



Project Title	Hybrid Eco Responsible Optimized European Solution
Project Acronym	HEROES
Grant Agreement No.	956874
Start Date of Project	01.03.2021
Duration of Project	24 Months
Project Website	heroes-project.eu

D3.3 – Initial Platform Mockup

Work Package	WP3
Lead Author (Org)	Jorik Remy (UCit)
Contributing Author(s) (Org)	Anaëlle Dambreville (UCit) Benjamin Depardon (UCit) Sablin Amon (UCit)
Reviewed by	Julita Corbalan (BSC) Jose E. Torres (HPCNow!) Davide Pastorino (Dolt) Emanuele Viale (Dolt)
Approved by	Name (organization)
Due Date	28/02/2022
Date	13/04/2022
Version	V2.0

Dissemination Level

<input checked="" type="checkbox"/>	PU: Public
<input type="checkbox"/>	PP: Restricted to other programme participants (including the Commission)
<input type="checkbox"/>	RE: Restricted to a group specified by the consortium (including the Commission)
<input type="checkbox"/>	CO: Confidential, only for members of the consortium (including the Commission)



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

Versioning and contribution history

Version	Date	Author	Notes
0.1	25.02.2022	Benjamin Depardon (UCit)	TOC and V0.1
0.2	09.03.2022	Jorik Remy (UCit)	Add: Executive Summary, Updated architecture, Containerized Platform and Automated CI/CD
0.3	10.03.2022	Jorik Remy (UCit)	Add: Figures for Use Cases / Updated Architecture
0.4	11.03.2022	Jorik Remy (UCit)	Add: Figures
0.5	14.03.2022	Jorik Remy (UCit)	Add: Fill Use Cases Update: Introduction
0.6	16.03.2022	Jorik Remy (UCit)	Update: Platform as Organized User / Administrator
0.7	19.03.2022	Jorik Remy (UCit)	Update: Architecture
0.8	22.03.2022	Jorik Remy (UCit)	Add: Containerized Platform Services / Conclusion
0.9	23.03.2022	Jorik Remy (UCit)	Add: Service containerization / deployment
1.0	24.03.2022	Jorik Remy (UCit)	Update: Architecture / Conclusion
1.1	30.03.2022	Jorik Remy (UCit)	Review corrections
1.2	01.04.2022	Jorik Remy (UCit)	Review corrections
1.3	05.04.2022	Jorik Remy (UCit)	Review corrections
2.0	13.04.2022	Corentin LEFEVRE (Neovia)	Final version approved by the Management Board

Disclaimer

This document contains information which is proprietary to the HEROES Consortium. Neither this document nor the information contained herein shall be used, duplicated or communicated by any means to a third party, in whole or parts, except with the prior consent of the HEROES Consortium.



Table of Contents

List of Figures	3
References and Applicable Documents	4
List of Acronyms and Abbreviations	5
Executive Summary	6
1 Introduction	7
2 Mockup Platform Use Cases	7
2.1 Platform access as User of an Organization	9
2.1.1 Authentication	9
2.1.2 Data Management	11
2.1.3 Workflow and Job Management	13
2.2 Platform access as an Administrator of an Organization	17
2.2.1 User creation process	17
2.2.2 User deletion process	18
2.2.3 Examples of actions on users	18
2.3 Platform access as a HEROES Administrator	20
2.3.1 Organization creation process	20
2.3.2 Organization deletion process	20
2.3.3 Examples of actions on organizations	21
3 Updated Architecture	22
4 Containerized Platform and Automated CI/CD	25
4.1 Platform Development Environment	25
4.2 Containerized Platform Services	26
4.2.1 Available services	26
4.2.2 Service containerization and deployment	27
5 Conclusion	28

List of Figures

FIGURE 1. INTERACTION BETWEEN THE SERVICE GATEWAY AND INTERNALS APIs WITHIN THE HEROES PLATFORM	8
FIGURE 2. FASTAPI – GLOBAL VIEW	8
FIGURE 3. USER: IDENTITY MANAGEMENT FUNCTIONS	9
FIGURE 4. USER: DATA MANAGEMENT FUNCTIONS	11
FIGURE 5. USER: WORKFLOW MANAGEMENT FUNCTIONS	13
FIGURE 6. ORGANIZATION ADMIN FUNCTIONS.....	17
FIGURE 7. HEROES ADMIN FUNCTIONS.....	20
FIGURE 8. HEROES UPDATED ARCHITECTURE	24
FIGURE 9. DO IT SYSTEMS LAB WITH ITS GITLAB, KUBERNETES AND ARGOCD PLATFORM TO SUPPORT THE HEROES PLATFORM DEVELOPMENT AND DEPLOYMENT.....	25
FIGURE 10. ARGOCD – SERVICES VIEW	26
FIGURE 11. RABBITMQ – WORKERS DEPLOYED WITHIN THE HEROES INITIAL PLATFORM	27
FIGURE 12. ARGOCD – CREATION SERVICE: SOURCE PATH AND REPOSITORY SELECTION OF THE SERVICE	28



References and Applicable Documents

- [1] HEROES Initial Platform Mockup demonstration video, April 2022, <https://www.youtube.com/watch?v=faCE4rJdYkI>
- [2] FastAPI, FastAPI documentation, April 2022, <https://fastapi.tiangolo.com/>
- [3] Keycloak, official website, April 2022, <https://www.keycloak.org/>
- [4] MinIO, project website, April 2022, <https://min.io/>
- [5] Nextflow, official website, April 2022, <https://www.nextflow.io/>
- [6] RabbitMQ, official website, April 2022, <https://www.rabbitmq.com/>
- [7] Kubernetes, official website, April 2022, <https://kubernetes.io/>
- [8] ArgoCD, ArgoCD documentation, April 2022, <https://argo-cd.readthedocs.io/>
- [9] Gitlab, official website, April 2022, <https://about.gitlab.com/>



List of Acronyms and Abbreviations

Terminology/Acronym	Description
API	Application Programming Interface
CSP	Cloud Service Provider
DoA	Description of Action
EC	European Commission
GA	Grant Agreement to the project
HPC	High-Performance Computing
KPI	Key Performance Indicator
ML	Machine-Learning
UI	User Interface
WID	Workflow ID
WTID	Workflow Template ID



Executive Summary

The HEROES Project is aiming at developing an innovative European software solution allowing industrial and scientific user communities to easily submit complex Simulation and ML (Machine Learning) workflows to HPC (High Performance Computing) Data Centres and Cloud Infrastructures. It will allow them to take informed decisions and select the best platform to achieve their goals on time, within budget and with the best energy efficiency.

