How to Setup GakuNin RDM Development Environment

学認 RDM 開発環境セットアップ

This document describes the details on how to setup the GakuNin RDM Development environment with Wasabi Lab.

References

GakuNin RDM Open Source

- https://github.com/RCOSDP/RDM-developer-guide/blob/master/Environment.md
- [FYI] AWS ECS: https://github.com/RCOSDP/RDM-developer-guide/blob/master/SetupOsfOnEC2.md (No confirmed, but is the official document provided from NII)

Setup Development Lab

VM Instance

The following is the instance used for the GakuNin RDM development environment:

CPU: 2 Core
Memory: 32GB (could be less but the docker image build may fail)
Storage: 100GB or more

NOTE: If you are to create a new VM with a clean Ubuntu installation, you will need to set the IP Address of your server static.

How to setup static IP Address:

Copy 00-network-config.yaml to /etc/netplan/ and apply

```
# cp 00-network-config.yaml /etc/netplan/
# sudo netplan generate
# sudo netplan apply
# sudo systemctl restart systemd-networkd
```

Setup on Ubuntu 22.04 LTS

Clone Project Templates

Git Hub Repository clonining

$ git clone https://github.com/luizcarloskazuyukifukaya/GakuNinRDMLocalDevelopmentEnv

You should get ./GakuNinRDMLocalDevelopmentEnv directory created.

Inside the directory:

```
wasabi@gakuninrdm-vm-2:~/GakuNinRDMLocalDevelopmentEnv$ ls -al
total 128
drwxrwxr-x  8 wasabi wasabi  4096 Dec  3 08:26 .
drwxr-x--- 11 wasabi wasabi  4096 Dec  3 08:26 ..
-rw-rw-r--  1 wasabi wasabi   451 Dec  3 08:20 00-network-config.yaml
-rwxrwxr-x  1 wasabi wasabi   650 Dec  3 08:24 apply-bug-fixed.sh
-rwxrwxr-x  1 wasabi wasabi   250 Dec  3 08:20 assets-setup.sh
-rwxrwxr-x  1 wasabi wasabi   218 Dec  3 08:20 build-images.sh
```

```
-rwxrwxr-x  1 wasabi wasabi   254 Dec  3 08:20 copy-settings.sh
-rwxrwxr-x  1 wasabi wasabi    30 Dec  3 08:20 delete-repositories.sh
-rwxrwxr-x  1 wasabi wasabi  1372 Dec  3 08:20 downgrade-docker.sh
-rwxrwxr-x  1 wasabi wasabi   567 Dec  3 08:20 get-github-repositories.sh
drwxrwxr-x  8 wasabi wasabi  4096 Dec  3 08:24 .git
-rw-rw-r--  1 wasabi wasabi    96 Dec  3 08:20 .gitiginore
-rw-rw-r--  1 wasabi wasabi 35149 Dec  3 08:20 LICENSE
drwxrwxr-x 18 wasabi wasabi  4096 Dec  3 08:26 RDM-ember-osf-web
drwxrwxr-x  6 wasabi wasabi  4096 Dec  3 08:26 RDM-modular-file-renderer
drwxrwxr-x 20 wasabi wasabi  4096 Dec  3 08:26 RDM-osf.io
drwxrwxr-x  6 wasabi wasabi  4096 Dec  3 08:26 RDM-waterbutler
-rw-rw-r--  1 wasabi wasabi  1401 Dec  3 08:20 README.md
-rwxrwxr-x  1 wasabi wasabi    81 Dec  3 08:20 remove-all-volumes.sh
-rwxrwxr-x  1 wasabi wasabi   158 Dec  3 08:20 reset-environment.sh
-rwxrwxr-x  1 wasabi wasabi    76 Dec  3 08:20 start-core-component-services.sh
-rwxrwxr-x  1 wasabi wasabi   318 Dec  3 08:20 uninstall-environment.sh
-rwxrwxr-x  1 wasabi wasabi   142 Dec  3 08:22 update-git.sh
drwxrwxr-x  2 wasabi wasabi  4096 Dec  3 08:23 updates
```

Clone Project RDM projects

./get-github-repositories.sh

```
git config --global user.name 'Luiz Carlos Kazuyuki Fukaya'
git config --global user.email ' luizcarloskazuyukifukaya@gmail.com '
git config --global core.editor 'code --wait'
git config --global merge.tool 'code --wait "$MERGED"'
git config --global push.default simple
git config --list  # githuconfig confirmation
...
```

Apply Environment Specific Changes

There environment specific value you need to apply to the source codes. All are included in the updated directory.

Updates directory:

```
wasabi@gakuninrdm-vm-2:~/GakuNinRDMLocalDevelopmentEnv$ ls -al updates
total 56
drwxrwxr-x 2 wasabi wasabi 4096 Dec  3 08:23 .
drwxrwxr-x 8 wasabi wasabi 4096 Dec  3 08:41 ..
-rw-rw-r-- 1 wasabi wasabi 4267 Dec  3 08:20 addons.s3compat.static.settings.json
-rw-rw-r-- 1 wasabi wasabi  533 Dec  3 08:20 admin.base.settings.local.py
-rw-rw-r-- 1 wasabi wasabi 1209 Dec  3 08:20 api.base.settings.local.py
-rw-rw-r-- 1 wasabi wasabi  783 Dec  3 08:20 dev.txt
-rw-rw-r-- 1 wasabi wasabi  672 Dec  3 08:22 .docker-compose.env
-rw-rw-r-- 1 wasabi wasabi  352 Dec  3 08:20 .docker-compose.mfr.env
-rw-rw-r-- 1 wasabi wasabi  662 Dec  3 08:20 .docker-compose.osf-web.env
```

```
-rw-rw-r-- 1 wasabi wasabi 1222 Dec  3 08:20 docker-compose.override.yml
-rw-rw-r-- 1 wasabi wasabi  309 Dec  3 08:20 .docker-compose.wb.env
-rw-rw-r-- 1 wasabi wasabi 4127 Dec  3 08:20 website.settings.local.py
```

(IMPORTANT) For all files inside the update directory, replace "192.168.197.203" to YOUR_IP_ADDRESS (.e.g. 192.168.197.202).

./website/settings/local.py

```
PROTOCOL = 'https://' if SECURE_MODE else 'http://'
DOMAIN = PROTOCOL + '192.168.197.203:5000/'
INTERNAL_DOMAIN = DOMAIN
API_DOMAIN = PROTOCOL + '192.168.197.203:8000/'

USE_EXTERNAL_EMBER = True
PROXY_EMBER_APPS = True
#EMBER_DOMAIN = environ.get('EMBER_DOMAIN', 'localhost')
EMBER_DOMAIN = environ.get('EMBER_DOMAIN', '192.168.197.203')
```

Once done, aply the changes:

./apply-bug-fixes.sh

```
#!/bin/bash

cp updates/.docker-compose.env RDM-osf.io/
cp updates/.docker-compose.osf-web.env RDM-osf.io/
cp updates/.docker-compose.wb.env RDM-osf.io/
cp updates/.docker-compose.mfr.env RDM-osf.io/
cp updates/admin.base.settings.local.py RDM-osf.io/admin/base/settings/local.py
cp updates/website.settings.local.py RDM-osf.io/website/settings/local.py
cp updates/api.base.settings.local.py RDM-osf.io/api/base/settings/local.py
cp updates/docker-compose.override.yml ./RDM-osf.io/
# Wasabi Endpoints information updates
cp updates/addons.s3compat.static.settings.json RDM-osf.io/addons/s3compat/static/

cp updates/dev.txt RDM-osf.io/requirements/
```

Build Container Images

Installation of docker and docker composer are required. So, please follow the instructions bellow:

Setup docker

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-22-04

Setup docker compose

https://docs.docker.jp/v1.12/compose/install.html

Build Docker images for each module

./build-images.sh

```
sudo docker build RDM-ember-osf-web -t rdm-osf-web:dev
sudo docker build RDM-modular-file-renderer -t rdm-mfr:dev
sudo docker build RDM-waterbutler -t rdm-wb:dev
sudo docker build RDM-osf.io -t rdm-osf:dev
```

ATTENTION: This will take a very long time, mostly like half hour or more.

