Drutopia Documentation

Rosemary Mann, Nedjo Rogers, Clayton Dewey, Benjamin Melanç

Sep 15, 2019

Getting started

Drutopia is an initiative within the Drupal project based in social justice values and focused on building collectively owned online tools. Current focuses include two Drupal distributions aimed at grassroots groups, ensuring that the latest technology is accessible to low-resourced communities.

The code is open source, and available on GitLab, as are our issues.

CHAPTER 1

Quickstart

There are two quick ways to get Drutopia up and running: PHP on your local computer or a full, production-like environment provided by a virtual machine.

In both cases we'll end with the Drutopia distribution codebase, built by Composer, in a directory called `my-drutopia-site` (please change this to whatever you would like in the instructions below!).

1.1 PHP

1.1.1 Prerequisites

1. PHP (5.5.9+, 7.1+ preferred; test in a terminal with `php --version`).
2. Composer.

   Note: If you have composer installed globally you can use `composer` instead of `./composer.phar` in the first command.

1.1.2 Instructions

Run these commands in a terminal

1. `./composer.phar create-project drutopia/drutopia_template:dev-master --keep-vcs --no-interaction my-drutopia-site`
2. `cd my-drutopia-site/web && php -S localhost:8008`
3. Open localhost:8008 and follow the installation instructions; leaving everything at defaults will be fine.

See the README that was downloaded in that step for more.

1.2 Virtual machine

1.2.1 Prerequisites

1. Install VirtualBox (check your app store or package manager)
2. Install Vagrant (check your app store or package manager)

1.2.2 Instructions

1. `git clone git@gitlab.com:drutopia/drutopia_vm.git my-drutopia-site`
2. `cd drutopia_vm`
3. `vagrant up` (this runs a site install via drush but unfortunately no Drutopia features are enabled).
4. Log in at my-drutopia-site.test/user with the credentials username: admin, password: admin
5. Enable Drutopia modules at my-drutopia-site.test/admin/modules

You can access the virtual machine (for use of Drush etc) with `vagrant ssh`.

CHAPTER 2

Getting Plugged In

We encourage members, supporters, and contributors to join the conversation through video calls and Mattermost conversations too. Always keep in mind the *Code of Conduct*.

2.1 General meetings

During the summer of 2019 we have general meetings every other Tuesday from 2 pm–2:30 pm Eastern Time (June 11, June 25, July 9, July 23, August 6, August 20, September 3).

Newcomers and anyone who feels they need to catch up with what's going on a bit are invited to join.

You can join us using Zoom:

https://zoom.us/j/695543337

Or by telephone:

> In US dial: +1 646 558 8656
> International numbers available:
> https://zoom.us/zoomconference?m=l7EUL8Ml2IXLmCDn-W-y0Rzd_wz8Dzq1
> Meeting ID: 695 543 337

Additional times to talk will be scheduled as needed, so if you have something to discuss, please be in touch!

2.2 Chat.Drutopia.org

We use the Mattermost messaging app to discuss ideas and get help from each other. Mattermost is a free and open source chat system that is similar to the proprietary Slack.

Chat through the web browser at https://chat.drutopia.org

Or install the Mattermost app for your device https://about.mattermost.com/download/

2.3 Issue queues

The many projects that make up the Drutopia initiative are in our GitLab group: http://gitlab.com/drutopia/

Post questions or feedback or requests to the queue of the relevant project; if you're not sure you can always use the main Drutopia distribution issue queue: https://gitlab.com/drutopia/drutopia/issues/

Try searching the issues first: https://gitlab.com/groups/drutopia/issues

2.4 E-mail

If you have any questions or trouble getting set up contact us at info@drutopia.org and we'd be happy to help.

CHAPTER 3

Manifesto

Drutopia is an initiative to revolutionize the way we build online tools. We're combining the principles of software freedom, community ownership and intersectional politics to co-develop technologies that meet our needs and reflect our values.

Tools such as Squarespace and NationBuilder have lowered the cost for launching a website. However, these tools are limited in functionality and push people getting into web development for a cause into the dead end of proprietary services.

We intend to develop a framework upon which an ecosystem of open source distributions will flourish- meeting the unique needs of grassroots nonprofits, unions, food co-ops, small businesses and others making big change on small budgets. We will also use this framework to build a member-owned platform spanning multiple participating hosts. Members of the platform will drive the vision of the project including, crucially, guiding the development of new features.

We are looking for designers, authors, community organizers, media strategists, artists, developers, open-source enthusiasts, information architects, project managers, activists, and anyone passionate about community technology to join us.

If you support this initiative, add your voice to the statement at drutopia.org

CHAPTER 4

Drutopia Points of Unity

Drutopia is an initiative to revolutionize the way we build online tools. We're combining the principles of software freedom, community ownership, and intersectional politics to co-develop technologies that meet our needs and reflect our values.

As participants in the Drutopia initiative we agree to:

- Be inclusive regarding gender, gender identity, sexual orientation, ethnicity, ability, age, religion, geography, and class.
- Commit to protection of personal information and privacy and freedom from surveillance.
- Value collaboration and cooperation above competition.
- Place human needs over private profit.
- Foster non-hierarchical structures and collective decision-making.

CHAPTER 5

Purpose and Goals

5.1 Mission

Drutopia combines the principles of software freedom, community ownership and intersectional politics to co-develop technologies that meet our needs and reflect our values.

5.2 Points of Unity

As participants in the Drutopia initiative we agree to:

- Be inclusive regarding gender, gender identity, sexual orientation, ethnicity, ability, age, religion, geography, and class.
- Commit to protection of personal information and privacy and freedom from surveillance.
- Value collaboration and cooperation above competition.
- Place human needs over private profit.
- Foster non-hierarchical structures and collective decision-making.

5.3 Goals

1. Release a Drupal distribution that meets crucial needs for networked grassroots organizations.
2. Launch fully functional drutopia.org
3. Establish a worker co-op product team
4. Build a sustainable and healthy number of members

5.4 drutopia.org goals

1. Use compelling content to convince grassroots activists to become members.
2. Provide technical information to plug open source community members into the project.
3. Gather donations by creating content that inspires, addresses critiques and demonstrates impact.

CHAPTER 6

Roadmap

There are four main components to the Drutopia project: the organization, the base distribution, the (first) platform cooperative, called Solidarity, and the Drutopia promotional website.

You can track our progress in more detail from our Milestones page.

6.1 Drutopia Organization

6.1.1 Incorporate as a Cooperative

Completed

- Adopt a Fiscal Sponsor
- Define membership

In Progress

- Decision making structure

Planned

- Cooperative Bylaws
- Incorporation location (ie: which country and state/province is best to incorporate into?)
- Cooperative application

6.2 Drutopia Base Distribution

Completed

- User Research
- Information Architecture

- Actions
- Articles
- Blog Posts
- Campaigns
- Customizable homepage
- Events
- Groups
- Profiles
- Users
- Starter theme

In Progress

- Publish content to social media platforms

Planned

- On-site donations
- Resource Library

6.3 Solidarity Platform Cooperative 1.0

Completed

- Research
- Information Architecture

Planned

- Articles
  - A contributor can relate an action to an article.
  - A contributor can relate a group to a an article.
  - A contributor can relate a campaign to an article.
- Blog Posts
  - A contributor can relate an action to a blog post.
  - A contributor can relate a group to a blog post.
  - A contributor can relate a campaign to a blog post.
- Campaigns
  - A contributor can relate an action to a campaign.
  - A contributor can relate a group to a campaign.

6.4 drutopia.org upgrade

Completed

- Research
- Information Architecture
- Survey
- White Paper
- Contact

In Progress

- Copywriting
- Customized homepage
- Initial Articles
- Membership form
- Donation form

CHAPTER 7

Using your Drutopia site

Drutopia End User Documentation

7.1 Login/logout

Log in via the login block in the footer region using your user name and password.

Note: If you do not have the user login block enabled, append `/user/login` to your website address (for example demo.drutopia.org/user/login) and you will get to the log in box.

If you ever forget your password, you can have a new one-time login link emailed to you. Use it to log in and then re-set your password immediately.

Note: When using this one-time log in do not do anything else on the site before setting your password or you will be prompted for your old password which you do not have.

To log out use the logout button in the login block (in the footer region on a standard Drutopia site). Alternately, go `/user/logout`.

7.2 Using the admin menu toolbar

- As a site manager (when you are logged with with manager permissions) you will see a toolbar at the top of the website. This is a handy way to navigate to the areas you may need to access for site configuration and other administrative tasks.
- Clicking on the top level (e.g., Content, Structure) will take you to the administration page for that section. So you can click on the Structure button and go to a page where you can select the appropriate tool within this section, for example block layout or taxonomy.
- The tool bar also has dropdown menu functionality, so you can also navigate directly by hovering over the main button and navigating directly to a subsection or even an item within that (for example to add a block by hovering over the Structure button, and then Blocks, leading to an Add custom block link).

- You can also navigate to the administrative sections of your website by entering the complete address, e.g., https://websitename/admin/structure/block.

7.3 Roles and user accounts

- A key reason for using a content management system such as Drupal is to spread out the work of adding content to your site. Ensure that all staff members or volunteers who are in a position to contribute to the site have an account set up for them and are assigned the appropriate role for the tasks they will be carrying out on the site.

7.3.1 Adding a new user account

- Start by logging in with a site manager account.
- Once logged in, with the manager role, you will see a toolbar at the top of the page.
- Select the "People" link.
- Start by clicking the button "Add user".
- Type in a user name as well as an email and a password. Note that any email address may only be used for one user account.
- You can opt to inform a user of their new role (they will be sent a one time login link).
- Save the new user.
- The next step is to assign this new user to a role. Roles determine what permissions a particular user has (that is what they are allowed to do on the site.)

7.4 User roles

The following roles are available on an Drutopia site:

- **Anonymous user**: A site visitor who has not logged in (Exists by default).
- **Authenticated user**: Anyone who has registered on the site, and authenticated by logging in. (Exists by default).
- **Contributor**: A contributor is a member of staff or a volunteer who is going to be posting content but not editing other people's content.
- **Editor**: The editor can edit other people's content as well as post and edit their own.
- **Manager**: The manager has permissions to add users and groups as well as some site configuration permissions.

```
.. note::
For a hosted version of Drutopia the highest permission level is site manager. For an␣
↪independently managed Drutopia site, you would also have access to an administrator␣
↪role which has all permissions on the site and hence has security implications.
```

On the main user administration page (`/admin/people`) you will see a list of users. Select the user you want to assign a role to by checking the box beside their user name. Then use the drop down selector under Action to choose the role you want to apply to the selected user. Then click the Apply to selected items button.

You can also remove roles from users this way as well as blocking and deleting users.

7.5 Getting going with your new site

There is very little initial configuration that is required with your new Drutopia site. The following configuration is recommended:

- You should start by clicking the Configuration button in the toolbar, and selecting the Site information section (or navigate directly to `/admin/config/system/site-information`.) Here you can configure the site name, slogan, and site email address.
- If you did not do so during the install process, ensure that you have a default country selected. Under configuration, choose Regional settings. If no default country is set, the calendars will often be off by one day. Even if you set this during install, it is advisable to visit this page and re-save your settings to ensure proper calendar functioning.
- Also on the Configuration page, select Date and time, where you can change the display of the time settings as you wish for either AM/PM for 24 hour clock. Even is you are satisfied with the time/date displays, click the save button to ensure that correct displays for the calendar view.
- Configure any desired social media follow links for your site. Follow the link to the Social Media block (admin/structure/block/manage/socialmedialinks) and enter the URL for any service you wish to appear. Save your changes.
- After installing your site you may be prompted to Rebuild your content access permissions. Click the Rebuild permissions link.
- Configure your contact form (see below).
- Customize or delete the sample content (see below)

7.6 Sample content

Drutopia includes sample content. This will make it even easier for users new to Drutopia and Drupal to get going with a new site. The sample content will give a visual snapshot of what your site will look like as it is filled with content, as well as providing pieces of content that can be edited to quickly get you started. You can use the sample content in two ways:

1. For the piece of content you want to use, select the edit tab and then replace the title, other fields, body text and image with your own, and save making it real content for your site. As noted above, the sample content is created using a filtered HTML text format. You can switch that to use a WYSIWYG (what you see is what you get) text editor by selecting the desired text format below the body field.
2. Or you can use them merely as examples, reviewing how they are formed based on their various fields and then un-publishing or deleting them and starting fresh.

7.7 Creating new content

7.7.1 Selecting the appropriate content type

The first step in creating a new piece of content for your site, is deciding what content type is most suitable.

A Drutopia website may include the following content types (depending on which features you have enabled):

- **Action**: An action is a specific, single action a user can take.
- **Article**: Use articles for time-sensitive content like news, press releases or blog posts.

- **Basic page**: Use basic pages for your static content, such as an 'About us' page.
- **Blog**: Use blog for personal or journal-like posts.
- **Campaign**: A campaign includes background information as well as ability to list demands and updates.
- **Event**: An event contains a date.
- **Landing page**: Landing pages can be used for custom pages such as the home page.
- **People**: Use people content type for people such as staff, volunteers, contributors.
- **Resource**: A resource can be either a file, such as a PDF, or a link, such as a website URL or an embedded video.

It is helpful to pay attention to the content type and use the same content type for similar posts to ensure consistency. For example, many posts will be displayed in different parts of the site based on the content type.

- Each content type may have slightly different fields to be filled in but there will be many similarities.