This document presents an approach for the realization of the HEROES Initial Platform Mock-Up. It includes the updates and improvement of the high-level architecture.

The development of this document is closely linked to the ongoing developments of WP2 and WP3 of the HEROES Project.

The architecture, schemas and tools described in this document will be constantly reviewed and adapted to satisfy all emergent needs of the HEROES platform throughout the life of the Project.

This document contains all essential information used to drive the implementation tasks during the development of the HEROES Initial Platform.



1 Introduction

The purpose of this document is to provide information about the end-to-end development and deployment of a first Initial Platform Mockup.

The objective of the “Initial Platform Mockup” was to create a platform, based on the HEROES architecture platform available within the deliverable “D3.1 – Architecture Design” to provide basic version of the actions designed in the previous documents, which includes:

- Identity Management
 - The capacity of the platform to provide an authentication method to the Identity Management module from the UI/API (User Interface / Application Programme Interface) module for users registered within organization of the platform.
 - The capacity to provide basic authorizations to the users of the platform corresponding to their organization and roles.
- Data Management
 - The capacity of the platform to provide a data management method, including the possibility to upload, download and move data within the platform by the authenticated and authorized users.
- Workflow and Job Management
 - The capacity to manage, submit, monitor and any other basic action by the Workflow and Job Management module depending on the authenticated user requests.
- Platform Administration
 - The capacity to manage organizations (e.g.: creation, deletion) and users within the organizations by the authenticated and privileged users through the UI/API module

These basic actions, available within the HEROES Initial Platform, are detailed in the next sections as key use cases that can be achieved within the platform. Accompanying this document, a video [1] introduces and demonstrates some of these “Use Cases” with the Initial Platform Mockup.

2 Mockup Platform Use Cases

The initial Mockup Platform is an aggregation of the HEROES different modules mocked-up to demonstrate the HEROES platform functionalities. It gives the possibility to handle workflows and job submission on conventional HPC infrastructures as well as Cloud Service Providers (CSP) for their users within the platform.



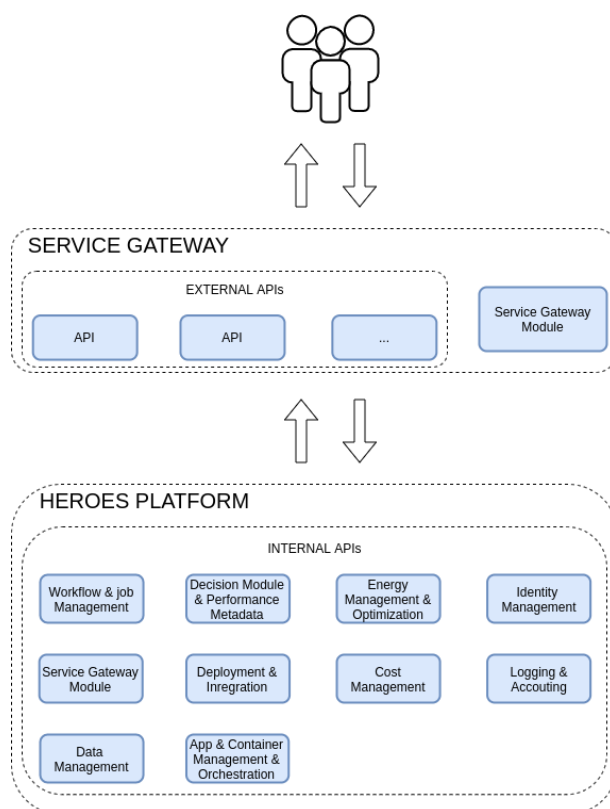


Figure 1. Interaction between the Service Gateway and Internals APIs within the HEROES Platform

The “UI/API module” allows the users and the administrators to perform basic actions over the Initial Platform through the “Service Gateway”, using the Python FastAPI [2] module and a specific HEROES API code developed within the HEROES project in Python. That API integrates a detailed documentation split in multiple parts to clarify the possible actions (see Figure 2).

Identity Management	Authentication and authorization functions	▼
Data Management	Data Management functions	▼
Workflow Management	Workflow Management functions	▼
Organization Admin	Organization Administration functions	▼
HEROES Admin	Platform Administration functions	▼
Schemas		▼

Figure 2. FastAPI – Global view

As described in the following sub sections 2.1 to 2.3, the “Use Cases” have been split between three points of view to simplify the role approach before involving more rights granularity. The points of view are resumed by:



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

- The HEROES organization side
- The customer organization side, which includes
 - The “user” of the organization (e.g.: engineer, researcher, scientist)
 - The “administrator” of the organization (e.g.: workflow manager, project manager)

The APIs presented in this document are not in their final format. The presentation details their current state at the time of the Initial Platform Mockup. Additional APIs will be designed and added in the next steps of the HEROES project to integrate the modules (e.g.: Cost Management module, Logging & Accounting module) not implemented in the Initial Platform Mockup.

2.1 Platform access as User of an Organization

The “Organization Users” are able to realize various actions of data management and workflow execution depending on their access rights and projects.

2.1.1 Authentication

The authentication provided by the Identity Management module (see Figure 3) is embedding Keycloak [3] to manages the user accounts of each organization. The authentication requires: valid organization name, username, and password to confirm the user’s identity, authorizations, and responds to the user with a token.

