OMERO.web Installation doc Documentation

Release 0.1.0

The Open Microscopy Environment

Apr 27, 2021

CONTENTS

OMERO.web is a Python 3 client of the OMERO platform that provides a web-based UI and JSON API. This section provides links to detailed step-by-step walkthroughs describing how to install, customize, maintain and run OMERO.web for several systems. OMERO.web is installed **separately** from the OMERO.server.

OMERO.web can be deployed with:

- WSGI using a WSGI capable web server such as NGINX and Gunicorn
- the built-in Django lightweight development server. This type of deployment should only be used for **testing** purpose only; see the Developers Deployment page.

If you need help configuring your firewall rules, see Security for more details.

Depending upon which platform you are using, you may find a more specific walkthrough listed below. The guides use the example of deploying OMERO.web with NGINX and Gunicorn. OMERO can automatically generate a configuration file for your webserver. The location of the file will depend on your system, please refer to your webserver's manual. See in the section *Customizing your OMERO.web installation* in the various walkthroughs for more options.

CHAPTER

ONE

CONFIGURATION

You will find in the various guides how to create the NGINX OMERO configuration file and the configuration steps for the NGINX and Gunicorn. Advanced Gunicorn setups are also described to enable the download of binary data and to handle multiple clients on a single worker thread switching context as necessary while streaming binary data from OMERO.server. Depending on the traffic and scale of the repository you should configure connections and speed limits on your server to avoid blocking resources.

To set the various OMERO.web properties, check the OMERO.web configuration glossary.

CHAPTER

TWO

UPGRADING

Starting with OMERO 5.6, OMERO.server and OMERO.web installations are assumed to be separate throughout documentation, each with its own virtualenv and installation directory.

OMERO.web upgrade

CHAPTER

THREE

OPTIMIZING OMERO AS A DATA REPOSITORY

This section explains how to customize the appearance and functionality of OMERO.web to host images for groups or public viewing.

Publishing data using OMERO.web

OMERO.web UI customization

CHAPTER

FOUR

INSTALLATION WALKTHROUGHS

Recommended:

OMERO.web installation on CentOS 7 and IcePy 3.6 Instructions for installing OMERO.web from scratch on CentOS 7 with Ice 3.6.

OMERO.web installation on Ubuntu 20.04 and IcePy 3.6 Instructions for installing OMERO.web from scratch on Ubuntu 20.04 with Ice 3.6.

OMERO.web installation on Debian 10 and IcePy 3.6 Instructions for installing OMERO.web from scratch on Debian 10 with Ice 3.6.

Others:

OMERO.web installation on CentOS 8 and IcePy 3.6 Instructions for installing OMERO.web from scratch on CentOS 8 with Ice 3.6.

OMERO.web installation on Ubuntu 18.04 and IcePy 3.6 Instructions for installing OMERO.web from scratch on Ubuntu 18.04 with Ice 3.6.

OMERO.web installation on Debian 9 and IcePy 3.6 Instructions for installing OMERO.web from scratch on Debian 9 with Ice 3.6.

4.1 OMERO.web installation on CentOS 7 and IcePy 3.6

Please first read server installation on CentOS 7.

This is an example walkthrough for installing OMERO.web in a **virtual environment** using a dedicated system user. Installing OMERO.web in a virtual environment is the preferred way. For convenience in this walkthrough, we will use the **omero-web system user** and define the main OMERO.web configuration options as environment variables. Since 5.6, a new `OMERODIR` variable is used, you should first unset `OMERO_HOME` (if set) before beginning the installation process. By default, Python 3.6 is installed.

The following steps are run as root.

If required, first create a local system user omero-web and create directory:

```
useradd -m omero-web

mkdir -p /opt/omero/web/omero-web/etc/grid
chown -R omero-web /opt/omero/web/omero-web
```

4.1.1 Installing prerequisites

The following steps are run as root.

Install dependencies:

```
yum -y install epel-release

yum -y install unzip

yum -y install python3

yum -y install nginx
```

Optional: if you wish to use the Redis cache, install Redis:

```
yum -y install redis

systemctl enable redis.service

systemctl start redis.service
```

4.1.2 Creating a virtual environment

The following steps are run as root.

Create the virtual environment. This is the recommended way to install OMERO.web:

```
python3 -mvenv /opt/omero/web/venv3
```

Install ZeroC IcePy 3.6:

```
/opt/omero/web/venv3/bin/pip install --upgrade https://github.com/ome/zeroc-ice-py-
↪centos7/releases/download/0.2.1/zeroc_ice-3.6.5-cp36-cp36m-linux_x86_64.whl
```

Install OMERO.web:

```
/opt/omero/web/venv3/bin/pip install "omero-web>=5.6.1"
```

4.1.3 Installing OMERO.web apps

A number of apps are available to add functionality to OMERO.web, such as OMERO.figure and OMERO.iviewer. See the main website for a list of released apps. These apps are optional and can be installed, as the **root user**, via **`pip`** to your OMERO.web virtual environment and configured as the **omero-web system user**, at any time.

4.1.4 Configuring OMERO.web

The following steps are run as the omero-web system user.

For convenience the main OMERO.web configuration options have been defined as environment variables. You can either use your own values, or alternatively use the following ones:

```
export WEBSESSION=True
export OMERODIR=/opt/omero/web/omero-web
```

Configure OMERO.web and create the NGINX OMERO configuration file to be included in a system-wide NGINX configuration by redirecting the output of the command `omero web config nginx` below into a file. If an attempt is made to access OMERO.web whilst it is not running, the generated NGINX configuration file will automatically display a maintenance page:

```
export PATH=/opt/omero/web/venv3/bin:$PATH


omero web config nginx --http "${WEBPORT}" --servername "${WEBSERVER_NAME}" > /opt/
↪omero/web/omero-web/nginx.conf.tmp
```

OMERO.web offers a number of configuration options. The configuration changes **will not be applied** until Gunicorn is restarted using `omero web restart`. The Gunicorn workers are managed **separately** from other OMERO processes. You can check their status or stop them using `omero web status` or `omero web stop`.

- Session engine:
- OMERO.web offers alternative session backends to automatically delete stale data using the cache session store backend, see Django cached session documentation for more details.
- Redis requires django-redis in order to be used with OMERO.web. We assume that Redis has already been installed. To configure the cache, run:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.
↪cache.
RedisCache", "LOCATION": "redis://127.0.0.1:6379/0"}}'
```

- After installing all the cache prerequisites set the following:

```
omero config set omero.web.session_engine django.contrib.sessions.
↪backends.cache
```

- Use a prefix:

  By default OMERO.web expects to be run from the root URL of the webserver. This can be changed by setting `omero.web.prefix` and `omero.web.static_url`. For example, to make OMERO.web appear at *http://example.org/omero/*:

```
omero config set omero.web.prefix '/omero'
omero config set omero.web.static_url '/omero/static/'
```

  and regenerate your webserver configuration.

All configuration options can be found on various sections of developers/index.html#web_index developers documentation. For the full list, refer to *Configuration* properties.

The most popular configuration options include:

- Debug mode, see `omero.web.debug`.

- Customizing OMERO clients e.g. to add your own logo to the login page (`omero.web.login_logo`) or use an index page as an alternative landing page for users (`omero.web.index_template`). See *OMERO.web UI customization* for further information.
- Enabling a public user see *Publishing data using OMERO.web*.

4.1.5 Configuring Gunicorn

The following steps are run as the omero-web system user.

Additional settings can be configured by changing the properties below. Before changing the properties, run `export PATH=/opt/omero/web/venv3/bin:$PATH`:

- `omero.web.wsgi_workers` to (2 x NUM_CORES) + 1

Note: **Do not** scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

- `omero.web.wsgi_args` Additional arguments. For more details check Gunicorn Documentation. For example to enable **debugging**, run the following command:

```
omero config set omero.web.wsgi_args -- "--log-level=DEBUG --error-logfile=/opt/
↪omero/web/omero-web/var/log/error.log"
```

4.1.6 Setting up CORS

The following steps are run as root.

Cross Origin Resource Sharing allows web applications hosted at other origins to access resources from your OMERO.web installation. This can be achieved using the django-cors-headers app with additional configuration of OMERO.web. See the django-cors-headers page for more details on the settings:

```
/opt/omero/web/venv3/bin/pip install 'django-cors-headers<3.3'
```

The following steps are run as the omero-web system user.

Configure CORS. An `index` is used to specify the ordering of middleware classes. It is important to add the `CorsMiddleware` as the first class and `CorsPostCsrfMiddleware` as the last. You can specify allowed origins in a whitelist, or allow all, for example:

```
omero config append omero.web.middleware '{"index": 0.5, "class": "corsheaders.
↪middleware.CorsMiddleware"}'
omero config append omero.web.middleware '{"index": 10, "class": "corsheaders.
↪middleware.CorsPostCsrfMiddleware"}'
omero config set omero.web.cors_origin_whitelist '["https://hostname.example.com"]'
# or to allow all
omero config set omero.web.cors_origin_allow_all True
```

4.1.7 Configuring NGINX

The following steps are run as root.

Copy the generated configuration file into the NGINX configuration directory, disable the default configuration and start NGINX:

```
sed -i.bak -re 's/( default_server.*)/; #\1/' /etc/nginx/nginx.conf
if [ -f /etc/nginx/conf.d/default.conf ]; then
    mv /etc/nginx/conf.d/default.conf /etc/nginx/conf.d/default.disabled
fi
cp /opt/omero/web/omero-web/nginx.conf.tmp /etc/nginx/conf.d/omeroweb.conf

systemctl enable nginx

systemctl start nginx
```

For production servers you may need to add additional directives to the configuration file, for example to enable HTTPS. As an alternative to manually modifying the generated file you can generate a minimal configuration and include this in your own manually created NGINX file, such as `/etc/nginx/conf.d/omero-web.conf`:

```
omero web config nginx-location > /opt/omero/web/omero-web/omero-web-
↪location.include
```

This requires more initial work but in the future you can automatically regenerate your OMERO.web configuration and your additional configuration settings will still apply.

> **Note:** If you need help configuring your firewall rules, see the sysadmins/server-security.html page.

4.1.8 Running OMERO.web

The following steps are run as root.

Install WhiteNoise:

```
/opt/omero/web/venv3/bin/pip install --upgrade whitenoise
```

Optional: Install Django Redis:

```
/opt/omero/web/venv3/bin/pip install 'django-redis<4.9'
```

The following steps are run as the omero-web system user.

Optional: Configure the cache:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.cache.
↪RedisCache","LOCATION": "redis://127.0.0.1:6379/0"}}'
omero config set omero.web.session_engine 'django.contrib.sessions.backends.cache'
```

Configure WhiteNoise and start OMERO.web manually to test the installation:

```
omero config append -- omero.web.middleware '{"index": 0, "class": "whitenoise.
↪middleware.WhiteNoiseMiddleware"}'

omero web start
```

(continues on next page)

(continued from previous page)

```
# Test installation e.g. curl -sL localhost:4080

omero web stop
```

4.1.9 Automatically running OMERO.web

The following steps are run as root.

Should you wish to run OMERO.web automatically, a *systemd.service* file could be created. See below an example file *omero-web-systemd.service*:

```
[Unit]
Description=OMERO.web
# Not mandatory, NGINX may be running on a different server
Requires=nginx.service
After=network.service

[Service]
User=omero-web
Type=forking
PIDFile=/opt/omero/web/omero-web/var/django.pid
Restart=no
RestartSec=10
Environment="PATH=/opt/omero/web/venv3/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/
↪sbin:/usr/sbin"
Environment="OMERODIR=/opt/omero/web/omero-web"
ExecStart=/opt/omero/web/venv3/bin/omero web start
ExecStop=/opt/omero/web/venv3/bin/omero web stop

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

Copy the *systemd.service* file, then enable and start the service:

```
cp omero-web-systemd.service /etc/systemd/system/omero-web.service

systemctl daemon-reload

systemctl enable omero-web.service

systemctl stop omero-web.service

systemctl start omero-web.service
```

4.1.10 Maintaining OMERO.web

The following steps are run as the omero-web system user.

You can manage the sessions using the following configuration options and commands:

- Session cookies `omero.web.session_expire_at_browser_close`:
  - A boolean that determines whether to expire the session when the user closes their browser. See Django Browser-length sessions vs. persistent sessions documentation for more details. The default value is `True`:

```
omero config set omero.web.session_expire_at_browser_close "True"
```

  - The age of session cookies, in seconds. The default value is `86400`:

```
omero config set omero.web.session_cookie_age 86400
```

- Clear session:

  Each session for a logged-in user in OMERO.web is kept in the session store. Stale sessions can cause the store to grow with time. OMERO.web uses by default the OS file system as the session store backend and does not automatically purge stale sessions, see Django file-based session documentation for more details. It is therefore the responsibility of the OMERO administrator to purge the session cache using the provided management command:

```
omero web clearsessions
```

  It is recommended to call this command on a regular basis, for example as a `daily cron job`, see Django clearing the session store documentation for more information.

4.1.11 Troubleshooting

The following steps are run as the omero-web system user.

In order to identify why OMERO.web is not available run `omero web status`. Then consult NGINX `error.log` and `/opt/omero/web/omero-web/var/log/OMEROweb.log`.

4.1.12 Configuring Gunicorn advanced options

OMERO.web deployment can be configured with sync and async workers. **Sync workers** are faster and recommended for a data repository with *Download restrictions*. If you wish to offer users the ability to download data then you have to use **async workers**. OMERO.web is able to handle multiple clients on a single worker thread switching context as necessary while streaming binary data from OMERO.server. Depending on the traffic and scale of the repository you should configure connections and speed limits on your server to avoid blocking resources. We recommend you run benchmark and performance tests. It is also possible to apply *Download restrictions* and offer alternative access to binary data.

Note: Handling streaming request/responses requires proxy buffering to be turned off. For more details refer to Gunicorn deployment and NGINX configuration.

Note: `omero.web.application_server.max_requests` should be set to 0

See Gunicorn design for more details.

Experimental: Sync workers

The following steps are run as root.

Install futures:

```
/opt/omero/web/venv3/bin/pip install futures
```

The following steps are run as the omero-web system user.

To find out more about the number of worker threads for handling requests, see Gunicorn threads. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class
omero config set omero.web.wsgi_threads $(2-4 x NUM_CORES)
```

Experimental: Async workers

The following steps are run as root.

