Salesforce CLI Command Reference

Salesforce, Winter '24

@salesforcedocs

Last updated: September 22, 2023

CONTENTS

SALESFORCE CLI COMMAND REFERENCE

This command reference contains information about the Salesforce CLI commands and their flags. The first section contains commands for the `sf`-style commands, the second for the `sfdx`-style commands.

Salesforce CLI Release Notes

Use the Release Notes to learn about the most recent updates and changes to Salesforce CLI.

We release new versions of Salesforce CLI weekly. Read the weekly release notes to learn about new features, changes, and bug fixes in both the current release and the release candidate.

sf

This section contains information about the `sf` commands and their parameters.

This version of the `sf` command reference includes details about version 2.8.10 of the `sf` executable of Salesforce CLI and the following plug-in versions:

- `@salesforce/plugin-login` version 1.2.29
- `@salesforce/plugin-env` version 2.1.25
- `@salesforce/plugin-deploy-retrieve` version 1.17.8
- `@salesforce/plugin-settings` version 1.4.28
- `@salesforce/plugin-functions` version 1.21.11
- `@salesforce/plugin-info` version 2.6.40
- `@salesforce/plugin-sobject` version 0.2.6
- `@salesforce/plugin-limits` version 2.3.33
- `@salesforce/plugin-schema` version 2.3.25
- `@salesforce/plugin-custom-metadata` version 2.1.41
- `@salesforce/plugin-data` version 2.5.8

- `@salesforce/plugin-community` version 2.3.15
- `@salesforce/plugin-signups` version 1.4.34
- `@salesforce/plugin-user` version 2.3.32
- `@salesforce/plugin-org` version 2.10.6
- `@salesforce/plugin-packaging` version 1.24.0
- `@salesforce/plugin-templates` version 55.5.11
- `@salesforce/plugin-apex` version 2.3.14
- `@salesforce/plugin-auth` version 2.8.16
- `@salesforce/plugin-dev` version 1.1.10
- `@salesforce/sfdx-plugin-lwc-test` version 1.0.2
- `@salesforce/plugin-devops-center` version 1.1.4
- `@salesforce/plugin-marketplace` version 0.1.3

For information about installing Salesforce CLI, see the *Salesforce CLI Setup Guide*.

For information about Salesforce CLI changes, see the *Salesforce CLI Release Notes*.

`alias` Commands

Use the alias commands to manage your aliases.

`alias list`

List all aliases currently set on your local computer.

Description for `alias list`

Aliases are global, which means that you can use all the listed aliases in any Salesforce DX project on your computer.

Examples for `alias list`

List all the aliases you've set:

```
sf alias list
```

Usage

`sf alias list`
`[--json]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

Aliases for `alias list`

```
force:alias:list
```

`alias set`

Set one or more aliases on your local computer.

Description for `alias set`

Aliases are user-defined short names that make it easier to use the CLI. For example, users often set an alias for a scratch org usernames because they're long and unintuitive. Check the --help of a CLI command to determine where you can use an alias.

You can associate an alias with only one value at a time. If you set an alias multiple times, the alias points to the most recent value. Aliases are global; after you set an alias, you can use it in any Salesforce DX project on your computer.

Use quotes to specify an alias value that contains spaces. You typically use an equal sign to set your alias, although you don't need it if you're setting a single alias in a command.

Examples for `alias set`

Set an alias for a scratch org username:

```
sf alias set my-scratch-org=test-sadbiytjsupn@example.com
```

Set multiple aliases with a single command:

```
sf alias set my-scratch-org=test-sadbiytjsupn@example.com
my-other-scratch-org=test-ss0xut7txzxf@example.com
```

Set an alias that contains spaces:

```
sf alias set my-alias='alias with spaces'
```

Set a single alias without using an equal sign:

```
sf alias set my-scratch-org test-ss0xut7txzxf@example.com
```

Usage

`sf alias set`
  `[--json]`

Flags

`--json`
  Optional

  Format output as json.

  Type: boolean

Aliases for `alias set`

```
force:alias:set
```

`alias unset`

Unset one or more aliases that are currently set on your local computer.

Description for `alias unset`

Aliases are global, so when you unset one it's no longer available in any Salesforce DX project.

Examples for `alias unset`

Unset an alias:

```
sf alias unset my-alias
```

Unset multiple aliases with a single command:

```
sf alias unset my-alias my-other-alias
```

Unset all aliases:

```
sf alias unset --all [--no-prompt]
```

Usage

`sf alias unset`

`[--json]`

`[-a]`

`[-p]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-a` | `--all`

Optional

Unset all currently set aliases.

Type: boolean

`-p` | `--no-prompt`

Optional

Don't prompt the user for confirmation when unsetting all aliases.

Type: boolean

Aliases for `alias unset`

```
force:alias:unset
```

`analytics` Commands

Work with analytics assets.

analytics generate template

Generate a simple Analytics template.

`analytics generate template`

Generate a simple Analytics template.

Description for `analytics generate template`

The metadata files associated with the Analytics template must be contained in a parent directory called "waveTemplates" in your package directory. Either run this command from an existing directory of this name, or use the --output-dir flag to generate one or point to an existing one.

Examples for `analytics generate template`

Generate the metadata files for a simple Analytics template file called myTemplate in the force-app/main/default/waveTemplates directory:

```
sf analytics generate template --name myTemplate --output-dir
force-app/main/default/waveTemplates
```

Usage

`sf analytics generate template`

```
[--json]
[-d OUTPUT-DIR]
[--api-version API-VERSION]
-n NAME
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-n | --name NAME`

Required

Name of the Analytics template.

Type: option

Aliases for **`analytics generate template`**

```
force:analytics:template:create
```

`apex` Commands

Use the apex commands to create Apex classes, execute anonymous blocks, view your logs, run Apex tests, and view Apex test results.

apex generate class
Generate an Apex class.

apex generate trigger
Generate an Apex trigger.

apex get log
Fetch the specified log or given number of most recent logs from the org.

apex get test
Display test results for a specific asynchronous test run.

apex list log
Display a list of IDs and general information about debug logs.

apex run
Execute anonymous Apex code entered on the command line or from a local file.

apex run test
Invoke Apex tests in an org.

apex tail log
Activate debug logging and display logs in the terminal.

`apex generate class`

Generate an Apex class.

Description for **`apex generate class`**

Generates the Apex *.cls file and associated metadata file. These files must be contained in a parent directory called "classes" in your package directory. Either run this command from an existing directory of this name, or use the --output-dir flag to generate one or point to an existing one.

Examples for **`apex generate class`**

Generate two metadata files associated with the MyClass Apex class (MyClass.cls and MyClass.cls-meta.xml) in the current directory:

```
sf apex generate class --name MyClass
```

Similar to previous example, but generates the files in the "force-app/main/default/classes" directory:

```
sf apex generate class --name MyClass --output-dir force-app/main/default/classes
```

Usage

`sf apex generate class`

```
[--json]

-n NAME

[-t TEMPLATE]

[-d OUTPUT-DIR]

[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated Apex class.

The name can be up to 40 characters and must start with a letter.

Type: option

`-t | --template TEMPLATE`

Optional

Template to use for file creation.

Supplied parameter values or default values are filled into a copy of the template.

Type: option

Permissible values are: ApexException, ApexUnitTest, DefaultApexClass, InboundEmailService

Default value: DefaultApexClass

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `apex generate class`

```
force:apex:class:create
```

`apex generate trigger`

Generate an Apex trigger.

Description for `apex generate trigger`

Generates the Apex trigger *.trigger file and associated metadata file. These files must be contained in a parent directory called "triggers" in your package directory. Either run this command from an existing directory of this name, or use the --output-dir flag to generate one or point to an existing one.

If you don't specify the --sobject flag, the .trigger file contains the generic placeholder SOBJECT; replace it with the Salesforce object you want to generate a trigger for. If you don't specify --event, "before insert" is used.

Examples for `apex generate trigger`

Generate two files associated with the MyTrigger Apex trigger (MyTrigger.trigger and MyTrigger.trigger-meta.xml) in the current directory:

```
sf apex generate trigger --name MyTrigger
```

Similar to the previous example, but generate the files in the "force-app/main/default/triggers" directory:

```
sf apex generate trigger --name MyTrigger --output-dir force-app/main/default/triggers
```

Generate files for a trigger that fires on the Account object before and after an insert:

```
sf apex generate trigger --name MyTrigger --sobject Account --event "before insert,after
insert"
```

Usage

`sf apex generate trigger`

```
  [--json]
  -n NAME
  [-t TEMPLATE]
  [-d OUTPUT-DIR]
  [--api-version API-VERSION]
  [-s SOBJECT]
  [-e EVENT]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated Apex trigger

The name can be up to 40 characters and must start with a letter.

Type: option

`-t | --template TEMPLATE`

Optional

Template to use for file creation.

Supplied parameter values or default values are filled into a copy of the template.

Type: option

Permissible values are: ApexTrigger

Default value: ApexTrigger

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-s | --sobject SOBJECT`

Optional

Salesforce object to generate a trigger on.

Type: option

Default value: SOBJECT

`-e | --event EVENT`

Optional

Events that fire the trigger.

Type: option

Permissible values are: before insert, before update, before delete, after insert, after update, after delete, after undelete

Default value: before insert

Aliases for `apex generate trigger`

```
force:apex:trigger:create
```

`apex get log`

Fetch the specified log or given number of most recent logs from the org.

Description for `apex get log`

To get the IDs for your debug logs, run "sf apex log list". Executing this command without flags returns the most recent log.

Examples for `apex get log`

Fetch the log in your default org using an ID:

```
sf apex get log --log-id <log id>
```

Fetch the log in the org with the specified username using an ID:

```
sf apex get log --log-id <log id> --target-org me@my.org
```

Fetch the two most recent logs in your default org:

```
sf apex get log --number 2
```

Similar to previous example, but save the two log files in the specified directory:

```
sf apex get log --output-dir /Users/sfdxUser/logs --number 2
```

Usage

`sf apex get log`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`[-i LOG-ID]`

`[-n NUMBER]`

`[-d OUTPUT-DIR]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o` | `--target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-i` | `--log-id LOG-ID`

Optional

ID of the specific log to display.

Type: option

-n | --number NUMBER

Optional

Number of the most recent logs to display.

Type: option

-d | --output-dir OUTPUT-DIR

Optional

Directory for saving the log files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Aliases for `apex get log`

```
force:apex:log:get
```

apex get test

Display test results for a specific asynchronous test run.

Description for `apex get test`

Provide a test run ID to display test results for an enqueued or completed asynchronous test run. The test run ID is displayed after running the "sf apex test run" command.

Examples for `apex get test`

Display test results for your default org using a test run ID:

```
sf apex get test --test-run-id <test run id>
```

Similar to previous example, but output the result in JUnit format:

```
sf apex get test --test-run-id <test run id> --result-format junit
```

Also retrieve code coverage results and output in JSON format:

```
sf apex get test --test-run-id <test run id> --code-coverage --json
```

Specify a directory in which to save the test results from the org with the specified username (rather than your default org):

```
sf apex get test --test-run-id <test run id> --code-coverage --output-dir <path to outputdir>
 --target-org me@myorg',
```

Usage

sf apex get test

[--json]

-o TARGET-ORG

[--api-version API-VERSION]

```
-i TEST-RUN-ID
[-c]
[-d OUTPUT-DIR]
[-r RESULT-FORMAT]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-i | --test-run-id TEST-RUN-ID`

Required

ID of the test run.

Type: option

`-c | --code-coverage`

Optional

Retrieve code coverage results.

Type: boolean

`-d | --output-dir OUTPUT-DIR`

Optional

Directory in which to store test result files.

Type: option

`-r | --result-format RESULT-FORMAT`

Optional

Format of the results.

Type: option

Permissible values are: human, tap, junit, json

Default value: human

Aliases for `apex get test`

```
force:apex:test:report
```

`apex list log`

Display a list of IDs and general information about debug logs.

Description for `apex list log`

Run this command in a project to list the IDs and general information for all debug logs in your default org.

To fetch a specific log from your org, obtain the ID from this command's output, then run the “sf apex log get” command.

Examples for `apex list log`

List the IDs and information about the debug logs in your default org:

```
sf apex list log
```

Similar to previous example, but use the org with the specified username:

```
sf apex list log --target-org me@my.org
```

Usage

`sf apex list log`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `apex list log`

```
force:apex:log:list
```

`apex run`

Execute anonymous Apex code entered on the command line or from a local file.

Description for `apex run`

If you don’t run this command from within a Salesforce DX project, you must specify the —-target-org flag.

To execute your code interactively, run this command with no flags. At the prompt, enter all your Apex code; press CTRL-D when you're finished. Your code is then executed in a single execute anonymous request.

For more information, see "Anonymous Blocks" in the Apex Developer Guide.

Examples for `apex run`

Execute the Apex code that's in the ~/test.apex file in the org with the specified username:

```
sf apex run --target-org testusername@salesforce.org --file ~/test.apex
```

Similar to previous example, but execute the code in your default org:

```
sf apex run --file ~/test.apex
```

Run the command with no flags to start interactive mode; the code will execute in your default org when you exit. At the prompt, start type Apex code and press the Enter key after each line. Press CTRL+D when finished.

```
sf apex run
```

Usage

`sf apex run`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
[-f FILE]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-f | --file FILE`
Optional

Path to a local file that contains Apex code.

Type: option

Aliases for `apex run`

```
force:apex:execute
```

`apex run test`

Invoke Apex tests in an org.

Description for `apex run test`

Specify which tests to run by using the --class-names, --suite-names, or --tests flags. Alternatively, use the --test-level flag to run all the tests in your org, local tests, or specified tests.

To see code coverage results, use the --code-coverage flag with --result-format. The output displays a high-level summary of the test run and the code coverage values for classes in your org. If you specify human-readable result format, use the --detailed-coverage flag to see detailed coverage results for each test method run.

By default, Apex tests run asynchronously and immediately return a test run ID. You can use the --wait flag to specify the number of minutes to wait; if the tests finish in that timeframe, the command displays the results. If the tests haven't finished by the end of the wait time, the command displays a test run ID. Use the "sf apex get test --test-run-id" command to get the results.

NOTE: The testRunCoverage value (JSON and JUnit result formats) is a percentage of the covered lines and total lines from all the Apex classes evaluated by the tests in this run.

Examples for `apex run test`

Run all Apex tests and suites in your default org:

```
sf apex run test
```

Run the specified Apex test classes in your default org and display results in human-readable form:

```
sf apex run test --class-names MyClassTest --class-names MyOtherClassTest --result-format
 human
```

Run the specified Apex test suites in your default org and include code coverage results and additional details:

```
sf apex run test --suite-names MySuite --suite-names MyOtherSuite --code-coverage
--detailed-coverage
```

Run the specified Apex tests in your default org and display results in human-readable output:

```
sf apex run test --tests MyClassTest.testCoolFeature --tests MyClassTest.testAwesomeFeature
 --tests AnotherClassTest --tests namespace.TheirClassTest.testThis --result-format human
```

Run all tests in the org with the specified username with the specified test level; save the output to the specified directory:

```
sf apex run test --test-level RunLocalTests --output-dir <path to outputdir> --target-org
 me@my.org
```

Usage

`sf apex run test`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
[-c]
[-d OUTPUT-DIR]
[-l TEST-LEVEL]
[-n CLASS-NAMES]
[-r RESULT-FORMAT]
[-s SUITE-NAMES]
[-t TESTS]
[-w WAIT]
[-y]
[-v]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-c | --code-coverage`

Optional

Retrieve code coverage results.

Type: boolean

`-d | --output-dir OUTPUT-DIR`

Optional

Directory in which to store test run files.

Type: option

`-l | --test-level TEST-LEVEL`

Optional

Level of tests to run; default is RunLocalTests.

Here's what the levels mean:

- RunSpecifiedTests — Only the tests that you specify are run.

- RunLocalTests — All tests in your org are run, except the ones that originate from installed managed packages.

- RunAllTestsInOrg — All tests are in your org and in installed managed packages are run

Type: option

Permissible values are: RunLocalTests, RunAllTestsInOrg, RunSpecifiedTests

`-n | --class-names CLASS-NAMES`

Optional

Apex test class names to run; default is all classes.

If you select --class-names, you can't specify --suite-names or --tests.

For multiple classes, repeat the flag for each.

--class-names Class1 --class-names Class2

Type: option

`-r | --result-format RESULT-FORMAT`

Optional

Format of the test results.

Type: option

Permissible values are: human, tap, junit, json

Default value: human

`-s | --suite-names SUITE-NAMES`

Optional

Apex test suite names to run; default is all suites.

If you select --suite-names, you can't specify --class-names or --tests.

For multiple suites, repeat the flag for each.

--suite-names Suite1 --suite-names Suite2

Type: option

`-t | --tests TESTS`

Optional

Apex test class names or IDs and, if applicable, test methods to run; default is all tests.

If you specify --tests, you can't specify --class-names or --suite-names

For multiple tests, repeat the flag for each.

--tests Test1 --tests Test2

Type: option

`-w | --wait WAIT`
Optional

Sets the streaming client socket timeout in minutes; specify a longer wait time if timeouts occur frequently.

Type: option

`-y | --synchronous`
Optional

Runs test methods from a single Apex class synchronously; if not specified, tests are run asynchronously.

Type: boolean

`-v | --detailed-coverage`
Optional

Display detailed code coverage per test.

Type: boolean

Aliases for `apex run test`

```
force:apex:test:run
```

`apex tail log`

Activate debug logging and display logs in the terminal.

Description for `apex tail log`

You can also pipe the logs to a file.

Examples for `apex tail log`

Activate debug logging:

```
sf apex tail log
```

Specify a debug level:

```
sf apex tail log --debug-level MyDebugLevel
```

Skip the trace flag setup and apply default colors:

```
sf apex tail log --color --skip-trace-flag
```

Usage

`sf apex tail log`
```
  [--json]
```

```
-o TARGET-ORG
[--api-version API-VERSION]
[-c]
[-d DEBUG-LEVEL]
[-s]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-c | --color`
Optional

Apply default colors to noteworthy log lines.

Type: boolean

`-d | --debug-level DEBUG-LEVEL`
Optional

Debug level to set on the DEVELOPER_LOG trace flag for your user.

Type: option

`-s | --skip-trace-flag`
Optional

Skip trace flag setup. Assumes that a trace flag and debug level are fully set up.

Type: boolean

Aliases for `apex tail log`

```
force:apex:log:tail
```

`cmdt` Commands

Generate custom metadata types and their records.

cmdt generate field
Generate a field for a custom metadata type based on the provided field type.

cmdt generate fromorg
Generate a custom metadata type and all its records from a Salesforce object.

cmdt generate object
Generate a new custom metadata type in the current project.

cmdt generate record
Generate a new record for a given custom metadata type in the current project.

cmdt generate records
Generate new custom metadata type records from a CSV file.

`cmdt generate field`

Generate a field for a custom metadata type based on the provided field type.

Description for `cmdt generate field`

Similar to a custom object, a custom metadata type has a list of custom fields that represent aspects of the metadata.

This command creates a metadata file that describes the new custom metadata type field. By default, the file is created in a "fields" directory in the current directory. Use the --output-directory to generate the file in the directory that contains the custom metadata type metdata files, such as "force-app/main/default/objects/MyCmdt__mdt" for the custom metadata type called MyCmdt.

Examples for `cmdt generate field`

Generate a metadata file for a custom checkbox field and add the file to the MyCmdt__mdt/fields directory:

```
sf cmdt generate field --name MyCheckboxField --type Checkbox --output-directory
force-app/main/default/objects/MyCmdt__mdt
```

Generate a metadata file for a custom picklist field and add a few values:

```
sf cmdt generate field --name MyPicklistField --type Picklist --picklist-values A
--picklist-values B --picklist-values C --output-directory
force-app/main/default/objects/MyCmdt__mdt
```

Generate a metadata file for a custom number field and specify 2 decimal places:

```
sf cmdt generate field --name MyNumberField --type Number --decimal-places 2
--output-directory force-app/main/default/objects/MyCmdt__mdt
```

Usage

`sf cmdt generate field`

```
[--json]
-n NAME
-f TYPE
[-p PICKLIST-VALUES]
[-s DECIMAL-PLACES]
```

```
[-l LABEL]
[-d OUTPUT-DIRECTORY]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-n | --name NAME`
Required

Unique name for the field.

Type: option

`-f | --type TYPE`
Required

Type of the field.

You can't use this command to create a custom metadata type field of type "Metadata Relationship". Use the Salesforce Setup UI instead.

Type: option

Permissible values are: Checkbox, Date, DateTime, Email, Number, Percent, Phone, Picklist, Text, TextArea, LongTextArea, Url

`-p | --picklist-values PICKLIST-VALUES`
Optional

Picklist values; required for picklist fields.

Type: option

`-s | --decimal-places DECIMAL-PLACES`
Optional

Number of decimal places to use for number or percent fields.

The value must be greater than or equal to zero. Default value is 0.

Type: option

`-l | --label LABEL`
Optional

Label for the field.

Type: option

`-d | --output-directory OUTPUT-DIRECTORY`
Optional

Directory to store newly-created field definition files.

New files are automatically created in the "fields" directory. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Aliases for **`cmdt generate field`**

```
force:cmdt:field:create
```

```
cmdt:field:create
```

`cmdt generate fromorg`

Generate a custom metadata type and all its records from a Salesforce object.

Description for **`cmdt generate fromorg`**

Use this command to migrate existing custom objects or custom settings in an org to custom metadata types. If a field of the Salesforce object is of an unsupported type, the field type is automatically converted to text. Run "sf cmdt generate field --help" to see the list of supported cmdt field types, listed in the --type flag summary. Use the --ignore-unsupported to ignore these fields.

This command creates the metadata files that describe the new custom metadata type and its fields in the "force-app/main/default/objects/TypeName__mdt" directory by default, where "TypeName" is the value of the required --dev-name flag. Use --type-output-directory to create them in a different directory.

Examples for **`cmdt generate fromorg`**

Generate a custom metadata type from a custom object called MySourceObject__c in your default org:

```
sf cmdt generate fromorg --dev-name MyCMDT --sobject MySourceObject__c
```

Generate a custom metadata type from a custom object in an org with alias my-scratch-org; ignore unsupported field types instead of converting them to text:

```
sf cmdt generate fromorg --dev-name MyCMDT --sobject MySourceObject__c --ignore-unsupported
 --target-org my-scratch-org
```

Generate a protected custom metadata type from a custom object:

```
sf cmdt generate fromorg --dev-name MyCMDT --sobject MySourceObject__c --visibility Protected
```

Generate a protected custom metadata type from a custom setting with a specific singular and plural label:

```
sf cmdt generate fromorg --dev-name MyCMDT --label "My CMDT" --plural-label "My CMDTs"
--sobject MySourceSetting__c --visibility Protected
```

Generate a custom metadata type and put the resulting metadata files in the specified directory:

```
sf cmdt generate fromorg --dev-name MyCMDT --sobject MySourceObject__c
--type-output-directory path/to/my/cmdt/directory
```

Generate a custom metadata type and put the resulting record metadata file(s) in the specified directory:

```
sf cmdt generate fromorg --dev-name MyCMDT --sobject MySourceObject__c --records-output-dir
 path/to/my/cmdt/record/directory
```

Usage

```
sf cmdt generate fromorg
  [--json]
```

```
-o TARGET-ORG
[--api-version API-VERSION]
-n DEV-NAME
[-l LABEL]
[-p PLURAL-LABEL]
[-v VISIBILITY]
-s SOBJECT
[-i]
[-d TYPE-OUTPUT-DIRECTORY]
[-r RECORDS-OUTPUT-DIR]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-n | --dev-name DEV-NAME`

Required

Name of the custom metadata type.

Type: option

`-l | --label LABEL`

Optional

Label for the custom metadata type.

Type: option

`-p | --plural-label PLURAL-LABEL`

Optional

Plural version of the label value; if blank, uses label.

Type: option

`-v | --visibility VISIBILITY`

Optional

Who can see the custom metadata type.

For more information on what each option means, see this topic in Salesforce Help: https://help.salesforce.com/s/articleView?id=sf.custommetadatatypes_ui_create.htm&type=5.

Type: option

Permissible values are: PackageProtected, Protected, Public

Default value: Public

`-s | --sobject SOBJECT`
Required

API name of the source Salesforce object used to generate the custom metadata type.

Type: option

`-i | --ignore-unsupported`
Optional

Ignore unsupported field types.

In this context, "ignore" means that the fields aren't created. The default behavior is to create fields of type text and convert the field values to text.

Type: boolean

`-d | --type-output-directory TYPE-OUTPUT-DIRECTORY`
Optional

Directory to store newly-created custom metadata type files.

Type: option

Default value: force-app/main/default/objects

`-r | --records-output-dir RECORDS-OUTPUT-DIR`
Optional

Directory to store newly-created custom metadata record files.

Type: option

Default value: force-app/main/default/customMetadata

Aliases for `cmdt generate fromorg`

```
force:cmdt:generate
```

`cmdt generate object`

Generate a new custom metadata type in the current project.

Description for `cmdt generate object`

This command creates a metadata file that describes the new custom metadata type. By default, the file is created in the MyCustomType__mdt directory in the current directory, where MyCustomType is the value of the required --type-name flag. Use the --output-directory to generate the file in a package directory with other custom metadata types, such as "force-app/main/default/objects".

Examples for `cmdt generate object`

Generate a custom metadata type with developer name 'MyCustomType'; this name is also used as the label:

```
sf cmdt generate object --type-name MyCustomType
```

Generate a protected custom metadata type with a specific label:

```
sf cmdt generate object --type-name MyCustomType --label "Custom Type" --plural-label
"Custom Types" --visibility Protected
```

Usage

`sf cmdt generate object`

`[--json]`

`-n TYPE-NAME`

`[-l LABEL]`

`[-p PLURAL-LABEL]`

`[-v VISIBILITY]`

`[-d OUTPUT-DIRECTORY]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --type-name TYPE-NAME`

Required

Unique object name for the custom metadata type.

The name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Type: option

`-l | --label LABEL`

Optional

Label for the custom metadata type.

Type: option

`-p | --plural-label PLURAL-LABEL`

Optional

Plural version of the label value; if blank, uses label.

Type: option

`-v | --visibility VISIBILITY`

Optional

Who can see the custom metadata type.

For more information on what each option means, see this topic in Salesforce Help: https://help.salesforce.com/s/articleView?id=sf.custommetadatatypes_ui_create.htm&type=5.

Type: option

Permissible values are: PackageProtected, Protected, Public

Default value: Public

`-d | --output-directory OUTPUT-DIRECTORY`

Optional

Directory to store the newly-created custom metadata type files

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Aliases for `cmdt generate object`

```
force:cmdt:create
```

```
cmdt:create
```

`cmdt generate record`

Generate a new record for a given custom metadata type in the current project.

Description for `cmdt generate record`

The custom metadata type must already exist in your project. You must specify a name for the new record. Use name=value pairs to specify the values for the fields, such as MyTextField="some text here" or MyNumberField=32.

Examples for `cmdt generate record`

Create a record metadata file for custom metadata type 'MyCMT' with specified values for two custom fields:

```
sf cmdt generate record --type-name MyCMT__mdt --record-name MyRecord My_Custom_Field_1=Foo
 My_Custom_Field_2=Bar
```

Create a protected record metadata file for custom metadata type 'MyCMT' with a specific label and values specified for two custom fields:

```
sf cmdt generate record --type-name MyCMT__mdt --record-name MyRecord --label "My Record"
 --protected true My_Custom_Field_1=Foo My_Custom_Field_2=Bar
```

Usage

`sf cmdt generate record`

`[--json]`

`-t TYPE-NAME`

`-n RECORD-NAME`

`[-l LABEL]`

`[-p PROTECTED]`

```
[-i INPUT-DIRECTORY]
[-d OUTPUT-DIRECTORY]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-t | --type-name TYPE-NAME`

Required

API name of the custom metadata type to create a record for; must end in "__mdt".

Type: option

`-n | --record-name RECORD-NAME`

Required

Name of the new record.

Type: option

`-l | --label LABEL`

Optional

Label for the new record.

Type: option

`-p | --protected PROTECTED`

Optional

Protect the record when it's in a managed package.

Protected records can only be accessed by code in the same managed package namespace.

Type: option

Permissible values are: true, false

Default value: false

`-i | --input-directory INPUT-DIRECTORY`

Optional

Directory from which to get the custom metadata type definition from.

Type: option

Default value: force-app/main/default/objects

`-d | --output-directory OUTPUT-DIRECTORY`

Optional

Directory to store newly-created custom metadata record files.

Type: option

Default value: force-app/main/default/customMetadata

Aliases for `cmdt generate record`

```
force:cmdt:record:create
```

```
cmdt:record:create
```

`cmdt generate records`

Generate new custom metadata type records from a CSV file.

Description for `cmdt generate records`

The custom metadata type must already exist in your project. By default, the Name column is used to determine the record name; use the --name-column flag to specify a different column.

Examples for `cmdt generate records`

Generate record metadata files from values in a CSV file for the custom metadata type MyCmdt. Use 'Name' as the column that specifies the record name:

```
sf cmdt generate records --csv path/to/my.csv --type-name MyCmdt
```

Generate record metadata files from a CSV file in the directory different from the default, and use 'PrimaryKey' as the column that specifies the record name:

```
sf cmdt generate records --csv path/to/my.csv --type-name MyCmdt --input-directory
path/to/my/cmdt/directory --name-column "PrimaryKey"
```

Usage

`sf cmdt generate records`

`[--json]`

`-f CSV`

`-t TYPE-NAME`

`[-i INPUT-DIRECTORY]`

`[-d OUTPUT-DIRECTORY]`

`[-n NAME-COLUMN]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-f | --csv CSV`

Required

Pathname of the CSV file.

Type: option

`-t | --type-name TYPE-NAME`

Required

API name of the custom metadata type to create a record for.

The '__mdt' suffix is appended to the end of the name if it's omitted.

Type: option

`-i | --input-directory INPUT-DIRECTORY`

Optional

Directory from which to get the custom metadata type definition from.

Type: option

Default value: force-app/main/default/objects

`-d | --output-directory OUTPUT-DIRECTORY`

Optional

Directory to store newly-created custom metadata record files.

Type: option

Default value: force-app/main/default/customMetadata

`-n | --name-column NAME-COLUMN`

Optional

Column used to determine the name of the record.

Type: option

Default value: Name

Aliases for `cmdt generate records`

```
force:cmdt:record:insert
```

```
cmdt:record:insert
```

`community` Commands

Create and publish an Experience Cloud site.

community create

Create an Experience Cloud site using a template.

community list template

Retrieve the list of templates available in your org.

community publish

Publish an Experience Builder site to make it live.

`community create`

Create an Experience Cloud site using a template.

Description for `community create`

Run the "community list template" command to see the templates available in your org. See 'Which Experience Cloud Template Should I Use?' in Salesforce Help for more information about the different template types available. (https://help.salesforce.com/s/articleView?id=sf.siteforce_commtemp_intro.htm&type=5)

When you create a site with the Build Your Own (LWR) template, you must also specify the AuthenticationType value using the format templateParams.AuthenticationType=value, where value is AUTHENTICATED or AUTHENTICATED_WITH_PUBLIC_ACCESS_ENABLED. Name and values are case-sensitive. See 'DigitalExperienceBundle' in the Metadata API Developer Guide for more information. (https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/meta_digitalexperiencebundle.htm)

The site creation process is an async job that generates a jobId. To check the site creation status, query the BackgroundOperation object and enter the jobId as the Id. See 'BackgroundOperation' in the Object Reference for the Salesforce Platform for more information. (https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_backgroundoperation.htm)

If the job doesn't complete within 10 minutes, it times out. You receive an error message and must restart the site creation process. Completed jobs expire after 24 hours and are removed from the database.

When you run this command, it creates the site in preview status, which means that the site isn't yet live. After you finish building your site, you can make it live.

If you have an Experience Builder site, publish the site using the "community publish" command to make it live.

If you have a Salesforce Tabs + Visualforce site, to activate the site and make it live, update the status field of the Network type in Metadata API. (https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/meta_network.htm) Alternatively, in Experience Workspaces, go to Administration | Settings, and click Activate.

For Experience Builder sites, activating the site sends a welcome email to site members.

Examples for `community create`

Create an Experience Cloud site using template 'Customer Service' and URL path prefix 'customers':

```
sf community create --name 'My Customer Site' --template-name 'Customer Service'
--url-path-prefix customers --description 'My customer site'
```

Create a site using 'Partner Central' template:

```
sf community create --name partnercentral --template-name 'Partner Central' --url-path-prefix
 partners
```

Create a site using the 'Build Your Own (LWR)' template with authentication type of UNAUTHENTICATED:

```
sf community create --name lwrsite --template-name 'Build Your Own (LWR)' --url-path-prefix
 lwrsite templateParams.AuthenticationType=UNAUTHENTICATED
```

Usage

`sf community create`

```
  [--json]

  -n NAME

  -t TEMPLATE-NAME

  [-p URL-PATH-PREFIX]

  [-d DESCRIPTION]

  -o TARGET-ORG
```

```
[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the site to create.

Type: option

`-t | --template-name TEMPLATE-NAME`

Required

Template to use to create a site.

An example of a template is Customer Service. Run the "community template list" command to see which templates are available in your org.

Type: option

`-p | --url-path-prefix URL-PATH-PREFIX`

Optional

URL to append to the domain created when Digital Experiences was enabled for this org.

For example, if your domain name is https://MyDomainName.my.site.com and you create a customer site, enter 'customers' to create the unique URL https://MyDomainName.my.site.com/customers.

Type: option

`-d | --description DESCRIPTION`

Optional

Description of the site.

The description displays in Digital Experiences - All Sites in Setup and helps with site identification.

Type: option

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `community create`

```
force:community:create
```

`community list template`

Retrieve the list of templates available in your org.

Description for `community list template`

See 'Which Experience Cloud Template Should I Use?' (https://help.salesforce.com/s/articleView?id=sf.siteforce_commtemp_intro.htm&type=5) in Salesforce Help for more information about the different template types available for Experience Cloud.

Examples for `community list template`

Retrieve the template list from an org with alias my-scratch-org:

```
sf community list template --target-org my-scratch-org
```

Usage

`sf community list template`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `community list template`

```
force:community:template:list
```

`community publish`

Publish an Experience Builder site to make it live.

Description for `community publish`

Each time you publish a site, you update the live site with the most recent updates. When you publish an Experience Builder site for the first time, you make the site's URL live and enable login access for site members.

In addition to publishing, you must activate a site to send a welcome email to all site members. Activation is also required to set up SEO for Experience Builder sites. To activate a site, update the status field of the Network type in Metadata API. (https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/meta_network.htm)Alternatively, in Experience Workspaces, go to Administration | Settings, and click Activate.

An email notification informs you when your changes are live on the published site. The site publish process is an async job that generates a jobId. To check the site publish status manually, query the BackgroundOperation object and enter the jobId as the Id. See 'BackgroundOperation' in the Object Reference for the Salesforce Platform for more information. (https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_backgroundoperation.htm)

If the job doesn't complete within 15 minutes, it times out. You receive an error message and must restart the site publish process. Completed jobs expire after 24 hours and are removed from the database.

Examples for `community publish`

Publish the Experience Builder site with name "My Customer Site':

```
sf community publish --name 'My Customer Site'
```

Usage

`sf community publish`

```
[--json]
-n NAME
-o TARGET-ORG
[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the Experience Builder site to publish.

Type: option

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for **`community publish`**

```
force:community:publish
```

`config` Commands

Commands to configure Salesforce CLI.

config get
Get the value of a configuration variable.

config list
List the configuration variables that you've previously set.

config set
Set one or more configuration variables, such as your default org.

config unset
Unset local or global configuration variables.

`config get`

Get the value of a configuration variable.

Description for `config get`

Run "sf config list" to see all the configuration variables you've set. Global configuration variable are always displayed; local ones are displayed if you run the command in a project directory. Run "sf config set" to set a configuration variable.

Examples for `config get`

Get the value of the "target-org" configuration variable.

```
sf config get target-org
```

Get multiple configuration variables and display whether they're set locally or globally:

```
sf config get target-org api-version --verbose
```

Usage

`sf config get`
`[--json]`
`[--verbose]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

`--verbose`
Optional

Display whether the configuration variables are set locally or globally.

Type: boolean

Aliases for `config get`

```
force:config:get
```

`config list`

List the configuration variables that you've previously set.

Description for `config list`

Global configuration variables apply to any Salesforce DX project and are always displayed. If you run this command from a project directory, local configuration variables are also displayed.

Examples for `config list`

List both global configuration variables and those local to your project:

```
$ sf config list
```

Usage

`sf config list`
`[--json]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

Aliases for `config list`

```
force:config:list
```

`config set`

Set one or more configuration variables, such as your default org.

Description for `config set`

Use configuration variables to set CLI defaults, such as your default org or the API version you want the CLI to use. For example, if you set the "target-org" configuration variable, you don't need to specify it as a "sf deploy metadata" flag if you're deploying to your default org.

Local configuration variables apply only to your current project. Global variables, specified with the --global flag, apply in any Salesforce DX project.

The resolution order if you've set a flag value in multiple ways is as follows:

1. Flag value specified at the command line.

2. Local (project-level) configuration variable.

3. Global configuration variable.

Run "sf config list" to see the configuration variables you've already set and their level (local or global).

Examples for `config set`

Set the local target-org configuration variable to an org username:

```
sf config set target-org=me@my.org
```

Set the local target-org configuration variable to an alias:

```
sf config set target-org=my-scratch-org
```

Set the global target-org configuration variable:

```
sf config set --global target-org=my-scratch-org
```

Set a single configuration variable without using an equal sign; this syntax doesn't work when setting multiple configuration variables:

```
sf config set target-org me@my.com
```

Usage

`sf config set`

`[--json]`

`[-g]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-g | --global`

Optional

Set the configuration variables globally, so they can be used from any Salesforce DX project.

Type: boolean

Aliases for `config set`

```
force:config:set
```

`config unset`

Unset local or global configuration variables.

Description for `config unset`

Local configuration variables apply only to your current project. Global configuration variables apply in any Salesforce DX project.

Examples for `config unset`

Unset the local "target-org" configuration variable:

```
sf config unset target-org
```

Unset multiple configuration variables globally:

```
sf config unset target-org api-version --global
```

Usage

`sf config unset`

`[--json]`

`[-g]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-g | --global`

Optional

Unset the configuration variables globally, so they can no longer be used from any Salesforce DX project.

Type: boolean

Aliases for `config unset`

```
force:config:unset
```

`data` Commands

Manage records in your org.

`data create record`

Create and insert a record into a Salesforce or Tooling API object.

Description for `data create record`

You must specify a value for all required fields of the object.

When specifying fields, use the format <fieldName>=<value>. Enclose all field-value pairs in one set of double quotation marks, delimited by spaces. Enclose values that contain spaces in single quotes.

This command inserts a record into Salesforce objects by default. Use the --use-tooling-api flag to insert into a Tooling API object.

Examples for `data create record`

Insert a record into the Account object of your default org; only the required Name field has a value:

```
sf data create record --sobject Account --values "Name=Acme"
```

Insert an Account record with values for two fields, one value contains a space; the command uses the org with alias "my-scratch":

```
sf data create record --sobject Account --values "Name='Universal Containers'
Website=www.example.com" --target-org my-scratch
```

Insert a record into the Tooling API object TraceFlag:

```
sf data create record --use-tooling-api --sobject TraceFlag --values
"DebugLevelId=7dl170000008U36AAE StartDate=2022-12-15T00:26:04.000+0000
ExpirationDate=2022-12-15T00:56:04.000+0000 LogType=CLASS_TRACING
TracedEntityId=01p17000000R6bLAAS"
```

Usage

`sf data create record`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`-s SOBJECT`

`-v VALUES`

`[-t]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-s | --sobject SOBJECT`

Required

API name of the Salesforce or Tooling API object that you're inserting a record into.

Type: option

`-v | --values VALUES`

Required

Values for the flags in the form <fieldName>=<value>, separate multiple pairs with spaces.

Type: option

`-t | --use-tooling-api`

Optional

Use Tooling API so you can insert a record in a Tooling API object.

Type: boolean

Aliases for `data create record`

```
force:data:record:create
```

`data delete bulk`

Bulk delete records from an org using a CSV file. Uses Bulk API 2.0.

Description for `data delete bulk`

The CSV file must have only one column ("Id") and then the list of record IDs you want to delete, one ID per line.

When you execute this command, it starts a job, displays the ID, and then immediately returns control of the terminal to you by default. If you prefer to wait, set the --wait flag to the number of minutes; if it times out, the command outputs the IDs. Use the job ID to check the status of the job with the "sf data delete resume" command.

Examples for `data delete bulk`

Bulk delete Account records from your default org using the list of IDs in the "files/delete.csv" file:

```
sf data delete bulk --sobject Account --file files/delete.csv
```

Bulk delete records from a custom object in an org with alias my-scratch and wait 5 minutes for the command to complete:

```
sf data delete bulk --sobject MyObject__c --file files/delete.csv --wait 5 --target-org
my-scratch
```

Usage

`sf data delete bulk`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`-f FILE`

`-s SOBJECT`

`[-w WAIT]`

`[-a]`

```
[--verbose]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-f | --file FILE`
Required

CSV file that contains the IDs of the records to delete.

Type: option

`-s | --sobject SOBJECT`
Required

API name of the Salesforce object, either standard or custom, that you want to delete records from.

Type: option

`-w | --wait WAIT`
Optional

Number of minutes to wait for the command to complete before displaying the results.

Type: option

Default value: 0 minutes

`-a | --async`
Optional

Run the command asynchronously.

Type: boolean

`--verbose`
Optional

Print verbose output of failed records if result is available.

Type: boolean

`data delete record`

Deletes a single record from a Salesforce or Tooling API object.

Description for **`data delete record`**

Specify the record you want to delete with either its ID or with a list of field-value pairs that identify the record. If your list of fields identifies more than one record, the delete fails; the error displays how many records were found.

When specifying field-value pairs, use the format <fieldName>=<value>. Enclose all field-value pairs in one set of double quotation marks, delimited by spaces. Enclose values that contain spaces in single quotes.

This command deletes a record from Salesforce objects by default. Use the --use-tooling-api flag to delete from a Tooling API object.

Examples for **`data delete record`**

Delete a record from Account with the specified (truncated) ID:

```
sf data delete record --sobject Account --record-id 00180XX
```

Delete a record from Account whose name equals "Acme":

```
sf data delete record --sobject Account --where "Name=Acme"
```

Delete a record from Account identified with two field values, one that contains a space; the command uses the org with alias "my-scratch":

```
sf data delete record --sobject Account --where "Name='Universal Containers' Phone='(123)
 456-7890'" --target-org myscratch
```

Delete a record from the Tooling API object TraceFlag with the specified (truncated) ID:

```
sf data delete record --use-tooling-api --sobject TraceFlag --record-id 7tf8c
```

Usage

`sf data delete record`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`-s SOBJECT`

`[-i RECORD-ID]`

`[-w WHERE]`

`[-t]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-s | --sobject SOBJECT`

Required

API name of the Salesforce or Tooling API object that you're deleting a record from.

Type: option

`-i | --record-id RECORD-ID`

Optional

ID of the record you're deleting.

Type: option

`-w | --where WHERE`

Optional

List of <fieldName>=<value> pairs that identify the record you want to delete.

Type: option

`-t | --use-tooling-api`

Optional

Use Tooling API so you can delete a record from a Tooling API object.

Type: boolean

Aliases for `data delete record`

```
force:data:record:delete
```

`data delete resume`

Resume a bulk delete job that you previously started. Uses Bulk API 2.0.

Description for `data delete resume`

The command uses the job ID returned by the "sf data delete bulk" command or the most recently-run bulk delete job.

Examples for `data delete resume`

Resume a bulk delete job from your default org using an ID:

```
sf data delete resume --job-id 750xx000000005sAAA
```

Resume the most recently run bulk delete job for an org with alias my-scratch:

```
sf data delete resume --use-most-recent --target-org my-scratch
```

Usage

`sf data delete resume`

```
[--json]
[-o TARGET-ORG]
[-i JOB-ID]
[--use-most-recent]
[--wait WAIT]
[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Optional

Org alias or username to use for the target org.

Type: option

`-i | --job-id JOB-ID`

Optional

ID of the job you want to resume.

Type: option

`--use-most-recent`

Optional

Use the ID of the most recently-run bulk job.

Type: boolean

Default value: true

`--wait WAIT`

Optional

Number of minutes to wait for the command to complete before displaying the results.

Type: option

Default value: 0 minutes

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`data export tree`

Export data from an org into one or more JSON files.

Description for `data export tree`

Specify a SOQL query, either directly at the command line or read from a file, to retrieve the data you want to export. The exported data is written to JSON files in sObject tree format, which is a collection of nested, parent-child records with a single root record. Use these JSON files to import data into an org with the "sf data import tree" command.

If your SOQL query references multiple objects, the command generates a single JSON file by default. You can specify the --plan flag to generate separate JSON files for each object and a plan definition file that aggregates them. You then specify just this plan definition file when you import the data into an org.

The SOQL query can return a maximum of 2,000 records. For more information, see the REST API Developer Guide. (https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/resources_composite_sobject_tree.htm).

Examples for `data export tree`

Export records retrieved with the specified SOQL query into a single JSON file in the current directory; the command uses your default org:

```
sf data export tree --query "SELECT Id, Name, (SELECT Name, Address__c FROM Properties__r)
 FROM Broker__c"
```

Export data using a SOQL query in the "query.txt" file and generate JSON files for each object and a plan that aggregates them:

```
sf data export tree --query query.txt --plan
```

Prepend "export-demo" before each generated file and generate the files in the "export-out" directory; run the command on the org with alias "my-scratch":

```
sf data export tree --query query.txt --plan --prefix export-demo --output-dir export-out
 --target-org my-scratch
```

Usage

`sf data export tree`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`-q QUERY`

`[-p]`

`[-x PREFIX]`

`[-d OUTPUT-DIR]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-q | --query QUERY`

Required

SOQL query, or filepath of a file that contains the query, to retrieve records.

Type: option

`-p | --plan`

Optional

Generate multiple sObject tree files and a plan definition file for aggregated import.

Type: boolean

`-x | --prefix PREFIX`

Optional

Prefix of generated files.

Type: option

`-d | --output-dir OUTPUT-DIR`

Optional

Directory in which to generate the JSON files; default is current directory.