The token retrieved after a successful authentication is needed to perform any actions through the HEROES API, depending on the user’s authorizations and rights.

Identity Management Authentication and authorization functions	
POST	/organization/auth/login Login the user after UserCredentials check
POST	/organization/auth/logout Logout the authenticated user
GET	/organization/auth/refresh_token Refresh the access token of the authenticated user
GET	/organization/auth/me List all information available about the authenticated user

Figure 3. User: Identity Management functions

2.1.1.1 Example of user login

The following is an example of the user login request to the UI/API module with valid credentials from the Identity Management module.

```
curl POST https://fastapi.heroes.doit.priv/organization/auth/login
-H 'accept: application/json'
```



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

```
-H 'Content-Type: application/json'
-d '{
    "organization": "ucit",
    "username": "jremy",
    "password": "*****"
}'
```

Response:

```
{
  "access_token": "eyJhbGciOi..._EzxN8EXA",
  "expires_in": 3600,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOi...Kg06DqO9-E",
  "token_type": "Bearer",
  "id_token": "eyJhbGciOi...Zk7B7Pqg",
  "session_state": "19574e0a-8c34-4660-b840-6852a213b60b",
  "scope": "openid profile email tpm",
  "organization": "ucit"
}
```

Any subsequent calls to the UI/API module must be done with a valid token, otherwise the user will get an error and will not be able to access the target functionality. Unless otherwise stated, the following examples are presented as invocation with a valid token provided by the Identity Management module after successful login.

2.1.1.2 Example of user information

The following is an example of the user information request to the UI/API module.

```
curl -X GET https://fastapi.heroes.doit.priv/organization/auth/me
-H 'accept: application/json'
-H 'token : {eyJhbGciOi..._EzxN8EXA}'
```

Response:

```
{
  "sub": "dc4bfff76-0970-46d4-b2d3-fe92eff84b66",
  "minioPolicy": [
    "userPolicy"
  ],
  "email_verified": true,
  "groups": [],
  "preferred_username": "jremy",
  "given_name": "",
  "family_name": "",
  "email": "jremy@ucit.fr"
}
```



2.1.2 Data Management

The data management provided by the Data Management module (see Figure 4) is embedding MinIO [4] to allows each organization to store its data to its own versioned object storage.

Data Management <small>Data Management functions</small>		^
GET	/organization/data/list List all buckets available for the authenticated user	▼
POST	/organization/data/bucket Create a new bucket in organization minio server	▼
DELETE	/organization/data/bucket Delete bucket from organization minio server	▼
GET	/organization/data/bucket/{bucket}/list List all objects presents in the target bucket of the authenticated user	▼
GET	/organization/data/download Download file from bucket	▼
POST	/organization/data/upload Upload file to bucket	▼

Figure 4. User: Data Management functions

2.1.2.1 Example of bucket list

The following is an example of the user requesting the list of its accessible buckets to the UI/API module.

```
curl -X GET https://fastapi.heroes.doit.priv/organization/data/list
-H 'accept: application/json'
-H 'token : {eyJhbGciOiJIUzI1NiIsInR5cGE6IjoiYzYxN8EXA}'
```

The UI/API modules responds to this request by a list of buckets accessible by the authenticated user:

```
[
  {
    "bucket_name": "jremy",
    "creation_date": "2022-03-07T15:05:59.771000+00:00"
  },
  {
    "bucket_name": "projectA",
    "creation_date": "2022-03-07T15:10:12.462000+00:00"
  }
]
```

2.1.2.2 Example of data listing

The following is an example of the user requesting the list of its data within one of its accessible buckets (in green) to the UI/API module.

```
curl -X GET https://fastapi.heroes.doit.priv/organization/data/bucket/jremy/list
-H 'accept: application/json'
-H 'token : {eyJhbGciOiJIUzI1NiIsInR5cGE6IjoiYzYxN8EXA}'
```



The UI/API modules responds to this request by a detailed list of files presents in the selected bucket:

```
[
  {
    "bucket_name": "jremy",
    "content_type": null,
    "etag": "6a909b8056edc869ff57f33bbce4f5ba",
    "is_delete_marker": false,
    "is_dir": false,
    "is_latest": null,
    "last_modified": "2022-03-22T11:23:45.262000+00:00",
    "metadata": {},
    "object_name": "textFile_jremy.txt",
    "owner_id": "02d6176db174dc93cb1b899f7c6078f08654445fe8cf1b6ce98d8855f66bdbf4",
    "owner_name": "minio",
    "size": 30,
    "storage_class": "STANDARD",
    "version_id": null
  }
]
```

2.1.2.3 Example of data upload

The following is an example of the user uploading a data file ("textFile_jremy.txt" in this example) to one of his buckets ("jremy") through the UI/API module.

```
curl POST https://fastapi.heroes.doit.priv/organization/data/upload?&bucket=jremy
-H 'accept: application/json'
-H 'token : {eyJhbGciOiJIUzI1NiIsInR5cGE6ImVudCIsImVudC5kbm90b8056edc869ff57f33bbce4f5ba'
-H 'Content-Type: multipart/form-data'
-F 'file=@textFile_jremy.txt;type=application/octet-stream'
```

The UI/API modules responds to this request by the description of the newly uploaded data to the user's bucket:

```
{
  "_bucket_name": "jremy",
  "_object_name": "textFile_jremy.txt",
  "_version_id": null,
  "etag": "6a909b8056edc869ff57f33bbce4f5ba",
  "_http_headers": {
    "Accept-Ranges": "bytes",
    "Content-Length": "0",
    "Content-Security-Policy": "block-all-mixed-content",
    "ETag": "6a909b8056edc869ff57f33bbce4f5ba",
    "Server": "MinIO",
    "Strict-Transport-Security": "max-age=31536000; includeSubDomains",
    "Vary": "Origin, Accept-Encoding",
    "X-Amz-Request-Id": "16DEB0B3DDE8D29A",
    "X-Content-Type-Options": "nosniff",
    "X-Xss-Protection": "1; mode=block",
    "Date": "Tue, 22 Mar 2022 11:33:20 GMT"
  },
  "_last_modified": null,
  "_location": null
}
```



2.1.2.4 Example of data download

The following is an example of the user downloading an object (file) from one of his buckets (in green) through the UI/API module with a valid token provided by the Identity Management module after successful login.

```
curl -o textFile_jremy.txt -X GET
https://fastapi.heroes.doit.priv/organization/data/download?&bucket=jremy&file=textFile_jremy.txt
-H 'accept: application/json'
-H 'token : {eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZXQ6Ij09EzXN8EXA}'
```

The UI/API modules responds to this request by transferring the selected file.