7.7.2 Step-by-step instructions for adding content

Using the top of page toolbar, select "Add content" from the dropdown menu "Content". You will see a list of content types to select from. For example if you choose "article" you will get a form to fill in with the following elements:

- **Title**: Give your post a title. This is a required field.
- **Article type**: You can choose to categorize your articles by type. Possible types include news (comes by default) and any other article type terms you choose to add. (See taxonomy section below.) Other content types also have a "type" field.
- **Authors**: Article and blog content type allow you to link the content to any author that has a "people" entry. This field uses an autocomplete, so you start by typing in the person's name. You will then see suggested matches. Click on the match to complete. Should you wish to add more than one author, click the "Add another item" button and repeat the autocomplete process.
- **Image**: Click the "Browse" button to begin to add an image. To upload a new image, first browse for the image and then select. Allowed image types are: png, gif, jpg and jpeg. The recommended image size is 1200 by 675 pixels. The images that accompany posts are used in numerous displays so it is highly recommended that you include an image wherever possible.
- **Summary**: The summary should be a one to two sentence summary of your post. This will appear in various displays on the site. This is a required field.
- **Body paragraph**: The body paragraph field is where the main content of your post will be placed. By default, a text paragraph will be open for you to begin typing in your content.
  - A text editor is installed to help with formating. Different role types have access to different levels of input options for formatting. You only have access to the level of formatting options to which your assigned role grants you access for security reasons.
  - Ideally text should not be pasted directly in from Word as it tends to create problems. If you have already word processed your content, copy it from your document and paste it into a text editor to clear formatting before pasting in.
- You can also add image or file paragraphs to add additional images or to upload file documents, such as PDFs. The article and basic page content types also have a video paragraph to let you add an embedded video (that is a video hosted elsewhere such as YouTube or Vimeo).
- You can use the "drag and drop" arrow tool to rearrange your paragraph order.
- **Topics**: Topics allow you to categorize your content across a number of specified content types.

- **Tags**: Tags allow content to be sorted by terms that describe the content. Using tags on posts will provide website visitors alternate ways to locate content. Enter tags as a comma-separated list.

There are a number of other collapsed items which can all be opened to reveal further input options if you have the required level of permission. Most can just be left at their defaults for the particular content type which have been pre-configured for the most likely setting. But if you want to change any of those settings you may do so. The most likely ones you may need to change would be:

- **Revision log message**: In the future, when editing a post you can log the changes made.
- **Menu settings**: For most posts, you will not need to do anything under menu settings, but if for example you want an item to appear in the menu system, you can check the "provide menu link" box and configure the menu links. (More on menus below.)
- **Comment settings**: Can be changed from open to closed.
- **URL path settings**: Post are automatically giver a URL alias which is a user friendly path. If you would like to manually provide a path, you can uncheck the "Generate automatic URL alias" box and enter your own path.
- **Authoring information**: If you need to change the author or date, do so here.
- **Publishing options**:
  - Promoted to front page (this promotes a piece of content to the home page, is the default for the article content type.)
  - Sticky at top of lists (makes the piece of content "stick" as the top piece of content. Only one piece can be so marked at the time. You will need to "unsticky" a piece of content before you "sticky" another one.

At the end of the form you may click the "Preview" button to see how it will look or just go ahead and click the "Save and publish" button. There is also a "Save as unpublished" drop down option on the Save button for work that you want to save but needs review before publishing.

7.8 Specialized paragraphs

There are some specialized paragraphs that allow content editors to do more sophisticated work.

7.8.1 Storyline

A storyline is a simple way to show a chronology or tell a story in a graphical format. A storyline can include specific dates but unlike some timelines does not require this, allowing you to organize in any manner you like.

- To use the storyline feature, you'll need to enable both Drutopia Storyline as well as Drutopia Page Storyline.
- Once these two features are enabled, the basic page content type will have an additional Storyline field.
- The initial body paragraph field provides a place to add opening text that will display at the top of your storyline.
- The storyline header paragraph contains a single field for a header, such as a year. This is open by default to start you off.
- Next, add a storyline item paragraph by clicking that button. This paragraph will have two fields, a heading where you can add a more specific date and a text field where you add the text you want to appear in your storyline.
- Continue to add storyline items which fit under one header, adding new headers as desired.
- You can rearrange using the drag and drop tool as needed and then save your page.

7.8.2 FAQ

The FAQ paragraph type is available on both page and article content types.

- You can create a whole FAQ page by adding a series of FAQ paragraphs which each contain a question and answer (or title and information).
- Alternately, you can insert an FAQ paragraph into an article or page as a way of providing more information on a subject but that can be opened or closed by a site visitor.

7.9 Editing content

It is very easy to go back into a piece of content you have created and make any changes required.

- When you are logged in, and if you have appropriate permissions, when viewing a piece of content you will see an edit tab on the top of the piece of content. By clicking that tab you will return to the form on which you created the content.
- You can also access content for editing via the content administration page described below.
- Simply make any changes as you did when you were first creating the piece of content and save those changes.

7.10 Content administration

To view a list of all content that exists click the "Content" button on the toolbar. This is a valuable location for site managers in managing the content on the site, with tabs for Content and Comments.

- Under the content tab, you can filter by content type, allowing you to search for all events for example.
- You can also filter by status: published or promoted.
- To update any piece of content, you can check the box next to it. Then, select the update option you want (for example, *Promote content to front page*) and click the *Apply to selected items* button.
- Click the edit link to go to a particular piece of content directly in edit mode.
- Using the Comments tab you can approve or delete comments posted on your site.

7.11 Home page setup

The home page consists of a number of elements. It can include:

- A home page hero image with accompanying text about your organization and a link to your About page.
- The latest four articles that have been promoted to the home page.
- The latest four actions that have been promoted to the home page.
- A home page featured block with an image, text and link to any page on your site.
- The latest four resources that have been promoted to the home page.
- The latest four blog posts that have been promoted to the home page.
- The latest four campaigns that have been promoted to the home page.
- Three custom blocks that appear at the bottom of the page and can link to any internal or external page.

- If your site does not have a particular feature enabled or there is no content of that type, the block will not appear.

7.12 Editing the home page blocks

There are five custom blocks that appear on the home page (two wide slides, and three smaller ones), each containing an image, some text and a link.

- The easiest way to edit is to use the edit tool (with a pencil icon) that appears when you hover over the top right-hand corner. Click the edit button and then select Edit.
- Start by typing in the page on the site you want your home page block to lead to. For the home page hero block (at the top) this will likely be an "About" page. For the other home page block, it can be to any page that you are featuring.
- You can also edit the Link text (which by default will read "Learn more."
- Remove the default image and upload your own image. Ideally the images for the homepage block should be 1500px by 750px. You will need to take care in selecting the images for these blocks.
- Edit the text that will appear over the image. This should be one or two sentences.
- Save your changes.
- Visit the home page to see how your new block looks. It is very likely that you will need to make further changes to ensure the text is visible with the image you have selected and not too long.
- The three bottom region blocks can be edited in a similar way, but for these it is the link text that will appear as in a title and the text should be extremely brief.

7.13 Menu

By default certain menu items will appear in the main menu of a Drutopia site. These will vary depending on which features you have enabled.

- The main menu consists of the tabs on the top of the page, which may include: Actions, Articles, Blog, Campaigns, Events, Groups, Resources.
- To change menu links, go to menu section (under Structure). You can edit existing links or add new ones. You can also rearrange the order of menu items by dragging them. Ensure you save your changes.
- There is a footer menu which includes links to About, Contact and People. You may also choose to add new links to pages such as legal information or a privacy policy.
- A menu always needs a path which can be found for any piece of content or section in the address bar when you are viewing that piece of content (just what comes after the site name).

7.14 Themes

Drutopia ships with a default theme (which determines how your Drupal site looks) called Octavia (in honor of Octavia Butler). Octavia has been built as a subtheme of Bulma using the Bulma framework.

- Click on the settings page for Octavia (admin/appearance/settings/octavia) to adjust any settings (note that editing the Global settings will not work, you must edit the Octavia settings).
- You can upload your own logo and/or favicon.

- As always, ensure you save your configuration.

7.15 Taxonomy

The taxonomy system is a way of categorizing content. A vocabulary is created and within that vocabulary terms are added.

- The topics vocabulary is available for a sub-set of Drutopia content types. Topics are used to categorize content across multiple content types and terms should be well-thought out to capture the key issues or topics that will help your site visitors explore content.
- The tags vocabulary is created and available for a variety of Drutopia content types. Terms can be added to this vocabulary "on the fly" as you are creating pieces of content.
- Some content types have a "type" field, such as Article type, which allow you to categorize by a defined set of terms. So for example, for event, you may want to categorize as meeting, fundraiser, workshop etc.
- To add terms to a vocabulary, select taxonomy from under the structure button in the toolbar (admin/structure/taxonomy). Choose the vocabulary you want to work with and select the add terms button. Add a new term and save. Or edit the name of an existing term, should you wish it to be slightly altered.

7.16 Related content

You can enable the Related Content feature to have related content show on the bottom of a node page. Content is related based on terms that both pieces of content share in common.

7.17 Contact form

The Contact form is enabled by default. You can personalize the email recipient and a return message at `admin/structure/contact/manage/feedback`

7.18 Drutopia Groups

Groups allows you to create sub-sections on your site for groups that can be defined in a number of ways. Groups could be chapters or branches of your organization. Or they could be used to group users together who might have shared interests.

- Only basic group functionality has been created for Drutopia as further customization will be provided by the Solidarity distribution which is being built off of Drutopia.
- However, if you're starting with Drutopia a site administrator can add some of the more complex functionality with a small amount of configuration, such as linking groups to particular content types.
- To work with groups, navigate to the groups admin page via the toolbar. Here you can edit an existing group or add a new group. As well as similar fields that other content types have, groups have fields for contact information such as a phone number, email and address.

7.19 Search

Site search includes all content types except for Landing Pages. If you want a page to end up in site search, use the Basic Page content type instead.

7.20 Leaving feedback

Feedback on the software and platform is welcomed. When leaving feedback, keep the following in mind -

- Be direct, yet kind with feedback.
- For bugs, state what is not working. See Bug Reports/Testing.
- For features, form this as a user story. See Feature Requests.
- For how questions, form these as a question.

If you are a platform member, submit feedback (bugs, features, how tos) in your GitLab project. Otherwise, feedback should be left in the general Drutopia project.

CHAPTER 8

Customizing & Styling Drutopia

Sitebuilder and Themer Documentation

8.1 Introduction and clarification

Most day-to-day usage of Drutopia, including for site managers, is amply covered in [Drutopia's end-user documentation, "Using your Drutopia site").

This documentation is only for people who want to customize advanced configuration or create their own styles, or themes in Drupal parlance.

CHAPTER 9

Theming Drutopia

Drutopia supports being themed with all the flexibility and power of any Drupal site. Consult the Drupal 8 theming documentation for more information.

The Drutopia distribution itself comes with the Octavia theme, a sub-theme of Bulma. Because Octavia includes templates corresponding to Drutopia's features, and because the Bulma CSS framework it relies on is pretty great, we recommend building your Drutopia theme as a sub-theme of Octavia.

!!! note

```
Why is the Bulma CSS framework pretty great?  The lightweight, configurable approach␣
↪to styling web pages is based on [Flexbox [title="CSS Flexible Box Layout]](https://
↪developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout) and fully␣
↪responsive to varying screen sizes.  It's relatively easy to learn with a simple␣
↪syntax requiring minimal HTML, but also highly customizable through the␣
↪(syntatically) awesome power of [Sass [title="Syntactically Awesome␣
↪Styleshets]](https://sass-lang.com/).
```

Still, Bulma is only one of many possible approaches. We would love to see your preferred framework (Bootstrap, Zurb Foundation, etc.) adapted to Drutopia, and would especially be eager to help you create a Drutopia-ready base theme to give people a head start in styling their Drutopia sites with other CSS frameworks. (As Octavia serves as a base theme for Drutopia but is itself a sub-theme Bulma, a logical starting point for a new base theme for Drutopia use would be creating a sub-theme of Bootstrap or Zurb Foundation.) Please *get in touch*!

If you plan to style Drutopia with Bulma by creating a sub-theme of Octavia, please read the documentation as there's plenty of quick hints that will save you lots of time.

CHAPTER 10

Making a subtheme of Octavia (using Bulma to style your Drutopia site)

In this tutorial, we'll create a subtheme of the Octavia theme and use it to produce a customized version of Bulma.

Recommended reading: the Customize section of the Bulma documentation. We'll be using `node-sass`, so if you like you can skip the other two options covered there (Sass CLI and webpack).

In this example, we'll create a new Drupal theme called `calla`.

10.1 Caveats

- Subtheming, while supported and encouraged in Drupal for a long time, still has edge cases that can cause problems. For example, Drutopia is applying a patch to fix Display Suite templates. The only known pitfall is that overriding parent theme library assets is far more brittle than it ought to be. This guide will walk through that— but of course it's the unknown pitfalls that we have to worry about.

10.2 Documentation for setting up a local development environment

10.2.1 Get the code

If you have a Drutopia project working already, clone your project's repository. If not, start fresh by getting Drutopia running locally using the *method of your choice*. For this documentation we will presume you are using a DrutopiaVM-based project:

- Clone the Drutopia VM repository.
- Follow the instructions in the README.md file in the root of the repository to get the VM up and running.

10.2.2 Base install

- When you first open up the `http://drutopia-vm.local` site, install Drutopia.

10.3 Set up for theme development

- Enable Twig debugging. See the tips here. You'll need to change the permissions on the `sites/default` folder to allow creating the `services.php` file.

10.4 Create a new theme based on Octavia

10.4.1 Create a new directory

Create a directory with the name of your theme, `calla`. Typically you create a `themes/custom` directory and create your new directory there.

10.4.2 Create a `*.info.yml` file

The info file identifies your theme. For details, see the drupal.org documentation page Defining a theme with an .info.yml file.

