# Drush

TABLE OF CONTENTS





---

## Overview

Drush is a command line shell and Unix scripting interface for Drupal. Your life will go much better if you use Drush! Drush core ships with lots of useful commands and generators. Similarly, it runs update.php, executes SQL queries, runs content migrations, and miscellaneous utilities like cron or cache rebuild. Drush can be extended by 3rd party commandfiles.

From https://www.drush.org/latest/

Drush is extensible with the ability to create drush commands and drush scripts.

## Drush commands

These are new commands that you can add to your modules to allow drush to do useful things. Modules sometimes include drush commands e.g. search api (https://www.drupal.org/project/search_api) More about this below.

Modules that have drush 9 commands need the following

1. drush.services.yml
2. The implementation class which extends the DrushCommands class in module/src/Commands.
3. In the class, you annotate the function with the command
4. A section in the composer.json file referencing the extra, drush, services.

See below for examples of each.

### Generate commands

Drush can generate all the files to create a new command. Use:

```
$ drush generate drush-command-file
```

Or `drush dcf`

### Drush command Example

The (Search API module)[https://www.drupal.org/project/search_api] has a set of Drush commands.

Here is `web/modules/contrib/search_api/drush.services.yml`

```
services:
  search_api.commands:
    class: Drupal\search_api\Commands\SearchApiCommands
    arguments: ['@entity_type.manager', '@module_handler', '@event_dispatcher']
    tags:
      - { name: drush.command }
```

Here is the beginning of the `web/modules/contrib/search_api/src/Commands/SearchApiCommands.php` which includes the class `SearchApiCommands` which extends the `DrushCommands` class:

```php
<?php

namespace Drupal\search_api\Commands;

use Consolidation\OutputFormatters\StructuredData\RowsOfFields;
use Drupal\Core\Entity\EntityTypeManagerInterface;
use Drupal\Core\Extension\ModuleHandlerInterface;
use Drupal\search_api\Contrib\RowsOfMultiValueFields;
use Drupal\search_api\Utility\CommandHelper;
use Drush\Commands\DrushCommands;
use Psr\Log\LoggerInterface;
use Symfony\Contracts\EventDispatcher\EventDispatcherInterface;

/**
 * Defines Drush commands for the Search API.
 */
class SearchApiCommands extends DrushCommands {

  /**
   * The command helper.
   *
   * @var \Drupal\search_api\Utility\CommandHelper
   */
  protected $commandHelper;

  /**
   * Constructs a SearchApiCommands object.
   *
   * @param \Drupal\Core\Entity\EntityTypeManagerInterface $entityTypeManager
   *   The entity type manager.
   * @param \Drupal\Core\Extension\ModuleHandlerInterface $moduleHandler
   *   The module handler.
   * @param \Symfony\Contracts\EventDispatcher\EventDispatcherInterface $eventDispatcher
   *   The event dispatcher.
   *
   * @throws \Drupal\Component\Plugin\Exception\InvalidPluginDefinitionException
   *   Thrown if the "search_api_index" or "search_api_server" entity types'
   *   storage handlers couldn't be loaded.
   * @throws \Drupal\Component\Plugin\Exception\PluginNotFoundException
   *   Thrown if the "search_api_index" or "search_api_server" entity types are
   *   unknown.
   */
  public function __construct(EntityTypeManagerInterface $entityTypeManager, ModuleHandlerInterface $moduleHandler, EventDispatcherInterface $eventDispatcher) {
    parent::__construct();

    $this->commandHelper = new CommandHelper($entityTypeManager, $moduleHandler, $eventDispatcher, 'dt');
  }
```

Here is an example command (search_api:enable) from Search_API. This is an excerpt from web/modules/contrib/search_api/src/Commands/SearchApiCommands.php. Looking at the annotation you can see all the details of the command as well as the aliases. This annotation provides the ability to issue the command drush search_api:enable. The aliases allow you to use drush sapi-en or drush search-api-enable as well. Note. You can always add aliases etc. like drush @apc.dev sapi-en.

```
/**
 * Enables one disabled search index.
 *
 * @param string $indexId
 *   A search index ID.
 *
 * @throws \Drupal\search_api\ConsoleException
 *   Thrown if no indexes could be loaded.
 *
 * @command search-api:enable
 *
 * @usage drush search-api:enable node_index
 *   Enable the search index with the ID node_index.
 *
 * @aliases sapi-en,search-api-enable
 */
public function enable($indexId) {
  $this->commandHelper->enableIndexCommand([$indexId]);
}
```

You also need a `composer.json` in the custom module directory. Note the `extra`, `drush`, `services` that reference the `drush.services.yml` file:

Here is the version that drush generates by default:

```
{
    "name": "org/tea_vote_cache",
    "type": "drupal-drush",
    "extra": {
        "drush": {
            "services": {
                "drush.services.yml": "^10"
            }
        }
    }
}
```