2.1.3 Workflow and Job Management

The Workflow Management provided by the Workflow and Job Management module allows the organization's user to list, select and submit workflows to an HPC centre or CSP.

The module (see Figure 5) is embedding Nextflow [5] as a workflow manager (see deliverable D2.2) framework instead of Ryax (as was previously selected in deliverable D3.1).

Workflow Management <small>Workflow Management functions</small>	
GET	/organization/workflow/list List all workflows available for the authenticated user
GET	/organization/workflow/running List all running workflows available for the authenticated user
POST	/organization/workflow/submit Submit a workflow
GET	/organization/workflow/{workflow_uid} Get information and status about a submitted workflow
DELETE	/organization/workflow/{workflow_uid} Cancel a running workflow

Figure 5. User: Workflow Management functions

The execution of a submitted workflow is done through the following steps:

- The user submits a selected workflow template id (wtid – generated by HEROES) corresponding to a unique workflow with FastAPI
- FastAPI receives the selected workflow template with parameters and ask to the Decision Module the best execution place and configuration for the workflow
- FastAPI sends a message to the Workflow and Job Management module with the submitted workflow configuration generated by the Decision Module and generate a dynamically unique workflow id (wid)
- The Workflow and Job Management module receives the selected workflow and a list of task / cluster tuples
 - Nextflow executes each task of the selected workflow to the correct cluster (through SSH)



- Moves the input data from the organization / user / dynamic workflow id (generated by HEROES) bucket of the user at the beginning of the task
- Moves the output data to the organization / user / dynamic workflow id bucket (generated by HEROES) of the user at the end of the task
- Nextflow sends a message to the Service Gateway to signal the end of the workflow

2.1.3.1 Example of workflow listing

The following is an example of the user requesting the list of its accessible workflows to the UI/API module.

```
curl -X GET https://fastapi.heroes.doit.priv/organization/workflow/list
-H 'accept: application/json'
-H 'token : {eyJhbGciOi..._EzxN8EXA}'
```

The UI/API modules responds to this request by a list of workflow accessible by the authenticated user.

```
[
  {
    "id": "wf1",
    "name": "workflow_1",
    "description": "This workflow is described by a string"
  },
  {
    "id": "wf2",
    "name": "workflow_2",
    "description": "This workflow is described by a string"
  }
]
```

2.1.3.2 Example of workflow submission

The following is an example of a user submitting the selected “wf1” workflow template id (wtid) to the UI/API module.

```
curl -X POST https://fastapi.heroes.doit.priv/organization/workflow/submit
-H 'accept: application/json'
-H 'token : {eyJhbGciOi..._EzxN8EXA}'
-d '{
  "workflow_template_id": "wf1",
  "parameters": {}
}'
```

Response:

```
{
  "wid": "*****",
}
```



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

```

"owner": {
  "organization": "ucit",
  "user": "jremy"
},
"configuration": {
  "wtid": "wf1",
  "name": "workflow_1"
},
"status": "SUBMIT_IN_PROGRESS"
}

```

As shown in the previous example, a workflow id (wid) is generated after the submission of a workflow. This “wid” can be used by an authenticated and authorized user to monitor the execution, retrieve the output data or information of the submitted workflow.

2.1.3.3 Example of workflow running list request

The following is an example of a user requesting the list of his accessible running workflow to the UI/API module.

```

curl -X GET https://fastapi.heroes.doit.priv/organization/workflow/running
-H 'accept: application/json'
-H 'token : {eyJhbGciOiJIeXN8EzN8EXA}'

```

The UI/API modules responds to this request by a list of running workflow accessible by the authenticated user:

```

[
  {
    "wid": "*****",
    "owner": {
      "organization": "ucit",
      "user": "jremy"
    },
    "configuration": {
      "wtid": "wf1",
      "name": "workflow_1",
    }
  },
  {
    "wid": "*****",
    "owner": {
      "organization": "ucit",
      "user": "jremy"
    },
    "configuration": {
      "wtid": "wf2",
      "name": "workflow_1",
    }
  }
]

```



2.1.3.4 Example of submitted workflow information request

The following is an example of a user requesting information about a specific workflow previously submitted to the UI/API module.

```
curl -X GET https://fastapi.heroes.doit.priv/organization/workflow/{workflow_id}
-H 'accept: application/json'
-H 'token : {eyJhbGciOi4iLl_EzxN8EXA}'
```

Reponse:

```
{
  "wid": "*****",
  "owner": {
    "organization": "ucit",
    "user": "jremy"
  },
  "configuration": {
    "wtid": "wf1",
    "name": "workflow_1",
  },
  "status": "RUN_IN_PROGRESS",
  "submit_time": "22/02/2022:22H11M22",
  "run_time": "22/02/2022:22H22M22"
}
```

2.1.3.5 Example of workflow cancel request

The following is an example of a user cancelling his running workflow to the UI/API module.

```
curl -X DELETE
-H 'accept: application/json'
-H 'token : {eyJhbGciOi4iLl_EzxN8EXA}'
```