The success result should be like the following:

```
=> [85/90] COPY ./addons/metadata/static/ ./addons/metadata/static/
0.1s
=> [86/90] RUN     yarn install --frozen-lockfile     && mkdir -p ./website/static/built/     && invoke
build_js_config_fil  381.1s
=> [87/90] COPY ./ ./                                                                    2.8s
=> [88/90] RUN pybabel compile -d ./website/translations                                                    1.0s
=> [89/90] RUN pybabel compile -D django -d ./admin/translations
1.0s
=> [90/90] RUN for module in        api.base.settings        admin.base.settings     ; do        export
DJANGO_SETTINGS_M  30.8s
=> exporting to image                                                                    31.3s
=> => exporting layers                                                                   31.2s
=> => writing image
sha256:fec3ea216cb3bcc855023a72c24de0bf47e6b42cfdd44d167ab7b8e884e4d439
0.0s
=> => naming to docker.io/library/rdm-osf:dev                                                          0.0s

1 warning found (use docker --debug to expand):
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format
(line 198)
wasabi@gakuninrdm-vm:~/gakuninrdm$
```

Application Configuration

Reference: https://github.com/RCOSDP/RDM-osf.io/blob/develop/README-docker-compose.md#docker-and-os-setup > Application Configuration

Note: After making changes to Environment Variables or Volume Mounts you will need to recreate the container(s)

Application Settings (OSF & OSF API local.py)

Change to Module Directory

$ cd RDM-osf.io/

Copy and Update Local Settings (./apply-bug-fixes.sh should already create the necessary files)

```
cp ./tasks/local-dist.py ./tasks/local.py
```

DOMAIN environment valuable

File: .docker-compose.env

DOMAIN=http://192.168.197.202:500/
(if this is missing, the links is not created correctly, and the navigation won't work as expected)

Application Runtime

NOTE: Running docker containers detached (-d) will execute them in the background, if you would like to view/follow their console log output use the following command.

$ docker-compose logs -f --tail 1000 web

Application Environment

Docker Compose Up Requirements

$ **sudo *docker-compose up requirements mfr_requirements wb_requirements***

NOTE: When the various requirements installations are complete these containers will exit. You should only need to run these containers after pulling code that changes python requirements or if you update the python requirements.

Run the project

Start Core Component Services (Detached)

$ sudo docker-compose up -d elasticsearch postgres mongo rabbitmq

Remove your existing node_modules and start the assets watcher (Detached)

$ rm -Rf ./node_modules

$ sudo docker-compose up -d assets

$ sudo docker-compose up -d admin_assets

NOTE: The first time the assets container is run it will take Webpack/NPM up to 15 minutes to compile resources. When you see the BowerJS build occurring it is likely a safe time to move forward with starting the remaining containers.

Then, we are ready to proceed to next step below.

Start the Services (Detached)

$ sudo docker-compose up -d mfr wb* wb_worker *fakecas sharejs

ATTENTION: Make sure the DEBUG environment variable is set to '1' with:

.docker-compose.wb.env

```
DEBUG=1
...
```

Tips: After changing the docker compose environment file, you might need to recreate the container image by executing the following command:
$ sudo docker-compose up --force-recreate --no-deps wb

Run migrations and create preprint providers

When starting with an empty database you will need to run migrations and populate preprint providers. See the Running arbitrary commands section below for instructions.

NOTE: **When starting with an empty database you will need to run migrations and populate preprint providers**. See the Running arbitrary commands section below for instructions.

Run migrations:

sudo docker-compose run --rm web python3 manage.py migrate

Populate institutions:
IMPORTANT: You must have run migrations first.

$ sudo docker-compose run --rm web python3 -m scripts.populate_institutions -e test -a

Populate preprint, registration, and collection providers:

Information: the required providers and subjects will be created automatically when you run migration.

$ sudo docker-compose run --rm web python3 manage.py populate_fake_providers

Populate citation styles:

$ sudo docker-compose run --rm web python3 -m scripts.parse_citation_styles

Start ember_osf_web

$ **sudo docker-compose up -d ember_osf_web**

Note:

https://github.com/RCOSDP/RDM-osf.io/blob/develop/README-docker-compose.md#running-arbitrary-commands

Running arbitrary commands

- View logs: `$ docker-compose logs -f --tail 100 <container_name>`
    - *NOTE: CTRL-c will exit*
- Run migrations:
    - After creating migrations, resetting your database, or starting on a fresh install you will need to run migrations to make the needed changes to database. This command looks at the migrations on disk and compares them to the list of migrations in the `django_migrations` database table and runs any migrations that have not been run.
        - `docker-compose run --rm web python3 manage.py migrate`

- Populate institutions:
  - After resetting your database or with a new install you will need to populate the table of institutions. **You must have run migrations first.**
    - `docker-compose run --rm web python3 -m scripts.populate_institutions -e test -a`
- Populate preprint, registration, and collection providers:
  - After resetting your database or with a new install, the required providers and subjects will be created automatically **when you run migrations.** To create more:
    - `docker-compose run --rm web python3 manage.py populate_fake_providers`
- Populate citation styles
  - Needed for api v2 citation style rendering.
    - `docker-compose run --rm web python3 -m scripts.parse_citation_styles`
- Start ember_osf_web
  - Needed for quickfiles feature:
    - `docker-compose up -d ember_osf_web`
- OPTIONAL: Register OAuth Scopes
  - Needed for things such as the ember-osf dummy app
    - `docker-compose run --rm web python3 -m scripts.register_oauth_scopes`
- OPTIONAL: Create migrations:
  - After changing a model you will need to create migrations and apply them. Migrations are python code that changes either the structure or the data of a database. This will compare the django models on disk to the database, find the differences, and create migration code to change the database. If there are no changes this command is a noop.
    - `docker-compose run --rm web python3 manage.py makemigrations`
- OPTIONAL: Destroy and recreate an empty database:
  - **WARNING**: This will delete all data in your database.
    - `docker-compose run --rm web python3 manage.py reset_db --noinput`

Start OS Web, API Server and Preprints (Detached)

Run services

$ sudo docker-compose up -d assets admin_assets mfr wb wb_worker fakecas sharejs worker web api admin ember_osf_web

