Administrator documentation

Release Version: 2.0
Designed by: Evgeniya Malikova (evgeniya.malikova@port.ac.uk)
22/07/2022

Contents

Introduction

Space missions and ground-based facilities collect increasingly huge amounts of data that demand novel approaches to data processing, storage, visualization and analysis.
The ViaLactea project is an ecosystem that offers the Astrophysics and Planetary communities highly interactive visual analytic interfaces enabling effective exploitation of

multi-wavelength observations of the Milky Way Galactic Plane, ranging from the near-infrared to the radio spectrum. ViaLactea strongly promotes FAIR data and Open Science practices and is integrated within the European Open Science Cloud (EOSC).

As a part of this research, the ViaLactea Web (VLW) solution is is developed as a collaborative web solutions for multi-user support underpinned by efficient remote server CPU and GPU rendering, and support of mobile and desktop devices. All underlying data is managed by a dedicated data service, namely the ViaLactea Knowledge Base, that provides object catalogs and Spectral Energy Distribution model outputs to carry out correlation analysis workflows for studying the star formation process in our Galaxy. The overall performance experiences is defined by remote GPU and CPU visualisation server performance.

The NEANIAS ViaLactea Web is available at
https://visivo-server.oact.inaf.it.
The main requirements are
Internet Access and Firefox Web browser
Microsoft or Google account for authentication
Access to the VLKB service through the NEANIAS Service Management System (SMS) (see section 1)

The service demo video is available at https://youtu.be/F6Q4xiMbHqg ; https://youtu.be/gAqBPd2d8uc

The User documentation is available online at https://vlw.readthedocs.io/en/latest/

Main components and system Architecture

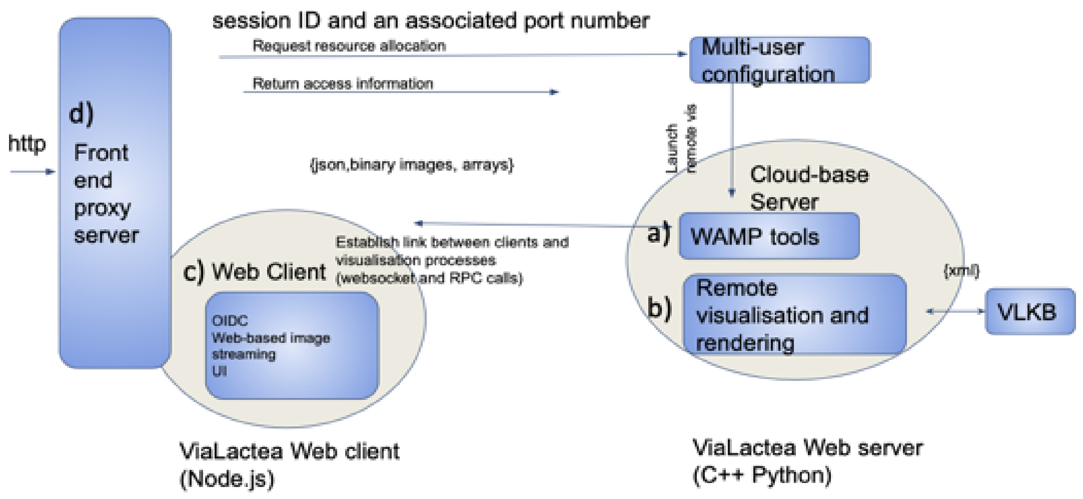
session ID and an associated port number
Request resource allocation
Return access information
Multi-user configuration
Launch remote vis
http
d) Front end proxy server
{json,binary images, arrays}
Cloud-base Server
a) WAMP tools
Establish link between clients and visualisation processes (websocket and RPC calls)
c) Web Client
OIDC
Web-based image streaming
UI
b) Remote visualisation and rendering
{xml}
VLKB
ViaLactea Web client (Node.js)
ViaLactea Web server (C++ Python)

Figure 1: VLW concepts and technologies involved.

The key technologies and concepts are discussed in "Real-time Web-based Visualisation: The ViaLactea Service in NEANIAS" paper, that is available online .

