Drupal Console

hechoendrupal

Published with GitBook

Table of Contents

Drupal Console Documentation

> **Note: This project is a work-in-progress.**

This book documents the Drupal Console project.

Contribute to the project

You can contribute to improve this project on Github

Contribute to this documentation

You can contribute to improve this documentation on GitHub.

Supporting organizations

What is the Drupal Console?

The Drupal Console is a suite of tools run from a command line interface (CLI) to generate boilerplate code and interact with a Drupal 8 installation. From the ground up, it has been built to utilize the same modern PHP practices which were introduced in Drupal 8.

The Drupal Console makes use of the Symfony Console and other third party components which allows you to automatically generate most of the code needed for a Drupal 8 module. In addition, Drupal Console helps you interact with your Drupal installation.

Why should you care about it

Drupal 8 is more technically advanced compared to its predecessor and managing the increasing complexity of Drupal 8 could be a daunting task for anyone. Drupal Console is a suite of tools to help manage that complexity. Writing a Drupal 8 module now involves a lot more boilerplate code and there is a lot you need to know and do just to *get started* building a new module. These tasks can be repetitive and tedious, and can therefore create increased potential for errors. Fortunately, a lot of the new code can be automatically generated, using Drupal Console, without risk of copy/paste errors and with a lot of saved time.

Benefits of the Drupal Console:

- Takes advantage of the Symfony Console and other third-party components to generate PHP, YML, and other files.
- Takes advantage of other modern development practices.
- Saves development time, both during migration of existing Drupal modules and when writing new ones.
- Provides easy-to-learn tools that make Drupal 8 development, by extension, also easier to learn.
- Reduces development time for remaining Drupal 8 tasks and for development of new modules.

Follow along in this documentation as we explore the power of this exciting new set of tools.

How does Drupal Console help?

Generating the code and files required by a Drupal 8 module

Drupal Console provides a number of commands for creating module scaffolding and boilerplate code. For any command, you will be asked a series of questions about what you want to generate. Based on that user interaction, it will then generate the required boilerplate to build the requested component.

Interacting with your Drupal installation.

Drupal Console allows you to interact with your Drupal installation, from rebuilding caches, to listing routes, services, and modules, and interacting with the configuration management.

Learning Drupal 8

Drupal Console helps you learn Drupal 8. In addition to generating complex code, you can increase the verbosity of the code comments, to better understand the generated code and how to build on it, by using the `--learning` option.

Where do I find the project?

Project landing page

http://drupalconsole.com

Github repository

https://github.com/hechoendrupal/DrupalConsole

Documentation

http://hechoendrupal.gitbooks.io/drupal-console

Support chat

https://gitter.im/hechoendrupal/DrupalConsole

More information on Drupal.org project page

https://drupal.org/project/console

Getting the project

There are different ways to get the project on your local machine.

Our recommendation for getting the project on your local machine is by using the installer.

Using the Drupal Console Installer

You can install the Drupal Console locally by running the installer in your project directory, the installer will take care of downloading the necesary files to run drupal console on you computer.

Using curl:

```
$ curl -LSs http://drupalconsole.com/installer | php
```

Or if you don't have curl:

```
$ php -r "readfile('http://drupalconsole.com/installer');" | php
```

The installer script will simply check some php.ini settings, warn you if they are set incorrectly, and then download the latest console.phar in the current directory.

You can now execute console using:

```
$ php console.phar generate:module
```

You can place this file anywhere you wish. If you put it in your PATH, you can access it globally. On unixy systems you can even make it executable and invoke it without php.

Access console from anywhere on your system

```
$ mv console.phar /usr/local/bin/drupal
```

You can now execute console using:

```
$ drupal generate:module
```

NOTE: The name `drupal` is just an alias you can name it anything you like.

Install Drupal Console Using Composer

You can install this project using composer.

Install Drupal Console globally using composer:

```
$ composer global require drupal/console:@stable
```

Add the binary directory to your class path:

```
$ echo "PATH=$PATH:~/.composer/vendor/bin" >> ~/.bash_profile
```

You can now execute console using:

```
$ console generate:module
```

Download phar file

You can download the latest version of Console from the repository releases page at:

https://github.com/hechoendrupal/DrupalConsole

Make sure you download the console.phar file from the most current release.

Update project

Drupal 8 is under heavy development, to keep in sync with the latest changes. The easiest and recommended way of updating Drupal Console is using the self-update command.

Depending on the installation method:

Installed globally (and renamed to "drupal"):

```
$ drupal self-update
```

Installed globally (using composer):

```
$ composer global update drupal/console:@stable
```

Installed locally (running from directory where the console.phar has been downloaded):

```
$ php console.phar self-update
```

Using the project

Drupal Console provides two types of commands, `stand alone` and `container aware` commands.

Stand alone commands: These commands can run outside of a Drupal 8 site root.

Container aware commands: These commands must be run within a Drupal 8 site root.

Executing Drupal Console outside a Drupal site root

You can run Drupal Console form any directory on your system by using the `--root` option to define the Drupal root to be use in the command execution.

```
$ drupal --root=/var/www/drupal8.dev cr all
```

NOTE: Possible messages when executing Drupal Console outside a Drupal site root and no `--root` option provided.

When running the project outside of a Drupal 8 site root, the following message will be shown.

> In order to list all of the available commands, you should run this inside a drupal root directory.

When running the project within of a Drupal 8 site root, but site is not yet installed, the following message will be shown.

> In order to list all of the available commands you should install drupal first.

How to copy configuration files

The first task you should do after installing Drupal Console is to execute the `init` command. Executing this command will copy the project configurations files to your `~/.console/` directory. Overriding values on these copied files is how you can change DrupalConsole behaviour.

```
$ drupal init [--override]
```

Which files are copied when executing the `init` command.

```
~/.console/
├── aliases.yml
├── chain
│   ├── quick-start.yml
│   └── sample.yml
├── config.yml
├── console.rc
├── drupal.fish
└── sites
    └── sample.yml
```

How to download, install and serve Drupal 8

The easiest way to try Drupal 8 in your local machine is by executing the `chain` command and pass the option `--file=~/.console/chain/quick-start.yml` as shown on the following example.

```
$ drupal chain --file=~/.console/chain/quick-start.yml
```

> NOTE: You must execute `drupal init` before in order to copy the `~/.console/chain/quick-start.yml` on your system.