Start the GakuNin RDM Web, API Server, Preprints, and Registries (Detached)

$ sudo docker-compose up -d worker web api admin ember_osf_web

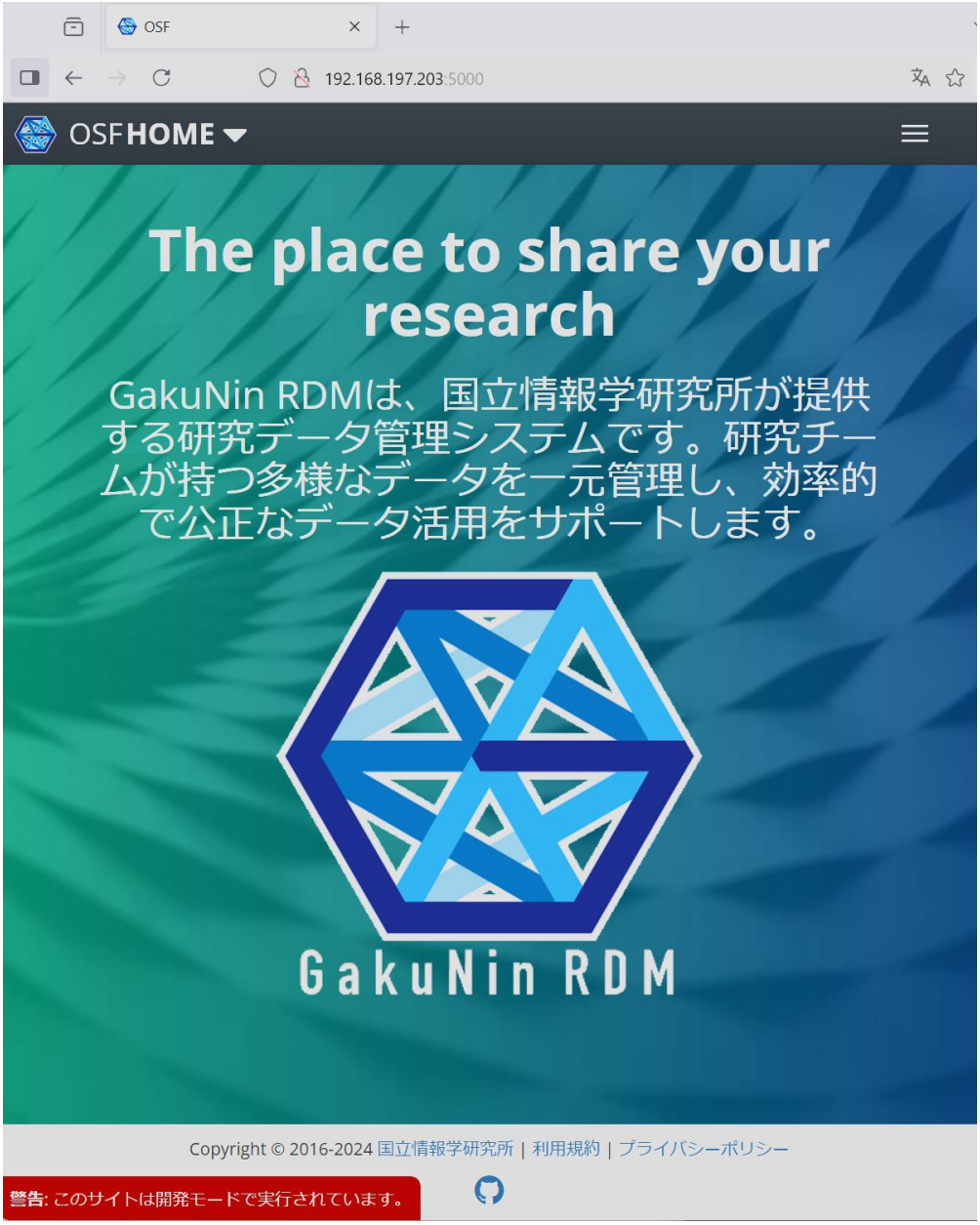
Access RMD page via web browser

http://192.168.168.167:5000/ ***(on the vm)***

$ curl http://192.168.168.167:5000/

On a remote browser:

