

Custom theme plugin installation

Congratulations on your new theme! The following instructions will guide you through the process of installing your custom theme locally in AtoM. You will need command-line access to the back-end of the application to do this.

Some general resources to get you started:

- [Command-line 101](#) (slides)
 - A basic overview of some common commands in unix/linux command-line interface
 - A quick internet search will also turn up dozens of other tutorials and cheat sheets if you are interested in learning more
- [AtoM command-line tasks: an introduction](#) (slides)
 - A slide-based overview of some of AtoM's command line tasks. See also:
 - [Command-line tasks](#) documentation
 - [AtoM troubleshooting](#) documentation

Installing the custom theme

1. Download and unzip the custom theme plugin folder. Your computer should already have a utility to unpack zipped files (try just double-clicking on the zipped folder and see if you are presented with an option to unzip it), but if not, there are many free utilities available that can do this, such as [PeaZip](#), [7Zip](#), [iZip](#), the [UnArchiver](#), and more. See also:
 - [Unzipping files on Windows](#)
 - [Unzipping files on Mac](#)
 - [Unzipping files on Linux](#)
2. If the unzipped folder is not already in a location where you will be able to access both it and AtoM from the command-line, move it there now.
3. Open your command-line terminal
4. First, let's add your institutional logo, which will be shown in the AtoM header. For your theme to properly use the logo, it needs to be:
 - In PNG format

- Named `logo.png`
- Placed in the root AtoM installation directory (if you have followed our recommended installation instructions, this is generally `/usr/share/nginx/atom`)

It's not a requirement, but you may also want your logo to have a transparent background, so it will look snappy against AtoM's header.

Let's copy your institutional logo into the AtoM images directory, and rename it `logo.png`. In the following command, replace the `path/to/my/` part of the command with the actual file path to the current location of your logo, and `institution-logo.png` with the current filename of your logo:

```
cp path/to/my/institution-logo.png /usr/share/nginx/atom/logo.png
```

6. Next, we want to copy the theme plugin directory we unzipped into AtoM's plugins directory. Once again, replace the file path and name of the theme folder in the first part of the command (and the name of the custom theme folder in the last part of the second command) with the correct information on your system:

```
cp -r path/to/my/arCustomTheme /usr/share/nginx/atom/plugins/arCustomTheme/
```

7. Almost done! Now we will want to compile the custom theme (remember to swap in the actual name of your custom theme plugin in the command):

```
make -C /usr/share/nginx/atom/plugins/arCustomTheme
```

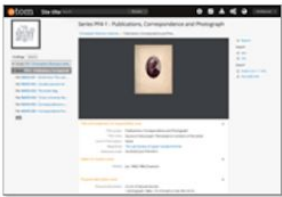

8. Now, we will navigate to AtoM's root directory and then we'll [clear the application cache](#) and [restart services](#). You can read more about what each of these commands does in [this slide deck](#), or on the [Troubleshooting](#) documentation page. The following assumes you have followed our recommended installation instructions, and are using Ubuntu 16.04 with PHP 7.0. If you are using Ubuntu 14.04, see the links for alternative commands. Below are 4 commands - run them one at a time. Note that the last command is

optional, if you are using memcached in your installation as an additional caching engine. If you're not sure, feel free to try the command, and don't worry if it doesn't work (it won't break anything):

```
cd /usr/share/nginx/atom
php symfony cc
sudo systemctl restart php7.0-fpm
sudo systemctl restart memcached
```

- 9. We should be done with the command-line for now. Open your web browser, navigate to your AtoM site, login, and then navigate to Admin > Themes. You should now see your custom theme listed on the page. Enable it, and save.

List themes

Name	Version	Enabled
 arDominionPlugin Theme plugin made from scratch with some JavaScript magic. Cross-browser compatibility tested. Based in Twitter Bootstrap 2.0, 940px two-column layout, slightly responsive.	0.0.1	<input type="radio"/>
arCustomPlugin Theme plugin, extension of arDominionPlugin.	0.0.1	<input checked="" type="radio"/>
 arArchivesCanadaPlugin Theme plugin, extension of arDominionPlugin.	0.0.1	<input type="radio"/>

Save

- 10. Congratulations! Your new custom theme should now be properly installed.

Future upgrades and custom themes

AtoM's [upgrade process](#) essentially involves installing a new version of AtoM alongside your old one, and then copying your data into it before decommissioning the old site. This means that whenever you upgrade, you will need to repeat the above steps. Keep a copy of your theme's institutional logo, the theme plugin directory, and these instructions somewhere secure, so you can find and follow them again in the future.