The `chain` command helps you to automate command execution, allowing you to define an external YAML file containing the definition name, option and arguments of several commands and execute that list based on the sequence defined in the file.

The content of the provided `~/.console/chain/quick-start.yml` file is:

```
commands:
  - command: site:new
    arguments:
      site-name: drupal8.dev
      version: 8.0.0
  - command: site:install
    options:
        root: /Users/jmolivas/develop/drupal/sites/drupal8.dev
        langcode: en
        db-type: sqlite
        db-file: sites/default/files/.ht.sqlite
        site-name: 'Drupal 8 Quick Start'
        site-mail: admin@example.com
        account-name: admin
        account-mail: admin@example.com
        account-pass: admin
        generate-inline: true
    arguments:
        profile: standard
  - command: server
```

The previous configuration will execute several commands, in this case commands that will download and install Drupal using SQLite, and finally start the PHP's built in server, now you only need to open your browser and point it to 127.0.0.1:8088.

You can duplicate or make changes on the provided YAML file, to add commands for download modules `module:download`, install modules `module:install` , import configurations `config:import` and restore your database `database:restore` or any other command provided by DrupalConsole or a custom command by your own module.

How to use Drupal Console in a multi-site installation

Drupal Console provides support for Drupal multi-site installations. This project provides the `multisite:debug` command to debug multi-site installations and the `--uri` option to interact with multi-site installations.

How to list all known multi sites

```
$ drupal multisite:debug

+---------------------+--------------------------------+
| Site                | Directory                      |
+---------------------+--------------------------------+
| drupal8.dev         | /var/www/drupal8.dev/default   |
| drupal8.multi.dev   | /var/www/drupal8.dev/multi.dev |
+---------------------+--------------------------------+

 Sites are written using the format: <port>.<domain>.<path>
```

How to execute a command against a multi-site installation

```
$ drupal --uri=http://drupal8.multi.dev cr all
```

How to use Drupal Console in a remote site installation

Drupal Console allows you to run commands on your local server but actually execute them on a remote server.

Setting up your local computer to use a remote site requires a little configuration.

Edit global configuration

You can provide global configuration to remote connections at the copied file `~/.console/config.yml`. This information is grouped within the `remote` key.

```
application:
  ...
  remote:
    user: root
    port: 22
    console: /usr/local/bin/drupal
    options:
    arguments:
    keys:
      public: ~/.ssh/id_rsa.pub
      private: ~/.ssh/id_rsa
      passphrase: ~/.ssh/passphrase.txt
```

Edit specific site configuration

You can provide specific site configuration by duplicating the copied site file at `~/.console/sites/sample.yml` with a new same at `~/.console/sites/`.

```
local:
  root: /var/www/drupal8.dev
  host: local
dev:
  root: /var/www/html/drupal
  host: 140.211.10.62
  user: drupal
prod:
  root: /var/www/html/docroot
  host: live.drupal.org
  user: drupal
  console: /var/www/html/docroot/console.phar
```

Debug sites.

You can list all known local and remote sites by executing the `site:debug` command.

```
$ drupal site:debug

+--------------------+-----------------+------------------------+
| Site               | Host            | Root                   |
+--------------------+-----------------+------------------------+
| sample.local       | local           | /var/www/drupal8.dev   |
| sample.dev         | 140.211.10.62   | /var/www/html/drupal   |
| sample.prod        | live.drupal.org | /var/www/html/docroot  |
+--------------------+-----------------+------------------------+
```

You can show the site configuration details by passing the site name as argument to the `site:debug` command.

```
$ drupal site:debug sample.dev

user: drupal
port: 22
console: /usr/local/bin/drupal
options:
arguments:
keys:
    public: ~/.ssh/id_rsa.pub
    private: ~/.ssh/id_rsa
    passphrase: ~/.ssh/passphrase.txt
root: /var/www/html/drupal
host: 140.211.10.62
remote: true
```

NOTE: As you may notice the global configuration and the specific site configuration are merged when debugging a site. You can override any global configuration by adding those keys on the site specific configuration.

Available Drupal Console Commands

Note: Drupal Console commands *must* be run from the root of a Drupal 8 installation.

Drupal Console Command	Details
about	Display basic information about Drupal Console project
chain	Chain command execution
drush	Run drush from console.
help	Displays help for a command
init	Copy configuration files to user home directory.
list	Lists commands22
self-update	Update the console to latest version.
server	Runs PHP built-in web server
cache	
cache:rebuild	Rebuild and clear all site caches.
config	
config:debug	Show the current configuration.
config:edit	Edit the selected configuration.
config:export	Export current application configuration.
config:export:content:type	Export a specific content type and their fields.
config:export:single	Export single configuration as yml file.
config:export:view	commands.config.export.view.description
config:import	Import configuration to current application.
config:import:single	Import the selected configuration.
config:override	Override config value in active configuration.
container	
container:debug	Displays current services for an application.
cron	
cron:debug	List of modules implementing a cron
cron:execute	Execute cron implementations by module or execute all crons

cron:release	Release cron system lock to run cron again
database	
database:client	Launch a DB client if it's available
database:connect	Launch a DB client if it's available
database:dump	Dump structure and contents of MySQL databases and tables
database:log:clear	Remove events from DBLog table, filters are available
database:log:debug	Display current log events for the application
database:restore	Restore structure and contents of MySQL databases and tables
generate	
generate:authentication:provider	Generate an Authentication Provider
generate:command	Generate commands for the console.
generate:controller	Generate & Register a controller
generate:doc:dash	Generate the DrupalConsole.docset package for Dash
generate:doc:gitbook	Generate documentations for Commands
generate:entity:bundle	Generate a new content type (node / entity bundle)
generate:entity:config	Generate a new config entity
generate:entity:content	Generate a new content entity
generate:event:subscriber	Generate an event subscriber
generate:form	Generate a new "FormBase"
generate:form:alter	Generate an implementation of hook_form_alter() or hook_form_FORM_ID_alter
generate:form:config	Generate a new "ConfigFormBase"
generate:module	Generate a module.
generate:permissions	Generate module permissions
generate:plugin:block	Generate a plugin block
generate:plugin:condition	Generate a plugin condition.
generate:plugin:field	Generate field type, widget and formatter plugins.
generate:plugin:fieldformatter	Generate field formatter plugin.
generate:plugin:fieldtype	Generate field type plugin.
generate:plugin:fieldwidget	Generate field widget plugin.