Type: option

Aliases for `data export tree`

```
force:data:tree:export
```

`data get record`

Retrieve and display a single record of a Salesforce or Tooling API object.

Description for `data get record`

Specify the record you want to retrieve with either its ID or with a list of field-value pairs that identify the record. If your list of fields identifies more than one record, the command fails; the error displays how many records were found.

When specifying field-value pairs, use the format <fieldName>=<value>. Enclose all field-value pairs in one set of double quotation marks, delimited by spaces. Enclose values that contain spaces in single quotes.

The command displays all the record's fields and their values, one field per terminal line. Fields with no values are displayed as "null".

This command retrieves a record from Salesforce objects by default. Use the --use-tooling-api flag to retrieve from a Tooling API object.

Examples for `data get record`

Retrieve and display a record from Account with the specified (truncated) ID:

```
sf data get record --sobject Account --record-id 00180XX
```

Retrieve a record from Account whose name equals "Acme":

```
sf data get record --sobject Account --where "Name=Acme"
```

Retrieve a record from Account identified with two field values, one that contains a space; the command uses the org with alias "my-scratch":

```
sf data get record --sobject Account --where "Name='Universal Containers' Phone='(123)
456-7890'" --target-org myscratch
```

Retrieve a record from the Tooling API object TraceFlag with the specified (truncated) ID:

```
sf data get record --use-tooling-api --sobject TraceFlag --record-id 7tf8c
```

Usage

`sf data get record`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
-s SOBJECT
[-i RECORD-ID]
[-w WHERE]
[-t]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-s | --sobject SOBJECT`

Required

API name of the Salesforce or Tooling API object that you're retrieving a record from.

Type: option

`-i | --record-id RECORD-ID`
Optional

ID of the record you’re retrieving.

Type: option

`-w | --where WHERE`
Optional

List of <fieldName>=<value> pairs that identify the record you want to display.

Type: option

`-t | --use-tooling-api`
Optional

Use Tooling API so you can retrieve a record from a Tooling API object.

Type: boolean

Aliases for `data get record`

```
force:data:record:get
```

`data import tree`

Import data from one or more JSON files into an org.

Description for `data import tree`

The JSON files that contain the data are in sObject tree format, which is a collection of nested, parent-child records with a single root record. Use the "sf data export tree" command to generate these JSON files.

If you used the --plan flag when exporting the data to generate a plan definition file, use the --plan flag to reference the file when you import. If you're not using a plan, use the --files flag to list the files. If you specify multiple JSON files that depend on each other in a parent-child relationship, be sure you list them in the correct order.

The sObject Tree API supports requests that contain up to 200 records. For more information, see the REST API Developer Guide. (https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/resources_composite_sobject_tree.htm)

Examples for `data import tree`

Import the records contained in two JSON files into the org with alias "my-scratch":

```
sf data import tree --files Contact.json,Account.json --target-org my-scratch
```

Import records using a plan definition file into your default org:

```
sf data import tree --plan Account-Contact-plan.json
```

Usage

`sf data import tree`

```
[--json]

-o TARGET-ORG

[--api-version API-VERSION]

[-f FILES]

[-p PLAN]

[--config-help]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-f | --files FILES`

Optional

Comma-separated and in-order JSON files that contain the records, in sObject tree format, that you want to insert.

Type: option

`-p | --plan PLAN`

Optional

Plan definition file to insert multiple data files.

Type: option

`--config-help`

Optional

Display schema information for the --plan configuration file to stdout; if you specify this flag, all other flags except --json are ignored.

Type: boolean

Aliases for `data import tree`

```
force:data:tree:import
```

data query

Execute a SOQL query.

Description for data query

Specify the SOQL query at the command line with the --query flag or read the query from a file with the --file flag.

If your query returns more than 10,000 records, specify the --bulk flag. The command then runs the query using Bulk API 2.0, which has higher limits than the default API used by the command.

When using --bulk, the command waits 3 minutes by default for the query to complete. Use the --wait parameter to specify a different number of minutes to wait, or set --wait to 0 to immediately return control to the terminal. If you set --wait to 0, or you use the --async flag, or the command simply times out, the command displays an ID. Pass this ID to the the "data query resume" command using the --bulk-query-id flag to get the results; pass the ID to the "data resume" command to get the job status.

Examples for data query

Specify a SOQL query at the command line; the command uses your default org:

```
sf data query --query "SELECT Id, Name, Account.Name FROM Contact"
```

Read the SOQL query from a file called "query.txt"; the command uses the org with alias "my-scratch":

```
sf data query --file query.txt --target-org my-scratch
```

Use Tooling API to run a query on the ApexTrigger Tooling API object:

```
sf data query --query "SELECT Name FROM ApexTrigger" --use-tooling-api
```

Use Bulk API 2.0 to run a query that returns many rows, and return control to the terminal immediately:

```
sf data query --query "SELECT Id FROM Contact" --bulk --wait 0
```

Usage

sf data query

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
[-q QUERY]
[-f FILE]
[-t]
[-b]
[-w WAIT]
[--async]
[--all-rows]
[-r RESULT-FORMAT]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-q | --query QUERY`
Optional

SOQL query to execute.

Type: option

`-f | --file FILE`
Optional

File that contains the SOQL query.

Type: option

`-t | --use-tooling-api`
Optional

Use Tooling API so you can run queries on Tooling API objects.

Type: boolean

`-b | --bulk`
Optional

Use Bulk API 2.0 to run the query.

Type: boolean

`-w | --wait WAIT`
Optional

Time to wait for the command to finish, in minutes.

Type: option

`--async`
Optional

Use Bulk API 2.0, but don't wait for the job to complete.

Type: boolean

`--all-rows`
Optional

Include deleted records. By default, deleted records are not returned.

Type: boolean

`-r | --result-format RESULT-FORMAT`

Optional

Format to display the results; the --json flag overrides this flag.

Type: option

Permissible values are: human, json, csv

Default value: human

Aliases for `data query`

```
force:data:soql:query
```

`data query resume`

View the status of a bulk query.

Description for `data query resume`

Run this command using the job ID returned from the "sf data query --bulk" command.

Examples for `data query resume`

View the status of a bulk query with the specified ID:

```
sf data query resume --bulk-query-id 7500x000005BdFzXXX
```

Usage

`sf data query resume`

`[--json]`

`[-o TARGET-ORG]`

`[--api-version API-VERSION]`

`[-r RESULT-FORMAT]`

`[-i BULK-QUERY-ID]`

`[--use-most-recent]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

-o | --target-org TARGET-ORG
Optional

Org alias or username to use for the target org.

Type: option

--api-version API-VERSION
Optional

Override the api version used for api requests made by this command

Type: option

-r | --result-format RESULT-FORMAT
Optional

Format to display the results; the --json flag overrides this flag.

Type: option

Permissible values are: human, json, csv

Default value: human

-i | --bulk-query-id BULK-QUERY-ID
Optional

Job ID of the bulk query.

Type: option

--use-most-recent
Optional

Use the most recent bulk query ID from cache.

Type: boolean

Aliases for `data query resume`

```
force:data:soql:bulk:report
```

`data resume` (Deprecated)

The command `data resume` has been deprecated. View the status of a bulk data load job or batch.

Description for `data resume`

Run this command using the job ID or batch ID returned from the "sf data delete bulk" or "sf data upsert bulk" commands.

Examples for `data resume`

View the status of a bulk load job:

```
sf data resume --job-id 750xx000000005sAAA
```

View the status of a bulk load job and a specific batches:

```
sf data resume --job-id 750xx000000005sAAA --batch-id 751xx000000005nAAA
```

Usage

```
sf data resume
   [--json]
   -o TARGET-ORG
   [--api-version API-VERSION]
   [-b BATCH-ID]
   -i JOB-ID
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-b | --batch-id BATCH-ID`
Optional

ID of the batch whose status you want to view; you must also specify the job ID.

Type: option

`-i | --job-id JOB-ID`
Required

ID of the job whose status you want to view.

Type: option

`data update record`

Updates a single record of a Salesforce or Tooling API object.

Description for `data update record`

Specify the record you want to update with either its ID or with a list of field-value pairs that identify the record. If your list of fields identifies more than one record, the delete fails; the error displays how many records were found.

When using field-value pairs for both identifying the record and specifiyng the new field values, use the format <fieldName>=<value>. Enclose all field-value pairs in one set of double quotation marks, delimited by spaces. Enclose values that contain spaces in single quotes.

This command updates a record in Salesforce objects by default. Use the --use-tooling-api flag to update a Tooling API object.

Examples for `data update record`

Update the Name field of an Account record with the specified (truncated) ID:

```
sf data update record --sobject Account --record-id 001D0 --values "Name=NewAcme"
```

Update the Name field of an Account record whose current name is 'Old Acme':

```
sf data update record --sobject Account --where "Name='Old Acme'" --values "Name='New
Acme'"
```

Update the Name and Website fields of an Account record with the specified (truncated) ID:

```
sf data update record --sobject Account --record-id 001D0 --values "Name='Acme III'
Website=www.example.com"
```

Update the ExpirationDate field of a record of the Tooling API object TraceFlag using the specified (truncated) ID:

```
sf data update record -t --sobject TraceFlag --record-id 7tf170000009cUBAAY --values
"ExpirationDate=2017-12-01T00:58:04.000+0000"
```

Usage

`sf data update record`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`-s SOBJECT`

`[-i RECORD-ID]`

`[-w WHERE]`

`-v VALUES`

`[-t]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

-s | --sobject SOBJECT
Required

API name of the Salesforce or Tooling API object that contains the record you're updating.

Type: option

-i | --record-id RECORD-ID
Optional

ID of the record you're updating.

Type: option

-w | --where WHERE
Optional

List of <fieldName>=<value> pairs that identify the record you want to update.

Type: option

-v | --values VALUES
Required

Fields that you're updating, in the format of <fieldName>=<value> pairs.

Type: option

-t | --use-tooling-api
Optional

Use Tooling API so you can update a record in a Tooling API object.

Type: boolean

Aliases for `data update record`

```
force:data:record:update
```

data upsert bulk

Bulk upsert records to an org from a CSV file. Uses Bulk API 2.0.

Description for `data upsert bulk`

An upsert refers to inserting a record into a Salesforce object if the record doesn't already exist, or updating it if it does exist.

When you execute this command, it starts a job, displays the ID, and then immediately returns control of the terminal to you by default. If you prefer to wait, set the --wait flag to the number of minutes; if it times out, the command outputs the IDs. Use the job and batch IDs to check the status of the job with the "sf data upsert resume" command.

See "Prepare CSV Files" in the Bulk API Developer Guide for details on formatting your CSV file. (https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/datafiles_prepare_csv.htm)

Examples for `data upsert bulk`

Bulk upsert records to the Contact object in your default org:

```
sf data upsert bulk --sobject Contact --file files/contacts.csv --external-id Id
```

Bulk upsert records to a custom object in an org with alias my-scratch and wait 5 minutes for the command to complete:

```
sf data upsert bulk --sobject MyObject__c --file files/file.csv --external-id MyField__c
--wait 5 --target-org my-scratch
```

Usage

`sf data upsert bulk`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`-f FILE`

`-s SOBJECT`

`[-w WAIT]`

`[-a]`

`[--verbose]`

`-i EXTERNAL-ID`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-f | --file FILE`

Required

CSV file that contains the IDs of the records to delete.

Type: option

`-s | --sobject SOBJECT`

Required

API name of the Salesforce object, either standard or custom, that you want to delete records from.

Type: option

`-w | --wait WAIT`

Optional

Number of minutes to wait for the command to complete before displaying the results.

Type: option

Default value: 0 minutes

`-a | --async`

Optional

Run the command asynchronously.

Type: boolean

`--verbose`

Optional

Print verbose output of failed records if result is available.

Type: boolean

`-i | --external-id EXTERNAL-ID`

Required

Name of the external ID field, or the Id field.

Type: option

`data upsert resume`

Resume a bulk upsert job that you previously started. Uses Bulk API 2.0.

Description for `data upsert resume`

The command uses the job ID returned from the "sf data upsert bulk" command or the most recently-run bulk upsert job.

Examples for `data upsert resume`

Resume a bulk upsert job from your default org using an ID:

```
sf data upsert resume --job-id 750xx000000005sAAA
```

Resume the most recently run bulk upsert job for an org with alias my-scratch:

```
sf data upsert resume --use-most-recent --target-org my-scratch
```

Usage

`sf data upsert resume`

`[--json]`

`[-o TARGET-ORG]`

`[-i JOB-ID]`

`[--use-most-recent]`

`[--wait WAIT]`

`[--api-version API-VERSION]`

Flags

`--json`
: Optional

  Format output as json.

  Type: boolean

`-o | --target-org TARGET-ORG`
: Optional

  Org alias or username to use for the target org.

  Type: option

`-i | --job-id JOB-ID`
: Optional

  ID of the job you want to resume.

  Type: option

`--use-most-recent`
: Optional

  Use the ID of the most recently-run bulk job.

  Type: boolean

  Default value: true

`--wait WAIT`
: Optional

  Number of minutes to wait for the command to complete before displaying the results.

  Type: option

  Default value: 0 minutes

`--api-version API-VERSION`
: Optional

  Override the api version used for api requests made by this command

  Type: option

`deploy` Commands

Commands to deploy artifacts to an environment.

deploy functions
: Deploy a Salesforce Function to an org from your local project.

`deploy functions`

Deploy a Salesforce Function to an org from your local project.

Description for `deploy functions`

You must run this command from within a git repository. Only committed changes to Functions are deployed. The active branch is deployed unless specified otherwise with `--branch`.

Examples for `deploy functions`

Deploy a Salesforce Function:

```
sf deploy functions --connected-org org-alias
```

Deploy to 'deploy-branch':

```
sf deploy functions --connected-org org-alias --branch deploy-branch
```

Overwrite the remote repository:

```
sf deploy functions --connected-org org-alias --force
```

Usage

`sf deploy functions`

`[--json]`

`-o CONNECTED-ORG`

`[-b BRANCH]`

`[--force]`

`[-q]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --connected-org CONNECTED-ORG`

Required

Username or alias for the org that the compute environment should be connected to.

Type: option

`-b | --branch BRANCH`

Optional

Deploy the latest commit from a branch different from the currently active branch.

Type: option

`--force`

Optional

Ignore warnings and overwrite remote repository (not allowed in production).

Type: boolean

-q | --quiet

Optional

Limit the amount of output displayed from the deploy process.

Type: boolean

`dev` Commands

Commands for sf plugin development.

dev audit messages

Audit messages in a plugin's messages directory to locate unused messages and missing messages that have references in source code.

dev configure repo

Configure a GitHub repo for the GitHub Actions pipeline.

dev configure secrets

Ensures a GitHub repo has correct access to secrets based on its workflows.

dev convert messages

Convert a .json messages file into Markdown.

dev convert script

Convert a script file that contains deprecated sfdx-style commands to use the new sf-style commands instead.

dev generate command

Generate a new sf command.

dev generate flag

Generate a flag for an existing command.

dev generate library

Generate a new library.

dev generate plugin

Generate a new sf plugin.

`dev audit messages`

Audit messages in a plugin's messages directory to locate unused messages and missing messages that have references in source code.

Examples for `dev audit messages`

Audit messages using default directories:

```
sf dev audit messages
```

Audit messages in the "messages" directory in the current working directory; the plugin's source directory is in "src":

```
sf dev audit messages --messages-dir ./messages --source-dir ./src
```

Usage

`sf dev audit messages`

`[--json]`

`[-p PROJECT-DIR]`

`[-m MESSAGES-DIR]`

`[-s SOURCE-DIR]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-p | --project-dir PROJECT-DIR`

Optional

Location of the project where messages are to be audited.

Type: option

Default value: .

`-m | --messages-dir MESSAGES-DIR`

Optional

Directory that contains the plugin's message files.

The default is the "messages" directory in the current working directory.

Type: option

Default value: messages

`-s | --source-dir SOURCE-DIR`

Optional

Directory that contains the plugin's source code.

The default is the "src" directory in the current working directory.

Type: option

Default value: src

`dev configure repo`

Configure a GitHub repo for the GitHub Actions pipeline.

Description for `dev configure repo`

Sets up labels and exempts the CLI bot for branch protection and PR rules.

Examples for `dev configure repo`

Configure the repo "testPackageRelease", with owner "salesforcecli", for GitHub Actions.

```
sf dev configure repo --repository salesforcecli/testPackageRelease
```

Usage

`sf dev configure repo`

```
[--json]
-r REPOSITORY
[-d]
[-b BOT]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-r | --repository REPOSITORY`

Required

GitHub owner/repo for which you want to configure GitHub Actions.

Type: option

`-d | --dry-run`

Optional

Make no changes.

Type: boolean

`-b | --bot BOT`

Optional

GitHub login/username for the bot.

Type: option

Default value: SF-CLI-BOT

`dev configure secrets`

Ensures a GitHub repo has correct access to secrets based on its workflows.

Description for `dev configure secrets`

Inspects a repo's yaml files and verifies that secrets required are available for the repo (either set at the repo level or shared via organization-level secrets).

This command requires scope:admin permissions to inspect the org secrets and admin access to the repo to inspect the repo secrets.

Examples for `dev configure secrets`

Ensure secrets access for the repo "testPackageRelease", with owner "salesforcecli":

```
sf dev configure secrets --repository salesforcecli/testPackageRelease
```

Usage

`sf dev configure secrets`

`[--json]`

`-r REPOSITORY`

`[-d]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-r | --repository REPOSITORY`

Required

Github owner/repo.

Type: option

`-d | --dry-run`

Optional

Make no changes.

Type: boolean

`dev convert messages`

Convert a .json messages file into Markdown.

Description for `dev convert messages`

Preserves the filename and the original messages file, then creates a new file with the Markdown extension and standard headers for the command and flag summaries, descriptions, and so on. After you review the new Markdown file, delete the old .json file.

Examples for `dev convert messages`

Convert the my-command.json message file into my-command.md with the standard messages headers:

```
sf dev convert messages --filename my-command.json
```

Similar to previous example, but specify the plugin project directory:

```
sf dev convert messages --project-dir ./path/to/plugin --filename my-command.json
```

Usage

```
sf dev convert messages
   [--json]
   [-p PROJECT-DIR]
   -f FILE-NAME
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-p | --project-dir PROJECT-DIR`

Optional

Location of the project whose messages are to be converted.

Type: option

Default value: .

`-f | --file-name FILE-NAME`

Required

Filename to convert.

Type: option

`dev convert script`

Convert a script file that contains deprecated sfdx-style commands to use the new sf-style commands instead.

Description for `dev convert script`

Important: Use this command only to get started on the sfdx->sf script migration. We don't guarantee that the new sf-style command replacements work correctly or as you expect. You must test, and probably update, the new script before putting it into production. We also don't guarantee that the JSON results are the same as before.

This command can convert a large part of your script, but possibly not all. There are some sfdx-style commands that don't have an obvious sf-style equivalent. In this case, this command doesn't replace the sfdx-style command but instead adds a comment to remind you that you must convert it manually. See the Salesforce CLI Command Reference for migration information about each deprecated sfdx-style command: https://developer.salesforce.com/docs/atlas.en-us.sfdx_cli_reference.meta/sfdx_cli_reference/cli_reference.htm.

This command is interactive; as it scans your script, it prompts you when it finds an sfdx-style command or flag and asks if you want to convert it to the displayed suggestion. The command doesn't update the script file directly; rather, it creates a new file whose name is the original name but with "-converted" appended to it. The script replaces all instances of "sfdx" with "sf". For each prompt you answer "y" to, the command replaces the sfdx-style names with their equivalent sf-style ones. For example, "sfdx force:apex:execute --targetusername myscratch" is replaced with "sf apex run --target-org myscratch".

Examples for `dev convert script`

Convert the YAML file called "myScript.yml" located in the current directory; the new file that contains the replacements is called "myScript-converted.yml":

```
sf dev convert script --script ./myScript.yml
```

Usage

`sf dev convert script`

`[--json]`

`-s SCRIPT`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-s | --script SCRIPT`

Required

Filepath to the script you want to convert.

Type: option

`dev generate command`

Generate a new sf command.

Description for `dev generate command`

You must run this command from within a plugin directory, such as the directory created with the "sf dev generate plugin" command.

The command generates basic source files, messages (*.md), and test files for your new command. The Typescript files contain import statements for the minimum required Salesforce libraries, and scaffold some basic code. The new type names come from the value you passed to the --name flag.

The command updates the package.json file, so if it detects conflicts with the existing file, you're prompted whether you want to overwrite the file. There are a number of package.json updates required for a new command, so we recommend you answer "y" so the command takes care of them all. If you answer "n", you must update the package.json file manually.

Examples for `dev generate command`

Generate the files for a new "sf my exciting command":

```
sf dev generate command --name my:exciting:command
```

Usage

`sf dev generate command`

`[--json]`

```
-n NAME
[--force]
[--nuts]
[--unit]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-n | --name NAME`
Required

Name of the new command. Use colons to separate the topic and command names.

Type: option

`--force`
Optional

Overwrite existing files.

Type: boolean

`--nuts`
Optional

Generate a NUT test file for the command.

Type: boolean

Default value: true

`--unit`
Optional

Generate a unit test file for the command.

Type: boolean

Default value: true

`dev generate flag`

Generate a flag for an existing command.

Description for `dev generate flag`

You must run this command from within a plugin directory, such as the directory created with the "sf dev generate plugin" command.

This command is interactive. It first discovers all the commands currently implemented in the plugin, and asks you which you want to create a new flag for. It then prompts for other flag properties, such as its long name, optional short name, type, whether it's required, and so on. Long flag names must be kebab-case and not camelCase. The command doesn't let you use an existing long or short flag name. When the command completes, the Typescript file for the command is updated with the code for the new flag.

Use the --dry-run flag to review new code for the command file without actually udpating it.

Examples for `dev generate flag`

Generate a new flag and update the command file:

```
sf dev generate flag
```

Don't actually update the command file, just view the generated code:

```
sf dev generate flag --dry-run
```

Usage

`sf dev generate flag`

`[--json]`

`[-d]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-d | --dry-run`

Optional

Print new flag code instead of adding it to the command file.

Type: boolean

`dev generate library`

Generate a new library.

Description for `dev generate library`

This command is interactive. You're prompted for information to populate the new library, such as the npm scope (which must start with "@"), the name and description of the library, and its GitHub organization. The command clones the 'forcedotcom/library-template' GitHub repository, installs the library's npm package dependencies using yarn install, and updates the package properties.

When the command completes, your new library contains a few sample source and test files to get you started.

Examples for `dev generate library`

```
sf dev generate library
```

Usage

`sf dev generate library`

`[--json]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

`dev generate plugin`

Generate a new sf plugin.

Description for `dev generate plugin`

This command is interactive. You're prompted for information to populate your new plugin, such as its name, description, author, and percentage of code coverage you want. The command clones the 'salesforcecli/plugin-template-sf' GitHub repository, installs the plug-in's npm package dependencies using yarn install, and updates the package properties.

When the command completes, your new plugin contains the source, message, and test files for a sample "sf hello world" command.

Examples for `dev generate plugin`

```
sf dev generate plugin
```

Usage

`sf dev generate plugin`
`[--json]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

Aliases for `dev generate plugin`

```
plugins:generate
```

`doctor` Commands

Tools for diagnosing problems with Salesforce CLI.

doctor
Gather CLI configuration data and run diagnostic tests to discover and report potential problems in your environment.

doctor

Gather CLI configuration data and run diagnostic tests to discover and report potential problems in your environment.

Description for doctor

When you run the doctor command without parameters, it first displays a diagnostic overview of your environment. It then writes a detailed diagnosis to a JSON file in the current directory. Use the --outputdir to specify a different directory. To run diagnostic tests on a specific plugin, use the --plugin parameter. If the plugin isn't listening to the doctor, then you get a warning.

Use the --command parameter to run a specific command in debug mode; the doctor writes both stdout and stderr to *.log files that you can provide to Salesforce Customer Support or attach to a GitHub issue.

Plugin providers can also implement their own doctor diagnostic tests by listening to the "sf-doctor" event and running plugin specific tests that are then included in the doctor diagnostics log.

Examples for doctor

Run CLI doctor diagnostics:

```
sf doctor
```

Run CLI doctor diagnostics and the specified command, and write the debug output to a file:

```
sf doctor --command "force:org:list --all"
```

Run CLI doctor diagnostics for a specific plugin:

```
sf doctor --plugin @salesforce/plugin-source
```

Usage

sf doctor

[--json]

[-c COMMAND]

[-p PLUGIN]

[-d OUTPUT-DIR]

[-i]

Flags

--json

Optional

Format output as json.

Type: boolean

-c | --command COMMAND

Optional

Command to run in debug mode; results are written to a log file.

Type: option

`-p | --plugin PLUGIN`

Optional

Specific plugin on which to run diagnostics.

Type: option

`-d | --output-dir OUTPUT-DIR`

Optional

Directory to save all created files rather than the current working directory.

Type: option

`-i | --create-issue`

Optional

Create a new issue on our GitHub repo and attach all diagnostic results.

Type: boolean

`env` Commands

Commands to manage your environments, such as orgs and compute environments.

env compute collaborator add

Add a Heroku user as a collaborator on this Functions account, allowing them to attach Heroku add-ons to compute environments.

env create compute

Create a compute environment for use with Salesforce Functions.

env delete

Delete an environment.

env display

Display details about an environment.

env list

List the environments you’ve created or logged into.

env log (Beta)

Stream log output for an environment.

env log tail

Stream log output for an environment.

env logdrain add

Add log drain to a specified environment.

env logdrain list

List log drains connected to a specified environment.

env logdrain remove

Remove log drain from a specified environment.

env open

Open an environment in a web browser.

env var get
Display a single config variable for an environment.

env var list
List your environment's config vars in a table.

env var set
Set a single config value for an environment.

env var unset
Unset a single config value for an environment.

`env compute collaborator add`

Add a Heroku user as a collaborator on this Functions account, allowing them to attach Heroku add-ons to compute environments.

Examples for `env compute collaborator add`

Add a Heroku user as a collaborator on this Functions account.

```
sf env compute collaborator add --heroku-user example@heroku.com
```

Usage

`sf env compute collaborator add`

`[--json]`

`-h HEROKU-USER`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-h | --heroku-user HEROKU-USER`

Required

Email address of the Heroku user you're adding as a collaborator.

Type: option

`env create compute`

Create a compute environment for use with Salesforce Functions.

Description for `env create compute`

Compute environments must be connected to a Salesforce org. By default the command uses your local environment's connected org. Use the '--connected-org' flag to specify a specific org. Run 'sf env list' to see a list of environments.

Examples for `env create compute`

Create a compute environment to run Salesforce Functions:

```
sf env create compute
```

Connect the environment to a specific org:

```
sf env create compute --connected-org=org-alias
```

Create an alias for the compute environment:

```
sf env create compute --alias environment-alias
```

Usage

`sf env create compute`

`[--json]`

`[-o CONNECTED-ORG]`

`[-a ALIAS]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --connected-org CONNECTED-ORG`

Optional

Username or alias for the org that the compute environment should be connected to.

Type: option

`-a | --alias ALIAS`

Optional

Alias for the created environment.

Type: option

`env delete`

Delete an environment.

Description for `env delete`

You must include the name of the environment to delete using '--target-compute'. Run 'sf env list' to see a list of environments.

Running this command will prompt a confirmation. If you want to skip this confirmation, use the '--confirm' flag and the environment alias to skip confirmation.

Examples for `env delete`

Delete a compute environment:

```
sf env delete --target-compute environment-alias
```

Delete without a confirmation step:

```
sf env delete --target-compute environment-alias --confirm environment-alias
```

Usage

`sf env delete`
  `[--json]`
  `[-e TARGET-COMPUTE]`
  `[--confirm CONFIRM]`

Flags

`--json`
  Optional
  Format output as json.
  Type: boolean

`-e | --target-compute TARGET-COMPUTE`
  Optional
  Environment name.
  Type: option

`--confirm CONFIRM`
  Optional
  Confirmation name.
  Type: option

`env display`

Display details about an environment.

Description for `env display`

Specify an environment with either the username you used when you logged into the environment with "sf login", or the alias you gave the environment when you created it. Run "sf env list" to view all your environments and their aliases.

Output depends on the type of environment.

Examples for `env display`

Display details about the "myEnv" environment:

<%- config.bin %> <%- command.id %> --target-env myEnv

Usage

`sf env display`
`[--json]`
`[-e TARGET-ENV]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-e | --target-env TARGET-ENV`
Optional

Environment alias or login user.

Type: option

`env list`

List the environments you've created or logged into.

Description for `env list`

By default, the command displays active environments.

Output is displayed in multiple tables, one for each environment type.

The compute environment table shows the alias, information about the connected orgs, the project name, and more.

Use the table manipulation flags, such as --filter and --sort, to change how the data is displayed.

Run "sf env display" to view details about a specific environment.

Examples for `env list`

List all active environments:

```
sf env list
```

List both active and inactive environments:

```
sf env list --all
```

Don't truncate the displayed output and instead wrap text that's wider than your terminal:

```
sf env list --no-truncate
```

Display only the table data, not the headers, in comma-separated value (csv) format:

```
sf env list --csv --no-header
```

Usage

```
sf env list
   [--json]
   [-a]
   [--columns COLUMNS]
   [--csv]
   [--filter FILTER]
   [--no-header]
   [--no-truncate]
   [--output OUTPUT]
   [--sort SORT]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-a | --all`

Optional

Show all environments, even inactive ones.

Type: boolean

`--columns COLUMNS`

Optional

List of columns to display.

Type: option

`--csv`

Optional

Output in csv format [alias: --output=csv]

Type: boolean

`--filter FILTER`

Optional

Filter property by partial string matching.

Type: option

`--no-header`

Optional

Hide table header from output.

Type: boolean

`--no-truncate`

Optional

Don't truncate output to fit screen.

Type: boolean

`--output OUTPUT`

Optional

Format in which to display the output.

Type: option

Permissible values are: csv, json, yaml

`--sort SORT`

Optional

Column to sort by (prepend '-' for descending).

Type: option

`env log` (Beta)

Stream log output for an environment.

Note: This feature is a Beta Service. Customers may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms (https://www.salesforce.com/company/legal/agreements/).

Examples for `env log`

Stream log output:

```
sf env log --target-compute environment-alias
```

Usage

`sf env log`

`[--json]`

`[-e TARGET-COMPUTE]`

`[-n NUM]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-e | --target-compute TARGET-COMPUTE`

Optional

Compute environment name to retrieve logs.

Type: option

-n | --num NUM

Optional

Number of lines to display.

Type: option

env log tail

Stream log output for an environment.

Examples for env log tail

Stream log output:

```
sf env log tail --target-compute environment-alias
```

Usage

sf env log tail

[--json]

[-e TARGET-COMPUTE]

Flags

--json

Optional

Format output as json.

Type: boolean

-e | --target-compute TARGET-COMPUTE

Optional

Compute environment name to retrieve logs.

Type: option

env logdrain add

Add log drain to a specified environment.

Description for env logdrain add

Both '--target-compute' and '--url' are required flags. '--url' should be a HTTP or HTTPS URL that can receive the log drain messages.

Examples for env logdrain add

Add a log drain:

```
sf env logdrain add --target-compute environment-name --url https://path/to/logdrain
```

Usage

`sf env logdrain add`

[`--json`]

[`-e TARGET-COMPUTE`]

[`-l DRAIN-URL`]

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-e | --target-compute TARGET-COMPUTE`

Optional

Environment name.

Type: option

`-l | --drain-url DRAIN-URL`

Optional

Endpoint that will receive sent logs.

Type: option

`env logdrain list`

List log drains connected to a specified environment.

Examples for `env logdrain list`

List log drains:

```
sf env logdrain list --target-compute environment-alias
```

List log drains as json:

```
sf env logdrain list --target-compute environment-alias --json
```

Usage

`sf env logdrain list`

[`--json`]

[`-e TARGET-COMPUTE`]

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-e | --target-compute TARGET-COMPUTE`

Optional

Environment name.

Type: option

`env logdrain remove`

Remove log drain from a specified environment.

Description for `env logdrain remove`

Both '--target-compute' and '--drain-url' are required flags.

Examples for `env logdrain remove`

Remove a logdrain:

```
sf env logdrain remove --target-compute environment-alias --url https://path/to/logdrain
```

Usage

`sf env logdrain remove`

`[--json]`

`[-e TARGET-COMPUTE]`

`[-l DRAIN-URL]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-e | --target-compute TARGET-COMPUTE`

Optional

Environment name.

Type: option

`-l | --drain-url DRAIN-URL`

Optional

Log drain url to remove.

Type: option

`env open`

Open an environment in a web browser.

Description for `env open`

Each of your environments is associated with an instance URL, such as https://login.salesforce.com. To open a specific web page, specify the portion of the URL after "<URL>/" with the --path flag.

Examples for `env open`

Open the compute environment with alias "test-compute":

```
sf env open --target-env test-compute
```

View the URL but don't launch it in a browser:

```
sf env open --target-env test-compute --url-only
```

Open the environment in the Google Chrome browser:

```
sf env open --target-env test-compute --url-only --browser chrome
```

Usage

`sf env open`

`[--json]`

`[-p PATH]`

`[-r]`

`[-e TARGET-ENV]`

`[--browser BROWSER]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-p | --path PATH`

Optional

Path to append to the end of the login URL.

Type: option

`-r | --url-only`

Optional

Display the URL, but don’t launch it in a browser.

Type: boolean

`-e | --target-env TARGET-ENV`

Optional

Login user or alias of the environment to open.

Type: option

--browser BROWSER

Optional

Browser in which to open the environment.

You can specify that the environment open in one of the following browsers: Firefox, Safari, Google Chrome, or Windows Edge. If you don't specify --browser, the environment opens in your default browser. The exact names of the browser applications differ depending on the operating system you're on; check your documentation for details.

Type: option

env var get

Display a single config variable for an environment.

Description for env var get

You must provide the '--target-compute' flag and the key to retrieve.

Examples for env var get

Get a config variable:

```
sf env var get [KEY] --target-compute environment-alias
```

Usage

sf env var get

`[--json]`

`[-e TARGET-COMPUTE]`

Flags

--json

Optional

Format output as json.

Type: boolean

-e | --target-compute TARGET-COMPUTE

Optional

Environment name.

Type: option

env var list

List your environment's config vars in a table.

Description for env var list

Use the '--json' flag to return config vars in JSON format.

Examples for **`env var list`**

List config vars:

```
sf env var list --target-compute environment-alias
```

List in JSON format:

```
sf env var list --target-compute environment-alias --json
```

Usage

`sf env var list`

`[--json]`

`[-e TARGET-COMPUTE]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-e | --target-compute TARGET-COMPUTE`

Optional

Environment name.

Type: option

`env var set`

Set a single config value for an environment.

Examples for **`env var set`**

Set a config value:

```
sf env var set [KEY]=[VALUE] --target-compute environment-alias
```

Usage

`sf env var set`

`[--json]`

`[-e TARGET-COMPUTE]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

-e | --target-compute TARGET-COMPUTE

Optional

Environment name.

Type: option

env var unset

Unset a single config value for an environment.

Description for env var unset

Run 'sf env var list' to see a list of config values that can be unset.

Examples for env var unset

Unset a value:

```
sf env var unset --target-compute environment-alias
```

Usage

sf env var unset

`[--json]`

`[-e TARGET-COMPUTE]`

Flags

--json

Optional

Format output as json.

Type: boolean

-e | --target-compute TARGET-COMPUTE

Optional

Environment name.

Type: option

force Commands

Legacy commands for backward compatibility.

force data bulk delete

Bulk delete records from an org using a CSV file. Uses Bulk API 1.0.

force data bulk status

View the status of a bulk data load job or batch. Uses Bulk API 1.0.

force data bulk upsert

Bulk upsert records to an org from a CSV file. Uses Bulk API 1.0.

force lightning lwc test create

force lightning lwc test run

force lightning lwc test setup

force org clone (Deprecated)

The command `force org clone` has been deprecated and will be removed in v60.0 or later. Clone a sandbox org.

force org create (Deprecated)

The command `force org create` has been deprecated. Create a scratch org or sandbox.

force org delete (Deprecated)

The command `force org delete` has been deprecated. Delete a scratch or sandbox org.

force org status (Deprecated)

The command `force org status` has been deprecated and will be removed in v60.0 or later. Check the status of a sandbox, and if complete, authenticate to it.

force user password generate

Generate a random password for scratch org users.

`force data bulk delete`

Bulk delete records from an org using a CSV file. Uses Bulk API 1.0.

Description for `force data bulk delete`

The CSV file must have only one column ("Id") and then the list of record IDs you want to delete, one ID per line.

When you execute this command, it starts a job and one or more batches, displays their IDs, and then immediately returns control of the terminal to you by default. If you prefer to wait, set the --wait flag to the number of minutes; if it times out, the command outputs the IDs. Use the job and batch IDs to check the status of the job with the "sf force data bulk status" command. A single job can contain many batches, depending on the length of the CSV file.

Examples for `force data bulk delete`

Bulk delete Account records from your default org using the list of IDs in the "files/delete.csv" file:

```
sf force data bulk delete --sobject Account --file files/delete.csv
```

Bulk delete records from a custom object in an org with alias my-scratch and wait 5 minutes for the command to complete:

```
sf force data bulk delete --sobject MyObject__c --file files/delete.csv --wait 5 --target-org
 my-scratch
```

Usage

`sf force data bulk delete`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
```

```
-f FILE
-s SOBJECT
[-w WAIT]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-f | --file FILE`
Required

CSV file that contains the IDs of the records to delete.

Type: option

`-s | --sobject SOBJECT`
Required

API name of the Salesforce object, either standard or custom, that you want to delete records from.

Type: option

`-w | --wait WAIT`
Optional

Number of minutes to wait for the command to complete before displaying the results.

Type: option

Default value: 0 minutes

`force data bulk status`

View the status of a bulk data load job or batch. Uses Bulk API 1.0.

Description for `force data bulk status`

Run this command using the job ID or batch ID returned from the "sf force data bulk delete" or "sf force data bulk upsert" commands.

Examples for `force data bulk status`

View the status of a bulk load job in your default org:

```
sf force data bulk status --job-id 750xx000000005sAAA
```

View the status of a bulk load job and a specific batches in an org with alias my-scratch:

```
sf force data bulk status --job-id 750xx000000005sAAA --batch-id 751xx000000005nAAA
--target-org my-scratch
```

Usage

`sf force data bulk status`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`[-b BATCH-ID]`

`-i JOB-ID`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Org alias or username to use for the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-b | --batch-id BATCH-ID`

Optional

ID of the batch whose status you want to view; you must also specify the job ID.

Type: option

`-i | --job-id JOB-ID`

Required

ID of the job whose status you want to view.

Type: option

`force data bulk upsert`

Bulk upsert records to an org from a CSV file. Uses Bulk API 1.0.

Description for `force data bulk upsert`

An upsert refers to inserting a record into a Salesforce object if the record doesn't already exist, or updating it if it does exist.

When you execute this command, it starts a job and one or more batches, displays their IDs, and then immediately returns control of the terminal to you by default. If you prefer to wait, set the --wait flag to the number of minutes; if it times out, the command outputs the IDs. Use the job and batch IDs to check the status of the job with the "sf force data bulk status" command. A single job can contain many batches, depending on the length of the CSV file.

See "Prepare CSV Files" in the Bulk API Developer Guide for details on formatting your CSV file. (https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/datafiles_csv_preparing.htm)

By default, the job runs the batches in parallel, which we recommend. You can run jobs serially by specifying the --serial flag. But don't process data in serial mode unless you know this would otherwise result in lock timeouts and you can't reorganize your batches to avoid the locks.

Examples for `force data bulk upsert`

Bulk upsert records to the Contact object in your default org:

```
sf --sobject Contact --file files/contacts.csv --external-id Id
```

Bulk upsert records to a custom object in an org with alias my-scratch and wait 5 minutes for the command to complete:

```
sf force data bulk upsert --sobject MyObject__c --file files/file.csv --external-id
MyField__c --wait 5 --target-org my-scratch
```

Usage

`sf force data bulk upsert`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
-i EXTERNAL-ID
-f FILE
-s SOBJECT
[-w WAIT]
[-r]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

-o | --target-org TARGET-ORG
Required

Org alias or username to use for the target org.

Type: option

--api-version API-VERSION
Optional

Override the api version used for api requests made by this command

Type: option

-i | --external-id EXTERNAL-ID
Required

Name of the external ID field, or the Id field.

Type: option

-f | --file FILE
Required

CSV file that contains the records to upsert.

Type: option

-s | --sobject SOBJECT
Required

API name of the Salesforce object, either standard or custom, that you want to upsert records to.

Type: option

-w | --wait WAIT
Optional

Number of minutes to wait for the command to complete before displaying the results.

Type: option

Default value: 0 minutes

-r | --serial
Optional

Run batches in serial mode.

Type: boolean

force lightning lwc test create

Description for `force lightning lwc test create`

creates a Lightning web component test file with boilerplate code inside a __tests__ directory.

Examples for `force lightning lwc test create`

```
$ sfdx force:lightning:lwc:test:create -f force-app/main/default/lwc/myButton/myButton.js
```

Usage

`sf force lightning lwc test create`

`[--json]`

`[--loglevel LOGLEVEL]`

`-f FILEPATH`

Flags

`--json`

Optional

format output as json

Type: boolean

`--loglevel LOGLEVEL`

Optional

logging level for this command invocation

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-f | --filepath FILEPATH`

Required

path to Lightning web component .js file to create a test for

Type: string

`force lightning lwc test run`

Description for `force lightning lwc test run`

invokes Lightning Web Components Jest unit tests.

Examples for `force lightning lwc test run`

```
$ sfdx force:lightning:lwc:test:run
```

```
$ sfdx force:lightning:lwc:test:run -w
```

Usage

`sf force lightning lwc test run`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-d]`

`[--watch]`

Flags

`--json`
Optional

format output as json

Type: boolean

`--loglevel LOGLEVEL`
Optional

logging level for this command invocation

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-d | --debug`
Optional

run tests in debug mode

Type: boolean

`--watch`
Optional

run tests in watch mode

Type: boolean

`force lightning lwc test setup`

Description for `force lightning lwc test setup`

install Jest unit testing tools for Lightning Web Components.

Examples for `force lightning lwc test setup`

```
$ sfdx force:lightning:lwc:test:setup
```

Usage

`sf force lightning lwc test setup`
`[--json]`

`[--loglevel LOGLEVEL]`

Flags

`--json`
Optional

format output as json

Type: boolean

`--loglevel LOGLEVEL`

Optional

logging level for this command invocation

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`force org clone` (Deprecated)

The command `force org clone` has been deprecated and will be removed in v60.0 or later. Clone a sandbox org.

Description for `force org clone`

There are two ways to clone a sandbox: either specify a sandbox definition file or provide key=value pairs at the command line. Key-value pairs at the command-line override their equivalent sandbox definition file values. In either case, you must specify both the "SandboxName" and "SourceSandboxName" options to set the names of the new sandbox and the one being cloned, respectively.

Set the --targetusername (-u) parameter to a production org with sandbox licenses. The --type (-t) parameter is required and must be set to "sandbox".

Examples for `force org clone`

```
$ sf force org clone -t sandbox -f config/dev-sandbox-def.json -u prodOrg -a MyDevSandbox
```

```
$ sf force org clone -t sandbox SandboxName=NewClonedSandbox
SourceSandboxName=ExistingSandbox -u prodOrg -a MyDevSandbox
```

Usage

`sf force org clone`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`-t TYPE`

`[-f DEFINITIONFILE]`

`[-s]`

`[-a SETALIAS]`

`[-w WAIT]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-t | --type TYPE`
Required

Type of org to create.

Type: option

Permissible values are: sandbox

`-f | --definitionfile DEFINITIONFILE`
Optional

Path to the sandbox definition file.

Type: option

`-s | --setdefaultusername`
Optional

Set the cloned org as your default.

Type: boolean

`-a | --setalias SETALIAS`
Optional

Alias for the cloned org.

Type: option

`-w | --wait WAIT`
Optional

Number of minutes to wait while polling for status.

Sets the streaming client socket timeout, in minutes. If the streaming client socket has no contact from the server for a number of minutes, the client exits. Specify a longer wait time if timeouts occur frequently.

Type: option

`force org create` (Deprecated)

The command `force org create` has been deprecated. Create a scratch org or sandbox.

Description for `force org create`

Creates a scratch org or a sandbox org using the values specified in a configuration file or key=value pairs that you specify on the command line. Values specified on the command line override values in the configuration file. Specify a configuration file or provide key=value pairs while creating a scratch org or a sandbox. When creating scratch orgs, —targetdevhubusername (-v) must be a Dev

Hub org. When creating sandboxes, the --targetusername (-u) must be a production org with sandbox licenses. The —type (-t) is required if creating a sandbox.

Examples for `force org create`

```
$ sf force org create -f config/enterprise-scratch-def.json -a MyScratchOrg
```

```
$ sf force org create edition=Developer -a MyScratchOrg -s -v devHub
```

```
$ sf force org create -f config/enterprise-scratch-def.json -a ScratchOrgWithOverrides
username=testuser1@mycompany.org
```

```
$ sf force org create -t sandbox -f config/dev-sandbox-def.json -a MyDevSandbox -u prodOrg
```

Usage

`sf force org create`

`[--json]`

`[-o TARGET-ORG]`

`[-v TARGET-DEV-HUB]`

`[--api-version API-VERSION]`

`[-t TYPE]`

`[-f DEFINITIONFILE]`

`[-n]`

`[-c]`

`[-i CLIENTID]`

`[-s]`

`[-a SETALIAS]`

`[-w WAIT]`

`[-d DURATIONDAYS]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Optional

Username or alias of the production org that contains the sandbox license.

Type: option

`-v | --target-dev-hub TARGET-DEV-HUB`

Optional

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-t | --type TYPE`

Optional

Type of org to create.

Type: option

Permissible values are: scratch, sandbox

Default value: scratch

`-f | --definitionfile DEFINITIONFILE`

Optional

Path to an org definition file.

Type: option

`-n | --nonamespace`

Optional

Create the scratch org with no namespace.

Type: boolean

`-c | --noancestors`

Optional

Do not include second-generation package ancestors in the scratch org.

Type: boolean

`-i | --clientid CLIENTID`

Optional

Connected app consumer key; not supported for sandbox org creation.

Type: option

`-s | --setdefaultusername`

Optional

Set the created org as the default username.

