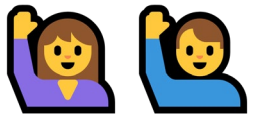
VANTAGE WORKSHOP

Python client

Running a PET analysis using the Python client



Please interrupt me if you have a question!

Jointly organized by:
netherlands eScience center
KNL

Overview

VANTAGE WORKSHOP

Questions

- How to **connect** to the vantage6 server?
- How to explore the **Client object**?
- How to check details of a **collaborations**?
- How to start a **compute task**?
- How to collect the **results** of a finished computation?

Objectives

- Understand the basic concepts of the vantage6 Python client.

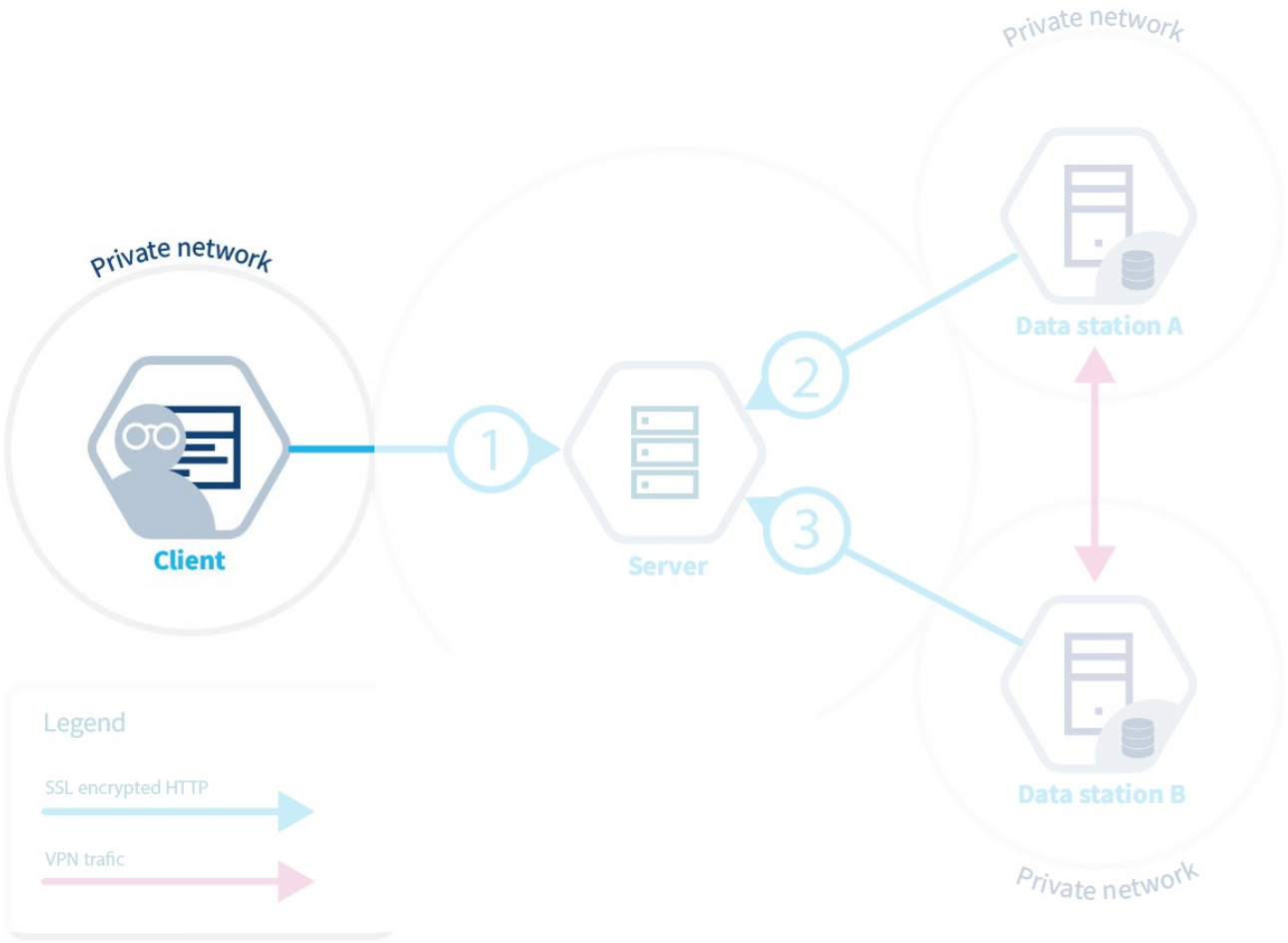
use the Python client to …

- Connect to the vantage6 server.
- Use the Python client to get details of a collaboration.
- Create a task using the Python client.
- Collect the results of a finished computation using the Python client.