Here is the composer.json from a project where I used (Search API module)[https://www.drupal.org/project/search_api] module at web/modules/contrib/search_api/composer.json:

```
{
  "name": "drupal/search_api",
  "description": "Provides a generic framework for modules offering search capabilities.",
  "type": "drupal-module",
  "homepage": "https://www.drupal.org/project/search_api",
  "authors": [
    {
      "name": "Thomas Seidl",
      "homepage": "https://www.drupal.org/u/drunken-monkey"
    },
    {
      "name": "Nick Veenhof",
      "homepage": "https://www.drupal.org/u/nick_vh"
    },
    {
      "name": "See other contributors",
      "homepage":"https://www.drupal.org/node/790418/committers"
    }
  ],
  "support": {
    "issues": "https://www.drupal.org/project/issues/search_api",
    "irc": "irc://irc.freenode.org/drupal-search-api",
    "source": "https://git.drupalcode.org/project/search_api"
  },
  "license": "GPL-2.0-or-later",
  "require-dev": {
    "drupal/language_fallback_fix": "@dev",
    "drupal/search_api_autocomplete": "@dev"
  },
  "suggest": {
    "drupal/facets": "Adds the ability to create faceted searches.",
    "drupal/search_api_autocomplete": "Allows adding autocomplete suggestions to search fields.",
    "drupal/search_api_solr": "Adds support for using Apache Solr as a backend."
  },
  "extra": {
    "drush": {
      "services": {
        "drush.services.yml": "^9 || ^10"
      }
    }
  },
  "conflict": {
    "drupal/search_api_solr": "2.* || 3.0 || 3.1"
  }
}
```

## Drush commands and parameters

You can specify the commands with a member function and all its parameters like this:

Here we have a command

```
public function commandWarmVotingCache($scope = 'current', $program_nid = 0, $options = ['option-name' => 'default']): int {
```

This expects a command like: `drush cwarm current 12345`

> **NOTE**
> The $options parameter is not generally for you to use. Drush uses it for all sorts of parameters internally. It is interesting to look at the values in the debugger.

### Output messages on screen

Here we use `print`:

```
$program_title = $program_node->getTitle();
print "Warming cache for program_nid: $program_nid Title: $program_title\n";
$progress = new Progress($program_nid);
$max_vote_number = $program_node->get('field_srp_vote_number')->value;
for ($vote_number = 0; $vote_number <= $max_vote_number; $vote_number++) {
  $status = $progress->getTeamKssCompletionStatus($vote_number);
  print "vote_number: $vote_number, Team KSS completion status: $status\n";
}
```

You can use the Drupal logger to output data to the terminal. If you specify `->info` instead of `->notice` below you will only see the output in the terminal if you add -vvv to the drush command i.e. `drush cwarm current -vvv`

```
\Drupal::logger('tea_teks_voting')->notice("Drush cache warm requested. Scope: $scope, Program_nid = $program_nid");
```

```
if ($rc === 0 ) {
  $this->logger()->success(dt('Operation completed.'));
}
else {
  $this->logger()->error(dt('Failed to complete processing.'));
}
```

## Drush Scripts

These are PHP Scripts that run after a full Drupal bootstrap. From https://www.drush.org/latest/commands/php_script/#options : A useful alternative to eval command when your php is lengthy or you can't be bothered to figure out bash quoting. If you plan to share a script with others, consider making a full Drush command instead, since that's more self-documenting. Drush provides command line options to the script via a variable called `$extra`.

You can execute these scripts with `drush scr script` (where script is the filename of the script to execute.). If you put the script in the `docroot` (or `web`) directory, you don't need to specify a path.

**TL;DR**

1. Put script file in docroot (No need to make it executable.)
2. Requires opening `<?php` tag
3. Execute with `drush scr scriptname arg1 arg2`
4. Arguments show up in the `$extra array` i.e. `$extra[0]`...
5. Print with `$this->output()->writeln`, `print "\n"` or `echo "\n"`

### Run your scripts

e.g. to execute example.drush

```
$ ddev drush php:script example.drush
```

I prefer the alias

```
$ddev drush scr example.drush
```

If you want to ssh into the DDEV container, you can use:

```
$ ddev ssh

$ drush scr example.drush
```

If you have drush installed globally, you can also run it from the host with:

```
$ drush scr example.drush
```

## Arguments example

Here is a simple example example2.drush showing arguments:

```
<?php

use Drush\Drush;

$this->output()->writeln("Hello world!");
// $extra is an array if you issue arguments e.g.
// drush scr example2.drush abc def
$this->output()->writeln("The extra options/arguments to this command were:");
$this->output()->writeln(print_r($extra, true));
```

Here is the script being run via ddev and showing its output:

```
$ ddev drush scr example2.drush abc def
Hello world!
The extra options/arguments to this command were:
Array
(
    [0] => abc
    [1] => def
)
```

## Arguments example2

Here is opinion_renamer.php script that expects a parameter and dies if it doesn't find it. Also if the second parameter is R, it reports that it is reverting titles.

```
<?php

print ("usage: drush scr test1 <acg term id> <R>\n");
print ("e.g. 976 is the termid for John Smith.\n");



$acg = $extra[0] ?? '';
if (empty($acg)) {
  die("no acg termid");
}
$undo = FALSE;
if (isset($extra[1])) {
  $revert = $extra[1];
  if (strtoupper($revert) == "R") {
    $undo = TRUE;
    print("reverting titles for acg $acg\n");
  }
}
```