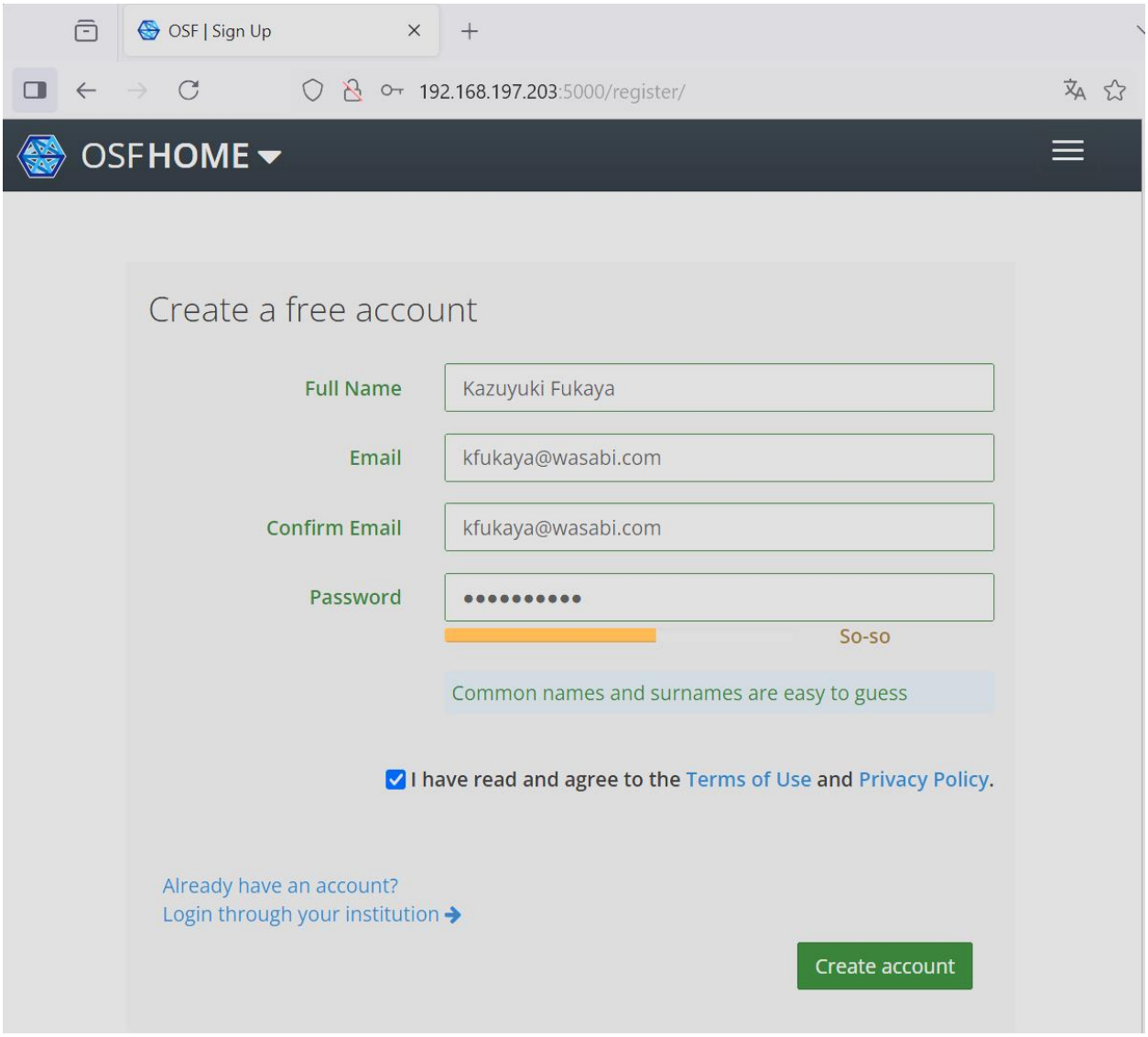
http://192.168.197.203:5000/



Setup GakuNin RDM Portal

Create User

Method 1: https://192.168.197.203:5000/register/



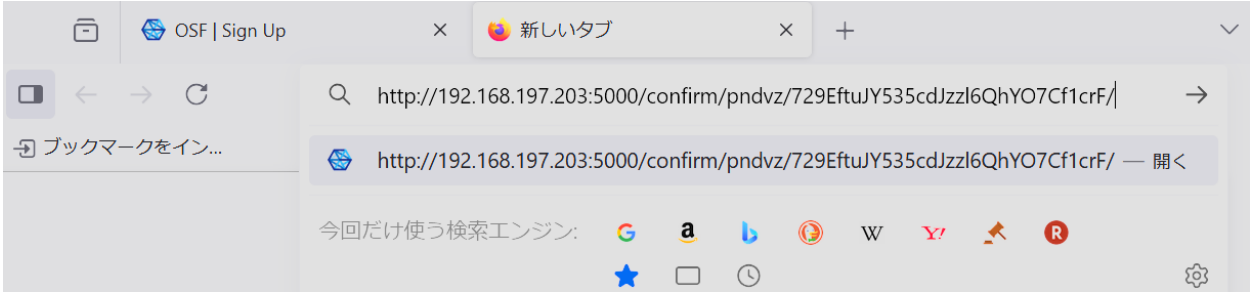
Capture web container log:

 # sudo docker-compose logs –f web

```
web_1          | <tr>
web_1          |  <td style="border-collapse: collapse;">
web_1          |   Hello Kazuyuki Fukaya,<br>
web_1          |   <br>
web_1          |   Thank you for registering for an account on the GakuNin RDM.<br>
web_1          |   <br>
web_1          |   Please verify your email address by visiting this link:<br>
web_1          |   <br>
```

```
web_1          |
http://192.168.197.203:5000/confirm/pndvz/729EftuJY535cdJzzl6QhYO7Cf1crF/<br>
web_1          |    <br>
web_1          |    The GRDM Team<br>
web_1          |
web_1          | </tr>
```

Copy the confirmation URL and paste to your web browser to activate the user



Method 2: RDM shell:
docker-compose run --rm web invoke shell

Create an OSF User with python code:

```
user = OSFUser.create(username=<your_user@cos.io>,password=<password>,fullname=<full name>)
```

Note: User's registration status is to be shifted to True by adding `user.is_registered = True` `user.have_email = True` and `user.date_confirmed = timezone.now()`.

Due to the lack of an inherited mail server in local environment, the newly created account is unable to send confirmation emails making the account unconfirmed.

Save your user with user.save() Commit the changes with commit()

```
user =
OSFUser.create(username=<your_user@cos.io>,password=<password>,fullname=<full
name>)
# user = OSFUser.objects.get(username='kfukaya@wasabi.com')

user.is_registered = True
user.have_email = True
user.admin = True
user.is_staff =  True
user.date_confirmed = timezone.now()

user.save()
commit()
```

Method 3: script

- script: (RDM-osf.io/scripts/)
  - `docker-compose run --rm web python3 -m scripts.create_fakes –user kfukaya@wasabi.com`
  -

NOTE: If the user is already created, you can use **OSFUser.objects.get(username**='email address') to create the instance of user.