generate:plugin:imageeffect	Generate image effect plugin.
generate:plugin:imageformatter	Generate image formatter plugin.
generate:plugin:rest:resource	Generate plugin rest resource
generate:plugin:rulesaction	Generate a plugin rule action
generate:plugin:type:annotation	Generate a plugin type with annotation discovery
generate:plugin:type:yaml	Generate a plugin type with Yaml discovery
generate:plugin:views:field	Generate a custom plugin view field.
generate:routesubscriber	Generate a RouteSubscriber
generate:service	Generate service
generate:theme	Generate a theme.
locale	
locale:language:add	Add a language to be supported by your site
locale:language:delete	Delete a language to be supported by your site
locale:translation:status	List available translation updates
migrate	
migrate:debug	Display current migration available for the application
migrate:execute	Execute a migration available for application
migrate:setup	Load and create the relevant migrations for a provided legacy database
module	
module:debug	Display current modules available for application
module:download	Download module or modules in application
module:install	Install module or modules in the application
module:uninstall	Uninstall module or modules in the application
multisite	
multisite:debug	List all multisites available in system
rest	
rest:debug	Display current rest resource for the application
rest:disable	Disable a rest resource for the application
rest:enable	Enable a rest resource for the application
router	

router:debug	Displays current routes for the application
router:rebuild	Rebuild routes for the application
site	
site:debug	List all known local and remote sites.
site:install	Install a Drupal project
site:maintenance	Switch site into maintenance mode
site:mode	Switch system performance configuration
site:new	Create a new Drupal project
site:status	View current Drupal Installation status
test	
test:debug	List Test Units available for the application.
test:run	Run Test unit from tests available for application
theme	
theme:debug	Displays current themes for the application
theme:download	Install theme or themes in the application
theme:install	Install theme or themes in the application
theme:uninstall	Uninstall theme or themes in the application
update	
update:debug	Display current updates available for the application
update:execute	Execute a specific Update N function in a module, or execute all
user	
user:login:clear:attempts	Clear login failed attempts for an account.
user:login:url	Returns a one-time user login url.
user:password:hash	Generate a hash from a plaintext password.
user:password:reset	Reset password for a specific user.
views	
views:debug	Display current views resources for the application
views:disable	Disable a View
views:enable	Enable a View
yaml	
	Compare two YAML files in order to find differences

	between them.
yaml:merge	Merge one or more YAML files in a new YAML file. Latest values are preserved.
yaml:split	Split a YAML file using indent as separator criteria
yaml:update:key	Replace a YAML key in a YAML file.
yaml:update:value	Update a value for a specific key in a YAML file.

Available options

Option	Details
--help	Display this help message
--quiet	Do not output any message
--verbose	Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
--version	Display this application version
--ansi	Force ANSI output
--no-ansi	Disable ANSI output
--no-interaction	Do not ask any interactive question
--root	Define the Drupal root to be use in command execution
--shell	Launch the shell.
--env	The Environment name.
--no-debug	Switches off debug mode.
--learning	Generate a verbose code output.
--generate-chain	Print execution options and arguments as yaml output to be used in chain command
--generate-inline	Print execution options and arguments as inline call to be use in the future
--generate-doc	application.console.arguments.generate-doc
--target	application.console.arguments.target
--uri	URI of the Drupal site to use (for multisite environments or when running on an alternate port)

Available arguments

Argument	Details
command	The command to execute

about

The **about** command Display basic information about Drupal Console project

Usage:

```
$ drupal about
```

chain

The **chain** command Chain command execution

Usage:

```
$ drupal chain [options]
```

Available options

Option	Details
--file	User defined file containing commands to get executed.

drush

The **drush** command Run drush from console.

Usage:

```
$ drupal drush [arguments]
```

Available arguments

Argument	Details
args	Drush arguments.

help

The **help** command Displays help for a command

Usage:

```
$ drupal help [arguments] [options]
```

Available options

Option	Details
--xml	To output help as XML
--format	The output format (txt, xml, json, or md)
--raw	To output raw command help

Available arguments

Argument	Details
command_name	The command name

init

The **init** command Copy configuration files to user home directory.

Usage:

```
$ drupal init [options]
```

Available options

Option	Details
--override	Override configurations files

list

The **list** command Lists commands22

Usage:

```
$ drupal list [arguments] [options]
```

Available options

Option	Details
--xml	To output list as XML
--raw	To output raw command list
--format	The output format (txt, xml, json, or md)

Available arguments

Argument	Details
namespace	The namespace name

self-update

The **self-update** command Update the console to latest version.

Usage:

```
$ drupal self-update
```

server

The **server** command Runs PHP built-in web server

Usage:

```
$ drupal server [arguments]
```

Available arguments

Argument	Details
address	The address:port values

Examples

- Run using default address argument value 127.0.0.1.8088

  ```
  $ drupal server
  ```

- Passing address argument to use a different port number

  ```
  $ drupal server 127.0.0.1:8089
  ```

- Running default address argument values, using --root option to define the Drupal root

  ```
  $ drupal --root=/var/www/drupal8.dev server
  ```

cache:rebuild

The **cache:rebuild** command Rebuild and clear all site caches.

Usage:

```
$ drupal cache:rebuild [arguments]
```

Available arguments

Argument	Details
cache	Only clear a specific cache.

Examples

- Rebuild all caches

  ```
  $ drupal cr all
  ```

- Rebuild discovery cache

  ```
  $ drupal cr discovery
  ```

config:debug

The **config:debug** command Show the current configuration.

Usage:

```
$ drupal config:debug [arguments]
```

Available arguments

Argument	Details
config-name	Configuration name.

config:edit

The **config:edit** command Edit the selected configuration.

Usage:

```
$ drupal config:edit [arguments]
```

Available arguments

Argument	Details
config-name	Configuration name.
editor	Editor.

config:export

The **config:export** command Export current application configuration.

Usage:

```
$ drupal config:export [options]
```

Available options

Option	Details
--directory	commands.config.export.options.directory
--tar	commands.config.export.options.tar

config:export:content:type

The **config:export:content:type** command Export a specific content type and their fields.