Client

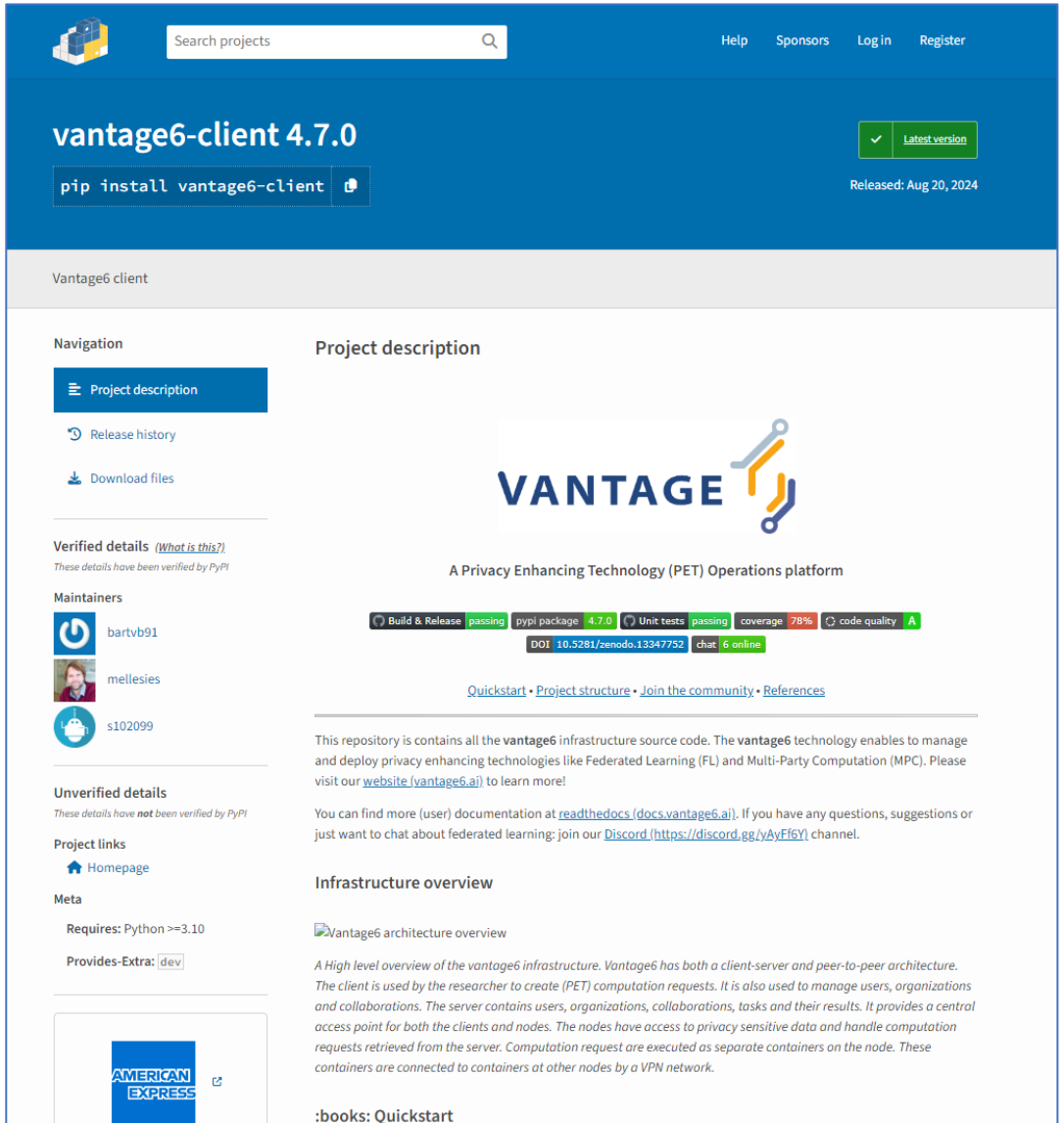
VANTAGE WORKSHOP

Interact with the server
Create computation tasks, manage organizations, collaborations, users and collecting results



vantage6-client package

VANTAGE WORKSHOP



Client functions

All client functions are collected in a single package. This package is published as [vantage6-client (pypi.org)](pypi.org).