## Hello world example

Here is an example of a script from https://raw.githubusercontent.com/drush-ops/drush/11.x/examples/helloworld.script:

```php
<?php

//
// This example demonstrates how to write a drush
// script. These scripts are run with the php:script command.
//
use Drush\Drush;

$this->output()->writeln("Hello world!");
$this->output()->writeln("The extra options/arguments to this command were:");
$this->output()->writeln(print_r($extra, true));

//
// We can check which site was bootstrapped via
// the '@self' alias, which is defined only if
// there is a bootstrapped site.
//
$self = Drush::aliasManager()->getSelf();;
if (!$self->hasRoot()) {
  $this->output()->writeln('No bootstrapped site.');
}
else {
  $this->output()->writeln('The following site is bootstrapped:');
  $this->output()->writeln(print_r($self->legacyRecord(), true));
}
```

Executing it looks like this:

```
$ ddev drush scr example.drush
Hello world!
The extra options/arguments to this command were:
Array
(
)

The following site is bootstrapped:
Array
(
    [root] => /var/www/html/web
    [uri] => https://d9book2.ddev.site
)
```

## Database query example

Here is example3.drush with a database query:

```php
<?php

use Drush\Drush;

$database = \Drupal::database();
$query = $database->query("SELECT nid, vid, type  FROM {node} n");
$results = $query->fetchAll();

$result_count = count($results);
$this->output()->writeln("Result count = " . $result_count);
```

## entityQuery example

Here is equery1.php showing looping through results of the query

```
<?php

$query = \Drupal::entityQuery('node');
$query->condition('status', 1);
$query->condition('type', 'vote');
$query->sort('title', 'ASC');
$nids = $query->execute();

if (empty($nids)) {
  $this->output()->writeln("No votes found");
}
else {
  $this->output()->writeln("Found " . count($nids) . " articles");
}

foreach ($nids as $nid) {
  $node = \Drupal\node\Entity\Node::load($nid);
  $title = $node->getTitle();
  $this->output()->writeln("Nid: " . $nid . " title: " . $title);
}
```

You could also change values and update the nodes using code like this:

```
foreach ($nids as $nid) {
  $node = \Drupal\node\Entity\Node::load($nid);
  $title = $node->getTitle();

  $old_title = substr($title, 0, 7);
  print("$title to $old_title\n");
  $node->set('title', $old_title);
  $node->save();
}
print "finished.\n";
return;
```

## Drush does that?

Who isn't impressed by the things Drush does? It really shows off the incredible talent of Moshe Weitzman and the team that keep Drush moving. Drush can do almost anything. Here are a few that I like:

### List blocks

To get drush to list all the blocks on your site.

```
drush ev "print_r(array_keys(\Drupal::service('plugin.manager.block')->getDefinitions()));"
```

which outputs something like:

```
    [38] => system_messages_block
    [39] => system_powered_by_block
    [40] => user_login_block
    [41] => views_block:comments_recent-block_1
    [42] => views_block:content_recent-block_1
    [43] => views_block:events-block_1
    [44] => views_block:related_products_and_services-block_1
    [45] => views_block:who_s_new-block_1
    [46] => views_block:who_s_online-who_s_online_block
    [47] => views_exposed_filter_block:news_listing_for_news_landing-page_1
    [48] => local_actions_block
    [49] => local_tasks_block
    [50] => page_title_block
    [51] => broken
)
```

### Generate code

Drush can write modules (maybe not quite as well as ChatGPT), but they will save you tons of time. Use drush generate module and or drush generate controller to get a nice starting point for you to write your own controllers.

For more, on generating controllers see https://www.drush.org/latest/generators/controller/

This is what it looks like to generate a controller:

```
$ drush generate controller

 Welcome to controller generator!
---------------------------------

 Module machine name [web]:
 ➤ general

 Class [GeneralController]:
 ➤ ExampleController

 Would you like to inject dependencies? [No]:
 ➤

 Would you like to create a route for this controller? [Yes]:
 ➤

 Route name [general.example]:
 ➤ general.book_example

 Route path [/general/example]:
 ➤ /general/book_example

 Route title [Example]:
 ➤ Book Example

 Route permission [access content]:
 ➤

 The following directories and files have been created or updated:
--------------------------------------------------------------------
 • /Users/selwyn/Sites/d9book2/web/modules/custom/general/general.routing.yml
 • /Users/selwyn/Sites/d9book2/web/modules/custom/general/src/Controller/ExampleController.php
```

## Sanitize Databases

From https://www.drush.org/latest/commands/sql_sanitize/

Sanitize the database by removing or obfuscating user data.

Commandfiles may add custom operations by implementing:

- \@hook on-event sql-sanitize-confirms. Display summary to user before confirmation.
- \@hook post-command sql-sanitize. Run queries or call APIs to perform sanitizing

Several working commandfiles may be found at https://github.com/drush-ops/drush/tree/11.x/src/Drupal/Commands/sql

## Delete nodes

To delete all node entities in batches of 5 use:

```
drush entity:delete node --chunks=5
```

More from https://www.drush.org/latest/commands/entity_delete/

- drush entity:delete node --bundle=article. Delete all article entities.
- drush entity:delete shortcut. Delete all shortcut entities.
- drush entity:delete node 22,24. Delete nodes 22 and 24.
- drush entity:delete node --exclude=9,14,81. Delete all nodes except node 9, 14 and 81.
- drush entity:delete user. Delete all users except uid=1.
- drush entity:delete node --chunks=5. Delete all node entities in steps of 5.