Install Gevent >= 0.13:

```
/opt/omero/web/venv3/bin/pip install 'gevent>=0.13'
```

The following steps are run as the omero-web system user.

To find out more about the maximum number of simultaneous clients, see Gunicorn worker-connections. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class gevent
omero config set omero.web.wsgi_worker_connections 1000
omero config set omero.web.application_server.max_requests 0
```

4.1.13 SELinux

The following steps are run as root.

If you are running a system with SELinux enabled and are unable to access OMERO.web you may need to adjust the security policy:

```
if [ $(getenforce) != Disabled ]; then

    yum -y install policycoreutils-python
    setsebool -P httpd_read_user_content 1
    setsebool -P httpd_enable_homedirs 1
    semanage port -a -t http_port_t -p tcp 4080

fi
```

4.2 OMERO.web installation on CentOS 8 and IcePy 3.6

Please first read server installation on CentOS 8.

This is an example walkthrough for installing OMERO.web in a **virtual environment** using a dedicated system user. Installing OMERO.web in a virtual environment is the preferred way. For convenience in this walkthrough, we will use the **omero-web system user** and define the main OMERO.web configuration options as environment variables. Since 5.6, a new `OMERODIR` variable is used, you should first unset `OMERO_HOME` (if set) before beginning the installation process. By default, Python 3.6 is installed.

The following steps are run as root.

If required, first create a local system user omero-web and create directory:

```
useradd -m omero-web

mkdir -p /opt/omero/web/omero-web/etc/grid
chown -R omero-web /opt/omero/web/omero-web
```

4.2.1 Installing prerequisites

The following steps are run as root.

Install dependencies:

```
yum -y install epel-release

yum -y install unzip

yum -y install python3

yum -y install nginx
```

Optional: if you wish to use the Redis cache, install Redis:

```
yum -y install redis python3-redis

systemctl enable redis.service

systemctl start redis.service
```

4.2.2 Creating a virtual environment

The following steps are run as root.

Create the virtual environment. This is the recommended way to install OMERO.web:

```
python3 -mvenv /opt/omero/web/venv3
```

Install ZeroC IcePy 3.6:

```
/opt/omero/web/venv3/bin/pip install --upgrade https://github.com/ome/zeroc-ice-
↪centos8/releases/download/0.0.1/zeroc_ice-3.6.5-cp36-cp36m-linux_x86_64.whl
```

Install OMERO.web:

```
/opt/omero/web/venv3/bin/pip install "omero-web>=5.6.1"
```

4.2.3 Installing OMERO.web apps

A number of apps are available to add functionality to OMERO.web, such as OMERO.figure and OMERO.iviewer. See the main website for a list of released apps. These apps are optional and can be installed, as the **root user**, via **`pip`** to your OMERO.web virtual environment and configured as the **omero-web system user**, at any time.

4.2.4 Configuring OMERO.web

The following steps are run as the omero-web system user.

For convenience the main OMERO.web configuration options have been defined as environment variables. You can either use your own values, or alternatively use the following ones:

```
export WEBSESSION=True
export OMERODIR=/opt/omero/web/omero-web
```

Configure OMERO.web and create the NGINX OMERO configuration file to be included in a system-wide NGINX configuration by redirecting the output of the command `omero web config nginx` below into a file. If an attempt is made to access OMERO.web whilst it is not running, the generated NGINX configuration file will automatically display a maintenance page:

```
export PATH=/opt/omero/web/venv3/bin:$PATH


omero web config nginx --http "${WEBPORT}" --servername "${WEBSERVER_NAME}" > /opt/
↪omero/web/omero-web/nginx.conf.tmp
```

OMERO.web offers a number of configuration options. The configuration changes **will not be applied** until Gunicorn is restarted using `omero web restart`. The Gunicorn workers are managed **separately** from other OMERO processes. You can check their status or stop them using `omero web status` or `omero web stop`.

- Session engine:
- OMERO.web offers alternative session backends to automatically delete stale data using the cache session store backend, see Django cached session documentation for more details.
- Redis requires django-redis in order to be used with OMERO.web. We assume that Redis has already been installed. To configure the cache, run:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.
↪cache.
RedisCache", "LOCATION": "redis://127.0.0.1:6379/0"}}'
```

- After installing all the cache prerequisites set the following:

```
omero config set omero.web.session_engine django.contrib.sessions.
↪backends.cache
```

- Use a prefix:

  By default OMERO.web expects to be run from the root URL of the webserver. This can be changed by setting `omero.web.prefix` and `omero.web.static_url`. For example, to make OMERO.web appear at *http://example.org/omero/*:

```
omero config set omero.web.prefix '/omero'
omero config set omero.web.static_url '/omero/static/'
```

and regenerate your webserver configuration.

All configuration options can be found on various sections of developers/index.html#web_index developers documentation. For the full list, refer to *Configuration* properties.

The most popular configuration options include:

- Debug mode, see `omero.web.debug`.
- Customizing OMERO clients e.g. to add your own logo to the login page (`omero.web.login_logo`) or use an index page as an alternative landing page for users (`omero.web.index_template`). See *OMERO.web UI customization* for further information.
- Enabling a public user see *Publishing data using OMERO.web*.

4.2.5 Configuring Gunicorn

The following steps are run as the omero-web system user.

Additional settings can be configured by changing the properties below. Before changing the properties, run `export PATH=/opt/omero/web/venv3/bin:$PATH`:

- `omero.web.wsgi_workers` to (2 x NUM_CORES) + 1

Note: **Do not** scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

- `omero.web.wsgi_args` Additional arguments. For more details check Gunicorn Documentation. For example to enable **debugging**, run the following command:

```
omero config set omero.web.wsgi_args -- "--log-level=DEBUG --error-logfile=/opt/
↪omero/web/omero-web/var/log/error.log"
```

4.2.6 Setting up CORS

The following steps are run as root.

Cross Origin Resource Sharing allows web applications hosted at other origins to access resources from your OMERO.web installation. This can be achieved using the django-cors-headers app with additional configuration of OMERO.web. See the django-cors-headers page for more details on the settings:

```
/opt/omero/web/venv3/bin/pip install 'django-cors-headers<3.3'
```

The following steps are run as the omero-web system user.

Configure CORS. An `index` is used to specify the ordering of middleware classes. It is important to add the `CorsMiddleware` as the first class and `CorsPostCsrfMiddleware` as the last. You can specify allowed origins in a whitelist, or allow all, for example:

```
omero config append omero.web.middleware '{"index": 0.5, "class": "corsheaders.
↪middleware.CorsMiddleware"}'
omero config append omero.web.middleware '{"index": 10, "class": "corsheaders.
↪middleware.CorsPostCsrfMiddleware"}'
```

(continues on next page)

(continued from previous page)

```
omero config set omero.web.cors_origin_whitelist '["https://hostname.example.com"]'
# or to allow all
omero config set omero.web.cors_origin_allow_all True
```

4.2.7 Configuring NGINX

The following steps are run as root.

Copy the generated configuration file into the NGINX configuration directory, disable the default configuration and start NGINX:

```
sed -i.bak -re 's/( default_server.*)/; #\1/' /etc/nginx/nginx.conf
if [ -f /etc/nginx/conf.d/default.conf ]; then
    mv /etc/nginx/conf.d/default.conf /etc/nginx/conf.d/default.disabled
fi
cp /opt/omero/web/omero-web/nginx.conf.tmp /etc/nginx/conf.d/omeroweb.conf

systemctl enable nginx

systemctl start nginx
```

For production servers you may need to add additional directives to the configuration file, for example to enable HTTPS. As an alternative to manually modifying the generated file you can generate a minimal configuration and include this in your own manually created NGINX file, such as `/etc/nginx/conf.d/omero-web.conf`:

```
omero web config nginx-location > /opt/omero/web/omero-web/omero-web-
↪location.include
```

This requires more initial work but in the future you can automatically regenerate your OMERO.web configuration and your additional configuration settings will still apply.

> **Note:** If you need help configuring your firewall rules, see the sysadmins/server-security.html page.

4.2.8 Running OMERO.web

The following steps are run as root.

Install WhiteNoise:

```
/opt/omero/web/venv3/bin/pip install --upgrade whitenoise
```

Optional: Install Django Redis:

```
/opt/omero/web/venv3/bin/pip install 'django-redis<4.9'
```

The following steps are run as the omero-web system user.

Optional: Configure the cache:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.cache.
↪RedisCache","LOCATION": "redis://127.0.0.1:6379/0"}}'
omero config set omero.web.session_engine 'django.contrib.sessions.backends.cache'
```

Configure WhiteNoise and start OMERO.web manually to test the installation:

```
omero config append -- omero.web.middleware '{"index": 0, "class": "whitenoise.
↪middleware.WhiteNoiseMiddleware"}'

omero web start

# Test installation e.g. curl -sL localhost:4080

omero web stop
```

4.2.9 Automatically running OMERO.web

The following steps are run as root.

Copy the *systemd.service* file, then enable and start the service:

```
cp omero-web-systemd.service /etc/systemd/system/omero-web.service

systemctl daemon-reload

systemctl enable omero-web.service

systemctl stop omero-web.service

systemctl start omero-web.service
```

4.2.10 Maintaining OMERO.web

The following steps are run as the omero-web system user.

You can manage the sessions using the following configuration options and commands:

- Session cookies `omero.web.session_expire_at_browser_close`:
  - A boolean that determines whether to expire the session when the user closes their browser. See Django Browser-length sessions vs. persistent sessions documentation for more details. The default value is `True`:

    ```
    omero config set omero.web.session_expire_at_browser_close "True"
    ```

  - The age of session cookies, in seconds. The default value is `86400`:

    ```
    omero config set omero.web.session_cookie_age 86400
    ```

- Clear session:

  Each session for a logged-in user in OMERO.web is kept in the session store. Stale sessions can cause the store to grow with time. OMERO.web uses by default the OS file system as the session store backend and does not automatically purge stale sessions, see Django file-based session documentation for more details. It is therefore the responsibility of the OMERO administrator to purge the session cache using the provided management command:

  ```
  omero web clearsessions
  ```

  It is recommended to call this command on a regular basis, for example as a `daily cron job`, see Django clearing the session store documentation for more information.

4.2.11 Troubleshooting

The following steps are run as the omero-web system user.

In order to identify why OMERO.web is not available run `omero web status`. Then consult NGINX `error.log` and `/opt/omero/web/omero-web/var/log/OMEROweb.log`.

4.2.12 Configuring Gunicorn advanced options

OMERO.web deployment can be configured with sync and async workers. **Sync workers** are faster and recommended for a data repository with *Download restrictions*. If you wish to offer users the ability to download data then you have to use **async workers**. OMERO.web is able to handle multiple clients on a single worker thread switching context as necessary while streaming binary data from OMERO.server. Depending on the traffic and scale of the repository you should configure connections and speed limits on your server to avoid blocking resources. We recommend you run benchmark and performance tests. It is also possible to apply *Download restrictions* and offer alternative access to binary data.

Note: Handling streaming request/responses requires proxy buffering to be turned off. For more details refer to Gunicorn deployment and NGINX configuration.

Note: `omero.web.application_server.max_requests` should be set to 0

See Gunicorn design for more details.

Experimental: Sync workers

The following steps are run as root.

Install futures:

```
/opt/omero/web/venv3/bin/pip install futures
```

The following steps are run as the omero-web system user.

To find out more about the number of worker threads for handling requests, see Gunicorn threads. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class
omero config set omero.web.wsgi_threads $(2-4 x NUM_CORES)
```

Experimental: Async workers

The following steps are run as root.

Install Gevent >= 0.13:

```
/opt/omero/web/venv3/bin/pip install 'gevent>=0.13'
```

The following steps are run as the omero-web system user.