Configuring a VLW on Centos 8

3.1 Introduction

This document describes how to set up a ViaLacteaWeb server instance on Centos 8. The VLW core uses the VTK-based multi-user configuration. The system set-up is much similar ParaViewWeb configuration on Ubuntu described on the Kitware official documentation and follows similar principles.

The process can be split into the following steps that are considered in separate sections below:

1. Configuring the Centos 8 distribution so all the needed component are available
2. Building and installing VTK configured for remote server use
3. Configuring the launcher utility that is a part of wslink package (initially see ParaViewWeb launcher for details)
4. Configuring Apache as a proxy server for multi-user set up
5. Structuring the Web Site content for client part

3.2 Preparation

The following packages should be installed on Centos 8:

- Python 3 (below we consider python 3.8)
- Cmake>=3.12

- Apache

CMake installation

The Centos 8 installation includes CMake 3.11. In this document to compile PVW the CMake was updated to cmake 3.15 .

Currently this cmake version is compiled on INAF and can be accessed from a separate location /usr/local/bin

Apache installation

The Apache installation on Centos 8 is much different from official documentation on Ubuntu, so below we describe it more in details.
Before installing Apache, check the firewall settings.
The list of services that are already allowed should include http and http service

```
firewall-cmd --permanent --list-all
```

If not, the service should be added:

```
firewall-cmd --permanent --add-service=http
firewall-cmd --reload
```

Update the system before Apache set up

```
sudo dnf update
```

And install Apache on Centos 8

```
sudo dnf install httpd
```

The launching and checking Apache on Centos 8 is done with following commands:

```
sudo systemctl start httpd - launch
sudo systemctl status httpd - check that it is running (should be active)
```

By default Apache is running on 80 port. The test web page can be previewed in web browser from http://visivo-server.oact.inaf.it

To enable apache starting automatically on system rebooting run

```
sudo systemctl enable httpd
```

Setting up the directory structure and python configuration and packages required

Directories structure for VTK web server follows the logic of ParaviewWeb server described in official documentation with following differences:

1) */data/pv/pv-5.9/share/web* is a directory for default VLW visualisation application

2) The resulting directory for all python packages and VTK installation is:

/data/pv/pv-5.9/lib64/python3.8/site-packages
where */data/pv/pv-5.9/* - is VTK Web server and VLW C++ and Python libraries. This installation directory should be accessible for pvw-user.

We set up python virtual environment for further use as follows:

```
python3.8 -m venv /data/pv/pv-5.9
source /data/pv/pv-5.9/bin/activate
```

And install the twisted and wslink packages required for remote visualisation:

```
pip3.8 install twisted
pip3.8 install wslink
```

3.3 ViaLacteaWeb building and installation

VTK version 9.0.1 can be downloaded from the official Git repository.

build VTK and VLW according to appendix 1. We assume that the build can be an off-screen version with EGL support or OSMesa based CPU-rendering. The cmake prefix installation path for VTK and VLW should be /data/pv/pv-5.9.

After building, the sudo make install will install all necessary files to /data/pv/pv-5.9 , the VTK installation folder that is accessible to pvw-user (see below)

3.4 ViaLacteaWeb user creation and setting up rights

The process of user creation and setting up rights for folders /data/pv/pv-5.9/ and /data/pvw/ are similar to official guide .

The client files are stored in /data/www/vlw .The python server files are stored in /data/pv/pv-5.9/share/vtkjsserver .

Make sure the the rights for directories /data/www/vlw and /data/pv/pv-5.9/share/vtkjsserver for pvw-user are properly set

```
sudo chown -R pvw-user /data/pv
sudo chgrp -R pvw-user /data/pv
chcon -R --reference=/var/www  /data/www


sudo touch /data/proxy.txt

sudo chown pvw-user /data/proxy.txt

sudo chgrp www-data /data/proxy.txt
sudo chmod 660 /data/proxy.txt
```

Also, check that proper writing rights for temporary files writing are set in folder used, in our case it is `/home/pvw-user/"` (see section 3.5).