Type: boolean

`-a | --setalias SETALIAS`

Optional

Alias for the created org.

Type: option

`-w | --wait WAIT`

Optional

Streaming client socket timeout (in minutes).

Type: option

-d | --durationdays DURATIONDAYS

Optional

Duration of the scratch org (in days) (default:7, min:1, max:30).

Type: option

Default value: 7

`force org delete` (Deprecated)

The command `force org delete` has been deprecated. Delete a scratch or sandbox org.

Description for `force org delete`

Salesforce CLI marks the org for deletion in either the Dev Hub org (for scratch orgs) or production org (for sandboxes) and then deletes all local references to the org from your computer.

To mark the org for deletion without being prompted to confirm, specify --noprompt.

Examples for `force org delete`

```
$ sf force org delete -u me@my.org
```

```
$ sf force org delete -u MyOrgAlias -p
```

Usage

sf force org delete

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`[-p]`

Flags

--json

Optional

Format output as json.

Type: boolean

-o | --target-org TARGET-ORG

Required

Username or alias of the target org.

Type: option

--api-version API-VERSION

Optional

Override the api version used for api requests made by this command

Type: option

`-p | --no-prompt`
Optional

No prompt to confirm deletion.

Type: boolean

`force org status` (Deprecated)

The command `force org status` has been deprecated and will be removed in v60.0 or later. Check the status of a sandbox, and if complete, authenticate to it.

Description for `force org status`

Use this command to check the status of your sandbox creation or clone and, if the sandbox is ready, authenticate to it.

Use the --wait (-w) parameter to specify the number of minutes that the command waits for the sandbox creation or clone to complete before returning control of the terminal to you.

Set the --target-org (-o) parameter to the username or alias of the production org that contains the sandbox license.

Examples for `force org status`

```
$ sf force org status --sandboxname DevSbx1 --setalias MySandbox -u prodOrg
```

```
$ sf force org status --sandboxname DevSbx1 --wait 45 --setdefaultusername -u prodOrg
```

Usage

`sf force org status`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`-n SANDBOXNAME`

`[-s]`

`[-a SETALIAS]`

`[-w WAIT]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-n | --sandboxname SANDBOXNAME`

Required

Name of the sandbox org to check status for.

Type: option

`-s | --setdefaultusername`

Optional

Set the created or cloned org as your default.

Type: boolean

`-a | --setalias SETALIAS`

Optional

Alias for the created or cloned org.

Type: option

`-w | --wait WAIT`

Optional

Number of minutes to wait while polling for status.

Type: option

`force user password generate`

Generate a random password for scratch org users.

Description for `force user password generate`

By default, new scratch orgs contain one admin user with no password. Use this command to generate or change a password for any scratch org user. After it's set, you can’t unset a password, you can only change it.

To change the password strength, set the --complexity flag to a value between 0 and 5. Each value specifies the types of characters used in the generated password:

0 - lower case letters only

1 - lower case letters and numbers only

2 - lower case letters and symbols only

3 - lower and upper case letters and numbers only

4 - lower and upper case letters and symbols only

5 - lower and upper case letters and numbers and symbols only

To see a password that was previously generated, run "org display user".

Examples for **`force user password generate`**

Generate a password for the original admin user of your default scratch org:

```
sf force user password generate
```

Generate a password that contains 12 characters for the original admin user of the scratch org with alias "my-scratch":

```
sf force user password generate --length 12 --target-org my-scratch
```

Generate a password for your default scratch org admin user that uses lower and upper case letters and numbers only:

```
sf force user password generate --complexity 3
```

Generate a password for the specified users in the default scratch org:

```
sf force user password generate --on-behalf-of user1@my.org --on-behalf-of user2@my.org
--on-behalf-of user3@my.org
```

Usage

`sf force user password generate`

```
[--json]
[-o ON-BEHALF-OF]
[-l LENGTH]
[-c COMPLEXITY]
-u TARGET-ORG
[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --on-behalf-of ON-BEHALF-OF`

Optional

Comma-separated list of usernames or aliases to assign the password to.

Type: option

`-l | --length LENGTH`

Optional

Number of characters in the generated password; valid values are between 8 and 100.

Type: option

Default value: 13

`-c | --complexity COMPLEXITY`

Optional

Level of password complexity or strength; the higher the value, the stronger the password.

Type: option

Default value: 5

-u | --target-org TARGET-ORG

Required

Scratch org alias or login user.

Type: option

--api-version API-VERSION

Optional

Override the api version used for api requests made by this command

Type: option

`generate` Commands

Commands to generate a project, create a function, and more.

generate function

Create a Salesforce Function with basic scaffolding specific to a given language.

`generate function`

Create a Salesforce Function with basic scaffolding specific to a given language.

Description for `generate function`

Both '--language' and '--name' are required flags. Function names must start with a capital letter.

Examples for `generate function`

Create a JavaScript function:

```
sf generate function --function-name myfunction --language javascript
```

Usage

sf generate function

`[--json]`

`[-n FUNCTION-NAME]`

`-l LANGUAGE`

Flags

--json

Optional

Format output as json.

Type: boolean

-n | --function-name FUNCTION-NAME
Optional

Function name. Must start with a capital letter.

Type: option

-l | --language LANGUAGE
Required

The language in which the function is written.

Type: option

Permissible values are: java, javascript, python, typescript

info Commands

Access Salesforce CLI information from the command line.

info releasenotes display
Display Salesforce CLI release notes on the command line.

info releasenotes display

Display Salesforce CLI release notes on the command line.

Description for info releasenotes display

By default, this command displays release notes for the currently installed CLI version on your computer. Use the --version flag to view release notes for a different release.

Examples for info releasenotes display

Display release notes for the currently installed CLI version:

```
sf info releasenotes display stable, stable-rc, latest, latest-rc, rc
```

Display release notes for CLI version 7.120.0:

```
sf info releasenotes display --version 7.120.0 stable, stable-rc, latest, latest-rc, rc
```

Display release notes for the CLI version that corresponds to a tag (stable, stable-rc, latest, latest-rc, rc):

```
sf info releasenotes display --version latest
```

Usage

sf info releasenotes display
`[--json]`

`[-v VERSION]`

Flags

`--json`
: Optional

  Format output as json.

  Type: boolean

`-v` | `--version VERSION`
: Optional

  CLI version or tag for which to display release notes.

  Type: option

Aliases for `info releasenotes display`

```
whatsnew
```

`lightning` Commands

Work with Lightning Web and Aura components.

lightning generate app
: Generate a Lightning App.

lightning generate component
: Generate a bundle for an Aura component or a Lightning web component.

lightning generate event
: Generate a Lightning Event.

lightning generate interface
: Generate a Lightning Interface.

lightning generate test
: Generate a Lightning test.

`lightning generate app`

Generate a Lightning App.

Description for `lightning generate app`

Generates a Lightning App bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Examples for `lightning generate app`

Generate the metadata files for a Lightning app bundle called "myapp" in the current directory:

```
sf lightning generate app --name myapp
```

Similar to the previous example, but generate the files in the "force-app/main/default/aura" directory:

```
sf lightning generate app --name myapp --output-dir force-app/main/default/aura
```

Usage

`sf lightning generate app`

```
[--json]
-n NAME
[-t TEMPLATE]
[-d OUTPUT-DIR]
[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated Lightning App.

The name can be up to 40 characters and must start with a letter.

Type: option

`-t | --template TEMPLATE`

Optional

Template to use for file creation.

Supplied parameter values or default values are filled into a copy of the template.

Type: option

Permissible values are: DefaultLightningApp

Default value: DefaultLightningApp

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `lightning generate app`

```
force:lightning:app:create
```

`lightning generate component`

Generate a bundle for an Aura component or a Lightning web component.

Description for `lightning generate component`

Generates the bundle in the specified directory or the current working directory. The bundle consists of multiple files in a directory with the designated name. Lightning web components are contained in the directory with name "lwc", Aura components in "aura".

To generate a Lightning web component, pass "--type lwc" to the command. If you don’t specify --type, Salesforce CLI generates an Aura component by default.

Examples for `lightning generate component`

Generate the metadata files for an Aura component bundle in the current directory:

```
sf lightning generate component --name mycomponent
```

Generate a Lightning web component bundle in the current directory:

```
sf lightning generate component --name mycomponent --type lwc
```

Generate an Aura component bundle in the "force-app/main/default/aura" directory:

```
sf lightning generate component --name mycomponent --output-dir force-app/main/default/aura
```

Generate a Lightning web component bundle in the "force-app/main/default/lwc" directory:

```
sf lightning generate component --name mycomponent --type lwc --output-dir
force-app/main/default/lwc
```

Usage

`sf lightning generate component`

```
[--json]
-n NAME
[-t TEMPLATE]
[-d OUTPUT-DIR]
[--api-version API-VERSION]
[--type TYPE]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

-n | --name NAME

Required

Name of the generated Lightning Component.

The name can be up to 40 characters and must start with a letter.

Type: option

-t | --template TEMPLATE

Optional

Template to use for file creation.

Supplied parameter values or default values are filled into a copy of the template.

Type: option

Permissible values are: default, analyticsDashboard, analyticsDashboardWithStep

Default value: default

-d | --output-dir OUTPUT-DIR

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

--api-version API-VERSION

Optional

Override the api version used for api requests made by this command

Type: option

--type TYPE

Optional

Type of the component bundle.

Type: option

Permissible values are: aura, lwc

Default value: aura

Aliases for `lightning generate component`

```
force:lightning:component:create
```

`lightning generate event`

Generate a Lightning Event.

Description for `lightning generate event`

Generates a Lightning Event bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Examples for `lightning generate event`

Generate the metadata files for a Lightning event bundle called "myevent" in the current directory:

```
sf lightning generate event --name myevent
```

Similar to previous example, but generate the files in the "force-app/main/default/aura" directory:

```
sf lightning generate event --name myevent --output-dir force-app/main/default/aura
```

Usage

`sf lightning generate event`

```
[--json]
-n NAME
[-t TEMPLATE]
[-d OUTPUT-DIR]
[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated Lightning Event.

The name can be up to 40 characters and must start with a letter.

Type: option

`-t | --template TEMPLATE`

Optional

Template to use for file creation.

Supplied parameter values or default values are filled into a copy of the template.

Type: option

Permissible values are: DefaultLightningEvt

Default value: DefaultLightningEvt

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `lightning generate event`

```
force:lightning:event:create
```

`lightning generate interface`

Generate a Lightning Interface.

Description for `lightning generate interface`

Generates a Lightning Interface bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Examples for `lightning generate interface`

Generate the metadata files for a Lightning interface bundle called "myinterface" in the current directory:

```
sf lightning generate interface --name myinterface
```

Similar to the previous example but generate the files in the "force-app/main/default/aura" directory:

```
sf lightning generate interface --name myinterface --output-dir force-app/main/default/aura
```

Usage

`sf lightning generate interface`

`[--json]`

`-n NAME`

`[-t TEMPLATE]`

`[-d OUTPUT-DIR]`

`[--api-version API-VERSION]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated Lightning Interface.

The name can be up to 40 characters and must start with a letter.

Type: option

`-t | --template TEMPLATE`

Optional

Template to use for file creation.

Supplied parameter values or default values are filled into a copy of the template.

Type: option

Permissible values are: DefaultLightningIntf

Default value: DefaultLightningIntf

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `lightning generate interface`

```
force:lightning:interface:create
```

`lightning generate test`

Generate a Lightning test.

Description for `lightning generate test`

Generates the test in the specified directory or the current working directory. The .resource file and associated metadata file are generated.

Examples for `lightning generate test`

Generate the metadata files for the Lightning test called MyLightningTest in the current directory:

```
sf lightning generate test --name MyLightningTest
```

Similar to the previous example but generate the files in the "force-app/main/default/lightningTests" directory:

```
sf lightning generate test --name MyLightningTest --output-dir
force-app/main/default/lightningTests
```

Usage

`sf lightning generate test`

```
[--json]

-n NAME

[-t TEMPLATE]

[-d OUTPUT-DIR]

[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated Lightning Test.

Name of the new Lightning test; can be up to 40 characters and must start with a letter.

Type: option

`-t | --template TEMPLATE`

Optional

Template to use for file creation.

Supplied parameter values or default values are filled into a copy of the template.

Type: option

Permissible values are: DefaultLightningTest

Default value: DefaultLightningTest

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `lightning generate test`

```
force:lightning:test:create
```

`limits` Commands

Display an org's limits.

limits api display
: Display information about limits in your org.

limits recordcounts display
: Display record counts for the specified standard or custom objects.

`limits api display`

Display information about limits in your org.

Description for `limits api display`

For each limit, this command returns the maximum allocation and the remaining allocation based on usage. See this topic for a description of each limit: https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/resources_limits.htm.

Examples for `limits api display`

Display limits in your default org:

```
sf limits api display
```

Display limits in the org with alias "my-scratch-org":

```
sf limits api display --target-org my-scratch-org
```

Usage

`sf limits api display`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `limits api display`

```
force:limits:api:display
```

```
org:list:limits
```

`limits recordcounts display`

Display record counts for the specified standard or custom objects.

Description for `limits recordcounts display`

Use this command to get an approximate count of the records in standard or custom objects in your org. These record counts are the same as the counts listed in the Storage Usage page in the Setup UI. The record counts are approximate because they're calculated asynchronously and your org's storage usage isn't updated immediately. To display all available record counts, run the command without the --sobject flag.

Examples for `limits recordcounts display`

Display all available record counts in your default org:

```
sf limits recordcounts display
```

Display record counts for the Account, Contact, Lead, and Opportunity objects in your default org:

```
sf limits recordcounts display --sobject Account --sobject Contact --sobject Lead --sobject
 Opportunity
```

Display record counts for the Account and Lead objects for the org with alias "my-scratch-org":

```
sf limits recordcounts display --sobject Account --sobject Lead --target-org my-scratch-org
```

Usage

`sf limits recordcounts display`

`[--json]`

`[-s SOBJECT]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

-s | --sobject SOBJECT
Optional

API name of the standard or custom object for which to display record counts.

Type: option

-o | --target-org TARGET-ORG
Required

Username or alias of the target org.

Type: option

--api-version API-VERSION
Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `limits recordcounts display`

```
force:limits:recordcounts:display
```

```
org:list:sobject:record-counts
```

`login` Commands

Commands to log in to an environment.

login (Deprecated)
The command `login` has been deprecated and will be removed in v58.0 or later. Log interactively into an environment.

login functions
Log in to Salesforce Functions.

login functions jwt
Login using JWT instead of default web-based flow. This will authenticate you with both sf and Salesforce Functions.

`login` (Deprecated)

The command `login` has been deprecated and will be removed in v58.0 or later. Log interactively into an environment.

Description for `login`

Logging into an environment authorizes the CLI to run other commands that connect to that environment.

Examples for `login`

Log in interactively:

```
sf login
```

Usage

`sf login`
`[--json]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

`login functions`

Log in to Salesforce Functions.

Description for `login functions`

This step is required to develop or deploy Salesforce Functions.

Examples for `login functions`

Log in to Salesforce Functions:

```
sf login functions
```

Usage

`sf login functions`
`[--json]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

`login functions jwt`

Login using JWT instead of default web-based flow. This will authenticate you with both sf and Salesforce Functions.

Description for `login functions jwt`

Use this command when executing from a script.

Examples for `login functions jwt`

Log in using JWT:

```
sf login functions jwt --username example@username.org --keyfile file.key --clientid 123456
```

Log in and specify the org alias and URL, set as default org and default Dev Hub, and format output as JSON:

```
sf login functions jwt --username example@username.org --keyfile file.key --clientid 123456
 --alias org-alias --set-default --set-default-dev-hub --instance-url
https://path/to/instance --json
```

Usage

`sf login functions jwt`

`[--json]`

`-u USERNAME`

`-f KEYFILE`

`-i CLIENTID`

`[-l INSTANCE-URL]`

`[-a ALIAS]`

`[-d]`

`[-v]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-u | --username USERNAME`

Required

Authentication username.

Type: option

`-f | --keyfile KEYFILE`

Required

Path to JWT keyfile.

Type: option

`-i | --clientid CLIENTID`

Required

OAuth client ID.

Type: option

`-l | --instance-url INSTANCE-URL`

Optional

The login URL of the instance the org lives on.

Type: option

`-a | --alias ALIAS`

Optional

Alias for the org.

Type: option

`-d | --set-default`

Optional

Set the org as the default that all org-related commands run against.

Type: boolean

`-v | --set-default-dev-hub`

Optional

Set the org as the default Dev Hub for scratch org creation.

Type: boolean

`logout` Commands

Commands to log out of an environment.

logout (Deprecated)

The command `logout` has been deprecated and will be removed in v58.0 or later. Log out interactively from environments.

logout functions

Log out of your Salesforce Functions account.

`logout` (Deprecated)

The command `logout` has been deprecated and will be removed in v58.0 or later. Log out interactively from environments.

Description for `logout`

By default, the command prompts you to select which environments you want to log out of. Use --no-prompt to not be prompted and log out of all environments.

Examples for `logout`

Interactively select the environments to log out of:

```
sf logout
```

Log out of all environments, without being prompted:

```
sf logout --no-prompt
```

Usage

`sf logout`
    `[--json]`
    `[--no-prompt]`

Flags

`--json`
Optional
Format output as json.
Type: boolean

`--no-prompt`
Optional
Don't prompt for confirmation; logs you out of all environments.
Type: boolean

`logout functions`

Log out of your Salesforce Functions account.

Examples for `logout functions`

Log out:

```
sf logout functions
```

Usage

`sf logout functions`
    `[--json]`

Flags

`--json`
Optional
Format output as json.
Type: boolean

`org` Commands

Commands to create and manage orgs and scratch org users.

org assign permset
Assign a permission set to one or more users of a scratch org.

org assign permsetlicense

Assign a permission set license to one or more users of a scratch org.

org create sandbox

Create a sandbox org.

org create scratch

Create a scratch org.

org create shape

Create a scratch org configuration (shape) based on the specified source org.

org create snapshot (Pilot)

Create a snapshot of a scratch org.

org create user

Create a user for a scratch org.

org delete sandbox

Delete a sandbox.

org delete scratch

Delete a scratch org.

org delete shape

Delete all org shapes for a target org.

org delete snapshot (Pilot)

Delete a scratch org snapshot.

org display

Display information about an org.

org display user

Display information about a Salesforce user.

org generate password

Generate a random password for scratch org users.

org get snapshot (Pilot)

Get details about a scratch org snapshot.

org list

List all orgs you've created or authenticated to.

org list auth

List authorization information about the orgs you created or logged into.

org list metadata

List the metadata components and properties of a specified type.

org list metadata-types

Display details about the metadata types that are enabled for your org.

org list shape

List all org shapes you've created.

org list snapshot (Pilot)

List scratch org snapshots.

org list users
List all locally-authenticated users of an org.

org login access-token
Authorize an org using an existing Salesforce access token.

org login device
Authorize an org using a device code.

org login jwt
Log in to a Salesforce org using a JSON web token (JWT).

org login sfdx-url
Authorize an org using a Salesforce DX authorization URL stored in a file.

org login web
Log in to a Salesforce org using the web server flow.

org logout
Log out of a Salesforce org.

org open
Open your default scratch org, or another specified org, in a browser.

org resume sandbox
Check the status of a sandbox creation, and log in to it if it's ready.

org resume scratch
Resume the creation of an incomplete scratch org.

`org assign permset`

Assign a permission set to one or more users of a scratch org.

Description for `org assign permset`

To specify an alias for the --target-org or --on-behalf-of flags, use the CLI username alias, such as the one you set with the "alias set" command. Don't use the value of the Alias field of the User Salesforce object for the org user.

To assign multiple permission sets, either set multiple --name flags or a single --name flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --on-behalf-of.

Examples for `org assign permset`

Assign two permission sets called DreamHouse and CloudHouse to original admin user of your default scratch org:

```
sf org assign permset --name DreamHouse --name CloudHouse
```

Assign the Dreamhouse permission set to the original admin user of the scratch org with alias "my-scratch":

```
sf org assign permset --name DreamHouse --target-org my-scratch
```

Assign the Dreamhouse permission set to the specified list of users of your default scratch org:

```
sf org assign permset --name DreamHouse --on-behalf-of user1@my.org --on-behalf-of user2
--on-behalf-of user
```

Usage

```
sf org assign permset
   [--json]
   -n NAME
   [-b ON-BEHALF-OF]
   -o TARGET-ORG
   [--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Permission set to assign.

Type: option

`-b | --on-behalf-of ON-BEHALF-OF`

Optional

Username or alias to assign the permission set to.

Type: option

`-o | --target-org TARGET-ORG`

Required

Scratch org alias or login user.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`org assign permsetlicense`

Assign a permission set license to one or more users of a scratch org.

Description for `org assign permsetlicense`

To specify an alias for the --target-org or --on-behalf-of flags, use the CLI username alias, such as the one you set with the "alias set" command. Don't use the value of the Alias field of the User Salesforce object for the org user.

To assign multiple permission sets, either set multiple --name flags or a single --name flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --on-behalf-of.

Examples for `org assign permsetlicense`

Assign the DreamHouse permission set license to original admin user of your default scratch org:

```
sf org assign permsetlicense --name DreamHouse
```

Assign two permission set licenses to the original admin user of the scratch org with alias "my-scratch":

```
sf org assign permsetlicense --name DreamHouse --name CloudHouse --target-org my-scratch
```

Assign the Dreamhouse permission set license to the specified list of users of your default scratch org:

```
sf org assign permsetlicense --name DreamHouse --on-behalf-of user1@my.org --on-behalf-of
 user2 --on-behalf-of user3
```

Usage

`sf org assign permsetlicense`

`[--json]`

`-n NAME`

`[-b ON-BEHALF-OF]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the permission set license to assign.

Type: option

`-b | --on-behalf-of ON-BEHALF-OF`

Optional

Usernames or alias to assign the permission set license to.

Type: option

`-o | --target-org TARGET-ORG`

Required

Scratch org alias or login user.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`org create sandbox`

Create a sandbox org.

Description for `org create sandbox`

There are two ways to create a sandbox org: specify a definition file that contains the sandbox options or use the --name and --license-type flags to specify the two required options. If you want to set an option other than name or license type, such as apexClassId, you must use a definition file.

Examples for `org create sandbox`

Create a sandbox org using a definition file and give it the alias "MyDevSandbox". The production org that contains the sandbox license has the alias "prodOrg".

```
sf org create sandbox -f config/dev-sandbox-def.json --alias MyDevSandbox --target-org
prodOrg
```

Create a sandbox org by directly specifying its name and type of license (Developer) instead of using a definition file. Set the sandbox org as your default.

```
sf org create sandbox --name mysandbox --license-type Developer --alias MyDevSandbox
--target-org prodOrg --set-default
```

Usage

`sf org create sandbox`

```
[--json]
[-f DEFINITION-FILE]
[-s]
[-a ALIAS]
[-w WAIT]
[-i POLL-INTERVAL]
[--async]
[-n NAME]
[-c CLONE]
[-l LICENSE-TYPE]
-o TARGET-ORG
[--no-prompt]
[--no-track-source]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-f | --definition-file DEFINITION-FILE`

Optional

Path to a sandbox definition file.

The sandbox definition file is a blueprint for the sandbox. You can create different definition files for each sandbox type that you use in the development process. See <https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_sandbox_definition.htm> for all the options you can specify in the defintion file.

Type: option

`-s | --set-default`

Optional

Set the sandbox org as your default org.

Type: boolean

`-a | --alias ALIAS`

Optional

Alias for the sandbox org.

When you create a sandbox, the generated usernames are based on the usernames present in the production org. To ensure uniqueness, the new usernames are appended with the name of the sandbox. For example, the username "user@example.com" in the production org results in the username "user@example.com.mysandbox" in a sandbox named "mysandbox". When you set an alias for a sandbox org, it's assigned to the resulting username of the user running this command.

Type: option

`-w | --wait WAIT`

Optional

Number of minutes to wait for the sandbox org to be ready.

If the command continues to run after the wait period, the CLI returns control of the terminal to you and displays the "sf org resume sandbox" command you run to check the status of the create. The displayed command includes the job ID for the running sandbox creation.

Type: option

Default value: 30 minutes

`-i | --poll-interval POLL-INTERVAL`

Optional

Number of seconds to wait between retries.

Type: option

Default value: 30 seconds

`--async`

Optional

Request the sandbox creation, but don't wait for it to complete.

The command immediately displays the job ID and returns control of the terminal to you. This way, you can continue to use the CLI. To check the status of the sandbox creation, run "sf org resume sandbox".

Type: boolean

`-n | --name NAME`

Optional

Name of the sandbox org.

The name must be a unique alphanumeric string (10 or fewer characters) to identify the sandbox. You can't reuse a name while a sandbox is in the process of being deleted.

Type: option

`-c | --clone CLONE`

Optional

Name of the sandbox org to clone.

The value of clone must be an existing sandbox in the same target-org.

Type: option

`-l | --license-type LICENSE-TYPE`

Optional

Type of sandbox license.

Type: option

Permissible values are: Developer, Developer_Pro, Partial, Full

`-o | --target-org TARGET-ORG`

Required

Username or alias of the production org that contains the sandbox license.

When it creates the sandbox org, Salesforce copies the metadata, and optionally data, from your production org to the new sandbox org.

Type: option

`--no-prompt`

Optional

Don't prompt for confirmation about the sandbox configuration.

Type: boolean

`--no-track-source`

Optional

Do not use source tracking for this sandbox.

We recommend you enable source tracking in Developer and Developer Pro sandbox, which is why it's the default behavior. Source tracking allows you to track the changes you make to your metadata, both in your local project and in the sandbox, and to detect any conflicts between the two.

To disable source tracking in the new sandbox, specify the --no-track-source flag. The main reason to disable source tracking is for performance. For example, while you probably want to deploy metadata and run Apex tests in your CI/CD jobs, you probably don't want to incur the costs of source tracking (checking for conflicts, polling the SourceMember object, various file system operations.) This is a good use case for disabling source tracking in the sandbox.

Type: boolean

Aliases for `org create sandbox`

```
env:create:sandbox
```

`org create scratch`

Create a scratch org.

Description for `org create scratch`

There are two ways to create a scratch org: either specify a definition file that contains the options or use the --edition flag to specify the one required option.

For either method, you can also use these flags; if you use them with --definition-file, they override their equivalent option in the scratch org definition file:

* --description

* --name (equivalent to the "orgName" option)

* --username

* --release

* --edition

* --admin-email (equivalent to the "adminEmail" option)

* --source-org (equivalent to the "sourceOrg" option)

If you want to set options other than the preceding ones, such as org features or settings, you must use a definition file.

You must specify a Dev Hub to create a scratch org, either with the --target-dev-hub flag or by setting your default Dev Hub with the target-dev-hub configuration variable.

Examples for `org create scratch`

Create a Developer edition scratch org using your default Dev Hub and give the scratch org an alias:

```
sf org create scratch --edition developer --alias my-scratch-org
```

Create a scratch org with a definition file. Specify the Dev Hub using its alias, set the scratch org as your default, and specify that it expires in 3 days:

```
sf org create scratch --target-dev-hub MyHub --definition-file
config/project-scratch-def.json --set-default --duration-days 3
```

Create a preview Enterprise edition scratch org; for use only during Salesforce release transition periods:

```
sf org create scratch --edition enterprise --alias my-scratch-org --target-dev-hub MyHub
--release preview
```

Usage

```
sf org create scratch
  [--json]
  [-a ALIAS]
  [--async]
  [-d]
  [-f DEFINITION-FILE]
  -v TARGET-DEV-HUB
```

```
[-c]
[-e EDITION]
[-m]
[-y DURATION-DAYS]
[-w WAIT]
[--api-version API-VERSION]
[-i CLIENT-ID]
[-t]
[--username USERNAME]
[--description DESCRIPTION]
[--name NAME]
[--release RELEASE]
[--admin-email ADMIN-EMAIL]
[--source-org SOURCE-ORG]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-a | --alias ALIAS`

Optional

Alias for the scratch org.

New scratch orgs include one administrator by default. The admin user's username is auto-generated and looks something like test-wvkpnfm5z113@example.com. When you set an alias for a new scratch org, it's assigned this username.

Type: option

`--async`

Optional

Request the org, but don't wait for it to complete.

The command immediately displays the job ID and returns control of the terminal to you. This way, you can continue to use the CLI. To resume the scratch org creation, run "sf org resume scratch".

Type: boolean

`-d | --set-default`

Optional

Set the scratch org as your default org

Type: boolean

`-f | --definition-file DEFINITION-FILE`

Optional

Path to a scratch org definition file.

The scratch org definition file is a blueprint for the scratch org. It mimics the shape of an org that you use in the development life cycle, such as acceptance testing, packaging, or production. See <https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_scratch_orgs_def_file.htm> for all the option you can specify in the definition file.

Type: option

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Overrides the value of the target-dev-hub configuration variable, if set.

Type: option

`-c | --no-ancestors`

Optional

Don't include second-generation managed package (2GP) ancestors in the scratch org.

Type: boolean

`-e | --edition EDITION`

Optional

Salesforce edition of the scratch org. Overrides the value of the "edition" option in the definition file, if set.

The editions that begin with "partner-" are available only if the Dev Hub org is a Partner Business Org.

Type: option

Permissible values are: developer, enterprise, group, professional, partner-developer, partner-enterprise, partner-group, partner-professional

`-m | --no-namespace`

Optional

Create the scratch org with no namespace, even if the Dev Hub has a namespace.

Type: boolean

`-y | --duration-days DURATION-DAYS`

Optional

Number of days before the org expires.

Type: option

Default value: 7 days

`-w | --wait WAIT`

Optional

Number of minutes to wait for the scratch org to be ready.

If the command continues to run after the wait period, the CLI returns control of the terminal to you and displays the job ID. To resume the scratch org creation, run the org resume scratch command and pass it the job ID.

Type: option

Default value: 5 minutes

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-i | --client-id CLIENT-ID`

Optional

Consumer key of the Dev Hub connected app.

Type: option

`-t | --track-source`

Optional

Use source tracking for this scratch org. Set --no-track-source to disable source tracking.

We recommend you enable source tracking in scratch orgs, which is why it's the default behavior. Source tracking allows you to track the changes you make to your metadata, both in your local project and in the scratch org, and to detect any conflicts between the two.

To disable source tracking in the new scratch org, specify the --no-track-source flag. The main reason to disable source tracking is for performance. For example, while you probably want to deploy metadata and run Apex tests in your CI/CD jobs, you probably don't want to incur the costs of source tracking (checking for conflicts, polling the SourceMember object, various file system operations.) This is a good use case for disabling source tracking in the scratch org.

Type: boolean

Default value: true

`--username USERNAME`

Optional

Username of the scratch org admin user. Overrides the value of the "username" option in the definition file, if set.

The username must be unique within the entire scratch org and sandbox universe. You must add your own logic to ensure uniqueness.

Omit this flag to have Salesforce generate a unique username for your org.

Type: option

`--description DESCRIPTION`

Optional

Description of the scratch org in the Dev Hub. Overrides the value of the "description" option in the definition file, if set.

Type: option

`--name NAME`

Optional

Name of the org, such as "Acme Company". Overrides the value of the "orgName" option in the definition file, if set.

Type: option

`--release RELEASE`

Optional

Release of the scratch org as compared to the Dev Hub release.

By default, scratch orgs are on the same release as the Dev Hub. During Salesforce release transition periods, you can override this default behavior and opt in or out of the new release.

Type: option

Permissible values are: preview, previous

`--admin-email ADMIN-EMAIL`

Optional

Email address that will be applied to the org's admin user. Overrides the value of the "adminEmail" option in the definition file, if set.

Type: option

`--source-org SOURCE-ORG`

Optional

15-character ID of the org whose shape the new scratch org will be based on. Overrides the value of the "sourceOrg" option in the definition file, if set.

Type: option

Aliases for `org create scratch`

```
env:create:scratch
```

`org create shape`

Create a scratch org configuration (shape) based on the specified source org.

Description for `org create shape`

Scratch org shapes mimic the baseline setup (features, limits, edition, and Metadata API settings) of a source org without the extraneous data and metadata.

Run "sf org list shape" to view the available org shapes and their IDs.

To create a scratch org from an org shape, include the "sourceOrg" property in the scratch org definition file and set it to the org ID of the source org. Then create a scratch org with the "sf force:org:create" command.

Examples for `org create shape`

Create an org shape for the source org with alias SourceOrg:

```
sf org create shape --target-org SourceOrg
```

Usage

`sf org create shape`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

-o | --target-org TARGET-ORG

Required

Username or alias of the target org.

Type: option

--api-version API-VERSION

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `org create shape`

```
force:org:shape:create
```

`org create snapshot` (Pilot)

Create a snapshot of a scratch org.

Note: We provide the `org create snapshot` command to selected customers through an invitation-only pilot program that requires agreement to specific terms and conditions. Pilot programs are subject to change, and we can't guarantee acceptance. The `org create snapshot` command isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features.

Description for `org create snapshot`

A snapshot is a point-in-time copy of a scratch org. The copy is referenced by its unique name in a scratch org definition file.

Use "sf org get snapshot" to get details, including status, about a snapshot creation request.

To create a scratch org from a snapshot, include the "snapshot" option (instead of "edition") in the scratch org definition file and set it to the name of the snapshot. Then use "sf force:org:create" to create the scratch org.

Examples for `org create snapshot`

Create a snapshot called "Dependencies" using the source scratch org ID and your default Dev Hub org:

```
sf org create snapshot --source-org 00Dxx0000000000 --name Dependencies --description
'Contains PackageA v1.1.0'
```

Create a snapshot called "NightlyBranch" using the source scratch org username and a Dev Hub org with alias NightlyDevHub:

```
sf org create snapshot --source-org myuser@myorg --name NightlyBranch --description 'Contains
 PkgA v2.1.0 and PkgB 3.3.0' --target-dev-hub NightlyDevHub
```

Usage

sf org create snapshot

`[--json]`

`-v TARGET-DEV-HUB`

```
[--api-version API-VERSION]
-o SOURCE-ORG
-n NAME
[-d DESCRIPTION]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`
Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-o | --source-org SOURCE-ORG`
Required

ID or locally authenticated username or alias of scratch org to snapshot.

Type: option

`-n | --name NAME`
Required

Unique name of snapshot.

Type: option

`-d | --description DESCRIPTION`
Optional

Description of snapshot.

Use this description to document the contents of the snapshot. We suggest that you include a reference point, such as a version control system tag or commit ID.

Type: option

Aliases for `org create snapshot`

```
force:org:snapshot:create
```

`org create user`

Create a user for a scratch org.

Description for `org create user`

A scratch org includes one administrator user by default. For testing purposes, however, you sometimes need to create additional users.

The easiest way to create a user is to let this command assign default or generated characteristics to the new user. If you want to customize your new user, create a definition file and specify it with the --definition-file flag. In the file, you can include all the User sObject (SSalesforce object) fields and Salesforce DX-specific options, as described in "User Definition File for Customizing a Scratch Org User" (https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_scratch_orgs_users_def_file.htm). You can also specify these options on the command line.

If you don't customize your new user, this command creates a user with the following default characteristics:

* The username is the existing administrator’s username prepended with a timestamp, such as 1505759162830_test-wvkpnfm5z113@example.com.

* The user’s profile is Standard User.

* The values of the required fields of the User sObject are the corresponding values of the administrator user.

* The user has no password.

Use the --set-alias flag to assign a simple name to the user that you can reference in later CLI commands. This alias is local and different from the Alias field of the User sObject record of the new user, which you set in the Setup UI.

When this command completes, it displays the new username and user ID. Run the "org display user" command to get more information about the new user.

For more information about user limits, defaults, and other considerations when creating a new scratch org user, see https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_scratch_orgs_users.htm.

Examples for `org create user`

Create a user for your default scratch org and let this command generate a username, user ID, and other characteristics:

```
sf org create user
```

Create a user with alias "testuser1" using a user definition file. Set the "profileName" option to "Chatter Free User", which overrides the value in the defintion file if it also exists there. Create the user for the scratch org with alias "my-scratch":

```
sf org create user --set-alias testuser1 --definition-file config/project-user-def.json
profileName='Chatter Free User' --target-org my-scratch
```

Create a user by specifying the username, email, and perm set assignment at the command line; command fails if the username already exists in Salesforce:

```
sf org create user username=testuser1@my.org email=me@my.org permsets=DreamHouse
```

Create a user with a definition file, set the email value as specified (overriding any value in the definition file), and generate a password for the user. If the username in the definition file isn't unique, the command appends the org ID to make it unique:

```
sf org create user --definition-file config/project-user-def.json email=me@my.org
generatepassword=true --set-unique-username
```

Usage

`sf org create user`

```
  [--json]
  [-a SET-ALIAS]
```

```
[-f DEFINITION-FILE]
[-s]
-o TARGET-ORG
[--api-version API-VERSION]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-a | --set-alias SET-ALIAS`
Optional

Set an alias for the created username to reference in other CLI commands.

Type: option

`-f | --definition-file DEFINITION-FILE`
Optional

File path to a user definition file for customizing the new user.

The user definition file uses JSON format and can include any Salesforce User sObject field and Salesforce DX-specific options. See https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_scratch_orgs_users_def_file.htm for more information.

Type: option

`-s | --set-unique-username`
Optional

Force the username, if specified in the definition file or at the command line, to be unique by appending the org ID.

The new user's username must be unique across all Salesforce orgs and in the form of an email address. If you let this command generate a username for you, it's guaranteed to be unique. If you specify an existing username in a definition file, the command fails. Set this flag to force the username to be unique; as a result, the username might be different than what you specify in the definition file.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `org create user`

```
force:user:create
```

`org delete sandbox`

Delete a sandbox.

Description for `org delete sandbox`

Salesforce CLI marks the org for deletion in the production org that contains the sandbox licenses and then deletes all local references to the org from your computer.

Specify a sandbox with either the username you used when you logged into it, or the alias you gave the sandbox when you created it. Run "sf org list" to view all your orgs, including sandboxes, and their aliases.

Examples for `org delete sandbox`

Delete a sandbox with alias my-sandbox:

```
sf org delete sandbox --target-org my-sandbox
```

Specify a username instead of an alias:

```
sf org delete sandbox --target-org myusername@example.com.qa
```

Delete the sandbox without prompting to confirm :

```
sf org delete sandbox --target-org my-sandbox --no-prompt
```

Usage

`sf org delete sandbox`

```
[--json]
-o TARGET-ORG
[-p]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Sandbox alias or login user.

Type: option

`-p | --no-prompt`

Optional

Don't prompt the user to confirm the deletion.

Type: boolean

Aliases for `org delete sandbox`

```
env:delete:sandbox
```

`org delete scratch`

Delete a scratch org.

Description for `org delete scratch`

Salesforce CLI marks the org for deletion in the Dev Hub org and then deletes all local references to the org from your computer.

Specify a scratch org with either the username or the alias you gave the scratch org when you created it. Run "sf org list" to view all your orgs, including scratch orgs, and their aliases.

Examples for `org delete scratch`

Delete a scratch org with alias my-scratch-org:

```
sf org delete scratch --target-org my-scratch-org
```

Specify a username instead of an alias:

```
sf org delete scratch --target-org test-123456-abcdefg@example.com
```

Delete the scratch org without prompting to confirm :

```
sf org delete scratch --target-org my-scratch-org --no-prompt
```

Usage

`sf org delete scratch`

`[--json]`

`-o TARGET-ORG`

`[-p]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Scratch org alias or login user.

Type: option

-p | --no-prompt

Optional

Don't prompt the user to confirm the deletion.

Type: boolean

Aliases for `org delete scratch`

```
env:delete:scratch
```

org delete shape

Delete all org shapes for a target org.

Description for `org delete shape`

A source org can have only one active org shape. If you try to create an org shape for a source org that already has one, the previous shape is marked inactive and replaced by a new active shape. If you don't want to create scratch orgs based on this shape, you can delete the org shape.

Examples for `org delete shape`

Delete all org shapes for the source org with alias SourceOrg:

```
sf org delete shape --target-org SourceOrg
```

Delete all org shapes without prompting:

```
sf org delete shape --target-org SourceOrg --no-prompt
```

Usage

sf org delete shape

[--json]

-o TARGET-ORG

[--api-version API-VERSION]

[-p]

Flags

--json

Optional

Format output as json.

Type: boolean

-o | --target-org TARGET-ORG

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-p | --no-prompt`

Optional

Don't prompt for confirmation.

Type: boolean

Aliases for `org delete shape`

```
force:org:shape:delete
```

`org delete snapshot` (Pilot)

Delete a scratch org snapshot.

Note: We provide the `org delete snapshot` command to selected customers through an invitation-only pilot program that requires agreement to specific terms and conditions. Pilot programs are subject to change, and we can't guarantee acceptance. The `org delete snapshot` command isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features.

Description for `org delete snapshot`

Dev Hub admins can delete any snapshot, while users can delete only their own unless a Dev Hub admin gives the user Modify All permissions.

Examples for `org delete snapshot`

Delete a snapshot from the default Dev Hub using the snapshot ID:

```
sf org delete snapshot --snapshot 0Oo...
```

Delete a snapshot from the specified Dev Hub using the snapshot name:

```
sf org delete snapshot --snapshot BaseSnapshot --target-dev-hub SnapshotDevHub
```

Usage

`sf org delete snapshot`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
-s SNAPSHOT
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`
Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-s | --snapshot SNAPSHOT`
Required

Name or ID of snapshot to delete.

The IDs of scratch org snapshots start with 0Oo.

Type: option

Aliases for `org delete snapshot`

```
force:org:snapshot:delete
```

`org display`

Display information about an org.

Description for `org display`

Output includes your access token, client Id, connected status, org ID, instance URL, username, and alias, if applicable.

Use --verbose to include the SFDX auth URL. WARNING: The SFDX auth URL contains sensitive information, such as a refresh token that can be used to access an org. Don't share or distribute this URL or token.

Including --verbose displays the sfdxAuthUrl property only if you authenticated to the org using "org login web" (not "org login jwt").

Examples for `org display`

Display information about your default org:

```
$ sf org display
```

Display information, including the sfdxAuthUrl property, about the org with alias TestOrg1:

```
$ sf org display --target-org TestOrg1 --verbose
```

Usage

`sf org display`

```
[--json]

-o TARGET-ORG

[--api-version API-VERSION]

[--verbose]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`--verbose`

Optional

Display the sfdxAuthUrl property.

Type: boolean

Aliases for `org display`

```
force:org:display
```

`org display user`

Display information about a Salesforce user.

Description for `org display user`

Output includes the profile name, org ID, access token, instance URL, login URL, and alias if applicable. The displayed alias is local and different from the Alias field of the User sObject record of the new user, which you set in the Setup UI.

Examples for `org display user`

Display information about the admin user of your default scratch org:

```
sf org display user
```

Display information about the specified user and output in JSON format:

```
sf org display user --target-org me@my.org --json
```

Usage

`sf org display user`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `org display user`

```
force:user:display
```

`org generate password`

Generate a random password for scratch org users.

Description for `org generate password`

By default, new scratch orgs contain one admin user with no password. Use this command to generate or change a password for any scratch org user. After it's set, you can’t unset a password, you can only change it.

To change the password strength, set the --complexity flag to a value between 0 and 5. Each value specifies the types of characters used in the generated password:

0 - lower case letters only

1 - lower case letters and numbers only

2 - lower case letters and symbols only

3 - lower and upper case letters and numbers only

4 - lower and upper case letters and symbols only

5 - lower and upper case letters and numbers and symbols only

To see a password that was previously generated, run "org display user".

Examples for `org generate password`

Generate a password for the original admin user of your default scratch org:

```
sf org generate password
```

Generate a password that contains 12 characters for the original admin user of the scratch org with alias "my-scratch":

```
sf org generate password --length 12 --target-org my-scratch
```

Generate a password for your default scratch org admin user that uses lower and upper case letters and numbers only:

```
sf org generate password --complexity 3
```

Generate a password for the specified users in the default scratch org:

```
sf org generate password --on-behalf-of user1@my.org --on-behalf-of user2@my.org
--on-behalf-of user3@my.org
```

Usage

`sf org generate password`

`[--json]`

`[-b ON-BEHALF-OF]`

`[-l LENGTH]`

`[-c COMPLEXITY]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-b | --on-behalf-of ON-BEHALF-OF`

Optional

Comma-separated list of usernames or aliases to assign the password to.

Type: option

`-l | --length LENGTH`

Optional

Number of characters in the generated password; valid values are between 8 and 100.

Type: option

Default value: 13

`-c | --complexity COMPLEXITY`

Optional

Level of password complexity or strength; the higher the value, the stronger the password.

Type: option

Default value: 5

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`org get snapshot` (Pilot)

Get details about a scratch org snapshot.

Note: We provide the `org get snapshot` command to selected customers through an invitation-only pilot program that requires agreement to specific terms and conditions. Pilot programs are subject to change, and we can't guarantee acceptance. The `org get snapshot` command isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features.

Description for `org get snapshot`

Snapshot creation can take a while. Use this command with the snapshot name or ID to check its creation status. After the status changes to Active, you can use the snapshot to create scratch orgs.

To create a snapshot, use the "sf org create snapshot" command. To retrieve a list of all snapshots, use "sf org list snapshot".

Examples for `org get snapshot`

Get snapshot details using its ID and the default Dev Hub org:

```
sf org get snapshot --snapshot 0Oo...
```

Get snapshot details using its name from a Dev Hub org with alias SnapshotDevHub:

```
sf org get snapshot --snapshot Dependencies --target-dev-hub SnapshotDevHub
```

Usage

`sf org get snapshot`

`[--json]`

`-v TARGET-DEV-HUB`

```
[--api-version API-VERSION]
-s SNAPSHOT
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`
Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-s | --snapshot SNAPSHOT`
Required

Name or ID of snapshot to retrieve.

The IDs of scratch org snapshots start with 0Oo.

Type: option

Aliases for `org get snapshot`

```
force:org:snapshot:get
```

`org list`

List all orgs you’ve created or authenticated to.

Examples for `org list`

List all orgs you've created or authenticated to:

```
$ sf org list
```

List all orgs, including expired, deleted, and unknown-status orgs; don't include the connection status:

```
$ sf org list --skip-connection-status --all
```

List orgs and remove local org authorization info about non-active scratch orgs:

```
$ sf org list --clean
```

Usage

`sf org list`

```
[--json]
[--verbose]
[--all]
[--clean]
[-p]
[--skip-connection-status]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`--verbose`

Optional

List more information about each org.