In the theme's directory, create a new file called `themename.info.yml`. In our case, that's `calla.info.yml`. It needs three sections:

- Basic information such as the name and description.
- `regions`, which are the areas of the page that blocks can be put into. See drupal.org documentation. At the most basic, the regions should be the same as used in the parent theme, Octavia.
- `libraries`, which allows us to substitute the parent theme's `.css` file with our customized version.

Here's an example. We'll cover the file mentioned in the `libraries` section (`dist/css/global.css`) later in the tutorial.

```
name: Calla
description: Customization of Drutopia's Bulma-based Octavia theme.
type: theme
base theme: octavia
core: 8.x

regions:
  header: 'Header'
  navbar_branding: 'Branding'
  header_search: 'Search'
  header_tabs: 'Tabs'
  primary_menu: 'Primary menu'
  secondary_menu: 'Secondary menu'
  highlighted: 'Highlighted'
  help: 'Help'
  content: 'Content'
  sidebar_first: 'Sidebar'
  tile_one: 'Tile 1'
  tile_two: 'Tile 2'
  tile_three: 'Tile 3'
  tile_four: 'Tile 4'
  tile_five: 'Tile 5'
  bottom: 'Bottom'
  footer: 'Footer'
```

(continues on next page)

(continued from previous page)

```
libraries-override:
  bulma/global:
    css:
      base:
        /themes/contrib/octavia/dist/css/bulma.css: css/mystyles.css
```

10.4.3 Set up theme development

We'll be writing files in SCSS, which needs to be compiled into CSS. In order to do this, we need some software.

The first step is to install node.js.

10.4.4 Set up and use `node-sass`

Once you have `node.js` installed, follow the documentation on customizing Bulma with `node-sass`, with some minor variations:

In step 2, we need to install an additional package. This is because Octavia uses a Bulma extension to style timelines.

```
npm install bulma-extensions --save-dev
```

In step 5, while you're editing `package.json` file to add the needed scripts, you can make some other changes like adding an author and setting the license. Here's an example of how it might look after edits:

```
{
  "name": "calla",
  "version": "1.0.0",
  "description": "A subtheme of Octavia for use with the Drutopia Drupal distribution
↪",
  "main": "sass/mystyles.scss",
  "scripts": {
    "css-build": "node-sass --omit-source-map-url sass/mystyles.scss css/mystyles.css
↪",
    "css-watch": "npm run css-build -- --watch",
    "start": "npm run css-watch"
  },
  "author": "Some One",
  "license": "GPL",
  "devDependencies": {
    "bulma": "^0.7.2",
    "bulma-extensions": "^3.0.0",
    "node-sass": "^4.9.4",
  }
}
```

In step 6, the example `mystyles.scss` file given in step 6 uses only specific Bulma elements, but we need the full deal. We also need to import the Bulma timeline. So delete the lines from here on down:

```
// Import only what you need from Bulma
```

Replace them with this:

```
// Add bulma and the timeline extension.
@import "../node_modules/bulma/bulma.sass";
@import '../node_modules/bulma-extensions/bulma-timeline/dist/css/bulma-timeline.sass
↪';
```

When you have completed these steps, you should end up with a customized version of Bulma written to `css/mystyles.css`.

10.4.5 Create config files

- Copy the `config` folder from Octavia.
- In all subdirectories of the copied `config` folder:
  - Rename file names to use the new theme name rather than `octavia`. In our example, we replace `octavia` with `calla` in all file names.
  - Within those files, replace any reference to `octavia` with your new theme name.

10.4.6 Install your theme

- Your theme should now be ready to install.

CHAPTER 11

Decision Making

Drutopia is a member-owned cooperative. All members have equal voting power to elect the Leadership Team, the ability to become a project maintainer, set the strategic goals of Drutopia and inform the technical roadmap.

11.1 Membership

To become a member of Drutopia, a person must do to the following -

- Pay an annual membership due of $600 (or $50 a month)
- Agree to the Drutopia Code of Conduct

11.2 Leadership Team

The Leadership Team can make strategic decisions regarding Drutopia. This includes vision, mission, goals, outreach strategy and target audiences, licensing. Decisions are made by consensus.

11.2.1 Joining the Leadership Team

The current leadership team is:

- Ben
- Clayton
- Leslie
- Nedjo
- Robert
- Rosemary

We are currently looking for a representative for front-end development. Open positions will be filled by consensus from the current leadership team.

Once the project has been established for a longer time a democratic process driven by Drutopia members at large will be implemented.

11.2.2 Leadership Team Elections

Elections of the Leadership Team will happen annually, once membership is open to the public.

11.3 Working Groups

Working Groups make tactical decisions. Each working group has a working group lead. This lead also has the project maintainer role on the project. These are informed by the strategic decisions made by the leadership team. This includes approving pull requests, creating feature requests, determining technical and architectural approaches to the project.

All commits are made by creating a pull request. Any maintainer can approve a pull request. Any Drutopia member who has had a commit accepted to the project can become a maintainer if they'd like. If at any time a maintainer violates the Drutopia Code of Conduct their maintainer privileges will be suspended until maintainers and the leadership team resolve the issue.

The current working groups (and working group leads) are-

- Design
- Front-end development @nedjo
- Back-end development @nedjo
- Site Building @rosemarymann
- User Experience @cedewey
- Outreach/Marketing @mlncn

See working groups for more detail.

CHAPTER 12

Working Groups

12.1 User experience working group #UX

Team lead: Clayton Dewey clayton@drutopia.orgThe user experience working group is focusing at the moment on laying the foundation for a fantastic user experience for both contributors and supporters of Drutopia as well as the end user of the Drutopia tools we are building.

To do this we are doing the following -**Contributor UX**

1. Expanding on the current README.MD
2. Writing a CONTRIBUTING.MD
3. Defining a Code of Conduct

End User UX

1. Conducting user research through surveys and interviews
2. Adding the survey to drutopia.org

I am looking for both help in the following ways

1. Joining the working group in a regular or semi-regular way
2. Completing the survey (for now answer the questions and email them to me)
3. Being interviewed (email me letting me know and we can set up a time!)
4. Sending me contact info for folks you think would be down to fill out the survey or be interviewed
5. Share the survey directly with folks (again have them email me their answers for now)
6. Interview people directly (email me and we can walkthrough what that looks like, which shouldn't take long to do)
7. Adding the survey as a form on Drutopia (if you can do this, let me know and I'll add you as a contributor to the project)

12.2 Project management working group

Team lead: Leslie Glynn leslie@drutopia.orgGoal: to help the Drutopia project through

- tracking the objectives/issues of each of the working groups
- documenting and tracking the project schedule and milestones
- assisting the leadership team through reporting on project status

12.3 Users working group

Team lead: Leslie Glynn leslie@drutopia.orgGoal: to help the Drutopia project through

- communicating with potential users of Drutopia to identify features of interest
- finding commonality of desired features among the various users
- assisting the leadership team in defining features and priorities in the Drutopia roadmap

12.4 Documentation working group

Team lead: Rosemary Mann rosemary@drutopia.orgWe'll be tackling all areas of documentation including end user documentation for non-technical users. While some of these tasks will not happen until we're further along in the development cycle, we want to ensure that solid documentation goes hand in hand with development.

Documentation issues

12.5 Outreach/marketing working group

Team lead: Ben Melançon ben@drutopia.orgOutreach/marketing seeks to ensure that every web designer, styler, developer, and site builder knows of Drutopia's goals and approaches, and knows how to participate in Drutopia if they are interested. We also work to communicate the potential of Drutopia to organizations who need websites, and to distill the membership offering so they can make the decision to join.

12.6 Development working group

Team lead: Nedjo Rogers nedjo@drutopia.org

The development working provides technical guidance and support to the Drutopia initiative.

Development issues

12.6.1 Technical specifications

- Produce and maintain technical specifications of Drutopia-compatible features and distributions.
- Evaluate and make recommendations on specific solutions to be used in Drutopia. Example: will there be a standard base theme?

12.6.2 Code development

- Contribute to and maintain Drutopia extensions. See the list of extensions used in Drutopia.

12.6.3 Current priorities

Looking to get involved? Here are some current priorities and gaps.

- Review of the draft technical guide.
- Open issues in the Configuration Sync module.
- Lead development for planned Configuration Merge and Block Theme Sync modules.

12.7 Co-op working group

Interim team lead: Rosemary Mann rosemary@drutopia.orgOne of the core aims of Drutopia is to create a different kind of economic model. To this end we're are envisioning the creation of a platform co-op where end user groups are members. We have also explored the idea of a multi-stakeholder co-op so that worker members are also included. This working group will tackle the research, business planning, and organizational work to create a new co-op.

12.8 Features working group

Interim team lead: Nedjo Rogers nedjo@drutopia.orgAt its most basic, a distribution is a set of features that will work seamlessly together but that can also be used separately or, ideally, in other distributions. Building out those features is much like site building but with the extra complexity of each being a generic solution and with challenges of interoperability and consistency. The Features working group will take on the concrete work of developing and fine-tuning the features that will become Drutopia.

CHAPTER 13

Hosted (SaaS) Drutopia Standards

To be recommended by Drutopia as a host, these qualifications—at minimum—must be met:

- All people or organizations hosted on the platform are members of the Drutopia cooperative, and the membership fee is passed on to the cooperative.
- The resources dedicated to each site meet the Drupal 8 system requirements.
- The host must support SSL (https://)
- Drupal core must be supported with security updates
- All approved Drutopia contributed modules must be supported with security updates (supporting additional is allowed of course)
- All approved Drutopia configuration sets must be available for use (supporting additional is allowed of course)
- All approved Drutopia install profiles must be available on new site launch (supporting additional is allowed of course)

CHAPTER 14

Platform Cooperatives

Drutopia is built to support the proliferation of platform cooperatives, online platforms owned collectively by its members.

The first of these platform cooperatives is designed by grassroots organizers to help one another act, mobilize and support one another.

If you are interested in joining an existing platform coop or starting your own using the Drutopia architecture, contact the Drutopia leadership team at info@drutopia.org

14.1 Solidarity Libre Software as a Service

A hosted option is available for $50/month which provides the following:

- Platform Cooperative membership
- Security updates
- Email aliases
- NextCloud (and other MayFirst services)
- Hosting Support
- Training & Documentation
- Mattermost Support Channel
- One hour “ask anything” each month

CHAPTER 15

Drutopia Software Cooperative

The Drutopia project is governed cooperatively, with a leadership team providing strategic guidance, technical leads deciding technical decisions and members driving forward the roadmap.

15.1 Membership

Membership is open to all, simply -

- Pay an annual membership due of $10-100 ($50 recommended) or one time lifetime membership of $100
- Agree to the Drutopia Code of Conduct >drutopia-code-of-conduct.html> and Points of Unity

Members are encouraged to request and vote up/down features, report bugs and contribute to the project.

15.2 Leadership Team

The Leadership Team makes strategic decisions regarding Drutopia. This includes vision, mission, goals, outreach strategy and target audiences, and licensing. Decisions are made by consensus.

Joining the Leadership Team

The current leadership team is:

- Ben
- Clayton
- Leslie
- Nedjo
- Rosemary

We are currently looking for a representative for front-end development. Open positions will be filled by consensus from the current leadership team.

Once the project has been established for a longer time a democratic process driven by Drutopia members at large will be implemented.

15.2.1 Leadership Team Elections

Elections of the Leadership Team will happen annually, with our first election to be held November 2018. Each member gets on vote.

15.3 Technical Leads

Technical leads make the final call on technical decisions.

Current technical leads are:

- Ben
- Clayton
- Nedjo
- Rosemary

Any member is welcome to request becoming a technical lead by

- asking a current tech lead to become one
- a current tech lead then nominates them
- One other tech lead must approve the person
- Once approved, that person's gitlab account is elevated to the Owner role

CHAPTER 16

Drutopia Partners

A great way to serve grassroots organizations is by becoming a Drutopia Partner. Partnership is available to both agencies and freelancers.

As a Drutopia Partner you and your client(s) receives

- membership to the Drutopia Software Cooperative
- a local, test and live environment with continuous integration for each client site
- membership to MayFirst/PeopleLink, which includes the following for each client site: * Web hosting * Email * Email lists * NextCloud accounts * Discourse accounts
- same day Drupal and Drutopia updates applied by the Drutopia team
- a Git repository with full control over your client's theme and certain configuration settings
- Support forum to ask questions and report bugs.
- Public recognition on drutopia.org

To become a Drutopia Partner,

- read and agree to the Points of Unity and Code of Conduct
- sign up each of your Platform Cooperative clients up
- commit to supporting the client's theme and configuration
- pay Drutopia Platform Cooperative membership dues on time
- contribute improvements back to the Drutopia project when possible
- serve as a helpful intermediary between your clients and the Drutopia project

CHAPTER 17

Contributing

Hello and welcome! There's always a lot happening, but we always have time to help orient folks getting involved for the first time. The best way to get involved is to *join the conversation through video calls and text chat*.

It's also helpful to know how we work together. The *Code of Conduct* outlines how we center collaboration and respect with one another.

Some types of contributions we're set up to welcome!:

17.1 Bug Reports/Testing

When something is broken—preventing normal/typical use of Drutopia—please write a bug report.

17.1.1 How to report a bug

17.2 Development

We welcome coders of all backgrounds and skillsets! Drutopia is a great place to build and learn.

To get started,

1. Review the *Drutopia technical guide*
2. *Join the conversation*
3. *Follow the development documentation*

As soon as you are working on changes or additions to code, configuration, design, or documentation, create a new branch to commit your work to.

Merge requests are the primary mechanism we use to change Drutopia. This allows you tell others about changes that you've pushed to your branch.

Once a merge request is sent, the rest of the team can review the set of changes, discuss potential modifications, and even push follow-up commits if necessary. GitLab will correctly make merge requests against the main branch by default; for Drupal projects this is usually 8.x-1.x.