To find out more about the maximum number of simultaneous clients, see Gunicorn worker-connections. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class gevent
omero config set omero.web.wsgi_worker_connections 1000
omero config set omero.web.application_server.max_requests 0
```

4.2.13 SELinux

The following steps are run as root.

If you are running a system with SELinux enabled and are unable to access OMERO.web you may need to adjust the security policy:

```
if [ $(getenforce) != Disabled ]; then

    yum -y install policycoreutils-python
    setsebool -P httpd_read_user_content 1
    setsebool -P httpd_enable_homedirs 1
    semanage port -a -t http_port_t -p tcp 4080

fi
```

4.3 OMERO.web installation on Ubuntu 18.04 and IcePy 3.6

Please first read server installation on Ubuntu 18.04.

This is an example walkthrough for installing OMERO.web in a **virtual environment** using a dedicated system user. Installing OMERO.web in a virtual environment is the preferred way. For convenience in this walkthrough, we will use the **omero-web system user** and define the main OMERO.web configuration options as environment variables. Since 5.6, a new `OMERODIR` variable is used, you should first unset `OMERO_HOME` (if set) before beginning the installation process. By default, Python 3.6 is installed.

The following steps are run as root.

If required, first create a local system user omero-web and create directory:

```
useradd -m omero-web

mkdir -p /opt/omero/web/omero-web/etc/grid
chown -R omero-web /opt/omero/web/omero-web
```

4.3.1 Installing prerequisites

The following steps are run as root.

Install dependencies:

```
apt-get update

apt-get -y install unzip
apt-get -y install python3
apt-get -y install python3-venv

apt-get -y install nginx
```

Optional: if you wish to use the Redis cache, install Redis:

```
apt-get -y install redis-server

service redis-server start
```

4.3.2 Creating a virtual environment

The following steps are run as root.

Create the virtual environment. This is the recommended way to install OMERO.web:

```
python3 -mvenv /opt/omero/web/venv3
```

Install ZeroC IcePy 3.6:

```
/opt/omero/web/venv3/bin/pip install --upgrade https://github.com/ome/zeroc-ice-
↪ubuntu1804/releases/download/0.2.0/zeroc_ice-3.6.5-cp36-cp36m-linux_x86_64.whl
```

Install OMERO.web:

```
/opt/omero/web/venv3/bin/pip install "omero-web>=5.6.1"
```

4.3.3 Installing OMERO.web apps

A number of apps are available to add functionality to OMERO.web, such as OMERO.figure and OMERO.iviewer. See the main website for a list of released apps. These apps are optional and can be installed, as the **root user**, via **`pip`** to your OMERO.web virtual environment and configured as the **omero-web system user**, at any time.

4.3.4 Configuring OMERO.web

The following steps are run as the omero-web system user.

For convenience the main OMERO.web configuration options have been defined as environment variables. You can either use your own values, or alternatively use the following ones:

```
export WEBSESSION=True
export OMERODIR=/opt/omero/web/omero-web
```

Configure OMERO.web and create the NGINX OMERO configuration file to be included in a system-wide NGINX configuration by redirecting the output of the command `omero web config nginx` below into a file. If an attempt is made to access OMERO.web whilst it is not running, the generated NGINX configuration file will automatically display a maintenance page:

```
export PATH=/opt/omero/web/venv3/bin:$PATH


omero web config nginx --http "${WEBPORT}" --servername "${WEBSERVER_NAME}" > /opt/
↪omero/web/omero-web/nginx.conf.tmp
```

OMERO.web offers a number of configuration options. The configuration changes **will not be applied** until Gunicorn is restarted using `omero web restart`. The Gunicorn workers are managed **separately** from other OMERO processes. You can check their status or stop them using `omero web status` or `omero web stop`.

- Session engine:
- OMERO.web offers alternative session backends to automatically delete stale data using the cache session store backend, see Django cached session documentation for more details.
- Redis requires django-redis in order to be used with OMERO.web. We assume that Redis has already been installed. To configure the cache, run:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.
↪cache.
RedisCache", "LOCATION": "redis://127.0.0.1:6379/0"}}'
```

- After installing all the cache prerequisites set the following:

```
omero config set omero.web.session_engine django.contrib.sessions.
↪backends.cache
```

- Use a prefix:

  By default OMERO.web expects to be run from the root URL of the webserver. This can be changed by setting `omero.web.prefix` and `omero.web.static_url`. For example, to make OMERO.web appear at *http://example.org/omero/*:

```
omero config set omero.web.prefix '/omero'
omero config set omero.web.static_url '/omero/static/'
```

  and regenerate your webserver configuration.

All configuration options can be found on various sections of developers/index.html#web_index developers documentation. For the full list, refer to *Configuration* properties.

The most popular configuration options include:

- Debug mode, see `omero.web.debug`.
- Customizing OMERO clients e.g. to add your own logo to the login page (`omero.web.login_logo`) or use an index page as an alternative landing page for users (`omero.web.index_template`). See *OMERO.web UI customization* for further information.
- Enabling a public user see *Publishing data using OMERO.web*.

4.3.5 Configuring Gunicorn

The following steps are run as the omero-web system user.

Additional settings can be configured by changing the properties below. Before changing the properties, run `export PATH=/opt/omero/web/venv3/bin:$PATH`:

- `omero.web.wsgi_workers` to (2 x NUM_CORES) + 1

Note: **Do not** scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

- `omero.web.wsgi_args` Additional arguments. For more details check Gunicorn Documentation. For example to enable **debugging**, run the following command:

```
omero config set omero.web.wsgi_args -- "--log-level=DEBUG --error-logfile=/opt/
↪omero/web/omero-web/var/log/error.log"
```

4.3.6 Setting up CORS

The following steps are run as root.

Cross Origin Resource Sharing allows web applications hosted at other origins to access resources from your OMERO.web installation. This can be achieved using the django-cors-headers app with additional configuration of OMERO.web. See the django-cors-headers page for more details on the settings:

```
/opt/omero/web/venv3/bin/pip install 'django-cors-headers<3.3'
```

The following steps are run as the omero-web system user.

Configure CORS. An `index` is used to specify the ordering of middleware classes. It is important to add the `CorsMiddleware` as the first class and `CorsPostCsrfMiddleware` as the last. You can specify allowed origins in a whitelist, or allow all, for example:

```
omero config append omero.web.middleware '{"index": 0.5, "class": "corsheaders.
↪middleware.CorsMiddleware"}'
omero config append omero.web.middleware '{"index": 10, "class": "corsheaders.
↪middleware.CorsPostCsrfMiddleware"}'
omero config set omero.web.cors_origin_whitelist '["https://hostname.example.com"]'
# or to allow all
omero config set omero.web.cors_origin_allow_all True
```

4.3.7 Configuring NGINX

The following steps are run as root.

Copy the generated configuration file into the NGINX configuration directory, disable the default configuration and start NGINX:

```
sed -i.bak -re 's/( default_server.*)/; #\1/' /etc/nginx/nginx.conf
rm /etc/nginx/sites-enabled/default
cp /opt/omero/web/omero-web/nginx.conf.tmp /etc/nginx/conf.d/omeroweb.conf

service nginx start
```

For production servers you may need to add additional directives to the configuration file, for example to enable HTTPS. As an alternative to manually modifying the generated file you can generate a minimal configuration and include this in your own manually created NGINX file, such as `/etc/nginx/conf.d/omero-web.conf`:

```
omero web config nginx-location > /opt/omero/web/omero-web/omero-web-
↪location.include
```

This requires more initial work but in the future you can automatically regenerate your OMERO.web configuration and your additional configuration settings will still apply.

> **Note:** If you need help configuring your firewall rules, see the sysadmins/server-security.html page.

4.3.8 Running OMERO.web

The following steps are run as root.

Install WhiteNoise:

```
/opt/omero/web/venv3/bin/pip install --upgrade whitenoise
```

Optional: Install Django Redis:

```
/opt/omero/web/venv3/bin/pip install 'django-redis<4.9'
```

The following steps are run as the omero-web system user.

Optional: Configure the cache:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.cache.
↪RedisCache","LOCATION": "redis://127.0.0.1:6379/0"}}'
omero config set omero.web.session_engine 'django.contrib.sessions.backends.cache'
```

Configure WhiteNoise and start OMERO.web manually to test the installation:

```
omero config append -- omero.web.middleware '{"index": 0, "class": "whitenoise.
↪middleware.WhiteNoiseMiddleware"}'

omero web start

# Test installation e.g. curl -sL localhost:4080

omero web stop
```

4.3.9 Automatically running OMERO.web

The following steps are run as root.

Should you wish to run OMERO.web automatically, a *init.d* file could be created. See below an example file *omero-web-init.d*:

```
#!/bin/bash
#
# /etc/init.d/omero-web
# Subsystem file for "omero" web
#
### BEGIN INIT INFO
# Provides:             omero-web
# Required-Start:       $local_fs $remote_fs $network $time omero postgresql
# Required-Stop:        $local_fs $remote_fs $network $time omero postgresql
# Default-Start:        2 3 4 5
# Default-Stop:         0 1 6
# Short-Description:    OMERO.web
### END INIT INFO
#
### Redhat
# chkconfig: - 98 02
# description: init file for OMERO.web
###
```

(continues on next page)

(continued from previous page)

```
RETVAL=0
prog=omero-web

# Read configuration variable file if it is present
[ -r /etc/default/$prog ] && . /etc/default/$prog

OMERO_USER=${OMERO_USER:-omero-web}
OMERO=/opt/omero/web/venv3/bin/omero
OMERODIR=/opt/omero/web/omero-web
VENVDIR=${VENVDIR:-/opt/omero/web/venv3}

start() {
    echo -n $"Starting $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web start" &> /dev/null && echo -n ' OMERO.web'
    sleep 5
    RETVAL=$?
    [ "$RETVAL" = 0 ]
        echo
}

stop() {
    echo -n $"Stopping $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web stop" &> /dev/null && echo -n ' OMERO.web'
    RETVAL=$?
    [ "$RETVAL" = 0 ]
        echo
}

status() {
    echo -n $"Status $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web status"
    RETVAL=$?
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    status)
        status
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart|status}"
        RETVAL=1
esac
exit $RETVAL
```

Copy the *init.d* file, then configure the service:

```
cp omero-web-init.d /etc/init.d/omero-web
chmod a+x /etc/init.d/omero-web

update-rc.d -f omero-web remove
update-rc.d -f omero-web defaults 98 02
```

Start up services:

```
service redis-server start

cron
service nginx start
service omero-web restart
```

4.3.10 Maintaining OMERO.web

The following steps are run as the omero-web system user.

You can manage the sessions using the following configuration options and commands:

- Session cookies `omero.web.session_expire_at_browser_close`:
  - A boolean that determines whether to expire the session when the user closes their browser. See Django Browser-length sessions vs. persistent sessions documentation for more details. The default value is `True`:

    ```
    omero config set omero.web.session_expire_at_browser_close "True"
    ```

  - The age of session cookies, in seconds. The default value is `86400`:

    ```
    omero config set omero.web.session_cookie_age 86400
    ```

- Clear session:

  Each session for a logged-in user in OMERO.web is kept in the session store. Stale sessions can cause the store to grow with time. OMERO.web uses by default the OS file system as the session store backend and does not automatically purge stale sessions, see Django file-based session documentation for more details. It is therefore the responsibility of the OMERO administrator to purge the session cache using the provided management command:

  ```
  omero web clearsessions
  ```

  It is recommended to call this command on a regular basis, for example as a `daily cron job`, see Django clearing the session store documentation for more information.

4.3.11 Troubleshooting

The following steps are run as the omero-web system user.

In order to identify why OMERO.web is not available run `omero web status`. Then consult NGINX `error.log` and `/opt/omero/web/omero-web/var/log/OMEROweb.log`.

4.3.12 Configuring Gunicorn advanced options

OMERO.web deployment can be configured with sync and async workers. **Sync workers** are faster and recommended for a data repository with *Download restrictions*. If you wish to offer users the ability to download data then you have to use **async workers**. OMERO.web is able to handle multiple clients on a single worker thread switching context as necessary while streaming binary data from OMERO.server. Depending on the traffic and scale of the repository you should configure connections and speed limits on your server to avoid blocking resources. We recommend you run benchmark and performance tests. It is also possible to apply *Download restrictions* and offer alternative access to binary data.

Note: Handling streaming request/responses requires proxy buffering to be turned off. For more details refer to Gunicorn deployment and NGINX configuration.

Note: `omero.web.application_server.max_requests` should be set to 0

See Gunicorn design for more details.

Experimental: Sync workers

The following steps are run as root.

Install futures:

```
/opt/omero/web/venv3/bin/pip install futures
```

The following steps are run as the omero-web system user.

To find out more about the number of worker threads for handling requests, see Gunicorn threads. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class
omero config set omero.web.wsgi_threads $(2-4 x NUM_CORES)
```

Experimental: Async workers

The following steps are run as root.

Install Gevent >= 0.13:

```
/opt/omero/web/venv3/bin/pip install 'gevent>=0.13'
```

The following steps are run as the omero-web system user.