In can simply be installed using:

install client

```
pip install vantage6-client
```

Jointly organized by:

netherlands eScience center KNL

Connect & authenticate

CHALLENGE

VANTAGE WORKSHOP

client.py

```python
# config.py

server_url = "https://<vantage6-server-address>"
server_port = 443
server_api = "/api"

username = "MY USERNAME"
password = "MY PASSWORD"

# Path to the private key, if encryption is enabled. Can be None if
# encryption is not used. Note that this key is the organization's
# private key. In case of this workshop we do not use encryption, so
# this can be None.
organization_key = None
```

client.py

```python
# client.py

from vantage6.client import Client

# It is assumed here that the `config.py` you just created is in the current
# directory. If it is not, then you need to make sure it can be found on
# your PYTHONPATH
import config

# Initialize the client object, and authenticate
client = Client(config.server_url, config.server_port, config.server_api,
                log_level='debug')
client.authenticate(config.username, config.password)
# In the case of 2FA, you should also include the 6-digit code:
# client.authenticate(config.username, config.password, '123456')

# Setup the encryption. In case no encryption is used, this line can be
# omitted or `config.organization_key` should be set to None
client.setup_encryption(config.organization_key)
```

Jointly organized by:
netherlands eScience center
KNL

Connect & authenticate

SOLUTION

VANTAGE WORKSHOP

1. Activate the conda environment with the vantage6 client installed.

```bash
conda activate v6-workshop
```

2. Create the **config.py** with your credentials and connection details.

3. Create the **client.py**.

4. Run the **client.py** script.

Jointly organized by:
netherlands eScience center
KNL

Using the client

VANTAGE WORKSHOP

Resource	Description
client.user	Manage users including your own user details
client.organization	Manage organizations or the organization that you are part of
client.rule	View all available permission rules
client.role	Manage roles (are collections of rules)
client.collaboration	Manage collaborations
client.task	Create new tasks and view their run data
client.result	Obtain results from the tasks
client.util	Provides utility functions for the vantage6 Python client. For example, to reset your password
client.node	Manage nodes
client.store	Manage an algorithm stores
client.algorithm	Manage algorithms that can be used for the computations

Using the client

VANTAGE WORKSHOP

Resource	Description
client.user	Manage users including your own user details
client.organization	Manage organizations or the organization that you are part of
client.node	Manage nodes
client.store	Manage an algorithm stores
client.algorithm	Manage algorithms that can be used for the computations

Permissions

The authenticated user is only able to perform allowed operations on allowed resources defined by its assigned permissions

Jointly organized by:

netherlands eScience center KNL

The 5 basic operations

VANTAGE WORKSHOP

GET

client.*resource*.get(ID)

Get a specific resource by its ID

get

```
client.organization.get(1)

# output:
{
    'nodes': '/api/node?organization_id=1',
    'public_key': '',
    'studies': '/api/study?organization_id=1',
    'name': 'Huckleberry Holdings',
    'tasks': '/api/task?init_org_id=1',
    'address2': '',
    'users': '/api/user?organization_id=1',
    'domain': 'huckleberryholdings.mc',
    'country': 'Monaco',
    'zipcode': '98000',
    'runs': '/api/run?organization_id=1',
    'address1': '4747 Huckleberry Ln',
    'id': 1,
    'collaborations': '/api/collaboration?organization_id=1'
}
```

The 5 basic operations

VANTAGE WORKSHOP

LIST

client.*resource*.list()

Get all resources from a certain type

Pagination

In order to keep the server performant, the list operation returns a paginated object.