17.3 Design

Much design, in particular user interface (UI) and user experience (UX) design are embodied in the fixes and features worked on in *development* and we welcome design, usability, and accessibility review at all points in the development process; please get involved!

Ultimately we plan to have a vibrant selection of designs for people building sites on Drutopia to choose from. Drutopia designs will be implemented as Drupal themes, using Twig templates, but these targeted use cases of Drutopia profiles will make it easier to produce a great-looking and great-working theme than the extremely expansive use cases a generic theme must support.

We also have need of designs to mock up better ways visitors and content editors can interact with existing features, and to prototype new features.

Use any of our *communication channels* to introduce yourself and a project member will work with you on getting started.

17.4 Documentation

We document the project through a documentation repository on GitLab. Every page, like this one, has an "Edit on GitLab" link at the top; however, we strongly recommend you clone the repository and edit locally so you don't lose your edits in GitLab's unstable forms. The format is either Markdown or Sphinx-enhanced ReStructuredText; the latter is recommended.

We try to keep in mind these tips for beginner-friendly documentation. It's a high bar and there's lot's of work needed to get there.

We keep track of documentation needs by posting issues to the project repository. See *the list of outstanding documentation issues <https://gitlab.com/drutopia/documentation/issues?label_name%5B%5D=documentation>_* for places you can help!

17.5 Feature requests

If you've got a great idea, we want to hear about it! Before making a suggestion, here are a few handy tips on what to consider:

- *Search to see if the feature has already been requested <https://gitlab.com/drutopia/drutopia-distribution/issues?label_name%5B%5D=suggestion>*.
- TODO write a section like this? Check out What makes it into Drutopia core? - this explains the guidelines for what fits into the scope and aims of the project
- Remember, your suggestion can't get adopted if people don't understand it. Please provide as much detail and context as possible. Explain the use case and why it is likely to be common. The strongest vote in favor of any feature request is clear demand among potential users/members of Drutopia.
- When making the request, tag it with "suggestion". TODO create issue template for this.

17.6 Project management

Do you have skills as a project manager or have interest in helping to plan, track and bring Drutopia features to completion? Here are some of the things you can do to become involved in the project management aspect of the project:

- Sign up as a supporter and find out more about the project
- Join our core project management team. Contact leslie@drutopia.org for more info on this.
- We can use help: * Creating user stories for a Drutopia initiative of interest to you * Breaking out the Drutopia initiatives into a set of tasks that can be worked on by the initiative team * Checking in on the progress of these tasks to ensure that overall Drutopia project schedules are being met * Reporting the status of an initiative to the initiative team

17.7 Outreach

Drutopia's public presence includes our web site, our projects on Gitlab and Drupal.org, social media, and (gasp) in-person gatherings at conferences and meetups. Here are just some of the ways you can help Drutopia:

- Sign up as a supporter
- Retweet and share Drutopia content
- Join our social media team to tweet and post to Facebook on behalf of Drutopia
- Respond to questions and comments on social media
- Plan and/or host hack-a-thons, trainings and presentations
- Blog about Drutopia!

17.8 Research

A major focus of Drutopia is building well designed tools that meet users' needs. Join the collaborative research effort to ensure that grassroots organization's voices are heard.

- Interview grassroots organizations (we have questions to get your started)
- Share our survey at drutopia.org/survey with people who would benefit from Drutopia
- Join our core research team to analyze our research data and make suggestions for the roadmap and enhancements to the platform. Contact clayton@drutopia.org for more info on this.

17.8.1 Reporting bugs

1. Make sure the bug isn't already resolved. Search for similar issues in the bug category 1. Make sure you can reproduce your problem on a clean installation of Drutopia.

- If you're also testing on your own Drutopia development instance, be sure to use the Master release branch, aka the Tests Passed channel.
- If possible, submit a Pull Request with a failing test, or;
- if you'd rather take matters into your own hands, try fix the bug yourself.

1. Make a report of everything you know about the bug so far by opening an issue in GitLab about it.

When the bug is fixed, you can usually expect to see an update posted on the reporting topic.

17.8.2 Developing for Drutopia

This document covers writing code for and contributing configuration to Drutopia features and the Drutopia distribution.

Drutopia so far consists of four bodies of code:

- Drupal 8 core.
- Drutopia features (Drutopia Article, etc.).
- Drutopia install profile (drutopia).
- Contributed Drupal modules and themes:
- Features
- Configuration Update Manager

Repositories

Drutopia features and install profile are hosted on GitLab and mirrored on Drupal.org.

For introductions on using GitLab, see:

- GitLab issues documentation.
- GitLab merge requests documentation.
- The video (on YouTube) Getting started with Git and GitLab.

Issue queues

To post suggestions or bug reports, use the issue queues for the GitLab projects linked above. See the GitLab documentation on creating an issue.

Geting set up with Git and GitLab

To get started contributing code-level changes to Drutopia features or the Drutopia distribution, see the GitLab basics documentation page for tips.

You will need to:

- Install Git on your computer.
- Create an account on GitLab and add your SSH key.

Local development environment setup

There are many possible ways to set up a local development environment. These instructions will assume a basic LAMP (Linux, Apache, MySQL/MariaDB, PHP) setup. For more information, see the drupal.org documentation on setting up a local development environment. See also the drupal.org page on where to look for support.

Useful utilities

Two utilities to help with Drupal site development are Drush and Drupal Console. See the Drush and Drupal Console documentation for installation instructions and usage tips.

Download and install Composer

Composer is a dependency manager for PHP. It manages core dependencies for Drupal Core and we use it for the same purpose with Drutopia.

Install composer

Install Drutopia

Run `composer create-project drutopia/drutopia_dev_template:dev-master --keep-vcs --no-interaction drutopia-dev-site`

- Visit your site and follow the Drupal installation instructions.
- Log into your new site and install the modules needed for development. These include:
- All Drutopia feature modules (which are listed on the modules page under the heading "Drutopia").
- Features UI and its dependencies (Features and Configuration Manager Base).
- See the README for more. One hint: `git pull` to get the latest version of the composer template we all use for development!

Work on an issue

We use the Features module for managing changes to configuration in Drutopia features and the distribution. See the documentation on Features for Drupal 8 for background.

Improvements or fixes should have an accompanying issue in one of the GitLab project issue queues.

- Before you begin, ensure you have committed any changes made for a previous issue, since you'll be reverting your site and otherwise may lose those changes.
- If no GitLab issue exists, create one. If one does exist, make sure it's assigned to you.
- For each project in your local dev site, first make sure you're working with the latest code.
- The install profile lives at `profiles/contrib/drutopia` and each of the feature modules at `modules/contrib/`:
- Check out the development branch.

```
git checkout 8.x-1.x
```

- Pull in any changes from the origin:

```
git pull --rebase origin 8.x-1.x
```

- Ensure your site reflects the latest code base. You may do this by reinstalling, or by using Drush to restore the site features to their default state. **Warning: you will lose any uncommitted changes you've made.** From the site's directory, type:

- `drush features-import-all` or, if you are sure you don't want to review the changes you're losing, `drush features-import-all -y`
- For each project you're going to be working on, create a new branch to work on the issue. A branch like this is often called a *feature branch*. Here is an example for an issue on the Drutopia Site project to enhance the default admin toolbar, where 10 is the issue number.

```
git checkout -b 10-admin_toolbar
```

Make and export the changes

For each feature you're working on, make sure you're working with the latest code.

- `git pull --rebase origin 8.x-1.x`

On the local development site you installed, make the changes required to resolve the issue. This might involve downloading and installing new modules and/or making configuration changes.

Now navigate to the Features admin page and regenerate the feature or features you've worked on. See the Features module documentation on generating features. Regenerate the feature. You will need to delete the previously exported version of the feature and replace that with the new export–making sure you don't delete the Git information in a `.git` directory.

Creating a new feature

We encourage proactive contributions to Drutopia, though it could be helpful to check in with the team about your feature idea.

We determine if a feature is included in Drutopia Base, or in a more specific distribution based on whether the majority of Drutopia user would benefit from the feature. The final call on this can be made by any technical lead.

If you are creating a new feature:

- Create a new project for the feature on GitLab in the drutopia group.
- Clone the project locally.
- On the site where you've done your development, navigate to the Features admin page and generate the feature to package up the configuration you've worked on. Start by editing the configuration for the features bundle assignment plugins.
- Generate the feature and copy its files to the directory where you cloned the repository. For example, if the repository is in a directory called `drutopia_example`, you will end up with a file `drutopia_example/drutopia_example.info.yml`.
- Create a `composer.json` file for the feature. If your feature depends on other Drupal modules, add those dependencies to the composer file's `require` section. You can use an existing Drutopia feature's composer file as a model.
- Add and commit the files, making sure you're creating a new `8.x-1.x` branch before committing:
  - $ `cd /path/to/drutopia_example`
  - $ `git checkout -b 8.x-1.x`
  - $ `git add .`
  - $ `git commit -m "Initial commit of Drutopia Example feature.`
  - $ `git push origin 8.x-1.x`

Registering your new feature

1. Create a new project on drupal.org:
    - uncheck the option to include an issue queue
    - for project home page, enter the GitLab project URL
    - assign the user 'drutopia_gitlab' the permission "Write to VCS"
    - assign the users mlncn and nedjo (optionally rosemarymann and cedewey) full rights to the project
    - push the 8.x-1.x branch
    - create a dev release for the 8.x-1.x release
    - edit the project to set 8.x-1.x as the default branch
    - administer releases to set 1 as the recommended major version
2. Edit drutopia_projects.txt file in https://gitlab.com/drutopia/hackysync to add the new project.
3. Create a new project on GitHub (mlncn)
4. Edit the `composer.json` file in the Drutopia install profile to include the new feature.
5. Edit the `composer.json` file in the Drutopia SDK to include the new feature.
6. Edit the `composer.json` file in Drutopia VM to include the new feature.
7. Edit the file `drutopia.subprofiles.yml` in the Drutopia project to add the new feature to the subprofiles.

Add or update code dependencies

If you added any new code dependencies like modules or themes to the feature you're working on, or if you need to update to a new version of those dependencies or of Drupal core, you will need to add these changes to `composer.json` files. These files are used by Drutopia developers.

Updating Drupal core

To update to a new release of Drupal core:

- Edit the value for `drutopia-core` in the `composer.json` file in the Drutopia repository.

Adding or updating contributed modules and themes

First, determine if the module or theme you're adding or updating is specific to a given Drutopia module (for example, adding the Admin Toolbar module to the Drutopia Site module). If so:

- Edit the module's `composer.json` file, adding a section for a new module or theme or changing the version number for an existing one.

Commit your changes

Once you have completed the changes, you can commit them to your local code base. Note that you may need to follow these steps in more than one project. For example, you may have added a field storage to the Drutopia Core module and added the corresponding field to a second Drutopia module.

- Review what changes have been made. This command will list off changed, added, or deleted files.

```
$ git status
```

- Add the changed, added, or deleted files.

```
$ git add -A
```

- Commit your changes and push them to GitLab. In this example, `10-admin_toolbar` is the name of the feature branch we're working in.

```
$ git commit -m "Issue #10: add Admin toolbar."
$ git push origin 10-admin_toolbar
```

Submit a merge request

Back in the GitLab project you're working on, use the website interface to submit what's called a "merge request". See the documentation on submitting merge requests..

In your merge request, reference the issue your work is addressing.

Tips

Before starting more development, get up-to-date with::

```
composer update
```

CHAPTER 18

Code of Conduct

18.1 1. Purpose

A primary goal of Drutopia is to be inclusive to the largest number of contributors, with the most varied and diverse backgrounds possible. As such, we are committed to providing a friendly, safe and welcoming environment for all, regardless of gender, sexual orientation, ability, ethnicity, socioeconomic status, and religion (or lack thereof).

This code of conduct outlines our expectations for all those who participate in our community, as well as the consequences for unacceptable behavior.

We invite all those who participate in Drutopia to help us create safe and positive experiences for everyone.

18.2 2. Expected Behavior

The following behaviors are expected and requested of all community members:

- Participate in an authentic and active way. In doing so, you contribute to the health and longevity of this community.
- Exercise consideration and respect in your speech and actions.
- Attempt collaboration before conflict.
- Refrain from demeaning, discriminatory, or harassing behavior and speech.
- Be mindful of your surroundings and of your fellow participants. Alert community leaders if you notice a dangerous situation, someone in distress, or violations of this Code of Conduct, even if they seem inconsequential.
- Remember that community event venues may be shared with members of the public; please be respectful to all patrons of these locations.

18.3 3. Unacceptable Behavior

The following behaviors are considered harassment and are unacceptable within our community:

- Violence, threats of violence or violent language directed against another person.
- Sexist, racist, homophobic, transphobic, ableist or otherwise discriminatory jokes and language.
- Posting or displaying sexually explicit or violent material.
- Posting or threatening to post other people's personally identifying information ("doxing").
- Personal insults, particularly those related to gender, sexual orientation, race, religion, or disability.
- Inappropriate photography or recording.
- Inappropriate physical contact. You should have someone's consent before touching them.
- Unwelcome sexual attention. This includes, sexualized comments or jokes; inappropriate touching, groping, and unwelcomed sexual advances.
- Deliberate intimidation, stalking or following (online or in person).
- Advocating for, or encouraging, any of the above behavior.
- Sustained disruption of community events, including talks and presentations.

18.4 4. Consequences of Unacceptable Behavior

Unacceptable behavior from any community member, including sponsors and those with decision-making authority, will not be tolerated.

Anyone asked to stop unacceptable behavior is expected to comply immediately.

If a community member engages in unacceptable behavior, the community organizers may take any action they deem appropriate, up to and including a temporary ban or permanent expulsion from the community without warning (and without refund in the case of a paid event).