To find out more about the maximum number of simultaneous clients, see Gunicorn worker-connections. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class gevent
omero config set omero.web.wsgi_worker_connections 1000
omero config set omero.web.application_server.max_requests 0
```

4.4 OMERO.web installation on Ubuntu 20.04 and IcePy 3.6

Please first read server installation on Ubuntu 20.04.

This is an example walkthrough for installing OMERO.web in a **virtual environment** using a dedicated system user. Installing OMERO.web in a virtual environment is the preferred way. For convenience in this walkthrough, we will use the **omero-web system user** and define the main OMERO.web configuration options as environment variables. Since 5.6, a new `OMERODIR` variable is used, you should first unset `OMERO_HOME` (if set) before beginning the installation process. By default, Python 3.8 is installed.

The following steps are run as root.

If required, first create a local system user omero-web and create directory:

```
useradd -m omero-web

mkdir -p /opt/omero/web/omero-web/etc/grid
chown -R omero-web /opt/omero/web/omero-web
```

4.4.1 Installing prerequisites

The following steps are run as root.

Install dependencies:

```
apt-get update

apt-get -y install unzip
apt-get -y install python3
apt-get -y install python3-venv

apt-get -y install nginx
```

Optional: if you wish to use the Redis cache, install Redis:

```
apt-get -y install redis-server

service redis-server start
```

4.4.2 Creating a virtual environment

The following steps are run as root.

Create the virtual environment. This is the recommended way to install OMERO.web:

```
python3 -mvenv /opt/omero/web/venv3
```

Install ZeroC IcePy 3.6:

```
/opt/omero/web/venv3/bin/pip install --upgrade https://github.com/ome/zeroc-ice-
↪ubuntu2004/releases/download/0.2.0/zeroc_ice-3.6.5-cp38-cp38-linux_x86_64.whl
```

Install OMERO.web:

```
/opt/omero/web/venv3/bin/pip install "omero-web>=5.6.1"
```

4.4.3 Installing OMERO.web apps

A number of apps are available to add functionality to OMERO.web, such as OMERO.figure and OMERO.iviewer. See the main website for a list of released apps. These apps are optional and can be installed, as the **root user**, via **pip** to your OMERO.web virtual environment and configured as the **omero-web system user**, at any time.

4.4.4 Configuring OMERO.web

The following steps are run as the omero-web system user.

For convenience the main OMERO.web configuration options have been defined as environment variables. You can either use your own values, or alternatively use the following ones:

```
export WEBSESSION=True
export OMERODIR=/opt/omero/web/omero-web
```

Configure OMERO.web and create the NGINX OMERO configuration file to be included in a system-wide NGINX configuration by redirecting the output of the command `omero web config nginx` below into a file. If an attempt is made to access OMERO.web whilst it is not running, the generated NGINX configuration file will automatically display a maintenance page:

```
export PATH=/opt/omero/web/venv3/bin:$PATH


omero web config nginx --http "${WEBPORT}" --servername "${WEBSERVER_NAME}" > /opt/
↪omero/web/omero-web/nginx.conf.tmp
```

OMERO.web offers a number of configuration options. The configuration changes **will not be applied** until Gunicorn is restarted using `omero web restart`. The Gunicorn workers are managed **separately** from other OMERO processes. You can check their status or stop them using `omero web status` or `omero web stop`.

- Session engine:
- OMERO.web offers alternative session backends to automatically delete stale data using the cache session store backend, see Django cached session documentation for more details.
- Redis requires django-redis in order to be used with OMERO.web. We assume that Redis has already been installed. To configure the cache, run:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.
↪cache.
RedisCache", "LOCATION": "redis://127.0.0.1:6379/0"}}'
```

- After installing all the cache prerequisites set the following:

```
omero config set omero.web.session_engine django.contrib.sessions.
↪backends.cache
```

- Use a prefix:

  By default OMERO.web expects to be run from the root URL of the webserver. This can be changed by setting `omero.web.prefix` and `omero.web.static_url`. For example, to make OMERO.web appear at *http://example.org/omero/*:

```
omero config set omero.web.prefix '/omero'
omero config set omero.web.static_url '/omero/static/'
```

  and regenerate your webserver configuration.

All configuration options can be found on various sections of developers/index.html#web_index developers documentation. For the full list, refer to *Configuration* properties.

The most popular configuration options include:

- Debug mode, see `omero.web.debug`.
- Customizing OMERO clients e.g. to add your own logo to the login page (`omero.web.login_logo`) or use an index page as an alternative landing page for users (`omero.web.index_template`). See *OMERO.web UI customization* for further information.
- Enabling a public user see *Publishing data using OMERO.web*.

4.4.5 Configuring Gunicorn

The following steps are run as the omero-web system user.

Additional settings can be configured by changing the properties below. Before changing the properties, run `export PATH=/opt/omero/web/venv3/bin:$PATH`:

- `omero.web.wsgi_workers` to (2 x NUM_CORES) + 1

Note: **Do not** scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

- `omero.web.wsgi_args` Additional arguments. For more details check Gunicorn Documentation. For example to enable **debugging**, run the following command:

```
omero config set omero.web.wsgi_args -- "--log-level=DEBUG --error-logfile=/opt/
↪omero/web/omero-web/var/log/error.log"
```

4.4.6 Setting up CORS

The following steps are run as root.

Cross Origin Resource Sharing allows web applications hosted at other origins to access resources from your OMERO.web installation. This can be achieved using the django-cors-headers app with additional configuration of OMERO.web. See the django-cors-headers page for more details on the settings:

```
/opt/omero/web/venv3/bin/pip install 'django-cors-headers<3.3'
```

The following steps are run as the omero-web system user.

Configure CORS. An `index` is used to specify the ordering of middleware classes. It is important to add the `CorsMiddleware` as the first class and `CorsPostCsrfMiddleware` as the last. You can specify allowed origins in a whitelist, or allow all, for example:

```
omero config append omero.web.middleware '{"index": 0.5, "class": "corsheaders.
↪middleware.CorsMiddleware"}'
omero config append omero.web.middleware '{"index": 10, "class": "corsheaders.
↪middleware.CorsPostCsrfMiddleware"}'
omero config set omero.web.cors_origin_whitelist '["https://hostname.example.com"]'
# or to allow all
omero config set omero.web.cors_origin_allow_all True
```

4.4.7 Configuring NGINX

The following steps are run as root.

Copy the generated configuration file into the NGINX configuration directory, disable the default configuration and start NGINX:

```
sed -i.bak -re 's/( default_server.*)/; #\1/' /etc/nginx/nginx.conf
rm /etc/nginx/sites-enabled/default
cp /opt/omero/web/omero-web/nginx.conf.tmp /etc/nginx/conf.d/omeroweb.conf

service nginx start
```

For production servers you may need to add additional directives to the configuration file, for example to enable HTTPS. As an alternative to manually modifying the generated file you can generate a minimal configuration and include this in your own manually created NGINX file, such as `/etc/nginx/conf.d/omero-web.conf`:

```
omero web config nginx-location > /opt/omero/web/omero-web/omero-web-
↪location.include
```

This requires more initial work but in the future you can automatically regenerate your OMERO.web configuration and your additional configuration settings will still apply.

> **Note:** If you need help configuring your firewall rules, see the sysadmins/server-security.html page.

4.4.8 Running OMERO.web

The following steps are run as root.

Install WhiteNoise:

```
/opt/omero/web/venv3/bin/pip install --upgrade whitenoise
```

Optional: Install Django Redis:

```
/opt/omero/web/venv3/bin/pip install 'django-redis<4.9'
```

The following steps are run as the omero-web system user.

Optional: Configure the cache:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.cache.
↪RedisCache","LOCATION": "redis://127.0.0.1:6379/0"}}'
omero config set omero.web.session_engine 'django.contrib.sessions.backends.cache'
```

Configure WhiteNoise and start OMERO.web manually to test the installation:

```
omero config append -- omero.web.middleware '{"index": 0, "class": "whitenoise.
↪middleware.WhiteNoiseMiddleware"}'

omero web start

# Test installation e.g. curl -sL localhost:4080

omero web stop
```

4.4.9 Automatically running OMERO.web

The following steps are run as root.

Should you wish to run OMERO.web automatically, a *init.d* file could be created. See below an example file *omero-web-init.d*:

```
#!/bin/bash
#
# /etc/init.d/omero-web
# Subsystem file for "omero" web
#
### BEGIN INIT INFO
# Provides:             omero-web
# Required-Start:       $local_fs $remote_fs $network $time omero postgresql
# Required-Stop:        $local_fs $remote_fs $network $time omero postgresql
# Default-Start:        2 3 4 5
# Default-Stop:         0 1 6
# Short-Description:    OMERO.web
### END INIT INFO
#
### Redhat
# chkconfig: - 98 02
# description: init file for OMERO.web
###

RETVAL=0
prog=omero-web

# Read configuration variable file if it is present
[ -r /etc/default/$prog ] && . /etc/default/$prog

OMERO_USER=${OMERO_USER:-omero-web}
OMERO=/opt/omero/web/venv3/bin/omero
OMERODIR=/opt/omero/web/omero-web
VENVDIR=${VENVDIR:-/opt/omero/web/venv3}

start() {
    echo -n $"Starting $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web start" &> /dev/null && echo -n ' OMERO.web'
    sleep 5
    RETVAL=$?
    [ "$RETVAL" = 0 ]
        echo
}

stop() {
    echo -n $"Stopping $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web stop" &> /dev/null && echo -n ' OMERO.web'
    RETVAL=$?
    [ "$RETVAL" = 0 ]
        echo
}

status() {
    echo -n $"Status $prog:"
```

(continues on next page)

(continued from previous page)

```
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web status"
    RETVAL=$?
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    status)
        status
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart|status}"
        RETVAL=1
esac
exit $RETVAL
```

Copy the *init.d* file, then configure the service:

```
cp omero-web-init.d /etc/init.d/omero-web
chmod a+x /etc/init.d/omero-web

update-rc.d -f omero-web remove
update-rc.d -f omero-web defaults 98 02
```

Start up services:

```
service redis-server start

cron
service nginx start
service omero-web restart
```

4.4.10 Maintaining OMERO.web

The following steps are run as the omero-web system user.

You can manage the sessions using the following configuration options and commands:

- Session cookies `omero.web.session_expire_at_browser_close`:
  - A boolean that determines whether to expire the session when the user closes their browser. See Django Browser-length sessions vs. persistent sessions documentation for more details. The default value is `True`:

    ```
    omero config set omero.web.session_expire_at_browser_close "True"
    ```

  - The age of session cookies, in seconds. The default value is `86400`:

```
omero config set omero.web.session_cookie_age 86400
```

- Clear session:

  Each session for a logged-in user in OMERO.web is kept in the session store. Stale sessions can cause the store to grow with time. OMERO.web uses by default the OS file system as the session store backend and does not automatically purge stale sessions, see Django file-based session documentation for more details. It is therefore the responsibility of the OMERO administrator to purge the session cache using the provided management command:

  ```
  omero web clearsessions
  ```

  It is recommended to call this command on a regular basis, for example as a `daily cron job`, see Django clearing the session store documentation for more information.

4.4.11 Troubleshooting

The following steps are run as the omero-web system user.

In order to identify why OMERO.web is not available run `omero web status`. Then consult NGINX `error.log` and `/opt/omero/web/omero-web/var/log/OMEROweb.log`.

4.4.12 Configuring Gunicorn advanced options

OMERO.web deployment can be configured with sync and async workers. **Sync workers** are faster and recommended for a data repository with *Download restrictions*. If you wish to offer users the ability to download data then you have to use **async workers**. OMERO.web is able to handle multiple clients on a single worker thread switching context as necessary while streaming binary data from OMERO.server. Depending on the traffic and scale of the repository you should configure connections and speed limits on your server to avoid blocking resources. We recommend you run benchmark and performance tests. It is also possible to apply *Download restrictions* and offer alternative access to binary data.

Note: Handling streaming request/responses requires proxy buffering to be turned off. For more details refer to Gunicorn deployment and NGINX configuration.

Note: `omero.web.application_server.max_requests` should be set to 0

See Gunicorn design for more details.

Experimental: Sync workers

The following steps are run as root.

Install futures:

```
/opt/omero/web/venv3/bin/pip install futures
```

The following steps are run as the omero-web system user.

To find out more about the number of worker threads for handling requests, see Gunicorn threads. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class
omero config set omero.web.wsgi_threads $(2-4 x NUM_CORES)
```

Experimental: Async workers

The following steps are run as root.

Install Gevent >= 0.13:

```
/opt/omero/web/venv3/bin/pip install 'gevent>=0.13'
```

The following steps are run as the omero-web system user.