Usage:

```
$ drupal config:export:content:type [arguments] [options]
```

Available options

Option	Details
--module	The Module name.
--optional-config	Export content type as an optional YAML configuration in your module

Available arguments

Argument	Details
content_type	Content Type to be exported

config:export:single

The **config:export:single** command Export single configuration as yml file.

Usage:

```
$ drupal config:export:single [arguments]
```

Available arguments

Argument	Details
config-name	Configuration name.
directory	Define export directory to save configuration output.

config:export:view

The **config:export:view** command commands.config.export.view.description

Usage:

```
$ drupal config:export:view [arguments] [options]
```

Available options

Option	Details
--module	The Module name.
--optional-config	commands.config.export.view.options.optional-config
--include-module-dependencies	commands.config.export.view.options.include-module-dependencies

Available arguments

Argument	Details
view-id	commands.config.export.view.arguments.view-id

config:import

The **config:import** command Import configuration to current application.

Usage:

```
$ drupal config:import [options]
```

Available options

Option	Details
--file	commands.config.import.option.file
--remove-files	commands.config.import.option.keep-files

config:import:single

The **config:import:single** command Import the selected configuration.

Usage:

```
$ drupal config:import:single [arguments]
```

Available arguments

Argument	Details
config-name	Configuration name.
input-file	Path to the import files.

config:override

The **config:override** command Override config value in active configuration.

Usage:

```
$ drupal config:override [arguments]
```

Available arguments

Argument	Details
config-name	Configuration name.
key	Key
value	Value

container:debug

The **container:debug** command Displays current services for an application.

Usage:

```
$ drupal container:debug
```

cron:debug

The **cron:debug** command List of modules implementing a cron

Usage:

```
$ drupal cron:debug
```

cron:execute

The **cron:execute** command Execute cron implementations by module or execute all crons

Usage:

```
$ drupal cron:execute [arguments]
```

Available arguments

Argument	Details
module	The Module name.

cron:release

The **cron:release** command Release cron system lock to run cron again

Usage:

```
$ drupal cron:release
```

database:client

The **database:client** command Launch a DB client if it's available

Usage:

```
$ drupal database:client [arguments]
```

Available arguments

Argument	Details
database	Database key from settings.php

database:connect

The **database:connect** command Launch a DB client if it's available

Usage:

```
$ drupal database:connect [arguments]
```

Available arguments

Argument	Details
database	Database key from settings.php

database:dump

The **database:dump** command Dump structure and contents of MySQL databases and tables

Usage:

```
$ drupal database:dump [arguments] [options]
```

Available options

Option	Details
--file	commands.database.dump.option.file

Available arguments

Argument	Details
database	Database key from settings.php

database:log:clear

The **database:log:clear** command Remove events from DBLog table, filters are available

Usage:

```
$ drupal database:log:clear [arguments] [options]
```

Available options

Option	Details
--type	commands.database.log.clear.options.type
--severity	commands.database.log.clear.options.severity
--user-id	commands.database.log.clear.options.user-id

Available arguments

Argument	Details
event-id	commands.database.log.clear.arguments.event-id

database:log:debug

The **database:log:debug** command Display current log events for the application

Usage:

```
$ drupal database:log:debug [arguments] [options]
```

Available options

Option	Details
--type	Filter events by a specific type
--severity	Filter events by a specific level of severity
--user-id	Filter events by a specific user id
--limit	Limit results to a specific number
--offset	Starting point of a limit

Available arguments

Argument	Details
event-id	DBLog event ID

database:restore

The **database:restore** command Restore structure and contents of MySQL databases and tables

Usage:

```
$ drupal database:restore [arguments] [options]
```

Available options

Option	Details
--file	The filename for your database backup file

Available arguments

Argument	Details
database	Database key from settings.php

generate:authentication:provider

The **generate:authentication:provider** command Generate an Authentication Provider

Usage:

```
$ drupal generate:authentication:provider [options]
```

Available options

Option	Details
--module	The Module name.
--class	Authentication Provider class
--provider-id	Provider ID

generate:command

The **generate:command** command Generate commands for the console.

Usage:

```
$ drupal generate:command [options]
```

Available options

Option	Details
--module	The Module name.
--class	The Class that describes the command. (Must end with the word 'Commmand').
--name	The Command name.
--container-aware	Is the command aware of the drupal site installation when executed

generate:controller

The **generate:controller** command Generate & Register a controller

Usage:

```
$ drupal generate:controller [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Controller Class name
--controller-title	Title of the controller
--method-name	The action method name
--route	The route path
--services	Load services from the container.
--test	Generate a test class

generate:doc:dash

The **generate:doc:dash** command Generate the DrupalConsole.docset package for Dash

Usage:

```
$ drupal generate:doc:dash [options]
```

Available options

Option	Details
--path	The path to the directory where the docset will be saved.

generate:doc:gitbook

The **generate:doc:gitbook** command Generate documentations for Commands

Usage:

```
$ drupal generate:doc:gitbook [arguments] [options]
```

Available options

Option	Details
--path	The path to render the documentation
--help	Display this help message
--quiet	Do not output any message
--verbose	Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
--version	Display this application version
--ansi	Force ANSI output
--no-ansi	Disable ANSI output
--no-interaction	Do not ask any interactive question
--root	Define the Drupal root to be use in command execution
--shell	Launch the shell.
--env	The Environment name.
--no-debug	Switches off debug mode.
--learning	Generate a verbose code output.
--generate-chain	Print execution options and arguments as yaml output to be used in chain command
--generate-inline	Print execution options and arguments as inline call to be use in the future
--generate-doc	application.console.arguments.generate-doc
--target	application.console.arguments.target
--uri	URI of the Drupal site to use (for multisite environments or when running on an alternate port)

Available arguments

Argument	Details
command	The command to execute

generate:entity:bundle

The **generate:entity:bundle** command Generate a new content type (node / entity bundle)

Usage:

```
$ drupal generate:entity:bundle [options]
```

Available options

Option	Details
--module	The Module name.
--bundle-name	The content type's machine name
--bundle-title	The content type's human-readable name

generate:entity:config

The **generate:entity:config** command Generate a new config entity

Usage:

```
$ drupal generate:entity:config [options]
```

Available options

Option	Details
--module	The Module name.
--entity-class	The config entity class
--entity-name	The config entity name
--label	The label

generate:entity:content

The **generate:entity:content** command Generate a new content entity

Usage:

```
$ drupal generate:entity:content [options]
```

Available options

Option	Details
--module	The Module name.
--entity-class	The content entity class
--entity-name	The content entity name
--label	The label

generate:event:subscriber

The **generate:event:subscriber** command Generate an event subscriber

Usage:

```
$ drupal generate:event:subscriber [options]
```

Available options

Option	Details
--module	The Module name.
--name	commands.generate.service.options.name
--class	commands.generate.service.options.class
--events	Load services from the container.
--services	Load services from the container.

generate:form

The **generate:form** command Generate a new "FormBase"

Usage:

```
$ drupal generate:form [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	The form class name
--form-id	The Form id
--services	Load services from the container.
--inputs	Create inputs in a form.
--routing	Update routing

generate:form:alter

The **generate:form:alter** command Generate an implementation of hook_form_alter() or hook_form_FORM_ID_alter

Usage:

```
$ drupal generate:form:alter [options]
```

Available options

Option	Details
--module	The Module name.
--form-id	Form ID to alter
--inputs	Create inputs in a form.

generate:form:config

The **generate:form:config** command Generate a new "ConfigFormBase"

Usage:

```
$ drupal generate:form:config [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	The form class name
--form-id	The Form id
--services	Load services from the container.
--inputs	Create inputs in a form.
--routing	Update routing

generate:module

The **generate:module** command Generate a module.

Usage:

```
$ drupal generate:module [options]
```

Available options

Option	Details
--module	The Module name
--machine-name	The machine name (lowercase and underscore only)
--module-path	The path of the module
--description	Module description
--core	Core version
--package	Module package
--feature	Set module compatible with Features module
--composer	Add a composer.json file
--dependencies	Module dependencies separated by commas (i.e. context, panels)

generate:permissions

The **generate:permissions** command Generate module permissions

Usage:

```
$ drupal generate:permissions [options]
```

Available options

Option	Details
--module	The Module name.
--permissions	Create permissions.

generate:plugin:block

The **generate:plugin:block** command Generate a plugin block

Usage:

```
$ drupal generate:plugin:block [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin class name
--label	Plugin label
--plugin-id	Plugin id
--theme-region	Theme region to render Plugin Block
--inputs	Create inputs in a form.
--services	Load services from the container.

generate:plugin:condition

The **generate:plugin:condition** command Generate a plugin condition.

Usage:

```
$ drupal generate:plugin:condition [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin condition class name
--label	Plugin condition label
--plugin-id	Plugin condition id
--context-definition-id	Context definition ID
--context-definition-label	Context definition label
--context-definition-required	Context definition is requiered (TRUE/FALSE)

generate:plugin:field

The **generate:plugin:field** command Generate field type, widget and formatter plugins.

Usage:

```
$ drupal generate:plugin:field [options]
```

Available options

Option	Details
--module	The Module name.
--type-class-name	Field type plugin class name
--type-label	Field type plugin label
--type-plugin-id	Field type plugin id
--type-description	commands.generate.plugin.field.options.type-type-description
--formatter-class-name	commands.generate.plugin.field.options.class-name
--formatter-label	Field formatter plugin label
--formatter-plugin-id	Field formatter plugin id
--widget-class-name	Field formatter plugin class name
--widget-label	Field widget plugin label
--widget-plugin-id	Field widget plugin id
--field-type	Field type the formatter and widget plugin can be used with
--default-widget	Default field widget of the field type plugin
--default-formatter	Default field formatter of field type plugin

generate:plugin:fieldformatter

The **generate:plugin:fieldformatter** command Generate field formatter plugin.

Usage:

```
$ drupal generate:plugin:fieldformatter [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin class name
--label	Plugin label
--plugin-id	Plugin id
--field-type	Field type the plugin can be used with

generate:plugin:fieldtype

The **generate:plugin:fieldtype** command Generate field type plugin.

Usage:

```
$ drupal generate:plugin:fieldtype [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin class name
--label	Plugin label
--plugin-id	Plugin id
--description	Plugin Description
--default-widget	Default field widget of this plugin
--default-formatter	Default field formatter of this plugin

generate:plugin:fieldwidget

The **generate:plugin:fieldwidget** command Generate field widget plugin.

Usage:

```
$ drupal generate:plugin:fieldwidget [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin class name
--label	Plugin label
--plugin-id	Plugin id
--field-type	Field type the plugin can be used with

generate:plugin:imageeffect

The **generate:plugin:imageeffect** command Generate image effect plugin.

Usage:

```
$ drupal generate:plugin:imageeffect [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin class name
--label	Plugin label
--plugin-id	Plugin id
--description	Plugin Description

generate:plugin:imageformatter

The **generate:plugin:imageformatter** command Generate image formatter plugin.

Usage:

```
$ drupal generate:plugin:imageformatter [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin class name
--label	Plugin label
--plugin-id	Plugin id

generate:plugin:rest:resource

The **generate:plugin:rest:resource** command Generate plugin rest resource

Usage:

```
$ drupal generate:plugin:rest:resource [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin Rest Resource class
--name	commands.generate.service.options.name
--plugin-id	Plugin Rest Resource id
--plugin-label	Plugin Rest Resource Label
--plugin-url	Plugin Rest Resource URL
--plugin-states	Plugin Rest Resource States

generate:plugin:rulesaction

The **generate:plugin:rulesaction** command Generate a plugin rule action

Usage:

```
$ drupal generate:plugin:rulesaction [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin class name
--label	Plugin label
--plugin-id	Plugin id
--type	Action Type (user or node)
--category	Plugin category
--context	Plugin context

generate:plugin:type:annotation

The **generate:plugin:type:annotation** command Generate a plugin type with annotation discovery

Usage:

```
$ drupal generate:plugin:type:annotation [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin type class name
--machine-name	commands.generate.plugin.type.annotation.options.plugin-id
--label	Plugin type label

generate:plugin:type:yaml

The **generate:plugin:type:yaml** command Generate a plugin type with Yaml discovery

Usage:

```
$ drupal generate:plugin:type:yaml [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Plugin type class name
--plugin-name	Plugin type machine name
--plugin-file-name	Plugin file name

generate:plugin:views:field

The **generate:plugin:views:field** command Generate a custom plugin view field.