3.5 Configuration of launcher and start script

The process correlates with official guide .

We configure launcher as follows:

```
sudo vi /data/pvw/conf/launcher.json
```

```
{

"resources": [ {"port_range": [9001, 9019], "host": "localhost"} ],

"sessionData": {

"updir": "/Home"

},

"configuration": {

"log_dir": "/data/pvw/logs",

"host": "localhost",

"endpoint": "paraview",

"sessionURL":
"ws://visivo-server.oact.inal.it/proxy?sessionId=${id}&path=ws",

"timeout": 25,

"upload_dir": "/data/pvw/upload",

"fields": ["file", "host", "port", "updir"],

"port": 9020,

"proxy_file": "/data/proxy.txt"

},

"properties": {

"dataDir": "/home/pvw-user/",

"python_exec": "/data/pv/pv-current/bin/pvpython",

"vtkpython" : "/data/pv/pv-5.9/bin/vtkpython",

"vtk_python_path": "/data/pv/pv-5.9/share/vtkjsserver"
```

```
        },

        "apps": {

        "cone" : {

        "cmd" : [

        "${vtkpython}", "${vtk_python_path}/vtkw-server.py", "--port", "${port}",
        "--updir","${dataDir}","--session", "${id}" ],

        "ready_line" : "Starting factory"

        }

        }

        }
```

Note that `"dataDir": "/home/pvw-user/" is used to store temporary files - the results of VLKB queries. For each user the folder with session-id is created as a part of multi-user configuration`

The correct path to launcher, that is a part of wslink package distribution and python site-packages directory are to be explicitly set in start script, that we write as follows:

```
$ sudo vi /data/pvw/bin/start.sh



        #!/bin/bash



        export DISPLAY=:0.0

        export PYTHONPATH="/data/pv/pv-5.9/lib64/python3.8/site-packages/"

        export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/data/pv/pv-5.9/lib64

        export PATH=$PATH:/data/pv/pv-5.9/bin



        vtkpython /data/pv/pv-5.9/lib64/python3.8/site-packages/wslink/launcher.py
        /data/pvw/conf/launcher.json &
```

3.6 Configuring Apache

Compared to official PVW set up documentation for Ubuntu, a2enmod utility is not available. The virtual host on CentOS should be set up manually according to Apache documentation

The following Apache modules should be enabled on Centos 8 to enable reverse proxying and websocket tunelling:

mod_vhost_alias
mod_proxy
mod_proxy_http
mod_proxy_wstunnel
mod_rewrite

To check that this modules are enabled execute
hpptd -M

If they are not, edit the /etc/httpd/conf.modules.d/00-proxy.conf with the following:
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so

A separate virtual host is created in directory /data/www with a directory for log files:

```
sudo mkdir -p /data/www/
sudo mkdir -p /data/logs
```

Edit file permissions:

```
sudo chown -R $USER:$USER /data/www
sudo chmod -R 755 /data/www
```

Create 2 directories: the sites-available and sites-enabled:

```
sudo mkdir /etc/httpd/sites-available
sudo mkdir /etc/httpd/sites-enabled
```

Modify the Apache web server's main configuration file and instruct Apache where to locate the virtual host inside the sites-enabled directory. For this the "*IncludeOptional sites-enabled/*.conf*" should be added to */etc/httpd/conf/httpd.conf*
Alternatively, you can create a symbolic link to the newly created `inaf.conf`

```
sudo ln -s /etc/httpd/sites-available/inaf.conf /etc/httpd/conf.d/inaf.conf
```

The configuration file for virtual host inaf.conf looks like this:

```
vi  /etc/httpd/sites-available/inaf.conf

      <VirtualHost *:80>
           ServerName   visivo-server.oact.inaf.it
           ServerAdmin  YOUR_EMAIL@COMPANY.COM
           DocumentRoot /data/www


           <Directory /data/www>
               Options Indexes FollowSymLinks
               Order allow,deny
               Allow from all
               AllowOverride None
               Require all granted
           </Directory>

           # Handle launcher forwarding
           ProxyPass /paraview http://localhost:9020/paraview

           # Handle WebSocket forwarding
           RewriteEngine On
           RewriteMap session-to-port txt:/data/proxy.txt
           RewriteCond %{QUERY_STRING} ^sessionId=(.*)&path=(.*)$ [NC]
           RewriteRule ^/proxy.*$  ws://${session-to-port:%1}/%2  [P]
         </VirtualHost>
```