Reponse:

```
{
  "wid": "*****",
  "owner": {
    "organization": "ucit",
    "user": "jremy"
  },
  "configuration": {
    "wtid": "wf1",
    "name": "workflow_1",
  },
  "status": "CANCEL IN PROGRESS",
  "submit_time": "22/02/2022:22H11M22",
  "run_time": "22/02/2022:22H22M22",
  "end_time": "22/02/2022:22H33M22"
}
```



2.2 Platform access as an Administrator of an Organization

The “Organization Administrators” are able to realize various actions for their organization, as the creation of a new users, groups and roles as so as any needed action involving the users, groups and roles in their own organization. (see Figure 6)

Organization Admin <small>Organization Administration functions</small>		^
GET	/organization/admin/users/ List users	▼
POST	/organization/admin/users/ Create new user	▼
GET	/organization/admin/users/{username} Get user information	▼
DELETE	/organization/admin/users/{username} Delete user	▼
PATCH	/organization/admin/users/{username} Update user	▼
GET	/organization/admin/groups/ List groups	▼
POST	/organization/admin/groups/ Create group	▼
GET	/organization/admin/groups/{groupname} Get group information	▼
DELETE	/organization/admin/groups/{groupname} Delete group	▼
PATCH	/organization/admin/groups/{groupname} Update group	▼
GET	/organization/admin/roles/ List roles	▼
POST	/organization/admin/roles/ Add role to user	▼
GET	/organization/admin/roles/{rolename} Get role information	▼

Figure 6. Organization Admin functions

2.2.1 User creation process

The user creation can be requested by a HEROES’ administrator to the UI/API module. This process requires arguments such:

- The organization of the future user
- The username
- The user mail address
- Optionally one-time password for the future user also a one-time password is generated by the platform

The creation process includes:

- User creation within Keycloak with the Identity Management module
- User right and policies attributions
 - MinIO policy allowing the user to manage his own bucket



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

- User bucket creation within its MinIO server organization with the Data Management module

2.2.2 User deletion process

The user deletion can be requested by a HEROES administrator to the UI/API module. This process requires arguments such:

- The organization of the user
- The username

The deletion process includes:

- User deletion within Keycloak with the Identity Management module
- User bucket deletion of its MinIO server organization with the Data Management module

2.2.3 Examples of actions on users

The following examples detail the actions available for the privileged users over the users of its organization and the HEROES administrators over other organizations.

2.2.3.1 Example of user creation

The following is an example of privileged user requesting the creation of a new user of an organization to the UI/API module.

```
curl -X POST
  https://fastapi.heroes.doit.priv/organization/admin/users/?user_name=demouse
r&user_email=demouser@ucit.fr&user_password=UCit2022!!&organization=ucit
-H 'accept: application/json'
-H 'token : {eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTUzN8EXA}'
```

Response:

```
{
  "id": "*****",
  "createdTimestamp": 1646664587139,
  "username": "demouser",
  "enabled": true,
  "firstName": "",
  "lastName": "",
  "email": "jremy@ucit.fr",
  "attributes": {
    "minioPolicy": [
      "userPolicy"
    ]
  }
}
```



2.2.3.2 Example of user listing

The following is an example of privileged user requesting the user list of an organization to the UI/API module.

```
curl -X GET https://fastapi.heroes.doit.priv/organization/admin/users
-H 'accept: application/json'
-H 'token : {eyJhbGciOi..._EzN8EXA}'
```

The UI/API modules responds to this request by a list of the organization users with detailed information including but not limited to:

```
[
  {
    "id": "*****",
    "createdTimestamp": 1646664587139,
    "username": "jremy",
    "enabled": true,
    "firstName": "",
    "lastName": "",
    "email": "jremy@ucit.fr",
    "attributes": {
      "minioPolicy": [
        "userPolicy"
      ]
    }
  },
  {
    "id": "*****",
    "createdTimestamp": 1648116368935,
    "username": "demouser",
    "enabled": true,
    "firstName": "",
    "lastName": "",
    "email": "demouser@ucit.fr",
    "attributes": {
      "minioPolicy": [
        "userPolicy"
      ]
    }
  }
]
```



2.3 Platform access as a HEROES Administrator

The “HEROES administrators” will be able to realize various actions over the platform (see Figure 7), as the creation of a new organization (tenant) as so as any needed action involving the organization users.

HEROES Admin Platform Administration functions	
GET	/heroes/admin/organizations/ List Organizations
POST	/heroes/admin/organizations/ Create a new organization
DELETE	/heroes/admin/organizations/ Delete organization
GET	/heroes/admin/organizations/{organization_target} Read Organization
GET	/heroes/admin/organizations/{organization_id}/clients/ Read Clients
POST	/heroes/admin/organizations/{organization_id}/clients/ Create Client For Organization
GET	/heroes/admin/organizations/{organization_id}/servers/ List Organization Servers
POST	/heroes/admin/organizations/{organization_id}/servers/ Create a new organization server
DELETE	/heroes/admin/organizations/{organization_id}/servers/ Delete organization

Figure 7. HEROES Admin functions

2.3.1 Organization creation process

The creation of an organization within the HEROES Initial Platform is an aggregation of multiple tasks launched when a HEROES administrator requests a new organization to the UI/API module. Those tasks depend on the required services associated to each organization, including:

- The creation of the organization within Identity Management module
 - Including a first “root” user based on the mail address of the organization’s owner
- The creation of a MinIO server independent to store the data of the organization within the Data Management module

2.3.2 Organization deletion process

The deletion of an organization within the HEROES Initial Platform is an aggregation of multiple tasks launched when a HEROES administrator requests the deletion of a specific organization to the UI/API module. Those tasks depend on the required services associated to each organization, including:

- The deletion of the organization within Identity Management module
 - Every information relative to the users, groups, roles and other type of configuration are deleted



- The deletion of the organization MinIO server within the Data Management module and the permanent deletion of any type of data previously stored by the organization

2.3.3 Examples of actions on organizations

The following examples detail the action available for the HEROES administrators over the organizations within the HEROES Initial Platform.