To find out more about the maximum number of simultaneous clients, see Gunicorn worker-connections. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class gevent
omero config set omero.web.wsgi_worker_connections 1000
omero config set omero.web.application_server.max_requests 0
```

4.5 OMERO.web installation on Debian 9 and IcePy 3.6

Please first read server installation on Debian 9.

This is an example walkthrough for installing OMERO.web in a **virtual environment** using a dedicated system user. Installing OMERO.web in a virtual environment is the preferred way. For convenience in this walkthrough, we will use the **omero-web system user** and define the main OMERO.web configuration options as environment variables. Since 5.6, a new `OMERODIR` variable is used, you should first unset `OMERO_HOME` (if set) before beginning the installation process. By default, Python 3.5 is installed.

The following steps are run as root.

If required, first create a local system user omero-web and create directory:

```
useradd -m omero-web

mkdir -p /opt/omero/web/omero-web/etc/grid
chown -R omero-web /opt/omero/web/omero-web
```

4.5.1 Installing prerequisites

The following steps are run as root.

Install dependencies:

```
apt-get update

apt-get -y install unzip

apt-get -y install python3
apt-get -y install python3-venv

apt-get -y install nginx
```

Optional: if you wish to use the Redis cache, install Redis:

```
apt-get -y install redis-server

service redis-server start
```

4.5.2 Creating a virtual environment

The following steps are run as root.

Create the virtual environment. This is the recommended way to install OMERO.web:

```
python3 -mvenv /opt/omero/web/venv3
```

Install ZeroC IcePy 3.6:

```
/opt/omero/web/venv3/bin/pip install --upgrade https://github.com/ome/zeroc-ice-py-
↪debian9/releases/download/0.2.0/zeroc_ice-3.6.5-cp35-cp35m-linux_x86_64.whl
```

Install OMERO.web:

```
/opt/omero/web/venv3/bin/pip install "omero-web>=5.6.1"
```

4.5.3 Installing OMERO.web apps

A number of apps are available to add functionality to OMERO.web, such as OMERO.figure and OMERO.iviewer. See the main website for a list of released apps. These apps are optional and can be installed, as the **root user**, via **`pip`** to your OMERO.web virtual environment and configured as the **omero-web system user**, at any time.

4.5.4 Configuring OMERO.web

The following steps are run as the omero-web system user.

For convenience the main OMERO.web configuration options have been defined as environment variables. You can either use your own values, or alternatively use the following ones:

```
export WEBSESSION=True
export OMERODIR=/opt/omero/web/omero-web
```

Configure OMERO.web and create the NGINX OMERO configuration file to be included in a system-wide NGINX configuration by redirecting the output of the command `omero web config nginx` below into a file. If an attempt is made to access OMERO.web whilst it is not running, the generated NGINX configuration file will automatically display a maintenance page:

```
export PATH=/opt/omero/web/venv3/bin:$PATH


omero web config nginx --http "${WEBPORT}" --servername "${WEBSERVER_NAME}" > /opt/
↪omero/web/omero-web/nginx.conf.tmp
```

OMERO.web offers a number of configuration options. The configuration changes **will not be applied** until Gunicorn is restarted using `omero web restart`. The Gunicorn workers are managed **separately** from other OMERO processes. You can check their status or stop them using `omero web status` or `omero web stop`.

- Session engine:
- OMERO.web offers alternative session backends to automatically delete stale data using the cache session store backend, see Django cached session documentation for more details.
- Redis requires django-redis in order to be used with OMERO.web. We assume that Redis has already been installed. To configure the cache, run:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.
↪cache.
RedisCache", "LOCATION": "redis://127.0.0.1:6379/0"}}'
```

- After installing all the cache prerequisites set the following:

```
omero config set omero.web.session_engine django.contrib.sessions.
↪backends.cache
```

- Use a prefix:

  By default OMERO.web expects to be run from the root URL of the webserver. This can be changed by setting `omero.web.prefix` and `omero.web.static_url`. For example, to make OMERO.web appear at *http://example.org/omero/*:

```
omero config set omero.web.prefix '/omero'
omero config set omero.web.static_url '/omero/static/'
```

  and regenerate your webserver configuration.

All configuration options can be found on various sections of developers/index.html#web_index developers documentation. For the full list, refer to *Configuration* properties.

The most popular configuration options include:

- Debug mode, see `omero.web.debug`.
- Customizing OMERO clients e.g. to add your own logo to the login page (`omero.web.login_logo`) or use an index page as an alternative landing page for users (`omero.web.index_template`). See *OMERO.web UI customization* for further information.
- Enabling a public user see *Publishing data using OMERO.web*.

4.5.5 Configuring Gunicorn

The following steps are run as the omero-web system user.

Additional settings can be configured by changing the properties below. Before changing the properties, run `export PATH=/opt/omero/web/venv3/bin:$PATH`:

- `omero.web.wsgi_workers` to (2 x NUM_CORES) + 1

> **Note:** **Do not** scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

- `omero.web.wsgi_args` Additional arguments. For more details check Gunicorn Documentation. For example to enable **debugging**, run the following command:

```
omero config set omero.web.wsgi_args -- "--log-level=DEBUG --error-logfile=/opt/
↪omero/web/omero-web/var/log/error.log"
```

4.5.6 Setting up CORS

The following steps are run as root.

Cross Origin Resource Sharing allows web applications hosted at other origins to access resources from your OMERO.web installation. This can be achieved using the django-cors-headers app with additional configuration of OMERO.web. See the django-cors-headers page for more details on the settings:

```
/opt/omero/web/venv3/bin/pip install 'django-cors-headers<3.3'
```

The following steps are run as the omero-web system user.

Configure CORS. An `index` is used to specify the ordering of middleware classes. It is important to add the `CorsMiddleware` as the first class and `CorsPostCsrfMiddleware` as the last. You can specify allowed origins in a whitelist, or allow all, for example:

```
omero config append omero.web.middleware '{"index": 0.5, "class": "corsheaders.
↪middleware.CorsMiddleware"}'
omero config append omero.web.middleware '{"index": 10, "class": "corsheaders.
↪middleware.CorsPostCsrfMiddleware"}'
omero config set omero.web.cors_origin_whitelist '["https://hostname.example.com"]'
# or to allow all
omero config set omero.web.cors_origin_allow_all True
```

4.5.7 Configuring NGINX

The following steps are run as root.

Copy the generated configuration file into the NGINX configuration directory, disable the default configuration and start NGINX:

```
sed -i.bak -re 's/( default_server.*)/; #\1/' /etc/nginx/nginx.conf
mv /etc/nginx/sites-available/default /etc/nginx/sites-available/default.disabled
if [ -f /etc/nginx/sites-enabled/default ]; then
    rm /etc/nginx/sites-enabled/default
fi
cp /opt/omero/web/omero-web/nginx.conf.tmp /etc/nginx/conf.d/omeroweb.conf

service nginx start
```

For production servers you may need to add additional directives to the configuration file, for example to enable HTTPS. As an alternative to manually modifying the generated file you can generate a minimal configuration and include this in your own manually created NGINX file, such as `/etc/nginx/conf.d/omero-web.conf`:

```
omero web config nginx-location > /opt/omero/web/omero-web/omero-web-
↪location.include
```

This requires more initial work but in the future you can automatically regenerate your OMERO.web configuration and your additional configuration settings will still apply.

> **Note:** If you need help configuring your firewall rules, see the sysadmins/server-security.html page.

4.5.8 Running OMERO.web

The following steps are run as root.

Install WhiteNoise:

```
/opt/omero/web/venv3/bin/pip install --upgrade whitenoise
```

Optional: Install Django Redis:

```
/opt/omero/web/venv3/bin/pip install 'django-redis<4.9'
```

The following steps are run as the omero-web system user.

Optional: Configure the cache:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.cache.
↪RedisCache","LOCATION": "redis://127.0.0.1:6379/0"}}'
omero config set omero.web.session_engine 'django.contrib.sessions.backends.cache'
```

Configure WhiteNoise and start OMERO.web manually to test the installation:

```
omero config append -- omero.web.middleware '{"index": 0, "class": "whitenoise.
↪middleware.WhiteNoiseMiddleware"}'

omero web start

# Test installation e.g. curl -sL localhost:4080

omero web stop
```

4.5.9 Automatically running OMERO.web

The following steps are run as root.

Should you wish to run OMERO.web automatically, a *init.d* file could be created. See below an example file *omero-web-init.d*:

```
#!/bin/bash
#
# /etc/init.d/omero-web
# Subsystem file for "omero" web
#
### BEGIN INIT INFO
# Provides:             omero-web
# Required-Start:       $local_fs $remote_fs $network $time omero postgresql
# Required-Stop:        $local_fs $remote_fs $network $time omero postgresql
# Default-Start:        2 3 4 5
# Default-Stop:         0 1 6
# Short-Description:    OMERO.web
### END INIT INFO
#
# chkconfig: - 98 02
# description: init file for OMERO.web
###

RETVAL=0
```

(continues on next page)

(continued from previous page)

```
prog=omero-web

# Read configuration variable file if it is present
[ -r /etc/default/$prog ] && . /etc/default/$prog

OMERO_USER=${OMERO_USER:-omero-web}
OMERO=/opt/omero/web/venv3/bin/omero
OMERODIR=/opt/omero/web/omero-web
VENVDIR=${VENVDIR:-/opt/omero/web/venv3}

start() {
    echo -n $"Starting $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web start" &> /dev/null && echo -n ' OMERO.web'
    sleep 5
    RETVAL=$?
    [ "$RETVAL" = 0 ]
        echo
}

stop() {
    echo -n $"Stopping $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web stop" &> /dev/null && echo -n ' OMERO.web'
    RETVAL=$?
    [ "$RETVAL" = 0 ]
        echo
}

status() {
    echo -n $"Status $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web status"
    RETVAL=$?
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    status)
        status
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart|status}"
        RETVAL=1
esac
exit $RETVAL
```

Copy the *init.d* file, then configure the service:

```
cp omero-web-init.d /etc/init.d/omero-web
chmod a+x /etc/init.d/omero-web

update-rc.d -f omero-web remove
update-rc.d -f omero-web defaults 98 02
```

Start up services:

```
service redis-server start


service nginx start
service omero-web restart
```

4.5.10 Maintaining OMERO.web

The following steps are run as the omero-web system user.

You can manage the sessions using the following configuration options and commands:

- Session cookies `omero.web.session_expire_at_browser_close`:
  - A boolean that determines whether to expire the session when the user closes their browser. See Django Browser-length sessions vs. persistent sessions documentation for more details. The default value is `True`:

    ```
    omero config set omero.web.session_expire_at_browser_close "True"
    ```

  - The age of session cookies, in seconds. The default value is `86400`:

    ```
    omero config set omero.web.session_cookie_age 86400
    ```

- Clear session:

  Each session for a logged-in user in OMERO.web is kept in the session store. Stale sessions can cause the store to grow with time. OMERO.web uses by default the OS file system as the session store backend and does not automatically purge stale sessions, see Django file-based session documentation for more details. It is therefore the responsibility of the OMERO administrator to purge the session cache using the provided management command:

  ```
  omero web clearsessions
  ```

  It is recommended to call this command on a regular basis, for example as a `daily cron job`, see Django clearing the session store documentation for more information.

4.5.11 Troubleshooting

The following steps are run as the omero-web system user.

In order to identify why OMERO.web is not available run `omero web status`. Then consult NGINX `error.log` and `/opt/omero/web/omero-web/var/log/OMEROweb.log`.

4.5.12 Configuring Gunicorn advanced options

OMERO.web deployment can be configured with sync and async workers. **Sync workers** are faster and recommended for a data repository with *Download restrictions*. If you wish to offer users the ability to download data then you have to use **async workers**. OMERO.web is able to handle multiple clients on a single worker thread switching context as necessary while streaming binary data from OMERO.server. Depending on the traffic and scale of the repository you should configure connections and speed limits on your server to avoid blocking resources. We recommend you run benchmark and performance tests. It is also possible to apply *Download restrictions* and offer alternative access to binary data.

Note: Handling streaming request/responses requires proxy buffering to be turned off. For more details refer to Gunicorn deployment and NGINX configuration.

Note: `omero.web.application_server.max_requests` should be set to 0

See Gunicorn design for more details.

Experimental: Sync workers

The following steps are run as root.

Install futures:

```
/opt/omero/web/venv3/bin/pip install futures
```

The following steps are run as the omero-web system user.

To find out more about the number of worker threads for handling requests, see Gunicorn threads. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class
omero config set omero.web.wsgi_threads $(2-4 x NUM_CORES)
```

Experimental: Async workers

The following steps are run as root.

Install Gevent >= 0.13:

```
/opt/omero/web/venv3/bin/pip install 'gevent>=0.13'
```

The following steps are run as the omero-web system user.