In case of selinux(Security Enhanced Linux), additional steps should be made. The selinux case can be checked with:

```
sestatus

SELinux status: enabled
....
```

There are two options

1) Disable selinux
2) Configure selinux with
    a) Allow selinux policy to Apache to can network connect via setsebool
       `setsebool -P httpd_can_network_connect on`

    b) Apply the proper security context to the directory to /data/www
       `chcon -R --reference=/var/www  /data/www`

Restart Apache to enable virtual host.

3.6 Setting up the ViaLacteaWeb Web Site

The process is similar to official documentation

3.7 Launch and configure for autostart on reboot

Similartlly to the official documentation the pvw-user launches the server part by /data/pvw/bin/start.sh script. To enable autorestart on reboot we configure the service as follows:

1) create a service

```
sudo nano /lib/systemd/system/pvstart.service
[Unit]
Description=VLW start script

[Service]
ExecStart=/data/pvw/bin/start.sh
Type=forking

[Install]
WantedBy=multi-user.target
```

Reload the systemctl daemon:

```
sudo systemctl daemon-reload
```

2) edit your /etc/sudoers file with visudo to allow the pvw-user to be able to use systemctl command to run service:

```
# vlw server commands

Cmnd_Alias VLW_CMDS = /usr/bin/systemctl start pvstart.service, /usr/bin/systemctl stop pvstart.service, /usr/bin/systemctl enable --now pvstart.service

pvw_user ALL=(ALL) NOPASSWD: VLW_CMDS
```

2) login as pvw-user and start the service:
Pavariew

```
sudo  systemctl enable --now pvstart.service
```

Also, the administrator can crontab a job for deleting temporary files in vlw defalt folder like:

```
0 0 * * * find /home/pvw-user -mindepth 1 ! -regex '^/home/pvw-user/files\(/.*\)?' -exec sudo rm -rf {} +
```

3.8 Setting up the certificate for secure connection

The steps below enable the web content to be loaded over https and secure WebSocket connection via wss. Please note that “mixed content” will be blocked by the browser.

3.8.1 Secure apache with let’s encrypt

The Certbot is used to set up a TLS/SSL certificate with the Apache web server. To install the tool:

```
        sudo dnf install epel-release
        sudo dnf install certbot python3-certbot-apache mod_ssl
```

The certificate is obtained interactively with:

```
        sudo certbot --apache
```

Please note, that apache configuration will be rewritten as follows (with adjustment for wss connection):

```
<IfModule mod_ssl.c>
<VirtualHost visivo-server.oact.inaf.it:443>
    ServerName   visivo-server.oact.inaf.it
    ServerAdmin  evgeniya.malikova@port.ac.uk
    DocumentRoot /data/www

     ErrorLog /data/www/logs/error.log
     CustomLog /data/www/logs/access.log combined

    <Directory /data/www>
       Options Indexes FollowSymLinks
       Order allow,deny
       Allow from all
       AllowOverride All
       Require all granted
RewriteEngine On
RewriteBase /


RewriteRule ^auth/signwin$ main.html [L]
RewriteRule ^/index\.html$ - [L]



RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.html [L]
    </Directory>
```

```
  # Handle launcher forwarding
   ProxyPreserveHost On
   ProxyPass /paraview http://localhost:9020/paraview

   # Handle WebSocket forwarding
   RewriteEngine On
   RewriteMap session-to-port txt:/data/proxy.txt
   RewriteCond %{QUERY_STRING} ^sessionId=(.*)&path=(.*)$ [NC]
   RewriteRule ^/proxy.*$  ws://${session-to-port:%1}/%2  [P]



 #RewriteCond %{SERVER_NAME} =visivo-server.oact.inaf.it
 #RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

The cron job for autorenewal make sure that the following services are enabled and check contents of cron.d folder:
/lib/systemd/system/certbot.service
/lib/systemd/system/certbot.timer
/etc/cron.d/certbot

To test renewal:

```
sudo certbot renew --dry-run
```

To check certificate info:

```
certbot certificates
```

3.8.2 Secure websocket

The launcher configuration should be changed for wss connection:
`sudo vi /data/pvw/conf/launcher.json`

```
{
"resources": [ {"port_range": [9001, 9019], "host": "localhost"} ],
"sessionData": {
"updir": "/Home"
},
"configuration": {
```

```
"log_dir": "/data/pvw/logs",

"host": "localhost",

"endpoint": "paraview",

"sessionURL":
"wss://visivo-server.oact.inaf.it/proxy?sessionId=${id}&path=ws",

"timeout": 25,

"upload_dir": "/data/pvw/upload",

"fields": ["file", "host", "port", "updir"],

"port": 9020,

"proxy_file": "/data/proxy.txt"

},

"properties": {

"dataDir": "/home/pvw-user/",

"python_exec": "/data/pv/pv-current/bin/pvpython",

"vtkpython" : "/data/pv/pv-5.9/bin/vtkpython",

"vtk_python_path": "/data/pv/pv-5.9/share/vtkjsserver"

},

"apps": {

"cone" : {

"cmd" : [

"${vtkpython}", "${vtk_python_path}/vtkw-server.py", "--port", "${port}",
"--updir","${dataDir}","--session", "${id}" ],

"ready_line" : "Starting factory"

}

}

}
```

Docker and Kubernetis

4.1 Introduction

The scheme basic concepts are similar to running ParaviewWeb from the oficial kitware container as described here.
The VLW container is build with similar concepts in mind to PVW images and are available at Github repository

Directories structure VLW web server follows the logic of ParaviewWeb server

1) /data/www will contains the html/js/css files that compose your web site that will leverage VLW client
2) /data/pv will contains the VLW installation binaries.
3) /data/pvw will contains the VLW process launcher that will dynamically trigger a new process for each visualization session.
4) /data/logs will contains the Apache logs that will serve your VLW virtual host.
5) */data/pv/pv-5.9/share/web* is a directory for default VLW visualisation application
6) The resulting directory for all python packages and VTK installation is:

/data/pv/pv-5.9/lib64/python3.8/site-packages
where */data/pv/pv-5.9/* - is VTK Web server and VLW C++ and Python libraries. This installation directory should be accessible for pvw-user.

The ready to use docker images are available at:
https://hub.docker.com/layers/222332814/emalikova/vialacteaweb/, Docker file is also available at Github repository

Similar to PV doc (https://kitware.github.io/paraviewweb/docs/docker.html),
1) The folder with local fist files should be mounted
2) GPU drivers should be accessible from docker (see *NVIDIA Container Toolkit*)
3) *SERVER_NAME environment variable should be defined*

Example:

```
export SERVER_NAME=localhost
export PROTOCOL=ws
docker run -v <fits_files>:/home/pvw-user/files -p 0.0.0.0:80:80 --name test3 --gpus all -e
"SERVER_NAME=${SERVER_NAME}" -it emalikova/vialacteaweb:latest
```

3.9 Kubernetis

The template of yaml file for Kubernetis, that sets SERVER_NAME environment variable and run all related scripts can be found at Github repository

Appendix 1: Building VTK 9.0.1 for remote visualisation

VTK should be compiled with python bindings and remote visualisation enabled (web module). In addition on the server, off-screen rendering (EGL-based or OSMesa-based) is usually enabled.

Below is an example of the cmake configuration with EGL enabled for off-screen rendering on the server (one should adjust accordingly the paths to python directories):