## Delete redirects

Delete all redirects on local site

```
drush @self entity:delete redirect
```

## Watch the watchdog log

```
$ drush watchdog:tail

348 09/Jan 19:29 cron Info Cron run completed.
347 09/Jan 19:29 cron Info Execution of update_cron() took 2504.96ms.
346 09/Jan 19:29 cron Warning Attempting to re-run cron while it is already running.
345 09/Jan 19:29 cron Info Starting execution of update_cron(), execution of system_cron() took 1420.76ms.
344 09/Jan 19:29 cron Info Starting execution of system_cron(), execution of search_cron() took 123.02ms.
343 09/Jan 19:29 cron Info Starting execution of search_cron(), execution of node_cron() took 56.93ms.
342 09/Jan 19:29 cron Info Starting execution of node_cron(), execution of history_cron() took 23.6ms.
341 09/Jan 19:29 cron Info Starting execution of history_cron(), execution of file_cron() took 48.27ms.
```

## Read Drupal config

To display any config value, use drush cget:

```
$ drush cget admin_toolbar.settings

menu_depth: 4
```

Or

```
$ drush cget user.settings
_core:
  default_config_hash: w314Zp7B4zzzzzzzzzv-KwZO2E1fSSK0
langcode: en
anonymous: Anonymous
verify_mail: true
notify:
  cancel_confirm: true
  password_reset: true
  status_activated: true
  status_blocked: false
  status_canceled: false
  register_admin_created: true
  register_no_approval_required: true
  register_pending_approval: true
register: visitors_admin_approval
cancel_method: user_cancel_block
password_reset_timeout: 86400
password_strength: true
```

And yes, you can use config:set (or cset) to change configuration. For more see https://www.drush.org/latest/commands/config_get/

### Check if current config matches exported config

When working on a project, to make sure you have exported any changes to config so you can check it into git, use drush cst and drush cex to export.

```
$ drush cst
 [notice] No differences between DB and sync directory.
```

```
$ drush cex -y
```

More at https://www.drush.org/latest/commands/config_status/ and https://www.drush.org/latest/commands/config_export/

Importing the config (and overwriting what is currently in the database is accomplished with:

```
$ drush cim -y
```

### User Login

```
$ drush uli 1 /admin/config (uid = 1, go directly to /admin/config)

$ drush uli --name=Sarah

$ drush uli --mail=Phase3TestPark@mightycitizen.com

$ drush @wolfman.local uli --no-browser

$ drush uli --no-browser
http://default/user/reset/1/1673815967/cLRjg5XAOlDzr5UrkirPg-kmX8KWGSwILUeyteZbsUU/login
```

Note. The URL above isn't a usable URL but you can just paste everything starting at /user onto the end of the real URL in your browser e.g. https://d9book2/ddev.site/ user/reset/1/1673815967/cLRjg5XAOlDzr5UrkirPg-kmX8KWGSwILUeyteZbsUU/login

```
$ drush @nicer.prod uli --no-browser --uid=11

$ drush uli --no-browser --uri=team.ddev.site --mail=TestPub1@team.com

$ drush uli --no-browser --uri=https://team.ddev.site --mail=arnie.williamson@team.com

$ drush uli --no-browser --uri=https://d9book2.ddev.site https://d9book2.ddev.site/user/reset/1/1673816036/RMTR1AVzcR-GeBlW1ckyqJZ-1kvA3gebidmPghSjMg4/login
```

More at https://www.drush.org/latest/commands/user_login/

### Drupal:directory

Return devel directory

```
$ drush dd devel

/Users/selwyn/Sites/d9book2/web/modules/contrib/devel
```

Return files directory

```
$ drush dd files

/Users/selwyn/Sites/d9book2/web/sites/default/files
```

Return Drupal root directory

```
$ drush dd

/Users/selwyn/Sites/d9book2/web
```

Navigate to the files directory.

```
cd $(drush dd files)
```

## Global Install

To install drush globally so you can just issue drush commands from anywhere on your system (mac only) use:

```
$ composer global require drush/drush:\^8.0
```

And add this to your .bashrc or .zshrc:

```
alias drush = '~.composer/vendor/drush/drush/drush'
```

## Drush aliases

These allow Drush to reach out to your client websites from your local machine. The alias files should be stored in /drush/sites (that is above your docroot so they will never be publicly accessible) and can be checked into your repo. If you have a file named /drush/sites/abc.site.yml you will be able to issue drush commands like drush @abc.dev status or drush @abc.prod sql-dump \>dbprod.sql from within the directory for your abc site.