You can control the page number and the number of items per page using the **page** and **per_page** parameter.

list

```
client.organization.list()

# output:
[
    {'id': 1, 'name': 'Huckleberry Holdings', ...}
    {'id': 2, 'name': 'Lychee Labs', ...},
    {'id': 3, 'name': 'Pineapple Paradigm', ...},
    {'id': 4, 'name': 'Huckleberry Hub', ...},
    {'id': 5, 'name': 'Mango Matrix', ...},
    {'id': 6, 'name': 'Apple Innovations', ...},
    {'id': 7, 'name': 'eScience center', ...},
    {'id': 8, 'name': 'Grapefruit Group', ...},
    {'id': 9, 'name': 'Raspberry Revolution', ...},
]
```

The 5 basic operations

VANTAGE WORKSHOP

CREATE

client.*resource*.create()

Update a specific resource

create

```
client.organization.create(
    name='new_organization',
    address1='street 1',
    address2='',
    zipcode='1234AB',
    country='NL',
    domain='example.com',
)

# output:
{
    'nodes': '/api/node?organization_id=172',
    'public_key': '',
    'studies': '/api/study?organization_id=172',
    'name': 'new_organization',
    'tasks': '/api/task?init_org_id=172',
    'address2': '',
    'users': '/api/user?organization_id=172',
    'domain': 'example.com',
    'country': 'NL',
    'zipcode': '1234AB',
    'runs': '/api/run?organization_id=172',
    'address1': 'street 1',
    'id': 172,
    'collaborations': '/api/collaboration?organization_id=172'
```

The 5 basic operations

VANTAGE WORKSHOP

UPDATE

client.*resource*.update()

Update a specific resource

update

```python
client.organization.update(id_, name='new_name')

# output:
{
    'nodes': '/api/node?organization_id=173',
    'public_key': '',
    'studies': '/api/study?organization_id=173',
    'name': 'new_name',
    'tasks': '/api/task?init_org_id=173',
    'address2': '',
    'users': '/api/user?organization_id=173',
    'domain': 'example.com',
    'country': 'NL',
    'zipcode': '1234AB',
    'runs': '/api/run?organization_id=173',
    'address1': 'street 1',
    'id': 173,
    'collaborations': '/api/collaboration?organization_id=173'
}
```

The 5 basic operations

VANTAGE WORKSHOP

DELETE

client.*resource*.delete()

Update a specific resource

delete

```
client.organization.delete(1)

# output:
--> Organization id=1 was removed from the database
```

Jointly organized by:
netherlands eScience center
KNL

Collect collaboration details

VANTAGE WORKSHOP

CHALLENGE

Before starting a task, you need to know the details of the collaboration you are working with.

Use the Python client to get the details of the collaborations you have access to. Write down the **name** and **ID** of each collaboration.

HINT

You can use the argument **fields** to return only the keys of interest. In this case that would be ‘id’ and ‘name’.

Collect collaboration details

SOLUTION

VANTAGE WORKSHOP

Different output

Your contents of your output will look different as your user has access to different collaborations!

solution

```
client.collaboration.list(fields=['id', 'name'])

# output:
[
    {'id': 168, 'name': 'Lychee Labs', ...},
    {'id': 158, 'name': 'Pineapple Paradigm', ...},
    {'id': 155, 'name': 'Huckleberry Hub', ...},
    {'id': 140, 'name': 'Mango Matrix', ...},
    {'id': 128, 'name': 'Apple Innovations', ...},
    {'id': 170, 'name': 'eScience center', ...},
    {'id': 165, 'name': 'Grapefruit Group', ...},
    {'id': 145, 'name': 'Raspberry Revolution', ...},
    {'id': 136, 'name': 'Ivy Berry Solutions', ...},
    {'id': 166, 'name': 'Huckleberry Holdings', ...}
]
```

… operations

VANTAGE WORKSHOP

Some resources do not provide all five operations, and some resources provide different operations. For example:

1. It is not possible to create, update or delete rules. Therefore, client.rule.create, client.rule.update and client.rule.delete do not exist.

2. The client.task has a client.task.kill method which can stop a task that currently is running.

Top Level and Util methods

Non resource related methods in the client.