Other OSF User attributes

In RDM Shell

```
from pprint import pprint
def print_object_members(obj):
  pprint(vars(obj)) # Example usage
user =
OSFUser.create(username=<your_user@cos.io>,password=<password>,fullname=<full
name>)
print_object_members(user)


user.is_admin
True
```

```
        'get_absolute_url': <function OSFUser.get_absolute_url at 0x7f415a1e9d90>,
        'get_activity_points': <function OSFUser.get_activity_points at 0x7f415a1ed598>,
        'get_addon_names': <function OSFUser.get_addon_names at 0x7f415a1e9e18>,
        'get_anonymous': <staticmethod object at 0x7f414ef49400>,
        'get_claim_url': <function OSFUser.get_claim_url at 0x7f415a1ecd90>,
        'get_confirmation_token': <function OSFUser.get_confirmation_token at 0x7f415a1eae18>,
        'get_confirmation_url': <function OSFUser.get_confirmation_url at 0x7f415a1eaea0>,
        'get_full_name': <function OSFUser.get_full_name at 0x7f415a1e9ea0>,
        'get_idp_attr': <function OSFUser.get_idp_attr at 0x7f415a24e950>,
```

```
        'get_idp_entity_ids': <function OSFUser.get_idp_entity_ids at 0x7f415a1ed950>,
        'get_next_by_created': <function curry.<locals>._curried at 0x7f415a20e400>,
        'get_next_by_date_registered': <function curry.<locals>._curried at 0x7f415a1ef7b8>,
        'get_next_by_modified': <function curry.<locals>._curried at 0x7f415a20e7b8>,
        'get_node_comment_timestamps': <function OSFUser.get_node_comment_timestamps at
0x7f415a1ed730>,
        'get_ongoing_job_school': <function OSFUser.get_ongoing_job_school at 0x7f415a1ed9d8>,
        'get_or_create_cookie': <function OSFUser.get_or_create_cookie at 0x7f415a1ed620>,
        'get_previous_by_created': <function curry.<locals>._curried at 0x7f415a20e730>,
        'get_previous_by_date_registered': <function curry.<locals>._curried at 0x7f415a1ef840>,
        'get_previous_by_modified': <function curry.<locals>._curried at 0x7f415a20e840>,
        'get_projects_in_common': <function OSFUser.get_projects_in_common at 0x7f415a1ecb70>,
        'get_recently_added': <function OSFUser.get_recently_added at 0x7f415a1eca60>,
        'get_short_name': <function OSFUser.get_short_name at 0x7f415a1e9f28>,
        'get_summary': <function OSFUser.get_summary at 0x7f415a1ec598>,
        'get_unclaimed_record': <function OSFUser.get_unclaimed_record at 0x7f415a1ecd08>,
        'get_unconfirmed_email_for_token': <function OSFUser.get_unconfirmed_email_for_token at
0x7f415a1ea840>,
        'get_unconfirmed_emails_exclude_external_identity': <function
OSFUser.get_unconfirmed_emails_exclude_external_identity at 0x7f415a1ea8c8>,
        'given_name': <django.db.models.query_utils.DeferredAttribute object at 0x7f415a1f2668>,
        'given_name_initial': <property object at 0x7f415a24ab88>,
        'given_name_ja': <django.db.models.query_utils.DeferredAttribute object at 0x7f415a1f2e48>,
        'group_connected_email_records': <django.db.models.query_utils.DeferredAttribute object at
0x7f415a1f2390>,
        'group_logs': <django.db.models.fields.related_descriptors.ReverseManyToOneDescriptor object at
0x7f4159b68358>,
        'group_role': <function OSFUser.group_role at 0x7f415a1e9d08>,
        'groups': <django.db.models.fields.related_descriptors.ManyToManyDescriptor object at
0x7f415a194320>,
        'guids': <django.contrib.contenttypes.fields.ReverseGenericManyToOneDescriptor object at
0x7f415a206cf8>,
        'has_resources': <property object at 0x7f415a1eb3b8>,
        'has_usable_username': <function OSFUser.has_usable_username at 0x7f415a1e9ae8>,
        'have_email': True,
        'id': <django.db.models.query_utils.DeferredAttribute object at 0x7f415a206b00>,
        'institutionalcontributor_set': <django.db.models.fields.related_descriptors.ReverseManyToOneDescriptor
object at 0x7f415a194c50>,
        'institutionentitlement_set': <django.db.models.fields.related_descriptors.ReverseManyToOneDescriptor
object at 0x7f4159a0f5f8>,
        'institutions': <django.db.models.fields.related_descriptors.ManyToManyDescriptor object at
0x7f415a194dd8>,
        'is_admin': <property object at 0x7f415a1eb228>,
        'is_affiliated_institution': <property object at 0x7f415a1eb2c8>,
        'is_affiliated_with_institution': <function OSFUser.is_affiliated_with_institution at 0x7f415a1ed378>,
        'is_affiliated_with_institution_id': <function OSFUser.is_affiliated_with_institution_id at 0x7f415a1ed1e0>,
        'is_allowed_storage_id': <function OSFUser.is_allowed_storage_id at 0x7f415a1ed2f0>,
        'is_allowed_storage_location_id': <function OSFUser.is_allowed_storage_location_id at 0x7f415a1ed268>,
        'is_allowed_to_use_institution': <function OSFUser.is_allowed_to_use_institution at 0x7f415a1ed400>,
        'is_anonymous': <property object at 0x7f415a24af48>,
        'is_authenticated': <property object at 0x7f415a24aef8>,
        'is_confirmed': <property object at 0x7f415a24a9f8>,
        'is_disabled': <property object at 0x7f415a24aa48>,
        'is_full_account_required_info': <property object at 0x7f415a24a7c8>,
```

```
'is_institutional_admin': <property object at 0x7f415a1eb318>,
'is_invited': <django.db.models.query_utils.DeferredAttribute object at 0x7f415a1f2160>,
'is_merged': <property object at 0x7f415a24aa98>,
'is_registered': True,
'is_staff': <django.db.models.query_utils.DeferredAttribute object at 0x7f415a1fd9e8>,
'is_super_admin': <property object at 0x7f415a1eb278>,
'is_valid_user': <property object at 0x7f415a1eb1d8>,
```

Institution Admin

Registering and Associating Users with Institutions

To test functions related to institutions, you need to associate users with institutions. Follow these steps:

Registering Institution Information

First, register institution information in the running RDM database. This is only necessary the first time:

$ sudo docker compose run --rm web python3 -m scripts.populate_institutions -e test -a

Executing this command will register test institutions like "Virginia Tech [Test]".

Associating Users with Institutions

Next, use the shell feature to link a user to an institution:

Launch the shell:

$ sudo docker compose run --rm web invoke shell

Once the Python prompt appears, execute a script to retrieve the user model:

In[1]: from osf.models import OSFUser
In[2]: user = OSFUser.objects.get(username='example@email.com')

This script retrieves the user model with the specified email address.

To associate the user with an institution, continue with the following code:

In[3]: from osf.models import Institution
In[4]: institution = Institution.objects.get(name='Virginia Tech [Test]')
In[5]: user.affiliated_institutions.add(institution)

In[6]: user.is_staff = True # Enable User as Institution Administrator
In[7]: user.save()
In[8]: commit()
Transaction committed.
New transaction opened.

This associates the user with the specified institution as administrator.

Get Institutions details

You need to get the institution name to be used when associate with a user. You can get the list of institution as the following:
Launch the shell:

$ sudo docker compose run --rm web invoke shell

Once the Python prompt appears, execute a script to retrieve the institution name list:

In [3]: [i.name for i in Institution.objects.all()]

Out[3]:

['Access to Justice Lab [Test]',

'Arizona State University [Test]',

'Brown University [Test]',

'Boys Town [Test]',

'Boston University [Test]',

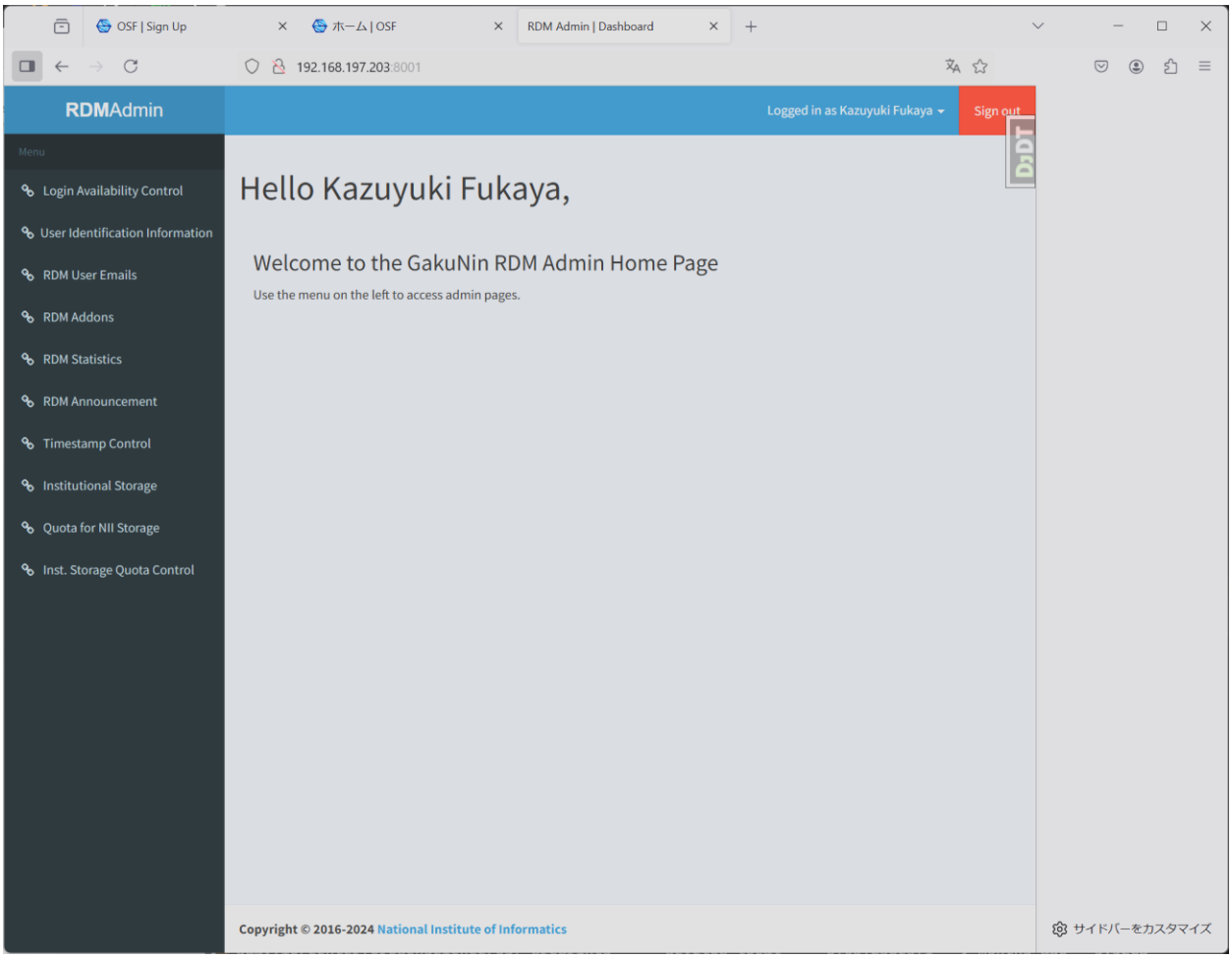
....