Type: boolean

`--all`

Optional

Include expired, deleted, and unknown-status scratch orgs.

Type: boolean

`--clean`

Optional

Remove all local org authorizations for non-active scratch orgs. Use "org logout" to remove non-scratch orgs.

Type: boolean

`-p | --no-prompt`

Optional

Don't prompt for confirmation.

Type: boolean

`--skip-connection-status`

Optional

Skip retrieving the connection status of non-scratch orgs.

Type: boolean

Aliases for `org list`

```
force:org:list
```

org list auth

List authorization information about the orgs you created or logged into.

Description for **org list auth**

This command uses local authorization information that Salesforce CLI caches when you create a scratch org or log into an org. The command doesn't actually connect to the orgs to verify that they're still active. As a result, this command executes very quickly. If you want to view live information about your authorized orgs, such as their connection status, use the "org list" command.

Examples for **org list auth**

List local authorization information about your orgs:

```
sf org list auth
```

Usage

sf org list auth
`[--json]`

Flags

--json

Optional

Format output as json.

Type: boolean

Aliases for **org list auth**

```
force:auth:list
```

```
auth:list
```

org list metadata

List the metadata components and properties of a specified type.

Description for **org list metadata**

Use this command to identify individual components in your manifest file or if you want a high-level view of particular metadata types in your org. For example, you can use this command to return a list of names of all the CustomObject or Layout components in your org, then use this information in a retrieve command that returns a subset of these components.

The username that you use to connect to the org must have the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Examples for `org list metadata`

List the CustomObject components, and their properties, in the org with alias "my-dev-org":

```
$ sf org list metadata --metadata-type CustomObject --target-org my-dev-org
```

List the CustomObject components in your default org, write the output to the specified file, and use API version 57.0:

```
$ sf org list metadata --metadata-type CustomObject --api-version 57.0 --output-file
/path/to/outputfilename.txt
```

List the Dashboard components in your default org that are contained in the "folderSales" folder, write the output to the specified file, and use API version 57.0:

```
$ sf org list metadata --metadata-type Dashboard --folder folderSales --api-version 57.0
--output-file /path/to/outputfilename.txt
```

Usage

`sf org list metadata`

`[--json]`

`[--api-version API-VERSION]`

`-o TARGET-ORG`

`[-f OUTPUT-FILE]`

`-m METADATA-TYPE`

`[--folder FOLDER]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`--api-version API-VERSION`

Optional

API version to use; default is the most recent API version.

Override the api version used for api requests made by this command

Type: option

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`-f | --output-file OUTPUT-FILE`

Optional

Pathname of the file in which to write the results.

Type: option

`-m | --metadata-type METADATA-TYPE`

Required

Metadata type to be retrieved, such as CustomObject; metadata type names are case-sensitive.

Type: option

`--folder FOLDER`

Optional

Folder associated with the component; required for components that use folders; folder names are case-sensitive.

Examples of metadata types that use folders are Dashboard, Document, EmailTemplate, and Report.

Type: option

Aliases for `org list metadata`

```
force:mdapi:listmetadata
```

`org list metadata-types`

Display details about the metadata types that are enabled for your org.

Description for `org list metadata-types`

The information includes Apex classes and triggers, custom objects, custom fields on standard objects, tab sets that define an app, and many other metadata types. Use this information to identify the syntax needed for a <name> element in a manifest file (package.xml).

The username that you use to connect to the org must have the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Examples for `org list metadata-types`

Display information about all known and enabled metadata types in the org with alias "my-dev-org" using API version 57.0:

```
$ sf org list metadata-types --api-version 57.0 --target-org my-dev-org
```

Display only the metadata types that aren't yet supported by Salesforce CLI in your default org and write the results to the specified file:

```
$ sf org list metadata-types --output-file /path/to/outputfilename.txt --filter-known
```

Usage

`sf org list metadata-types`

`[--json]`

`[--api-version API-VERSION]`

`-o TARGET-ORG`

`[-f OUTPUT-FILE]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

`--api-version API-VERSION`
Optional

API version to use; default is the most recent API version.

Override the api version used for api requests made by this command

Type: option

`-o | --target-org TARGET-ORG`
Required

Username or alias of the target org.

Type: option

`-f | --output-file OUTPUT-FILE`
Optional

Pathname of the file in which to write the results.

Directing the output to a file makes it easier to extract relevant information for your package.xml manifest file. The default output destination is the terminal or command window console.

Type: option

Aliases for `org list metadata-types`

```
force:mdapi:describemetadata
```

`org list shape`

List all org shapes you’ve created.

Description for `org list shape`

The output includes the alias, username, and ID of the source org, the status of the org shape creation, and more. Use the org ID to update your scratch org configuration file so you can create a scratch org based on this org shape.

Examples for `org list shape`

List all org shapes you've created:

```
sf org list shape
```

List all org shapes in JSON format and write the output to a file:

```
sf org list shape --json > tmp/MyOrgShapeList.json
```

Usage

`sf org list shape`
  `[--json]`

Flags

`--json`
  Optional

  Format output as json.

  Type: boolean

Aliases for `org list shape`

```
force:org:shape:list
```

`org list snapshot` (Pilot)

List scratch org snapshots.

Note: We provide the `org list snapshot` command to selected customers through an invitation-only pilot program that requires agreement to specific terms and conditions. Pilot programs are subject to change, and we can't guarantee acceptance. The `org list snapshot` command isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features.

Description for `org list snapshot`

You can view all the snapshots in a Dev Hub that you have access to. If you're an admin, you can see all snapshots associated with the Dev Hub org. If you're a user, you can see only your snapshots unless a Dev Hub admin gives you View All permissions.

To create a snapshot, use the "sf org create snapshot" command. To get details about a snapshot request, use "sf org get snapshot".

Examples for `org list snapshot`

List snapshots in the default Dev Hub:

```
sf org list snapshot
```

List snapshots in the Dev Hub with alias SnapshotDevHub:

```
sf org list snapshot --target-dev-hub SnapshotDevHub
```

Usage

`sf org list snapshot`
  `[--json]`

  `-v TARGET-DEV-HUB`

  `[--api-version API-VERSION]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`
Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `org list snapshot`

```
force:org:snapshot:list
```

`org list users`

List all locally-authenticated users of an org.

Description for `org list users`

For scratch orgs, the list includes any users you've created with the "org create user" command; the original scratch org admin user is marked with "(A)". For other orgs, the list includes the users you used to authenticate to the org.

Examples for `org list users`

List the locally-authenticated users of your default org:

```
sf org list users
```

List the locally-authenticated users of the specified org:

```
sf org list users --target-org me@my.org
```

Usage

`sf org list users`
`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

Flags

`--json`
: Optional

  Format output as json.

  Type: boolean

`-o | --target-org TARGET-ORG`
: Required

  Username or alias of the target org.

  Type: option

`--api-version API-VERSION`
: Optional

  Override the api version used for api requests made by this command

  Type: option

Aliases for `org list users`

```
force:user:list
```

`org login access-token`

Authorize an org using an existing Salesforce access token.

Description for `org login access-token`

By default, the command runs interactively and asks you for the access token. If you previously authorized the org, the command prompts whether you want to overwrite the local file. Specify --no-prompt to not be prompted.

To use the command in a CI/CD script, set the SFDX_ACCESS_TOKEN environment variable to the access token. Then run the command with the --no-prompt parameter.

Examples for `org login access-token`

Authorize an org on https://mycompany.my.salesforce.com; the command prompts you for the access token:

```
sf org login access-token --instance-url https://mycompany.my.salesforce.com
```

Authorize the org without being prompted; you must have previously set the SFDX_ACCESS_TOKEN environment variable to the access token:

```
sf org login access-token --instance-url https://dev-hub.my.salesforce.com --no-prompt
```

Usage

`sf org login access-token`

`[--json]`

`-r INSTANCE-URL`

`[-d]`

```
[-s]
[-a ALIAS]
[-p]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-r | --instance-url INSTANCE-URL`
Required

URL of the instance that the org lives on.

If you specify an --instance-url value, this value overrides the sfdcLoginUrl value in your sfdx-project.json file.

To specify a My Domain URL, use the format https://yourcompanyname.my.salesforce.com.

To specify a sandbox, set --instance-url to https://MyDomainName--SandboxName.sandbox.my.salesforce.com.

Type: option

`-d | --set-default-dev-hub`
Optional

Set the authenticated org as the default Dev Hub.

Type: boolean

`-s | --set-default`
Optional

Set the authenticated org as the default that all org-related commands run against.

Type: boolean

`-a | --alias ALIAS`
Optional

Alias for the org.

Type: option

`-p | --no-prompt`
Optional

Don't prompt for confirmation.

Type: boolean

Aliases for `org login access-token`

```
force:auth:accesstoken:store
```

```
auth:accesstoken:store
```

`org login device`

Authorize an org using a device code.

Description for `org login device`

Use this command to allow a device to connect to an org.

When you run this command, it first displays an 8-digit device code and the URL for verifying the code on your org. The default instance URL is https://login.salesforce.com, so if the org you're authorizing is on a different instance, use the --instance-url. The command waits while you complete the verification. Open a browser and navigate to the displayed verification URL, enter the code, then click Connect. If you aren't already logged into your org, log in, and then you're prompted to allow the device to connect to the org. After you successfully authorize the org, you can close the browser window.

Examples for `org login device`

Authorize an org using a device code, give the org the alias TestOrg1, and set it as your default Dev Hub org:

```
sf org login device --set-default-dev-hub --alias TestOrg1
```

Authorize an org in which you've created a custom connected app with the specified client ID (consumer key):

```
sf org login device --client-id <OAuth client id>
```

Authorize a sandbox org with the specified instance URL:

```
sf org login device --instance-url
https://MyDomainName--SandboxName.sandbox.my.salesforce.com
```

Usage

`sf org login device`

`[--json]`

`[-i CLIENT-ID]`

`[-r INSTANCE-URL]`

`[-d]`

`[-s]`

`[-a ALIAS]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-i | --client-id CLIENT-ID`

Optional

OAuth client ID (also called consumer key) of your custom connected app.

Type: option

`-r | --instance-url INSTANCE-URL`

Optional

URL of the instance that the org lives on.

If you specify an --instance-url value, this value overrides the sfdcLoginUrl value in your sfdx-project.json file.

To specify a My Domain URL, use the format https://yourcompanyname.my.salesforce.com.

To specify a sandbox, set --instance-url to https://MyDomainName--SandboxName.sandbox.my.salesforce.com.

Type: option

`-d | --set-default-dev-hub`

Optional

Set the authenticated org as the default Dev Hub.

Type: boolean

`-s | --set-default`

Optional

Set the authenticated org as the default that all org-related commands run against.

Type: boolean

`-a | --alias ALIAS`

Optional

Alias for the org.

Type: option

Aliases for `org login device`

```
force:auth:device:login
```

```
auth:device:login
```

`org login jwt`

Log in to a Salesforce org using a JSON web token (JWT).

Description for `org login jwt`

Use this command in automated environments where you can't interactively log in with a browser, such as in CI/CD scripts.

Logging into an org authorizes the CLI to run other commands that connect to that org, such as deploying or retrieving a project. You can log into many types of orgs, such as sandboxes, Dev Hubs, Env Hubs, production orgs, and scratch orgs.

Complete these steps before you run this command:

1. Create a digital certificate (also called digital signature) and the private key to sign the certificate. You can use your own key and certificate issued by a certification authority. Or use OpenSSL to create a key and a self-signed digital certificate.

2. Store the private key in a file on your computer. When you run this command, you set the --jwt-key-file flag to this file.

3. Create a custom connected app in your org using the digital certificate. Make note of the consumer key (also called client id) that's generated for you. Be sure the username of the user logging in is approved to use the connected app. When you run this command, you set the --client-id flag to the consumer key.

See https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_auth_jwt_flow.htm for more information.

We recommend that you set an alias when you log into an org. Aliases make it easy to later reference this org when running commands that require it. If you don't set an alias, you use the username that you specified when you logged in to the org. If you run multiple commands that reference the same org, consider setting the org as your default. Use --set-default for your default scratch org or sandbox, or --set-default-dev-hub for your default Dev Hub.

Examples for `org login jwt`

Log into an org with username jdoe@example.org and on the default instance URL (https://login.salesforce.org). The private key is stored in the file /Users/jdoe/JWT/server.key and the command uses the connected app with consumer key (client id) 04580y4051234051.

```
sf org login jwt --username jdoe@example.org --jwt-key-file /Users/jdoe/JWT/server.key
--client-id 04580y4051234051
```

Set the org as the default and give it an alias:

```
sf org login jwt --username jdoe@example.org --jwt-key-file /Users/jdoe/JWT/server.key
--client-id 04580y4051234051 --alias ci-org --set-default
```

Set the org as the default Dev Hub and give it an alias:

```
sf org login jwt --username jdoe@example.org --jwt-key-file /Users/jdoe/JWT/server.key
--client-id 04580y4051234051 --alias ci-dev-hub --set-default-dev-hub
```

Log in to a sandbox using URL https://MyDomainName--SandboxName.sandbox.my.salesforce.com:

```
sf org login jwt --username jdoe@example.org --jwt-key-file /Users/jdoe/JWT/server.key
--client-id 04580y4051234051 --alias ci-org --set-default --instance-url
https://MyDomainName--SandboxName.sandbox.my.salesforce.com
```

Usage

`sf org login jwt`

`[--json]`

`-o USERNAME`

`-f JWT-KEY-FILE`

`-i CLIENT-ID`

`[-r INSTANCE-URL]`

`[-d]`

`[-s]`

`[-a ALIAS]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --username USERNAME`
Required

Username of the user logging in.

Type: option

`-f | --jwt-key-file JWT-KEY-FILE`
Required

Path to a file containing the private key.

Type: option

`-i | --client-id CLIENT-ID`
Required

OAuth client ID (also called consumer key) of your custom connected app.

Type: option

`-r | --instance-url INSTANCE-URL`
Optional

URL of the instance that the org lives on.

If you specify an --instance-url value, this value overrides the sfdcLoginUrl value in your sfdx-project.json file.

To specify a My Domain URL, use the format https://yourcompanyname.my.salesforce.com.

To specify a sandbox, set --instance-url to https://MyDomainName--SandboxName.sandbox.my.salesforce.com.

Type: option

`-d | --set-default-dev-hub`
Optional

Set the authenticated org as the default Dev Hub.

Type: boolean

`-s | --set-default`
Optional

Set the authenticated org as the default that all org-related commands run against.

Type: boolean

`-a | --alias ALIAS`
Optional

Alias for the org.

Type: option

Aliases for `org login jwt`

```
force:auth:jwt:grant
```

```
auth:jwt:grant
```

`org login sfdx-url`

Authorize an org using a Salesforce DX authorization URL stored in a file.

Description for **`org login sfdx-url`**

The Salesforce DX (SFDX) authorization URL must have the format "force://<clientId>:<clientSecret>:<refreshToken>@<instanceUrl>". NOTE: The SFDX authorization URL uses the "force" protocol, and not "http" or "https". Also, the "instanceUrl" inside the SFDX authorization URL doesn't include the protocol ("https://").

You have three options when creating the authorization file. The easiest option is to redirect the output of the "sf org display --verbose --json" command into a file. For example, using an org with alias my-org that you've already authorized:

```
    $ sf org display --target-org my-org --verbose --json > authFile.json
```

The resulting JSON file contains the URL in the "sfdxAuthUrl" property of the "result" object. You can then reference the file when running this command:

```
    $ sf org login sfdx-url --sfdx-url-file authFile.json
```

NOTE: The "sf org display --verbose" command displays the refresh token only for orgs authorized with the web server flow, and not the JWT bearer flow.

You can also create a JSON file that has a top-level property named sfdxAuthUrl whose value is the authorization URL. Finally, you can create a normal text file that includes just the URL and nothing else.

Examples for **`org login sfdx-url`**

Authorize an org using the SFDX authorization URL in the files/authFile.json file:

```
sf org login sfdx-url --sfdx-url-file files/authFile.json
```

Similar to previous example, but set the org as your default and give it an alias MyDefaultOrg:

```
sf org login sfdx-url --sfdx-url-file files/authFile.json --set-default --alias MyDefaultOrg
```

Usage

`sf org login sfdx-url`

`[--json]`

`-f SFDX-URL-FILE`

`[-d]`

`[-s]`

`[-a ALIAS]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-f | --sfdx-url-file SFDX-URL-FILE`

Required

Path to a file that contains the Salesforce DX authorization URL.

Type: option

-d | --set-default-dev-hub
Optional

Set the authenticated org as the default Dev Hub.

Type: boolean

-s | --set-default
Optional

Set the authenticated org as the default that all org-related commands run against.

Type: boolean

-a | --alias ALIAS
Optional

Alias for the org.

Type: option

Aliases for `org login sfdx-url`

```
force:auth:sfdxurl:store
```

```
auth:sfdxurl:store
```

`org login web`

Log in to a Salesforce org using the web server flow.

Description for `org login web`

Opens a Salesforce instance URL in a web browser so you can enter your credentials and log in to your org. After you log in, you can close the browser window.

Logging into an org authorizes the CLI to run other commands that connect to that org, such as deploying or retrieving metadata. You can log into many types of orgs, such as sandboxes, Dev Hubs, Env Hubs, production orgs, and scratch orgs.

We recommend that you set an alias when you log into an org. Aliases make it easy to later reference this org when running commands that require it. If you don’t set an alias, you use the username that you specified when you logged in to the org. If you run multiple commands that reference the same org, consider setting the org as your default. Use --set-default for your default scratch org or sandbox, or --set-default-dev-hub for your default Dev Hub.

By default, this command uses the global out-of-the-box connected app in your org. If you need more security or control, such as setting the refresh token timeout or specifying IP ranges, create your own connected app using a digital certificate. Make note of the consumer key (also called cliend id) that’s generated for you. Then specify the consumer key with the --client-id flag.

Examples for `org login web`

Run the command with no flags to open the default Salesforce login page (https://login.salesforce.com):

```
sf org login web
```

Log in to your Dev Hub, set it as your default Dev Hub, and set an alias that you reference later when you create a scratch org:

```
sf org login web --set-default-dev-hub --alias dev-hub
```

Log in to a sandbox and set it as your default org:

```
sf org login web --instance-url https://MyDomainName--SandboxName.sandbox.my.salesforce.com
 --set-default
```

Use --browser to specify a specific browser, such as Google Chrome:

```
sf org login web --instance-url https://MyDomainName--SandboxName.sandbox.my.salesforce.com
 --set-default --browser chrome
```

Use your own connected app by specifying its consumer key (also called client ID):

```
sf org login web --instance-url https://MyDomainName--SandboxName.sandbox.my.salesforce.com
 --set-default --browser chrome --client-id 04580y4051234051
```

Usage

`sf org login web`

`[--json]`

`[-b BROWSER]`

`[-i CLIENT-ID]`

`[-r INSTANCE-URL]`

`[-d]`

`[-s]`

`[-a ALIAS]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-b | --browser BROWSER`

Optional

Browser in which to open the org.

If you don’t specify --browser, the command uses your default browser. The exact names of the browser applications differ depending on the operating system you're on; check your documentation for details.

Type: option

Permissible values are: chrome, edge, firefox

`-i | --client-id CLIENT-ID`

Optional

OAuth client ID (also called consumer key) of your custom connected app.

Type: option

`-r | --instance-url INSTANCE-URL`

Optional

URL of the instance that the org lives on.

If you specify an --instance-url value, this value overrides the sfdcLoginUrl value in your sfdx-project.json file.

To specify a My Domain URL, use the format https://yourcompanyname.my.salesforce.com.

To specify a sandbox, set --instance-url to https://MyDomainName--SandboxName.sandbox.my.salesforce.com.

Type: option

`-d | --set-default-dev-hub`

Optional

Set the authenticated org as the default Dev Hub.

Type: boolean

`-s | --set-default`

Optional

Set the authenticated org as the default that all org-related commands run against.

Type: boolean

`-a | --alias ALIAS`

Optional

Alias for the org.

Type: option

Aliases for `org login web`

```
force:auth:web:login
```

```
auth:web:login
```

`org logout`

Log out of a Salesforce org.

Description for `org logout`

If you run this command with no flags and no default org set in your config or environment, it first displays a list of orgs you've created or logged into, with none of the orgs selected. Use the arrow keys to scroll through the list and the space bar to select the orgs you want to log out of. Press Enter when you're done; the command asks for a final confirmation before logging out of the selected orgs.

The process is similar if you specify --all, except that in the initial list of orgs, they're all selected. Use --target-org to logout of a specific org. In both these cases by default, you must still confirm that you want to log out. Use --no-prompt to never be asked for confirmation when also using --all or --target-org.

Be careful! If you log out of a scratch org without having access to its password, you can't access the scratch org again, either through the CLI or the Salesforce UI.

Examples for `org logout`

Interactively select the orgs to log out of:

```
sf org logout
```

Log out of the org with username me@my.org:

```
sf org logout --target-org me@my.org
```

Log out of all orgs after confirmation:

```
sf org logout --all
```

Logout of the org with alias my-scratch and don't prompt for confirmation:

```
sf org logout --target-org my-scratch --no-prompt
```

Usage

`sf org logout`

`[--json]`

`[-o TARGET-ORG]`

`[-a]`

`[-p]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Optional

Username or alias of the target org.

Type: option

`-a | --all`

Optional

Include all authenticated orgs.

All orgs includes Dev Hubs, sandboxes, DE orgs, and expired, deleted, and unknown-status scratch orgs.

Type: boolean

`-p | --no-prompt`

Optional

Don't prompt for confirmation.

Type: boolean

Aliases for `org logout`

```
force:auth:logout
```

```
auth:logout
```

`org open`

Open your default scratch org, or another specified org, in a browser.

Description for `org open`

To open a specific page, specify the portion of the URL after "https://MyDomainName.my.salesforce.com/" as the value for the --path flag. For example, specify "--path lightning" to open Lightning Experience, or specify "--path /apex/YourPage" to open a Visualforce page.

Use the --source-file to open a Lightning Page from your local project in Lightning App Builder. Lightning page files have the suffix .flexipage-meta.xml, and are stored in the "flexipages" directory.

To generate a URL but not launch it in your browser, specify --url-only.

To open in a specific browser, use the --browser flag. Supported browsers are "chrome", "edge", and "firefox". If you don't specify --browser, the org opens in your default browser.

Examples for `org open`

Open your default org in your default browser:

```
$ sf org open
```

Open the org with alias MyTestOrg1 in the Firefox browser:

```
$ sf org open --target-org MyTestOrg1 --browser firefox
```

Display the navigation URL for the Lightning Experience page for your default org, but don't open the page in a browser:

```
$ sf org open --url-only --path lightning
```

Open a local Lightning page in your default org's Lightning App Builder:

```
$ sf org open --source-path force-app/main/default/flexipages/Hello.flexipage-meta.xml
```

Usage

`sf org open`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
[-b BROWSER]
[-p PATH]
[-r]
[-f SOURCE-FILE]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-b | --browser BROWSER`
Optional

Browser where the org opens.

Type: option

Permissible values are: chrome, edge, firefox

`-p | --path PATH`
Optional

Navigation URL path to open a specific page.

Type: option

`-r | --url-only`
Optional

Display navigation URL, but don't launch browser.

Type: boolean

`-f | --source-file SOURCE-FILE`
Optional

Path to an ApexPage or FlexiPage to open in Lightning App Builder.

Type: option

Aliases for `org open`

```
force:org:open
```

```
force:source:open
```

`org resume sandbox`

Check the status of a sandbox creation, and log in to it if it's ready.

Description for `org resume sandbox`

Sandbox creation can take a long time. If the original "sf org create sandbox" command either times out, or you specified the --async flag, the command displays a job ID. Use this job ID to check whether the sandbox creation is complete, and if it is, the command then logs into it.

You can also use the sandbox name to check the status or the --use-most-recent flag to use the job ID of the most recent sandbox creation.

Examples for `org resume sandbox`

Check the status of a sandbox creation using its name and specify a production org with alias "prodOrg":

```
sf org resume sandbox --name mysandbox --target-org prodOrg
```

Check the status using the job ID:

```
sf org resume sandbox --job-id 0GRxxxxxxxx
```

Check the status of the most recent sandbox create request:

```
sf org resume sandbox --use-most-recent
```

Usage

`sf org resume sandbox`

```
[--json]
[-w WAIT]
[-n NAME]
[-i JOB-ID]
[-l]
[-o TARGET-ORG]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for the sandbox org to be ready.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you and returns the job ID. To resume checking the sandbox creation, rerun this command.

Type: option

`-n | --name NAME`

Optional

Name of the sandbox org.

Type: option

`-i | --job-id JOB-ID`

Optional

Job ID of the incomplete sandbox creation that you want to check the status of.

The job ID is valid for 24 hours after you start the sandbox creation.

Type: option

`-l | --use-most-recent`

Optional

Use the most recent sandbox create request.

Type: boolean

`-o | --target-org TARGET-ORG`

Optional

Username or alias of the production org that contains the sandbox license.

When it creates the sandbox org, Salesforce copies the metadata, and optionally data, from your production org to the new sandbox org.

Type: option

Aliases for `org resume sandbox`

```
env:resume:sandbox
```

`org resume scratch`

Resume the creation of an incomplete scratch org.

Description for `org resume scratch`

When the original "sf org create scratch" command either times out or is run with the --async flag, it displays a job ID.

Run this command by either passing it a job ID or using the --use-most-recent flag to specify the most recent incomplete scratch org.

Examples for `org resume scratch`

Resume a scratch org create with a job ID:

```
sf org resume scratch --job-id 2SR3u0000008fBDGAY
```

Resume your most recent incomplete scratch org:

```
sf org resume scratch --use-most-recent
```

Usage

`sf org resume scratch`

`[--json]`

`[-i JOB-ID]`

`[-r]`

Flags

`--json`
: Optional

  Format output as json.

  Type: boolean

`-i | --job-id JOB-ID`
: Optional

  Job ID of the incomplete scratch org create that you want to resume.

  The job ID is the same as the record ID of the incomplete scratch org in the ScratchOrgInfo object of the Dev Hub.

  The job ID is valid for 24 hours after you start the scratch org creation.

  Type: option

`-r | --use-most-recent`
: Optional

  Use the job ID of the most recent incomplete scratch org.

  Type: boolean

Aliases for `org resume scratch`

```
env:resume:scratch
```

`package` Commands

Commands to develop and install unlocked packages and managed 2GP packages.

package create
: Create a package.

package delete
: Delete a package.

package install
: Install a version of a package in the target org.

package install report
: Retrieve the status of a package installation request.

package installed list
: List the org's installed packages.

package list
: List all packages in the Dev Hub org.

package uninstall
: Uninstall a second-generation package from the target org.

package uninstall report
: Retrieve the status of a package uninstall request.

package update
Update package details.

package version create
Create a package version in the Dev Hub org.

package version create list
List package version creation requests.

package version create report
Retrieve details about a package version creation request.

package version delete
Delete a package version.

package version displayancestry
Display the ancestry tree for a 2GP managed package version.

package version list
List all package versions in the Dev Hub org.

package version promote
Promote a package version to released.

package version report
Retrieve details about a package version in the Dev Hub org.

package version retrieve
Retrieve package metadata for a specified package version.

package version update
Update a package version.

`package create`

Create a package.

Description for `package create`

First, use this command to create a package. Then create a package version.

If you don’t have a namespace defined in your sfdx-project.json file, use --no-namespace.

Your --name value must be unique within your namespace.

Run 'sf package list to list all packages in the Dev Hub org.

Examples for `package create`

Create an unlocked package from the files in the "force-app" directory; uses your default Dev Hub org:

```
sf package create --name MyUnlockedPackage --package-type Unlocked --path force-app
```

Create a managed packaged from the "force-app" directory files, give the package a description, and use the specified Dev Hub org:

```
sf package create --name MyManagedPackage --description "Your Package Descripton"
--package-type Managed --path force-app --target-dev-hub devhub@example.com
```

Usage

```
sf package create
   [--json]
   -v TARGET-DEV-HUB
   [--api-version API-VERSION]
   -n NAME
   -t PACKAGE-TYPE
   [-d DESCRIPTION]
   [-e]
   -r PATH
   [--org-dependent]
   [-o ERROR-NOTIFICATION-USERNAME]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-n | --name NAME`

Required

Name of the package to create.

Type: option

`-t | --package-type PACKAGE-TYPE`

Required

Type of package.

The options for package type are Managed and Unlocked (Managed=DeveloperManagedSubscriberManaged, Unlocked=DeveloperControlledSubscriberEditable). These options determine upgrade and editability rules.

Type: option

Permissible values are: Managed, Unlocked

`-d | --description DESCRIPTION`

Optional

Description of the package.

Type: option

-e | --no-namespace
Optional

Create the package with no namespace; available only for unlocked packages.

This flag is useful when you're migrating an existing org to packages. But use a namespaced package for new metadata.

Type: boolean

-r | --path PATH
Required

Path to directory that contains the contents of the package.

Type: option

--org-dependent
Optional

Depends on unpackaged metadata in the installation org; applies to unlocked packages only.

Use Source Tracking in Sandboxes to develop your org-dependent unlocked package. For more information, see "Create Org-Dependent Unlocked Packages" in the Salesforce DX Developer Guide.

Type: boolean

-o | --error-notification-username ERROR-NOTIFICATION-USERNAME
Optional

Active Dev Hub user designated to receive email notifications for package errors.

Email notifications include information about unhandled Apex exceptions, and install, upgrade, or uninstall failures associated with your package.

Type: option

Aliases for `package create`

```
force:package:create
```

`package delete`

Delete a package.

Description for `package delete`

Specify the ID or alias of the package you want to delete.

Delete unlocked and second-generation managed packages. Before you delete a package, first delete all associated package versions.

Examples for `package delete`

Delete a package using its alias from your default Dev Hub org:

```
sf package delete --package "Your Package Alias"
```

Delete a package using its ID from the specified Dev Hub org:

```
sf package delete --package 0Ho... --target-dev-hub devhub@example.com
```

Usage

`sf package delete`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
[-n]
-p PACKAGE
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-n | --no-prompt`

Optional

Don’t prompt before deleting the package.

Type: boolean

`-p | --package PACKAGE`

Required

ID (starts with 0Ho) or alias of the package to delete.

Type: option

Aliases for `package delete`

```
force:package:delete
```

`package install`

Install a version of a package in the target org.

Description for **`package install`**

To install a package, specify a specific version of the package using the 04t package ID. The package and the version you specified installs in your default target org unless you supply the username for a different target org.

For package upgrades, to specify options for component deprecation or deletion of removed components, include an --upgrade-type value. To delete components that can be safely deleted and deprecate the others, specify --upgrade-type Mixed (the default). To deprecate all removed components, specify --upgrade-type DeprecateOnly. To delete all removed components, except for custom objects and custom fields, that don't have dependencies, specify --upgrade-type Delete. (Note: This option can result in the loss of data that is associated with the deleted components.) The default is Mixed.

Examples for **`package install`**

Install a package version with the specified ID in the org with username "me@example.com":

```
sf package install --package 04t... --target-org me@example.com
```

Install a package version with the specified alias into your default org:

```
sf package install --package awesome_package_alias
```

Install a package version with an alias that includes spaces into your default org:

```
sf package install --package "Awesome Package Alias"
```

Install an unlocked package version with the specified ID and deprecate all removed components:

```
sf package install --package 04t... --upgrade-type DeprecateOnly
```

Usage

`sf package install`
```
  [--json]
  -o TARGET-ORG
  [--api-version API-VERSION]
  [-w WAIT]
  [-k INSTALLATION-KEY]
  [-b PUBLISH-WAIT]
  [-r]
  -p PACKAGE
  [-a APEX-COMPILE]
  [-s SECURITY-TYPE]
  [-t UPGRADE-TYPE]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-w | --wait WAIT`

Optional

Number of minutes to wait for installation status.

Type: option

Default value: 0 minutes

`-k | --installation-key INSTALLATION-KEY`

Optional

Installation key for key-protected package (default: null).

Type: option

`-b | --publish-wait PUBLISH-WAIT`

Optional

Maximum number of minutes to wait for the Subscriber Package Version ID to become available in the target org before canceling the install request.

Type: option

Default value: 0 minutes

`-r | --no-prompt`

Optional

Don't prompt for confirmation.

Allows the following without an explicit confirmation response: 1) Remote Site Settings and Content Security Policy websites to send or receive data, and 2) --upgrade-type Delete to proceed.

Type: boolean

`-p | --package PACKAGE`

Required

ID (starts with 04t) or alias of the package version to install.

Type: option

`-a | --apex-compile APEX-COMPILE`

Optional

Compile all Apex in the org and package, or only Apex in the package; unlocked packages only.

Applies to unlocked packages only. Specifies whether to compile all Apex in the org and package, or only the Apex in the package.

For package installs into production orgs, or any org that has Apex Compile on Deploy enabled, the platform compiles all Apex in the org after the package install or upgrade operation completes.

This approach assures that package installs and upgrades don’t impact the performance of an org, and is done even if --apex-compile package is specified.

Type: option

Permissible values are: all, package

Default value: all

`-s | --security-type SECURITY-TYPE`

Optional

Security access type for the installed package. (deprecation notice: The default --security-type value will change from AllUsers to AdminsOnly in v47.0 or later.)

Type: option

Permissible values are: AllUsers, AdminsOnly

Default value: AdminsOnly

`-t | --upgrade-type UPGRADE-TYPE`

Optional

Upgrade type for the package installation; available only for unlocked packages.

For package upgrades, specifies whether to mark all removed components as deprecated (DeprecateOnly), to delete removed components that can be safely deleted and deprecate the others (Mixed), or to delete all removed components, except for custom objects and custom fields, that don't have dependencies (Delete). The default is Mixed. Can specify DeprecateOnly or Delete only for unlocked package upgrades.

Type: option

Permissible values are: DeprecateOnly, Mixed, Delete

Default value: Mixed

Aliases for `package install`

```
force:package:install
```

`package install report`

Retrieve the status of a package installation request.

Examples for `package install report`

Retrieve the status of a package installation request with the specified ID on your default org:

```
sf package install report --request-id 0Hf...
```

Similar to previous example, except use the org with username me@example.com:

```
sf package install report --request-id 0Hf... --target-org me@example.com
```

Usage

`sf package install report`

`[--json]`

```
-o TARGET-ORG
[--api-version API-VERSION]
-i REQUEST-ID
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-i | --request-id REQUEST-ID`
Required

ID of the package install request you want to check; starts with 0Hf.

Type: option

Aliases for `package install report`

```
force:package:install:report
```

`package installed list`

List the org's installed packages.

Examples for `package installed list`

List the installed packages in your default org:

```
sf package installed list
```

List the installed packages in the org with username me@example.com:

```
sf package installed list --target-org me@example.com
```

Usage

`sf package installed list`

```
[--json]
```

```
-o TARGET-ORG
[--api-version API-VERSION]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `package installed list`

```
force:package:installed:list
```

`package list`

List all packages in the Dev Hub org.

Description for `package list`

Description

Examples for `package list`

List all packages in the specified Dev Hub org:

```
sf package list --target-dev-hub devhub@example.com
```

List all packages details in the specified Dev Hub org, and show extended details about each package:

```
sf package list --target-dev-hub devhub@example.com --verbose
```

Usage

`sf package list`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
```

```
[--verbose]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`
Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`--verbose`
Optional

Display extended package detail.

Type: boolean

Aliases for `package list`

```
force:package:list
```

`package uninstall`

Uninstall a second-generation package from the target org.

Description for `package uninstall`

Specify the package ID for a second-generation package.

To list the org’s installed packages, run "sf package installed list".

To uninstall a first-generation package, from Setup, enter Installed Packages in the Quick Find box, then select Installed Packages.

Examples for `package uninstall`

Uninstall a package with specified ID from an org with username me@example.com:

```
sf package uninstall --package 04t... --target-org me@example.com
```

Uninstall a package with the specified alias from your default org:

```
sf package uninstall --package undesirable_package_alias
```

Uninstall a package with an alias that contains spaces from your default org:

```
sf package uninstall --package "Undesirable Package Alias"
```

Usage

`sf package uninstall`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`[-w WAIT]`

`-p PACKAGE`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-w | --wait WAIT`

Optional

Number of minutes to wait for uninstall status.

Type: option

Default value: 0 minutes

`-p | --package PACKAGE`

Required

ID (starts with 04t) or alias of the package version to uninstall.

Type: option

Aliases for `package uninstall`

```
force:package:uninstall
```

`package uninstall report`

Retrieve the status of a package uninstall request.

Examples for `package uninstall report`

Retrieve the status of a package uninstall in your default org using the specified request ID:

```
sf package uninstall report --request-id 06y...
```

Similar to previous example, but use the org with username me@example.com:

```
sf package uninstall report --request-id 06y... --target-org me@example.com
```

Usage

`sf package uninstall report`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`-i REQUEST-ID`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-i | --request-id REQUEST-ID`

Required

ID of the package uninstall request you want to check; starts with 06y.

Type: option

Aliases for `package uninstall report`

```
force:package:uninstall:report
```

`package update`

Update package details.

Description for `package update`

Specify a new value for each option you want to update.

Run "sf package list" to list all packages in the Dev Hub org.

Examples for `package update`

Update the name of the package with the specified alias; uses your default Dev Hub org:

```
sf package update --package "Your Package Alias" --name "New Package Name"
```

Update the description of the package with the specified ID; uses the specified Dev Hub org:

```
sf package update --package 0Ho... --description "New Package Description" --target-dev-hub
 devhub@example.com
```

Usage

`sf package update`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
-p PACKAGE
[-n NAME]
[-d DESCRIPTION]
[-o ERROR-NOTIFICATION-USERNAME]
[--enable-app-analytics]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-p | --package PACKAGE`

Required

ID (starts with 0Ho) or alias of the package to update.

Type: option

`-n | --name NAME`

Optional

New name of the package.

Type: option

`-d | --description DESCRIPTION`

Optional

New description of the package.

Type: option

`-o | --error-notification-username ERROR-NOTIFICATION-USERNAME`

Optional

Active Dev Hub user designated to receive email notifications for package errors.

Email notifications include information about unhandled Apex exceptions, and install, upgrade, or uninstall failures associated with your package.

Type: option

`--enable-app-analytics`

Optional

Enable AppExchange App Analytics usage data collection on this managed package and its components.

Type: boolean

Aliases for `package update`

```
force:package:update
```

`package version create`

Create a package version in the Dev Hub org.

Description for `package version create`

The package version is based on the package contents in the specified directory.

To retrieve details about a package version create request, including status and package version ID (04t), run "sf package version create report -i 08c...".

We recommend that you specify the --installation-key parameter to protect the contents of your package and to prevent unauthorized installation of your package.

To list package version creation requests in the org, run "sf package version create list".

To promote a package version to released, you must use the --code-coverage parameter. The package must also meet the code coverage requirements. This requirement applies to both managed and unlocked packages.

We don’t calculate code coverage for org-dependent unlocked packages, or for package versions that specify --skip-validation.

Examples for `package version create`

Create a package version from the contents of the "common" directory and give it an installation key of "password123"; uses your default Dev Hub org:

```
sf package version create --path common --installation-key password123
```

Create a package version from a package with the specified alias; uses the Dev Hub org with username devhub@example.com:

```
sf package version create --package "Your Package Alias" --installation-key password123
--target-dev-hub devhub@example.com
```

Create a package version from a package with the specified ID:

```
sf package version create --package 0Ho... --installation-key password123
```

Create a package version and skip the validation step:

```
sf package version create --path common --installation-key password123 --skip-validation
```

Usage

```
sf package version create
   [--json]
   -v TARGET-DEV-HUB
   [--api-version API-VERSION]
   [-b BRANCH]
   [-c]
   [-f DEFINITION-FILE]
   [-k INSTALLATION-KEY]
   [-x]
   [-p PACKAGE]
   [-d PATH]
   [--post-install-script POST-INSTALL-SCRIPT]
   [--post-install-url POST-INSTALL-URL]
   [--releasenotes-url RELEASENOTES-URL]
   [--skip-ancestor-check]
   [--skip-validation]
   [-t TAG]
   [--uninstall-script UNINSTALL-SCRIPT]
   [-e VERSION-DESCRIPTION]
   [-a VERSION-NAME]
   [-n VERSION-NUMBER]
```

```
[-w WAIT]
[--language LANGUAGE]
[--verbose]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-b | --branch BRANCH`

Optional

Name of the branch in your source control system that the package version is based on.

Type: option

`-c | --code-coverage`

Optional

Calculate and store the code coverage percentage by running the packaged Apex tests included in this package version.

Before you can promote and release a managed or unlocked package version, the Apex code must meet a minimum 75% code coverage requirement. We don't calculate code coverage for org-dependent unlocked packages or for package versions that specify --skip-validation.

Type: boolean

`-f | --definition-file DEFINITION-FILE`

Optional

Path to a definition file similar to scratch org definition file that contains the list of features and org preferences that the metadata of the package version depends on.

Type: option

`-k | --installation-key INSTALLATION-KEY`

Optional

Installation key for key-protected package. (either --installation-key or --installation-key-bypass is required)

Type: option

`-x | --installation-key-bypass`

Optional

Bypass the installation key requirement. (either --installation-key or --installation-key-bypass is required)

If you bypass this requirement, anyone can install your package.

Type: boolean

`-p | --package PACKAGE`

Optional

ID (starts with 0Ho) or alias of the package to create a version of.

Type: option

`-d | --path PATH`

Optional

Path to the directory that contains the contents of the package.

Type: option

`--post-install-script POST-INSTALL-SCRIPT`

Optional

Name of the post-install script; applies to managed packages only.

The post-install script is an Apex class within this package that is run in the installing org after installations or upgrades of this package version.

Type: option

`--post-install-url POST-INSTALL-URL`

Optional

Post-install instructions URL.

The contents of the post-installation instructions URL are displayed in the UI after installation of the package version.

Type: option

`--releasenotes-url RELEASENOTES-URL`

Optional

Release notes URL.

This link is displayed in the package installation UI to provide release notes for this package version to subscribers.

Type: option

`--skip-ancestor-check`

Optional

Overrides ancestry requirements, which allows you to specify a package ancestor that isn't the highest released package version.

Type: boolean

`--skip-validation`

Optional

Skip validation during package version creation; you can't promote unvalidated package versions.

Skips validation of dependencies, package ancestors, and metadata during package version creation. Skipping validation reduces the time it takes to create a new package version, but you can promote only validated package versions. Skipping validation can suppress important errors that can surface at a later stage. You can specify skip validation or code coverage, but not both. Code coverage is calculated during validation.

Type: boolean

`-t | --tag TAG`

Optional

Package version’s tag.

Type: option

`--uninstall-script UNINSTALL-SCRIPT`

Optional

Uninstall script name; applies to managed packages only.

The uninstall script is an Apex class within this package that is run in the installing org after uninstallations of this package.

Type: option

`-e | --version-description VERSION-DESCRIPTION`

Optional

Description of the package version to be created; overrides the sfdx-project.json value.

Type: option

`-a | --version-name VERSION-NAME`

Optional

Name of the package version to be created; overrides the sfdx-project.json value.

Type: option

`-n | --version-number VERSION-NUMBER`

Optional

Version number of the package version to be created; overrides the sfdx-project.json value.

Type: option

`-w | --wait WAIT`

Optional

Number of minutes to wait for the package version to be created.

Type: option

Default value: 0 minutes

`--language LANGUAGE`

Optional

Language for the package.

Specify the language using a language code listed under "Supported Languages" in Salesforce Help. If no language is specified, the language defaults to the language of the Dev Hub user who created the package.

Type: option

`--verbose`

Optional

Display verbose command output.

Display verbose command output. When polling for the status of the creation, this will output status and timeout data on a separate line for each poll request, which is useful in CI systems where timeouts can occur with long periods of no output from commands.

Type: boolean

Aliases for `package version create`

```
force:package:version:create
```

package version create list

List package version creation requests.

Description for package version create list

Shows the details of each request to create a package version in the Dev Hub org.

All filter parameters are applied using the AND logical operator (not OR).

To get information about a specific request, run "sf package version create report" and supply the request ID.

Examples for package version create list

List all package version creation requests in your default Dev Hub org:

```
sf package version create list
```

List package version creation requests from the last 3 days in the Dev Hub org with username devhub@example.com:

```
sf package version create list --created-last-days 3 --target-dev-hub
```

List package version creation requests with status Error:

```
sf package version create list --status Error
```

List package version creation requests with status InProgress:

```
sf package version create list --status InProgress
```

List package version creation requests with status Success that were created today:

```
sf package version create list --created-last-days 0 --status Success
```

Usage

`sf package version create list`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
[-c CREATED-LAST-DAYS]
[-s STATUS]
[--show-conversions-only]
[--verbose]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

-v | --target-dev-hub TARGET-DEV-HUB

Required

Username or alias of the Dev Hub org.

Type: option

--api-version API-VERSION

Optional

Override the api version used for api requests made by this command

Type: option

-c | --created-last-days CREATED-LAST-DAYS

Optional

Number of days since the request was created, starting at 00:00:00 of first day to now. Use 0 for today.

Type: option

-s | --status STATUS

Optional

Status of the version creation request, used to filter the list.

Type: option

Permissible values are: Queued, InProgress, Success, Error

--show-conversions-only

Optional

Filter the list output to display only converted package version.

Type: boolean

--verbose

Optional

Displays additional information at a slight performance cost, such as the version name and number for each package version create request.

Type: boolean

Aliases for `package version create list`

```
force:package:version:create:list
```

`package version create report`

Retrieve details about a package version creation request.

Description for `package version create report`

Specify the request ID for which you want to view details. If applicable, the command displays errors related to the request.

To show all requests in the org, run "sf package version create list".

Examples for `package version create report`

Retrieve details about the package version creation request with the specified ID; uses your default Dev Hub org:

```
sf package version create report --package-create-request-id 08c...
```

Retrieve details about the specified package version creation request in the Dev Hub org with username devhub@example.com:

```
sf package version create report --package-create-request-id 08c... --target-dev-hub
devhub@example.com
```

Usage

`sf package version create report`

`[--json]`

`-v TARGET-DEV-HUB`

`[--api-version API-VERSION]`

`-i PACKAGE-CREATE-REQUEST-ID`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-i | --package-create-request-id PACKAGE-CREATE-REQUEST-ID`

Required

ID (starts with 08c) of the package version creation request you want to display.

Type: option

Aliases for `package version create report`