Usage:

```
$ drupal generate:plugin:views:field [options]
```

Available options

Option	Details
--module	The Module name.
--class-name	Views plugin field class name
--title	Views plugin field title
--description	Views plugin field description

generate:routesubscriber

The **generate:routesubscriber** command Generate a RouteSubscriber

Usage:

```
$ drupal generate:routesubscriber [options]
```

Available options

Option	Details
--module	The Module name.
--name	Service name
--class	Class name

generate:service

The **generate:service** command Generate service

Usage:

```
$ drupal generate:service [options]
```

Available options

Option	Details
--module	The Module name.
--name	commands.generate.service.options.name
--class	commands.generate.service.options.class
--interface	commands.common.service.options.interface
--services	Load services from the container.

generate:theme

The **generate:theme** command Generate a theme.

Usage:

```
$ drupal generate:theme [options]
```

Available options

Option	Details
--theme	commands.generate.theme.options.module
--machine-name	The machine name (lowercase and underscore only)
--theme-path	commands.generate.theme.options.module-path
--description	Theme description
--core	Core version
--package	Theme package
--global-library	Global styling library name
--base-theme	Base theme (i.e. classy, seven)
--regions	Regions
--breakpoints	Breakpoints

locale:language:add

The **locale:language:add** command Add a language to be supported by your site

Usage:

```
$ drupal locale:language:add [arguments]
```

Available arguments

Argument	Details
language	Language for instance es or Spanish

locale:language:delete

The **locale:language:delete** command Delete a language to be supported by your site

Usage:

```
$ drupal locale:language:delete [arguments]
```

Available arguments

Argument	Details
language	Language for instance es or Spanish

locale:translation:status

The **locale:translation:status** command List available translation updates

Usage:

```
$ drupal locale:translation:status [arguments]
```

Available arguments

Argument	Details
language	Language for instance es or Spanish

migrate:debug

The **migrate:debug** command Display current migration available for the application

Usage:

```
$ drupal migrate:debug [arguments]
```

Available arguments

Argument	Details
tag	Migrate tag

migrate:execute

The **migrate:execute** command Execute a migration available for application

Usage:

```
$ drupal migrate:execute [arguments] [options]
```

Available options

Option	Details
--site-url	Site Source URL
--db-type	commands.migrate.setup.migrations.options.db-type
--db-host	Database Host
--db-name	Database Name
--db-user	Database User
--db-pass	Database Pass
--db-prefix	Database Prefix
--db-port	Database Port
--exclude	Migration id(s) to exclude

Available arguments

Argument	Details
migration-ids	Migration id(s)

migrate:setup

The **migrate:setup** command Load and create the relevant migrations for a provided legacy database

Usage:

```
$ drupal migrate:setup [options]
```

Available options

Option	Details
--db-type	Drupal Database type
--db-host	Database Host
--db-name	Database Name
--db-user	Database User
--db-pass	Database Pass
--db-prefix	Database Prefix
--db-port	Database Port

module:debug

The **module:debug** command Display current modules available for application

Usage:

```
$ drupal module:debug [options]
```

Available options

Option	Details	
--status	Module status [enabled	disabled]
--type	Module type [core	no-core]

module:download

The **module:download** command Download module or modules in application

Usage:

```
$ drupal module:download [arguments]
```

Available arguments

Argument	Details
module	commands.module.download.options.module
version	Module version i.e 1.x-dev

module:install

The **module:install** command Install module or modules in the application

Usage:

```
$ drupal module:install [arguments] [options]
```

Available options

Option	Details
--overwrite-config	Overwrite configuration active if necessary

Available arguments

Argument	Details
module	Module or modules to be enabled should be separated by a comma

module:uninstall

The **module:uninstall** command Uninstall module or modules in the application

Usage:

```
$ drupal module:uninstall [arguments]
```

Available arguments

Argument	Details
module	Module or modules to be uninstalled should be separated by a comma

multisite:debug

The **multisite:debug** command List all multisites available in system

Usage:

```
$ drupal multisite:debug
```

rest:debug

The **rest:debug** command Display current rest resource for the application

Usage:

```
$ drupal rest:debug [arguments] [options]
```

Available options

Option	Details	
--authorization	Rest resource status enabled	disabled

Available arguments

Argument	Details
resource-id	Rest ID

rest:disable

The **rest:disable** command Disable a rest resource for the application

Usage:

```
$ drupal rest:disable [arguments]
```

Available arguments

Argument	Details
resource-id	Rest ID

rest:enable

The **rest:enable** command Enable a rest resource for the application

Usage:

```
$ drupal rest:enable [arguments]
```

Available arguments

Argument	Details
resource-id	Rest ID

router:debug

The **router:debug** command Displays current routes for the application

Usage:

```
$ drupal router:debug [arguments]
```

Available arguments

Argument	Details
route-name	Route names

router:rebuild

The **router:rebuild** command Rebuild routes for the application

Usage:

```
$ drupal router:rebuild
```

site:debug

The **site:debug** command List all known local and remote sites.

Usage:

```
$ drupal site:debug [arguments]
```

Available arguments

Argument	Details
target	commands.site.debug.options.target

site:install

The **site:install** command Install a Drupal project

Usage:

```
$ drupal site:install [arguments] [options]
```

Available options

Option	Details
--langcode	Drupal language
--db-type	Drupal Database type to be use in install
--db-file	Drupal Database file to be use in install
--db-host	Database Host
--db-name	Database Name
--db-user	Database User
--db-pass	Database Pass
--db-prefix	Database Prefix
--db-port	Database Port
--site-name	Drupal site name
--site-mail	Drupal site mail
--account-name	Drupal administrator account name
--account-mail	Drupal administrator account mail
--account-pass	Drupal administrator account password

Available arguments

Argument	Details
profile	Drupal Profile to be install

site:maintenance

The **site:maintenance** command Switch site into maintenance mode

Usage:

```
$ drupal site:maintenance [arguments]
```

Available arguments

Argument	Details
mode	Site maintenance mode[on/off]

site:mode

The **site:mode** command Switch system performance configuration

Usage:

```
$ drupal site:mode [arguments]
```

Available arguments

Argument	Details
environment	Environment name [dev, prod]

site:new

The **site:new** command Create a new Drupal project

Usage:

```
$ drupal site:new [arguments]
```

Available arguments

Argument	Details
site-name	Site name
version	Specific Drupal version to download

site:status

The **site:status** command View current Drupal Installation status

Usage:

```
$ drupal site:status [options]
```

Available options

Option	Details
--format	commands.site.status.options.format

test:debug

The **test:debug** command List Test Units available for the application.