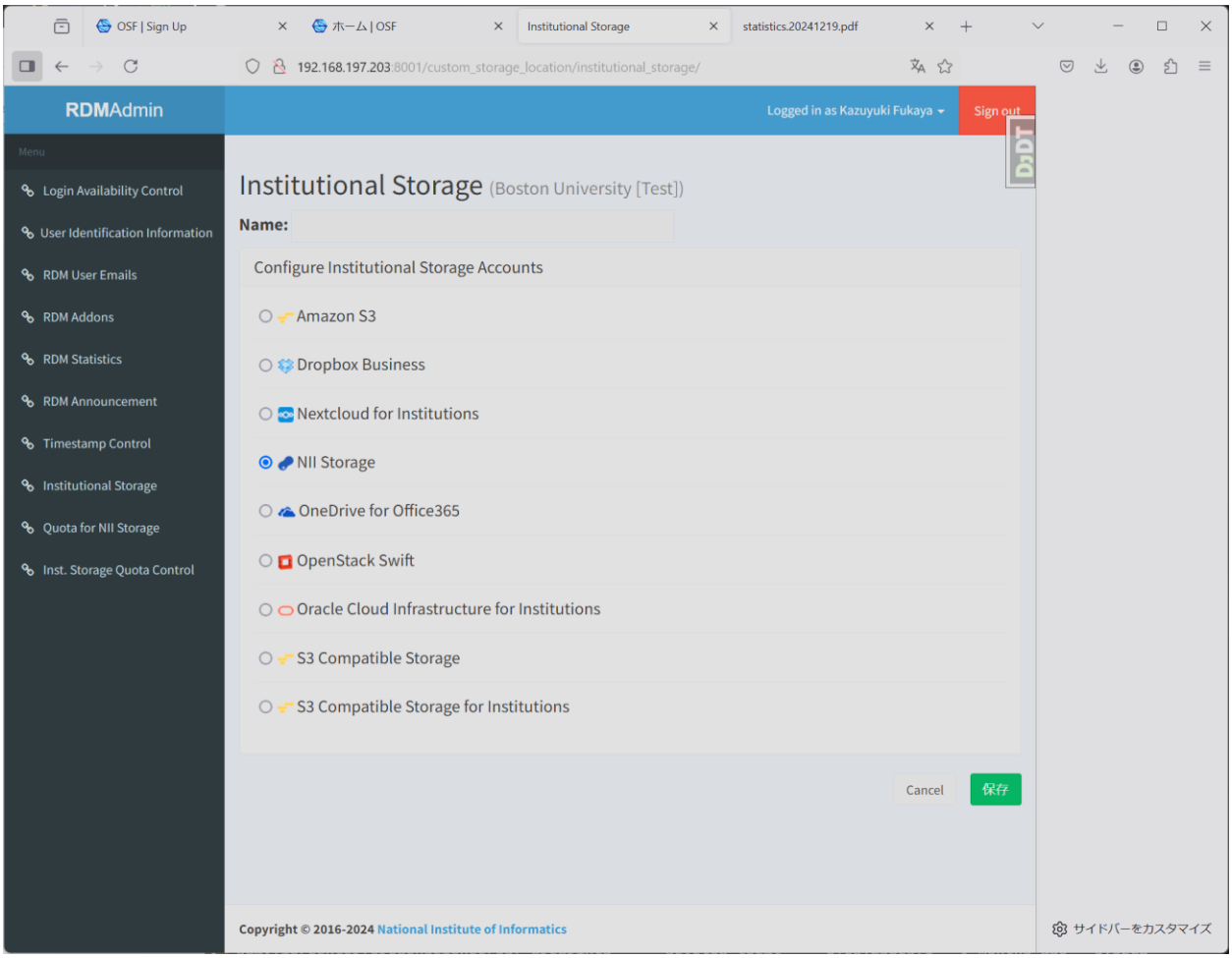
'Virginia Tech [Test]',

'Washington University in St. Louis [Test]']

Login as Institution Administrator

http://192.168.197.203:8001/


OSF | Sign Up
ホーム | OSF
RDM Admin | Dashboard
192.168.197.203:8001
RDMAdmin
Logged in as Kazuyuki Fukaya
Sign out
Menu
Login Availability Control
User Identification Information
RDM User Emails
RDM Addons
RDM Statistics
RDM Announcement
Timestamp Control
Institutional Storage
Quota for NII Storage
Inst. Storage Quota Control
Hello Kazuyuki Fukaya,
Welcome to the GakuNin RDM Admin Home Page
Use the menu on the left to access admin pages.
Copyright © 2016-2024 National Institute of Informatics
サイドバーをカスタマイズ



The user (kfukaya@wasabi.com) is registered as the administrator of ’Boston University [Test]’.

Token Creation

Need to register OAUth Scopes

- OPTIONAL: Register OAuth Scopes
  - Needed for things such as the ember-osf dummy app
    - `docker-compose run --rm web python3 -m scripts.register_oauth_scopes`