You can clear cache (cr), enable a module (en devel), dump a database to your local (sql-dump), sync files from remote to local:

```
drush @mysite cr
drush @mysite en devel
drush @mysite sql-dump >dbprod.sql
drush rsync <source> <dest> (To rsync files to/from sites.  More on that below)
```

These commands will only work on your computer if you have drush installed globally with all its requirements met. As usual, DDEV has a nice workaround. To use drush aliases with DDEV, you will need to setup your ssh keys to run in the containers (ddev auth ssh) and then

you can issue drush alias commands from within the containers. For example:

```
$ ddev auth ssh
$ drush @abc.test status

Drupal version   : 9.4.8
Site URI         : http://abcmaterialsstg.prod.acquia-sites.com
DB driver        : mysql
DB hostname      : db-1zzzzzzzzb.cdb.database.services.acquia.io
DB port          : 3306
DB username      : uMzzzzzzs
DB name          : 1f3zzzzzddb
Database         : Connected
Drupal bootstrap : Successful
Default theme    : abc
Admin theme      : seven
PHP binary       : /usr/local/php8.1/bin/php
PHP config       : /usr/local/php8.1/etc/cli/php.ini
PHP OS           : Linux
PHP version      : 8.1.10
Drush script     : /var/www/html/vendor/drush/drush/drush
Drush version    : 11.3.2
Etc.
```

### Example Drush alias file

Here is an example drush alias (`abc.site.yml`) where `abc.dev` refers to a site at the fictitious IP address of 123.45.67.123 and the `abc.devsp` refers to a site at 100.1.2.3. The first assumes there is an ssh key setup on the host whereas the second would expect a password to be entered each time a drush command is issued.

```
dev:
  host: 123.45.67.123
  user: master_gzzzzzz
  root: '/home/zzzz.cloudy.com/kzzzzzz/public_html/web'
  uri: 'https://doug.merrysite.com'
  ssh:
    tty: 0
  paths:
    drush-script: '/home/zzzz.cloudy.com/kzzzzzz/public_html/vendor/drush/drush/drush'

devsp:
  host: 100.1.2.3
  user: spolit
  root: '/var/www/dev.paris.com/web'
  uri: 'https://dev.paris.com'
  ssh:
    tty: 0
    options: '-o PasswordAuthentication=yes'
  paths:
    drush-script: '/var/www/dev.paris.com/vendor/drush/drush/drush'
```

### Example Acquia Drush alias files

Usually you will need to make your own drush alias files. Here is an example alias file from Acquia. If this file is named `/drush/sites/abc.site.yml` you will be able to issue drush commands like `drush @abc.dev status` or `drush @abc.prod sql-dump >dbprod.sql` from within the directory for your abc site. The command `drush @abc.dev status` will return the status for the dev site. The command `drush @abc.prod sql-dump >dbprod.sql` will download the entire production from the production server database into a local file called `dbprod.sql`. For DDEV use, you should gzip the file and then you can easily import it into your local DDEV database with a command like `ddev import-db --src=dbprod.sql.gz`.

Note. The section in the file below with ssh and tty in it is to resolve a problem (from [https://github.com/drush-ops/drush/issues/4004](https://github.com/drush-ops/drush/issues/4004) ) with

the sql-dump drush command. Check the afforementioned Github.com issue link for side-effects.

Also, a little unusual is the `prod.livdev` section below. This facilitates using Acquia's live dev environment feature but is not necessary for hosting environments other than Acquia.

For more on using Drush aliases with Acquia see https://docs.acquia.com/cloud-platform/manage/ssh/drush/aliases/

```
# Application 'abcmaterials', environment 'dev'.
dev:
  root: /var/www/html/docroot
  ac-site: abcmaterials
  ac-env: dev
  ac-realm: prod
  uri: abcmaterialsdev.prod.acquia-sites.com
  host: abcmaterialsdev.ssh.prod.acquia-sites.com
  user: abcmaterials.dev
  paths:
    drush-script: drush9
  ssh:
    tty: 0

# Application 'abcmaterials', environment 'test'.
test:
  root: /var/www/html/docroot
  ac-site: abcmaterials
  ac-env: test
  ac-realm: prod
  uri: abcmaterialsstg.prod.acquia-sites.com
  host: abcmaterialsstg.ssh.prod.acquia-sites.com
  user: abcmaterials.test
  paths:
    drush-script: drush9
  ssh:
    tty: 0

# Application 'abcmaterials', environment 'prod'.
prod:
  root: /var/www/html/abcmaterials.prod/docroot
  ac-site: abcmaterials
  ac-env: prod
  ac-realm: prod
  uri: abcmaterials.prod.acquia-sites.com
  ssh:
    tty: 0
  prod.livedev:
    parent: '@abcmaterials.prod'
    root: /mnt/gfs/abcmaterials.prod/livedev/docroot
  host: abcmaterials.ssh.prod.acquia-sites.com
  user: abcmaterials.prod
  paths:
    drush-script: drush9
  ssh:
    tty: 0
```