```
force:package:version:create:report
```

`package version delete`

Delete a package version.

Description for `package version delete`

Specify the ID or alias of the package version you want to delete.

Examples for `package version delete`

Delete a package version with the specified alias using your default Dev Hub org:

```
sf package version delete --package "Your Package Alias"
```

Delete a package version with the specified ID using the Dev Hub org with username "devhub@example.com":

```
sf package version delete --package 04t... --target-org devhub@example.com
```

Usage

`sf package version delete`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
[-n]
-p PACKAGE
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-n | --no-prompt`

Optional

Don’t prompt before deleting the package version.

Type: boolean

`-p | --package PACKAGE`

Required

ID (starts with 04t) or alias of the package version to delete.

Type: option

Aliases for **`package version delete`**

```
force:package:version:delete
```

`package version displayancestry`

Display the ancestry tree for a 2GP managed package version.

Examples for **`package version displayancestry`**

Display the ancestry tree for a package version with the specified alias, using your default Dev Hub org:

```
sf package version displayancestry --package package_version_alias
```

Similar to previous example, but display the output in DOT code:

```
sf package version displayancestry --package package_version_alias --dot-code
```

Display the ancestry tree for a package with the specified ID, using the Dev Hub org with username devhub@example.com:

```
sf package version displayancestry --package OHo... --target-dev-hub devhub@example.com
```

Display the ancestry tree of a package version with the specified ID, using your default Dev Hub org:

```
sf package version displayancestry --package 04t...
```

Usage

`sf package version displayancestry`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
-p PACKAGE
[--dot-code]
[--verbose]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-p | --package PACKAGE`

Required

ID or alias of the package (starts with 0Ho) or package version (starts with 04t) to display ancestry for.

If you specify a package ID (starts with 0Ho) or alias, the ancestor tree for every package version associated with the package ID is displayed. If you specify a package version (starts with 04t) or alias, the ancestry tree of the specified package version is displayed.

Type: option

`--dot-code`

Optional

Display the ancestry tree in DOT code.

You can use the DOT code output in graph visualization software to create tree visualizations.

Type: boolean

`--verbose`

Optional

Display both the package version ID (starts with 04t) and the version number (major.minor.patch.build) in the ancestry tree.

Type: boolean

Aliases for `package version displayancestry`

```
force:package:version:displayancestry
```

`package version list`

List all package versions in the Dev Hub org.

Description for `package version list`

Description

Examples for `package version list`

List package versions in your default Dev Hub org that were created in the last 3 days; show only the released versions and order the list using the PatchVersion field. Display extended details about each package version:

```
sf package version list --verbose --created-last-days 3 --released --order-by PatchVersion
```

List the released package versions for the two specified packages that were modified today; use the Dev Hub org with username devhub@example.com:

```
sf package version list --packages 0Ho000000000000,0Ho000000000001 --released
--modified-last-days 0 --target-dev-hub devhub@example.com
```

List all released package versions in your default Dev Hub org:

```
sf package version list --released
```

List package versions that were modified today in your default Dev Hub org; show limited details about each one:

```
sf package version list --concise --modified-last-days 0
```

List released package versions that were created in the last 3 days in your default Dev Hub org; show limited details:

```
sf package version list --concise --created-last-days 3 --released
```

List released package versions that were modified today for the two packages with specified aliases in your default Dev Hub org:

```
sf package version list --packages exp-mgr,exp-mgr-util --released --modified-last-days 0
```

Usage

`sf package version list`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
[-c CREATED-LAST-DAYS]
[--concise]
[--show-conversions-only]
[-m MODIFIED-LAST-DAYS]
[-p PACKAGES]
[-r]
[-o ORDER-BY]
[--verbose]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-c | --created-last-days CREATED-LAST-DAYS`

Optional

Number of days since the request was created, starting at 00:00:00 of first day to now. Use 0 for today.

Type: option

`--concise`
Optional

Display limited package version details.

Type: boolean

`--show-conversions-only`
Optional

Filter the list output to display only converted package version.

Type: boolean

`-m | --modified-last-days MODIFIED-LAST-DAYS`
Optional

Number of days since the items were modified, starting at 00:00:00 of first day to now. Use 0 for today.

Type: option

`-p | --packages PACKAGES`
Optional

Comma-delimited list of packages (aliases or 0Ho IDs) to list.

Type: option

`-r | --released`
Optional

Display released versions only (IsReleased=true).

Type: boolean

`-o | --order-by ORDER-BY`
Optional

Package version fields used to order the list.

Type: option

`--verbose`
Optional

Display extended package version details.

Type: boolean

Aliases for `package version list`

```
force:package:version:list
```

`package version promote`

Promote a package version to released.

Description for `package version promote`

Supply the ID or alias of the package version you want to promote. Promotes the package version to released status.

Examples for `package version promote`

Promote the package version with the specified ID to released; uses your default Dev Hub org:

```
sf package version promote --package 04t...
```

Promote the package version with the specified alias to released; uses the Dev Hub org with username devhub@example.com:

```
sf package version promote --package awesome_package_alias --target-dev-hub
devhub@example.com
```

Promote the package version with an alias that has spaces to released:

```
sf package version promote --package "Awesome Package Alias"
```

Usage

`sf package version promote`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
-p PACKAGE
[-n]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-p | --package PACKAGE`

Required

ID (starts with 04t) or alias of the package version to promote.

Type: option

`-n | --no-prompt`

Optional

Don't prompt to confirm setting the package version as released.

Type: boolean

Aliases for `package version promote`

```
force:package:version:promote
```

`package version report`

Retrieve details about a package version in the Dev Hub org.

Description for `package version report`

To update package version values, run "sf package version update".

Examples for `package version report`

Retrieve details about the package version with the specified ID from your default Dev Hub org:

```
sf package version report --package 04t...
```

Retrieve details about the package version with the specified alias (that contains spaces) from the Dev Hub org with username devhub@example.com:

```
sf package version report --package "Your Package Alias" --target-dev-hub devhub@example.com
```

Usage

`sf package version report`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
-p PACKAGE
[--verbose]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

-p | --package PACKAGE

Required

ID (starts with 04t) or alias of the package to retrieve details for.

Type: option

--verbose

Optional

Display extended package version details.

Type: boolean

Aliases for `package version report`

```
force:package:version:report
```

`package version retrieve`

Retrieve package metadata for a specified package version.

Description for `package version retrieve`

Retrieving a package version downloads the metadata into the directory you specify.

You can retrieve metadata for a second- or first-generation managed package, or an unlocked package.

Specify the subscriber package version ID (starts with 04t) and the path to an empty directory when you run this command.

Examples for `package version retrieve`

Retrieve package metadata for a subscriber package version ID (starts with 04t) into my-folder/ within your Salesforce DX project directory:

```
sf package version retrieve --package 04t... --output-dir my-folder --target-org my-scratch
```

If you omit --target-org, this command runs against your default org.

Usage

sf package version retrieve

[--json]

[--api-version API-VERSION]

-o TARGET-ORG

-p PACKAGE

[-d OUTPUT-DIR]

Flags

--json

Optional

Format output as json.

Type: boolean

--api-version API-VERSION
Optional

Override the api version used for api requests made by this command

Type: option

-o | --target-org TARGET-ORG
Required

Username or alias of the target org.

Type: option

-p | --package PACKAGE
Required

Subscriber package version ID (starts with 04t).

Type: option

-d | --output-dir OUTPUT-DIR
Optional

Path within your Salesforce DX project directory in which to download the metadata. This directory must be empty.

Type: option

Default value: force-app

package version update

Update a package version.

Description for `package version update`

Specify a new value for each option you want to update.

To display details about a package version, run "sf package version display".

Examples for `package version update`

Update the package version that has the specified alias (that contains spaces) with a new installation key "password123"; uses your default Dev Hub org:

```
sf package version update --package "Your Package Alias" --installation-key password123
```

Update the package version that has the specified ID with a new branch and tag; use the Dev Hub org with username devhub@example.com:

```
sf package version update --package 04t... --branch main --tag 'Release 1.0.7'
--target-dev-hub devhub@example.com
```

Update the package version that has the specified ID with a new description:

```
sf package version update --package 04t... --version-description "New Package Version
Description"
```

Usage

`sf package version update`

```
[--json]
-v TARGET-DEV-HUB
[--api-version API-VERSION]
-p PACKAGE
[-a VERSION-NAME]
[-e VERSION-DESCRIPTION]
[-b BRANCH]
[-t TAG]
[-k INSTALLATION-KEY]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-v | --target-dev-hub TARGET-DEV-HUB`

Required

Username or alias of the Dev Hub org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-p | --package PACKAGE`

Required

ID (starts with 04t) or alias of the package to update a version of.

Type: option

`-a | --version-name VERSION-NAME`

Optional

New package version name.

Type: option

`-e | --version-description VERSION-DESCRIPTION`

Optional

New package version description.

Type: option

`-b | --branch BRANCH`

Optional

New package version branch.

Type: option

`-t | --tag TAG`

Optional

New package version tag.

Type: option

`-k | --installation-key INSTALLATION-KEY`

Optional

New installation key for key-protected package (default: null)

Type: option

Aliases for `package version update`

```
force:package:version:update
```

`package1` Commands

Commands to develop first-generation managed and unmanaged packages.

package1 version create
Create a first-generation package version in the release org.

package1 version create get
Retrieve the status of a package version creation request.

package1 version display
Display details about a first-generation package version.

package1 version list
List package versions for the specified first-generation package or for the org.

`package1 version create`

Create a first-generation package version in the release org.

Description for `package1 version create`

The package version is based on the contents of the specified metadata package. Omit --managed-released if you want to create an unmanaged package version.

Examples for `package1 version create`

Create a first-generation package version from the package with the specified ID and name the package version "example"; use your default org:

```
sf package1 version create --package-id 033... --name example
```

Same as previous example, but provide a description and wait for 30 minutes for the package version to be created; use the specified org:

```
sf package1 version create --package-id 033... --name example --description "example
description" --wait 30 --target-org myorg@example.com
```

Usage

`sf package1 version create`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
-i PACKAGE-ID
-n NAME
[-d DESCRIPTION]
[-v VERSION]
[-m]
[-r RELEASE-NOTES-URL]
[-p POST-INSTALL-URL]
[-k INSTALLATION-KEY]
[-w WAIT]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-i | --package-id PACKAGE-ID`

Required

ID of the metadata package (starts with 033) of which you're creating a new version.

Type: option

`-n | --name NAME`

Required

Package version name.

Type: option

`-d | --description DESCRIPTION`

Optional

Package version description.

Type: option

`-v | --version VERSION`

Optional

Package version in major.minor format, for example, 3.2.

Type: option

`-m | --managed-released`

Optional

Create a managed package version.

To create a beta version, don't include this parameter.

Type: boolean

`-r | --release-notes-url RELEASE-NOTES-URL`

Optional

Release notes URL.

This link is displayed in the package installation UI to provide release notes for this package version to subscribers.

Type: option

`-p | --post-install-url POST-INSTALL-URL`

Optional

Post install URL.

The contents of the post-installation instructions URL are displayed in the UI after installation of the package version.

Type: option

`-k | --installation-key INSTALLATION-KEY`

Optional

Installation key for key-protected package (default: null).

Type: option

`-w | --wait WAIT`

Optional

Minutes to wait for the package version to be created (default: 2 minutes).

Type: option

Aliases for `package1 version create`

```
force:package1:version:create
```

`package1 version create get`

Retrieve the status of a package version creation request.

Examples for `package1 version create get`

Get the status of the creation request for the package version with the specified ID in your default org:

```
sf package1 version create get --request-id 0HD...
```

Same as previous example, but use the specified org:

```
sf package1 version create get --request-id 0HD... --target-org myorg@example.com
```

Usage

`sf package1 version create get`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
-i REQUEST-ID
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-i | --request-id REQUEST-ID`

Required

ID of the PackageUploadRequest (starts with 0HD).

Type: option

Aliases for `package1 version create get`

```
force:package1:version:create:get
```

`package1 version display`

Display details about a first-generation package version.

Examples for `package1 version display`

Display details about the first-generation package version with the specified ID in your default org:

```
sf package1 version display --package-version-id 04t...
```

Same as previous example, but use the specified org:

```
sf package1 version display --package-version-id 04t... --target-org myorg@example.com
```

Usage

`sf package1 version display`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
-i PACKAGE-VERSION-ID
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-i | --package-version-id PACKAGE-VERSION-ID`

Required

ID (starts with 04t) of the metadata package version whose details you want to display.

Type: option

Aliases for `package1 version display`

```
force:package1:version:display
```

`package1 version list`

List package versions for the specified first-generation package or for the org.

Examples for `package1 version list`

List all first-generation package versions in your default org:

```
sf package1 version list
```

List package versions for the specified first-generation package in the specifief org:

```
sf package1 version list --package-id 033... --target-org myorg@example.com
```

Usage

`sf package1 version list`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`[-i PACKAGE-ID]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-i | --package-id PACKAGE-ID`

Optional

Metadata package ID (starts with 033) whose package versions you want to list.

If not specified, shows all versions for all packages (managed and unmanaged) in the org.

Type: option

Aliases for `package1 version list`

```
force:package1:version:list
```

`plugins` Commands

Find and manage plugins

plugins discover
See a list of 3rd-party sf plugins you can install.

`plugins discover`

See a list of 3rd-party sf plugins you can install.

Examples for `plugins discover`

```
sf plugins discover
```

Usage

`sf plugins discover`
`[--json]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

`project` Commands

Work with projects, such as deploy and retrieve metadata.

project convert mdapi
Convert metadata retrieved via Metadata API into the source format used in Salesforce DX projects.

project convert source
Convert source-formatted files into metadata that you can deploy using Metadata API.

project delete source
Delete source from your project and from a non-source-tracked org.

project delete tracking
Delete all local source tracking information.

project deploy cancel
Cancel a deploy operation.

project deploy pipeline quick (Beta)
Quickly deploy a validated deployment to an org.

`project convert mdapi`

Convert metadata retrieved via Metadata API into the source format used in Salesforce DX projects.

Description for `project convert mdapi`

To use Salesforce CLI to work with components that you retrieved via Metadata API, first convert your files from the metadata format to the source format using this command.

To convert files from the source format back to the metadata format, run "sf project convert source".

To convert multiple metadata components, either set multiple --metadata <name> flags or a single --metadata flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --manifest and --source-dir.

Examples for `project convert mdapi`

Convert metadata formatted files in the specified directory into source formatted files; writes converted files to your default package directory:

```
$ sf project convert mdapi --root-dir path/to/metadata
```

Similar to previous example, but writes converted files to the specified output directory:

```
$ sf project convert mdapi --root-dir path/to/metadata --output-dir path/to/outputdir
```

Usage

`sf project convert mdapi`

`[--json]`

`[--api-version API-VERSION]`

`-r ROOT-DIR`

`[-d OUTPUT-DIR]`

`[-x MANIFEST]`

`[-p METADATA-DIR]`

`[-m METADATA]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-r | --root-dir ROOT-DIR`

Required

Root directory that contains the Metadata API–formatted metadata.

Type: option

`-d | --output-dir OUTPUT-DIR`

Optional

Directory to store your files in after they're converted to source format; can be an absolute or relative path.

Type: option

-x | --manifest MANIFEST

Optional

File path to manifest (package.xml) of metadata types to convert.

If you specify this parameter, don't specify --metadata or --source-dir.

Type: option

-p | --metadata-dir METADATA-DIR

Optional

Root of directory or zip file of metadata formatted files to convert.

The supplied paths can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all metadata types in the directory and its sub-directories).

If you specify this flag, don't specify --manifest or --metadata. If the comma-separated list you're supplying contains spaces, enclose the entire comma-separated list in one set of double quotes.

Type: option

-m | --metadata METADATA

Optional

Metadata component names to convert.

Type: option

Aliases for `project convert mdapi`

```
force:mdapi:convert
```

`project convert source`

Convert source-formatted files into metadata that you can deploy using Metadata API.

Description for `project convert source`

To convert source-formatted files into the metadata format, so that you can deploy them using Metadata API, run this command. Then deploy the metadata using "sf project deploy".

To convert Metadata API–formatted files into the source format, run "sf project convert mdapi".

To specify a package name that includes spaces, enclose the name in single quotes.

To convert multiple components, either set multiple --metadata <name> flags or a single --metadata flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --manifest and --source-dir.

Examples for `project convert source`

Convert source-formatted files in the specified directory into metadata-formatted files; writes converted files into a new directory:

```
$ sf project convert source --root-dir path/to/source
```

Similar to previous example, but writes converted files to the specified output directory and associates the files with the specified package:

```
$ sf project convert source --root-dir path/to/source --output-dir path/to/outputdir
--package-name 'My Package'
```

Usage

sf project convert source

```
[--json]
[--api-version API-VERSION]
[-r ROOT-DIR]
[-d OUTPUT-DIR]
[-n PACKAGE-NAME]
[-x MANIFEST]
[-p SOURCE-DIR]
[-m METADATA]
```

Flags

--json

Optional

Format output as json.

Type: boolean

--api-version API-VERSION

Optional

API Version to use in the generated project's manifest. By default, will use the version from sfdx-project.json

Override the api version used for api requests made by this command

Type: option

-r | --root-dir ROOT-DIR

Optional

Source directory other than the default package to convert.

Type: option

-d | --output-dir OUTPUT-DIR

Optional

Output directory to store the Metadata API–formatted files in.

Type: option

Default value: metadataPackage_1694018706254

-n | --package-name PACKAGE-NAME

Optional

Name of the package to associate with the metadata-formatted files.

Type: option

-x | --manifest MANIFEST

Optional

Path to the manifest (package.xml) file that specifies the metadata types to convert.

If you specify this parameter, don’t specify --metadata or --source-dir.

Type: option

`-p | --source-dir SOURCE-DIR`

Optional

Paths to the local source files to convert.

The supplied paths can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all metadata types in the directory and its sub-directories).

If you specify this parameter, don't specify --manifest or --metadata.

Type: option

`-m | --metadata METADATA`

Optional

Metadata component names to convert.

Type: option

Aliases for `project convert source`

```
force:source:convert
```

`project delete source`

Delete source from your project and from a non-source-tracked org.

Description for `project delete source`

Use this command to delete components from orgs that don't have source tracking. To remove deleted items from orgs that have source tracking enabled, "sf project deploy start".

When you run this command, both the local source file and the metadata component in the org are deleted.

To delete multiple metadata components, either set multiple --metadata <name> flags or a single --metadata flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --manifest and --source-dir.

Examples for `project delete source`

Delete all local Apex source files and all Apex classes from the org with alias "my-scratch":

```
sf project delete source --metadata ApexClass --target-org my-scratch
```

Delete a specific Apex class and a Profile that has a space in it from your default org; don't prompt for confirmation:

```
sf project delete source --metadata ApexClass:MyFabulousApexClass --metadata "Profile: My
 Profile" --no-prompt
```

Run the tests that aren't in any managed packages as part of the deletion; if the delete succeeds, and the org has source-tracking enabled, update the source tracking information:

```
sf project delete source --metadata ApexClass --test-level RunLocalTests --track-source
```

Delete the Apex source files in a directory and the corresponding components from your default org:

```
sf project delete source --source-dir force-app/main/default/classes
```

Usage

`sf project delete source`

```
[--json]
[--api-version API-VERSION]
-o TARGET-ORG
[-c]
[-w WAIT]
[--tests TESTS]
[-l TEST-LEVEL]
[-r]
[-m METADATA]
[-p SOURCE-DIR]
[-t]
[-f]
[--verbose]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`-c | --check-only`

Optional

Validate delete command but don't delete anything from the org or the local project.

IMPORTANT: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

Validates the deleted metadata and runs all Apex tests, but prevents the deletion from being saved to the org.

If you change a field type from Master-Detail to Lookup or vice versa, that change isn't supported when using the --chec-konly parameter to test a deletion (validation). This kind of change isn't supported for test deletions to avoid the risk of data loss or corruption. If a change that isn't supported for test deletions is included in a deletion package, the test deletion fails and issues an error.

If your deletion package changes a field type from Master-Detail to Lookup or vice versa, you can still validate the changes prior to deploying to Production by performing a full deletion to another test Sandbox. A full deletion includes a validation of the changes as part of the deletion process.

Note: A Metadata API deletion that includes Master-Detail relationships deletes all detail records in the Recycle Bin in the following cases.

1. For a deletion with a new Master-Detail field, soft delete (send to the Recycle Bin) all detail records before proceeding to delete the Master-Detail field, or the deletion fails. During the deletion, detail records are permanently deleted from the Recycle Bin and cannot be recovered.

2. For a deletion that converts a Lookup field relationship to a Master-Detail relationship, detail records must reference a master record or be soft-deleted (sent to the Recycle Bin) for the deletion to succeed. However, a successful deletion permanently deletes any detail records in the Recycle Bin.

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for the command to finish.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you.

Type: option

Default value: 33 minutes

`--tests TESTS`

Optional

Apex tests to run when --test-level is RunSpecifiedTests.

If a test name contains a space, enclose it in double quotes.

For multiple test names, use one of the following formats:

- Repeat the flag for multiple test names: --tests Test1 --tests Test2 --tests "Test With Space"

- Separate the test names with spaces: --tests Test1 Test2 "Test With Space"

Type: option

`-l | --test-level TEST-LEVEL`

Optional

Deployment Apex testing level.

Valid values are:

- NoTestRun — No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial orgs. This test level is the default for development environments.

- RunSpecifiedTests — Runs only the tests that you specify with the --tests flag. Code coverage requirements differ from the default coverage requirements when using this test level. Executed tests must comprise a minimum of 75% code coverage for each class and trigger in the deployment package. This coverage is computed for each class and trigger individually and is different than the overall coverage percentage.

- RunLocalTests — All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default for production deployments that include Apex classes or triggers.

- RunAllTestsInOrg — All tests in your org are run, including tests of managed packages.

If you don’t specify a test level, the default behavior depends on the contents of your deployment package and target org. For more information, see “Running Tests in a Deployment” in the Metadata API Developer Guide.

Type: option

Permissible values are: NoTestRun, RunSpecifiedTests, RunLocalTests, RunAllTestsInOrg

`-r | --no-prompt`

Optional

Don't prompt for delete confirmation.

Type: boolean

`-m | --metadata METADATA`

Optional

Metadata components to delete.

If you specify this parameter, don’t specify --source-dir.

Type: option

`-p | --source-dir SOURCE-DIR`

Optional

Source file paths to delete.

The supplied paths can be a single file (in which case the operation is applied to only one file) or a folder (in which case the operation is applied to all metadata types in the directory and its sub-directories).

If you specify this parameter, don’t specify --metadata.

Type: option

`-t | --track-source`

Optional

If the delete succeeds, update the source tracking information.

Type: boolean

`-f | --force-overwrite`

Optional

Ignore conflict warnings and overwrite changes to the org.

Type: boolean

`--verbose`

Optional

Verbose output of the delete result.

Type: boolean

Aliases for `project delete source`

```
force:source:delete
```

`project delete tracking`

Delete all local source tracking information.

Description for `project delete tracking`

WARNING: This command deletes or overwrites all existing source tracking files. Use with extreme caution.

Deletes all local source tracking information. When you next run 'project deploy preview', Salesforce CLI displays all local and remote files as changed, and any files with the same name are listed as conflicts.

Usage

```
sf project delete tracking
   [--json]
   [--api-version API-VERSION]
   -o TARGET-ORG
   [-p]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`-p | --no-prompt`

Optional

Don't prompt for source tracking override confirmation.

Type: boolean

Aliases for `project delete tracking`

```
force:source:tracking:clear
```

`project deploy cancel`

Cancel a deploy operation.

Description for `project deploy cancel`

Use this command to cancel a deploy operation that hasn't yet completed in the org. Deploy operations include standard deploys, quick deploys, deploy validations, and deploy cancellations.

Run this command by either passing it a job ID or specifying the --use-most-recent flag to use the job ID of the most recent deploy operation.

Examples for `project deploy cancel`

Cancel a deploy operation using a job ID:

```
sf project deploy cancel --job-id 0Af0x000017yLUFCA2
```

Cancel the most recent deploy operation:

```
sf project deploy cancel --use-most-recent
```

Usage

`sf project deploy cancel`

`[--json]`

`[--async]`

`[-i JOB-ID]`

`[-r]`

`[-w WAIT]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`--async`

Optional

Run the command asynchronously.

The command immediately returns the control of the terminal to you. This way, you can continue to use the CLI. To resume watching the cancellation, run "sf project deploy resume". To check the status of the cancellation, run "sf project deploy report".

Type: boolean

`-i | --job-id JOB-ID`

Optional

Job ID of the deploy operation you want to cancel.

These commands return a job ID if they time out or you specified the --async flag:

- sf project deploy start

- sf project deploy validate

- sf project deploy quick

- sf project deploy cancel

The job ID is valid for 10 days from when you started the deploy operation.

Type: option

`-r | --use-most-recent`

Optional

Use the job ID of the most recent deploy operation.

For performance reasons, this flag uses job IDs for deploy operations that started only in the past 3 days or less. If your most recent deploy operations was more than 3 days ago, this flag won't find a job ID.

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for the command to complete and display results.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you. To resume watching the cancellation, run "sf project deploy resume". To check the status of the cancellation, run "sf project deploy report".

Type: option

Aliases for `project deploy cancel`

```
deploy:metadata:cancel
```

`project deploy pipeline quick` (Beta)

Quickly deploy a validated deployment to an org.

Note: This feature is a Beta Service. Customers may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms (https://www.salesforce.com/company/legal/agreements/).

Description for `project deploy pipeline quick`

Before you run this command, first create a validated deployment with the "sf project deploy pipeline validate" command, which returns a job ID. Validated deployments haven't been deployed to the org yet; you deploy them with this command. Either pass the job ID to this command or use the --use-most-recent flag to use the job ID of the most recently validated deployment. For the quick deploy to succeed, the associated validated deployment must also have succeeded.

Executing this quick deploy command takes less time than a standard deploy because it skips running Apex tests. These tests were previously run as part of the validation. Validating first and then running a quick deploy is useful if the deployment to your production org take several hours and you don’t want to risk a failed deploy.

This command doesn't support source-tracking. The source you deploy overwrites the corresponding metadata in your org. This command doesn’t attempt to merge your source with the versions in your org.

Examples for `project deploy pipeline quick`

Run a quick deploy using your default Devops Center org and a job ID:

```
sf project deploy pipeline quick --job-id 0Af0x000017yLUFCA2
```

Asynchronously run a quick deploy of the most recently validated deployment using an org with alias "my-prod-org":

```
sf project deploy pipeline quick --async --use-most-recent --devops-center-username
my-prod-org
```

Usage

sf project deploy pipeline quick

[--json]

[--async]

[--concise]

[--verbose]

[-w WAIT]

-c DEVOPS-CENTER-USERNAME

[-i JOB-ID]

[-r]

Flags

--json

Optional

Format output as json.

Type: boolean

--async

Optional

Run the command asynchronously.

The command immediately returns the job ID and control of the terminal to you. This way, you can continue to use the CLI. To resume the deployment, run "sf project deploy pipeline resume". To check the status of the deployment, run "sf project deploy pipeline report".

Type: boolean

--concise

Optional

Show concise output of the command result.

Type: boolean

--verbose

Optional

Show verbose output of the command result.

Type: boolean

-w | --wait WAIT

Optional

Number of minutes to wait for command to complete and display results.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you and returns the job ID. To check the status of the operation, run "sf <%= command.id.split(' ').slice(0, -1).join(' ') %> report".

Type: option

Default value: 33 minutes

`-c | --devops-center-username DEVOPS-CENTER-USERNAME`

Required

Username or alias of the DevOps Center org.

Type: option

`-i | --job-id JOB-ID`

Optional

Job ID of the validated deployment to quick deploy.

The job ID is valid for 10 days from when you started the validation.

Type: option

`-r | --use-most-recent`

Optional

Use the job ID of the most recently validated deployment.

For performance reasons, this flag uses only job IDs that were validated in the past 3 days or less. If your most recent deployment validation was more than 3 days ago, this flag won't find the job ID.

Type: boolean

`project deploy pipeline report` (Beta)

Check the status of a pipeline deploy operation.

Note: This feature is a Beta Service. Customers may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms (https://www.salesforce.com/company/legal/agreements/).

Description for `project deploy pipeline report`

Run this command by either indicating a job ID or specifying the —use-most-recent flag to use the job ID of the most recent deploy operation.

Examples for `project deploy pipeline report`

Check the status using a job ID:

```
sf project deploy pipeline report --devops-center-username MyStagingSandbox --job-id
0Af0x000017yLUFCA2
```

Check the status of the most recent deploy operation:

```
sf project deploy pipeline report --devops-center-username MyStagingSandbox --use-most-recent
```

Usage

`sf project deploy pipeline report`

```
[--json]

-c DEVOPS-CENTER-USERNAME

[-i JOB-ID]

[-r]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-c | --devops-center-username DEVOPS-CENTER-USERNAME`

Required

Username or alias of the DevOps Center org.

Type: option

`-i | --job-id JOB-ID`

Optional

Job ID of the pipeline deployment to check the status of.

The job ID is valid for 10 days from when you started the deploy operation.

Type: option

`-r | --use-most-recent`

Optional

Use the job ID of the most recent deploy operation.

For performance reasons, this flag uses job IDs for deploy operations that started in the past 3 days or fewer. If your most recent operation was longer than 3 days ago, this flag won't find the job ID.

Type: boolean

`project deploy pipeline resume` (Beta)

Resume watching a pipeline deploy operation.

Note: This feature is a Beta Service. Customers may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms (https://www.salesforce.com/company/legal/agreements/).

Description for `project deploy pipeline resume`

Use this command to resume watching a pipeline deploy operation if the original command times out or you specified the --async flag.

Run this command by either indicating a job ID or specifying the --use-most-recent flag to use the job ID of the most recent deploy operation.

Examples for `project deploy pipeline resume`

Resume watching a deploy operation using a job ID:

```
sf project deploy pipeline resume --job-id 0Af0x000017yLUFCA2
```

Resume watching the most recent deploy operation:

```
sf project deploy pipeline resume --use-most-recent
```

Usage

`sf project deploy pipeline resume`

`[--json]`

`-c DEVOPS-CENTER-USERNAME`

`[-i JOB-ID]`

`[-r]`

`[--concise]`

`[--verbose]`

`[-w WAIT]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-c | --devops-center-username DEVOPS-CENTER-USERNAME`

Required

Username or alias of the DevOps Center org.

Type: option

`-i | --job-id JOB-ID`

Optional

Job ID of the pipeline deploy operation you want to resume.

These commands return a job ID if they time out or you specified the --async flag:

- sf project deploy pipeline start

- sf project deploy pipeline validate

- sf project deploy pipeline quick

The job ID is valid for 10 days from when you started the deploy operation.

Type: option

`-r | --use-most-recent`

Optional

Use the job ID of the most recent deploy operation.

For performance reasons, this flag uses job IDs for operations that started in the past 3 days or fewer. If your most recent operation was longer than 3 days ago, this flag won't find a job ID.

Type: boolean

`--concise`
Optional

Show concise output of the command result.

Type: boolean

`--verbose`
Optional

Show verbose output of the command result.

Type: boolean

`-w | --wait WAIT`
Optional

Number of minutes to wait for command to complete and display results.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you and returns the job ID. To check the status of the operation, run "sf <%= command.id.split(' ').slice(0, -1).join(' ') %> report".

Type: option

Default value: 33 minutes

`project deploy pipeline start` (Beta)

Deploy changes from a branch to the pipeline stage’s org.

Note: This feature is a Beta Service. Customers may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms (https://www.salesforce.com/company/legal/agreements/).

Description for `project deploy pipeline start`

Before you run this command, changes in the pipeline stage's branch must be merged in the source control repository.

Examples for `project deploy pipeline start`

Deploy changes in the Staging branch to the Staging environment (sandbox), if the previous stage is the bundling stage:

```
sf project deploy pipeline start --devops-center-project-name “Recruiting App” --branch-name
 staging --devops-center-username MyStagingSandbox --bundle-version-name 1.0
```

Deploy all changes in the main branch to the release environment:

```
sf project deploy pipeline start --devops-center-project-name “Recruiting App” --branch-name
 main --devops-center-username MyReleaseOrg --deploy-all
```

Usage

`sf project deploy pipeline start`
`[--json]`

```
-b BRANCH-NAME
[-v BUNDLE-VERSION-NAME]
[-a]
-p DEVOPS-CENTER-PROJECT-NAME
-c DEVOPS-CENTER-USERNAME
[-t TESTS]
[-l TEST-LEVEL]
[--async]
[-w WAIT]
[--verbose]
[--concise]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-b | --branch-name BRANCH-NAME`

Required

Name of the branch in the source control repository that corresponds to the pipeline stage that you want to deploy the changes to.

Type: option

`-v | --bundle-version-name BUNDLE-VERSION-NAME`

Optional

Version name of the bundle.

You must indicate the bundle version if deploying to the environment that corresponds to the first stage after the bundling stage.

Type: option

`-a | --deploy-all`

Optional

Deploy all metadata in the branch to the stage's org.

If you don’t specify this flag, only changes in the stage’s branch are deployed.

Type: boolean

`-p | --devops-center-project-name DEVOPS-CENTER-PROJECT-NAME`

Required

Name of the DevOps Center project.

Type: option

`-c | --devops-center-username DEVOPS-CENTER-USERNAME`

Required

Username or alias of the DevOps Center org.

Type: option

`-t | --tests TESTS`

Optional

Apex tests to run when --test-level is RunSpecifiedTests.

Separate multiple test names with commas. Enclose the entire flag value in double quotes if a test name contains spaces.

Type: option

`-l | --test-level TEST-LEVEL`

Optional

Deployment Apex testing level.

Valid values are:

- NoTestRun — No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial orgs. This test level is the default for development environments.

- RunSpecifiedTests — Runs only the tests that you specify with the --run-tests flag. Code coverage requirements differ from the default coverage requirements when using this test level. Executed tests must comprise a minimum of 75% code coverage for each class and trigger in the deployment package. This coverage is computed for each class and trigger individually and is different than the overall coverage percentage.

- RunLocalTests — All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default for production deployments that include Apex classes or triggers.

- RunAllTestsInOrg — All tests in your org are run, including tests of managed packages.

If you don’t specify a test level, the default behavior depends on the contents of your deployment package. For more information, see [Running Tests in a Deployment](https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/meta_deploy_running_tests.htm) in the "Metadata API Developer Guide".

Type: option

Permissible values are: NoTestRun, RunSpecifiedTests, RunLocalTests, RunAllTestsInOrg

`--async`

Optional

Run the command asynchronously.

The command immediately returns the job ID and control of the terminal to you. This way, you can continue to use the CLI. To resume the deployment, run "sf project deploy pipeline resume". To check the status of the deployment, run "sf project deploy pipeline report".

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for command to complete and display results.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you and returns the job ID. To check the status of the operation, run "sf <%= command.id.split(' ').slice(0, -1).join(' ') %> report".

Type: option

Default value: 33 minutes

`--verbose`

Optional

Show verbose output of the command result.

Type: boolean

`--concise`

Optional

Show concise output of the command result.

Type: boolean

`project deploy pipeline validate` (Beta)

Perform a validate-only deployment from a branch to the pipeline stage's org.

Note: This feature is a Beta Service. Customers may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms (https://www.salesforce.com/company/legal/agreements/).

Description for `project deploy pipeline validate`

A validation runs Apex tests to verify whether a deployment will succeed without actually deploying the metadata to your environment, so you can then quickly deploy the changes later without re-running the tests.

Examples for `project deploy pipeline validate`

Perform a validate-only deployment from the Staging branch to the Staging environment (sandbox):

```
sf project deploy pipeline validate --devops-center-project-name “Recruiting App”
--branch-name staging --devops-center-username MyStagingSandbox
```

Perform a validate-only deployment of all changes from the main branch to the release environment:

```
sf project deploy pipeline validate --devops-center-project-name “Recruiting App”
--branch-name main --devops-center-username MyReleaseOrg --deploy-all
```

Usage

`sf project deploy pipeline validate`

`[--json]`

`-b BRANCH-NAME`

`[-v BUNDLE-VERSION-NAME]`

`[-a]`

`-p DEVOPS-CENTER-PROJECT-NAME`

`-c DEVOPS-CENTER-USERNAME`

`[-t TESTS]`

`[-l TEST-LEVEL]`

`[--async]`

`[-w WAIT]`

`[--verbose]`

```
[--concise]
```

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-b | --branch-name BRANCH-NAME`
Required

Name of the branch in the source control repository that corresponds to the pipeline stage that you want to deploy the changes to.

Type: option

`-v | --bundle-version-name BUNDLE-VERSION-NAME`
Optional

Version name of the bundle.

You must indicate the bundle version if deploying to the environment that corresponds to the first stage after the bundling stage.

Type: option

`-a | --deploy-all`
Optional

Deploy all metadata in the branch to the stage's org.

If you don’t specify this flag, only changes in the stage’s branch are deployed.

Type: boolean

`-p | --devops-center-project-name DEVOPS-CENTER-PROJECT-NAME`
Required

Name of the DevOps Center project.

Type: option

`-c | --devops-center-username DEVOPS-CENTER-USERNAME`
Required

Username or alias of the DevOps Center org.

Type: option

`-t | --tests TESTS`
Optional

Apex tests to run when --test-level is RunSpecifiedTests.

Separate multiple test names with commas. Enclose the entire flag value in double quotes if a test name contains spaces.

Type: option

`-l | --test-level TEST-LEVEL`
Optional

Deployment Apex testing level.

Valid values are:

- NoTestRun — No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial orgs. This test level is the default for development environments.

- RunSpecifiedTests — Runs only the tests that you specify with the --run-tests flag. Code coverage requirements differ from the default coverage requirements when using this test level. Executed tests must comprise a minimum of 75% code coverage for each class and trigger in the deployment package. This coverage is computed for each class and trigger individually and is different than the overall coverage percentage.

- RunLocalTests — All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default for production deployments that include Apex classes or triggers.

- RunAllTestsInOrg — All tests in your org are run, including tests of managed packages.

If you don't specify a test level, the default behavior depends on the contents of your deployment package. For more information, see [Running Tests in a Deployment](https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/meta_deploy_running_tests.htm) in the "Metadata API Developer Guide".

Type: option

Permissible values are: NoTestRun, RunSpecifiedTests, RunLocalTests, RunAllTestsInOrg

`--async`

Optional

Run the command asynchronously.

The command immediately returns the job ID and control of the terminal to you. This way, you can continue to use the CLI. To resume the deployment, run "sf project deploy pipeline resume". To check the status of the deployment, run "sf project deploy pipeline report".

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for command to complete and display results.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you and returns the job ID. To check the status of the operation, run "sf <%= command.id.split(' ').slice(0, -1).join(' ') %> report".

Type: option

Default value: 33 minutes

`--verbose`

Optional

Show verbose output of the command result.

Type: boolean

`--concise`

Optional

Show concise output of the command result.

Type: boolean

project deploy preview

Preview a deployment to see what will deploy to the org, the potential conflicts, and the ignored files.

Description for `project deploy preview`

You must run this command from within a project.

The command outputs a table that describes what will happen if you run the "sf project deploy start" command. The table lists the metadata components that will be deployed and deleted. The table also lists the current conflicts between files in your local project and components in the org. Finally, the table lists the files that won't be deployed because they're included in your .forceignore file.

If your org allows source tracking, then this command displays potential conflicts between the org and your local project. Some orgs, such as production org, never allow source tracking. Source tracking is enabled by default on scratch and sandbox orgs; you can disable source tracking when you create the orgs by specifying the --no-track-source flag on the "sf org create scratch|sandbox" commands.

To preview the deployment of multiple metadata components, either set multiple --metadata <name> flags or a single --metadata flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --manifest and --source-dir.

Examples for `project deploy preview`

NOTE: The commands to preview a deployment and actually deploy it use similar flags. We provide a few preview examples here, but see the help for "sf project deploy start" for more examples that you can adapt for previewing.

Preview the deployment of source files in a directory, such as force-app, to your default org:

```
sf project deploy preview  --source-dir force-app
```

Preview the deployment of all Apex classes to an org with alias "my-scratch":

```
sf project deploy preview --metadata ApexClass --target-org my-scratch
```

Preview deployment of a specific Apex class:

```
sf project deploy preview --metadata ApexClass:MyApexClass
```

Preview deployment of all components listed in a manifest:

```
sf project deploy preview --manifest path/to/package.xml
```

Usage

`sf project deploy preview`

```
  [--json]
  [-c]
  [-x MANIFEST]
  [-m METADATA]
  [-d SOURCE-DIR]
  -o TARGET-ORG
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

-c | --ignore-conflicts

Optional

Don't display conflicts in preview of the deployment.

This flag applies only to orgs that allow source tracking. It has no effect on orgs that don't allow it, such as production orgs.

Type: boolean

-x | --manifest MANIFEST

Optional

Full file path for manifest (package.xml) of components to preview.

All child components are included. If you specify this flag, don't specify --metadata or --source-dir.

Type: option

-m | --metadata METADATA

Optional

Metadata component names to preview.

Type: option

-d | --source-dir SOURCE-DIR

Optional

Path to the local source files to preview.

The supplied path can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all metadata types in the directory and its subdirectories).

If you specify this flag, don't specify --metadata or --manifest.

Type: option

-o | --target-org TARGET-ORG

Required

Login username or alias for the target org.

Overrides your default org.

Type: option

Aliases for `project deploy preview`

```
deploy:metadata:preview
```

`project deploy quick`

Quickly deploy a validated deployment to an org.

Description for `project deploy quick`

Before you run this command, first create a validated deployment with the "sf project deploy validate" command, which returns a job ID. Validated deployments haven't been deployed to the org yet; you deploy them with this command. Either pass the job ID to this command or use the --use-most-recent flag to use the job ID of the most recently validated deployment. For the quick deploy to succeed, the associated validated deployment must also have succeeded.

Executing this quick deploy command takes less time than a standard deploy because it skips running Apex tests. These tests were previously run as part of the validation. Validating first and then running a quick deploy is useful if the deployment to your production org take several hours and you don't want to risk a failed deploy.

This command doesn't support source-tracking. The source you deploy overwrites the corresponding metadata in your org. This command doesn't attempt to merge your source with the versions in your org.

Examples for `project deploy quick`

Run a quick deploy to your default org using a job ID:

```
sf project deploy quick --job-id 0Af0x000017yLUFCA2
```

Asynchronously run a quick deploy of the most recently validated deployment to an org with alias "my-prod-org":

```
sf project deploy quick --async --use-most-recent --target-org my-prod-org
```

Usage

`sf project deploy quick`

```
[--json]
[--async]
[--concise]
[-i JOB-ID]
[-o TARGET-ORG]
[-r]
[--verbose]
[-w WAIT]
[-a API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`--async`

Optional

Run the command asynchronously.

The command immediately returns the control of the terminal to you. This way, you can continue to use the CLI. To resume watching the deploy, run "sf project deploy resume". To check the status of the deploy, run "sf project deploy report".

Type: boolean

`--concise`

Optional

Show concise output of the deploy result.

Type: boolean

`-i | --job-id JOB-ID`

Optional

Job ID of the deployment you want to quick deploy.

The job ID is valid for 10 days from when you started the validation.

Type: option

`-o | --target-org TARGET-ORG`

Optional

Login username or alias for the target org.

Overrides your default org.

Type: option

`-r | --use-most-recent`

Optional

Use the job ID of the most recently validated deployment.

For performance reasons, this flag uses only job IDs that were validated in the past 3 days or less. If your most recent deployment validation was more than 3 days ago, this flag won't find a job ID.

Type: boolean

`--verbose`

Optional

Show verbose output of the deploy result.

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for the command to complete and display results.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you. To resume watching the deploy, run "sf project deploy resume". To check the status of the deploy, run "sf project deploy report".

Type: option

Default value: 33 minutes

`-a | --api-version API-VERSION`

Optional

Target API version for the deploy.

Use this flag to override the default API version with the API version of your package.xml file. The default API version is the latest version supported by the CLI.

Type: option

Aliases for `project deploy quick`

```
deploy:metadata:quick
```

`project deploy report`

Check the status of a deploy operation.

Description for `project deploy report`

Deploy operations include standard deploys, quick deploys, deploy validations, and deploy cancellations.

Run this command by either passing it a job ID or specifying the --use-most-recent flag to use the job ID of the most recent deploy operation.

Examples for `project deploy report`

Check the status using a job ID:

```
sf project deploy report --job-id 0Af0x000017yLUFCA2
```

Check the status of the most recent deploy operation:

```
sf project deploy report --use-most-recent
```

Usage

`sf project deploy report`

```
[--json]
[-i JOB-ID]
[-r]
[--coverage-formatters COVERAGE-FORMATTERS]
[--junit]
[--results-dir RESULTS-DIR]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-i | --job-id JOB-ID`

Optional

Job ID of the deploy operation you want to check the status of.

These commands return a job ID if they time out or you specified the --async flag:

- sf project deploy start

- sf project deploy validate

- sf project deploy quick

- sf project deploy cancel

The job ID is valid for 10 days from when you started the deploy operation.

Type: option

-r | --use-most-recent

Optional

Use the job ID of the most recent deploy operation.

For performance reasons, this flag uses job IDs for deploy operations that started only in the past 3 days or less. If your most recent operation was more than 3 days ago, this flag won't find a job ID.

Type: boolean

--coverage-formatters COVERAGE-FORMATTERS

Optional

Format of the code coverage results.

For multiple formatters, repeat the flag for each formatter.

--coverage-formatters lcov --coverage-formatters clover

Type: option

Permissible values are: clover, cobertura, html-spa, html, json, json-summary, lcovonly, none, teamcity, text, text-summary

--junit

Optional

Output JUnit test results.

Type: boolean

--results-dir RESULTS-DIR

Optional

Output directory for code coverage and JUnit results; defaults to the deploy ID.

Type: option

Aliases for `project deploy report`

```
deploy:metadata:report
```

`project deploy resume`

Resume watching a deploy operation.

Description for `project deploy resume`

Use this command to resume watching a deploy operation if the original command times out or you specified the --async flag. Deploy operations include standard deploys, quick deploys, deploy validations, and deploy cancellations. This command doesn't resume the original operation itself, because the operation always continues after you've started it, regardless of whether you're watching it or not.

Run this command by either passing it a job ID or specifying the --use-most-recent flag to use the job ID of the most recent deploy operation.

Examples for `project deploy resume`

Resume watching a deploy operation using a job ID:

```
sf project deploy resume --job-id 0Af0x000017yLUFCA2
```

Resume watching the most recent deploy operation:

```
sf project deploy resume --use-most-recent
```

Usage

```
sf project deploy resume
   [--json]
   [--concise]
   [-i JOB-ID]
   [-r]
   [--verbose]
   [-w WAIT]
   [--coverage-formatters COVERAGE-FORMATTERS]
   [--junit]
   [--results-dir RESULTS-DIR]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`--concise`

Optional

Show concise output of the deploy operation result.