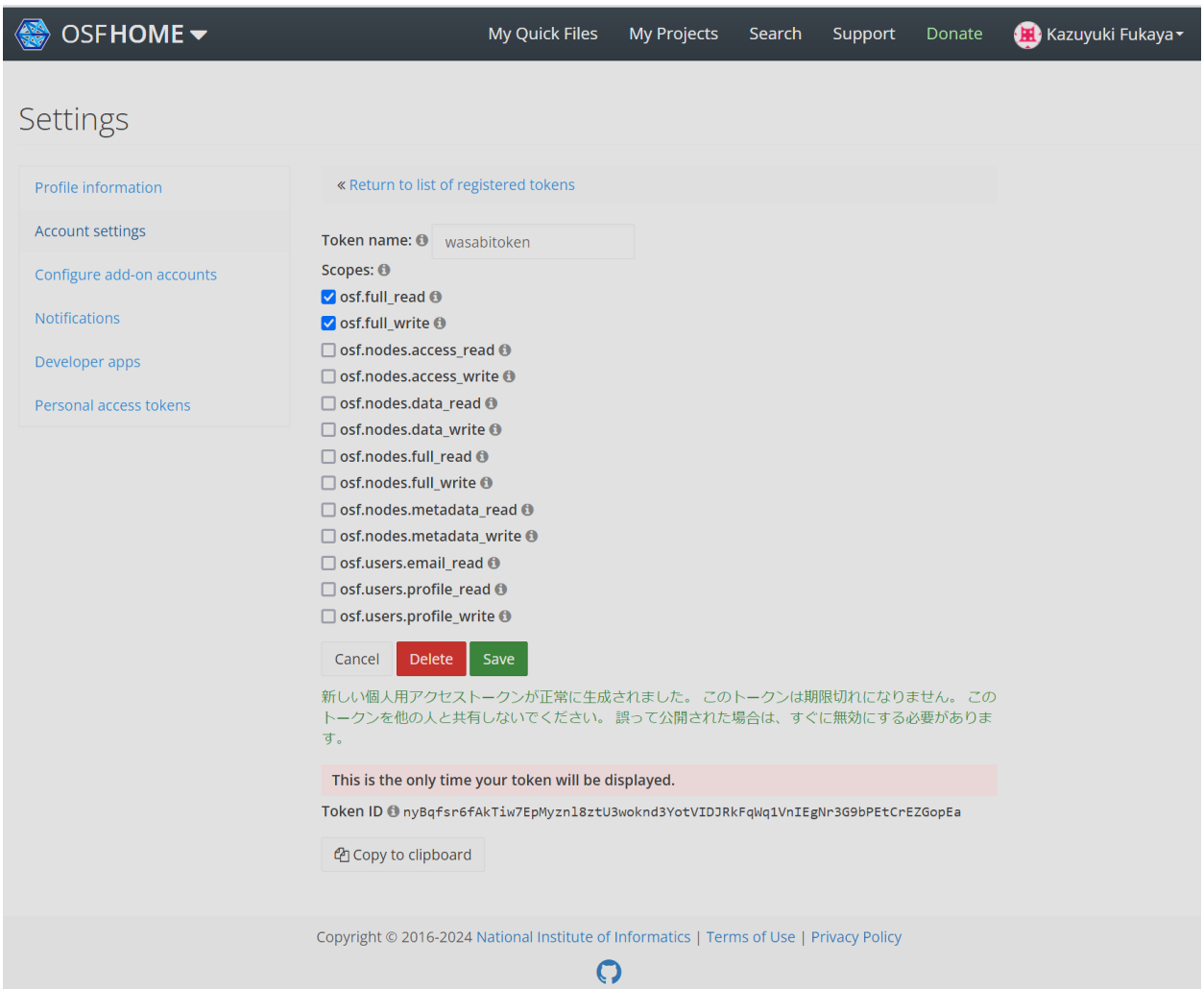
```
wasabi@gakuninrdm-vm-2:~/GakuNinRDMLocalDevelopmentEnv/RDM-osf.io$ docker-compose run --rm web python3 -m scripts.register_oauth_scopes
Creating rdm-osfio_web_run ... done
Skipping load of "webpack-assets.json" in DEBUG_MODE.
No migration settings loaded for OSFStorage, falling back to local dev. module 'website.settings' has no attribute 'MIGRATION_ENV'
No local.py settings file found
No local.py settings file found
No local.py settings file found
```

```
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
No local.py settings file found
None
[addons.binderhub.settings]  WARNING: No local.py settings file found
[raven.contrib.django.client.DjangoClient]  INFO: Raven is not configured (logging is disabled). Please
see the documentation for more information.
[addons.jupyterhub.settings]  WARNING: No local.py settings file found
[website.app]  INFO: Sentry disabled; Flask's debug mode enabled
Created new database entry for: osf.full_read
Created new database entry for: osf.full_write
Created new database entry for: osf.users.profile_read
Created new database entry for: osf.users.email_read
Created new database entry for: osf.users.profile_write
Created new database entry for: osf.nodes.metadata_read
Created new database entry for: osf.nodes.metadata_write
Created new database entry for: osf.nodes.data_read
Created new database entry for: osf.nodes.data_write
Created new database entry for: osf.nodes.access_read
Created new database entry for: osf.nodes.access_write
Created new database entry for: osf.nodes.full_read
Created new database entry for: osf.nodes.full_write
[__main__]  INFO: osf.users.create is not a publicly advertised scope; did not load into database
[__main__]  INFO: osf.admin is not a publicly advertised scope; did not load into database
```