VANTAGE WORKSHOP

Utils

top level methods

```python
# Change your password, you need to be
# authenticated before you can do this
client.util.change_my_password("old", "new")

# Generate a new organization private key
client.util.generate_private_key("/path/to/key.pem")

# Check the health of the server
client.util.get_server_health()

# Get the version of the vantage6 server
client.util.get_server_version()

# In case you forgot you password you can request
# a token to be send by email
client.util.reset_my_password(username_or_email)
# Then you can use this token to reset you passwowd
client.util.set_my_password(token_from_email)
```

Top Level

top level methods

```python
# Login to the vantage6 server
client.authenticate(username, password)

# Set collaboration ID for succesive calls.
# Can typically be overwritten by method
# parameters
client.setup_collaboration(id_)

# Configure the end-to-end encryption in
# case this is enabled in the collaboration
client.setup_encryption("/path/to/key.pem")

# Await the results from a task
client.wait_for_results(task_id)
```

VANTAGE
WORKSHOP

Method and parameter documentation

Each method has its own set of parameters, even the 5 basic operations can have different parameters.

Use the **help()** function in order to explore all available parameters and methods

ONLINE DOCS

All method/parameter documentation can also be found online at ***docs.vantage6.ai***

function documentation

```
help(client.organization.list)

# output:
# list(self, name: 'str' = None, country: 'int' = None,
#      collaboration: 'int' = None, study: 'int' = None,
#      page: 'int' = None, per_page: 'int' = None) -> 'list[dict]'
#
#    List organizations
#
#    Parameters
#    ----------
#    name: str, optional
#        Filter by name (with LIKE operator)
#    country: str, optional
#        Filter by country
#    collaboration: int, optional
#        Filter by collaboration id. If client.setup_collaboration() was called,
#        the previously setup collaboration is used. Default value is None
#
```

Find documentation

CHALLENGE

VANTAGE WORKSHOP

Find the documentation on how to reset your password 🔑 using the **help()** function.

HINT

Have a look at the overview of all client resources

Jointly organized by:
netherlands eScience center
KNL

Find documentation

SOLUTION

VANTAGE WORKSHOP

function documentation

```
help(client.util.change_my_password)

# output:
# change_my_password(current_password: 'str', new_password: 'str') -> 'dict'
#
#    Change your own password by providing your current password
#
#    Parameters
#    ----------
#    current_password : str
#        Your current password
#   new_password : str
#        Your new password
#
#    Returns
#    -------
#    dict
#        Message from the server
```

Jointly organized by:
netherlands eScience center
KNL

Identifiers are key

VANTAGE WORKSHOP

All resource items have a unique ID

This ID is used in the Python client to reference to a specific instance. For example, to view or update it.

User Interface

Identifiers can also be obtained from the UI. However, when working in the UI you will typically not worry about resource IDs.

Jointly organized by:
netherlands eScience center
KNL

Task preparation

VANTAGE WORKSHOP

1. Collect the IDs of all collaborations you have access to

get all collaboration IDs

```
client.collaboration.list(fields=['id', 'name'])
```

2. Collect the IDs of all organizations within this collaboration

get all organizations IDs of collaboration

```
client.organization.list(collaboration=<ID>, fields=['id', 'name'])
```

Task preparation

VANTAGE WORKSHOP

Checklist

Network

- ✓ ***Connection***
  Connect to the vantage6 server using the Python client.
- ✓ ***Collaboration Details***
  Use the Python client to get the details of the collaboration and its organizations you have access to.
- × *Node status*
  Check the status of the nodes

Average Algorithm

- ✓ ***Published***
  at harbor2.vantage6.ai/demo/average.
- ✓ ***Method***
  We are going to use the partial_average() method.
- ✓ ***Arguments***
  The function requires a column_name parameter, we are setting this to 'age'.

Node status
We have not checked yet if the nodes that we need are ready to receive computation requests

Jointly organized by:
netherlands eScience center
KNL

Node status

CHALLENGE

VANTAGE WORKSHOP

Use the Python client to check the status of the nodes that are part of the collaboration you are interested in.

HINT

Have a look at the client.node resource!