Usage:

```
$ drupal test:debug [arguments] [options]
```

Available options

Option	Details
--group	Group

Available arguments

Argument	Details
test-class	Test Class

test:run

The **test:run** command Run Test unit from tests available for application

Usage:

```
$ drupal test:run [arguments] [options]
```

Available options

Option	Details
--url	commands.test.run.arguments.url

Available arguments

Argument	Details
test-class	Test Class

theme:debug

The **theme:debug** command Displays current themes for the application

Usage:

```
$ drupal theme:debug [arguments]
```

Available arguments

Argument	Details
theme	Specific theme to debug

theme:download

The **theme:download** command Install theme or themes in the application

Usage:

```
$ drupal theme:download [arguments]
```

Available arguments

Argument	Details
theme	theme or themes to be installed should be separated by a comma
version	Theme version i.e 1.x-dev

theme:install

The **theme:install** command Install theme or themes in the application

Usage:

```
$ drupal theme:install [arguments] [options]
```

Available options

Option	Details
--set-default	Set theme as default theme

Available arguments

Argument	Details
theme	commands.theme.install.options.module

theme:uninstall

The **theme:uninstall** command Uninstall theme or themes in the application

Usage:

```
$ drupal theme:uninstall [arguments]
```

Available arguments

Argument	Details
theme	commands.theme.uninstall.options.module

update:debug

The **update:debug** command Display current updates available for the application

Usage:

```
$ drupal update:debug
```

update:execute

The **update:execute** command Execute a specific Update N function in a module, or execute all

Usage:

```
$ drupal update:execute [arguments]
```

Available arguments

Argument	Details
module	The Module name.
update-n	Specific Update N function to be executed

user:login:clear:attempts

The **user:login:clear:attempts** command Clear login failed attempts for an account.

Usage:

```
$ drupal user:login:clear:attempts [arguments]
```

Available arguments

Argument	Details
uid	User ID.

user:login:url

The **user:login:url** command Returns a one-time user login url.

Usage:

```
$ drupal user:login:url [arguments]
```

Available arguments

Argument	Details
user-id	User ID.

user:password:hash

The **user:password:hash** command Generate a hash from a plaintext password.

Usage:

```
$ drupal user:password:hash [arguments]
```

Available arguments

Argument	Details
password	Password(s) in text format

user:password:reset

The **user:password:reset** command Reset password for a specific user.

Usage:

```
$ drupal user:password:reset [arguments]
```

Available arguments

Argument	Details
user	User ID
password	Password in text format

views:debug

The **views:debug** command Display current views resources for the application

Usage:

```
$ drupal views:debug [arguments] [options]
```

Available options

Option	Details	
--tag	View tag	
--status	View status (Enabled	Disabled)

Available arguments

Argument	Details
view-id	View ID

views:disable

The **views:disable** command Disable a View

Usage:

```
$ drupal views:disable [arguments]
```

Available arguments

Argument	Details
view-id	View ID

views:enable

The **views:enable** command Enable a View

Usage:

```
$ drupal views:enable [arguments]
```

Available arguments

Argument	Details
view-id	View ID

yaml:diff

The **yaml:diff** command Compare two YAML files in order to find differences between them.

Usage:

```
$ drupal yaml:diff [arguments] [options]
```

Available options

Option	Details
--stats	Print statistics about YAML files comparation
--negate	Define mode diff or equal comparation, possible values TRUE/FALSE or 0/1
--limit	Limit results to a specific number
--offset	Starting point of a limit

Available arguments

Argument	Details
yaml-left	YAML file used as base to compare
yaml-right	YAML file used to find missing parts or differences with the base YAML file

yaml:merge

The **yaml:merge** command Merge one or more YAML files in a new YAML file. Latest values are preserved.

Usage:

```
$ drupal yaml:merge [arguments]
```

Available arguments

Argument	Details
yaml-destination	Path of new YAML file to store the result of merge.
yaml-files	Path of YAML files to merge

yaml:split

The **yaml:split** command Split a YAML file using indent as separator criteria

Usage:

```
$ drupal yaml:split [arguments] [options]
```

Available options

Option	Details
--indent-level	Split YAML file using a specific indent level
--file-output-prefix	commands.yaml.split.options.file-output-prefix
--file-output-suffix	commands.yaml.split.options.file-output-suffix
--starting-key	YAML Key from where start split useful to extract partial elements
--exclude-parents-key	Exclude parents key in file name generated

Available arguments

Argument	Details
yaml-file	commands.yaml.split.value.arguments.yaml-file

yaml:update:key

The **yaml:update:key** command Replace a YAML key in a YAML file.

Usage:

```
$ drupal yaml:update:key [arguments]
```

Available arguments

Argument	Details
yaml-file	Path of YAML file to update
yaml-key	YAML key to update
yaml-new-key	commands.yaml.update.value.arguments.yaml-new-key

yaml:update:value

The **yaml:update:value** command Update a value for a specific key in a YAML file.

Usage:

```
$ drupal yaml:update:value [arguments]
```

Available arguments

Argument	Details
yaml-file	Path of YAML file to update
yaml-key	YAML key to update
yaml-value	YAML value to update

Contributing new features

If you create a new custom command or something else which would be useful for *other* Drupal installations, please consider forking the Drupal Console project on GitHub. Then just create an issue for a "feature request", put your work in a new Git branch on your fork, publish your branch and add a pull request on GitHub (including your issue ID in the PR title). The Drupal Console maintainers are very happy to accept useful contributions and usually do so quite promptly.

Getting support with becoming a contributor

If you want to contribute to Drupal Console and have any difficulty getting oriented to the process, you can find Instant Messaging support on Gitter IM.