Additionally, please note that you may need to make manual changes to your theme over time as AtoM changes, to account for new modules, relocated files, or major overhauls of existing pages. Themes are not automatically updated, and it will be your responsibility to maintain your theme over time.

With every major version upgrade, be sure to consult the Custom theme upgrades section of the Upgrading documentation for suggestions of fixes you may need to make manually. See:

- [Upgrading with a custom theme plugin](#)

Depending on the complexity of your theme and the amount and size of changes made to AtoM in the new release, the tips included in the above link may not cover all required changes - you may need a developer who can add further customizations to your theme to make it work as expected with new pages or modules in AtoM. Below are some general resources to help your team understand how custom themes are created and maintained in AtoM.

How AtoM custom theme plugins generally work

AtoM was built using a PHP framework called [Symfony 1.x](#). AtoM custom theme plugins use Symfony's plugin architecture, which you can read about more generally [here](#). We generally recommend that new theme plugins extend AtoM's base Dominion theme, and the [arArchivesCanadaPlugin](#) is an example theme we include in all AtoM releases that demonstrates this. We have developed your custom theme following this design pattern.

This means that your custom plugin directory contains files, directives, and assets (such as background images, logos, etc) that, when found, will override AtoM's default version of these same assets. Where no alternative versions are found, AtoM will generally make use of the default theme's assets and styles.

To the right is an image showing the subdirectories found in a simple custom theme, which includes a custom homepage.

The `config` directory contains a file that registers the plugin in Symfony, so it shows up correctly in your Themes setting page, and works when you enable it. The CSS directory will contain any custom style sheets used in your theme - generally, you will find most of your customizations in the `custom.less` file found in the `less` subdirectory.

The `images` directory is where any custom assets, such as a splash image for your homepage, etc. will generally be kept.

The `modules` directory will contain files that override AtoM's existing modules. If your theme includes changes to the default menus included in AtoM for example, you will likely find those changes in a PHP file located in `modules/menus`. The number of subdirectories and files here will depend on your custom theme.

Depending on your theme, there may be other directories as well. For example, if your has a custom header, you would likely find a `_header.php` file in a `templates` subdirectory, or if your theme disables the "Browse by" menu that usually appears on the home page, then you might find a customized version of `homeSuccess.php` in the `templates` directory.

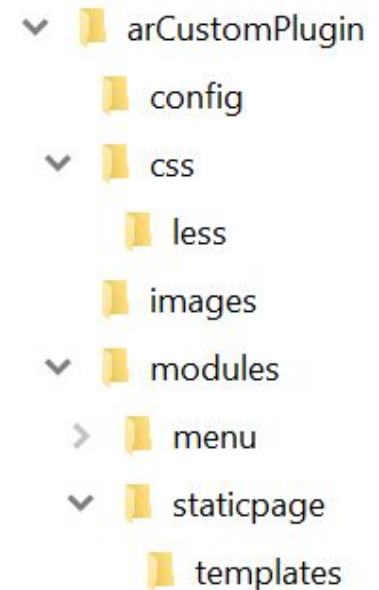
You will also find a Makefile in the root directory of your custom theme plugin. AtoM uses [lessc](#) to extend CSS and as a compiler, that allows us to minimize the CSS files for easier storage and distribution. This Makefile is necessary, and it's what allows you to compile your theme assets by running the `make` command.

Generally, any theme modifications you need to make will be made in one of these locations. You can add new CSS rules that override the base Dominion CSS to the `custom.less` file found in `css/less`. If we've added a new module or page template element to AtoM that you wish to customize in your theme, you can find the relevant file in AtoM's code, copy it into the relevant subdirectory in your custom theme plugin, and modify it there.

Additional resources

Here are some resources on how to develop custom themes for AtoM:

- [Creating custom themes](#) (documentation)
- [Creating custom themes](#) (slides)



Additionally, here are some general resources that might be useful for understanding how AtoM works and how to customize it:

- [Get to know AtoM's code base](#) (slides)
- [Development resources](#) (wiki pages)
- [Symfony 1.x documentation](#)

If you would like an easy way to install a local development environment so you can test your theme customizations before making changes on your production site, we recommend the AtoM Vagrant box. See:

- [Vagrant documentation](#)
- [Vagrant slides](#)

See also:

- [Community-developed custom themes](#) (wiki page)
- [AtoM's code repository](#) (GitHub)
 - [About AtoM's code repository](#) (wiki page)
- [AtoM documentation](#)
 - [Upgrading documentation](#)
- [Installing and upgrading AtoM](#) (slides)

If you get stuck, why not make a post in the AtoM user forum? See:

- [AtoM user forum](#)
- [About the AtoM user forum](#) (wiki page)

Good luck!