### Example Drush alias file from Drush project

You can view this on github at https://github.com/drush-ops/drush/blob/master/examples/example.site.yml

```
#
# Example of valid statements for an alias file.
```

```
# Basic Alias File Usage
#
# In its most basic form, the Drush site alias feature provides a way
# for teams to share short names that refer to the live and staging sites
# (usually remote) for a given Drupal site.
#
# 1. Make a local working clone of your Drupal site and then
#    `cd` to the project work to select it.
# 2. Add an alias file called $PROJECT/drush/sites/self.site.yml,
#    where $PROJECT is the project root (location of composer.json file).
# 3. Run remote commands against the shared live or stage sites
#
# Following these steps, a cache:rebuild on the live environment would be:
#
#   $ drush @live cache:rebuild
#
# The site alias file should be named `self.site.yml` because this name is
# special, and is used to define the different environments (usually remote)
# of the current Drupal site.
#
# The contents of the alias file should look something like the example below:
#
# @code
# # File: self.site.yml
# live:
#   host: server.domain.com
#   user: www-admin
#   root: /other/path/to/live/drupal
#   uri: http://example.com
# stage:
#   host: server.domain.com
#   user: www-admin
#   root: /other/path/to/stage/drupal
#   uri: http://stage.example.com
# @endcode
#
# The top-level element names (`live` and `stage` in the example above) are
# used to identify the different environments available for this site. These
# may be used on the command line to select a different target environment
# to operate on by prepending an `@` character, e.g. `@live` or `@stage`.
#
# All of the available aliases for a site's environments may be listed via:
#
#   $ drush site:alias @self
#
# The elements of a site alias environment are:
#
# - 'host': The fully-qualified domain name of the remote system
#   hosting the Drupal instance. **Important Note: The remote-host option
#   must be omitted for local sites, as this option controls various
#   operations, such as whether or not rsync parameters are for local or
#   remote machines, and so on.
# - 'user': The username to log in as when using ssh or rsync. If each user
#    has to use own username, you can create an environment variable which holds
#    the value, and reference via ${env.PROJECT_SSH_USER} (for example). Or you may
#    omit the `user:` item and specify a user in the `~/.ssh/config` file.
# - 'root': The Drupal root; must not be specified as a relative path.
# - 'uri': The value of --uri should always be the same as
#   when the site is being accessed from a web browser (e.g. http://example.com)
```

```
#
# Drush uses ssh to run commands on remote systems; all team members should
# install ssh keys on the target servers (e.g. via ssh-add).

# Advanced Site Alias File Usage
#
# It is also possible to create site alias files that reference other
# sites on the same local system. Site alias files for other local sites
# are usually stored in the directory `~/.drush/sites`; however, Drush does
# not search this location for alias files by default. To use this location,
# you must add the path in your Drush configuration file. For example,
# to re-add both of the default user alias path from Drush 8, put the following
# in your ~/.drush/drush.yml configuration file:
#
# @code
# drush:
#   paths:
#     alias-path:
#       - '${env.HOME}/.drush/sites'
#       - /etc/drush/sites
# @endcode
#
# The command `drush core:init` will automatically configure your
# ~/.drush/drush.yml configuration file to add `~/.drush/sites` and
# `/etc/drush/sites` as locations where alias files may be placed.
#
# A canonical alias named "example" that points to a local
# Drupal site named "http://example.com" looks like this:
#
# @code
# File: example.site.yml
# dev:
#   root: /path/to/drupal
#   uri: http://example.com
# @endcode
#
# Note that the first part of the filename (in this case "example")
# defines the name of the site alias, and the top-level key ("dev")
# defines the name of the environment.
#
# With these definitions in place, it is possible to run commands targeting
# the dev environment of the target site via:
#
#   $ drush @example.dev status
#
# This command is equivalent to the longer form:
#
#   $ drush --root=/path/to/drupal --uri=http://example.com status
#
# See "Additional Site Alias Options" below for more information.

# Converting Legacy Alias Files
#
# To convert legacy alias (*.aliases.drushrc.php) to yml, run the
# site:alias-convert command.

# Altering aliases:
#
# See examples/Commands/SiteAliasAlterCommands.php for an example.
```

```
# Environment variables:
#
# Site aliases may reference environment variables, just like any Drush config
# file. For example, ${env.PROJECT_SSH_USER} will be replaced by the value
# of the PROJECT_SSH_USER environment value.
#
# SSH site aliases may set environment variables via the 'env-vars' key.
# See below.

# Additional Site Alias Options
#
# Aliases are commonly used to define short names for
# local or remote Drupal installations; however, an alias
# is really nothing more than a collection of options.
#
# - 'docker': When specified, Drush executes via docker-compose exec rather than ssh
#    - 'service': the name of the container to run on.
#    - 'exec':
#      - 'options': Options for the exec subcommand.
# - 'os': The operating system of the remote server.  Valid values
#   are 'Windows' and 'Linux'. Be sure to set this value for all remote
#   aliases because the default value is PHP_OS if 'remote-host'
#   is not set, and 'Linux' (or $options['remote-os']) if it is. Therefore,
#   if you set a 'remote-host' value, and your remote OS is Windows, if you
#   do not set the 'OS' value, it will default to 'Linux' and could cause
#   unintended consequences, particularly when running 'drush sql-sync'.