Type: boolean

`-i | --job-id JOB-ID`

Optional

Job ID of the deploy operation you want to resume.

These commands return a job ID if they time out or you specified the --async flag:

- sf project deploy start

- sf project deploy validate

- sf project deploy quick

- sf project deploy cancel

The job ID is valid for 10 days from when you started the deploy operation.

Type: option

`-r | --use-most-recent`
Optional

Use the job ID of the most recent deploy operation.

For performance reasons, this flag uses job IDs for deploy operations that started only in the past 3 days or less. If your most recent operation was more than 3 days ago, this flag won't find a job ID.

Type: boolean

`--verbose`
Optional

Show verbose output of the deploy operation result.

Type: boolean

`-w | --wait WAIT`
Optional

Number of minutes to wait for the command to complete and display results.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you. To resume watching the deploy operation, run this command again. To check the status of the deploy operation, run "sf project deploy report".

Type: option

`--coverage-formatters COVERAGE-FORMATTERS`
Optional

Format of the code coverage results.

For multiple formatters, repeat the flag for each formatter.

--coverage-formatters lcov --coverage-formatters clover

Type: option

Permissible values are: clover, cobertura, html-spa, html, json, json-summary, lcovonly, none, teamcity, text, text-summary

`--junit`
Optional

Output JUnit test results.

Type: boolean

`--results-dir RESULTS-DIR`
Optional

Output directory for code coverage and JUnit results; defaults to the deploy ID.

Type: option

Aliases for `project deploy resume`

```
deploy:metadata:resume
```

`project deploy start`

Deploy metadata to an org from your local project.

Description for `project deploy start`

You must run this command from within a project.

Metadata components are deployed in source format by default. Deploy them in metadata format by specifying the --metadata-dir flag, which specifies the root directory or ZIP file that contains the metadata formatted files you want to deploy.

If your org allows source tracking, then this command tracks the changes in your source. Some orgs, such as production org, never allow source tracking. Source tracking is enabled by default on scratch and sandbox orgs; you can disable source tracking when you create the orgs by specifying the --no-track-source flag on the "sf org create scratch|sandbox" commands.

To deploy multiple metadata components, either set multiple --metadata <name> flags or a single --metadata flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --manifest and --source-dir.

Examples for `project deploy start`

Deploy local changes not in the org; uses your default org:

```
sf project deploy start
```

Deploy the source files in a directory to an org with alias "my-scratch":

```
sf project deploy start  --source-dir path/to/source --target-org my-scratch
```

Deploy a specific Apex class and the objects whose source is in a directory (both examples are equivalent):

```
sf project deploy start --source-dir path/to/apex/classes/MyClass.cls path/to/source/objects
```

```
sf project deploy start --source-dir path/to/apex/classes/MyClass.cls --source-dir
path/to/source/objects
```

Deploy all Apex classes:

```
sf project deploy start --metadata ApexClass
```

Deploy a specific Apex class:

```
sf project deploy start --metadata ApexClass:MyApexClass
```

Deploy specific Apex classes that match a pattern; in this example, deploy Apex classes whose names contain the string "MyApex":

```
sf project deploy start --metadata 'ApexClass:MyApex*'
```

Deploy all custom objects and Apex classes (both examples are equivalent):

```
sf project deploy start --metadata CustomObject ApexClass
```

```
sf project deploy start --metadata CustomObject --metadata ApexClass
```

Deploy all Apex classes and a profile that has a space in its name:

```
sf project deploy start --metadata ApexClass --metadata "Profile:My Profile"
```

Deploy all components listed in a manifest:

```
sf project deploy start --manifest path/to/package.xml
```

Run the tests that aren't in any managed packages as part of a deployment:

```
sf project deploy start --metadata ApexClass --test-level RunLocalTests
```

Usage

```
sf project deploy start
  [--json]
  [-a API-VERSION]
  [--async]
  [--concise]
  [--dry-run]
  [-c]
  [-r]
  [-g]
  [-x MANIFEST]
  [-m METADATA]
  [--metadata-dir METADATA-DIR]
  [--single-package]
  [-d SOURCE-DIR]
  -o TARGET-ORG
  [-t TESTS]
  [-l TEST-LEVEL]
  [--verbose]
  [-w WAIT]
  [--purge-on-delete]
  [--pre-destructive-changes PRE-DESTRUCTIVE-CHANGES]
  [--post-destructive-changes POST-DESTRUCTIVE-CHANGES]
  [--coverage-formatters COVERAGE-FORMATTERS]
  [--junit]
  [--results-dir RESULTS-DIR]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-a | --api-version API-VERSION`

Optional

Target API version for the deploy.

Use this flag to override the default API version with the API version of your package.xml file. The default API version is the latest version supported by the CLI.

Type: option

`--async`
Optional

Run the command asynchronously.

The command immediately returns the job ID and control of the terminal to you. This way, you can continue to use the CLI. To resume the deployment, run "sf project deploy resume". To check the status of the deployment, run "sf project deploy report".

Type: boolean

`--concise`
Optional

Show concise output of the deploy result.

Type: boolean

`--dry-run`
Optional

Validate deploy and run Apex tests but don’t save to the org.

Type: boolean

`-c | --ignore-conflicts`
Optional

Ignore conflicts and deploy local files, even if they overwrite changes in the org.

This flag applies only to orgs that allow source tracking. It has no effect on orgs that don't allow it, such as production orgs.

Type: boolean

`-r | --ignore-errors`
Optional

Ignore any errors and don’t roll back deployment.

When deploying to a production org, keep this flag set to false (default value). When set to true, components without errors are deployed and components with errors are skipped, and could result in an inconsistent production org.

Type: boolean

`-g | --ignore-warnings`
Optional

Ignore warnings and allow a deployment to complete successfully.

If a warning occurs and this flag is set to true, the success status of the deployment is set to true. When this flag is set to false, success is set to false, and the warning is treated like an error.

Type: boolean

`-x | --manifest MANIFEST`
Optional

Full file path for manifest (package.xml) of components to deploy.

All child components are included. If you specify this flag, don’t specify --metadata or --source-dir.

Type: option

`-m | --metadata METADATA`
Optional

Metadata component names to deploy. Wildcards (*) supported as long as you use quotes, such as 'ApexClass:MyClass*'

Type: option

`--metadata-dir METADATA-DIR`

Optional

Root of directory or zip file of metadata formatted files to deploy.

Type: option

`--single-package`

Optional

Indicates that the metadata zip file points to a directory structure for a single package.

Type: boolean

`-d | --source-dir SOURCE-DIR`

Optional

Path to the local source files to deploy.

The supplied path can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all metadata types in the directory and its subdirectories).

If you specify this flag, don't specify --metadata or --manifest.

Type: option

`-o | --target-org TARGET-ORG`

Required

Login username or alias for the target org.

Overrides your default org.

Type: option

`-t | --tests TESTS`

Optional

Apex tests to run when --test-level is RunSpecifiedTests.

If a test name contains a space, enclose it in double quotes.

For multiple test names, use one of the following formats:

- Repeat the flag for multiple test names: --tests Test1 --tests Test2 --tests "Test With Space"

- Separate the test names with spaces: --tests Test1 Test2 "Test With Space"

Type: option

`-l | --test-level TEST-LEVEL`

Optional

Deployment Apex testing level.

Valid values are:

- NoTestRun — No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial orgs. This test level is the default for development environments.

- RunSpecifiedTests — Runs only the tests that you specify with the --tests flag. Code coverage requirements differ from the default coverage requirements when using this test level. Executed tests must comprise a minimum of 75% code coverage for each class and trigger in the deployment package. This coverage is computed for each class and trigger individually and is different than the overall coverage percentage.

- RunLocalTests — All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default for production deployments that include Apex classes or triggers.

- RunAllTestsInOrg — All tests in your org are run, including tests of managed packages.

If you don't specify a test level, the default behavior depends on the contents of your deployment package and target org. For more information, see [Running Tests in a Deployment](https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/meta_deploy_running_tests.htm) in the "Metadata API Developer Guide".

Type: option

Permissible values are: NoTestRun, RunSpecifiedTests, RunLocalTests, RunAllTestsInOrg

`--verbose`

Optional

Show verbose output of the deploy result.

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for command to complete and display results.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you and returns the job ID. To resume the deployment, run "sf project deploy resume". To check the status of the deployment, run "sf project deploy report".

Type: option

Default value: 33 minutes

`--purge-on-delete`

Optional

Specify that deleted components in the destructive changes manifest file are immediately eligible for deletion rather than being stored in the Recycle Bin.

Type: boolean

`--pre-destructive-changes PRE-DESTRUCTIVE-CHANGES`

Optional

File path for a manifest (destructiveChangesPre.xml) of components to delete before the deploy

Type: option

`--post-destructive-changes POST-DESTRUCTIVE-CHANGES`

Optional

File path for a manifest (destructiveChangesPost.xml) of components to delete after the deploy.

Type: option

`--coverage-formatters COVERAGE-FORMATTERS`

Optional

Format of the code coverage results.

For multiple formatters, repeat the flag for each formatter.

--coverage-formatters lcov --coverage-formatters clover

Type: option

Permissible values are: clover, cobertura, html-spa, html, json, json-summary, lcovonly, none, teamcity, text, text-summary

`--junit`
Optional

Output JUnit test results.

Type: boolean

`--results-dir RESULTS-DIR`
Optional

Output directory for code coverage and JUnit results; defaults to the deploy ID.

Type: option

Aliases for `project deploy start`

```
deploy:metadata
```

`project deploy validate`

Validate a metadata deployment without actually executing it.

Description for `project deploy validate`

Use this command to verify whether a deployment will succeed without actually deploying the metadata to your org. This command is similar to "sf project deploy start", except you're required to run Apex tests, and the command returns a job ID rather than executing the deployment. If the validation succeeds, then you pass this job ID to the "sf project deploy quick" command to actually deploy the metadata. This quick deploy takes less time because it skips running Apex tests. The job ID is valid for 10 days from when you started the validation. Validating first is useful if the deployment to your production org take several hours and you don't want to risk a failed deploy.

You must run this command from within a project.

This command doesn't support source-tracking. When you quick deploy with the resulting job ID, the source you deploy overwrites the corresponding metadata in your org.

To validate the deployment of multiple metadata components, either set multiple --metadata <name> flags or a single --metadata flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --manifest and --source-dir.

Examples for `project deploy validate`

NOTE: These examples focus on validating large deployments. See the help for "sf project deploy start" for examples of deploying smaller sets of metadata which you can also use to validate.

Validate the deployment of all source files in a directory to the default org:

```
sf project deploy validate --source-dir path/to/source
```

Asynchronously validate the deployment and run all tests in the org with alias "my-prod-org"; command immediately returns the job ID:

```
sf project deploy validate --source-dir path/to/source --async --test-level RunAllTestsInOrg
 --target-org my-prod-org
```

Validate the deployment of all components listed in a manifest:

```
sf project deploy validate --manifest path/to/package.xml
```

Usage

`sf project deploy validate`

```
[--json]
[-a API-VERSION]
[--async]
[--concise]
[-x MANIFEST]
[-m METADATA]
[-d SOURCE-DIR]
[--metadata-dir METADATA-DIR]
[--single-package]
-o TARGET-ORG
[-t TESTS]
[-l TEST-LEVEL]
[--verbose]
[-w WAIT]
[--coverage-formatters COVERAGE-FORMATTERS]
[--junit]
[--results-dir RESULTS-DIR]
[--purge-on-delete]
[--pre-destructive-changes PRE-DESTRUCTIVE-CHANGES]
[--post-destructive-changes POST-DESTRUCTIVE-CHANGES]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-a | --api-version API-VERSION`

Optional

Target API version for the validation.

Use this flag to override the default API version with the API version of your package.xml file. The default API version is the latest version supported by the CLI.

Type: option

`--async`

Optional

Run the command asynchronously.

The command immediately returns the job ID and control of the terminal to you. This way, you can continue to use the CLI. To resume watching the validation, run "sf project deploy resume". To check the status of the validation, run "sf project deploy report".

Type: boolean

`--concise`

Optional

Show concise output of the validation result.

Type: boolean

`-x | --manifest MANIFEST`

Optional

Full file path for manifest (package.xml) of components to validate for deployment.

All child components are included. If you specify this flag, don't specify --metadata or --source-dir.

Type: option

`-m | --metadata METADATA`

Optional

Metadata component names to validate for deployment.

Type: option

`-d | --source-dir SOURCE-DIR`

Optional

Path to the local source files to validate for deployment.

The supplied path can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all metadata types in the directory and its subdirectories).

If you specify this flag, don't specify --metadata or --manifest.

Type: option

`--metadata-dir METADATA-DIR`

Optional

Root of directory or zip file of metadata formatted files to deploy.

Type: option

`--single-package`

Optional

Indicates that the metadata zip file points to a directory structure for a single package.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Login username or alias for the target org.

Overrides your default org.

Type: option

`-t | --tests TESTS`

Optional

Apex tests to run when --test-level is RunSpecifiedTests.

If a test name contains a space, enclose it in double quotes.

For multiple test names, use one of the following formats:

- Repeat the flag for multiple test names: --tests Test1 --tests Test2 --tests "Test With Space"

- Separate the test names with spaces: --tests Test1 Test2 "Test With Space"

Type: option

`-l | --test-level TEST-LEVEL`

Optional

Deployment Apex testing level.

Valid values are:

- RunSpecifiedTests — Runs only the tests that you specify with the --tests flag. Code coverage requirements differ from the default coverage requirements when using this test level. Executed tests must comprise a minimum of 75% code coverage for each class and trigger in the deployment package. This coverage is computed for each class and trigger individually and is different than the overall coverage percentage.

- RunLocalTests — All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default.

- RunAllTestsInOrg — All tests in your org are run, including tests of managed packages.

Type: option

Permissible values are: RunAllTestsInOrg, RunLocalTests, RunSpecifiedTests

Default value: RunLocalTests

`--verbose`

Optional

Show verbose output of the validation result.

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for the command to complete and display results.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you and returns the job ID. To resume watching the validation, run "sf project deploy resume". To check the status of the validation, run "sf project deploy report".

Type: option

Default value: 33 minutes

`--coverage-formatters COVERAGE-FORMATTERS`

Optional

Format of the code coverage results.

For multiple formatters, repeat the flag for each formatter.

--coverage-formatters lcov --coverage-formatters clover

Type: option

Permissible values are: clover, cobertura, html-spa, html, json, json-summary, lcovonly, none, teamcity, text, text-summary

`--junit`

Optional

Output JUnit test results.

Type: boolean

`--results-dir RESULTS-DIR`

Optional

Output directory for code coverage and JUnit results; defaults to the deploy ID.

Type: option

`--purge-on-delete`

Optional

Specify that deleted components in the destructive changes manifest file are immediately eligible for deletion rather than being stored in the Recycle Bin.

Type: boolean

`--pre-destructive-changes PRE-DESTRUCTIVE-CHANGES`

Optional

File path for a manifest (destructiveChangesPre.xml) of components to delete before the deploy

Type: option

`--post-destructive-changes POST-DESTRUCTIVE-CHANGES`

Optional

File path for a manifest (destructiveChangesPost.xml) of components to delete after the deploy.

Type: option

Aliases for `project deploy validate`

```
deploy:metadata:validate
```

`project generate`

Generate a Salesforce DX project.

Description for `project generate`

A Salesforce DX project has a specific structure and a configuration file (sfdx-project.json) that identifies the directory as a Salesforce DX project. This command generates the necessary configuration files and directories to get you started.

By default, the generated sfdx-project.json file sets the sourceApiVersion property to the default API version currently used by Salesforce CLI. To specify a different version, set the apiVersion configuration variable. For example: "sf config set apiVersion=57.0 --global".

Examples for `project generate`

Generate a project called "mywork":

```
sf project generate --name mywork
```

Similar to previous example, but generate the files in a directory called "myapp":

```
sf project generate --name mywork --default-package-dir myapp
```

Similar to prevoius example, but also generate a default package.xml manifest file:

```
sf project generate --name mywork --default-package-dir myapp --manifest
```

Generate a project with the minimum files and directories:

```
sf project generate --name mywork --template empty
```

Usage

`sf project generate`

```
[--json]
-n NAME
[-t TEMPLATE]
[-d OUTPUT-DIR]
[-s NAMESPACE]
[-p DEFAULT-PACKAGE-DIR]
[-x]
[--api-version API-VERSION]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated project.

Generates a project directory with this name; any valid directory name is accepted. Also sets the "name" property in the sfdx-project.json file to this name.

Type: option

`-t | --template TEMPLATE`

Optional

Template to use for project creation.

The template determines the sample configuration files and directories that this command generates. For example, the empty template provides these files and directory to get you started.

- .forceignore

- config/project-scratch-def.json

- sfdx-project.json

- package.json

- force-app (basic source directory structure)

The standard template provides a complete force-app directory structure so you know where to put your source. It also provides additional files and scripts, especially useful when using Salesforce Extensions for VS Code. For example:

- .gitignore: Use Git for version control.

- .prettierrc and .prettierignore: Use Prettier to format your Aura components.

- .vscode/extensions.json: When launched, Visual Studio Code, prompts you to install the recommended extensions for your project.

- .vscode/launch.json: Configures Replay Debugger.

- .vscode/settings.json: Additional configuration settings.

The analytics template provides similar files and the force-app/main/default/waveTemplates directory.

Type: option

Permissible values are: standard, empty, analytics

Default value: standard

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`-s | --namespace NAMESPACE`

Optional

Namespace associated with this project and any connected scratch orgs.

Type: option

`-p | --default-package-dir DEFAULT-PACKAGE-DIR`

Optional

Default package directory name.

Metadata items such as classes and Lightning bundles are placed inside this folder.

Type: option

Default value: force-app

`-x | --manifest`

Optional

Generate a manifest (package.xml) for change-set based development.

Generates a default manifest (package.xml) for fetching Apex, Visualforce, Lightning components, and static resources.

Type: boolean

`--api-version API-VERSION`

Optional

Will set this version as sourceApiVersion in the sfdx-project.json file

Override the api version used for api requests made by this command

Type: option

Aliases for `project generate`

```
force:project:create
```

`project generate manifest`

Create a project manifest that lists the metadata components you want to deploy or retrieve.

Description for `project generate manifest`

Create a manifest from a list of metadata components (--metadata) or from one or more local directories that contain source files (--source-dir). You can specify either of these parameters, not both.

Use --type to specify the type of manifest you want to create. The resulting manifest files have specific names, such as the standard package.xml or destructiveChanges.xml to delete metadata. Valid values for this parameter, and their respective file names, are:

* package : package.xml (default)

* pre : destructiveChangesPre.xml

* post : destructiveChangesPost.xml

* destroy : destructiveChanges.xml

See https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/meta_deploy_deleting_files.htm for information about these destructive manifest files.

Use --name to specify a custom name for the generated manifest if the pre-defined ones don't suit your needs. You can specify either --type or --name, but not both.

To include multiple metadata components, either set multiple --metadata <name> flags or a single --metadata flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --include-packages and --source-dir.

Examples for `project generate manifest`

Create a manifest for deploying or retrieving all Apex classes and custom objects:

```
$ sf project generate manifest --metadata ApexClass --metadata CustomObject
```

Create a manifest for deleting the specified Apex class:

```
$ sf project generate manifest --metadata ApexClass:MyApexClass --type destroy
```

Create a manifest for deploying or retrieving all the metadata components in the specified local directory; name the file myNewManifest.xml:

```
$ sf project generate manifest --source-dir force-app --name myNewManifest
```

Create a manifest from the metadata components in the specified org and include metadata in any unlocked packages:

```
$ sf project generate manifest --from-org test@myorg.com --include-packages unlocked
```

Usage

```
sf project generate manifest
   [--json]
   [--api-version API-VERSION]
   [-m METADATA]
   [-p SOURCE-DIR]
   [-n NAME]
   [-t TYPE]
   [-c INCLUDE-PACKAGES]
   [--from-org FROM-ORG]
   [-d OUTPUT-DIR]
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-m | --metadata METADATA`

Optional

Names of metadata components to include in the manifest.

Type: option

`-p | --source-dir SOURCE-DIR`

Optional

Paths to the local source files to include in the manifest.

Type: option

`-n | --name NAME`

Optional

Name of a custom manifest file to create.

Type: option

`-t | --type TYPE`

Optional

Type of manifest to create; the type determines the name of the created file.

Type: option

Permissible values are: pre, post, destroy, package

-c | --include-packages INCLUDE-PACKAGES

Optional

Package types (managed, unlocked) whose metadata is included in the manifest; by default, metadata in packages is ignored.

Type: option

Permissible values are: managed, unlocked

--from-org FROM-ORG

Optional

Username or alias of the org that contains the metadata components from which to build a manifest.

Type: option

-d | --output-dir OUTPUT-DIR

Optional

Directory to save the created manifest.

Type: option

Aliases for `project generate manifest`

```
force:source:manifest:create
```

`project list ignored`

Check your local project package directories for forceignored files.

Description for `project list ignored`

When deploying or retrieving metadata between your local project and an org, you can specify the source files you want to exclude with a .forceignore file. The .forceignore file structure mimics the .gitignore structure. Each line in .forceignore specifies a pattern that corresponds to one or more files. The files typically represent metadata components, but can be any files you want to exclude, such as LWC configuration JSON files or tests.

Examples for `project list ignored`

List all the files in all package directories that are ignored:

```
sf project list ignored
```

List all the files in a specific directory that are ignored:

```
sf project list ignored --source-dir force-app
```

Check if a particular file is ignored:

```
sf project list ignored --source-dir package.xml
```

Usage

sf project list ignored

[--json]

[-p SOURCE-DIR]

Flags

--json
Optional

Format output as json.

Type: boolean

-p | --source-dir SOURCE-DIR
Optional

File or directory of files that the command checks for foreceignored files.

Type: option

Aliases for `project list ignored`

```
force:source:ignored:list
```

`project reset tracking`

Reset local and remote source tracking.

Description for `project reset tracking`

WARNING: This command deletes or overwrites all existing source tracking files. Use with extreme caution.

Resets local and remote source tracking so that Salesforce CLI no longer registers differences between your local files and those in the org. When you next run 'project deploy preview', Salesforce CLI returns no results, even though conflicts might actually exist. Salesforce CLI then resumes tracking new source changes as usual.

Use the --revision parameter to reset source tracking to a specific revision number of an org source member. To get the revision number, query the SourceMember Tooling API object with the 'data soql' command. For example:

```
sf data query --query "SELECT MemberName, MemberType, RevisionCounter FROM SourceMember"
--use-tooling-api
```

Usage

sf project reset tracking

[--json]

-o TARGET-ORG

[--api-version API-VERSION]

[-r REVISION]

[-p]

Flags

`--json`
Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`
Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`
Optional

Override the api version used for api requests made by this command

Type: option

`-r | --revision REVISION`
Optional

SourceMember revision counter number to reset to.

Type: option

`-p | --no-prompt`
Optional

Don't prompt for source tracking override confirmation.

Type: boolean

Aliases for `project reset tracking`

```
force:source:tracking:reset
```

`project retrieve preview`

Preview a retrieval to see what will be retrieved from the org, the potential conflicts, and the ignored files.

Description for `project retrieve preview`

You must run this command from within a project.

The command outputs a table that describes what will happen if you run the "sf project retrieve start" command. The table lists the metadata components that will be retrieved and deleted. The table also lists the current conflicts between files in your local project and components in the org. Finally, the table lists the files that won't be retrieved because they're included in your .forceignore file.

If your org allows source tracking, then this command displays potential conflicts between the org and your local project. Some orgs, such as production org, never allow source tracking. Source tracking is enabled by default on scratch and sandbox orgs; you can disable source tracking when you create the orgs by specifying the --no-track-source flag on the "sf org create scratch|sandbox" commands.

Examples for `project retrieve preview`

Preview the retrieve of all changes from your default org:

```
sf project retrieve preview
```

Preview the retrieve when ignoring any conflicts from an org with alias "my-scratch":

```
sf project retrieve preview --ignore-conflicts --target-org my-scratch
```

Usage

`sf project retrieve preview`

`[--json]`

`[-c]`

`-o TARGET-ORG`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-c | --ignore-conflicts`

Optional

Don't display conflicts in the preview of the retrieval.

This flag applies only to orgs that allow source tracking. It has no effect on orgs that don't allow it, such as production orgs.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Login username or alias for the target org.

Overrides your default org.

Type: option

Aliases for `project retrieve preview`

```
retrieve:metadata:preview
```

`project retrieve start`

Retrieve metadata from an org to your local project.

Description for `project retrieve start`

You must run this command from within a project.

Metadata components are retrieved in source format by default. Retrieve them in metadata format by specifying the --target-metadata-dir flag, which retrieves the components into a ZIP file in the specified directory.

If your org allows source tracking, then this command tracks the changes in your source. Some orgs, such as production org, never allow source tracking. Source tracking is enabled by default on scratch and sandbox orgs; you can disable source tracking when you create the orgs by specifying the --no-track-source flag on the "sf org create scratch|sandbox" commands.

To retrieve multiple metadata components, either use multiple --metadata <name> flags or use a single --metadata flag with multiple names separated by spaces. Enclose names that contain spaces in one set of double quotes. The same syntax applies to --manifest and --source-dir.

Examples for `project retrieve start`

Retrieve remote changes from your default org:

```
sf project retrieve start
```

Retrieve the source files in a directory from an org with alias "my-scratch":

```
sf project retrieve start --source-dir path/to/source --target-org my-scratch
```

Retrieve a specific Apex class and the objects whose source is in a directory (both examples are equivalent):

```
sf project retrieve start --source-dir path/to/apex/classes/MyClass.cls
path/to/source/objects
```

```
sf project retrieve start --source-dir path/to/apex/classes/MyClass.cls --source-dir
path/to/source/objects
```

Retrieve all Apex classes:

```
sf project retrieve start --metadata ApexClass
```

Retrieve a specific Apex class:

```
sf project retrieve start --metadata ApexClass:MyApexClass
```

Retrieve specific Apex classes that match a pattern; in this example, retrieve Apex classes whose names contain the string "MyApex":

```
sf project retrieve start --metadata 'ApexClass:MyApex*'
```

Retrieve all custom objects and Apex classes (both examples are equivalent):

```
sf project retrieve start --metadata CustomObject ApexClass
```

```
sf project retrieve start --metadata CustomObject --metadata ApexClass
```

Retrieve all metadata components listed in a manifest:

```
sf project retrieve start --manifest path/to/package.xml
```

Retrieve metadata from a package:

```
sf project retrieve start --package-name MyPackageName
```

Retrieve metadata from multiple packages, one of which has a space in its name (both examples are equivalent):

```
sf project retrieve start --package-name Package1 "PackageName With Spaces" Package3
```

```
sf project retrieve start --package-name Package1 --package-name "PackageName With Spaces"
 --package-name Package3
```

Retrieve the metadata components listed in the force-app directory, but retrieve them in metadata format into a ZIP file in the "output" directory:

```
sf project retrieve start --source-dir force-app --target-metadata-dir output
```

Retrieve in metadata format and automatically extract the contents into the "output" directory:

```
sf project retrieve start --source-dir force-app --target-metadata-dir output --unzip
```

Usage

`sf project retrieve start`

`[--json]`

`[-a API-VERSION]`

`[-c]`

`[-x MANIFEST]`

`[-m METADATA]`

`[-n PACKAGE-NAME]`

`[-r OUTPUT-DIR]`

`[--single-package]`

`[-d SOURCE-DIR]`

`[-t TARGET-METADATA-DIR]`

`-o TARGET-ORG`

`[-w WAIT]`

`[-z]`

`[--zip-file-name ZIP-FILE-NAME]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-a | --api-version API-VERSION`

Optional

Target API version for the retrieve.

Use this flag to override the default API version, which is the latest version supported the CLI, with the API version in your package.xml file.

Type: option

-c | --ignore-conflicts

Optional

Ignore conflicts and retrieve and save files to your local filesystem, even if they overwrite your local changes.

This flag applies only to orgs that allow source tracking. It has no effect on orgs that don't allow it, such as production orgs.

Type: boolean

-x | --manifest MANIFEST

Optional

File path for the manifest (package.xml) that specifies the components to retrieve.

If you specify this parameter, don't specify --metadata or --source-dir.

Type: option

-m | --metadata METADATA

Optional

Metadata component names to retrieve. Wildcards (*) supported as long as you use quotes, such as 'ApexClass:MyClass*'

Type: option

-n | --package-name PACKAGE-NAME

Optional

Package names to retrieve.

Type: option

-r | --output-dir OUTPUT-DIR

Optional

Directory root for the retrieved source files.

The root of the directory structure into which the source files are retrieved.

If the target directory matches one of the package directories in your sfdx-project.json file, the command fails.

Running the command multiple times with the same target adds new files and overwrites existing files.

Type: option

--single-package

Optional

Indicates that the zip file points to a directory structure for a single package.

Type: boolean

-d | --source-dir SOURCE-DIR

Optional

File paths for source to retrieve from the org.

The supplied paths can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all source files in the directory and its subdirectories).

Type: option

-t | --target-metadata-dir TARGET-METADATA-DIR

Optional

Directory that will contain the retrieved metadata format files or ZIP.

Type: option

`-o | --target-org TARGET-ORG`
Required

Login username or alias for the target org.

Overrides your default org.

Type: option

`-w | --wait WAIT`
Optional

Number of minutes to wait for the command to complete and display results to the terminal window.

If the command continues to run after the wait period, the CLI returns control of the terminal window to you.

Type: option

Default value: 33 minutes

`-z | --unzip`
Optional

Extract all files from the retrieved zip file.

Type: boolean

`--zip-file-name ZIP-FILE-NAME`
Optional

File name to use for the retrieved zip file.

Type: option

Aliases for `project retrieve start`

```
retrieve:metadata
```

`run` Commands

Commands to run a function.

run function
Send a cloudevent to a function.

run function start
Build and run a Salesforce Function.

run function start container

run function start local

`run function`

Send a cloudevent to a function.

Examples for `run function`

Run a function:

```
sf run function --url http://path/to/function
```

Run a function with a payload and a JSON response:

```
sf run function --url http://path/to/function --payload '@file.json' --structured
```

Usage

`sf run function`

`[--json]`

`[-l FUNCTION-URL]`

`[-H HEADERS]`

`[-p PAYLOAD]`

`[-s]`

`[-o CONNECTED-ORG]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-l | --function-url FUNCTION-URL`

Optional

URL of the function to run.

Type: option

`-H | --headers HEADERS`

Optional

Set headers.

Type: option

`-p | --payload PAYLOAD`

Optional

Set the payload of the cloudevent as a JSON object or a path to a file via @file.json.

Type: option

`-s | --structured`

Optional

Set the cloudevent to be emitted as a structured JSON cloudevent.

Type: boolean

`-o | --connected-org CONNECTED-ORG`

Optional

Username or alias for the target org; overrides default target org.

Type: option

`run function start`

Build and run a Salesforce Function.

Description for `run function start`

Run this command from the directory of your Salesforce Functions project.

This command will run the target function locally (on the same operating system as this CLI), just like the `local` subcommand.

Examples for `run function start`

Build a function and start the invoker

```
sf run function start
```

Start the invoker with a specific language and port

```
sf run function start --port 5000 --language javascript
```

Usage

`sf run function start`

```
[-p PORT]
[-b DEBUG-PORT]
[-l LANGUAGE]
[-v]
```

Flags

`-p | --port PORT`

Optional

Port for running the function.

Type: option

Default value: 8080

`-b | --debug-port DEBUG-PORT`

Optional

Port for remote debugging.

Type: option

Default value: 9229

`-l | --language LANGUAGE`

Optional

The language that the function runs in.

Type: option

Permissible values are: auto, java, javascript, python, typescript

Default value: auto

-v | --verbose

Optional

Output additional logs.

Type: boolean

`run function start container`

Usage

`sf run function start container`

`run function start local`

Description for `run function start local`

Build and run a Salesforce Function locally.

Examples for `run function start local`

Build a function and start the invoker

```
sf run function start local
```

Start the invoker with a specific language and port

```
sf run function start local --port 5000 --language javascript
```

Usage

`sf run function start local`

[`-p PORT`]

[`-b DEBUG-PORT`]

[`-l LANGUAGE`]

Flags

-p | --port PORT

Optional

Port to bind the invoker to.

Type: option

Default value: 8080

`-b | --debug-port DEBUG-PORT`

Optional

Port to use for debugging the function.

Type: option

Default value: 9229

`-l | --language LANGUAGE`

Optional

The language in which the function is written.

Type: option

Permissible values are: auto, java, javascript, python, typescript

Default value: auto

`schema` Commands

Generate metadata files.

schema generate field
Generate metadata source files for a new custom field on a specified object.

schema generate platformevent
Generate metadata source files for a new platform event.

schema generate sobject
Generate metadata source files for a new custom object.

schema generate tab
Generate the metadata source files for a new custom tab on a custom object.

`schema generate field`

Generate metadata source files for a new custom field on a specified object.

Description for `schema generate field`

This command is interactive and must be run in a Salesforce DX project directory. You're required to specify the field's label with the "--label" flag. The command uses this label to provide intelligent suggestions for other field properties, such as its API name.

You can generate a custom field on either a standard object, such as Account, or a custom object. In both cases, the source files for the object must already exist in your local project before you run this command. If you create a relationship field, the source files for the parent object must also exist in your local directory. Use the command "sf metadata retrieve -m CustomObject:<object>" to retrieve source files for both standard and custom objects from your org. To create a custom object, run the "sf generate metadata sobject" command or use the Object Manager UI in your Salesforce org.

Examples for `schema generate field`

Create a field with the specified label; the command prompts you for the object:

```
sf schema generate field --label "My Field"
```

Specify the local path to the object's folder:

```
sf schema generate field --label "My Field" --object
force-app/main/default/objects/MyObject__c
```

Usage

`sf schema generate field`

`[--json]`

`-l LABEL`

`[-o OBJECT]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-l | --label LABEL`

Required

The field's label.

Type: option

`-o | --object OBJECT`

Optional

The directory that contains the object's source files.

The object source files in your local project are grouped in a directoy with the same name as the object. Custom object names always end in "__c". An example of the object directory for the Account standard object is "force-app/main/default/objects/Account" An example custom object directory is "force-app/main/default/objects/MyObject__c"

If you don't specify this flag, the command prompts you to choose from your local objects.

Type: option

Aliases for `schema generate field`

```
generate:metadata:field
```

`schema generate platformevent`

Generate metadata source files for a new platform event.

Description for `schema generate platformevent`

This command is interactive and must be run in a Salesforce DX project directory. You're required to specify the event's label with the "--label" flag. The command uses this label to provide intelligent suggestions for other event properties, such as its API name.

Examples for `schema generate platformevent`

Create a platform event with the specified label:

```
sf schema generate platformevent --label "My Platform Event"
```

Usage

`sf schema generate platformevent`

`[--json]`

`-l LABEL`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-l | --label LABEL`

Required

The platform event's label.

Type: option

Aliases for `schema generate platformevent`

```
generate:metadata:platformevent
```

`schema generate sobject`

Generate metadata source files for a new custom object.

Description for `schema generate sobject`

This command is interactive and must be run in a Salesforce DX project directory. You're required to specify the object's label with the "--label" flag. The command uses this label to provide intelligent suggestions for other object properties, such as its API name and plural label.

All Salesforce objects are required to have a Name field, so this command also prompts you for the label and type of the Name field. Run the "sf metadata generate field" command to create additional fields for the object.

To reduce the number of prompts, use the "--use-default-features" flag to automatically enable some features, such as reporting and search on the object.

Examples for `schema generate sobject`

Create a custom object with the specified label and be prompted for additional information:

```
sf schema generate sobject --label "My Object"
```

Create a custom object and enable optional features without prompting:

```
sf schema generate sobject --label "My Object" --use-default-features
```

Usage

`sf schema generate sobject`

`[--json]`

`-l LABEL`

`[-f]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-l` | `--label LABEL`

Required

The custom object's label.

Type: option

`-f` | `--use-default-features`

Optional

Enable all optional features without prompting.

Enables these features:

* Search: Allows users to find the custom object's records when they search, including SOSL.

* Feeds: Enables feed tracking.

* Reports: Allows reporting of the data in the custom object records.

* History: Enables object history tracking.

* Activities: Allows users to associate tasks and scheduled calendar events related to the custom object records.

* Bulk API: With Sharing and Streaming API, classifies the custom object as an Enterprise Application object.

* Sharing: With Bulk API and Streaming API, classifies the custom object as an Enterprise Application object.

* Streaming API: With Bulk API and Sharing, classifies the custom object as an Enterprise Application object.

Type: boolean

Aliases for `schema generate sobject`

```
generate:metadata:sobject
```

`schema generate tab`

Generate the metadata source files for a new custom tab on a custom object.

Description for `schema generate tab`

Custom tabs let you display custom object data or other web content in Salesforce. Custom tabs appear in Salesforce as an item in the app's navigation bar and in the App Launcher.

This command must be run in a Salesforce DX project directory. You must pass all required information to it with the required flags. The source files for the custom object for which you're generating a tab don't need to exist in your local project.

Examples for `schema generate tab`

Create a tab on the MyObject__c custom object:

```
sf schema generate tab --object MyObject__c --icon 54 --directory force-app/main/default/tabs
```

Usage

`sf schema generate tab`

```
[--json]
-o OBJECT
-d DIRECTORY
-i ICON
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --object OBJECT`

Required

API name of the custom object you're generating a tab for.

The API name for a custom object always ends in "__c", such as "MyObject__c".

Type: option

`-d | --directory DIRECTORY`

Required

Path to a "tabs" directory that will contain the source files for your new tab.

Type: option

`-i | --icon ICON`

Required

Number from 1 to 100 that specifies the color scheme and icon for the custom tab.

See https://lightningdesignsystem.com/icons/\#custom for the available icons.

Type: option

Default value: 1

Aliases for `schema generate tab`

```
generate:metadata:tab
```

`sobject` Commands

Commands to interact with Salesforce objects.

sobject describe
Display the metadata for a standard or custom object or a Tooling API object.

sobject list
List all Salesforce objects of a specified category.

`sobject describe`

Display the metadata for a standard or custom object or a Tooling API object.

Description for `sobject describe`

The metadata is displayed in JSON format. See this topic for a description of each property: https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/sforce_api_calls_describesobjects_describesobjectresult.htm.

This command displays metadata for Salesforce objects by default. Use the --use-tooling-api flag to view metadata for a Tooling API object.

Examples for `sobject describe`

Display the metadata of the "Account" standard object in your default org:

```
sf sobject describe --sobject Account
```

Display the metadata of the "MyObject__c" custom object in the org with alias "my-scratch-org":

```
sf sobject describe --sobject MyObject__c --target-org my-scratch-org
```

Display the metadata of the ApexCodeCoverage Tooling API object in your default org:

```
sf sobject describe --sobject ApexCodeCoverage --use-tooling-api
```

Usage

`sf sobject describe`

```
[--json]
-o TARGET-ORG
[--api-version API-VERSION]
-s SOBJECT
[-t]
```

Flags

--json
: Optional

  Format output as json.

  Type: boolean

-o | --target-org TARGET-ORG
: Required

  Username or alias of the target org.

  Type: option

--api-version API-VERSION
: Optional

  Override the api version used for api requests made by this command

  Type: option

-s | --sobject SOBJECT
: Required

  API name of the object to describe.

  Type: option

-t | --use-tooling-api
: Optional

  Use Tooling API to display metadata for Tooling API objects.

  Type: boolean

Aliases for `sobject describe`

```
force:schema:sobject:describe
```

`sobject list`

List all Salesforce objects of a specified category.

Description for `sobject list`

You can list the standard objects, custom objects, or all. The lists include only Salesforce objects, not Tooling API objects.

Examples for `sobject list`

List all objects in your default org:

```
sf sobject list --sobject all
```

List only custom objects in the org with alias "my-scratch-org":

```
sf sobject list --sobject custom --target-org my-scratch-org
```

Usage

`sf object list`

`[--json]`

`-o TARGET-ORG`

`[--api-version API-VERSION]`

`[-s SOBJECT]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-o | --target-org TARGET-ORG`

Required

Username or alias of the target org.

Type: option

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-s | --sobject SOBJECT`

Optional

Category of objects to list.

Type: option

Default value: ALL

Aliases for `sobject list`

```
force:schema:sobject:list
```

`static-resource` Commands

Work with static resources.

static-resource generate

Generate a static resource.

`static-resource generate`

Generate a static resource.

Description for `static-resource generate`

Generates the metadata resource file in the specified directory or the current working directory. Static resource files must be contained in a parent directory called "staticresources" in your package directory. Either run this command from an existing directory of this name, or use the --output-dir flag to create one or point to an existing one.

Examples for `static-resource generate`

Generate the metadata file for a static resource called MyResource in the current directory:

```
sf static-resource generate --name MyResource
```

Similar to previous example, but specifies a MIME type of application/json:

```
sf static-resource generate --name MyResource --type application/json
```

Generate the resource file in the "force-app/main/default/staticresources" directory:

```
sf static-resource generate --name MyResource --output-dir
force-app/main/default/staticresources
```

Usage

`sf static-resource generate`

`[--json]`

`-n NAME`

`[--type TYPE]`

`[-d OUTPUT-DIR]`

`[--api-version API-VERSION]`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated static resource.

This name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Type: option

`--type TYPE`

Optional

Content type (mime type) of the generated static resource.

Must be a valid MIME type such as application/json, application/javascript, application/zip, text/plain, text/css, etc.

Type: option

Default value: application/zip

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

Aliases for `static-resource generate`

```
force:staticresource:create
```

`visualforce` Commands

Work with Visualforce components.

visualforce generate component

Generate a Visualforce Component.

visualforce generate page

Generate a Visualforce Page.

`visualforce generate component`

Generate a Visualforce Component.

Description for `visualforce generate component`

The command generates the .Component file and associated metadata file in the specified directory or the current working directory by default.

Examples for `visualforce generate component`

Generate the metadata files for a Visualforce component in the current directory:

```
sf visualforce generate component --name mycomponent --label mylabel
```

Similar to previous example, but generate the files in the directory "force-app/main/default/components":

```
sf visualforce generate component --name mycomponent --label mylabel --output-dir components
```

Usage

`sf visualforce generate component`

```
[--json]

-n NAME

[-t TEMPLATE]

[-d OUTPUT-DIR]

[--api-version API-VERSION]

-l LABEL
```

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated Visualforce Component.

The name can be up to 40 characters and must start with a letter.

Type: option

`-t | --template TEMPLATE`

Optional

Template to use for file creation.

Supplied parameter values or default values are filled into a copy of the template.

Type: option

Permissible values are: DefaultVFComponent

Default value: DefaultVFComponent

`-d | --output-dir OUTPUT-DIR`

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

`--api-version API-VERSION`

Optional

Override the api version used for api requests made by this command

Type: option

`-l | --label LABEL`

Required

Visualforce Component label.

Type: option

Aliases for `visualforce generate component`

```
force:visualforce:component:create
```

`visualforce generate page`

Generate a Visualforce Page.

Description for `visualforce generate page`

The command generates the .Page file and associated metadata file in the specified directory or the current working directory by default.

Examples for `visualforce generate page`

Generate the metadata files for a Visualforce page in the current directory:

```
sf visualforce generate page --name mypage --label mylabel
```

Similar to previous example, but generate the files in the directory "force-app/main/default/pages":

```
sf visualforce generate page --name mypage --label mylabel --output-dir pages
```

Usage

`sf visualforce generate page`

`[--json]`

`-n NAME`

`[-d OUTPUT-DIR]`

`[--api-version API-VERSION]`

`-l LABEL`

Flags

`--json`

Optional

Format output as json.

Type: boolean

`-n | --name NAME`

Required

Name of the generated Visualforce Page.

The name can be up to 40 characters and must start with a letter.

Type: option

-d | --output-dir OUTPUT-DIR

Optional

Directory for saving the created files.

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: option

Default value: .

--api-version API-VERSION

Optional

Override the api version used for api requests made by this command

Type: option

-l | --label LABEL

Required

Visualforce Page label.

Type: option

Aliases for `visualforce generate page`

```
force:visualforce:page:create
```

`whoami` Commands

Commands to show information about yourself or your account.

whoami functions
Show information on your Salesforce Functions login.

`whoami functions`

Show information on your Salesforce Functions login.

Description for `whoami functions`

Returns your email and ID. Use '--show-token' to show your Salesforce Functions token.

Examples for `whoami functions`

Get account information:

```
sf whoami functions
```

Show token and output result as JSON:

```
sf whoami functions --show-token --json
```

Usage

`sf whoami functions`
`[--json]`

Flags

`--json`
Optional

Format output as json.

Type: boolean

Help for `sf` Commands

The `-h` and `--help` flags show details about `sf` topics and their commands.

The short `-h` flag shows a subset of the command-line help and is meant for quick reference. The long `--help` flag shows the complete command-line help.

The short help (`-h`) for commands has these parts.

1. **Short Description of Command**

   At the top of the `-h` output (with no heading), a short description of the command is shown.

2. **Usage**

   The command signature on the Usage line uses the docopt format.

   - All available flags are listed. Flags that have short names are listed using their short names.
   - Flags that take a value show `<value>` immediately after the flag's name.
   - Optional flags are in square brackets (`[` ... `]`).
   - Required flags have no annotation.
   - For flags that accept a limited set of values, the values are shown after the flag name, separated by pipes (`--flagname value1|value2|value3`).

3. **Flags**

   The Flags section lists all the command's flags, including their short name, long name, and purpose. Flags are grouped for easier reading, such as global flags and other groups relevant to a specific command.

   For flags that take multiple values, you have two ways to specify the values:

   - Specify the flag multiple times, where each flag takes a different single value.
   - Specify the flag one time, but separate all the values with a space.

   For example, the following commands are equivalent:

   ```
   sf deploy metadata --metadata ApexClass --metadata CustomObject --metadata
   AnotherCustomObject
   sf deploy metadata --metadata ApexClass CustomObject AnotherCustomObject
   ```

   Flags that accept a limited list of values include the values in parentheses, with the default value specified with the `default` keyword.

The long help (`--help`) for commands has the same parts as the preceding short help `-h`) and these additional parts.

1. **Description**

   Usage notes.

2. **Examples**

   All examples include a brief description.

3. **Flag Descriptions**

   Some flags have optional additional usage notes.

sfdx

This section contains information about the `sfdx`-style commands and their parameters.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

The `sfdx`-style commands continue to work the same as before, and any scripts that use the commands won't break. However, we recommend that you start using the equivalent `sf`-style commands on page 1.

The reference page for each `sfdx` command has been updated with information about its equivalent `sf` command. The page also shows how the flag names have changed between the `sfdx` and `sf` commands. Use this information to migrate your scripts.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf` commands. To view all the changes, read the CLI release notes starting with version 7.183.1 on January 12, 2023.

Migrate sfdx-Style Commands to the New sf-Style
When you're ready, we recommend that you start using the new `sf` commands in your continuous integration (CI) scripts and your day-to-day work as soon as possible. This section provides information on how to migrate. Migrating scripts are the focus, though this section also applies to running impromptu commands at a terminal. Remember, the `sfdx` commands, such as `force:org:create`, continue to work as before.

alias Namespace
Use the alias commands to manage username aliases.

auth Namespace
Use the auth commands to authorize a Salesforce org for use with the Salesforce CLI.

config Namespace
Use the config commands to view and set your Salesforce CLI configuration values. Set your default Dev Hub and scratch org, and your default instance URL, either globally or at the project level.

doctor
Gather CLI configuration data and run diagnostic tests to discover and report potential problems in your environment.

force Namespace
Commands to develop on the Salesforce Platform.

info Namespace
Access cli info from the command line.

Help for sfdx Commands

The `-h` | `--help` parameter shows details about `sfdx` topics and their commands.

Migrate **`sfdx`**-Style Commands to the New **`sf`**-Style

When you're ready, we recommend that you start using the new `sf` commands in your continuous integration (CI) scripts and your day-to-day work as soon as possible. This section provides information on how to migrate. Migrating scripts are the focus, though this section also applies to running impromptu commands at a terminal. Remember, the `sfdx` commands, such as `force:org:create`, continue to work as before.

Follow these high-level steps to migrate.

- Read an overview of the changes, including important usage differences. on page 275
- Run `dev convert script` to convert a large portion of your CI script. on page 276
- Manually update the remainder of your script. on page 276

Overview of Command and Usage Differences

The deprecated `sfdx` commands are different from the `sf` ones in these key ways. Other sections in this migration guide go into details.

- Some commands and flags have new names, but their behavior and JSON output is the same as their `sfdx` equivalents. To migrate, rename your existing commands and flags. For example, let's say you have this command.