2.3.3.1 Example of organization creation

The following is an example of HEROES administrator requesting the creation of a new organization to the UI/API module.

```
curl -X POST
  https://fastapi.heroes.doit.priv/heroes/admin/organizations/?user_email=example@doitsystems.fr
  -H 'accept: application/json'
  -H 'token : {eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZXQxN8EXA}'
  -d '{"name": "doitsystems"}'
```

The UI/API modules responds to this request by the basic information about the organization in creation with its current status, id, name and other information as shown below:

```
{
  "name": "doitsystems",
  "id": 2,
  "status": "CREATE_IN_PROGRESS"
}
```

As the creation of an organization is asynchronous, the status of the organizations can take those form as example:

- **CREATE_IN_PROGRESS** is the status used while the organization creation process, the organization is not available at this time.
- **CREATE_COMPLETE** is the final state of an organization, it means the organization is available and its internal services are fully functionals.

2.3.3.2 Example of organization deletion

The following is an example of HEROES administrator requesting the deletion of one organization within the HEROES Initial Platform to the UI/API module.

```
curl -X DELETE
  https://fastapi.heroes.doit.priv/heroes/admin/organizations/?organization_target=doitsystems
  -H 'accept: application/json'
```



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

```
-H 'token : {eyJhbGciOiI..._EzN8EXA}'
```

The UI/API modules responds to this request by the basic information about the organization targetted with its current status as shown below:

```
{  
  "name": "doitsystems",  
  "id": 2,  
  "status": "DELETE_IN_PROGRESS"  
}
```

As the deletion of an organization is asynchronous, the response of the organization in deletion is:

- **DELETE_IN_PROGRESS** is the status used while the organization deletion process, the organization is not available at this time.
- **NOT_FOUND** is the response returned when the organization is not or no longer present within the HEROES Platform

3 Updated Architecture

The developments carried out on the initial HEROES platform made it possible to verify a variety of assumptions, to remove a certain number of uncertainties and to evolve the overall architecture defined in the document D3.1 – Architecture Design. These updates involve technological evolutions as well as interconnection improvements between the platform modules and the components.

The uncertainties about the capacity of specific components such as the “Workflow manager framework” to communicate with the other modules and the Service Gateway led to the choice of RabbitMQ [6] as platform internal communication tool.

RabbitMQ is a messaging broker - an intermediary for messaging. It gives to the Service Gateway a method to send messages to the modules packaged with this broker and also allows the modules to be packaged with. It makes multiple modules able to receive messages from the Service Gateway and execute specific module requests depending on the received message

The Service Gateway has been explicitly designed all along the Initial Platform development because of its central role in the architecture design. The module has been designed to integrate FastAPI and a RabbitMQ client library into the platform to perform:

- The management of the ingress traffic coming from the users.
- The communication between the Service Gateway and the other modules within the HEROES Initial Platform.



The interaction with the modules can require specific information and secrets depending on the organization involved by the inbound requests. This need led to the deployment and dedication of a database to the storage of these sensitive informations alongside the FastAPI server within the UI/API module.

Subsequently to the integration of a RabbitMQ client within the FastAPI server, we proceeded to group the modules needing a communication tool into the “RabbitMQ workers”, resolving related uncertainties about internal communication between modules in the WP3. Which led to an updated architecture as seen Figure 8.

Concerning the Workflow and Job Management module, the “Workflow framework manager” considered previously in the document D3.1 – Architecture Design, named “Ryax” has been replaced by the open source alternative named “Nextflow”. The reasons of the replacement include but are not limited to:

- Singularity container’s usage to meet the container expectations
- The possibility to integrate custom features such as workflow running over multiple clusters

For more detailed explanations, refer to the D2.2 Workflow Containers – 6.1 Nextflow introduction.