18.5 5. Reporting Guidelines

If you are subject to or witness unacceptable behavior, or have any other concerns, please notify a community organizer as soon as possible. info@drutopia.org

Reporting Guidelines

Additionally, community organizers are available to help community members engage with local law enforcement or to otherwise help those experiencing unacceptable behavior feel safe. In the context of in-person events, organizers will also provide escorts as desired by the person experiencing distress.

18.6 6. Addressing Grievances

If you feel you have been falsely or unfairly accused of violating this Code of Conduct, you should notify Drutopia Leadership with a concise description of your grievance. Your grievance will be handled in accordance with our existing governing policies. (See *Drutopia Code of Conduct - Reporting Guide*)

18.7 7. Scope

We expect all community participants (contributors, paid or otherwise; sponsors; and other guests) to abide by this Code of Conduct in all community venues–online and in-person–as well as in all one-on-one communications pertaining to community business.

This code of conduct and its related procedures also applies to unacceptable behavior occurring outside the scope of community activities when such behavior has the potential to adversely affect the safety and well-being of community members.

18.8 8. Leadership Team Contact info

- Ben - ben@drutopia.org
- Clayton - clayton@drutopia.org
- Leslie - leslie@drutopia.org
- Nedjo - nedjo@drutopia.org
- Rosemary - rosemary@drutopia.org

18.9 9. License and attribution

This Code of Conduct is largely taken from The Citizen Code of Conduct distributed by Stumptown Syndicate under a Creative Commons Attribution-ShareAlike license.

18.9.1 Drutopia Code of Conduct - Reporting Guide

If you believe someone is violating the code of conduct we ask that you report it to the Drutopia Leadership Team by emailing info@drutopia.org. **All reports will be kept confidential**. In some cases we may determine that a public statement will need to be made. If that's the case, the identities of all victims and reporters will remain confidential unless those individuals instruct us otherwise.

If you believe anyone is in physical danger, please notify appropriate law enforcement first. If you are unsure what law enforcement agency is appropriate, please include this in your report and we will attempt to notify them.

In your report please include:

- Your contact info (so we can get in touch with you if we need to follow up)
- Names (real, nicknames, or pseudonyms) of any individuals involved. If there were other witnesses besides you, please try to include them as well.
- When and where the incident occurred. Please be as specific as possible.
- Your account of what occurred. If there is a publicly available record (e.g. a mailing list archive or a public IRC logger) please include a link.
- Any extra context you believe existed for the incident.
- If you believe this incident is ongoing.
- Any other information you believe we should have.

What happens after you file a report?

You will receive an email from the Drutopia Leadership Team acknowledging receipt immediately. We promise to acknowledge receipt within 24 hours (and will aim for much quicker than that).

The working group will immediately meet to review the incident and determine:

- What happened.
- Whether this event constitutes a code of conduct violation.
- Who the bad actor was.
- Whether this is an ongoing situation, or if there is a threat to anyone's physical safety.

If this is determined to be an ongoing incident or a threat to physical safety, the working groups' immediate priority will be to protect everyone involved. This means we may delay an "official" response until we believe that the situation has ended and that everyone is physically safe.

Once the working group has a complete account of the events they will make a decision as to how to response. Responses may include:

- Nothing (if we determine no violation occurred).
- A private reprimand from the working group to the individual(s) involved.
- A public reprimand.
- An imposed vacation (i.e. asking someone to "take a week off" from a mailing list or IRC).
- A permanent or temporary ban from some or all Drutopia spaces (mailing lists, IRC, etc.)
- A request for a public or private apology.

We'll respond within one week to the person who filed the report with either a resolution or an explanation of why the situation is not yet resolved.

Once we've determined our final action, we'll contact the original reporter to let them know what action (if any) we'll be taking. We'll take into account feedback from the reporter on the appropriateness of our response, but we don't guarantee we'll act on it.

Appealing

Only permanent resolutions (such as bans) may be appealed. To appeal a decision of the working group, contact the Drutopia Leadership Team at info@drutopia.org with your appeal and the leadership team will review the case.

CHAPTER 19

Research

We use the Design Justice Network Principles in researching, designing and building Drutopia.

"Design justice rethinks design processes, centralizing people who are normally marginalized by design and using collaborative creative practices to address the deepest challenges our communities face."

Here are the tools we've developed so far to help us conduct our research.

19.1 Survey

The survey exists at drutopia.org/survey

1. What is your name?
2. Describe the organization you work with.
3. What is your role in the organization?
4. What is your current annual budget for your website?
5. Less than $1,000
6. $1,000-5,000
7. $5,000-$15,000
8. $15,000-$30,000
9. $30,000-$100,000
10. Over $100,000
11. What is your current organization's website built on (eg: Wordpress, Squarespace, Wix, custom Content Management System)?
12. Drupal
13. Joomla

14. Nationbuilder
15. Squarespace
16. Weebly
17. Wix
18. Wordpress
19. Other. Please enter: [ ]
20. I don't know
21. How is your website hosted?
22. Shared hosting–name of hosting service if known 1/ VPS
23. How satisfied are you with your current website?
24. Scale: 1-5
25. What can you do well with your current website?
26. What is difficult/not possible to do with your current website?
27. How important is it that your website use free/open-source technology?
28. Very Important
29. Somewhat Important
30. Not Important
31. Unfamiliar with free/open-source technology
32. How important is that the technology tools you use be organized along democratic lines? (eg: tech cooperative, user-owned, etc.)
33. Very Important
34. Somewhat Important
35. Not Important
36. Rate the following website features from most to least important to your work.
37. Site analytics visualization
38. Blogging
39. Customer Relation Management (CRM integration)
40. Customizable templates (for layout and visual design)
41. Easy to add and manage content
42. Events
43. Mapping functionality
44. On-site donations
45. On-site fundraising campaigns (as opposed to needing to set up separate pages on GoFundMe, Funrazr, etc.)
46. Photo Gallery
47. Project/Chapter Directory
48. Resource Library

49. Strong Design
50. Other
51. How interested would you be in combining efforts with similar nonprofits to fund the development of technology otherwise out of your price range?
52. Very Interested
53. Somewhat Interested
54. Not Interested
55. What else should we know to build a user-owned platform that is relevant to nonprofits?

19.2 Interview

19.2.1 Questions

These are prompts that can be used in an interview of potential members. The goal is to validate or invalidate some of our assumptions about organizations' needs (see https://app.leanstack.com/canvases/245250). Pain points that resonate with people can then be tried as [Landing pages] to see if it can speak to a larger number of people through the medium of a web page— a necessity to scale. The goal with the potential member interviews, though, is to get people to open up and tell us about their organizations and their concerns. This gets us much more valuable information than getting people to say yes or no to a particular technology solution. (In the lean startup language, in particular from Ash Maurya's book Running Lean, this is the "problem interview" to understand our members' worldview. We're trying to mitigate product risk by seeing if people rank the problems we're trying to solve highly, market risk by seeing how people solve the problem today, and customer risk by seeing if the people who have the pain are willing and able to pay for solutions.) We should also be sure to get all the answers for the [User survey] from people who agree to be interviewed, either before the interview as a qualifying process or afterward as a follow up.

19.2.2 Potential prompts

1. What are your organization goals?
2. Who is it important that your organization reach for you to achieve your goals?
3. How do you try to reach them?
4. Are volunteers important to your organization meeting its goals?
5. How do you find volunteers?
6. If you don't use volunteers, why not?
7. Do you accept donations?
8. How important are donations to your organization?
9. Do you maintain a relationship with donors?
10. How do you thank donors and follow up with information?
11. How do you communicate with volunteers?
    - Text message
    - Email
1. How do volunteers commit to helping with work?

2. What do you need to tell your base?
3. How do you currently communicate this information to your base?
4. How do you maintain continuity of operations and institutional knowledge when staff or volunteers change?

19.3 Interview structure

(From Running Lean, chapter 7)

19.3.1 Set the stage

Thank you very much for taking the time to speak with us today.

We are currently working on a platform for web sites for grassroots organizations. We got the idea for the service from seeing friends and clients unable to get the web site quality and capabilities they needed at a price they could afford.

But before getting too far ahead of ourselves, we wanted to make sure other organizations share these problems and see whether this is a platform worth building.

The interview will work like this. I'll start by describing the main problems we are tackling, and then I'll ask if any of those resonate with you.

I'd like to stress that we don't have a finished product yet, and our objective is to learn from you, not to sell or pitch anything to you.

Does that sound good?

19.3.2 Collect demographics (test customer segment)

19.3.3 Tell a story (set problem context)

19.3.4 Problem ranking (test problem)

(State the top three problems and ask your prospects to rank them- change order presented in different interviews.)

Do you have any other communication or web technology problems or pet peeves I didn't talk about?

19.3.5 Explore member's worldview (test problem)

(This is the heart of the interview. The best script here is "no script". Go through each problem in turn. Ask the interviewees how they address the problem today. Then sit back and listen. Let them go into as much detail as they wish. Ask follow-up questions, but don't lead them or try to convince them of the merits of a problem (or solution).

19.3.6 Wrapping up (the hook and ask)

As I mentioned at the start, this isn't a finished product, but we are building a platform that will give organizations like yours a better experience and more powerful communication tools. The best way to describe the concept might be "SquareSpace for grassroots organizations" (replace SquareSpace with the name of the interviewee's existing service).

Based on what we talked about today, would you be willing to see the platform when we have something ready?

Also, we are looking to interview other people like yourself. Could you introduce us to other leaders of grassroots organizations?

19.3.7 Document results

(Take the five minutes immediately following an interview to document results while they're still fresh in your mind. Have two people present for the interview, such as notetaker and interviewer, independently fill this out. Then debrief and compare notes before entering into a system sharing the interview results with the rest of the team. Be sure to capture the words people used to describe their organization and their problems.)

19.4 Research Results

Below are the results from the research we have conducted so far.

19.4.1 Showing up for Racial Justice (Denver)

The Denver chapter of Showing up for Racial Justice has been experiencing a boom in interest. As a volunteer-driven group, it has been challenging to channel the energy of new participants meaningfully. Here are the user stories most important to them.

As a new supporter of SURJ, I want to know what action I can take right now, so that I can immediately do the work of racial justice. As a new supporter of SURJ, I want to know what kind of ongoing volunteer opportunities are available, so that I can contribute in meaningful ways.

As a volunteer coordinator, I want to know what skills and interests a new volunteer has, so that I can plug them into the right working group. As a volunteer coordinator, I want to know what experience a new volunteer has with racial justice, so that I can refer them to relevant resources for education. As a volunteer coordinator, I want to know what work volunteers have followed through with, so that I can determine how reliable they are.

CHAPTER 20

Adding a new module dependency

When in the course of feature development you need to add a module as a dependency, for example adding Video Embed Field module to Drutopia Core, we also need to ensure that this required module is enabled on sites that already have the feature module installed.

We do this by adding an update hook to the Drutopia feature module.

This means you may need to add a .install file to the feature module where there wasn't one befere.

Here is the full `drutopia_core.install` file for the `drutopia_core` module:

```
<?php

/**
 * @file
 * Contains install and update functions for Drutopia Resources.
 */

/**
 * Enable dependent module Video Embed Field.
 */
function drutopia_core_update_8101() {
  \Drupal::service('module_installer')->install(['video_embed_field']);
}
```

Note that if you were adding this module as a dependency to another Drutopia feature module (such as `drutopia_group`) your update hook would be named differently (`drutopia_group_update_8101()`) but would otherwise be the same. Ensure your update hook is numbered to be the last in the .install file you are editing.

If you have multiple modules to enable you could provide them to the function as `['video_embed_field', 'another_module']`.

All standard procedures for adding a dependency to a module still apply. Namely, add the module to both the .info.yml file and the composer.json file of the feature module:

Here is `drutopia_core.info.yml` with other dependencies removed for brevity:

```
name: 'Drutopia Core'
description: 'Provides core components required by other features.'
type: module
core: 8.x
dependencies:
  - video_embed_field
package: Drutopia
```

And here is the `composer.json` for `drutopia_core`, again with extraneous details removed:

```
{
  "name": "drupal/drutopia_core",
  "description": "Drutopia core is a base feature providing core components required
↪by other features.",
  "type": "drupal-module",
  "require": {
    "drupal/core": "^8.6",
    "drupal/video_embed_field": "^2.0"
  }
}
```

We *must* do this for feature modules with beta or full releases, but there's nothing stopping us from doing this for feature modules with alpha releases or even just in dev— existing update hooks are ignored when a module is installed, and won't ever be run. Therefore, adding update hooks for new dependencies is best practice for in-development features.

CHAPTER 21

Update an existing install into a development environment

If you want to do development on a bunch of Drutopia feature modules in an environment that was installed with something like `composer create-project drutopia/drutopia_template:dev-master --no-interaction DIRECTORY` (where DIRECTORY is whatever your directory is called) then you can follow these instructions.

NOTE As per the development documentation the recommended approach is to use Drutopia Dev Template.

When you've started with the release versions of Drutopia features so you should run the following script to replace those with git repositories before installing.

```
# Name this file fix_directories.sh and place it in the root directory of your
# Drutopia install (parallel to the vendor and web directories).
# Adjust file permissions so the file is executable.
# From a terminal in the directory you placed the file, run:
# ./fix_directories.sh
cd web/modules/contrib/
echo "Fixing Drutopia modules"
for dir in "drutopia_"*
do
  echo "Fixing $dir"
  rm -fr $dir
  git clone "git@gitlab.com:drutopia/$dir.git"
done
cd ../../profiles/contrib/
echo "Fixing drutopia install profile"
rm -fr drutopia
git clone git@gitlab.com:drutopia/drutopia.git
cd ../../themes/contrib/
echo "Fixing octavia theme"
rm -fr octavia
git clone git@gitlab.com:drutopia/octavia.git
cd ../../../
```

To use the script:

- Create a new file, `fix_directories.sh`, and put it in the root directory of your site.