Vm: 192.168.197.202

Token name: wasabitoken

Value: nyBqfsr6fAkTiw7EpMyznl8ztU3woknd3YotVIDJRkFqWq1VnIEgNr3G9bPEtCrEZGopEa



Newtoken: rBwEA9sVP95PpjzJOcx0FVDb86kTDONysIR21aYDX4N6L6IxOF5JMjlTZdMBy6K1TF56gR

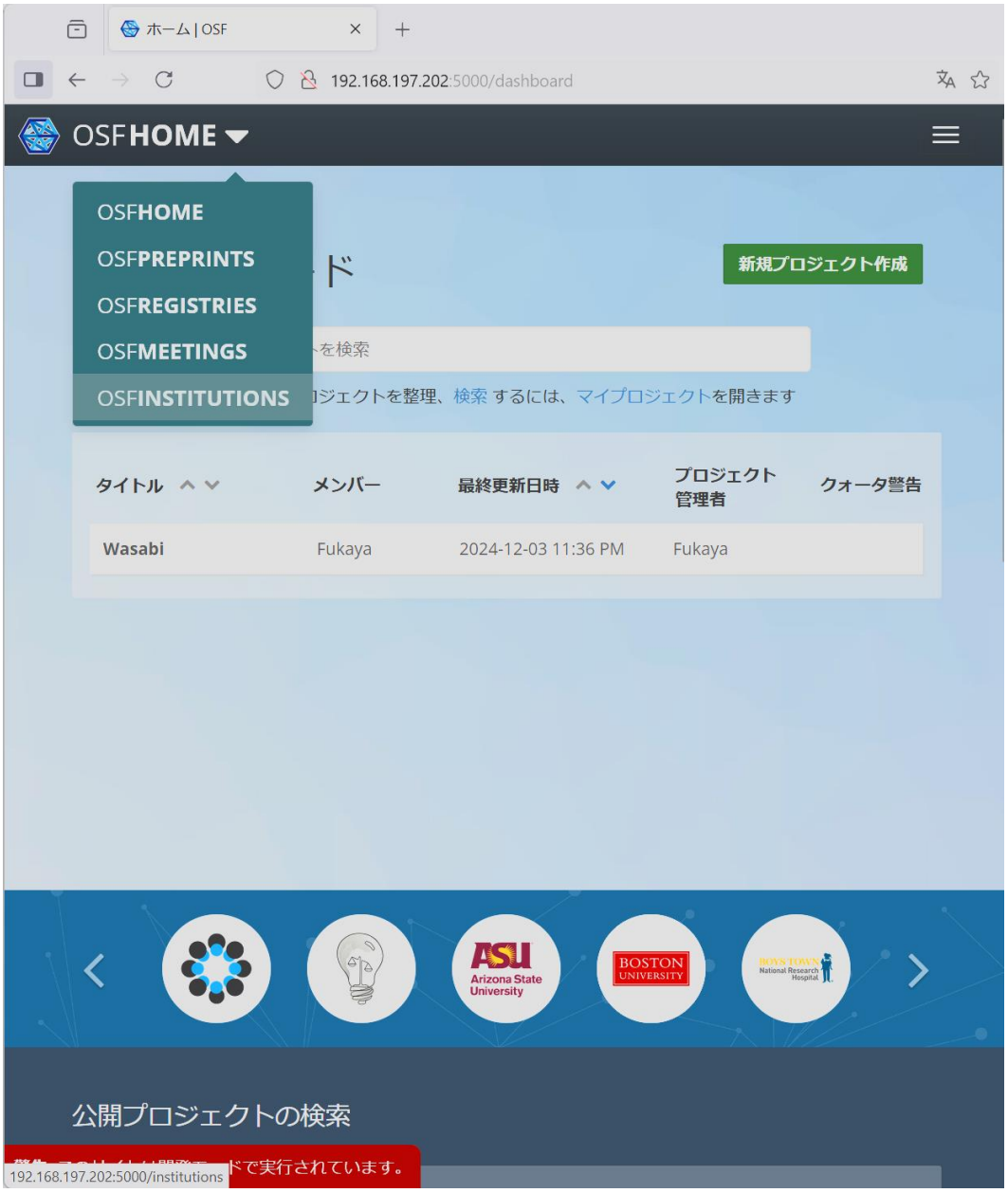
VM: 192.168.197.203

Gakuninrdmclient:
YUiSscutJOMWTB58jua6Mdj048O42NRmy2WQWhMLkRcdvLgsguy4EFq49bmjgZVKYaWGz4

Client CLI Tool (to manipulate files)

https://github.com/RCOSDP/rdmclient/blob/master/rdmclient_manual_jp.md

Institution



- Populate institutions:
  - After resetting your database or with a new install you will need to populate the table of institutions. **You must have run migrations first.**
    - ***docker-compose run --rm web python3 -m scripts.populate_institutions -e test -a***

Debug Environment

To view the logs for a given container:

docker-compose logs -f --tail 100 web

Note: CTL-c (exit log viewer)

To run a shell and perform operation within the container:

docker compose exec -it web /bin/bash

To execute a command without entering an interactive shell:

$ docker compose exec <service-name> <command>

Reset Environment

https://github.com/RCOSDP/RDM-osf.io/blob/develop/README-docker-compose.md#cleanup--docker-reset

Cleanup & Docker Reset

Resetting the Environment:

WARNING: All volumes and containers are destroyed

- `$ docker-compose down -v`

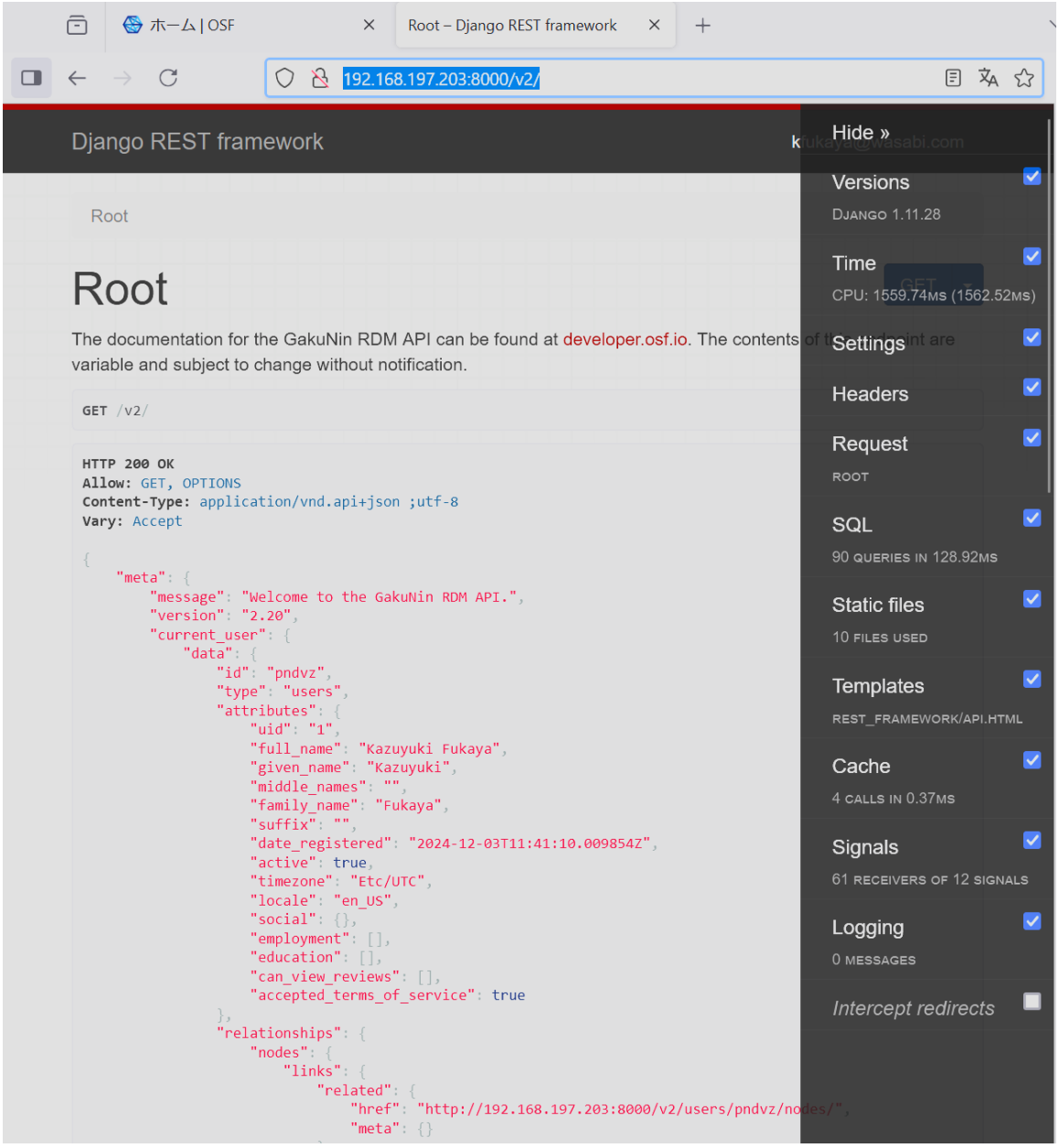
Delete a persistent storage volume:

WARNING: All postgres data will be destroyed.

- `$ docker-compose stop -t 0 postgres`
- `$ docker-compose rm postgres`
- `$ docker volume rm osfio_postgres_data_vol`

Django Models

http://192.168.197.203:8000/v2/



GiHub UPDATEs

Wasabi regions:

RDM-osf.io/addons/s3compat/static/settings.json

Some regions are not listed, and may better add those although Japanese universities not not going to use them.