Testing DSpace 7 Pull Requests

This page provides a guide for anyone to test new DSpace 7 features/bug fixes in open GitHub Pull Requests. No prior experience or technical knowledge (beyond basic command-line) is necessary. All steps are provided below.

Why test DSpace Pull Requests?

One of the larger challenges to fixing a bug or improving/adding a new feature is *end user testing.* This activity is often reliant on other developers, but can be helped along by those in non-technical roles. As a general policy, DSpace Committers **require at least two reviewers/testers** (from different institutions) for every Pull Request. While this policy helps ensure all new code is stable, it can result in an accidental "bottle-neck" or roadblock if a secondary reviewer/tester cannot be located quickly.

While we do require automated tests for every new Pull Request, automated tests are not always able to accurately replicate normal user behavior.

Therefore, when more people are able to get involved in testing new code, the result is more stable & bug-free code. Additionally, it increases the speed at which new code can be accepted – allowing us to get more features/fixes in each release of the software.

Getting Started

Required Software

In order to get started with testing DSpace Pull Requests (PRs), you need the following setup on your machine.

1. You need a GitHub login. This lets you add a comment to any PR you test, letting developers know if you found problems or if all your tests were successful.
2. Install GitHub CLI. This makes it easier to "checkout" (i.e. download) the PR code so you can test it.
3. Install Docker Desktop. This is the easiest way to get a DSpace backend installed locally for testing new (frontend or backend) PRs.
   a. *NOTE:* Docker recently changed their service agreement, but *Docker remains free* for personal use or non-commercial open source projects like DSpace.
   b. *NOTE for Windows 10:* Docker Desktop can sometimes become a bit "memory hungry" when it is running. It's best to either disable the "Start Docker Desktop when you log in" setting (so that you only run it when needed). Or, if you are using the default WSL2 backend, you can create a ".wslconfig" file in your user directory and limit the amount of memory you let Docker use (at least 4-6GB is recommended if you can spare it):

.wslconfig

```
[wsl2]
memory=6GB
```

Installing the Backend (on Docker)

If this is your first time using GitHub CLI (gh), you likely will need to authenticate with it using your GitHub.com login.

```
gh auth login
```

1. First, clone the backend codebase using GitHub CLI (gh). This downloads all the code from https://github.com/DSpace/DSpace into a local directory named "DSpace".

```
# Run from command-line where ever you want the "DSpace" directory of code to be created.
gh repo clone DSpace/DSpace

# Once downloaded, change into that directory
cd DSpace
```

2. If it is not already running, start Docker Desktop on your machine. It MUST be running for any "docker-compose" or "docker" commands to work.
    a. On Windows, if Docker Desktop is running, you'll see the Docker whale logo in your taskbar's notification area.
3. Using Docker, pull down the latest backend images from DockerHub. This downloads all the prebuilt Docker images so you don't need to rebuild them locally.

```
# This command should be run from the "DSpace" code directory
docker-compose -f docker-compose.yml -f docker-compose-cli.yml pull
```

4. Using Docker, start the Backend on Docker & *automatically install our Entities Test Data*

```
# First start the backend with the test data
docker-compose -p d7 -f docker-compose.yml -f dspace/src/main/docker-compose/db.entities.yml up -d

# Run "logs -f" to watch the logs for everything to start up.  Or, just wait a few minutes until
http://localhost:8080/server/ responds.
docker-compose -p d7 -f docker-compose.yml -f dspace/src/main/docker-compose/db.entities.yml logs -f
# (Click Ctrl+C to exit logs view)

# Finally, download the test data assestore (of files) and install it, reindexing all content
docker-compose -p d7 -f docker-compose-cli.yml -f dspace/src/main/docker-compose/cli.assetstore.yml run
dspace-cli
```

5. DONE! At this point, you should be able to go to http://localhost:8080/server/ and see a backend similar to https://api7.dspace.org/server/
    a. You should also see some test data in that backend. For example, clicking on the "collections" endpoint should return some Collections: http://localhost:8080/server/#http://localhost:8080/server/api/core/collections

Installing the Frontend (on Docker)

1. First, clone the frontend codebase using GitHub CLI (gh). This downloads all the code from https://github.com/DSpace/dspace-angular to a local directory named "dspace-angular". *You may wish to run this command from the same parent directory where you cloned the backend (just to keep this new "dspace-angular" folder next to the "DSpace" folder where the backend code resides).*

```
# Run from command-line where ever you want the "dspace-angular" directory of code to be created.
gh repo clone DSpace/dspace-angular

# Once downloaded, change into that directory
cd dspace-angular
```

2. If it is not already running, start Docker Desktop on your machine. It MUST be running for any "docker-compose" or "docker" commands to work.
    a. On Windows, if Docker Desktop is running, you'll see the Docker whale logo in your taskbar's notification area.
3. Using Docker, pull down the latest frontend images from DockerHub. This downloads all the prebuilt Docker images so you don't need to rebuild them locally.

```
# This command should be run from the "dspace-angular" code directory
docker-compose -f docker/docker-compose.yml pull
```

4. Using Docker, start the frontend/UI:

```
docker-compose -p d7 -f docker/docker-compose.yml up -d

# Optionally run "logs -f" to watch the logs for everything to start up.
docker-compose -p d7 -f docker/docker-compose.yml logs -f
# (Click Ctrl+C to exit logs view)
```

5. DONE! In a few minutes, the User Interface should be available at http://localhost:4000/   It should automatically be pointed at your Backend (also running on Docker)!

   a. Test it out by logging in using one of the demo accounts! Login: dspacedemo+admin@gmail.com , Password: dspace

> ⓘ You are now ready to test PRs! See below for how to do so.