To find out more about the maximum number of simultaneous clients, see Gunicorn worker-connections. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class gevent
omero config set omero.web.wsgi_worker_connections 1000
omero config set omero.web.application_server.max_requests 0
```

4.6 OMERO.web installation on Debian 10 and IcePy 3.6

Please first read server installation on Debian 10.

This is an example walkthrough for installing OMERO.web in a **virtual environment** using a dedicated system user. Installing OMERO.web in a virtual environment is the preferred way. For convenience in this walkthrough, we will use the **omero-web system user** and define the main OMERO.web configuration options as environment variables. Since 5.6, a new `OMERODIR` variable is used, you should first unset `OMERO_HOME` (if set) before beginning the installation process. By default, Python 3.6 is installed.

The following steps are run as root.

If required, first create a local system user omero-web and create directory:

```
useradd -m omero-web

mkdir -p /opt/omero/web/omero-web/etc/grid
chown -R omero-web /opt/omero/web/omero-web
```

4.6.1 Installing prerequisites

The following steps are run as root.

Install dependencies:

```
apt-get update

apt-get -y install unzip

apt-get -y install python3
apt-get -y install python3-venv

apt-get -y install nginx
```

Optional: if you wish to use the Redis cache, install Redis:

```
apt-get -y install redis-server

service redis-server start
```

4.6.2 Creating a virtual environment

The following steps are run as root.

Create the virtual environment. This is the recommended way to install OMERO.web:

```
python3 -mvenv /opt/omero/web/venv3
```

Install ZeroC IcePy 3.6:

```
/opt/omero/web/venv3/bin/pip install --upgrade https://github.com/ome/zeroc-ice-
↪debian10/releases/download/0.1.0/zeroc_ice-3.6.5-cp37-cp37m-linux_x86_64.whl
```

Install OMERO.web:

```
/opt/omero/web/venv3/bin/pip install "omero-web>=5.6.1"
```

4.6.3 Installing OMERO.web apps

A number of apps are available to add functionality to OMERO.web, such as OMERO.figure and OMERO.iviewer. See the main website for a list of released apps. These apps are optional and can be installed, as the **root user**, via **pip** to your OMERO.web virtual environment and configured as the **omero-web system user**, at any time.

4.6.4 Configuring OMERO.web

The following steps are run as the omero-web system user.

For convenience the main OMERO.web configuration options have been defined as environment variables. You can either use your own values, or alternatively use the following ones:

```
export WEBSESSION=True
export OMERODIR=/opt/omero/web/omero-web
```

Configure OMERO.web and create the NGINX OMERO configuration file to be included in a system-wide NGINX configuration by redirecting the output of the command `omero web config nginx` below into a file. If an attempt is made to access OMERO.web whilst it is not running, the generated NGINX configuration file will automatically display a maintenance page:

```
export PATH=/opt/omero/web/venv3/bin:$PATH


omero web config nginx --http "${WEBPORT}" --servername "${WEBSERVER_NAME}" > /opt/
↪omero/web/omero-web/nginx.conf.tmp
```

OMERO.web offers a number of configuration options. The configuration changes **will not be applied** until Gunicorn is restarted using `omero web restart`. The Gunicorn workers are managed **separately** from other OMERO processes. You can check their status or stop them using `omero web status` or `omero web stop`.

- Session engine:
- OMERO.web offers alternative session backends to automatically delete stale data using the cache session store backend, see Django cached session documentation for more details.
- Redis requires django-redis in order to be used with OMERO.web. We assume that Redis has already been installed. To configure the cache, run:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.
↪cache.
RedisCache", "LOCATION": "redis://127.0.0.1:6379/0"}}'
```

- After installing all the cache prerequisites set the following:

```
omero config set omero.web.session_engine django.contrib.sessions.
↪backends.cache
```

- Use a prefix:

  By default OMERO.web expects to be run from the root URL of the webserver. This can be changed by setting `omero.web.prefix` and `omero.web.static_url`. For example, to make OMERO.web appear at *http://example.org/omero/*:

```
omero config set omero.web.prefix '/omero'
omero config set omero.web.static_url '/omero/static/'
```

and regenerate your webserver configuration.

All configuration options can be found on various sections of developers/index.html#web_index developers documentation. For the full list, refer to *Configuration* properties.

The most popular configuration options include:

- Debug mode, see `omero.web.debug`.
- Customizing OMERO clients e.g. to add your own logo to the login page (`omero.web.login_logo`) or use an index page as an alternative landing page for users (`omero.web.index_template`). See *OMERO.web UI customization* for further information.
- Enabling a public user see *Publishing data using OMERO.web*.

4.6.5 Configuring Gunicorn

The following steps are run as the omero-web system user.

Additional settings can be configured by changing the properties below. Before changing the properties, run `export PATH=/opt/omero/web/venv3/bin:$PATH`:

- `omero.web.wsgi_workers` to (2 x NUM_CORES) + 1

Note: **Do not** scale the number of workers to the number of clients you expect to have. OMERO.web should only need 4-12 worker processes to handle many requests per second.

- `omero.web.wsgi_args` Additional arguments. For more details check Gunicorn Documentation. For example to enable **debugging**, run the following command:

```
omero config set omero.web.wsgi_args -- "--log-level=DEBUG --error-logfile=/opt/
↪omero/web/omero-web/var/log/error.log"
```

4.6.6 Setting up CORS

The following steps are run as root.

Cross Origin Resource Sharing allows web applications hosted at other origins to access resources from your OMERO.web installation. This can be achieved using the django-cors-headers app with additional configuration of OMERO.web. See the django-cors-headers page for more details on the settings:

```
/opt/omero/web/venv3/bin/pip install 'django-cors-headers<3.3'
```

The following steps are run as the omero-web system user.

Configure CORS. An `index` is used to specify the ordering of middleware classes. It is important to add the `CorsMiddleware` as the first class and `CorsPostCsrfMiddleware` as the last. You can specify allowed origins in a whitelist, or allow all, for example:

```
omero config append omero.web.middleware '{"index": 0.5, "class": "corsheaders.
↪middleware.CorsMiddleware"}'
omero config append omero.web.middleware '{"index": 10, "class": "corsheaders.
↪middleware.CorsPostCsrfMiddleware"}'
```

(continues on next page)

(continued from previous page)

```
omero config set omero.web.cors_origin_whitelist '["https://hostname.example.com"]'
# or to allow all
omero config set omero.web.cors_origin_allow_all True
```

4.6.7 Configuring NGINX

The following steps are run as root.

Copy the generated configuration file into the NGINX configuration directory, disable the default configuration and start NGINX:

```
sed -i.bak -re 's/( default_server.*)/; #\1/' /etc/nginx/nginx.conf
rm /etc/nginx/sites-enabled/default
cp /opt/omero/web/omero-web/nginx.conf.tmp /etc/nginx/conf.d/omeroweb.conf

service nginx start
```

For production servers you may need to add additional directives to the configuration file, for example to enable HTTPS. As an alternative to manually modifying the generated file you can generate a minimal configuration and include this in your own manually created NGINX file, such as `/etc/nginx/conf.d/omero-web.conf`:

```
omero web config nginx-location > /opt/omero/web/omero-web/omero-web-
↪location.include
```

This requires more initial work but in the future you can automatically regenerate your OMERO.web configuration and your additional configuration settings will still apply.

> **Note:** If you need help configuring your firewall rules, see the sysadmins/server-security.html page.

4.6.8 Running OMERO.web

The following steps are run as root.

Install WhiteNoise:

```
/opt/omero/web/venv3/bin/pip install --upgrade whitenoise
```

Optional: Install Django Redis:

```
/opt/omero/web/venv3/bin/pip install 'django-redis<4.9'
```

The following steps are run as the omero-web system user.

Optional: Configure the cache:

```
omero config set omero.web.caches '{"default": {"BACKEND": "django_redis.cache.
↪RedisCache","LOCATION": "redis://127.0.0.1:6379/0"}}'
omero config set omero.web.session_engine 'django.contrib.sessions.backends.cache'
```

Configure WhiteNoise and start OMERO.web manually to test the installation:

```
omero config append -- omero.web.middleware '{"index": 0, "class": "whitenoise.
↪middleware.WhiteNoiseMiddleware"}'

omero web start

# Test installation e.g. curl -sL localhost:4080

omero web stop
```

4.6.9 Automatically running OMERO.web

The following steps are run as root.

Should you wish to run OMERO.web automatically, a *init.d* file could be created. See below an example file *omero-web-init.d*:

```
#!/bin/bash
#
# /etc/init.d/omero-web
# Subsystem file for "omero" web
#
### BEGIN INIT INFO
# Provides:             omero-web
# Required-Start:       $local_fs $remote_fs $network $time omero postgresql
# Required-Stop:        $local_fs $remote_fs $network $time omero postgresql
# Default-Start:        2 3 4 5
# Default-Stop:         0 1 6
# Short-Description:    OMERO.web
### END INIT INFO
#
# chkconfig: - 98 02
# description: init file for OMERO.web
###

RETVAL=0
prog=omero-web

# Read configuration variable file if it is present
[ -r /etc/default/$prog ] && . /etc/default/$prog

OMERO_USER=${OMERO_USER:-omero-web}
OMERO=/opt/omero/web/venv3/bin/omero
OMERODIR=/opt/omero/web/omero-web
VENVDIR=${VENVDIR:-/opt/omero/web/venv3}

start() {
    echo -n $"Starting $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web start" &> /dev/null && echo -n ' OMERO.web'
    sleep 5
    RETVAL=$?
    [ "$RETVAL" = 0 ]
        echo
}

stop() {
```

(continues on next page)

(continued from previous page)

```
    echo -n $"Stopping $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web stop" &> /dev/null && echo -n ' OMERO.web'
    RETVAL=$?
    [ "$RETVAL" = 0 ]
        echo
}

status() {
    echo -n $"Status $prog:"
    su - ${OMERO_USER} -c ". ${VENVDIR}/bin/activate;OMERODIR=${OMERODIR} ${OMERO}␣
↪web status"
    RETVAL=$?
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    status)
        status
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart|status}"
        RETVAL=1
esac
exit $RETVAL
```

Copy the *init.d* file, then configure the service:

```
cp omero-web-init.d /etc/init.d/omero-web
chmod a+x /etc/init.d/omero-web

update-rc.d -f omero-web remove
update-rc.d -f omero-web defaults 98 02
```

Start up services:

```
service redis-server start


service nginx start
service omero-web restart
```

4.6.10 Maintaining OMERO.web

The following steps are run as the omero-web system user.

You can manage the sessions using the following configuration options and commands:

- Session cookies `omero.web.session_expire_at_browser_close`:
  - A boolean that determines whether to expire the session when the user closes their browser. See Django Browser-length sessions vs. persistent sessions documentation for more details. The default value is `True`:

```
omero config set omero.web.session_expire_at_browser_close "True"
```

  - The age of session cookies, in seconds. The default value is `86400`:

```
omero config set omero.web.session_cookie_age 86400
```

- Clear session:

  Each session for a logged-in user in OMERO.web is kept in the session store. Stale sessions can cause the store to grow with time. OMERO.web uses by default the OS file system as the session store backend and does not automatically purge stale sessions, see Django file-based session documentation for more details. It is therefore the responsibility of the OMERO administrator to purge the session cache using the provided management command:

```
omero web clearsessions
```

  It is recommended to call this command on a regular basis, for example as a `daily cron job`, see Django clearing the session store documentation for more information.

4.6.11 Troubleshooting

The following steps are run as the omero-web system user.

In order to identify why OMERO.web is not available run `omero web status`. Then consult NGINX `error.log` and `/opt/omero/web/omero-web/var/log/OMEROweb.log`.

4.6.12 Configuring Gunicorn advanced options

OMERO.web deployment can be configured with sync and async workers. **Sync workers** are faster and recommended for a data repository with *Download restrictions*. If you wish to offer users the ability to download data then you have to use **async workers**. OMERO.web is able to handle multiple clients on a single worker thread switching context as necessary while streaming binary data from OMERO.server. Depending on the traffic and scale of the repository you should configure connections and speed limits on your server to avoid blocking resources. We recommend you run benchmark and performance tests. It is also possible to apply *Download restrictions* and offer alternative access to binary data.

Note: Handling streaming request/responses requires proxy buffering to be turned off. For more details refer to Gunicorn deployment and NGINX configuration.

Note: `omero.web.application_server.max_requests` should be set to 0

See Gunicorn design for more details.

Experimental: Sync workers

The following steps are run as root.

Install futures:

```
/opt/omero/web/venv3/bin/pip install futures
```

The following steps are run as the omero-web system user.

