map(.Result = (.Result| map(select(.Properties."Accelerator:Type" == "GPU"))))'
```

The `fedcloud` command in first line in the example above prints all flavors on all sites for the *vo.access.egi.eu* virtual organization. The `jq` command in the second line filters only flavors with GPU accelerators. More examples are available at [8].

2.4 Python library

The FedCloud client source code is written in Python. Each module is typically divided into two parts:

- Frontend: This part processes user input parameters, calls backend functions to perform the actual operations, and displays the results.
- Backend: These functions are designed with minimal dependencies on other modules and avoid unintended side effects. This allows them to be reused as Python libraries within other tools.

  Here's an example of how to execute an OpenStack command on a specific site:

```
from fedcloudclient.openstack import fedcloud_openstack

error_code, result = fedcloud_openstack(
    token = access_token,
    site = “IISAS-FedCloud”,
    vo = “vo.access.egi.eu”,
    command = “image list”,
)
```

The examle above demonstrates how easy and elegant it is to execute an OpenStack command with site and VO abstraction in Python using the FedCloud client library. The full API of the library is available at [9].

2.5 Token management

Many operations in the EGI Cloud infrastructure require authentication and authorization. This is achieved using OpenID Connect (OIDC) access tokens obtained from the EGI Check-in service. These access tokens are sensitive credentials and must be handled securely. They also have short lifespans and cannot be extended. Managing access tokens effectively is a crucial aspect of the FedCloud client.

The FedCloud client supports two secure storage options for access tokens:

- oidc-agent: oidc-agent [10] is a set of tools designed to manage OIDC tokens and facilitate their use from the command line. The software requires installation on the client and runs as a daemon. Tokens are stored securely within the oidc-agent daemon in an encrypted format. The FedCloud client retrieves access tokens from oidc-agent through its API. Users can specify the oidc-agent account name holding the desired tokens using the `--oidc-account-name` option. This allows the FedCloud client to utilize the access tokens directly without storing them in plain text anywhere.
- Mytoken: Mytoken [11] is a web service that provides a secure and convenient way to obtain and manage OIDC access tokens. These tokens can be used for extended periods

and across multiple devices. Access tokens are retrieved from the service using unique identifiers called "mytokens". Similar to oidc-agent, the FedCloud client can retrieve access tokens from the Mytoken service and use them directly for authentication with OpenStack sites, eliminating the need to store the tokens in plain text.

3. Supporting cloud-native tools

Most cloud-native tools are designed to work within a single cloud site. They lack awareness of federation features like service catalogs, single sign-on, and virtual organizations. This can be inconvenient for users who want to migrate their services between OpenStack sites within the EGI Cloud federation. Traditionally, this requires manually editing site configuration files and changing parameters for the new site.

This section details how the FedCloud client can be used to support and integrate cloud-native tools like the OpenStack client or Terraform with the EGI Cloud federation, enabling interaction with multiple sites simultaneously.

3.1 Setting environment variables for external tools

Cloud-native tools typically rely on environment variables or configuration files to access site information. Essential site parameters include:

- Site location: the URL of the Keystone service for the OpenStack site ,
- Local project ID: the ID of the local project associated with the VO,
- Authentication credentials: the authentication method (username/password or access token) and corresponding credentials.

Manually editing these parameters can be inconvenient, as they are often lengthy and difficult to remember. The FedCloud client automates this process by setting all necessary parameters, allowing cloud-native tools to work seamlessly within the EGI Cloud federation.

The `fedcloud site env` command outputs environment variables with the appropriate values for the target OpenStack site and VO, ready to be used in scripting:

```
$ fedcloud site env --site IISAS-FedCloud --vo eosc-synergy.eu
export OS_AUTH_URL="https://cloud.ui.savba.sk:5000/v3/"
export OS_AUTH_TYPE="v3oidcaccesstoken"
export OS_IDENTITY_PROVIDER="egi.eu"
export OS_PROTOCOL="openid"
export OS_PROJECT_ID="51f736d36ce34b9ebdf196cfcabd24ee"
```

Here's an example of how to execute an OpenStack native client command on a specific site:

```
$ eval `fedcloud site env --site IISAS-FedCloud --vo eosc-synergy.eu`
$ openstack image list
```

In this way, you can leverage cloud-native tools like OpenStack command-line client within the EGI Cloud federation using site and VO abstraction.

3.2 Resource selection

Creating virtual machines across OpenStack sites within the EGI Cloud federation can be challenging due to heterogeneity in virtual machine configurations (flavors). Each site assigns unique names or IDs to flavors with the same hardware specifications. This inconsistency makes it difficult to migrate deployment scripts between sites.

To address this issue, the FedCloud client offers a new resource selection module. This module allows users to specify resource requirements in a flexible manner, even using logical operators like 'and' for complex criteria. For example, the following command searches for a flavor with two CPU cores and more than 4GB of memory on the *IISAS-FedCloud* site for the *vo.access.egi.eu* VO:

```
fedcloud select flavor --site IISAS-FedCloud --vo vo.access.egi.eu
      --flavor-specs "VCPUS==2 & RAM≥4096"
```

If multiple flavors meet the specified conditions, the command outputs them sorted by size, with the smallest flavor listed first. Similar commands are available for selecting images, networks, and other resources within the EGI Cloud federation.

4. Conclusion

The EGI Cloud infrastructure offers a rich ecosystem with many tools and services that native cloud tools might not directly support. The FedCloud client acts as an intermediary, enabling interactions between these tools and the ecosystem. It manages access tokens, browses service catalogs, interacts with the Dynamic DNS service, retrieves secrets from Vault, and more. Most importantly, it provides user-friendly abstractions of cloud resources through site and VO constructs, enabling the creation of working environments and the selection of resources. This allows native cloud tools to integrate seamlessly within the EGI Cloud federation, simplifying the overall process.

Acknowledgment

This work is supported by APVV grant No. APVV-20-0571 and VEGA grant No. 2/0131/23.

References

[1] Enol Fernandez-del-Castillo et al, “The EGI Federated Cloud e-Infrastructure”, Procedia Computer Science, vol. 68, pp. 196-205, 2015. https://doi.org/10.1016/j.procs.2015.09.235.

[2] Spiga, Daniele, et al. "The DODAS Experience on the EGI Federated Cloud." EPJ Web of Conferences. Vol. 245. EDP Sciences, 2020. https://doi.org/10.1051/epjconf/202024507033

[3] Marcus Hardt and Viet Tran, “EOSC SYNERGY HANDBOOK: A Handbook targeted at Computer Centre Management, Administrators, and Users of the Infrastructure”, 2022, http://hdl.handle.net/10261/280037.

[4] Calatrava, Amanda, et al. "A survey of the European Open Science Cloud services for expanding the capacity and capabilities of multidisciplinary scientific applications." Computer Science Review 49 (2023): 100571. https://doi.org/10.1016/j.cosrev.2023.100571

[5] EGI Check-in home. https://aai.egi.eu/. Accessed: 16 September 2024.

[6] GOCDB. https://goc.egi.eu/portal/index.php. Accessed: 16 September 2024.

[7] Secret management service. https://docs.egi.eu/users/security/secrets-store/. Accessed: 16 September 2024.

[8] Cheat sheet – fedcloudclient documentation. https://fedcloudclient.fedcloud.eu/cheat.html#mapping-and-filtering-results-from-openstack-commands. Accessed: 16 September 2024.

[9] FedCloud client API references. https://fedcloudclient.fedcloud.eu/fedcloudclient.html. Accessed: 16 September 2024.

[10] Zachmann, Gabriel. "OIDC-Agent: Managing OpenID Connect Tokens on the Command Line." (2018). https://dl.gi.de/items/d2f7214e-f4ba-4ea0-b6cb-efd3d9d7ce57

[11] Mytoken, https://mytoken-docs.data.kit.edu/. Accessed: 16 September 2024.

[12] Open Source Cloud Computing Infrastructure – OpenStack. https://www.openstack.org/. Accessed: 16 September 2024.