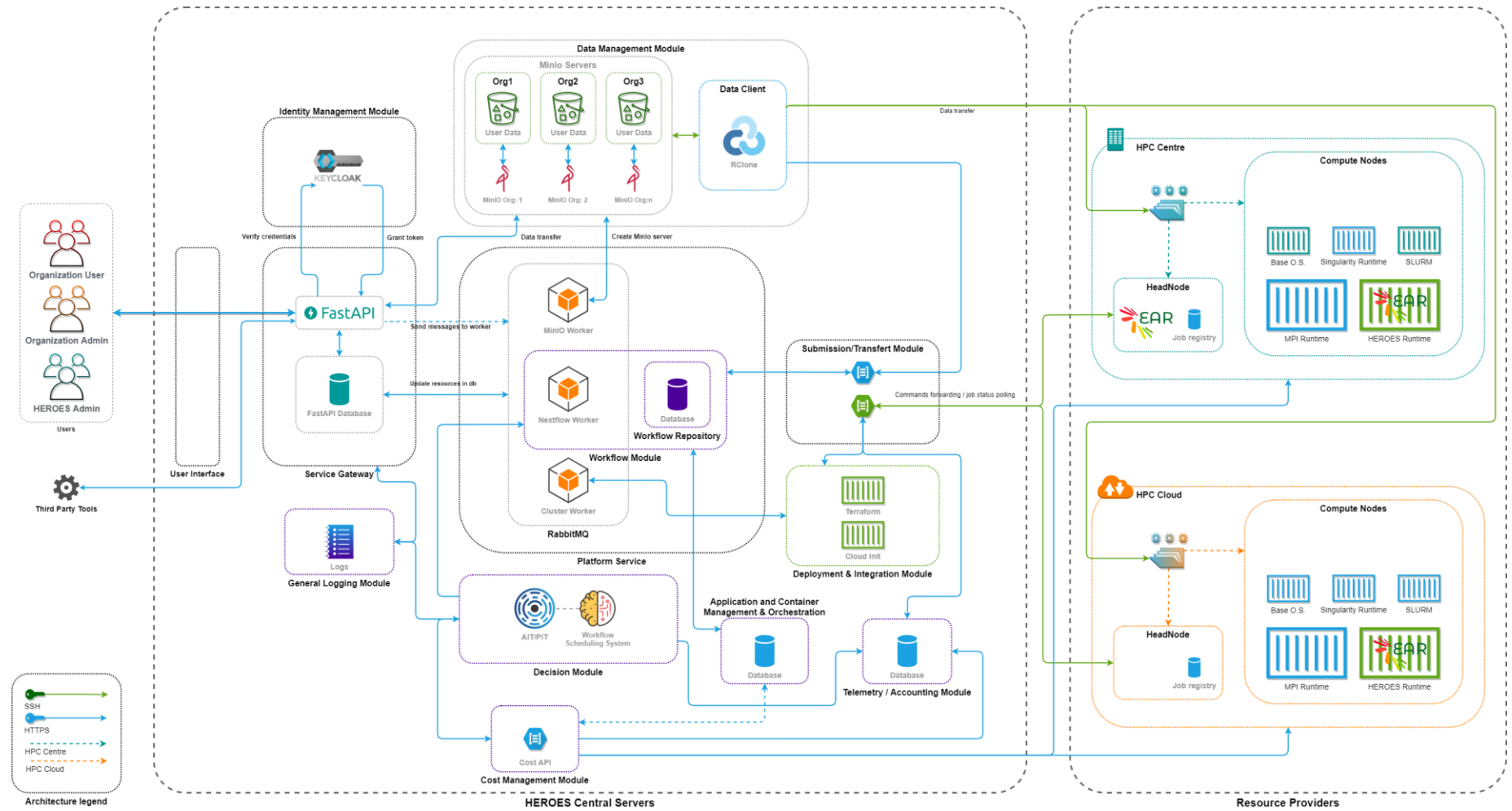


Figure 8. HEROES updated architecture



The HEROES project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956874. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Spain, Italy.

4 Containerized Platform and Automated CI/CD

In order to continue the modularly and reproducible architecture efforts, each service has been built within a container from versioned sources.

Kubernetes [7] and its control plane have been deployed on top of three nodes in the Do IT Systems development laboratory to run each container of the platform as “kubernetes application” (see Figure 9).

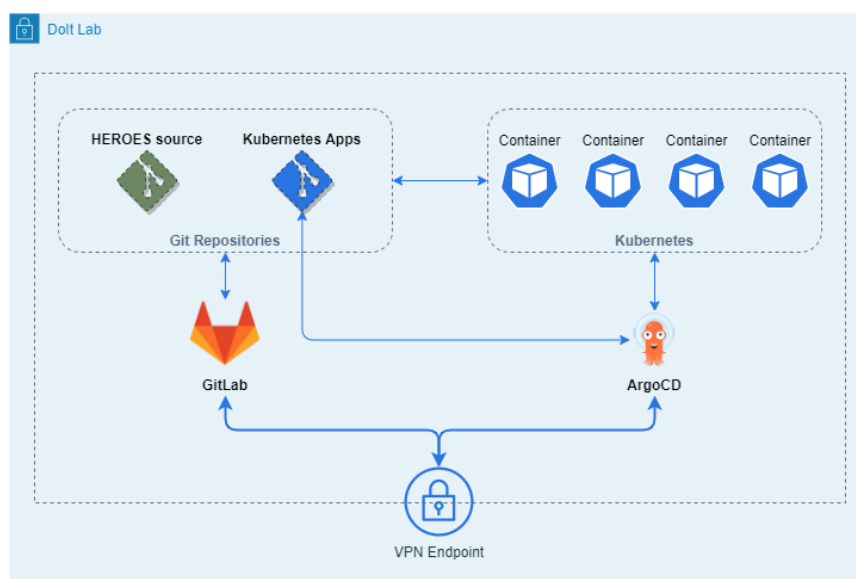


Figure 9. Do IT Systems Lab with its GitLab, Kubernetes and ArgoCD platform to support the HEROES platform development and deployment

This internal laboratory is reproducible as the entire platform can be deployed easily on any bare-metal or hardware infrastructure.

In addition, two other components, ArgoCD and GitLab have been deployed as detailed in the Section 4.1.

4.1 Platform Development Environment

The environment of development has been built around two key software: ArgoCD [8] and GitLab [9].

ArgoCD is a declarative, GitOps continuous delivery tool for Kubernetes. The core component is the “Application Controller”, which continuously monitors running applications and compares the live application state against the desired target state defined in the correct GitLab repository.



GitLab is an open-source code repository and collaborative software development platform. It also provides capabilities for issue tracking and CI/CD. The HEROES project is based in two different repositories:

- **HEROES Mock-Up** repository is used to store the development code relatives to the Initial Platform Mockup, including the container Docker files.
- **Kubernetes Apps** repository is used to store the Kubernetes deployment files, allowing ArgoCD to read any application configuration required by the Initial Platform Mockup.

4.2 Containerized Platform Services

4.2.1 Available services

Based on the modules developed, multiple services are available for the Initial Platform within the Dolt laboratory as it can be seen on Figure 10.