  ```
  force:apex:execute --targetusername <org> ---apexcodefile <file>
  ```

  Here's its `sf`-style equivalent.

  ```
  apex run --target-org <org> --file <file>
  ```

- For other `sfdx`-style commands, we created `sf` commands that likely behave differently, so migrating to them requires a bit more effort. The inputs and JSON output for these new commands are also likely different from their `sfdx` equivalents. For example, `force:org:create` is now two commands: `org create scratch` and `org create sandbox`.

  In these cases the `sfdx` command is still available for backward compatibility.

- Configuration and environment variables have new names. For example, `targetusername` is now `target-org`, and SFDX_DEFAULTUSERNAME is now SF_TARGET_ORG.
- We no longer use the `force` topic, except for a handful of commands that we kept for backward compatibility.

Read these usage differences between the `sfdx`-style and `sf`-style commands, and apply them when necessary.

- When flags for new `sf` commands take multiple values, you specify the flag multiple times, with each flag taking a different single value. For example:

  ```
  sf project deploy start --metadata ApexClass:SampleDataController --metadata
  ApexClass:PropertyController
  ```

  Previously, with the deprecated `sfdx`-style commands, you specified the flag one time and separated the values with commas. For example:

  ```
  sfdx force:source:deploy --metadata
  ApexClass:SampleDataController,ApexClass:PropertyController
  ```

  You can continue using this comma-separated style with existing commands before you migrate. But when you migrate to the `sf` commands, make sure that you use this new style because new and future commands don't support the comma-separated style.

An example is specifying multiple Apex test classes and code coverage formats to the `project deploy start` command. If you continue using commas, the command doesn't return an error, but the Apex tests probably didn't all run.

For example, use this syntax.

```
sf project deploy start --metadata ApexClass --tests FirstTest --tests SecondTest --tests
 "Third Test" --coverage-formatters json --coverage-formatters html
```

But don't use this syntax.

```
sf project deploy start --metadata ApexClass --tests FirstTest,SecondTest,"Third Test"
 --coverage-formatters json,html
```

- The `sf` commands accept either spaces or colons between topics, commands, and subcommands. For example, both of these command formats to get a configuration variable are valid.

```
sf config get target-org
sf config:get target-org
```

Run the `dev convert script` Command

Begin your migration by running the `dev convert script` command to update your CI scripts. The command replaces many of the `sfdx` commands and flags with their `sf` equivalents.

Warning: We provide the `dev convert script` command to get you started with the migration. To ensure that they work as expected, you must test the converted scripts thoroughly.

First, install the `plugin-dev` Salesforce CLI plugin, which contains the conversion command.

```
sfdx plugins install @salesforce/plugin-dev
```

Then pass your script file to the `dev convert script` command with the `--script` flag.

```
sfdx dev convert script --script ./myScript.yml
```

The command scans your script file, and each time it finds an `sfdx` command or flag, it prompts whether you want to replace it with the new `sf` equivalent. The command doesn't change your original file, it instead creates a file with the replacements, such as `myScript-converted.yml`.

While `dev convert script` can convert a large portion of your script, it likely can't convert all of it because there's not always a one-to-one mapping between the previous and new commands. In these cases, `dev convert script` doesn't replace the `sfdx`-style command but instead adds a comment that starts with `#ERROR`.

Migrate Scripts Manually

Because `dev convert script` typically can't convert your entire script, you must migrate the remainder of the commands manually. You can update your entire script manually if you want.

The easiest way to find the `sf`-style equivalent of a `sfdx` command is to read the `sfdx` section of the Salesforce CLI Command Reference. Each deprecated `sfdx` command displays information about the new equivalent `sf` command and the new flag names.

You can also look at the deprecation warnings when you run an `sfdx` command. The warnings display the new `sf`-style equivalent command and flag names. To display help information about the *new* equivalent command along with examples, run the old command with the `--help` flag.

Most commands are a simple one-to-one mapping, including flag name changes. Let's take `auth:jwt:grant` as an example. The reference tells you to use the new `org login jwt` command instead, and it lists how the flag names have changed. Here's an example of the deprecated `sfdx`-style command.

```
sfdx auth:jwt:grant --username jdoe@example.org --jwtkeyfile /Users/jdoe/JWT/server.key
--clientid 123456 --setdefaultdevhubusername
```

Here's the `sf`-style equivalent.

```
sf org login jwt --username jdoe@example.org --jwt-key-file /Users/jdoe/JWT/server.key
--client-id 123456 --set-default-dev-hub
```

The `force:apex` commands also have a one-to-one mapping to the new `sf`-style commands. Here's an example of the `force:apex:test:run` command.

```
sfdx force:apex:test:run --suitenames "MySuite,MyOtherSuite" --codecoverage
--detailedcoverage --targetusename my-scratch --outputdir tests/output"
```

Here's the `sf`-style equivalent.

```
sf apex run test --suite-names "MySuite,MyOtherSuite" --code-coverage --detailed-coverage
 --target-org my-scratch --output-dir tests/output"
```

Some commands aren't a direct one-to-one mapping, or their behavior changed, so migrating them requires more effort. For additional information about these commands, see these topics.

- force:source:* and force:madpi:* Commands Migration
- force:org:* Commands Migration
- force:data:bulk:* Commands Migration

High-Level Overview of Common Flag Name Changes

This table provides an overview of common Salesforce CLI flag name changes.

`sfdx`-**Style Flag Name**	`sf`-**Style Flag Name**
`--targetusername, -u`	`--target-org, -o`
`--targetdevhubusername,-v`	`--target-dev-hub, -v`
`--apiversion`	`--api-version`
`--loglevel`	No equivalent. Use the SF_LOG_LEVEL environment variable instead.
`--json`	`--json` (No change)

For less common flags, the `sf`-style name is often similar to the `sfdx`-style one, but it has dashes to make it easier to read. We also standardized many of the flags across all topics and commands, such as using `--output-dir` consistently for the directory to write the results of a command. Here are a few more examples.

- `project:create --outputdir` is now `project generate --output-dir`.
- `force:source:deploy --sourcepath` is now `project deploy start --source-dir`.
- `force:apex:class:create --classname` is now `apex generate class --name`.

- `force:package:create --errornotificationusername` is now `package create --error-notification-username`.

As always, for command and flag name changes for a specific deprecated `sfdx` command, see its reference page in the Salesforce CLI Command Reference.

force:source:* and force:madpi:* Commands Migration

Migrating the `force:source:*` and `force:mdapi:*` commands is straightforward in most cases, although some scenarios require some rework.

force:org:* Commands Migration

Migrating the `force:org:*` commands is straight forward in most cases, although some scenarios require some rework.

force:data:bulk:* Commands Migration

We added four new `sf` commands that use Bulk API 2.0 to upsert and delete data to and from your org. All the `sfdx` commands use Bulk API 1.0.

Configuration and Environment Variable Names Migration

Because the `dev convert script` conversion command doesn't update configuration and environment variables to their new names, we recommend that you update them manually to avoid deprecation warnings. Although the existing `sfdx`-style variable names continue to work, we recommend that you start using the new `sf`-style ones. When you use the old ones, you get a warning with the name of the new configuration and environment variable to use.

Source Tracking in New sf-Style Commands

Source tracking in the new `sf`-style commands works basically the same as in the `sfdx`-style commands, but with a few small differences outlined in this topic.

Mapping sfdx Commands to Their sf Equivalents

This table maps the `sfdx`-style commands, such as `force:org:create`, to their closest `sf`-style equivalent, such as `org create sandbox` or `org create scratch`. To help you migrate your continuous integration (CI) scripts to use the new `sf`-style commands, each `sfdx`-style entry links to a command reference page that provides more information.

Mapping sf Commands to Their sfdx Equivalents

This table maps the core `sf`-style commands, such as `org create sandbox`, to their closest `sfdx`-style equivalent, such as `force:org:create`.

`force:source:*` and `force:madpi:*` Commands Migration

Migrating the `force:source:*` and `force:mdapi:*` commands is straightforward in most cases, although some scenarios require some rework.

We introduced two `sf`-style commands, `project deploy start` and `project retrieve start`, to replace these six deprecated `force` commands.

- `force:source:push`
- `force:source:pull`
- `force:source:deploy`
- `force:source:retrieve`
- `force:mdapi:deploy`
- `force:mdapi:retrieve`

It was often confusing to determine which `force` command to use because they all have similar functionality. For example, both `force:source:push` and `force:source:deploy` move source format files from your project to the org. Now it's simple:

use `project deploy start` to deploy metadata to your org and `project deploy retrieve` to retrieve metadata from your org.

By default, both new commands work with files in source format. If you want to deploy or retrieve in metadata format, use the `--metadata-dir` or `--target-metadata-dir` flags, respectively.

The `project deploy|retrieve start` commands support source tracking. However, because these two commands encapsulate the functionality of the six `force` commands, source tracking works a bit differently. For more information, see Source Tracking in New sf-Style Commands.

The table summarizes the mapping between the `force:source:*` and `force:mdapi:*` commands to their new `sf`-style equivalents. The usage notes indicate if the mapping is a simple one-to-one. If it is, you migrate them by replacing their command and flag names in your scripts. Some command migrations require changes beyond simple name replacements, or the functionality has changed, as described in the usage notes. For more guidance, see the examples after the table.

For all command migrations, refer to the reference pages for each `force` command in the Salesforce CLI Command Reference for details.

- `force:source:*`
- `force:mdapi:*`

`force:source|mdapi:*` **Command**	**Equivalent `sf`-Style Command**	**Usage Notes**
`force:mdapi:convert`	`project convert mdapi`	One-to-one mapping
`force:mdapi:deploy`	`project deploy start`	The `project deploy start` command works with source and metadata format files. The default is source format. To deploy metadata format files, use the `--metadata-dir` flag.
`force:mdapi:deploy:cancel`	`project deploy cancel`	One-to-one mapping
`force:mdapi:deploy:report`	`project deploy report` or `project deploy resume`	The `force:mdapi:deploy:report` command does more than just report, it also resumes a deployment. We now provide two new commands for each task, which is more intuitive. The `project deploy resume` command doesn't include the `--wait -1` flag, which means "wait forever."Instead, specify a large number with the new commands to ensure enough time to complete the deployment.
`force:mdapi:describemetadata`	`org list metadata-types`	One-to-one mapping. The new command is in the `org` topic.
`force:mdapi:listmetadata`	`org list metadata`	One-to-one mapping. The new command is in the `org` topic.

`force:source|mdapi:*` **Command**	**Equivalent `sf`-Style Command**	**Usage Notes**
`force:mdapi:retrieve`	`project retrieve start`	The `project retrieve start` command works with source and metadata format files. The default is source format. To retrieve metadata format files, use the `--target-metadata-dir` flag.
`force:mdapi:retrieve:report`	No equivalent.	We removed this command and provide no new equivalent because it's not needed.
`force:source:convert`	`project convert source`	One-to-one mapping
`force:source:delete`	`project delete source`	One-to-one mapping
`force:source:deploy`	`project deploy start`	When deploying in source format, the `project deploy start` command always tracks your source if the org is enabled for source tracking. So there's no new equivalent for the `force:source --tracksource` flag. If you don't want to use source tracking, create an org that doesn't have source tracking enabled on page 289. Instead of `force:source:deploy --validateddeployrequestid`, use the new `project deploy validate` and `project deploy quick --job-id` commands.
`force:source:deploy:cancel`	`project deploy cancel`	One-to-one mapping
`force:source:deploy:report`	`project deploy report` or `project deploy resume`	The `force:source:deploy:report` command does more than just report, it also resumes a deployment. We now provide two new commands for each task, which is more intuitive.
`force:source:ignored:list`	`project list ignored`	One-to-one mapping
`force:source:manifest:create`	`project generate manifest`	One-to-one mapping
`force:source:open`	`org open --source-file`	The `force:source:open` command is now the `--source-file` flag of the `org open` command. Also, you can now specify a browser with the `--browser` flag.

`force:source|mdapi:*` **Command**	**Equivalent `sf`-Style Command**	**Usage Notes**
`force:source:pull`	`project retrieve start`	The new `project retrieve start` command has additional flags that you can use to fine-tune the retrieve.
`force:source:push`	`project deploy start`	The `project deploy start` command doesn't support the `pushPackageDirectoriesSequentially` property of `sfdx-project.json`. The `force:source:push` command uses this property to deploy packages sequentially. If you must deploy packages sequentially, and in a specific order, use separate `project deploy start` commands in the desired order. The new `project deploy start` command has additional flags that you can use to fine-tune the deploy.
`force:source:retrieve`	`project retrieve start`	When retrieving in source format, the `project retrieve start` command always tracks your source if the org is enabled for source tracking. So there's no new equivalent for the `force:source --tracksource` flag. If you don't want to use source tracking, create an org that doesn't have source tracking enabled on page 289.
`force:source:status`	`project deploy preview` or `project retrieve preview`	We now provide two separate commands to preview a deploy or a retrieve, which is more intuitive. These `preview` commands have flags that align with their non-preview commands, such as `project deploy start`. The `force:source:status` command shows local and remote changes.
`force:source:tracking:clear`	`project delete tracking`	One-to-one mapping
`force:source:tracking:reset`	`project reset tracking`	One-to-one mapping

A few examples can help you get started with these new commands.

Note: To differentiate the examples, we preface `sfdx`-style commands with `sfdx` and `sf`-style commands with `sf`. However, you can indicate either `sf` or `sfdx` when running any CLI command.

`force:source` Examples

This `force` command converts source-formatted files into metadata format.

```
sfdx force:source:convert --rootdir path/to/source --outputdir path/to/outputdir
--packagename "My Package"
```

Here's the `sf`-style equivalent.

```
sf project convert source --root-dir path/to/source --output-dir path/to/outputdir
--package-name 'My Package'
```

This `force` command deploys multiple metadata types.

```
sfdx force:source:deploy --metadata "ApexClass,CustomObject" --testlevel RunSpecifiedTests
 --runtests MyTests --targetusername my-scratch
```

Here's the `sf`-style equivalent in which the `--metadata` flag is specified multiple times.

```
sf project deploy start --metadata ApexClass --metadata CustomObject --test-level
RunSpecifiedTests --tests MyTests --target-org my-scratch
```

This `force` command pushes (deploys) all the changes in your project to an org.

```
sfdx force:source:push --targetusername myscratch --forceoverwrite --wait 10
```

Here's the `sf`-style equivalent.

```
sf project deploy start --target-org myscratch --ignore-conflicts --wait 10
```

This `force` command retrieves the source in the specified directories.

```
sfdx force:source:retrieve --sourcepath
"path/to/objects/MyCustomObject/fields/MyField.field-meta.xml, path/to/apex/classes"
```

Here's the `sf`-style equivalent in which the `--source-dir` flag is specified multiple times.

```
sf project retrieve start --source-dir
path/to/objects/MyCustomObject/fields/MyField.field-meta.xml --source-dir
path/to/apex/classes
```

This `force` command opens a metadata file in Lightning App Builder.

```
sfdx force:source:open --source-file
force-app/main/default/flexipages/Hello.flexipage-meta.xml --urlonly --targetusername
myscratch
```

Here's the `sf`-style equivalent that uses the `org open` command.

```
sf org open --source-path force-app/main/default/flexipages/Hello.flexipage-meta.xml
--url-only --target-org myscratch
```

This `force` command pulls (retrieves) all the changes in your org to your project.

```
sfdx force:source:pull --targetusername myscratch --forceoverwrite --wait 10
```

Here's the `sf`-style equivalent.

```
sf project retrieve start --target-org myscratch --ignore-conflicts --wait 10
```

This `force` command shows how your local project differs from the org.

```
sfdx force:source:status --targetusername myscratch --local
```

Here's the `sf`-style equivalent; the command requires that you specify what you want preview, in this case, with the `--manifest` flag.

```
sf project deploy preview --target-org myscratch --manifest package.xml
```

`force:mdapi` Examples

This `force` command deploys metadata format files in the specified directory.

```
sfdx force:mdapi:deploy --deploydir some/path --wait 1000 --checkonly --testlevel
RunAllTestsInOrg --targetusername my-test-org
```

There are two `sf`-style equivalents.

```
sf project deploy start --dry-run --metadata-dir some/path --wait 1000 --test-level
RunAllTestsInOrg --target-org my-test-org
sf project deploy validate --metadata-dir some/path --wait 1000 --test-level RunAllTestsInOrg
 --target-org my-test-org
```

This `force` command deploys a .zip file that contains metadata files.

```
sfdx force:mdapi:deploy sfdx force:mdapi:deploy --zipfile stuff.zip --resultsdir --junit
```

Here's the `sf`-style equivalent.

```
sf project deploy start --metadata-dir stuff.zip --results-dir --junit
```

This `force` command retrieves metadata defined in a manifest file into the target directory.

```
sfdx force:mdapi:retrieve --retrievetargetdir path/to/retrieve/dir --unpackaged package.xml
```

Here's the `sf`-style equivalent.

```
sf project retrieve start --target-metadata-dir path/to/retrieve/dir --manifest package.xml
```

Overview of New Commands and Functionality

In addition to the new `project deploy start` and `project retrieve start`, we introduced other commands and flags that improve the Salesforce CLI's usability.

- Preview a deployment to your org with the `project deploy preview` command.

  The command outputs a table that shows what happens when you run the `project deploy start` command. The table displays a preview of the metadata components that are deployed and deleted, and the current conflicts between your project and org. The table also lists the files that aren't deployed because they're included in your `.forceignore` file.

- Similarly, preview a retrieve from your org with the `project retrieve preview` command.
- Validate a deployment, and then quickly deploy it later, with the `project deploy validate` and `project deploy quick` command pair.

  Use `project deploy validate` to verify whether a deployment can succeed without actually deploying the metadata to your org. This command is similar to `project deploy start`, except that you're required to run Apex tests, and the command returns a job ID rather than actually executing the deployment. If the validation succeeds, then you pass this job ID to the `project` `deploy quick` command to actually deploy the metadata. This type of deploy takes less time because it skips running Apex tests.

You can also use the `--dry-run` flag of `project deploy start` to get a preview of a deploy. Use this preview method if you don't plan to later do a quick deploy. This way of previewing provides more flexibility because you can use all the flags of the `project deploy start` command, such as making destructive changes with the `--pre|post-destructive-changes` flags. The `project deploy validate` provides just a subset of the full deployment flags.

- These new deploy commands that take a job ID now also have the handy `--use-most-recent` flag to automatically use the job ID of the most recent deploy operation.
  - `project deploy cancel`
  - `project deploy quick`
  - `project deploy report`
  - `project deploy resume`
- These new deploy commands have the `--async` flag to run the command asynchronously.
  - `project deploy cancel`
  - `project deploy quick`
  - `project deploy resume`
  - `project deploy validate`
- Delete source from a non-source-tracked org with the `project delete source` command.

`force:org:*` Commands Migration

Migrating the `force:org:*` commands is straight forward in most cases, although some scenarios require some rework.

This table summarizes the mapping between the existing `force:org:*` commands and their new `sf`-style equivalents. The usage notes indicate if the mapping is one-to-one. If it is, you migrate them by changing their command and flag names as listed in the reference page for each `force:org:*` command. Some command migrations require more changes, as described in the usage notes. For more guidance, see the examples after the table.

`force:org:*` **Command**	**Equivalent** `sf`**-Style Command**	**Usage Notes**
`force:org:clone`	`org create sandbox`	To clone a sandbox, specify the `--clone` flag of the `org create sandbox` command and set it to the name of the sandbox being cloned. To specify the new sandbox's name, use the `--name` flag t. These two flags replace the `SandboxName` and `SourceSandboxName` key value pairs that you could specify with the `force:org:clone` command. See examples later in this section.

`force:org:*` **Command**	**Equivalent `sf`-Style Command**	**Usage Notes**
`force:org:create`	`org create scratch` or `org create sandbox`	We provide two commands to create an org, one for sandbox and one for scratch org. You can no longer specify key-value pairs, such as `sandboxName=FullSbx`, from the scratch or sandbox configuration file at the command line. You must either use the provided flags or update the definition file. See examples later in this section.
`force:org:delete`	`org delete scratch` or `org delete sandbox`	We provide two commands to delete an org, one for sandbox and one for scratch org.
`force:org:display`	`org display`	One-to-one mapping
`force:org:list`	`org list`	One-to-one mapping
`force:org:open`	`org open`	One-to-one mapping
`force:org:status`	`org resume sandbox`	The `force:org:status` command did more than report the status of a sandbox create, it also resumed a sandbox create if it wasn't complete. We now provide an explicit command to resume the org creation, which is more intuitive. To specify the incomplete sandbox creation job, use the `--job-id` or `--use-most-recent` flags.

We also introduced the command `org resume scratch` to resume a scratch org creation if it times out. Previously, you could no longer connect to it, and you manually deleted it from your Dev Hub org. Now you can resume where it left off using a job ID or the `--use-most-recent` flag. When the org creation finishes, the command automatically authenticates to the org, saves the org info locally, and deploys any configured settings.

A few examples can help you get started with these new commands.

Note: To differentiate the examples, we preface `sfdx`-style commands with `sfdx` and `sf`-style commands with `sf`. However, you can indicate either `sf` or `sfdx` when running any CLI command.

Let's start with the deprecated `force:org:create` command. The reference tells you to use either `org create sandbox` or `org create scratch`, depending on what you want to create. Let's say you want to migrate this `force` command.

```
sfdx force:org:create --definitionfile config/scratch-def.json --setalias MyScratchOrg
--targetdevhubusername MyDevHub --nonamespace --setdefaultusername
```

Because the command creates a scratch org, use this equivalent `sf` command.

```
sf org create scratch --definition-file config/scratch-def.json --alias MyScratchOrg
--target-dev-hub MyDevHub --no-namespace --set-default
```

This `force` example specifies scratch org options as key-value pairs at the command line, which is no longer allowed.

```
sfdx force:org:create adminEmail=me@email.com edition=Developer
username=admin_user@orgname.org --country=GB --targetdevhubusername MyDevHub
```

In the `sf`-style equivalent, use the `--edition`, `--admin-email`, and `--username` flags instead. But because `country` doesn't have an equivalent flag, you must specify a scratch org definition file that contains the `country` option. Here's what the new command looks like.

```
sf force:org:create --definition-file config/scratch-def.json --admin-email me@email.com
--edition=developer --username=admin_user@orgname.org --targetdevhubusername MyDevHub
```

In the previous example, the `--edition` flag takes lowercase values for Salesforce editions. To see the full list of valid editions, run `org create scratch -h`.

Here's an example of a scratch org definition file that contains the `country` option.

```
{
    "orgName": "Dreamhouse",
    "edition": "Developer",
    "country": "GB",
    "features": ["Walkthroughs", "EnableSetPasswordInApi"],
    "settings": {
        "lightningExperienceSettings": {
            "enableS1DesktopEnabled": true
        },
        "mobileSettings": {
            "enableS1EncryptedStoragePref2": false
        }
    }
}
```

This `force` example creates a sandbox.

```
sfdx force:org:create --type sandbox --definitionfile config/dev-sandbox-def.json --setalias
 MyDevSandbox --targetusername ProdOrg
```

Here's the equivalent `sf`-style command.

```
sf org create sandbox --definition-file config/dev-sandbox-def.json --alias MyDevSandbox
--target-org ProdOrg
```

This `force` example clones a sandbox by specifying the `SourceSandboxName` and `SandboxName` key-value pairs at the command line.

```
sfdx force:org:clone --type sandbox SourceSandboxName=ExistingSandbox
SandboxName=NewClonedSandbox --setalias MyDevSandbox --targetusername ProdOrg
```

In the `sf`-style command, use flags instead.

```
sf org create sandbox --clone ExistingSandbox --name NewClonedSandbox --alias MyDevSandbox
 --target-org ProdOrg
```

This `force` example deletes a scratch org.

```
sfdx force:org:delete --targetusername MyScratchOrg --noprompt
```

Here's the equivalent `sf`-style command.

```
sf org delete scratch --target-org MyScratchOrg --no-prompt
```

`force:data:bulk:*` Commands Migration

We added four new `sf` commands that use Bulk API 2.0 to upsert and delete data to and from your org. All the `sfdx` commands use Bulk API 1.0.

These new `sf` commands use Bulk API 2.0.

- `data delete bulk`
- `data delete resume`
- `data upsert bulk`
- `data upsert resume`

We generally recommend that you start using the new `sf` commands instead of these equivalent `sfdx` commands that use Bulk API 1.0.

- `force:data:bulk:delete`
- `force:data:bulk:upsert`
- `force:data:bulk:status`

However, one reason to keep using the `force:data:bulk:upsert` command is if you want to run the upsert serially with the `--serial` flag. The new Bulk API 2.0 commands don't support serial execution. For this reason, and for users who want to continue using Bulk API 1.0, we aren't deprecating the `force:data:bulk:*` commands at this time.

Configuration and Environment Variable Names Migration

Because the `dev convert script` conversion command doesn't update configuration and environment variables to their new names, we recommend that you update them manually to avoid deprecation warnings. Although the existing `sfdx`-style variable names continue to work, we recommend that you start using the new `sf`-style ones. When you use the old ones, you get a warning with the name of the new configuration and environment variable to use.

Configuration Variables

The `sfdx`-style configuration variables are aliased to their `sf`-style equivalents. As a result, you can use either the `sfdx` or the `sf` variable names with the `config` commands. But the commands always work on the `sf` variable names. For example, `config set` and `config unset` always set the configuration with the `sf` name, even if you specify the `sfdx` name. All `config` commands display the `sf` name in their outputs, even if you specified the `sfdx` name in the command.

These examples show the rules in action.

```
sf config set defaultusername=my-scratch-org
Warning: Deprecated config name: defaultusername. Please use target-org instead.
Set Config
===================================
| Name       Value           Success
| ────────── ─────────────── ───────
| target-org my-scratch-org true

sf config list
List Config
```

```
===========================================
| Name                Value          Location
| ─────────────────── ────────────── ────────
| target-org          my-scratch-org Local

sf config get defaultusername
Warning: Deprecated config name: defaultusername. Please use target-org instead.
Get Config
===================================
| Name       Value          Success
| ────────── ────────────── ───────
| target-org my-scratch-org true
```

Use this table to migrate your scripts to use the new `sf`-style configuration variable names.

`sfdx`-**style Configuration Value**	**Equivalent** `sf`**-style Configuration Variable**
`apiVersion`	`org-api-version`
`customOrgMetadataTemplates`	`org-custom-metadata-templates`
`defaultdevhubusername`	`target-dev-hub`
`defaultusername`	`target-org`
`disableTelemetry`	`disable-telemetry`
`instanceUrl`	`org-instance-url`
`maxQueryLimit`	`org-max-query-limit`
`restDeploy`	`org-metadata-rest-deploy`

Environment Variables

You can set both the new `sf`-style and old `sfdx`-style environment variables. However, if they're set to different values, Salesforce CLI uses the `sf` one and displays a warning.

To migrate most environment variables, change the initial SFDX to SF. However, some variables have bigger changes, as displayed in this table, while others haven't changed their name. For the full list, see Salesforce CLI Environment Variables.

Equivalent `sf`**-style Environment Variable**	**Equivalent** `sf`**-style Environment Variable**
SFDX_API_VERSION	SF_ORG_API_VERSION
SFDX_CUSTOM_ORG_METADATA_TEMPLATES	SF_ORG_CUSTOM_METADATA_TEMPLATES
SFDX_DEFAULTDEVHUBUSERNAME	SF_TARGET_DEV_HUB
SFDX_DEFAULTUSERNAME	SF_TARGET_ORG
SFDX_INSTANCE_URL	SF_ORG_INSTANCE_URL
SFDX_MAX_QUERY_LIMIT	SF_ORG_MAX_QUERY_LIMIT
SFDX_REST_DEPLOY	SF_ORG_METADATA_REST_DEPLOY

For example, here's how to set your default Dev Hub org to an alias with an environment variable before running the command to create a scratch org.

```
SF_TARGET_DEV_HUB
=MyDevHub sf org create scratch --definition-file config/scratch-def.json
```

Source Tracking in New `sf`-Style Commands

Source tracking in the new `sf`-style commands works basically the same as in the `sfdx`-style commands, but with a few small differences outlined in this topic.

These `sf`-style commands support source tracking.

- `project deploy start`
- `project delete source`
- `project retrieve start`

The `sf`-style commands encapsulate the functionality of these six deprecated `sfdx`-style commands.

- `force:source:push|pull`
- `force:source:deploy|retrieve`
- `force:mdapi:deploy|retrieve`

Let's start with deploying. The first time you run `project deploy start` on a scratch or sandbox org that allows source tracking, the command deploys all the source files from your local project. But when you next run the command, it deploys only the files that changed locally. If you use one of the flags to narrow the deploy list, `--source-dir`, `--metadata`, or `--manifest`, then the command deploys only the changed files in the specified directory, metadata, or manifest. If you don't specify any of the flags, then the command deploys all changes in the project, similar to how the `sfdx`-style command `force:source:push` works.

If you run `project retrieve start` on a newly created org, nothing happens because there are no changes to track yet. When you next run the command, any changes in the org are retrieved. These changes include updates from other users who connect to the org, not just your changes. If you don't specify `--source-dir`, `--metadata`, or `--manifest`, then all changes in the org are retrieved, just like the `sfdx`-style command `force:source:pull`.

If one of these commands detects a conflict in the files you're about to deploy or retrieve, the command displays the conflicts. To force the deployment or retrieval of the changes, use the `--ignore-conflicts` flag. This flag is similar to the `--forceoverwrite` flag of many of the `force:source` commands. For example:

```
sf project deploy start --source-dir force-app --ignore-conflicts
```

Determine If Your Org Allows Source Tracking

Source tracking works only if your target org allows it. Don't worry, you can still deploy or retrieve metadata to and from an org without source tracking. But the commands don't check for conflicts, and you must specify what you want to deploy or retrieve using an appropriate flag, such as `--source-dir`.

Here's how to determine whether your org allows source tracking.

- For Developer Edition orgs, production orgs, Partial Copy sandboxes, and Full sandboxes, source tracking isn't available.
- For Developer and Developer Pro sandboxes:
  - Source tracking is enabled if their associated production org has been enabled for source tracking.

  - Source tracking is possible when you create the sandbox with the `--no-track-source` flag of the `org create sandbox` command. For example:

    ```
    sf org create sandbox --definition-file config/dev-sandbox-def.json --target-org
    prodOrg --no-track-source
    ```

- Scratch orgs have source tracking by default.
  - You can opt out of source tracking when you create the scratch org with the `--no-track-source` flag of the `org create scratch` command. This flag affects only your local configuration, not the scratch org itself. Salesforce CLI sets a local configuration option `trackSource: false` as part of your authorization information to the org. If you log out of the scratch org and then log back in again, source tracking is enabled again by default. Here's how to create a scratch org with source tracking disabled.

    ```
     sf org create scratch --target-dev-hub=MyHub --definition-file
    config/project-scratch-def.json --no-track-source
    ```

SEE ALSO:

Salesforce DX Developer Guide: Track Changes Between Your Project and Org

Mapping `sfdx` Commands to Their `sf` Equivalents

This table maps the `sfdx`-style commands, such as `force:org:create`, to their closest `sf`-style equivalent, such as `org create sandbox` or `org create scratch`. To help you migrate your continuous integration (CI) scripts to use the new `sf`-style commands, each `sfdx`-style entry links to a command reference page that provides more information.

`sfdx`-**style Command**	**Equivalent** `sf`-**style Command**
`alias:list` on page 300	`alias list`
`alias:set` on page 301	`alias set`
`alias:unset` on page 302	`alias unset`
`auth:accesstoken:store` on page 306	`org login access-token`
`auth:device:login` on page 308	`org login device`
`auth:jwt:grant` on page 310	`org login jwt`
`auth:list` on page 303	`org list auth`
`auth:logout` on page 304	`org logout`
`auth:sfdxurl:store` on page 312	`org login sfdx-url`
`auth:web:login` on page 314	`org login web`
`autocomplete`	`autocomplete`
`config:get` on page 317	`config get`
`config:list` on page 318	`config list`
`config:set` on page 319	`config set`

`sfdx`**-style Command**	**Equivalent** `sf`**-style Command**
`config:unset` on page 320	`config unset`
`doctor`	`doctor`
`force:analytics:template:create` on page 325	`analytics generate template`
`force:apex:class:create` on page 327	`apex generate class`
`force:apex:execute` on page 328	`apex run`
`force:apex:log:get` on page 330	`apex get log`
`force:apex:log:list` on page 332	`apex list log`
`force:apex:log:tail` on page 333	`apex tail log`
`force:apex:test:report` on page 335	`apex get test`
`force:apex:test:run` on page 337	`apex run test`
`force:apex:trigger:create` on page 340	`apex generate trigger`
`force:cmdt:create` on page 342	`cmdt generate object`
`force:cmdt:field:create` on page 344	`cmdt generate field`
`force:cmdt:generate` on page 346	`cmdt generate fromorg`
`force:cmdt:record:create` on page 349	`cmdt generate record`
`force:cmdt:record:insert` on page 351	`cmdt generate records`
`force:community:create` on page 353	`community create`
`force:community:publish` on page 355	`community publish`
`force:community:template:list` on page 357	`community list template`
`force:data:bulk:delete` on page 359	• `data delete bulk` (Bulk API 2.0) • `force data bulk delete` (Bulk API 1.0)
`force:data:bulk:status` on page 361	• `data delete resume` (Bulk API 2.0) • `data upsert resume` (Bulk API 2.0) • `force data bulk status` (Bulk API 1.0)
`force:data:bulk:upsert` on page 362	• `data upsert bulk` (Bulk API 2.0) • `force data bulk upsert` (Bulk API 1.0)
`force:data:record:create` on page 365	`data create record`
`force:data:record:delete` on page 367	`data delete record`
`force:data:record:get` on page 369	`data get record`

`sfdx`-style Command	Equivalent `sf`-style Command
`force:data:record:update` on page 371	`data update record`
`force:data:soql:bulk:report` on page 374	`data query resume`
`force:data:soql:query` on page 375	`data query`
`force:data:tree:export` on page 378	`data export tree`
`force:data:tree:import` on page 380	`data import tree`
`force:lightning:app:create` on page 382	`lightning generate app`
`force:lightning:component:create` on page 384	`lightning generate component`
`force:lightning:event:create` on page 386	`lightning generate event`
`force:lightning:interface:create` on page 388	`lightning generate interface`
`force:lightning:lwc:test:create` on page 390	`force lightning lwc test create`
`force:lightning:lwc:test:run` on page 390	`force lightning lwc test run`
`force:lightning:lwc:test:setup` on page 392	`force lightning lwc test setup`
`force:lightning:test:create` on page 392	`lightning generate test`
`force:limits:api:display` on page 394	`limits api display`
`force:limits:recordcounts:display` on page 395	`limits recordcounts display`
`force:mdapi:convert` on page 398	`project convert mdapi`
`force:mdapi:deploy` on page 400	`project deploy start --metadata-dir`
`force:mdapi:deploy:cancel` on page 406	`project deploy cancel`
`force:mdapi:deploy:report` on page 408	`project deploy report|resume`
`force:mdapi:describemetadata` on page 411	`org list metadata-types`
`force:mdapi:listmetadata` on page 413	`org list metadata`
`force:mdapi:retrieve` on page 415	`project retrieve start --target-metadata-dir`
`force:mdapi:retrieve:report` on page 419	No equivalent.
`force:org:clone` on page 422	`org create sandbox --clone`
`force:org:create` on page 424	• `org create scratch` • `org create sandbox`
`force:org:delete` on page 427	• `org delete scratch` • `org delete sandbox`
`force:org:display` on page 429	`org display`

`sfdx`-style Command	Equivalent `sf`-style Command
`force:org:list` on page 431	`org list`
`force:org:open` on page 432	`org open`
`force:org:shape:create` on page 434	`org create shape`
`force:org:shape:delete` on page 435	`org delete shape`
`force:org:shape:list` on page 437	`org list shape`
`force:org:snapshot:create` on page 438	`org create snapshot`
`force:org:snapshot:delete` on page 440	`org delete snapshot`
`force:org:snapshot:get` on page 442	`org get snapshot`
`force:org:snapshot:list` on page 443	`org list snapshot`
`force:org:status` on page 445	`org resume sandbox`
`force:package1:version:create` on page 485	`package1 version create`
`force:package1:version:create:get` on page 487	`package1 version create get`
`force:package1:version:display` on page 489	`package1 version display`
`force:package1:version:list` on page 490	`package1 version list`
`force:package:create` on page 448	`package create`
`force:package:delete` on page 451	`package delete`
`force:package:install` on page 452	`package install`
`force:package:install:report` on page 455	`package install report`
`force:package:installed:list` on page 457	`package installed list`
`force:package:list` on page 458	`package list`
`force:package:uninstall` on page 460	`package uninstall`
`force:package:uninstall:report` on page 461	`package uninstall report`
`force:package:update` on page 463	`package update`
`force:package:version:create` on page 465	`package version create`
`force:package:version:create:list` on page 469	`package version create list`
`force:package:version:create:report` on page 471	`package version create report`
`force:package:version:delete` on page 473	`package version delete`
`force:package:version:displayancestry` on page 474	`package version displayancestry`
`force:package:version:list` on page 476	`package version list`

`sfdx`-**style Command**	**Equivalent** `sf`-**style Command**
`force:package:version:promote` on page 479	`package version promote`
`force:package:version:report` on page 480	`package version report`
`force:package:version:update` on page 482	`package version update`
`force:project:create` on page 491	`project generate`
`force:schema:sobject:describe` on page 494	`sobject describe`
`force:schema:sobject:list` on page 495	`sobject list`
`force:source:convert` on page 497	`project convert source`
`force:source:delete` on page 500	`project delete source`
`force:source:deploy` on page 503	`project deploy start`
`force:source:deploy:cancel` on page 511	`project deploy cancel`
`force:source:deploy:report` on page 513	`project deploy report|resume`
`force:source:ignored:list` on page 516	`project list ignored`
`force:source:manifest:create` on page 517	`project generate manifest`
`force:source:open` on page 520	`org open --source-file`
`force:source:pull` on page 522	`project retrieve start`
`force:source:push` on page 524	`project deploy start`
`force:source:retrieve` on page 526	`project retrieve start`
`force:source:status` on page 531	• `project deploy preview` • `project retrieve preview`
`force:source:tracking:clear` on page 533	`project delete tracking`
`force:source:tracking:reset` on page 535	`project reset tracking`
`force:staticresource:create` on page 537	`static-resource generate`
`force:user:create` on page 539	`org create user`
`force:user:display` on page 541	`org display user`
`force:user:list` on page 542	`org list users`
`force:user:password:generate` on page 544	`org generate password`
`force:user:permset:assign` on page 546	`org assign permset`
`force:user:permsetlicense:assign` on page 547	`org assign permsetlicense`
`force:visualforce:component:create` on page 549	`visualforce generate component`
`force:visualforce:page:create` on page 551	`visualforce generate page`

`sfdx`-**style Command**	**Equivalent** `sf`-**style Command**
`help`	`help`
`info:releasenotes:display` on page 553	`info releasenotes display`
`search`	`search`
`which`	`which`

Mapping `sf` Commands to Their `sfdx` Equivalents

This table maps the core `sf`-style commands, such as `org create sandbox`, to their closest `sfdx`-style equivalent, such as `force:org:create`.

`sf`-**style Command**	**Equivalent** `sfdx`-**style Command**
`alias set`	`alias:set`
`alias unset`	`alias:unset`
`alias list`	`alias:list`
`analytics generate template`	`force:analytics:template:create`
`apex generate class`	`force:apex:class:create`
`apex generate trigger`	`force:apex:trigger:create`
`apex run`	`force:apex:execute`
`apex get log`	`force:apex:log:get`
`apex list log`	`force:apex:log:list`
`apex tail log`	`force:apex:log:tail`
`apex get test`	`force:apex:test:report`
`apex run test`	`force:apex:test:run`
`autocomplete`	`autocomplete`
`cmdt generate object`	`force:cmdt:create`
`cmdt generate field`	`force:cmdt:field:create`
`cmdt generate fromorg`	`force:cmdt:generate`
`cmdt generate record`	`force:cmdt:record:create`
`cmdt generate records`	`force:cmdt:record:insert`
`community create`	`force:community:create`
`community list template`	`force:community:template:list`
`community publish`	`force:community:publish`

`sf`-style Command	Equivalent `sfdx`-style Command
`config get`	`config:get`
`config list`	`config:list`
`config set`	`config:set`
`config unset`	`config:unset`
`data create record`	`force:data:record:create`
`data delete bulk` (Uses Bulk API 2.0)	`force:data:bulk:delete` (Uses Bulk API 1.0)
`data delete resume` (Uses Bulk API 2.0)	`force:data:bulk:status` (Uses Bulk API 1.0)
`data delete record`	`force:data:record:delete`
`data export tree`	`force:data:tree:export`
`data get record`	`force:data:record:get`
`data import tree`	`force:data:tree:import`
`data query`	`force:data:soql:query`
`data query resume`	`force:data:soql:bulk:report`
`data resume` (Deprecated)	`force:data:bulk:status`
`data update record`	`force:data:record:update`
`data upsert bulk` (Uses Bulk API 2.0)	`force:data:bulk:upsert` (Uses Bulk API 1.0)
`data upsert resume` (Uses Bulk API 2.0)	`force:data:bulk:status` (Uses Bulk API 1.0)
`deploy` (interactive) Deprecated	No equivalent.
`doctor`	`doctor`
`help`	`help`
`info releasenotes display`	`info:releasenotes:display`
`lightning generate app`	`force:lightning:app:create`
`lightning generate component`	`force:lightning:component:create`
`lightning generate event`	`force:lightning:event:create`
`lightning generate interface`	`force:lightning:interface:create`
`lightning generate test`	`force:lightning:test:create`
`limits api display`	`force:limits:api:display`
`limits recordcounts display`	`force:limits:recordcounts:display`
`login` (interactive) Deprecated	No equivalent.
`logout` (interactive) Deprecated	No equivalent.

`sf`**-style Command**	**Equivalent** `sfdx`**-style Command**
`org assign permset`	`force:user:permset:generate`
`org assign permsetlicense`	`force:user:permsetlicense:generate`
`org create shape`	`force:org:shape:create`
`org create snapshot`	`force:org:snapshot:create`
`org create user`	`force:user:create`
`org create sandbox`	`force:org:create --type sandbox`
`org create scratch`	`force:org:create --type scratch`
`org delete sandbox`	`force:org:delete`
`org delete scratch`	`force:org:delete`
`org delete shape`	`force:org:shape:delete`
`org delete snapshot`	`force:org:snapshot:delete`
`org display user`	`force:user:display`
`org display`	`force:org:display`
`org generate password`	`force:user:password:generate`
`org get snapshot`	`force:org:snapshot:get`
`org list`	`force:org:list`, `auth:list`
`org list auth`	`auth:list`
`org list metadata`	`force:mdapi:listmetadata`
`org list metadata-types`	`force:mdapi:describemetadata`
`org list shape`	`force:org:shape:list`
`org list snapshot`	`force:org:snapshot:list`
`org list users`	`force:user:list`
`org login jwt`	`auth:jwt:grant`
`org login web`	`auth:web:login`
`org login access-token`	`auth:accesstoken:store`
`org login device`	`auth:device:login`
`org login sfdx-url`	`auth:sfdxurl:store`
`org logout`	`auth:logout`
`org open`	`force:org:open`
`org resume sandbox`	`force:org:status --wait`

`sf`-style Command	Equivalent `sfdx`-style Command
`org resume scratch`	No equivalent.
`package1 version create`	`force:package1:version:create`
`package1 version create get`	`force:package1:version:create:get`
`package1 version display`	`force:package1:version:display`
`package1 version list`	`force:package1:version:list`
`package create`	`force:package:create`
`package delete`	`force:package:delete`
`package install`	`force:package:install`
`package install report`	`force:package:install:report`
`package install list`	`force:package:install:list`
`package list`	`force:package:list`
`package uninstall`	`force:package:uninstall`
`package uninstall report`	`force:package:uninstall:report`
`package update`	`force:package:update`
`package version create`	`force:package:version:create`
`package version create list`	`force:package:version:create:list`
`package version create report`	`force:package:version:create:report`
`package version delete`	`force:package:version:delete`
`package version displayancestry`	`force:package:version:displayancestry`
`package version list`	`force:package:version:list`
`package version promote`	`force:package:version:promote`
`package version report`	`force:package:version:report`
`package version update`	`force:package:version:update`
`project convert mdapi`	`force:mdapi:convert`
`project convert source`	`force:source:convert`
`project delete source`	`force:source:delete`
`project delete tracking`	`force:source:tracking:clear`
`project deploy cancel`	`force:source:deploy:cancel`
`project deploy preview`	`force:source:status`

`sf`-style Command	Equivalent `sfdx`-style Command
`project deploy quick`	`force:source:deploy --validateddeployrequestid`
`project deploy report`	`force:source:deploy:report --wait 0`
`project deploy resume`	`force:source:deploy:report`
`project deploy start`	`force:source:deploy`
`project deploy validate`	`force:source:deploy --checkonly --testlevel >NoTestRun`
`project generate`	`force:project:create`
`project generate manifest`	`force:source:manifest:create`
`project list ignored`	`force:source:ignored:list`
`project reset tracking`	`force:source:tracking:reset`
`project retrieve start`	`force:source:retrieve`
`project retrieve preview`	`force:source:status`
`search`	`search`
`schema generate field`	No equivalent.
`schema generate platformevent`	No equivalent.
`schema generate sobject`	No equivalent.
`schema generate tab`	No equivalent.
`sobject describe`	`force:schema:sobject:describe`
`sobject list`	`force:schema:sobject:list`
`static-resource generate`	`force:staticresource:create`
`version`	`version`
`visualforce generate component`	`force:visualforce:component:create`
`visualforce generate page`	`force:visualforce:page:create`
`which`	`which`

`alias` Namespace

Use the alias commands to manage username aliases.

alias:list
List username aliases for the Salesforce CLI.

alias:set

Set username aliases for the Salesforce CLI.

alias:unset

Unsets aliases for the Salesforce CLI.

`alias:list`

List username aliases for the Salesforce CLI.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style alias list command instead. Here's how the flags changed between the `sfdx` and `sf` commands; if a flag isn't listed, the `sfdx` and `sf` names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 9, 2023.

Command Syntax

`sfdx alias:list`

`[--json]`

`[--loglevel LOGLEVEL]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

Aliases for `alias:list`

```
force:alias:list
```

`alias:set`

Set username aliases for the Salesforce CLI.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style alias set command instead. Here's how the flags changed between the `sfdx` and `sf` commands; if a flag isn't listed, the `sfdx` and `sf` names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 9, 2023.

Help for `alias:set`

You can associate an alias with only one username at a time. If you've set an alias multiple times, the alias points to the most recent username.

Examples for `alias:set`

```
sfdx alias:set YourAlias=username@example.com
```

```
sfdx alias:set YourAlias=username@example.com YourOtherAlias=devhub@example.com
```

Command Syntax

`sfdx alias:set`

`[--json]`

`[--loglevel LOGLEVEL]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

Aliases for `alias:set`

```
force:alias:set
```

`alias:unset`

Unsets aliases for the Salesforce CLI.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style alias unset command instead. Here's how the flags changed between the `sfdx` and `sf` commands; if a flag isn't listed, the `sfdx` and `sf` names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- New flag: `--all`
- New flag: `--no-prompt`

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 9, 2023.

Examples for `alias:unset`

```
sfdx alias:unset YourAlias
```

```
sfdx alias:unset YourAlias YourOtherAlias
```

Command Syntax

`sfdx alias:unset`

  `[--json]`

  `[--loglevel LOGLEVEL]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`auth` Namespace

Use the auth commands to authorize a Salesforce org for use with the Salesforce CLI.

auth:list
List auth connection information.

auth:logout
Log out from authorized orgs.

accesstoken Commands
Authorize an org using an access token.

device Commands
Authorize an org using a device code.

jwt Commands
Authorize an org using JWT.

sfdxurl Commands
Authorize an org using sfdxurl.

web Commands
Authorize an org using a web browser.

`auth:list`

List auth connection information.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org list auth command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--setdefaultdevhubusername`. New name: `--set-default-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Command Syntax

`sfdx auth:list`

```
[--json]

[--loglevel LOGLEVEL]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

Aliases for `auth:list`

```
force:auth:list
```

`auth:logout`

Log out from authorized orgs.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org logout command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Removed flag: `--apiversion`
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `auth:logout`

By default, this command logs you out from your default scratch org.

