Install & configure tools/plugins in the Up2U Moodle infrastructure

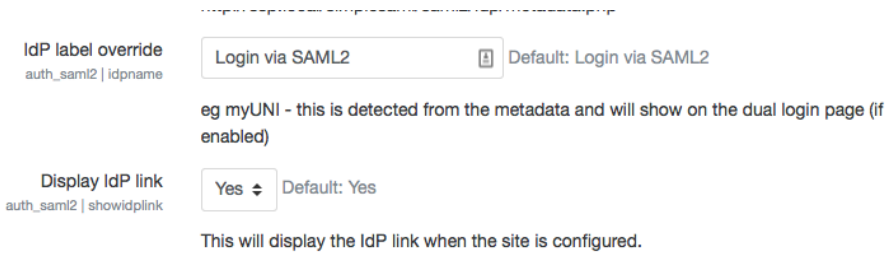
Please follow the steps below if you would like your tool to be integrated in the Up2U Moodle infrastructure.

To add or update new tools or plugins into Moodle for Up2U, we need both to install and configure them, with configuration requiring both common settings across all Up2U instances as well as settings which are specific to particular Up2U deployments. In order to simply repeated configuration and deployment, we have automated Moodle tool installation and configuration via a series of template config files and environment variable files.

Even if you have admin access to one of our deployments, do *not* install or configure tools by hand via the Moodle web interface; your changes will most likely be overwritten by future development, and will not be portable across deployments.

Step-by-step guide

1. Fork (or clone and create a branch of) our Moodle Github repository from https://github.com/up2university/docker-moodle. Follow the instructions there for building and configuring a local instance using Docker. In particular, copy and customise the files in `envs-templates/` to a folder `envs/` suit a local installation, and customise `docker-compose.yml` to point to local folders for, e.g., data. (Do *not* commit your `envs/` or `docker-compose.yml` to Github!)
2. Make sure your local Moodle instance runs.
3. Edit `moodle/Dockerfile` to add commands to download and install your tool or plugin to the appropriate locations in `/var/www/html/` inside the produced Docker image. Note that some types of tool (specifically, LTI-integrated tools) do not need any installation. In rare instances, you may need to edit `mysql/Dockerfile` too, but this is unlikely.
4. Rebuild and run your local Moodle instance.
5. In your browser, navigate to your local Moodle instance and log in as admin.
6. Configure your tool/plugin in the Moodle Site Administration interface. While doing so, make a note of the names of all Moodle configuration variables needed to configure your tool, and the values they need to have. For example, in the image below, we see settings for the SAML2 authentication plugin `auth_saml2` named `idpname` and `showidplink`, being given the values "Login via SAML2" and 1 (="Yes"), respectively.
    a. LTI tools and repository configuration are slightly different - go to step 8.



7. Create a file in `moodle/configs` named `template-<tool_name>.scsv`, copying one of the existing files in that folder. Make sure there is a line for every configuration variable which needs to be set for your tool, in the format `<tool_name>; <variable_name>; <value>` where `<tool_name>` is how Moodle refers to your tool, `<variable_name>` is the variable name and `<value>` is the value. If `<value>` should be the same across *all* Moodle instances, enter it directly here. If it will *vary* across instances, instead choose a label for it, e.g., `TOOL_NAME_VARIABLE_NAME`, and put `${TOOL_NAME_VARIABLE_NAME}` for `<value>`, e.g., `'auth_saml2'; 'showidplink'; 1 'auth_saml2'; 'idpname'; ${AUTH_SAML2_IDPNAME}`. Go to step 9.
8. For LTI and repository instances, look in `mysql/configs` and copy the relevant `template-<tool_name>*.sql` files for your tool, replacing labels (in the same format as in step 7) to fit your tool as appropriate, as well as any fixed values which should not vary across deployments. Go to step 9.
9. If you have any varying instances, create a file in `envs-templates/` for your tool called `<tool_name>.env`, and populate it in the format `TOOL_NAME_VAR1_NAME= TOOL_NAME_VAR2_NAME=` etc.
10. Copy your new `.env` file into `envs/` and edit it to contain specific values, e.g., `AUTH_SAML2_IDPNAME="eduGAIN Login"`.
11. Occasionally you may need to carry out more specific scripted customisation - look in `moodle/config/moosh-*` for examples.
12. Rebuild and run your Moodle instance. Reconfigure it with `docker-compose exec moodle /configure.sh` and `docker-compose exec mysql /configure.sh`.
13. Navigate to your local Moodle instance and verify that your settings have been applied correctly and your tool works.
14. Iterate from point 6 until it works; you may need to experiment with quoting, etc., in the `.env` files (see other `.env` files for examples)
15. Commit your changes to your branch or fork on Git (remember to exclude your `envs/` folder and `docker-compose.yml` modifications from the commit!) and raise a pull request against the main repository/branch. Separately, send concrete versions of your `.env` file(s) for each main Moodle instance to Allan and Micha, and we'll commit them to the project's private repository (`.env` files may contain authentication details /secrets which should not go on a public Github repository) and configure them for deployment

Related articles

- Install & configure tools/plugins in the Up2U Moodle infrastructure