Node status

SOLUTION

VANTAGE WORKSHOP

Get collaboration node statuses

```
client.node.list(collaboration=<ID>, fields=['id', 'name', 'status'])

# output:
[
    {
        'id': 155,
        'name': 'IKNL demo node',
        'status': 'offline'
    },
    ...
]
```

Get organization ID from the node that is offline (add organization to fields)

Node Status

The node status can be Online, Offline or None. None means that the node never successfully connected to the server yet

Excluding nodes

VANTAGE WORKSHOP

1. **One of the nodes is not online**

   We should exclude this node from the computation.

2. **Limited flexibility**

   The average algorithm does not allow for making sub selections of organizations within the collaboration.

2. **Study to the rescue**

   Studies allow to make sub selections of organizations within a collaboration. The study concept is automatically followed by the algorithm.

```
client.study.list(fields=('id','name', 'organizations'))
```

Jointly organized by:
netherlands eScience center
KNL

Get Study with online nodes

VANTAGE WORKSHOP

CHALLENGE

Identify which nodes are online in the following studies:

Study	Collaboration
[*] AGOT2024	[*]
[*] GGA2024	[*]

[*] Should be replaced with the collaboration name you have access to. This can be one of Oak Alliance, Pine Partners, Maple Consortium, Cedar Coalition, Birch Brotherhood, Redwood Union or Willow Network.

HINT

```
client.study.list()
client.node.list(study=<study_id>, collaboration=<collaboration_id>)
```

Jointly organized by:
netherlands eScience center
KNL

Task definition

VANTAGE WORKSHOP

1. **The collaboration, study and organization identifiers**

   The IDs we have collected earlier to reference to the correct collaboration and organizations.

2. **The algorithm to be executed**

   We are going to use the federated average algorithm package which contains two functions: partial_average() and central_average().

3. **The input parameters for the algorithm**

   We need to tell the algorithm from which variable (=column) we want to compute the average from.

Methods, args and Kwargs

We define a dictionary containing the method and its arg and kwargs.

input parameters

```
input_ = {
    'method': 'partial_average',
    'args': [],
    'kwargs': {'column_name': 'age'}
}

# This will trigger:
# partial_average(column_name="age")
```

Jointly organized by:
netherlands eScience center
KNL

Start the task

VANTAGE WORKSHOP

input parameters

```
input_ = {
    'method': 'partial_average',
    'args': [],
    'kwargs': {'column_name': 'age'}
}

# This will trigger:
# partial_average(column_name="age")
```

start the task

```
average_task = client.task.create(
   collaboration=<ID>,
   organizations=[<org_1_ID>,<org_2_ID>],
   study=<study_id>,
   name="name_for_the_task",
   image="harbor2.vantage6.ai/demo/average",
   description='',
   input_=input_,
   databases=[
      {'label': 'default'}
   ]
)
```

Jointly organized by:

netherlands eScience center KNL

Collect the results and aggregate

VANTAGE WORKSHOP

solution

```
task_id = average_task['id']
result = client.wait_for_results(task_id)
```

Collect the results from the server

The **client.wait_for_results** is a blocking method. This means that the instructions will continue to either the algorithm finished or failed.

Aggregate partials

We still need to combine the result from each data station to a global average.

solution

```
import json
global_sum = 0
global_count = 0
for output in results["data"]:
    output = json.loads(output["result"])
    global_sum += output["sum"]
    global_count += output["count"]

print(global_sum / global_count)
```

Central Task

CHALLENGE

VANTAGE WORKSHOP

Run another task but this time the **central_average**.

This function should be sent to a single organization!

HINT

Central tasks also take care of the orchastration

Inspect log files

CHALLENGE

VANTAGE WORKSHOP

1. Retrieve the log files from the central method from previous challenge.

2. Rerun the central method, but this time use a column name that does not exist in the dataset (e.g. abc123). Retrieve the log files from this task as well.

RUNS

Each task consists of several runs. Each node included in the task execution will at least have one run. But in case of multi-step algorithms or iterative algorithms, a node can have multiple runs.

Each run has a log file that contains information about the execution of the algorithm on the node.

HINT

Have a look at **client.run.from_task!**

Jointly organized by:
netherlands eScience center
KNL