- Adjust file permissions on `fix_directories.sh` as needed to add execute access.
- From the directory containing `fix_directories.sh`, run the following at the command line: `./fix_directories.sh`
- Visit your site and follow the Drupal installation instructions.
- Log into your new site and install the modules needed for development. These include:
- All Drutopia feature modules (which are listed on the modules page under the heading "Drutopia").
- Features UI and its dependencies (Features and Configuration Manager Base).

21.1 To update such an environment

It can be time consuming to pull in upstream changes for each of the many Drutopia features you're working with. The following shell script can help.

```
#!/bin/bash

# store the current dir
CUR_DIR=$(pwd)

# Let the person running the script know what's going on.
echo "Pulling in latest changes for all repositories..."

# Find all git repositories and update it to the master latest revision
for i in $(find . -name ".git" | cut -c 3-); do
    echo "";
    echo $i;

    # We have to go to the .git parent directory to call the pull command
    cd "$i";
    cd ..;

    # finally pull
    git pull --rebase origin 8.x-1.x;

    # let's get back to the CUR_DIR
    cd $CUR_DIR
done

echo "Complete!"
```

To use the script:

- Create a new file, `update-repositories.sh`, and put it in the parent directory of the directory of your Drutopia feature-module repositories.
- Adjust file permissions on `update-repositories.sh` as needed to add execute access.
- From the directory containing `update-repositories.sh`, run the following at the command line: `./update-repositories.sh`

CHAPTER 22

Extension selection criteria and candidate extensions

See also Drutopia *technical guide* and module usage information in #42.

22.1 Selection criteria

- Has a stable release on drupal.org, ideally with Drupal security team coverage.

22.2 Supported extensions

These are extensions already supported.

22.2.1 Configuration Update Manager

- Supplements the core Configuration Manager module, by providing a report that allows you to see the differences between the configuration items provided by the current versions of your installed modules, themes, and install profile, and the configuration on your site.
- Required by Features.
- Developer-only: should not be enabled by default.

22.2.2 Features

- Enables the capture and management of features in Drupal. A feature is a collection of Drupal entities which taken together satisfy a certain use-case.
- Developer-only: should not be enabled by default.

22.3 Candidate extensions

22.3.1 Modules

Address

- Provides functionality for storing, validating and displaying international postal addresses.

Admin Toolbar

- Improves the default Drupal Toolbar (the administration menu at the top of your site) to transform it into a drop-down menu, providing a fast access to all administration pages.

Advanced Editor Link

- Enhanced attributes for links.
- Integrates with LinkIt.

Block Visibility Groups

- Allows the site administrator to easily manage complex visibility settings that apply to any block placed in a visibility group.

Chaos Tools

- Provides functionality that didn't make it into Drupal Core 8.0.x.

Contact Storage

- Provides storage for Contact messages, which are fully-fledged entities in Drupal 8.
- In use on drutopia.org.

Devel

- A suite of modules containing fun for module developers and themers.
- Developer-only: should not be enabled by default.

Entity API

- Used for improvements to Drupal 8's Entity API which will be moved to Drupal core one day.
- Required by Profile.

Entity Legal

- Provides a solid, versionable, exportable and flexible method of storing legal documents such as Terms and Conditions and Privacy Policies.

Entityqueue

- Allows users to create queues of any entity type. Each queue is implemented as an Entityreference field, that can hold a single entity type.

Entity Reference Revisions

- Adds a Entity Reference field type with revision support.
- Required by Paragraphs.

Field Group

- Groups fields together. All fieldable entities will have the possibility to add groups to wrap their fields together. Fieldgroup comes with default HTML wrappers like vertical tabs, horizontal tabs, accordions, fieldsets or div wrappers.

Honeypot

- Uses both the honeypot and timestamp methods of deterring spam bots from completing forms on your Drupal site.

LinkIt

- Used for internal links (to nodes, taxonomy terms, etc.)
- Use the 5.x branch, majorly refactored.

Maxlength

- Allows you to set maximum length of any field on any form making use of the form API.

Paragraphs

- Comes with a new "paragraphs" field type that works like Entity Reference's. Simply add a new paragraphs field on any Content Type you want and choose which Paragraph Types should be available to end-users. They can then add as many Paragraph items as you allowed them to and reorder them at will.

Pathauto

- Automatically generates URL/path aliases for various kinds of content (nodes, taxonomy terms, users) without requiring the user to manually specify the path alias.

Profile

- Provides configurable user profiles.

Social Media Share

- Allows the user to share current page to different social media platforms.
- Most providers don't expose data.

Redirect

- Allows content editors to change content paths without breaking links.

Token

- Provides additional tokens not supported by core (most notably fields), as well as a UI for browsing tokens.
- Required by Entity Legal, Pathauto.

Video Embed Field

- Creates a simple field type that allows you to embed videos from YouTube and Vimeo and show their thumbnail previews simply by entering the video's url.

22.3.2 Themes

While Drutopia (as a platform) intends to leverage the Twig templating layer to enable site owners to have full control over their templates as well as CSS, the Drutopia distribution will include at least one theme and the Drutopia platform will invite further contributed themes that meet the following criteria:

- Responsive design
- Accessible contrasts for color blind and slightly visually impaired visitors

CHAPTER 23

Drutopia technical guide

23.1 Decision Making

If a technical issue arises that is not covered in tech guide then the developer is empowered to make a decision on their own but then have their work reviewed by a tech lead.

Tech Leads are:

- Ben
- Clayton
- Nedjo
- Rosemary

23.2 Principles and process

23.2.1 Principles

- Drutopia features should provide the basic functionality that most sites or distributions will need.
- Any advanced or specialized functionality should be provided in separate features that require more basic features.
- The ease of discovery and use of website feature for both admins and end users should be an overriding goal.
- Every main task should have only one primary admin interface.
- Common protocols and solutions should be used to ensure consistency and ease of use.

23.2.2 Uniqueness and compatibility

Wherever possible, each Drutopia feature should cover a unique area of functionality. For example, there should be only a single Drutopia feature that provides general blog functionality. Exceptions can be made where the underlying framework used is radically different and there is no common base to build on.

23.3 Available on the Drutopia infrastructure, mirrored on drupal.org

To ensure visibility, all Drutopia features should be hosted on the Drutopia infrastructure (currently on GitLab) and mirrored on drupal.org.

23.3.1 Standard solutions

To help produce uniformity among the various Drutopia features, standard solutions are adopted across Drutopia. For example, for mapping, there may be several Drupal modules available,–Leaflet, OpenLayers, and so on. To ensure consistency, one should be designated as the Drutopia standard.

In selecting between alternatives, the Drupal principles are relevant as well as the following additional criteria:

- *Community free software*: The candidate solution is a community-driven free software solution.
- *Non-proprietary*: The candidate solution is not limited to, branded by, or controlled by a single company.
- *Secure*: Any module must have a stable release on Drupal.org and so be supported by the Drupal security team.
- *Data security*: The candidate solution does not require posting data to an external source unless doing so is the explicit purpose.
- *Works out of the box* without requiring manual configuration, e.g., entry of an API key. (Except where required, as for payment processing)

Before introducing a new Drutopia feature that introduces a dependency that could be considered as a new standard solution, an issue must be posted on the Drutopia project proposing the new standard solution with justification.

Adopted standard solutions:

- Pathauto: where appropriate, paths variables should be set so that, if Pathauto is present, content types defined by a feature will get appropriate paths.
- Rules should be used for any configurable actions.

23.3.2 Opportunity for peer review

There should be an opportunity for review prior to a new Drutopia feature being posted. Such review will help ensure that new features indeed meet the Drutopia spec, e.g.,

- identify any potential components that could/should be pulled into more specific Drutopia features,
- identify properties that are too site-specific to warrant inclusion in Drutopia,
- if the feature introduces a new solution to standardize (e.g., an external library or a way to structure particular data), ensure a consensus prior to adoption.

Therefore each proposed Drutopia feature should be posted first as an issue on the drutopia-distribution project, with a link to the sandbox version of the proposed feature and two weeks allowed for peer review.

23.4 Technical guidelines

23.4.1 Configuration Altering

Sometimes we need to alter the configuration provided by a feature. We do this by using Config Actions.

23.4.2 Naming and namespaces

Code namespace

A Drutopia feature must include the Drutopia namespace.

- Example: `drutopia_event`.

In the Features module, this namespace is provided by a features *bundle*. A feature is named for the combination of a bundle prefix (`drutopia`) and a short name (`event`).

Item naming

Individual configuration items should be named using the short name of the target feature where possible. Examples:

- Machine name of a content type: `event`.
- Machine name of a field storage specific to blogs: `field_event_date`.

23.4.3 User roles and permissions

User roles

Drutopia features should use the following roles as appropriate.

- *administrator* is used for site administration tasks, such as installing and configuring modules, content types, and blocks.
- *contributor* is a Drupal user who contributes content, e.g., a staff member at an organization or company.
- *editor* is a Drupal user responsible for editing and administrating content, taxonomies, and comments.

These roles will be provided by the drutopia_core feature and so will be present if Drutopia is installed.

A Drutopia feature may provide additional roles, but such roles should have a scope no greater than the scope of the feature. For example, a blog feature might provide a *blogger* role. However, a technology blog feature should not provide a *blogger* role, since that role would be relevant to a more general blog feature.

Permissions

The editor role should not be expected to have *administer nodes* permissions.

For each content type introduced, roles should be assigned as follows:

- *contributor*: *create [type] content*, *delete own [type] content*, *edit own [type] content*
- *editor*: *create [type] content*, *delete all [type] content*, *edit all [type] content*

where [type] is the machine name of the content type being introduced by the feature.

Note that user permissions are not yet supported in the Features module–see this issue.

23.4.4 Paths and breadcrumbs

Primary path and nesting

Where a feature provides content or nested pages, these should wherever feasible be available at paths nested below the feature's primary path.

A feature that defines a content type should register a pathauto pattern that nests content of that type below the feature's primary page.

Example: a blog feature would set a pathauto pattern to `blog/[node:title]`.

The machine name of the pattern should be in the form `[entity type]_[bundle]`. Example: `node_blog`.

Menu location and breadcrumbs

Menu location and breadcrumbs for pages created by the feature should match the nesting used for paths.

23.4.5 Block visibility and theme regions

Block visibility

TBD: what page layout solution are we adopting?

23.4.6 Theme regions

TBD: how are we approaching theme regions?

23.4.7 Code and dependencies

A feature may not depend on any feature that is not fully compliant with this specification.

Versions

Features should use the recommended stable release versions of contributed modules and themes.

Libraries

Any external libraries used by a feature should be installed in a libraries directory if possible.

Patches

Patches should be avoided wherever possible. Ideally, patches should be used only up to and including beta releases, to include functionality expected to reach stable releases by the time the feature is out of beta.

Where feasible, patches should be replaced with workarounds in a feature's custom code that use Drupal APIs to achieve the same ends. These workarounds should include documentation identifying the patch that they relate to so they can be removed if/when that patch is accepted.

Drush make file/Composer

TBD: how should code dependencies be handled?

23.4.8 Content types and fields

Where a feature defines a content type, the following guidelines should be followed:

Image fields

- If a single image is desired to represent each piece of content in the content type, the content type should use field_image as used in e.g. drutopia_article.
- If media (images, audio, video, files) are desired, the content type should use field_media as used in e.g. drutopia_article.

23.4.9 Views

For consistency, the following conventions should be used for views included in features.

- Naming: if the view is primarily for presenting a single content type, it should be named for the machine name of that content type.
- At a minimum, views of a content type should include a page, block, and feed display, with the feed display attached to both the page and block. The page display should have a path matching the content type's machine name and the feed should have the same path with an .xml extension.

23.4.10 Help

A feature may include contextual help provided through the regular Drupal help API covering major components of the feature.

CHAPTER 24

How to produce a new content feature

These instructions will use the example of a yet to be built "resource" feature.

24.1 Naming conventions:

- See also the technical guide section on naming and namespaces.
- The content type will be called: `resource` (singular)
- Any field that is specific to this feature will be prefixed with `resource_`, so for example a reference field to a vocabulary specific to resource might be called `field_resource_type`.
- More naming conventions will be discussed in the relevant sections such as Pathauto, Views and Creating your feature.

24.2 Fields: what fields are re-used, what are required

- Many of the core field are reused between features.
- These include
- `field_summary` (required)
- `field_body_paragraph`
- `field_image`
- `field_topics`
- `field_tags`
- These fields should be added and configured as needed. Because we are using paragraphs for our main body, a required summary field is necessary for many display purposes and should always be included.

- When adding a new field, first determine if it will be specific to your feature or is perhaps a more generic field. If you think it is more generic, talk to a tech lead to consult. If it is specific, give it a name that includes the content type name, e.g. `field_resource_type`.
- When adding reused fields, if you have queries about any configuration, look at another content type feature for guidance.

24.2.1 Paragraph types:

- Drutopia features use paragraphs as the main body field (`field_body_paragraph`). For each feature you can set what paragraph types should be included. All `field_body_paragraph` fields should have at least text, image and file paragraphs. Other paragraph types can be included if they seem useful to your content type.
- If you think you need a new paragraph type, discuss with a tech lead.

24.2.2 Metatag

- The metatag field (`field_meta_tags`) should be included in every content type feature.