# - 'ssh': Contains settings used to control how ssh commands are generated
#   when running remote commands.
#   - 'options': Contains additional commandline options for the ssh command
#   itself, e.g. "-p 100"
#   - 'tty': Usually, Drush will decide whether or not to create a tty (via
#   the ssh '--t' option) based on whether the local Drush command is running
#   interactively or not. To force Drush to always or never create a tty,
#   set the 'ssh.tty' option to 'true' or 'false', respectively.
# - 'paths': An array of aliases for common rsync targets.
#   Relative aliases are always taken from the Drupal root.
#   - 'files': Path to 'files' directory.  This will be looked up if not
#     specified.
#   - 'drush-script': Path to the remote Drush command.
# - 'command': These options will only be set if the alias
#   is used with the specified command.  In the example below, the option
#   `--no-dump` will be selected whenever the @stage alias
#   is used in any of the following ways:
#     - `drush @stage sql-sync @self @live`
#     - `drush sql-sync @stage @live`
#     - `drush sql-sync @live @stage`
#   NOTE: Setting boolean options broke with Symfony 3. This will be fixed
#     in a future release. See: https://github.com/drush-ops/drush/issues/2956
# - 'env-vars': An array of key / value pairs that will be set as environment
#   variables.
#
# Complex example:
#
# @code
# # File: remote.site.yml
# live:
#   host: server.domain.com
#   user: www-admin
#   root: /other/path/to/drupal
```

```
#   uri: http://example.com
#   ssh:
#     options: '-p 100'
#   paths:
#     drush-script: '/path/to/drush'
#   env-vars:
#     PATH: /bin:/usr/bin:/home/www-admin/.composer/vendor/bin
#     DRUPAL_ENV: live
#   command:
#     site:
#       install:
#         options:
#           admin-password: 'secret-secret'
# @endcode

# Site Alias Files for Service Providers
#
# There are a number of service providers that manage Drupal sites as a
# service. Drush allows service providers to create collections of site alias
# files to reference all of the sites available to a single user. In order
# to do this, a new location must be defined in your Drush configuration
# file:
#
# @code
# drush:
#   paths:
#     alias-path:
#       - '${env.HOME}/.drush/sites/provider-name'
# @endcode
#
# Site aliases stored in this directory may then be referenced by its
# full alias name, including its location, e.g.:
#
#   $ drush @provider-name.example.dev
#
# Such alias files may still be referenced by their shorter name, e.g.
# `@example.dev`. Note that it is necessary to individually list every
# location where site alias files may be stored; Drush never does recursive
# (deep) directory searches for alias files.
#
# The `site:alias` command may also be used to list all of the sites and
# environments in a given location, e.g.:
#
#   $ drush site:alias @provider-name
#
# Add the option `--format=list` to show only the names of each site and
# environment without also showing the values in each alias record.

# Wildcard Aliases for Service Providers
#
# Some service providers that manage Drupal sites allow customers to create
# multiple "environments" for a site. It is common for these providers to
# also have a feature to automatically create Drush aliases for all of a
# user's sites. Rather than write one record for every environment in that
# site, it is also possible to write a single "wildcard" alias that represents
# all possible environments. This is possible if the contents of each
# environment alias are identical save for the name of the environment in
# one or more values. The variable `${env-name}` will be substituted with the
# environment name wherever it appears.
```

```
#
# Example wildcard record:
#
# @code
# # File: remote-example.site.yml
# '*':
#   host: ${env-name}.server.domain.com
#   user: www-admin
#   root: /path/to/${env-name}
#   uri: http://${env-name}.remote-example.com
# @endcode
#
# With a wildcard record, any environment name may be used, and will always
# match. This is not desirable in instances where the specified environment
# does not exist (e.g. if the user made a typo). An alias alter hook in a
# policy file may be used to catch these mistakes and report an error.
# @see SiteAliasAlterCommands for an example on how to do this.

# Developer Information
#
# See https://github.com/consolidation/site-alias for more developer
# information about Site Aliases.
#
# An example appears below. Edit to suit and remove the @code / @endcode and
# leading hashes to enable.
#
# @code
# # File: mysite.site.yml
# local:
# This environment is an example of the DockerCompose transport.
#   docker:
#     service: drupal
#     exec:
#       options: --user USER
# stage:
#   uri: http://stage.example.com
#   root: /path/to/remote/drupal/root
#   host: mystagingserver.myisp.com
#   user: publisher
#   os: Linux
#   paths:
#     - files: sites/mydrupalsite.com/files
#     - custom: /my/custom/path
#   command:
#     sql:
#       sync:
#         options:
#           no-dump: true
# dev:
#   root: /path/to/docroot
#   uri: https://dev.example.com
# @endcode

# Example of rsync with exclude-paths
#
# Note that most options typically passed to rsync via `drush rsync` are
# "passthrough options", which is to say they appear after the `--` separator
# on the command line. Passthrough options are actually arguments, and
# it is not possible to set default arguments in an alias record. The
# `drush rsync` command does support two options, `--mode` and `--exclude-paths`,
```

```
# which are interpreted directly by Drush. Default values for these options
# may be specified in an alias record, as shown below.
#
# @code
# dev:
#   root: /path/to/docroot
#   uri: https://dev.example.com
#   command:
#     core:
#       rsync:
#         options:
#           mode: rlptz
#           exclude-paths: 'css:imagecache:ctools:js:tmp:php:styles'
# @endcode
```