To find out more about the number of worker threads for handling requests, see Gunicorn threads. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class
omero config set omero.web.wsgi_threads $(2-4 x NUM_CORES)
```

Experimental: Async workers

The following steps are run as root.

Install Gevent >= 0.13:

```
/opt/omero/web/venv3/bin/pip install 'gevent>=0.13'
```

The following steps are run as the omero-web system user.

To find out more about the maximum number of simultaneous clients, see Gunicorn worker-connections. Additional settings can be configured by changing the following properties:

```
omero config set omero.web.wsgi_worker_class gevent
omero config set omero.web.wsgi_worker_connections 1000
omero config set omero.web.application_server.max_requests 0
```

4.7 Publishing data using OMERO.web

The OMERO.web framework allows raw data to be published using built-in tools or supplied through web services to external web pages. Selected datasets can be made visible to a 'public user' using the standard OMERO permissions system, ensuring you always have control over how users can interact with your data.

There are several ways of publishing data using OMERO.web:

- using a URL to launch the web-based Image viewer, as described in ViewPort, which can be accompanied by a thumbnail. For more details of how to load the thumbnail, see WebGateway.
- embedding the image viewport directly into other web pages, for more details see ViewPort
- allowing public access to the OMERO.web data manager
- writing your own app to host your public data (see CreateApp) and then allowing public access to the chosen URL for that app

The sections below describe how you might use these features and how to set them up.

4.7.1 Configuring public user

The OMERO.web framework supports auto-login for a single username / password. This means that any public visitors to certain OMERO.web pages will be automatically logged in and will be able to access the data available to the defined 'public user'.

To set this up on your OMERO.web installation:

- Create a group with read-only permissions (the name can be anything e.g. "public-data"). We recommend read-only permissions so that the public user will not be able to modify, delete or annotate data belonging to other members.
- Create a member of this group, noting the username and password (you will enter these below). Again, the First name, Last name, Username and Password can be anything you like.

  Note: If you add this member to other groups, all data in these groups will also become publicly accessible for as long as this user remains in the group.

- Enable the `omero.web.public.enabled` property and set `omero.web.public.user` and `omero.web.public.password`:

```
$ omero config set omero.web.public.enabled True

$ omero config set omero.web.public.user '<username>'

$ omero config set omero.web.public.password '<password>'
```

- By default the public user is only allowed to perform GET requests. This means that the public user will not be able to Create, Edit or Delete data, as these require POST requests. If you want to allow these actions from the public user, you can change the `omero.web.public.get_only` property:

```
$ omero config set omero.web.public.get_only false
```

- Set the `omero.web.public.url_filter`. This filter is a regular expression that will allow only matching URLs to be accessed by the public user. If this is not set, no URLs will be publicly available.

  You need to configure the url_filter to *allow* all URLs that are required for the pages you wish to be public but to *block* any URLs that you do not want public users to access.

  Some examples are listed below:

  - To allow all URLs from a single app, such as 'webgateway', use a filter for URLs that start with the app name. For example:

```
$ omero config set omero.web.public.url_filter '^/webgateway'
```

    This filter permits all URLs needed for the full image viewer. If you wish to block webgateway URLs for downloading data, use:

```
$ omero config set omero.web.public.url_filter '^/webgateway/(?!archived_
↪files|download_as)'
```

  - You may need to allow access to additional URLs for some apps. For example, the OMERO.iviewer also uses some `webgateway` and `api` URLs:

```
$ omero config set omero.web.public.url_filter '^/iviewer|webgateway|api'
```

  - You can use the full webclient UI for public browsing of images. Attempts by public user to create, edit or delete data will fail silently with the default `omero.web.public.get_only` setting above. You may also choose to disable various dialogs for these actions such as launching scripts or OME-TIFF export, for example:

```
$ omero config set omero.web.public.url_filter '^/(webadmin/myphoto/
↪|webclient/(?!(script_ui|ome_tiff|figure_script))|webgateway/(?!(archived_
↪files|download_as))|iviewer|api)'
```

- Set the `omero.web.public.server_id` which the public user will be automatically connected to. Default: 1 (the first server in the `omero.web.server_list`):

```
$ omero config set omero.web.public.server_id 1
```

If you enable public access to the main webclient but still wish registered users to be able to log in, the login page can always be accessed using a link of the form *https://your_host/webclient/login/*.

4.7.2 Full example of hosting data for a publication

Putting the pieces of this puzzle together, the following describes the steps of a complete workflow for using OMERO to host public data associated with a publication. It is illustrated using an example publication from the Swedlow lab in Dundee, Schleicher et al, 2017 with the data hosted at https://omero.lifesci.dundee.ac.uk/pub/schleicher-et-al-2017.

Ansible playbooks can be found describing how the production server in Dundee ("nightshade") was configured in the prod-playbooks repository on GitHub.

Group setup

A group-per-publication allows the public user to be selectively added (or removed) from given publications to decide their visibility.

1. Create a dedicated read-only group to host the raw data underlying the publication (see cli/usergroup).
2. Add all the authors of the paper to this new group.
3. Once you have configured OMERO.web to create a public user (see below), add the public user as a member of the newly created read-only group.

Configuring OMERO.web

If you wish to have an automatically logged-in public user while still giving your existing OMERO users an unchanged user experience (i.e. not automatically logging them in as the public user), a dedicated, separate web server for servicing the public workflows can be added and configured to point at your existing OMERO.server. This is the workflow adopted here by adding a public OMERO.web at https://omero.lifesci.dundee.ac.uk, without changing the existing internal OMERO.web.

1. Follow the steps in *Configuring public user* above on the chosen OMERO.web.
2. Also configure *the filter on the public user* on the chosen OMERO.web by setting `omero.web.public.url_filter` to allow 'webclient' so that the full webclient is visible for the public user, and thus the Data tree with Projects and Datasets is also browsable, as well as the Tags tab and the full image viewer.

Data migration

The data to be made public will need to be in the publication group to be considered "published".

1. Move the original images into the dedicated group using OMERO.web or OMERO.cli. The CLI is best used where Images or Datasets are cross-linked to other Datasets or Projects in the original group. The command `omero chgrp Project:$ID --include Dataset,Image` cuts the cross-links in the original group and preserves the Project/Dataset/Image hierarchy prepared for the move by the author.
2. If you have used OMERO.figure to create your figures for publication, you can always find the original data by using the 'info' tab, as shown in the OMERO.figure Help guide (OMERO.figure supports a complete figure creation workflow, including exporting figures into image processing applications for final adjustments - see the OMERO.figure Help guide for full details).
3. Having all the data belong to one user simplifies the UI experience for public users. If necessary, ownership of data can be transferred using the 'Chown' privilege (see sysadmins/restricted-admins and users/cli/chown).

Data layout

Once the data is in the dedicated read-only group, it can be reorganized and renamed to reflect the publication e.g. Projects can be renamed according to the corresponding figure panels in the manuscript while the names of the Datasets could be retained corresponding to different treatment conditions represented in each figure panel. For example, Project Schleicher_etal_figure7_c contains images underlying the publication Figure panel 7(c). Some Projects underlie two publication figure panels, such as Project Schleicher_etal_figure2_a_c where representative images are shown in panel (a) and the corresponding quantification is shown in panel (c) of Figure 2. This makes clear which original images are underlying which figure panels in the publication.

Data can also be tagged with OMERO tags to enhance the browsing possibilities through these data for any user with basic knowledge of OMERO. For example, see Tag:Schleicher_etal_figure1_a. The tags are highlighting the images displayed in the publication figures as images. The other, non-tagged images in the group are the ones used for analysis which produced the published numerical data.

Key-Value pairs can be used to add more detailed information about the study and publication. For example, go to Schleicher_etal_figure1_a and expand the 'Key-Value Pairs' section in the right-hand pane to display the content (see the Managing data guide for information on using Key-Value pairs).

Configuring URLs

The URL of the first Project (corresponding to the first figure in the publication) can be used for a DOI and data landing page. For example, Project 'Schleicher_etal_figure1_a' https://omero.lifesci.dundee.ac.uk/webclient/?show=project-27936 corresponds to http://dx.doi.org/10.17867/10000109.

Optionally, you can decide on a set pattern of URLs for this and future publications. For example, in Dundee we have established a pattern which supposes every new publication from our institution will be in a separate group, and this group will be directly navigable by the public user using the syntax: "server-address/pub/publication-identifier". This means for example, https://omero.lifesci.dundee.ac.uk/pub/schleicher-et-al-2017 is the link where "omero.lifesci.dundee.ac.uk" is the server address, and "schleicher-et-al-2017" is the publication-identifier.

This makes use of redirects allowing https://omero.lifesci.dundee.ac.uk/pub/schleicher-et-al-2017 to link to the correct group and Project in OMERO, just as the DOI above does. Redirects need to be set in the NGINX component of the OMERO.web installation dedicated to publication workflows. You can find our configuration for this example here on GitHub:

```
location /pub/schleicher-et-al-2017 {
    return 307 /webclient/?show=project-27936;
}
```

4.8 OMERO.web UI customization

The OMERO.web offer a flexible user interface that can be customized. The sections below describe how to set up these features.

Note that depending on the deployment choice, OMERO.web will not activate configuration changes until Gunicorn is restarted using **`omero web restart`**.

4.8.1 Index page

This allows you to add a homepage at <your-omero-server>/index/. Visitors to your root url at <your-omero-server>/ will get redirected here instead of redirecting to <your-omero-server>/webclient/.

Create new custom template in `/your/path/to/templates/mytemplate/index.html` and add the following:

```
$ omero config append omero.web.template_dirs '"/your/path/to/templates/"'
$ omero config set omero.web.index_template 'mytemplate/index.html'
```

4.8.2 Login page logo

`omero.web.login_logo` allows you to customize the webclient login page with your own logo. Logo images should ideally be 150 pixels high or less and will appear above the OMERO logo. You will need to host the image somewhere else and link to it with:

```
$ omero config set omero.web.login_logo 'http://www.url/to/image.png'
```

4.8.3 Login redirection

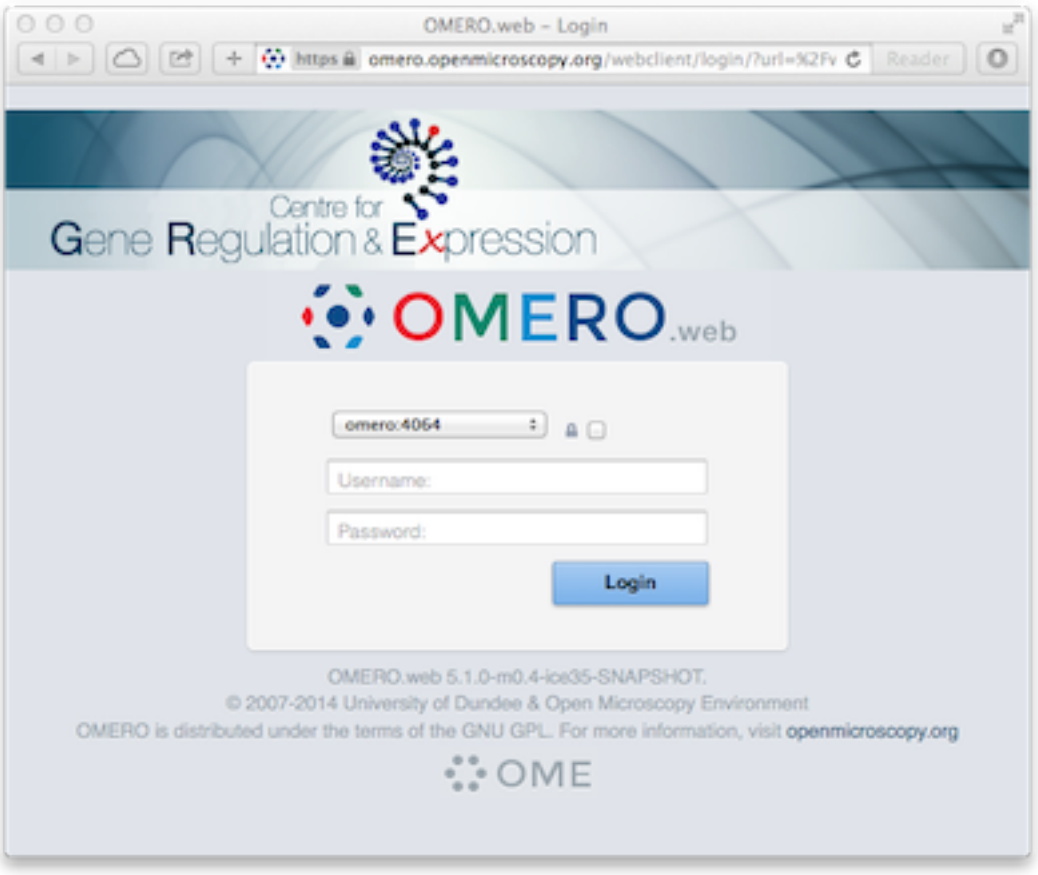
`omero.web.login_redirect` property redirects to the given location after logging in to named pages. In the example below, a user who tries to visit the `"webindex"` URL (`/webclient/`) will be redirected after login to a URL defined by the viewname `"load_template"`. The `"args"` are additional arguments to pass to Django's `reverse()` function and the `"query_string"` will be added to the URL:

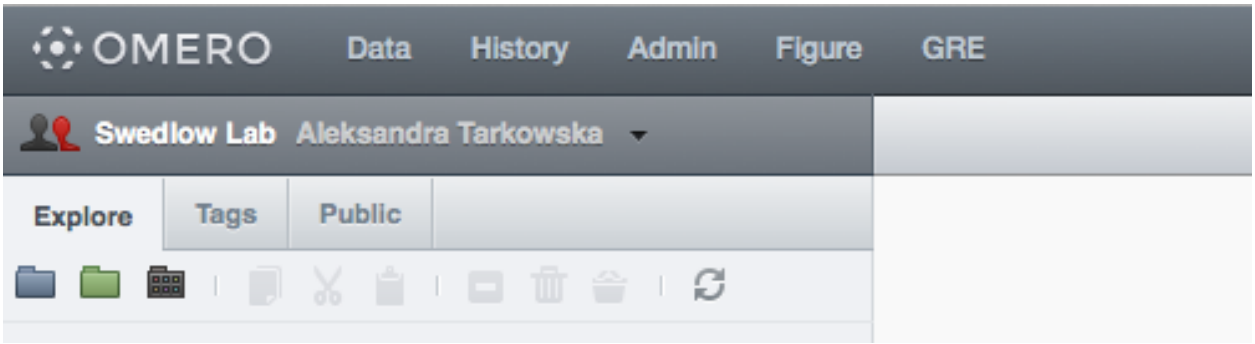
```
$ omero config set omero.web.login_redirect '{"redirect": ["webindex"], "viewname":
↪"load_template", "args":["userdata"], "query_string": "experimenter=-1"}'
```

4.8.4 Top links menu

`omero.web.ui.top_links` adds links to the top header:

```
$ omero config append omero.web.ui.top_links '["Figure", "figure_index", {"title":
↪"Open Figure in new tab", "target": "_blank"}]'
$ omero config append omero.web.ui.top_links '["GRE", "http://lifesci.dundee.ac.uk/gre
↪"]'
```


OMERO.web – Login
https omero.openmicroscopy.org/webclient/login/?url=%2Fv
Reader
Centre for
Gene Regulation & Expression
OMERO.web
omero:4064
Username:
Password:
Login
OMERO.web 5.1.0-m0.4-ice35-SNAPSHOT.
© 2007-2014 University of Dundee & Open Microscopy Environment
OMERO is distributed under the terms of the GNU GPL. For more information, visit openmicroscopy.org
OME


OMERO
Data
History
Admin
Figure
GRE
Swedlow Lab
Aleksandra Tarkowska
Explore
Tags
Public

4.8.5 Open With option

`omero.web.open_with` adds items to the 'Open with' options. This allows users to open selected images or other data with another web app or URL. See LinkingFromWebclient.

4.8.6 Include template in every page

An HTML template specified by `omero.web.base_include_template` will be included in every HTML page in OMERO.web. The template is inserted just before the `</body>` tag and can be used for adding a `<script>` such as Google analytics.

For example, create a file called `/your/path/to/templates/base_include.html` with:

```
<script>
    console.log("Hello World");
</script>
```

Set the following:

```
$ omero config append omero.web.template_dirs '"/your/path/to/templates/"'
$ omero config set omero.web.base_include_template 'base_include.html'
```

4.8.7 Group and Users in dropdown menu

Customize the groups and users dropdown menu by changing the labels or hiding the entire list:

```
$ omero config set omero.client.ui.menu.dropdown.leaders.label "Owners"
$ omero config set omero.client.ui.menu.dropdown.leaders.enabled true
$ omero config set omero.client.ui.menu.dropdown.colleagues.label "Members"
$ omero config set omero.client.ui.menu.dropdown.colleagues.enabled true
$ omero config set omero.client.ui.menu.dropdown.everyone.label "All Members"
$ omero config set omero.client.ui.menu.dropdown.everyone.enabled false
```

4.8.8 Orphaned container

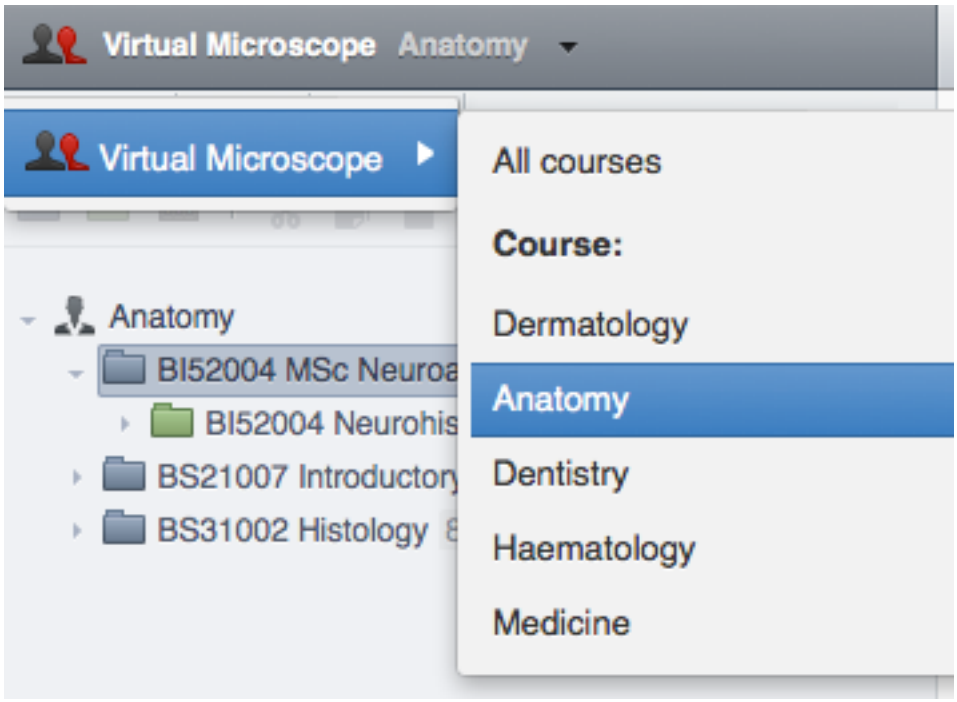
`omero.client.ui.tree.orphans.name` allows you to change the name of the "Orphaned images" container located in the client data manager tree:

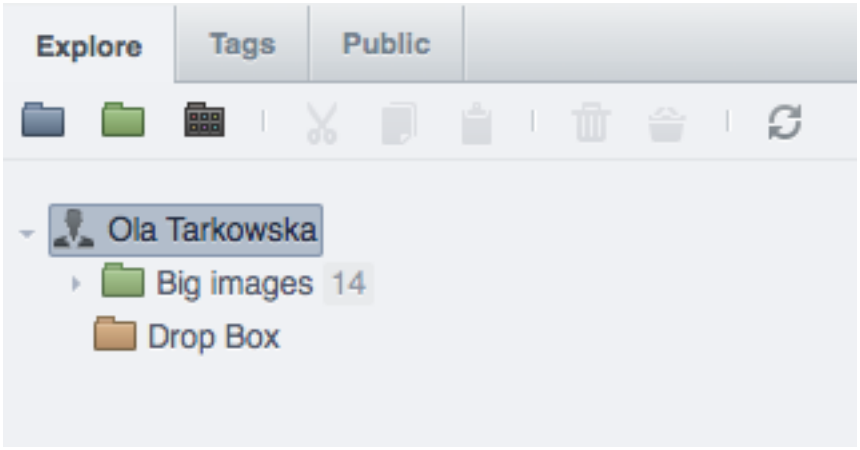
```
$ omero config set omero.client.ui.tree.orphans.name "Orphaned images"
```

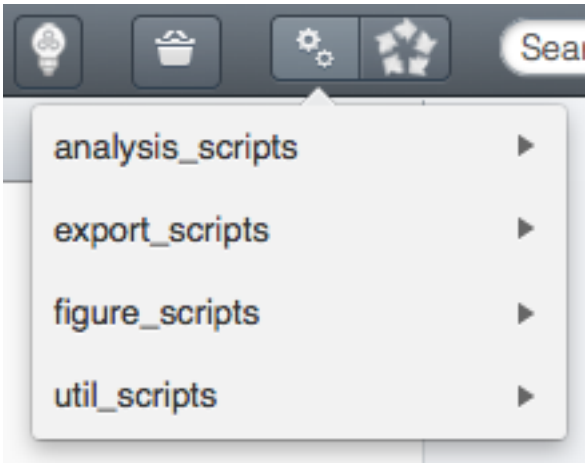
4.8.9 Disabling scripts

`omero.client.scripts_to_ignore` hides the scripts that the clients should not display:

```
$ omero config append omero.client.scripts_to_ignore "/my_scripts/script.py"
```


Virtual Microscope Anatomy
Virtual Microscope
All courses
Course:
Dermatology
Anatomy
Dentistry
Haematology
Medicine
Anatomy
BI52004 MSc Neuroa
BI52004 Neurohis
BS21007 Introductory
BS31002 Histology


Explore
Tags
Public
Ola Tarkowska
Big images 14
Drop Box


analysis_scripts
export_scripts
figure_scripts
util_scripts

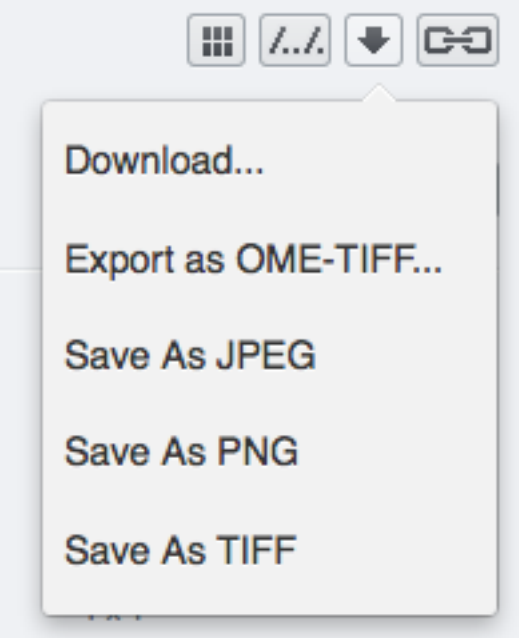
4.8.10 Download restrictions

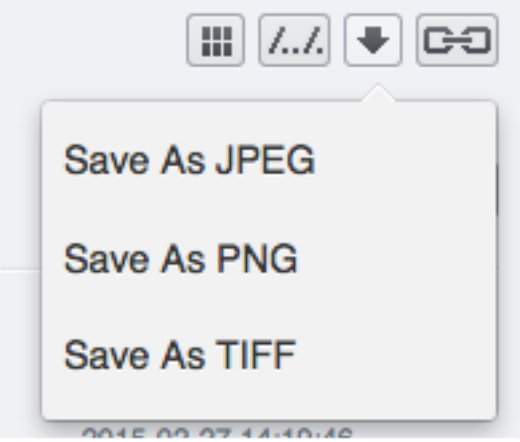
`omero.policy.binary_access` determines whether users can access binary files from disk. Binary access includes all attempts to download a file from the UI:

```
$ omero config set -- omero.policy.binary_access +read,+write,+image
```

or on a specific group with ID 15:

```
$ omero group info 15
$ omero obj map-set ExperimenterGroup:15 config -- omero.policy.binary_access +read,
↪+write,+image
```





4.9 OMERO.web upgrade

The OME team is committed to providing frequent, project-wide upgrades with security fixes, bug fixes and new functionality. We try to make the schedule for these releases as public as possible. You may want to take a look at the Trello boards for exactly what will go into a release. See also server-upgrade.

See the full details of OMERO.web features in the CHANGELOG.

This guide aims to be as definitive as possible so please do not be put off by the level of detail; upgrading should be a straightforward process.

4.9.1 Upgrade checklist

Check prerequisites

Before starting the upgrade, please ensure that you have reviewed and satisfied all the system requirements with correct versions for installation.

Upgrade

Make sure you have activated the correct virtual environment then upgrade OMERO.web via pip:

```
$ pip install --upgrade omero-web
```

If the `omero-web` upgrade *requires* an upgrade to `omero-py` (e.g. for new features), this will happen automatically above. However, even when an `omero-py` upgrade is not required, there may be some benefits to upgrading:

```
$ pip install --upgrade omero-py
```

Configuration

We now recommend that `omero-web` is installed in a separate python virtual environment.

If you are migrating to a new virtual environment, where `$OMERODIR` does not refer to a server with an existing config, you will need to export and re-import the configuration from your previous installation.

```
OLD_INSTALLATION/bin/omero config get --show-password > properties.backup

# omero-web virtual env
omero config load properties.backup
```

If you generated configuration stanzas using **`omero web config`** which enables OMERO.web via NGINX, you should regenerate your config files, remembering to merge in any of your own modifications if necessary. You should carry out this step even for minor version upgrades as there may be fixes which require it.

```
omero web config nginx > new.confg
```

More examples can be found under *Configuration*.

Plugin updates

OMERO.web plugins are very closely integrated into the webclient. For this reason, it is possible that an update of OMERO will cause issues with an older version of a plugin. It is best when updating the server to also install any available plugin updates according to their own documentation.

All official OMERO.web plugins can be installed from PyPI. You should remove all previously installed plugins and install the latest versions using pip.

Restart OMERO.web

Finally, restart OMERO.web with the following command:

```
$ omero web restart
```

Troubleshooting

If you encounter errors during an OMERO.web upgrade, etc., you should retain as much log information as possible, including the output of **`omero web diagnostics`** to the OMERO team via the mailing lists available on the support page.

Maintenance & Scaling

If you have not already done so, there are a number of additional steps that can be performed on your OMERO.web installation to improve its functioning. For example, you may need to set up a regular task to clear out any stale OMERO.web session files. More information can be found in the various walkthroughs available from *OMERO.web installation and maintenance*.

Additionally, it is recommended to use a WSGI-capable server such as NGINX. Information can be found under *OMERO.web installation and maintenance*.

Note: Support for Apache deployment has been dropped in 5.3.0.

If your organization's policies only allow Apache to be used as the external-facing web-server you should configure Apache to proxy connections to an NGINX instance running on your OMERO server i.e. use Apache as a reverse proxy. For more details see Apache mod_proxy documentation.

INDEX

Symbols

E

O