24.2.3 Form display

- The traditional Body field (`field_body`) has been maintained for migration purposes. It should always be hidden on the form display.
- Refer to another content feature such as `drutopia_article` as a model for the general ordering and setting guidelines.
- For the Body paragraph field, ensure the following settings are set:
- Edit mode: Open
- Add mode: Buttons
- Default paragraph type: Text

24.2.4 Display

- We do not use the default view mode, so ensure there are no fields showing.
- Enable the view modes that you envision using. At the very least there needs to be a full view mode and a teaser.
- We are using Display Suite so you need to set a Display Suite view mode before doing the configuration. The full content display is often using the "Three column stacked – 25/50/25" layout.
- Look to other content types to see how they are using some of the other more specialized view modes and which Display Suite layout they are using.
- If you want a Promoted block for the home page, you will need to format a Card view mode, see below for further details.
- For some of the view modes, you will need to add one or more field groups so that the theming will work correctly. Drutopia Article has the most developed view modes so it is a good example to work from.
- Where labels should be hidden, please use "Visually hidden" wherever possible.
- For the Image field, use Responsive image and set to wide or narrow as seems appropriate for your view mode.
- If you feel that you require a new view mode for your content type, please consult a tech lead.

- Enable and configure the search_index view mode with all labels hidden.

Formatting guidelines for card view mode

- Enable `card` in Custom display settings.
- Ensure this view mode is using the Display Suite one column layout.
- Add the image field to the content area, selecting responsive image, narrow and linked to content.
- Next, create a field group for the card view mode.
- This is a group of type HTML element and must be named Card content, `card_content`. Settings can be left at default.
- Add this field group to the content area.
- Then place into it the desired fields:
- Title, linked to content.
- Summary, trimmed to 180 characters.
- Add a `type` field or `tags` as desired.
- Save your card settings.

Note: When using the card view mode in a view, ensure that the field "Add views row classes" under setting is unchecked.

Display Suite fields

Where there is a need for specialized display in a view mode, we rely on Display Suite fields. Examples:

- To concatenate two fields, add a Display Suite token field and use tokens to concatenate the fields.
- If there is a need to place a block in the view mode:
  - Create the block as e.g. a view.
  - Add a Display Suite block field, selecting the block in question.

Whenever creating a Display Suite field, configure to limit it to only the particular entity bundles that are applicable. For example, if the field is only for events, limit it to the `event` bundle.

24.3 Pathauto

- A content type should always have an associated Pathauto pattern.
- Create a new pattern:
- Type: Content
- Path: `resources/[node:title]`
- Select the content type it applies to (for example Resource)
- Label: Node resource

24.4 Views

- As of the alpha5 release, content type views are based on a search index for that content type. Eventually we'll be adding facets to these listing pages.

24.4.1 Create an index for your content type

- Ensure you have `drutopia_dev` and `entity_clone` installed and enabled.
- Clone the node_dev index to create a search index for each content type limiting it to that one content type. Give it the same name as the content type and update the description.
- Configure the search_index view mode for your content type and use it in the rendered HTML field on the search index.
- Add any fields that are specific to the content type, for example a type vocabulary. This will be needed for any field that we want to use for facets.
- Flush caches before proceeding.

24.4.2 Create a view

- Create a new view based on the search index for your content type using the singular name of the content, for example `resource`.
- Most content types will need a page display and ideally a block display for promoted content. Other displays may be needed for a particular content type use case.

24.4.3 Page display

- The page display should use the plural in the path, for example `/resources`.
- A menu item should also be set here for easy export.
- Give it a normal menu entry, with the parent menu
- Give it a title, for example "Resources", and a description, such as "Get connected with our resources."
- Under format settings, ensure that Add views row classes is unchecked.
- Under format, show, use the rendered entity selecting an appropriate view mode (often card).
- Sort criteria
- Sticky
- Content: Authored on (desc)
- Set the machine name to `page_listing`
- Set caching to none.

24.4.4 Block display

- Create a block display for promoted content.
- Display name: Block Promoted

- Under Format choose Content and the Card view mode.
- Add a filter Content: Promoted to front page (= Yes)
- Limit to 4 items.
- Under Advanced, give it the machine name: `block_promoted`

24.5 Creating feature and determining where fields go

- To create a Drutopia feature you need to enable the Features and Feature UI modules.
- Via the features interface (`admin/config/development/features`), start by ensuring that you have Drutopia selected as your features bundle.
- In Drupal 8 much of the features bundling happens automatically. By using the assignment plugins at their default, your configuration should be assigned to a new feature that will be shown as unexported.
- The included configuration link will show what is included in a general way.
- The link to the feature itself will take you to a page where you can review the configuration in more detail, adding or removing any items if necessary.
- The most important areas to review are the field storage and field instance. Field storage should only include fields that are unique to your content type. Field instance will include all field used.
- If you have added a new shared field (after consultation) that field storage will need to be added to Drutopia Core and this needs to be done ahead of creating the new feature.
- The machine name of the feature will be `resource`. (Question: This is because it's automatic or should it be edited to `drutopia_resource` by convention?)
- Edit the **Description** if needed.
- Ensure Drutopia is set as the **Bundle**.
- Set the **Version** to `8.x-1.0-alpha1` if it's the first time. Once accepted into Drutopia, it should match the stability of the Drutopia distribution.
- To add the search_index view mode you will need to edit the feature, allow conflicts (as this view mode will be added to the seach feature by deafault) and manually add.
- Export your feature. (If using **Write**, you can set the **Path** to `modules/contrib` presuming you intend to contribute it.)
- After export, manually edit the info file to give it the name 'Drutopia Resource'.

24.6 Adding .json file

- Drutopia uses Composer so each feature needs a `.json` file.
- The `.json` file needs to includes direct dependencies only.
- Use another feature as a model and be sure to include any new modules you have needed for your feature.

24.7 Contribute the feature module

- Initialize a git repository in the feature module. For example, `cd modules/contrib/drutopia_resource`, `git init`, `git checkout -b 8.x-1.x`, `git branch -d master`, `git add .`,`git commit -m "Create the Drutopia Resource feature"`.
- Go to the Drutopia organization on GitLab and create a new project with the same folder/machine name as your feature module (or, if you do not have permissions in the Drutopia organization, start the project in your own namespace). Make it a public project (**Visibility Level** Public).
- Follow GitLab's instructions for an existing folder which we haven't already done; for example `git remote add origin git@gitlab.com:drutopia/drutopia_resource.git` and `git push -u origin 8.x-1.x`.

24.8 Adding permissions via config_actions

- Permissions are not directly exportable so they need to exported via config_actions.
- See the page on adding user permissions.

24.9 Default content

- Ideally a content type includes default content.
- See detailed instructions: Producing and exporting default content for Drutopia.

24.10 Settings provided by other modules

As a general guideline, we can export to regular features *only custom configuration that we have ourselves created*, such as a view mode or a field. If we're configuring using a settings form, that configuration will already be provided elsewhere (for example, by the `chosen` module). A hint that this may be occurring is that the name of the configuration includes another module's name, such as `chosen.settings.yml`. Another hint is if the configuration does not show up by default as available for adding to your feature (that is, you have to check the option to allow conflicts before it will show up).

We have two options for such configuration:

- The first is to add it to the install profile, which is allowed to override configuration provided by other modules. This option is appropriate if:
  - The configuration should always be active on all sites; and
  - The configuration is not required by other configuration or functionality. This consideration is because the install profile is typically installed last and nothing should require the install profile.
- Otherwise, we can add it as a config action per the `config_actions` module. See the Config Actions documentation.

For example, to change a configuration option provided by the `realname` module, you might add a file `config/actions/realname.settings.yml` with the following content:

```
path:
  - pattern
plugin: change
actions:
  'full name':
    value: '[user:field_user_first_name] [user:field_user_last_name]'
```

(where `full name` is an arbitrary key identifying the change).

CHAPTER 25

How to extend an existing content feature

There will often be use cases where a developer wants to extend an existing content type by adding additional fields without requiring that those fields be added to every site. For example, the Drutopia Event module provides an event content type. Say we want to provide an optional location field for every event. How to do this?

Drutopia features have been intentionally created in a modular way trying to limit the number of dependencies between features. This means that when we want to add fields that would otherwise create dependencies, we do so in what might be called a "bridging" module—a module that bridges by adding a new field to an existing content type provided by another Drutopia module. We do this using the Config Actions module.

The Drutopia Storyline feature provides a working example that can be used as a model to follow when using this pattern. You will be creating two new modules. They can, however, be bundled into one project.

Here's an outline of the steps to follow:

- Start by creating and configuring the field as well as its form and view displays. In the example, our new field is `field_storyline` and is added to the `page` content type.
- Create a new base feature (in our example `drutopia_storyline`).
  - This will provide the field storage for the new field.
  - Add a dependency on config_actions manually by editing the feature's `.info.yml` file .
  - Edit the exported `.yml` file for the field storage, setting `persist_with_no_fields` to `true`.
- Create the bridging module (in our example `drutopia_page_storyline`). You can add it directly to the directory of the base feature, so that they are both in the same project.
  - The field will be placed in the `config/install` folder.
  - Edit the `.info.yml` file of the exported feature to add dependencies on both the base feature (in our case, `drutopia_storyline` as well as the module that provides the content type you're adding the field to (in our case, `drutopia_page`).
  - Manually write config actions files to add your new field to relevant configuration. Save these files to an `actions` folder inside the `config` folder. The actions will include the changes impacted by adding the new field such as the form mode and any display mode used.

- To determine what code needs to go into the new config actions files, use Drupal core's built-in functionality to export a single configuration item. For example, in our case, assume one of the configuration changes was to add `field_storyline` to the `full` view mode the `page` content type.
  - Navigate to Configuration > Development > Synchronize Configuration > Export > Single. For "Configuration type" select "Entity view display" and for "Configuration name" select "node.page.full".
  - In the YAML markup that's displayed, look for occurrences of the name of the field you added. In our case, this is `field_storyline` and they occur in two places: under `dependencies.config` and under `content`. Both of these changes need to be registered in a config actions file.
  - Create a new file, naming it for the config item you're altering. The file name is given on the export form, below the YAML text area. In our case, the file will be named `core.entity_view_display.node.page.full.yml`. Rather than working up a new file from scratch, look for one you can model yours on. See the examples in Drutopia Page Storyline. From the exported YAML code, you can copy and paste in the entire section giving the configuration for your field. See the below example for `drutopia_page_storyline`, which includes comments.

```
plugin: 'add'
actions:
  # Add config dependency.
  config_dependencies:
    path: ["dependencies", "config"]
    # Edit the next line to use the name of your field.
    value: "field.field.node.page.field_storyline"
  # Add the entity form display settings.
  content:
    path: ["content"]
    value:
      # This is the portion to replace with what you copy from exported YAML.
      field_storyline:
        type: entity_reference_paragraphs
        weight: 3
        settings:
          title: Paragraph
          title_plural: Paragraphs
          edit_mode: open
          add_mode: button
          form_display_mode: default
          default_paragraph_type: storyline_header
        third_party_settings: {  }
        region: content
```

CHAPTER 26

Adding user permissions

26.1 Overview

The Config Actions module allows you to alter existing configuration. We're using it to add permissions to roles. We do so because the role needs to be provided by one feature (usually, `drutopia_core`), while permissions need to be provided by various other features, such as a feature that provides a content type.

26.1.1 Technical details

It works on the basis of plugins, each of which applies a different sort of action. One of the plugins is `add`, which will add to an existing configuration item.

The `path` you specify answers the question "what specific parts of the configuration do you want to change?" In our case, the path is `permissions` because that's the part of the user role we want to change.

Want to know more about how Config Actions works? There is extensive documentation available.

26.1.2 How to add permissions

- Configure your site and add the permission(s).
- In the feature you're working on, look for a directory called `config/actions`. If it does not already exist, create it.
- Edit the `composer.json` file and, if it doesn't already exist, add the line `"drupal/config_actions": "^1.0-beta1",` to the `require` section.
- Edit the `.info.yml` file and, if it isn't already there, add `- config_actions` to the dependencies section and ensure that `drutopia_core` is also there as a dependency.
- For each role that you added a permission to:
  - Determine if there is already a file `config/actions/user.role.[role_name].yml`, where [role_name] is the machine name of the role. If not, create it, using the following for the file content:

```
path:
  - permissions
plugin: add
actions:
```

- On the site, use the admin UI to export the role you added the permission to. You do this by selecting "Configuration > Development > Synchronize configuration > Export > Single item" (admin/config/development/configuration/single/export). For "Configuration type", select "Role" and then for "Configuration name" select the role you're exporting. An export of the role will appear in the form. Here is an example:

```
uuid: 19da04f2-c4c7-4f34-b47e-f52e2bc315fc
langcode: en
status: true
dependencies: {  }
id: manager
label: Manager
weight: 3
is_admin: null
permissions:
  - 'access administration pages'
  - 'access toolbar'
  - 'administer thingumajigs'
```

Under `permissions`, you get the machine names of all permissions assigned to the role. Look for the ones you assigned (note: the machine names may differ from what you saw through the UI). For each permission you're adding, add a line like the following to the user role file in `config/actions`.

```
  '[permission name]':
    value: '[permission name]'
```

where [permission name] is the machine name of the permission. For example, your file might end up like this:

```
path:
  - permissions
plugin: add
actions:
  'administer thingumajigs':
    value: 'administer thingumajigs'
```

26.1.3 Troubleshooting

If, after adding permissions with the config actions approach, you get the following cryptic message when doing a site install:

In ConfigEntityStorage.php line 252:

```
The entity does not have an ID.
```

It's quite possible you forgot to add or enable a module that provides the role (or possibly other related configuration).

CHAPTER 27

Synchronizing Configuration

Drutopia relies on a complex suite of modules that have been written to allow for shared configuration in Drupal 8.