Testing a Frontend PR

Based on the setup above, here's how to test a new User Interface / Frontend Pull Request in the "dspace-angular" project: https://github.com/DSpace/dspace-angular/pulls

1. These test instructions assume you are already running both the Frontend & Backend. If not, follow the instructions above, and then start them both:

   ```
   # Start backend
   cd DSpace
   docker-compose -p d7 up -d

   # Start frontend
   cd ../dspace-angular
   docker-compose -p d7 -f docker/docker-compose.yml up -d
   ```

2. Move into the directory where your Frontend code is:

   ```
   cd dspace-angular
   ```

3. Find the number of the Pull Request you want to test (it'll be the "#[number]" displayed after the PR name, or the end of the PR's URL). Checkout the code from that PR using it's number:

   ```
   # This would checkout https://github.com/DSpace/dspace-angular/pull/1383 for testing
   gh pr checkout 1383
   ```

4. Now, rebuild & restart your frontend based on that PR's code:

   ```
   docker-compose -p d7 -f docker/docker-compose.yml up -d --build
   # Even after the build completes, it make take a few minutes for the frontend to restart

   # You can check the status using "logs -f"
   docker-compose -p d7 -f docker/docker-compose.yml logs -f
   # (Click Ctrl+C to exit logs view)
   ```

5. Now test the User Interface to see if the changes made in that PR look to work properly. The PR's description should describe how to test it and what you should see.
   a. If you notice any odd behavior, check for errors in the UI using your browser's DevTools. See Troubleshoot an error#FindingtheErrorMessageintheUserInterface
   b. Based on what you find, add a comment to the PR. If it works, congratulate the developer. If it doesn't, let them know the error you saw (and what you clicked on when that error occurred).

Testing a Backend PR

Some frontend bugs or features also require backend changes. Here's how to install those backend PR changes. **Yes, you can even test a backend PR and frontend PR at the same time** by following the instructions in both "Testing a Frontend PR" and "Testing a Backend PR".

1. These test instructions assume you are already running both the Frontend & Backend. If not, follow the instructions above, and then start them both:

   ```
   # Start backend
   cd DSpace
   docker-compose -p d7 up -d

   # Start frontend
   cd ../dspace-angular
   docker-compose -p d7 -f docker/docker-compose.yml up -d
   ```

2. First, move into the directory where your Backend code is

```
cd DSpace
```

3. Find the number of the Pull Request you want to test (it'll be the "#[number]" displayed after the PR name, or the end of the PR's URL). Checkout the code from that PR using it's number:

```
# This would checkout https://github.com/DSpace/DSpace/pull/8007 for testing
gh pr checkout 8007
```

4. Now, rebuild & restart your backend based on that PR's code:

```
docker-compose -p d7 up -d --build
# Even after the build completes, it may take a few minutes for the backend to restart

# You can check the status using "logs -f"
docker-compose -p d7 logs -f
# (Click Ctrl+C to exit logs view)
```

5. Now test the User Interface to see if the changes made in that PR look to work properly. If there was a corresponding UI PR, make sure you also follow the instructions in "Testing a Frontend PR" to install that PR as well.
    a. If you notice any odd behavior, check for errors in the UI using your browser's DevTools. See Troubleshoot an error#FindingtheErrorMessageintheUserInterface
    b. Based on what you find, add a comment to the PR. If it works, congratulate the developer. If it doesn't, let them know the error you saw (and what you clicked on when that error occurred).

Other Tips

Need to startup the frontend & backend?

After restarting your computer, the frontend & backend will be stopped. If you previously installed both & simply need to start them up again, run the following:

```
# Start backend
cd DSpace
docker-compose -p d7 up -d

# Start frontend
cd ../dspace-angular
docker-compose -p d7 -f docker/docker-compose.yml up -d
```

Need to shutdown everything?

NOTE: While this shuts down the Docker containers, all the data/files/user accounts you created will be retained (and will appear the next time to start the containers)

```
# Stop the backend
cd DSpace
docker-compose -p d7 down

# Stop the frontend
cd ../dspace-angular
docker-compose -p d7 -f docker/docker-compose.yml down
```

Need to revert back to the "main" codebase (i.e. uninstall the PR code)?

```
# Checkout the main code (from either frontend or backend directory)
git checkout main

# Pull down any latest changes
git pull

# Rebuild/restart the frontend or backend
# For frontend: docker-compose -p d7 -f docker/docker-compose.yml up -d --build
# For backend: docker-compose -p d7 up -d --build
```

Need to remove everything from your machine?

NOTE: This removes all Docker containers from your system, including all data created in those containers. After running this, you delete everything. So, if you want to do future testing, you'd have to reinstall both the frontend & backend (using the instructions above) from scratch.

```
# First, shutdown everything that's running in Docker
docker stop $(docker ps -a -q)

# Now, delete all Docker containers from your machine
docker rm $(docker ps -a -q)

# Remove all Docker volumes (data) from your machine
docker volume rm $(docker volume ls -q)

# Finally, you may delete the "DSpace" and "dspace-angular" folders if you no longer wish to use them.
# You may also uninstall Docker Desktop and GitHub CLI if you wish.
```