```
cmake -DCMAKE_BUILD_TYPE=Release -DVTK_PYTHON_VERSION=3
-DCMAKE_INSTALL_PREFIX=/data/pv/pv-5.9 -DVTK_GROUP_ENABLE_Web=YES
-DVTK_WRAP_PYTHON=ON -DPython3_EXECUTABLE=/usr/bin/python3.8
-DVTK_GROUP_ENABLE_Qt=NO -DPython3_INCLUDE_DIR=/usr/include/python3.8
-DVTK_MODULE_ENABLE_VTK_WebCore=YES
-DVTK_MODULE_ENABLE_VTK_WebGLExporter=YES
DVTK_MODULE_ENABLE_VTK_WebPython =YES
-DVTK_MODULE_ENABLE_VTK_CommonPython=YES
-DVTK_MODULE_ENABLE_VTK_FiltersPython=YES
-DVTK_USE_OFFSCREEN_EGL=YES -DVTK_USE_X=NO
-DVTK_MODULE_ENABLE_VTK_WebPython=YES ../
```

By default, VTK is compiled with GPU rendering enabled. To reconfigure it for entirely CPU rendering OSMesa should be installed and enabled in VTK with -DVTK_OPENGL_HAS_OSMESA=ON.

Below is an example of what cmake configuration looks like (one should adjust accordingly the paths to python directories):

```
cmake -DCMAKE_BUILD_TYPE=Release -DVTK_PYTHON_VERSION=3
-DCMAKE_INSTALL_PREFIX=/data/pv/pv-5.9 -DVTK_GROUP_ENABLE_Web=YES
-DVTK_WRAP_PYTHON=ON -DPython3_EXECUTABLE=/usr/bin/python3.8
-DVTK_GROUP_ENABLE_Qt=NO -DPython3_INCLUDE_DIR=/usr/include/python3.8
-DVTK_MODULE_ENABLE_VTK_WebCore=YES
-DVTK_MODULE_ENABLE_VTK_WebGLExporter=YES
DVTK_MODULE_ENABLE_VTK_WebPython =YES
-DVTK_MODULE_ENABLE_VTK_CommonPython=YES
-DVTK_MODULE_ENABLE_VTK_FiltersPython=YES -DVTK_USE_X=OFF
-DOPENGL_xmesa_INCLUDE_DIR=IGNORE -DOPENGL_gl_LIBRARY=IGNORE
-DOSMESA_INCLUDE_DIR=/usr/include -DVTK_OPENGL_HAS_OSMESA=ON
-DVTK_USE_OFFSCREEN=ON -DVTK_MODULE_ENABLE_VTK_WebPython=YES
-DOPENGL_INCLUDE_DIR=/usr/include ../
```

```
cmake -DCMAKE_BUILD_TYPE=Release -DVTK_PYTHON_VERSION=3
-DCMAKE_INSTALL_PREFIX=/opt/VTK-9.0.1 -DVTK_GROUP_ENABLE_Web=YES
-DVTK_WRAP_PYTHON=ON -DPython3_EXECUTABLE=/usr/bin/python3.8
-DVTK_GROUP_ENABLE_Qt=NO -DPython3_INCLUDE_DIR=/usr/include/python3.8
-DPython3_LIBRARY=/usr/lib64/libpython3.8.so
-DVTK_MODULE_ENABLE_VTK_WebCore=YES
-DVTK_MODULE_ENABLE_VTK_WebGLExporter=YES
DVTK_MODULE_ENABLE_VTK_WebPython =YES
-DVTK_MODULE_ENABLE_VTK_CommonPython=YES
-DVTK_MODULE_ENABLE_VTK_FiltersPython=YES -DVTK_USE_X=OFF
-DOPENGL_xmesa_INCLUDE_DIR=IGNORE -DOPENGL_gl_LIBRARY=IGNORE
-DOSMESA_INCLUDE_DIR=/opt/osmesa/include
-DOSMESA_LIBRARY=/opt/osmesa/lib/libOSMesa.so
-DVTK_OPENGL_HAS_OSMESA=ON -DVTK_USE_OFFSCREEN=ON
-DVTK_MODULE_ENABLE_VTK_WebPython=YES
-DOPENGL_INCLUDE_DIR=/opt/osmesa/include
-DOSMESA_INCLUDE_DIR=/opt/osmesa/include
-DOPENGL_glu_LIBRARY=/opt/osmesa/lib/libGLU.so ../
```