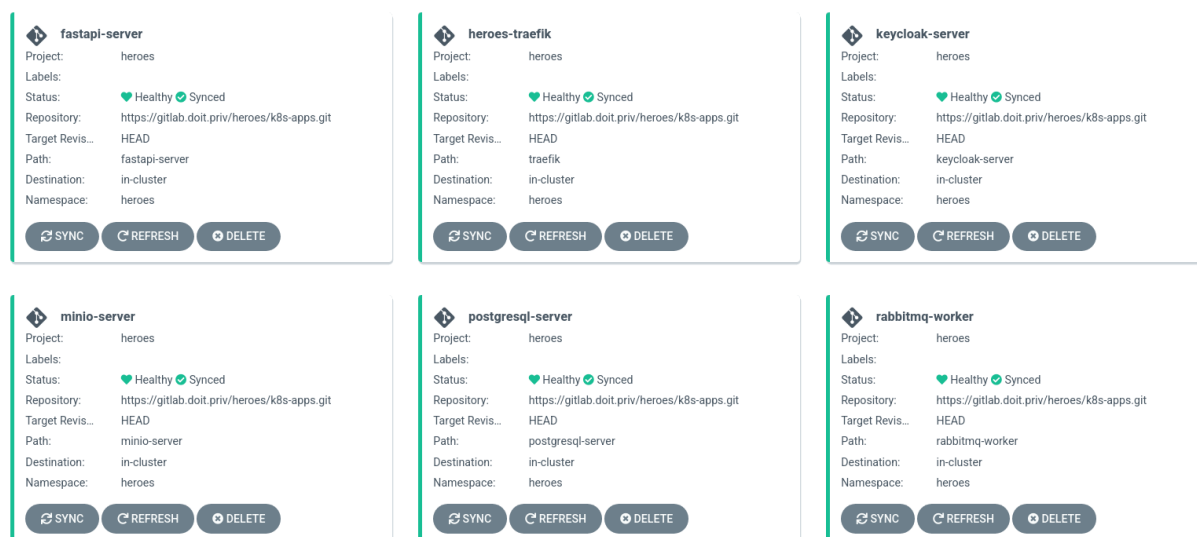


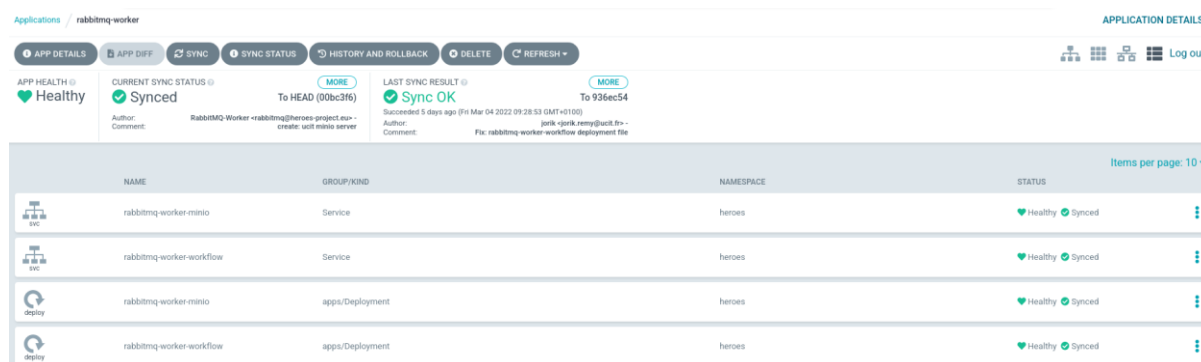
Figure 10. ArgoCD – Services view

The services deployed in the Dolt environment include but are not limited to:

- The FastAPI server as basic UI/API module
- The Keycloak server as Identity Management module
- A group of MinIO servers as Data Management module. After an organization creation or deletion request to the UI/API module, the MinIO worker detailed below pushes or removes a specific configuration of MinIO server in the GitLab “k8s-apps” repository corresponding to the organization associated. This has the effect of adding or removing a MinIO server from the group.



- A group of “RabbitMQ workers” as shown in Figure 11 including:
 - The MinIO worker pushes or removes the configuration of a MinIO server from the GitLab “k8s-apps” repository for any organization creation or deletion messaged by the UI/API module.
 - The Workflow worker submits, cancels and get permanent informations about the workflows.
 - The Cluster worker submits the creation and deletion of clusters in the Cloud for the organization.



NAME	GROUP/KIND	NAMESPACE	STATUS
rabbitmq-worker-minio	Service	heroes	Healthy Synced
rabbitmq-worker-workflow	Service	heroes	Healthy Synced
rabbitmq-worker-minio	apps/Deployment	heroes	Healthy Synced
rabbitmq-worker-workflow	apps/Deployment	heroes	Healthy Synced

Figure 11. RabbitMQ – Workers deployed within the HEROES Initial Platform

4.2.2 Service containerization and deployment

As each module has been mocked-up to correspond an ArgoCD service, the code development of applications integrated to the modules have been dissociated from the container packaging.

The containerization and deployment process of a service for the HEROES Initial Platform includes the next steps:

- The build of the image from its sources
 - Integrate the prerequisites, tools and software’s required by the service.
 - Integrate the code developed in the “HEROES Mock-Up” branch of the GitLab required by the service. As example, the “RabbitMQ workers” integrates the same installation of RabbitMQ with the correct worker associated.
 - Integrate the start process of the container including the launch of any application, software or service required within the container.
- The push of the image to the GitLab HEROES container registry which can store multiple repositories (tagged Docker images).



- The definition of the future ArgoCD service, including the image names defined in Kubernetes configuration files. These files are grouped in a directory named as the service and pushed to the GitLab “k8s-apps” branch.
- The creation of an ArgoCD service based on the Kubernetes configuration files of the service previously pushed in the “k8s-apps” branch as shown in Figure 12.

SOURCE

Repository URL

<https://gitlab.doit.priv/heroes/k8s-apps.git> GIT ✓

Revision

HEAD Branches ▼

Path

/new_service

Figure 12. ArgoCD – Creation Service: source path and repository selection of the service

5 Conclusion

In this document we presented the current HEROES Initial Platform that have been developed to validate the modules behaviour and the development method associated to the initial architecture design. With the objective of creating an Initial Platform mocking-up the modules defined as core functionality of the HEROES platform, it has been designed to evolve in association with the next steps of the project development.

The HEROES Initial Platform already integrates most of the required internal services, allowing to achieve some key use-cases and provide a development and test environment for the next steps of the HEROES Platform prototype.

The future of the HEROES Platform is the integration of a second module wave of the HEROES Architecture such as the Cost Management module, the Decision Module and the Logging & Accounting module.

The next steps will also include a continuous improvement of the modules presented in this document. As the development is permanently progressing, this document will be updated after each new module integration removing more uncertainties.