## Drush rsync

You can use aliases to rsync files using drush rsync. The syntax is: `drush rsync <source> <dest>` or more specifically: `drush rsync @sitename.prod:/path/to/files /path/to/local/destination`

You can use `@self` to represent the local site e.g. `drush rsync @labs.prod:%files @self:%files`

To copy everything in `./keys` folder to the `keys` folder on the host use:

```
$ drush rsync ./keys/ @sf:/home/master/applications/dir/public_html/keys/
```

Copy 1 file to the dir test server:

```
drush rsync dbprod.sql.gz @dir.testsp:/var/www/dir.tater.com/web/
```

You can also use `:%files` as a source or dest. Examples below:

```
drush rsync @fightclub.prod:/var/www/html/fightclub.prod/docroot/sites/default/files/ ~/Sites/fightclub/docroot/sites/default/files/

drush rsync @dak.prodsp:%files /Users/selwyn/Sites/dakbackup/web/sites/default/files/

drush rsync @dak.prodsp:/var/www/dak.tater.com/web/ /Users/selwyn/Sites/dakbackup/web/
```

Copy 1 file from local to @dak.dev: drush rsync dbprod-good.sql.gz @dak.dev:/home/1234567.cloudwaysapps.com/hzzzz/public_html/drush-backups/

## Drush SQL-sync

To sync databases use: `drush sql-sync <source> <destination>`

e.g.

```
$ drush sql-sync @flexy.stage @flex.dev
```

## SQL Queries

Examples of SQL queries. Note. you can update, delete or select.

```
$ drush sqlq "select nid, vid, type from node where type = 'vote'"
35 40 vote
36 41 vote
```

```
$ drush sqlq "select count(*) as redirects"
1
```

## Drush launcher

install drush launcher from https://github.com/drush-ops/drush-launcher

By installing the drush launcher globally on your local machine, you can simply type drush on the command line, and the launcher will find and execute the project specific version of drush located in your project's vendor directory.

from https://github.com/drush-ops/drush-launcher: In order to avoid dependency issues, it is best to require Drush on a per-project basis via Composer (`$ composer require drush/drush`). This makes Drush available to your project by placing it at vendor/bin/drush.

However, it is inconvenient to type `vendor/bin/drush` to execute Drush commands. By installing the drush launcher globally on your local machine, you can simply type drush on the command line, and the launcher will find and execute the project specific version of drush located in your project's vendor directory.

## Drupal 7 Drush scripts

Here are some old Drupal 7 Drush scripts in case they are of any interest

```
//content created my admin - d7
$uid = 1;
$result = db_query('SELECT n.nid, n.title, n.created
FROM {node} n WHERE n.uid = :uid', array(':uid' => $uid));
// Result is returned as a iterable object that returns a stdClass object on each iteration
foreach ($result as $record) {
  // Perform operations on $record->title, etc. here.
  // in this example the available data would be mapped to object properties:
  // $record->nid, $record->title, $record->created
  print_r($record);
}
```

```
// Login as admin
$admin_uid = 1;
$form_state = array('uid' => $admin_uid);
user_login_submit(array(), $form_state);
// Now the logged in user global $user object become available.
global $user;
print_r($user);
```

```
// Args
drush_print("Hello world!");
drush_print();
drush_print("The arguments to this command were:");
//
// If called with --everything, use drush_get_arguments
// to print the commandline arguments.  Note that this
// call will include 'php-script' (the drush command)
// and the path to this script.
//
if (drush_get_option('everything')) {
  drush_print("  " . implode("\n  ", drush_get_arguments()));
}
//
// If --everything is not included, then use
// drush_shift to pull off the arguments one at
// a time.  drush_shift only returns the user
// commandline arguments, and does not include
// the drush command or the path to this script.
//
else {
  while ($arg = drush_shift()) {
    drush_print('  ' . $arg);
  }
}

drush_print();
```

It is optiona to have <?php opening tag and a #! lines:

This script takes a filename as a parameter and checks if the file exists.

```
<?php
#!/Users/selwyn/.composer/vendor/bin/drush

drush_print("Hello world!");
drush_print();
$arg = drush_shift();
if ($arg) {
  if (file_exists($arg)) {
    drush_print("yep - it's here");
  }
}
```

## Resources

- Drush docs site https://www.drush.org/latest/
- Drush repository https://github.com/drush-ops/drush
- Drush code generators https://www.drush.org/latest/generators/all/
- Stack Exchange Drush issues https://drupal.stackexchange.com/questions/tagged/drush
- Drush issue queue on github https://github.com/drush-ops/drush/issues
- Creating Drush 9 commands and porting legacy commands October 2017 by Pawel Ginalski https://gbyte.dev/blog/creating-drush-9-commands-and-porting-legacy-commands
- Porting commands to Drush 9 from Moshe Weitzman September 2017 https://weitzman.github.io/blog/port-to-drush9

- Drush alias usage on Acquia https://docs.acquia.com/cloud-platform/manage/ssh/drush/aliases/
- Controlling multiple sites with Drush 9 from Joachim Noreiko Mar 2019 http://www.noreiko.com/blog/controlling-multiple-sites-drush-9

---

---

Back to top



Page last modified: Apr 24 2023.

Edit this page on GitHub