Standard GitHub Workflow

If you haven't yet contributed to a project on GitHub, or are not sure you remember the workflow, you might first want to read the documentation about pull requests. You may also wish to download the GitHub application (Mac | Windows, which simplifies the workflow a bit and provides a nice GUI for your contributions).

Github-flavored Markdown

This documentation is written in “Github-flavored Markdown”, which is rendered on Github, directly, as HTML. If you are *already* familiar with Markdown, GFL only adds a few useful tweaks, otherwise you may want to read Github’s Markdown Basics primer.

Project requirements

Download Git

We recommend downloading Git from http://git-scm.com/downloads

Download Composer

Run this in your terminal to get the latest Composer version:

```
curl -sS https://getcomposer.org/installer | php
```

Or if you don't have curl:

```
php -r "readfile('https://getcomposer.org/installer');" | php
```

This installer script will simply check some php.ini settings, warn you if they are set incorrectly, and then download the latest composer.phar in the current directory

You can run this terminal command to make Composer easily accessible, from anywhere on your system:

```
$ mv composer.phar /usr/local/bin/composer
```

Download Drupal 8

The Drupal Console project only supports Drupal 8; which you will need to download and install locally.

Download Drupal

```
$ drupal site:new drupal8.dev 8.0.0
$ cd drupal8.dev
```

Install Drupal using MySQL:

```
$ drupal site:install standard --langcode=en --db-type=mysql --db-host=127.0.0.1
  --db-name=drupal --db-user=root --db-pass=root --db-port=3306
  --site-name="Drupal 8 Site Install" --site-mail=admin@example.com
  --account-name=admin --account-mail=admin@example.com --account-pass=admin -n
```

Install Drupal using SQLite:

```
$ drupal site:install standard --langcode=en --db-type=sqlite
  --db-file=sites/default/files/.ht.sqlite --site-name="Drupal 8 Site Install"
  --site-mail=admin@example.com --account-name=admin --account-mail=admin@example.com
  --account-pass=admin -n
```

Start the PHP's built in server

```
$ drupal server
```

NOTE: Make sure you use your own user and database credentials when running `site:install` and never user root on production. In this example code, we accept all interactive questions, i.e. answering *“yes”* when passing the `-y` argument.

Getting the project

Fork

Fork your own copy of the Console repository to your account

Clone

Get a copy of your recently cloned version of console in your machine.

```
$ git clone git@github.com:[your-git-user-here]/DrupalConsole.git
```

Install dependencies

Now that you have cloned the project, you need to download dependencies via Composer.

```
$ cd /path/to/DrupalConsole
$ composer install
```

Running the project

After using Composer to download dependencies, you can run the project by executing:

```
$ bin/console
```

Create a symbolic link

You can run this command to easily access the Drupal Console from anywhere on your system:

```
$ sudo ln -s /path/to/DrupalConsole/bin/console /usr/local/bin/console.dev
```

NOTE: The name `console.dev` is just an alias you can name it anything you like.

Keeping your fork up to date

After some time your forked repository and the original one (called upstream) will eventually get out of sync leaving you with an old, unsupported version.

To sync changes you make in a fork with the original repository, you should:

Configuring a remote fork:

Specify and configure a new remote upstream repository that points to the upstream repository in Git.

```
git remote add upstream https://github.com/hechoendrupal/DrupalConsole.git
```

For detailed information please visit Github's guide Configuring a remote fork

Syncing your fork

Sync your fork to keep it up-to-date with the upstream repository.

```
git fetch upstream
git merge upstream/master
```

For detailed information please visit Github's guide Syncing a fork

Creating an Issue

If you found an issue or maybe you like to propose a new feature to the project, you need to access the following link:

https://github.com/hechoendrupal/DrupalConsole/issues/new

Please review the guidelines for contributing to Drupal Console at:

https://github.com/hechoendrupal/DrupalConsole/blob/master/CONTRIBUTING.md

Creating a Pull Request

Remember to create a branch before start developing! It's name should contain the issue id and a slug to tell what the thing you're working on is about, for example: `1337-lorem-ipsum`

If you haven't yet contributed to a project on GitHub, or aren't still sure what the workflow looks like, read the documentation about pull requests. You may also wish to download the GitHub application (Mac | Windows, which simplifies the workflow a bit and provides a nice GUI for your contributions).

Contribute to this documentation

This documentation is a work-in-progress and *you are welcome to pitch in and help*. Simply fork the Drupal Console Book on GitHub. If you haven't yet contributed to a project on GitHub, or aren't still sure what the workflow looks like, read the documentation about pull requests. You may also wish to download the GitHub application (Mac | Windows, which simplifies the workflow a bit and provides a nice GUI for your contributions).

Installation problems

When you run DrupalConsole from your Drupal 8 root directory, you can get different error messages, we will try to compile the reported issues and how to have them fixed.

Error message:

```
[PDOException] SQLSTATE[HY000] [2002] No such file or directory
```

You will need to edit your 'host' in your 'settings.php' file.

Navigate to `sites/default/settings.php`. In your `settings.php` file, change the `host` to read:

```
'host' => '127.0.0.1'
```

or if your 'settings.php' file already reads:

```
'host' => '127.0.0.1'
```

change it to read:

```
'host' => 'localhost'.
```

After you make the change, be sure to save the file and then run DrupalConsole again.

Error message:

```
[PDOException]
SQLSTATE[HY000] [2002] Can't connect to local MySQL server through socket '/tmp/mysql.soc
```

Creating a symlink pointing to `/tmp/mysql.sock`:

```
ln -s /path/to/your/mysql/data/mysql.sock /tmp/mysql.sock
```

Commands not listed

This document is a work-in-progress. At any time, it is possible that the Drupal Console project is ahead of the documentation. While we endeavor to keep this book up-to-date, it is always possible that some commands have been created for the Drupal Console, but are not yet described in this document. For a full list of supported commands, use the **list** command, e.g. `$ drupal list`

If you see a command that is not yet described here, you are also welcome to contribute to this documentation, using the **--help** output for the command as a simple starting point.

References

Drupal Console code repository

https://github.com/hechoendrupal/DrupalConsole

Documentation repository

https://github.com/hechoendrupal/drupal-console-book

Resources

- Symfony Components
- Drupal 8
- PHP the right way
- KNP University
- Build a module
- DrupalizeMe
- Git
- Composer
- Box