Drutopia member Nedjo Rogers has written comprehensive technical details in an eight-part series. If you want to understand what is at play here this series comprises the following posts:

- Managing Shared Configuration Part 1: Configuration Providers
- Managing Shared Configuration Part 2: Configuration Snapshots
- Managing Shared Configuration Part 3: Respecting Customizations
- Managing Shared Configuration Part 4: Configuration Alters
- Managing Shared Configuration Part 5: Updating from Extensions
- Managing Shared Configuration Part 6: Packaging Configuration with the Features Module
- Managing Shared Configuration Part 7: Core Configuration
- Managing Shared Configuration Part 8: Summary and Future Directions
- Bonus post: Drupal distributions, blocks, and subthemes

While the back-end is incredibly complex, the goal for site administrators is to provide a simple way for you to get new improvements from Drutopia and its features, while allowing you to keep any simple customizations you have made.

27.1 Importing configuration updates:

- Navigate to the Distribution Update page (admin/config/development/configuration/distro).
- The help text at the top of the page provides clear guidance:Any available configuration updates from installed modules or themes are displayed here.

  If you're an advanced user, you may wish to use the checkboxes to select which modules and themes to run updates from. Running some updates while skipping others can lead to unexpected results and so should be done with caution.

By default, changes will be merged into the site's active configuration so as to retain any customizations you've made. For example, if you've edited the label of a field for which updates are available, that edit will be retained.

Advanced users may choose instead to reset configuration to the state currently provided by installed modules, themes, and the install profile.

- For most instances, you will merge in all the changes by clicking the import button.
- You will get any new changes to features, such as new fields or a new paragraph type, while customizations you made, such as changing the title of a view, will be maintained. While you have now added the newly available elements, changes you have made to your site's configuration will not be overridden, such as a views title that you changed from Article to News.

27.2 Advanced reset options:

- There are also two advanced modes to reset configuration to its default settings (that is, overriding any customizations you have made).
- The partial reset option resets only configuration for which there is a change available.
- The full reset option, resets all configuration on the site (including for example the site name and email address) to the default values as if the site were just being installed.
- If you want to use these advanced options (only recommended for developers), start by selecting the update mode, and clicking the "Change update mode" button.
- Once the page has reloaded, you can review the changes and then if desired, click the Import button.

27.3 Importing configuration from a newly turned on feature:

- Importing configuration is also a tool to use when enabling a new Drutopia feature after your initial install.
- For example, when you initially installed Drutopia you did not select the Campaign feature. Now you'd like to add it to your site.
- Enable the feature from the modules page as usual. You'll get the campaign content type, but certain optional configuration will not be supplied, for example the block of promoted campaigns for the home page.
- You can easily pull in this block by importing the configuration as described above.
- Your home page will now be updated to include that home page block (and its default content).

CHAPTER 28

About Drupal Distributions

> Distributions provide site features and functions for a specific type of site as a single download containing Drupal core, contributed modules, themes, and pre-defined configuration. They make it possible to quickly set up a complex, use-specific site in fewer steps than if installing and configuring elements individually.
>
> —The drupal.org page listing Drupal distributions

28.1 General Introductions

- The Beginners Guide to Drupal Distributions.
- Introduction to distributions (drupal.org).

28.2 Technical Guides

- Drupal 8 Distribution Development and Devops Workflow (video from BadCamp 2016).

28.3 Case Studies

- Open Social, a Community and Intranet Solution.

28.4 Advanced Topics

- Distributions and Install Profiles: The Challenge and the Glory (video, DrupalCon Vienna 2017).

CHAPTER 29

Drutopia Base Distribution

Drutopia Base is the flagship libre software product of the Drutopia initiative. It provides the building blocks for the features grassroots groups need to organize, inspire, and mobilize.

29.1 Features

Drutopia Base comes with:

- Actions
- Articles
- Blog Posts
- Campaigns
- Groups
- Profile Pages
- Octavia : A Bulma-based theme

29.2 Roadmap

See Drutopia milestones on GitLab

CHAPTER 30

Solidarity

Solidarity is an online platform cooperative designed by grassroots organizers to help one another act, mobilize and support one another.

The platform is built around four main features: groups, campaigns, actions and news.

View the Solidarity Prototype

30.1 Personas

See solidarity-personas

30.2 Groups

Each group has a page within the website. A group can launch a campaign, call for an action and publish a news article. All of these pieces of content are displayed on a group's page, creating a mini site.

30.2.1 Potential Future Features

- Group membership
- Group only content
- Integration with other tools such as: Discourse, Facebook, Loomio, Slack, Twitter

30.3 Campaigns

A campaign page is a central place to explain the issue, publish news articles about its activity, list out demands, post calls to action and raise funds.

30.3.1 Potential Future Features

- Onsite Petition

30.4 Action

An action is a call for concrete work from supporters. It can stand on its own or be part of a campaign. It can have a goal and due date set,. For example, a call for 50 people to call a prison by June 13th demanding an investigation into prisoner abuse. An action can also be turned into a fundraiser, allowing people to donate directly to the site through Stripe. No longer will groups be dependent on proprietary platforms such as GoFundMe and IndieGoGo.

30.4.1 Potential Future Features

- SMS integration
- Badges and other game design features

30.5 News

News articles on Solidarity contain depth and functionality beyond what is found on a typical site. An article can be part of a larger campaign or associated with specific groups. By relating an action to the article, the post now becomes a mobilizing opportunity for readers. An article can also be posted automatically to social media accounts including: Facebook, Mastodon, and Twitter.

30.5.1 Potential Future Features

- Collaborative, moderated editing of an article (think wiki meets blog)
- SMS integration

30.6 Additional Features

Solidarity also boasts a customizable homepage, user management, content moderation tools and all of the extendability that comes with Drupal. Because it is open-source, it will always be free to download. Your data is yours and well structured to export to another platform at any time.

30.7 Future Features

We believe in constant improvement and iteration. Future features will be driven by the needs and feedback of the Solidarity community. Some of the additional features being talked about right now are:

- Additional themes
- Customizable page layouts and templates
- Events
- Resources

- More sophisticated fundraising
- Constituent Relationship Management integration
- Mobile app for your site

CHAPTER 31

Creating a Distribution using Drutopia

The *Drutopia base distribution* is designed to make it easy to build your own distribution using as much or as little of Drutopia as you require.

See *About Drupal Distributions* for introductory material.

31.1 Proposing a New Distribution

A new Drutopia distribution should be proposed as an issue in the Drutopia Organization project. The proposal should include a suggested name, purpose and initial list of features.

31.1.1 Criteria

A distribution qualifies for acceptance into the Drutopia ecosystem if it is:

- Appropriate : Meets the needs of a core Drutopia audience group (small budget-organization) and will build off of Drutopia Base
- Feasible : Can be implemented with tools presently available
- Resourced : Has a team capable and available to build and maintain the distribution

The final decision as to whether a distribution is added is made by the Leadership Team.

31.2 Design Features

Key design factors facilitate building distributions on Drutopia.

The Drutopia Core feature provides a standard set of site building components such as view modes, paragraph types, fields, user roles, and more that are ready to use in custom features.

Unlike in many distributions where all features are in a single repository, in Drutopia each feature has a distinct repository. This means it's easy to build a custom distribution using only the specific features you need. As your requirements evolve, you can return to add further Drutopia features.

Built on the Bulma CSS framework, the Octavia theme provides integration for a standard set of Drutopia elements such as view modes that can be reused in custom distributions built on Drutopia.

However, Octavia and Bulma are in no way required. Distribution developers may select whatever theme approach they desire and refer to Octavia as a model for what needs theming.

31.3 Deciding Whether to Base your Distribution on Drutopia

Criteria to consider include:

- *How closely does your use case match the nonprofit/grassroots focus of Drutopia?* While Drutopia could be relevant to a broad range of use cases, several of the features are specifically aimed at the needs of nonprofit organizations and grassroots groups.
- *How prepared and/or well positioned are you to contribute back?* Drutopia is set up to encourage groups building off the platform to also help enhance the platform in ways that meet their needs.
- *How flexible are you to meeting your specific needs in ways that meet broadly shared requirements?* Drutopia features are built to meet the common requirements of many organizations.
- *Do you want ongoing updates?* There are two basic types of Drupal distributions: starter kits (designed to be installed only once and then the site is taken in its own direction) and full featured distributions (which will be maintained, as well as having new functionality added). Drutopia is a full featured distribution, so if you're building a starter kit it may not be the best choice.

There are many potential advantages to building your distribution off of Drutopia, among them:

- *Out of the box functionality*. Some of your use cases may already be met by the growing list of Drutopia features.
- *Curated extensions*. The set of extensions included in Drutopia have already been evaluated and specifically selected.
- *Model features*. The existing Drutopia code base provides models of how to address many of the complexities of distribution development including default content, feature-specific user permissions, configuration overrides, configuration updates, and more.
- *Development documentation*. The Drutopia development documentation provides detailed guidelines for building features on the Drutopia model.
- *Updates*. Drutopia is actively maintained and updated.
- *Cooperative model*. Drutopia is structured on a cooperative model enabling members to both shape and benefit from development.

At the same time, there are potential tradeoffs that should be carefully considered, including:

- *Alpha status*. Drutopia is in alpha and so may undergo changes as it stabilizes to reach a first full release.
- *Architectural decisions*. As is generally true of Drupal-based distributions, Drutopia brings with it a certain set of assumptions and structural decisions. While in many cases it's possible to override and customize, the features will be of most value when they can be used straight up.

31.4 Create an Install Profile

An install profile is a necessary piece of a custom distribution. See How to Write a Drupal 8 Installation Profile.

Drupal core does not support inheritance in install profiles, meaning by default it's not possible to base one install profile on another.

That said, for the adventurous, there is a patch available to enable this functionality; see Allow profiles to provide a base/parent profile and load them in the correct order.

The default option is to build a stand-alone install profile. The easiest way to do so is to crib from the Drutopia install profile itself.

31.5 Create a Theme

For the simplest sub-distributions, it might be sufficient to use the Octavia theme directly:

```
composer require drupal/octavia
```

In most cases, the most effective starting place will be to sub-theme the Octavia theme that ships with Drutopia.

Install Octavia:

```
composer require drupal/octavia
```

Then follow the documentation for Creating a Drupal 8 sub-theme.

You can also, of course, write a custom theme. In doing so, you'll want to design for the components used in Drutopia, such as view modes.

You'll also need to provide some or all of the block configuration that's provided in the Octavia theme; see that theme's `config/optional` directory.

You may also wish to look at the Twig templates included with Octavia, as some may be relevant to your custom theme.

31.6 Select and Install Drutopia Features

Use composer to require an individual Drutopia feature like any other Drupal plugin. For example, for `drutopia_core`:

```
composer require drupal/drutopia_core
```

As a minimum, include the following features:

- Drutopia Core
- Drutopia Site

Most sites will also want some or all of:

- Drutopia Article
- Drutopia Comment
- Drutopia Page

CHAPTER 32

Hosting with Ansible

All components of Drutopia hosting are public as free/libre open source software as part of our commitment to Libre-SaaS.

But you don't need to know about any of this to have a site on the Drutopia platform.

Nor do you need to use this approach to hosting your own Drutopia site(s), but you can!

As befitting techincal documentation, the main resource is the Drutopia Host README

Every site on the platform has the concept of a *custom repository*—which is used only to get configuration, templates, and other non-PHP theme files like CSS and JavaScript—and a *drutopia_version* which is where all the code is defined.

For a truly custom site, as part of Drutopia's *platform as a service* or on your own server, you could have your site's repository available as a `drutopia_version` within the host definition.

Here is the documentation for Drutopia's private repository. This is for example only (or Drutopia maintainers) as the private hosting list of sites is indeed private.

32.1 Setup

In order to run builds for the official server(s):

- If not already, clone `drutopia_host`
- Clone this project into a subfolder of `drutopia_host`
- Also clone the `build_artifacts` site under the `drutopia_host` project

The above can be done with these commands:

```
git clone git@gitlab.com:drutopia-platform/drutopia_host.git
cd drutopia_host
git clone git@gitlab.com:drutopia-platform/hosting_private.git
git clone git@gitlab.com:drutopia-platform/build_artifacts.git
```

The contents of `drutopia_host` should look like this:

```
ansible.cfg
build_artifacts
host_vars
hosting_private
inventory_sample
provision.yml
README.md
roles
```

32.2 Quick start:

You will have a deploy.log created automatically in the current folder. You may wish to occasionally move this out of the way as it will get rather large.

The commands here assume you are in the hosting_private folder is the current working folder.

```
cd hosting_private
```

Ensure you have connectivity and correctly configured your vault key, etc with:

```
ansible -i inventory_live -m ping all
```

Update the entire system (base software, all builds, all sites!). Diff is optional, but I like having it:

```
ansible-playbook -i inventory ../provision.yml --diff
```

Update a single member, using whatever builds already exist on the server:

```
ansible-playbook -i inventory_live ../provision.yml --diff -e
"target_member=family_home_test" --tags=members
```

Individual builds can be updated and pushed to the server. Any builds marked not active will not build, but an individual one can also be targeted (i.e. skip all others) just like target_member:

```
ansible-playbook -i inventory_live ../provision.yml --diff -e
"target_build=stable" --tags="build,push"
```

Configuration checks are performed prior to import. If a configuration was changed on the server, the site update will stop. To force it to continue, either config_import_force the option for the site, or force ALL sites to import config (recommed to use only when combined with target_member). `ansible-playbook -i inventory_live ../provision.yml --diff -e "config_import_force_all=true, target_member=family_home_test" --tags=members`

32.3 Primary tags

- build - perform a local build of a drutopia version
- push - place a build (must be built locally already) on the server. Does not deploy to any site.
- members - configure sites including creating user/database/pushing user source/activating site/performing drupal actions.
- setup - commonly skipped after an initial provisioning, as this wraps all the primary configuration of the server (install apache, mysql, etc) (Add target build)