Examples for `auth:logout`

```
sfdx auth:logout -u me@my.org
```

```
sfdx auth:logout -a
```

```
sfdx auth:logout -p
```

Command Syntax

`sfdx auth:logout`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-a]
[-p]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-a | --all`

Optional

Includes all authenticated orgs: for example, Dev Hubs, sandboxes, DE orgs, and expired, deleted, and unknown-status scratch orgs.

Type: boolean

`-p | --noprompt`

Optional

Do not prompt for confirmation.

Type: boolean

Aliases for `auth:logout`

```
force:auth:logout
```

`accesstoken` Commands

Authorize an org using an access token.

auth:accesstoken:store
Authorize an org using an existing Salesforce access token.

`auth:accesstoken:store`

Authorize an org using an existing Salesforce access token.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org login access-token command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--setdefaultdevhubusername`. New name: `--set-default-dev-hub`.
- Changed flag name: Old name `--setalias`. New name: `--alias`.
- Changed flag name: Old name `--instanceurl`. New name: `--instance-url`.
- Changed flag name: Old name `--setdefaultusername`. New name: `--set-default`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `auth:accesstoken:store`

By default, the command runs interactively and asks you for the access token. If you previously authorized the org, the command prompts whether you want to overwrite the local file. Specify --noprompt to not be prompted.

To use the command in a CI/CD script, set the SFDX_ACCESS_TOKEN environment variable to the access token. Then run the command with the --noprompt parameter. "<org id>!<accesstoken>"

Examples for `auth:accesstoken:store`

```
sfdx auth:accesstoken:store --instanceurl https://mycompany.my.salesforce.com
```

$ export SFDX_ACCESS_TOKEN=00Dxx0000000000!xxxxx

```
sfdx auth:accesstoken:store --instanceurl https://dev-hub.my.salesforce.com --noprompt
```

Command Syntax

`sfdx auth:accesstoken:store`

```
[--json]
[--loglevel LOGLEVEL]
-r INSTANCEURL
[-d]
[-s]
[-a SETALIAS]
[-p]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-r | --instanceurl INSTANCEURL`

Required

The login URL of the instance the org lives on.

Type: url

`-d | --setdefaultdevhubusername`

Optional

Set the authenticated org as the default dev hub org for scratch org creation.

Type: boolean

`-s | --setdefaultusername`

Optional

Set the authenticated org as the default username that all commands run against.

Type: boolean

`-a | --setalias SETALIAS`

Optional

Set an alias for the authenticated org.

Type: string

`-p | --noprompt`

Optional

Do not prompt for confirmation.

Type: boolean

Aliases for `auth:accesstoken:store`

```
force:auth:accesstoken:store
```

`device` Commands

Authorize an org using a device code.

auth:device:login
Authorize an org using a device code.

`auth:device:login`

Authorize an org using a device code.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org login device command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--clientid`. New name: `--client-id`.
- Changed flag name: Old name `--setdefaultdevhubusername`. New name: `--set-default-dev-hub`.
- Changed flag name: Old name `--setalias`. New name: `--alias`.
- Changed flag name: Old name `--instanceurl`. New name: `--instance-url`.
- Changed flag name: Old name `--setdefaultusername`. New name: `--set-default`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `auth:device:login`

You must open a browser, navigate to the verification URL, and enter the code. Log in, if not already logged in, and you'll be prompted to allow the device to connect to the org.

Examples for `auth:device:login`

```
sfdx auth:device:login -d -a TestOrg1
```

```
sfdx auth:device:login -i <OAuth client id>
```

```
sfdx auth:device:login -r https://MyDomainName--SandboxName.sandbox.my.salesforce.com
```

Command Syntax

`sfdx auth:device:login`

```
[--json]
[--loglevel LOGLEVEL]
[-i CLIENTID]
[-r INSTANCEURL]
[-d]
[-s]
[-a SETALIAS]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-i | --clientid CLIENTID`

Optional

OAuth client ID (sometimes called the consumer key).

Type: string

`-r | --instanceurl INSTANCEURL`

Optional

The login URL of the instance the org lives on.

Type: url

`-d | --setdefaultdevhubusername`

Optional

Set the authenticated org as the default dev hub org for scratch org creation.

Type: boolean

`-s | --setdefaultusername`

Optional

Set the authenticated org as the default username that all commands run against.

Type: boolean

`-a | --setalias SETALIAS`

Optional

Set an alias for the authenticated org.

Type: string

Aliases for `auth:device:login`

```
force:auth:device:login
```

`jwt` Commands

Authorize an org using JWT.

auth:jwt:grant

Authorize an org using the JWT flow.

`auth:jwt:grant`

Authorize an org using the JWT flow.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org login jwt command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--jwtkeyfile`. New name: `--jwt-key-file`.
- Changed flag name: Old name `--clientid`. New name: `--client-id`.
- Changed flag name: Old name `--setdefaultdevhubusername`. New name: `--set-default-dev-hub`.
- Changed flag name: Old name `--setalias`. New name: `--alias`.
- Changed flag name: Old name `--username`. New name: Same, but with new short flag name`-o`.
- Changed flag name: Old name `--instanceurl`. New name: `--instance-url`.
- Changed flag name: Old name `--setdefaultusername`. New name: `--set-default`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `auth:jwt:grant`

Use a certificate associated with your private key that has been uploaded to a personal connected app.

If you specify an --instanceurl value, this value overrides the sfdcLoginUrl value in your sfdx-project.json file. To specify a My Domain URL, use the format MyDomainName.my.salesforce.com (not MyDomainName.lightning.force.com). To specify a sandbox, set --instanceurl to https://MyDomainName--SandboxName.sandbox.my.salesforce.com.

Examples for `auth:jwt:grant`

```
 sfdx auth:jwt:grant -u me@my.org -f <path to jwt key file> -i <OAuth client id>
```

```
 sfdx auth:jwt:grant -u me@my.org -f <path to jwt key file> -i <OAuth client id> -s -a
MyDefaultOrg
```

```
 sfdx auth:jwt:grant -u me@acme.org -f <path to jwt key file> -i <OAuth client id> -r
https://acme.my.salesforce.com
```

Command Syntax

`sfdx auth:jwt:grant`

```
[--json]
[--loglevel LOGLEVEL]
-u USERNAME
-f JWTKEYFILE
-i CLIENTID
[-r INSTANCEURL]
[-d]
[-s]
[-a SETALIAS]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --username USERNAME`

Required

Authentication username.

Type: string

`-f | --jwtkeyfile JWTKEYFILE`

Required

Path to a file containing the private key.

Type: filepath

-i | --clientid CLIENTID

Required

OAuth client ID (sometimes called the consumer key).

Type: string

-r | --instanceurl INSTANCEURL

Optional

The login URL of the instance the org lives on.

Type: url

-d | --setdefaultdevhubusername

Optional

Set the authenticated org as the default dev hub org for scratch org creation.

Type: boolean

-s | --setdefaultusername

Optional

Set the authenticated org as the default username that all commands run against.

Type: boolean

-a | --setalias SETALIAS

Optional

Set an alias for the authenticated org.

Type: string

Aliases for `auth:jwt:grant`

```
force:auth:jwt:grant
```

`sfdxurl` Commands

Authorize an org using sfdxurl.

auth:sfdxurl:store
Authorize an org using an SFDX auth URL stored within a file.

`auth:sfdxurl:store`

Authorize an org using an SFDX auth URL stored within a file.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org login sfdx-url command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

- Changed flag name: Old name `--setdefaultdevhubusername`. New name: `--set-default-dev-hub`.
- Changed flag name: Old name `--setalias`. New name: `--alias`.
- Changed flag name: Old name `--setdefaultusername`. New name: `--set-default`.
- Changed flag name: Old name `--sfdxurlfile`. New name: `--sfdx-url-file`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `auth:sfdxurl:store`

The SFDX auth URL must have the format "force://<clientId>:<clientSecret>:<refreshToken>@<instanceUrl>". NOTE: The SFDX auth URL uses the "force" protocol, and not "http" or "https". Also, the "instanceUrl" inside the SFDX auth URL doesn't include the protocol ("https://").

You have three options when creating the auth file. The easiest option is to redirect the output of the `sfdx force:org:display --verbose --json` command into a file. For example, using an org you have already authorized:

```
sfdx force:org:display -u <OrgUsername> --verbose --json > authFile.json
```

```
sfdx auth:sfdxurl:store -f authFile.json
```

The resulting JSON file contains the URL in the sfdxAuthUrl property inside of a results object. NOTE: The `force:org:display --verbose` command displays the refresh token only for orgs authorized with the web server flow, and not the JWT bearer flow.

You can also create a JSON file that has a top-level property named sfdxAuthUrl whose value is the auth URL. Finally, you can create a normal text file that includes just the URL and nothing else.

Examples for `auth:sfdxurl:store`

```
sfdx auth:sfdxurl:store -f <path to sfdxAuthUrl file>
```

```
sfdx auth:sfdxurl:store -f <path to sfdxAuthUrl file> -s -a MyDefaultOrg
```

Command Syntax

`sfdx auth:sfdxurl:store`

```
[--json]
[--loglevel LOGLEVEL]
-f SFDXURLFILE
[-d]
[-s]
[-a SETALIAS]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-f | --sfdxurlfile SFDXURLFILE`

Required

Path to a file containing the sfdx url.

Type: filepath

`-d | --setdefaultdevhubusername`

Optional

Set the authenticated org as the default dev hub org for scratch org creation.

Type: boolean

`-s | --setdefaultusername`

Optional

Set the authenticated org as the default username that all commands run against.

Type: boolean

`-a | --setalias SETALIAS`

Optional

Set an alias for the authenticated org.

Type: string

Aliases for `auth:sfdxurl:store`

```
force:auth:sfdxurl:store
```

`web` Commands

Authorize an org using a web browser.

auth:web:login

Authorize an org using the web login flow.

`auth:web:login`

Authorize an org using the web login flow.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org login web command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--clientid`. New name: `--client-id`.
- Changed flag name: Old name `--setdefaultdevhubusername`. New name: `--set-default-dev-hub`.
- Changed flag name: Old name `--setalias`. New name: `--alias`.
- Changed flag name: Old name `--instanceurl`. New name: `--instance-url`.
- Changed flag name: Old name `--setdefaultusername`. New name: `--set-default`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `auth:web:login`

If you specify an --instanceurl value, this value overrides the sfdcLoginUrl value in your sfdx-project.json file. To specify a My Domain URL, use the format MyDomainName.my.salesforce.com (not MyDomainName.lightning.force.com). To log in to a sandbox, set --instanceurl to https://MyDomainName--SandboxName.sandbox.my.salesforce.com.

To open in a specific browser, use the --browser parameter. Supported browsers are "chrome", "edge", and "firefox". If you don't specify --browser, the org opens in your default browser.

Examples for `auth:web:login`

```
sfdx auth:web:login -a TestOrg1
```

```
sfdx auth:web:login -i <OAuth client id>
```

```
sfdx auth:web:login -r https://MyDomainName--SandboxName.sandbox.my.salesforce.com
```

```
sfdx auth:web:login -a TestOrg1 -b firefox
```

Command Syntax

```
sfdx auth:web:login
   [--json]
   [--loglevel LOGLEVEL]
   [-b BROWSER]
   [-i CLIENTID]
   [-r INSTANCEURL]
   [-d]
   [-s]
   [-a SETALIAS]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-b | --browser BROWSER`

Optional

Browser where the org opens.

Type: enum

Permissible values are: chrome, edge, firefox

`-i | --clientid CLIENTID`

Optional

OAuth client ID (sometimes called the consumer key).

Type: string

`-r | --instanceurl INSTANCEURL`

Optional

The login URL of the instance the org lives on.

Type: url

`-d | --setdefaultdevhubusername`

Optional

Set the authenticated org as the default dev hub org for scratch org creation.

Type: boolean

`-s | --setdefaultusername`

Optional

Set the authenticated org as the default username that all commands run against.

Type: boolean

`-a | --setalias SETALIAS`

Optional

Set an alias for the authenticated org.

Type: string

Aliases for `auth:web:login`

```
force:auth:web:login
```

`config` Namespace

Use the config commands to view and set your Salesforce CLI configuration values. Set your default Dev Hub and scratch org, and your default instance URL, either globally or at the project level.

`config:get`

Get config var values for given names.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style config get command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 9, 2023.

Help for `config:get`

Gets the Salesforce CLI configuration values for your default scratch org, your default Dev Hub org, your default instance URL, or any combination of the three. To see your default scratch org username, include 'defaultusername'.

To see your default Dev Hub, include 'defaultdevhubusername'.

To see your default instance URL, include 'instanceUrl'.

To see the locations where your values are set, include the --verbose flag.

Examples for `config:get`

```
sfdx config:get defaultusername
```

```
sfdx config:get defaultusername defaultdevhubusername instanceUrl
```

```
sfdx config:get defaultusername defaultdevhubusername --verbose
```

Command Syntax

`sfdx config:get`

```
[--json]
[--loglevel LOGLEVEL]
[--verbose]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`--verbose`
Optional

Emit additional command output to stdout.

Type: boolean

Aliases for `config:get`

```
force:config:get
```

`config:list`

Lists the config variables that the Salesforce CLI uses for various commands and tasks.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style config list command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 9, 2023.

Command Syntax

`sfdx config:list`

```
[--json]
[--loglevel LOGLEVEL]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

Aliases for `config:list`

```
force:config:list
```

`config:set`

Sets the configuration variables that the Salesforce CLI uses for various commands and tasks.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style config set command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 9, 2023.

Help for `config:set`

Local variables apply only to your current project. Global variables apply in any directory.

Examples for `config:set`

```
sfdx config:set defaultusername=me@my.org defaultdevhubusername=me@myhub.org
```

```
sfdx config:set defaultdevhubusername=me@myhub.org -g
```

Command Syntax

`sfdx config:set`
`[--json]`
`[--loglevel LOGLEVEL]`
`[-g]`

Parameters

`--json`
Optional
Format output as JSON.
Type: boolean

`--loglevel LOGLEVEL`
Optional
The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.
Type: enum
Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL
Default value: warn

`-g | --global`
Optional
Sets the configuration variables globally, so they can be used from any directory.
Type: boolean

Aliases for `config:set`

```
force:config:set
```

`config:unset`

Unsets the local and global configuration variables for the Salesforce CLI.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style config unset command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 9, 2023.

Help for `config:unset`

Local variables apply only to your current project. Global variables apply in any directory.

Examples for `config:unset`

```
sfdx config:unset defaultusername defaultdevhubusername
```

```
sfdx config:unset defaultdevhubusername -g
```

Command Syntax

`sfdx config:unset`

```
[--json]
[--loglevel LOGLEVEL]
[-g]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-g` | `--global`

Optional

Unsets the configuration variables globally, so they can be used from any directory.

Type: boolean

Aliases for `config:unset`

```
force:config:unset
```

`doctor`

Gather CLI configuration data and run diagnostic tests to discover and report potential problems in your environment.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style doctor command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--createissue`. New name: `--create-issue`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 12, 2023.

Help for `doctor`

When you run the doctor command without parameters, it first displays a diagnostic overview of your environment. It then writes a detailed diagnosis to a JSON file in the current directory. Use the --outputdir to specify a different directory. To run diagnostic tests on a specific plugin, use the --plugin parameter. If the plugin isn't listening to the doctor, then you get a warning.

Use the --command parameter to run a specific command in debug mode; the doctor writes both stdout and stderr to *.log files that you can provide to Salesforce Customer Support or attach to a GitHub issue.

Plugin providers can also implement their own doctor diagnostic tests by listening to the "sf-doctor" event and running plugin specific tests that are then included in the doctor diagnostics log.

Examples for `doctor`

Run CLI doctor diagnostics:

```
sfdx doctor
```

Run CLI doctor diagnostics and the specified command, and write the debug output to a file:

```
sfdx doctor --command "force:org:list --all"
```

Run CLI doctor diagnostics for a specific plugin:

```
sfdx doctor --plugin @salesforce/plugin-source
```

Command Syntax

`sfdx doctor`

```
[--json]
[--loglevel LOGLEVEL]
[-c COMMAND]
[-p PLUGIN]
[-o OUTPUTDIR]
[-i]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-c | --command COMMAND`

Optional

Command to run in debug mode ; results are written to a log file.

Type: string

`-p | --plugin PLUGIN`

Optional

Specific plugin on which to run diagnostics .

Type: string

`-o | --outputdir OUTPUTDIR`

Optional

Directory to save all created files rather than the current working directory.

Type: directory

`-i | --createissue`

Optional

Create a new issue on our GitHub repo and attach all diagnostic results.

Type: boolean

`force` Namespace

Commands to develop on the Salesforce Platform.

analytics Commands

Work with analytics assets.

apex Commands

Use the apex commands to create Apex classes, execute anonymous blocks, view your logs, run Apex tests, and view Apex test results.

cmdt Commands

Create and update custom metadata types and their records.

community Commands

Use the community commands to create and publish an Experience Cloud site, and view a list of available templates in you org.

data Commands

Use the data commands to manipulate records in your org. Commands are available to help you work with various APIs. Import CSV files with the Bulk API. Export and import data with the SObject Tree Save API. Perform simple CRUD operations on individual records with the REST API.

lightning Commands

Use the lightning commands to create Aura components and Lightning web components. As of API version 45.0, you can build Lightning components using two programming models: Lightning Web Components, and the original model, Aura Components. Lightning web components and Aura components can coexist and interoperate on a page.

limits Commands

Display current org's limits.

mdapi Commands

Use the mdapi commands to retrieve and deploy Metadata API–formatted files that represent components in an org, or to convert Metadata API–formatted metadata into the source format used in Salesforce DX projects.

org Commands

Use the org commands to manage the orgs you use with Salesforce CLI. Create and delete scratch orgs, list your created and authorized orgs, and open orgs in your browser.

package Commands

Use the package commands to develop and install packages.

package1 Commands

Use the package1 commands to create and view first-generation package versions in your Dev Hub org.

project Commands

Use the project commands to set up a Salesforce DX project.

schema Commands

Use the schema commands to view information about the standard and custom objects in your org.

source Commands

Use the source commands to push and pull source to and from source-tracked orgs, to deploy and retrieve source to and from orgs, to see synchronization changes between your project and source-tracked orgs, and to convert your source to the metadata format for Metadata API deployments.

staticresource Commands

user Commands

Commands that perform user-related admin tasks.

visualforce Commands

Use the visualforce commands to create Visualforce pages and components.

`analytics` Commands

Work with analytics assets.

force:analytics:template:create

Creates a simple Analytics template in the specified directory. If you don't explicitly set the API version, it defaults to the current API version. The associated metadata files are created.

`force:analytics:template:create`

Creates a simple Analytics template in the specified directory. If you don't explicitly set the API version, it defaults to the current API version. The associated metadata files are created.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style analytics generate template command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--templatename`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:analytics:template:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
sfdx force:analytics:template:create -n myTemplate -d outputdir
```

Command Syntax

`sfdx force:analytics:template:create`

```
[--json]
[--loglevel LOGLEVEL]
[-d OUTPUTDIR]
[--apiversion APIVERSION]
-n TEMPLATENAME
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-n | --templatename TEMPLATENAME`

Required

The name of the Analytics template.

Type: string

`apex` Commands

Use the apex commands to create Apex classes, execute anonymous blocks, view your logs, run Apex tests, and view Apex test results.

force:apex:class:create

Creates an Apex class in the specified directory or the current working directory. If you don't explicitly set the API version, it defaults to the current API version. The .cls file and associated metadata file are created.

force:apex:execute

Executes one or more lines of anonymous Apex code entered on the command line, or executes the code in a local file.

force:apex:log:get

Fetches the specified log or given number of most recent logs from the scratch org. .

force:apex:log:list

Run this command in a project to list the IDs and general information for all debug logs in your default org.

force:apex:log:tail

Activates debug logging and displays logs in the terminal. You can also pipe the logs to a file.

force:apex:test:report

Provide a test run ID to display test results for an enqueued or completed asynchronous test run. The test run ID is displayed after running the "sfdx force:apex:test:run" command.

force:apex:test:run

Specify which tests to run by using the --classnames, --suites, or --tests parameters. Alternatively, use the --testlevel parameter to run all the tests in your org, local tests, or specified tests.

force:apex:trigger:create

Creates an Apex trigger in the specified directory or the current working directory. If you don't explicitly set the API version, it defaults to the current API version. The .trigger file and associated metadata file are created.

`force:apex:class:create`

Creates an Apex class in the specified directory or the current working directory. If you don't explicitly set the API version, it defaults to the current API version. The .cls file and associated metadata file are created.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style apex generate class command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--classname`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:apex:class:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
sfdx force:apex:class:create -n MyClass
```

```
sfdx force:apex:class:create -n MyClass -d classes
```

Command Syntax

`sfdx force:apex:class:create`

```
[--json]
[--loglevel LOGLEVEL]
-n CLASSNAME
[-t TEMPLATE]
[-d OUTPUTDIR]
[--apiversion APIVERSION]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-n | --classname CLASSNAME`

Required

The name of the new Apex class. The name can be up to 40 characters and must start with a letter.

Type: string

`-t | --template TEMPLATE`

Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: ApexException, ApexUnitTest, DefaultApexClass, InboundEmailService

Default value: DefaultApexClass

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`force:apex:execute`

Executes one or more lines of anonymous Apex code entered on the command line, or executes the code in a local file.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style apex run command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--apexcodefile`. New name: `--file`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `force:apex:execute`

If you don't run this command from within a Salesforce DX project, —-targetusername is required.

To execute your code interactively, run this command with no parameters. At the prompt, enter all your Apex code; press CTRL-D when you're finished. Your code is then executed in a single execute anonymous request.

For more information, see "Anonymous Blocks" in the Apex Developer Guide.

Examples for `force:apex:execute`

```
sfdx force:apex:execute -u testusername@salesforce.org -f ~/test.apex
```

```
sfdx force:apex:execute -f ~/test.apex
```

```
 sfdx force:apex:execute
Start typing Apex code. Press the Enter key after each line, then press CTRL+D when finished.
```

Command Syntax

`sfdx force:apex:execute`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`[-f APEXCODEFILE]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-f | --apexcodefile APEXCODEFILE`

Optional

Path to a local file that contains Apex code.

Type: filepath

`force:apex:log:get`

Fetches the specified log or given number of most recent logs from the scratch org. .

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style apex get log command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--logid`. New name: `--log-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `force:apex:log:get`

To get the IDs for your debug logs, run "sfdx force:apex:log:list".

Use the --logid parameter to return a specific log.

Use the --number parameter to return the specified number of recent logs.

Use the --outputdir parameter to specify the directory to store the logs in.

Executing this command without parameters returns the most recent log.

Examples for `force:apex:log:get`

```
sfdx force:apex:log:get -i <log id>
```

```
sfdx force:apex:log:get -i <log id> -u me@my.org
```

```
sfdx force:apex:log:get -n 2 -c
```

```
sfdx force:apex:log:get -d Users/Desktop/logs -n 2
```

Command Syntax

```
sfdx force:apex:log:get
   [--json]
   [--loglevel LOGLEVEL]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   [-i LOGID]
   [-n NUMBER]
   [-d OUTPUTDIR]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-i | --logid LOGID`

Optional

Id of the log to display.

Type: id

`-n | --number NUMBER`

Optional

Number of most recent logs to display.

Type: number

`-d | --outputdir OUTPUTDIR`

Optional

The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

`force:apex:log:list`

Run this command in a project to list the IDs and general information for all debug logs in your default org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style apex list log command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `force:apex:log:list`

To fetch a specific log from your org, obtain the ID from this command's output, then run the "sfdx force:apex:log:get" command.

Examples for `force:apex:log:list`

```
sfdx force:apex:log:list
```

```
sfdx force:apex:log:list -u me@my.org
```

Command Syntax

`sfdx force:apex:log:list`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`force:apex:log:tail`

Activates debug logging and displays logs in the terminal. You can also pipe the logs to a file.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style apex tail log command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--debuglevel`. New name: `--debug-level`.
- Changed flag name: Old name `--skiptraceflag`. New name: `--skip-trace-flag`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Examples for `force:apex:log:tail`

```
sfdx force:apex:log:tail
```

```
sfdx force:apex:log:tail --debuglevel MyDebugLevel
```

```
sfdx force:apex:log:tail -c -s
```

Command Syntax

`sfdx force:apex:log:tail`

`[--json]`

`[--loglevel LOGLEVEL]`

```
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-c]
[-d DEBUGLEVEL]
[-s]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sfdx/sfdx.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-c | --color`
Optional

Applies default colors to noteworthy log lines.

Type: boolean

`-d | --debuglevel DEBUGLEVEL`
Optional

Debug level to set on the DEVELOPER_LOG trace flag for your user.

Type: string

`-s | --skiptraceflag`
Optional

Skips trace flag setup. Assumes that a trace flag and debug level are fully set up.

Type: boolean

`force:apex:test:report`

Provide a test run ID to display test results for an enqueued or completed asynchronous test run. The test run ID is displayed after running the "sfdx force:apex:test:run" command.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style apex get test command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--codecoverage`. New name: `--code-coverage`.
- Changed flag name: Old name `--testrunid`. New name: `--test-run-id`.
- Changed flag name: Old name `--resultformat`. New name: `--result-format`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Examples for `force:apex:test:report`

```
sfdx force:apex:test:report -i <test run id>
```

```
sfdx force:apex:test:report -i <test run id> -r junit
```

```
sfdx force:apex:test:report -i <test run id> -c --json
```

```
sfdx force:apex:test:report -i <test run id> -c -d <path to outputdir> -u me@myorg
```

Command Syntax

`sfdx force:apex:test:report`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-i TESTRUNID
[-c]
[-d OUTPUTDIR]
[-r RESULTFORMAT]
[-w WAIT]
[--verbose]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-i | --testrunid TESTRUNID`
Required

The ID of the test run.

Type: string

`-c | --codecoverage`
Optional

Retrieves code coverage results.

Type: boolean

`-d | --outputdir OUTPUTDIR`
Optional

Directory to store test result files.

Type: string

`-r | --resultformat RESULTFORMAT`
Optional

Permissible values are: human, tap, junit, json.

Type: enum

Permissible values are: human, tap, junit, json

`-w | --wait WAIT`
Optional

Sets the streaming client socket timeout in minutes; specify a longer wait time if timeouts occur frequently.

Type: string

`--verbose`

Optional

Emit additional command output to stdout.

Type: boolean

`force:apex:test:run`

Specify which tests to run by using the --classnames, --suites, or --tests parameters. Alternatively, use the --testlevel parameter to run all the tests in your org, local tests, or specified tests.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style apex run test command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--codecoverage`. New name: `--code-coverage`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--apexcodefile`. New name: `--file`.
- Changed flag name: Old name `--testlevel`. New name: `--test-level`.
- Changed flag name: Old name `--classnames`. New name: `--class-names`.
- Changed flag name: Old name `--resultformat`. New name: `--result-format`.
- Changed flag name: Old name `--suitenames`. New name: `--suite-names`.
- Changed flag name: Old name `--detailedcoverage`. New name: `--detailed-coverage`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 2, 2023.

Help for `force:apex:test:run`

To see code coverage results, use the --codecoverage parameter with --resultformat. The output displays a high-level summary of the test run and the code coverage values for classes in your org. If you specify human-readable result format, use the --detailedcoverage parameter to see detailed coverage results for each test method run.

NOTE: The testRunCoverage value (JSON and JUnit result formats) is a percentage of the covered lines and total lines from all the Apex classes evaluated by the tests in this run.

Examples for **`force:apex:test:run`**

```
sfdx force:apex:test:run
```

```
sfdx force:apex:test:run -n "MyClassTest,MyOtherClassTest" -r human
```

```
sfdx force:apex:test:run -s "MySuite,MyOtherSuite" -c -v --json
```

```
sfdx force:apex:test:run -t
"MyClassTest.testCoolFeature,MyClassTest.testAwesomeFeature,AnotherClassTest,namespace.TheirClassTest.testThis"
 -r human
```

```
sfdx force:apex:test:run -l RunLocalTests -d <path to outputdir> -u me@my.org
```

Command Syntax

`sfdx force:apex:test:run`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-c]
[-d OUTPUTDIR]
[-l TESTLEVEL]
[-n CLASSNAMES]
[-r RESULTFORMAT]
[-s SUITENAMES]
[-t TESTS]
[-w WAIT]
[-y]
[--verbose]
[-v]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-c | --codecoverage`

Optional

Retrieves code coverage results.

Type: boolean

`-d | --outputdir OUTPUTDIR`

Optional

Directory to store test run files.

Type: string

`-l | --testlevel TESTLEVEL`

Optional

Specifies which tests to run, using one of these TestLevel enum values:.

Type: enum

Permissible values are: RunLocalTests, RunAllTestsInOrg, RunSpecifiedTests

`-n | --classnames CLASSNAMES`

Optional

Comma-separated list of Apex test class names to run; if you select --classnames, you can't specify --suitenames or --tests.

Type: string

`-r | --resultformat RESULTFORMAT`

Optional

Permissible values are: human, tap, junit, json.

Type: enum

Permissible values are: human, tap, junit, json

`-s | --suitenames SUITENAMES`

Optional

Comma-separated list of Apex test suite names to run; if you select --suitenames, you can't specify --classnames or --tests.

Type: string

`-t | --tests TESTS`

Optional

Comma-separated list of Apex test class names or IDs and, if applicable, test methods to run; if you specify --tests, you can't specify --classnames or --suitenames.

Type: string

`-w | --wait WAIT`

Optional

Sets the streaming client socket timeout in minutes; specify a longer wait time if timeouts occur frequently.

Type: string

`-y | --synchronous`

Optional

Runs test methods from a single Apex class synchronously; if not specified, tests are run ansynchronously.

Type: boolean

`--verbose`

Optional

Emit additional command output to stdout.

Type: boolean

`-v | --detailedcoverage`

Optional

Display detailed code coverage per test.

Type: boolean

`force:apex:trigger:create`

Creates an Apex trigger in the specified directory or the current working directory. If you don't explicitly set the API version, it defaults to the current API version. The .trigger file and associated metadata file are created.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style apex generate trigger command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--triggername`. New name: `--name`.
- Changed flag name: Old name `--triggerevents`. New name: `--event`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:apex:trigger:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
sfdx force:apex:trigger:create -n MyTrigger
```

```
sfdx force:apex:trigger:create -n MyTrigger -s Account -e 'before insert,after insert'
```

```
sfdx force:apex:trigger:create -n MyTrigger -d triggers
```

Command Syntax

`sfdx force:apex:trigger:create`

`[--json]`

`[--loglevel LOGLEVEL]`

`-n TRIGGERNAME`

`[-t TEMPLATE]`

`[-d OUTPUTDIR]`

`[--apiversion APIVERSION]`

`[-s SOBJECT]`

`[-e TRIGGEREVENTS]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-n | --triggername TRIGGERNAME`

Required

The name of the new Apex trigger. The name can be up to 40 characters and must start with a letter.

Type: string

`-t | --template TEMPLATE`

Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: ApexTrigger

Default value: ApexTrigger

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-s | --sobject SOBJECT`

Optional

The sObject to create an Apex trigger on.

Type: string

Default value: SOBJECT

`-e | --triggerevents TRIGGEREVENTS`

Optional

The events that cause the trigger to fire.

Type: array

Default value: before insert

`cmdt` Commands

Create and update custom metadata types and their records.

force:cmdt:create

Creates a new custom metadata type in the current project.

force:cmdt:field:create

Generate a custom metadata field based on the field type provided.

force:cmdt:generate

Generates a custom metadata type and all its records for the provided sObject.

force:cmdt:record:create

Create a new record for a given custom metadata type in the current project.

force:cmdt:record:insert

Create new custom metadata type records from a CSV file.

`force:cmdt:create`

Creates a new custom metadata type in the current project.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style cmdt generate object command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-directory`.
- Changed flag name: Old name `--plurallabel`. New name: `--plural-label`.
- Changed flag name: Old name `--typename`. New name: `--type-name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Examples for `force:cmdt:create`

Create a custom metadata type with developer name 'MyCustomType'; this name will also be used as the label:

```
sfdx force:cmdt:create --typename MyCustomType
```

Create a protected custom metadata type with a specific label:

```
sfdx force:cmdt:create --typename MyCustomType --label "Custom Type" --plurallabel "Custom
Types" --visibility Protected
```

Command Syntax

`sfdx force:cmdt:create`

```
[--json]
[--loglevel LOGLEVEL]
-n TYPENAME
[-l LABEL]
[-p PLURALLABEL]
[-v VISIBILITY]
[-d OUTPUTDIR]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-n | --typename TYPENAME`

Required

The unique name of the object in the API. This name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Type: string

`-l | --label LABEL`

Optional

A label for the custom metadata type.

Type: string

`-p | --plurallabel PLURALLABEL`

Optional

The plural version of the label value. If this flag is missing or blank, the singular label is used as the plural label.

Type: string

`-v | --visibility VISIBILITY`

Optional

The visibility of the custom metadata type.

Type: enum

Permissible values are: PackageProtected, Protected, Public

Default value: Public

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store the newly-created custom metadata type files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: directory

`force:cmdt:field:create`

Generate a custom metadata field based on the field type provided.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style cmdt generate field command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--decimalplaces`. New name: `--decimal-places`.
- Changed flag name: Old name `--fieldname`. New name: `--field-name`.

- Changed flag name: Old name `--fieldtype`. New name: `--field-type`.
- Changed flag name: Old name `--outputdir`. New name: `--output-directory`.
- Changed flag name: Old name `--picklistvalues`. New name: `--picklist-values`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Examples for `force:cmdt:field:create`

Create a metadata file for a custom checkbox field:

```
sfdx force:cmdt:field:create --fieldname MyField --fieldtype Checkbox
```

Create a metadata file for a custom picklist field:

```
sfdx force:cmdt:field:create --fieldname MyField --fieldtype Picklist --picklistvalues
"A,B,C"
```

Create a metadata file for a custom number field:

```
sfdx force:cmdt:field:create --fieldname MyField --fieldtype Number --decimalplaces 2
```

Command Syntax

`sfdx force:cmdt:field:create`

```
[--json]
[--loglevel LOGLEVEL]
-n FIELDNAME
-f FIELDTYPE
[-p PICKLISTVALUES]
[-s DECIMALPLACES]
[-l LABEL]
[-d OUTPUTDIR]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

-n | --fieldname FIELDNAME

Required

The unique name for the field.

Type: string

-f | --fieldtype FIELDTYPE

Required

The unique name for the field.

Type: enum

Permissible values are: Checkbox, Date, DateTime, Email, Number, Percent, Phone, Picklist, Text, TextArea, LongTextArea, Url

-p | --picklistvalues PICKLISTVALUES

Optional

A comma-separated list of picklist values. These values are required when creating a Picklist field.

Type: array

-s | --decimalplaces DECIMALPLACES

Optional

The number of decimal places to use for Number or Percent fields. The value must be greater than or equal to zero.

Type: number

-l | --label LABEL

Optional

The label for the field.

Type: string

-d | --outputdir OUTPUTDIR

Optional

The directory to store the newly-created field definition files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: directory

force:cmdt:generate

Generates a custom metadata type and all its records for the provided sObject.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style cmdt generate fromorg command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--devname`. New name: `--dev-name`.
- Changed flag name: Old name `--ignoreunsupported`. New name: `--ignore-unsupported`.

- Changed flag name: Old name `--plurallabel`. New name: `--plural-label`.
- Changed flag name: Old name `--recordsoutputdir`. New name: `--records-output-dir`.
- Changed flag name: Old name `--sobjectname`. New name: `--sobject`.
- Changed flag name: Old name `--typeoutputdir`. New name: `--type-output-directory`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Examples for `force:cmdt:generate`

Generate a custom metadata type from an sObject in the default target org:

```
sfdx force:cmdt:generate --devname MyCMDT --sobjectname MySourceObject__c
```

Generate a custom metadata type from an sObject in the specified target org; ignore unsupported field types instead of converting them to text:

```
sfdx force:cmdt:generate --devname MyCMDT --sobjectname MySourceObject__c
--ignoreunsupported --targetusername 'alias or user email of the org containing the source
type'
```

Generate a protected custom metadata type from an sObject in the default target org:

```
sfdx force:cmdt:generate --devname MyCMDT --sobjectname SourceCustomObject__c  --visibility
Protected
```

Generate a protected custom metadata type with a specific label from an sObject in the default target org:

```
sfdx force:cmdt:generate --devname MyCMDT --label "My CMDT" --plurallabel "My CMDTs"
--sobjectname SourceCustomSetting__c  --visibility Protected
```

Generate a custom metadata type from an sObject in the default target org; put the resulting type metadata file in the specified directory:

```
sfdx force:cmdt:generate --devname MyCMDT --sobjectname SourceCustomSetting__c
--typeoutputdir 'path/to/my/cmdt/directory'
```

Generate a custom metadata type from an sObject in the default target org; put the resulting record metadata file(s) in the specified directory:

```
sfdx force:cmdt:generate --devname MyCMDT --sobjectname SourceCustomSetting__c
--recordsoutputdir 'path/to/my/cmdt/record/directory'
```

Command Syntax

`sfdx force:cmdt:generate`

```
   [--json]
   [--loglevel LOGLEVEL]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   -n DEVNAME
   [-l LABEL]
   [-p PLURALLABEL]
```

```
[-v VISIBILITY]
-s SOBJECTNAME
[-i]
[-d TYPEOUTPUTDIR]
[-r RECORDSOUTPUTDIR]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-n | --devname DEVNAME`
Required

The name of the custom metadata type.

Type: string

`-l | --label LABEL`
Optional

The label for the custom metadata type.

Type: string

`-p | --plurallabel PLURALLABEL`
Optional

The plural version of the label value. If this flag is missing or blank, the singular label is used as the plural label.

Type: string

`-v | --visibility VISIBILITY`
Optional

The visibility of the custom metadata type.

Type: enum

Permissible values are: PackageProtected, Protected, Public

Default value: Public

`-s | --sobjectname SOBJECTNAME`

Required

The API name of the sObject source for custom metadata generation.

Type: string

`-i | --ignoreunsupported`

Optional

Ignore unsupported field types (these fields will not be created). The default is to create Text fields and convert the source value to text.

Type: boolean

`-d | --typeoutputdir TYPEOUTPUTDIR`

Optional

The directory to store newly-created custom metadata type files.

Type: directory

Default value: force-app/main/default/objects

`-r | --recordsoutputdir RECORDSOUTPUTDIR`

Optional

The directory to store newly-created custom metadata record files.

Type: directory

Default value: force-app/main/default/customMetadata

`force:cmdt:record:create`

Create a new record for a given custom metadata type in the current project.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style cmdt generate record command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--inputdir`. New name: `--input-directory`.
- Changed flag name: Old name `--outputdir`. New name: `--output-directory`.
- Changed flag name: Old name `--recordname`. New name: `--record-name`.
- Changed flag name: Old name `--typename`. New name: `--type-name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Examples for **`force:cmdt:record:create`**

Create a record metadata file for custom metadata type 'MyCMT' with values specified for two custom fields:

```
 sfdx force:cmdt:record:create --typename MyCMT__mdt --recordname MyRecord
My_Custom_Field_1=Foo My_Custom_Field_2=Bar
```

Create a protected record metadata file for custom metadata type 'MyCMT' with a specific label and values specified for two custom fields:

```
 sfdx force:cmdt:record:create --typename MyCMT__mdt --recordname MyRecord --label "My
Record" --protected true My_Custom_Field_1=Foo My_Custom_Field_2=Bar
```

Command Syntax

`sfdx force:cmdt:record:create`

`[--json]`

`[--loglevel LOGLEVEL]`

`-t TYPENAME`

`-n RECORDNAME`

`[-l LABEL]`

`[-p PROTECTED]`

`[-i INPUTDIR]`

`[-d OUTPUTDIR]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-t | --typename TYPENAME`

Required

The API name of the custom metadata type to create a record for.

Type: string

`-n | --recordname RECORDNAME`

Required

The name for the new record.

Type: string

`-l | --label LABEL`

Optional

The label for the new record.

Type: string

`-p | --protected PROTECTED`

Optional

Protect the record when it is in a managed package. Protected records can only be accessed by code in the same managed package namespace.

Type: string

Permissible values are: true, false

Default value: false

`-i | --inputdir INPUTDIR`

Optional

The directory to pull the custom metadata type definition from.

Type: directory

Default value: force-app/main/default/objects

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store newly-created custom metadata record files.

Type: directory

Default value: force-app/main/default/customMetadata

`force:cmdt:record:insert`

Create new custom metadata type records from a CSV file.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style cmdt generate records command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--filepath`. New name: `--csv`.
- Changed flag name: Old name `--inputdir`. New name: `--input-directory`.
- Changed flag name: Old name `--namecolumn`. New name: `--name-column`.
- Changed flag name: Old name `--outputdir`. New name: `--output-directory`.
- Changed flag name: Old name `--typename`. New name: `--type-name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Examples for `force:cmdt:record:insert`

Create record metadata files for type 'My_CMDT_Name' (from your local project) based on values in a CSV file, using 'Name' as the column that specifies the record name:

```
sfdx force:cmdt:record:insert --filepath path/to/my.csv --typename My_CMDT_Name
```

Create record metadata files for type 'My_CMDT_Name' (from the specified directory) based on values in a CSV file, using 'PrimaryKey' as the column that specifies the record name:

```
sfdx force:cmdt:record:insert --filepath path/to/my.csv --typename My_CMDT_Name --inputdir
"path/to/my/cmdt/directory" --namecolumn "PrimaryKey"
```

Command Syntax

`sfdx force:cmdt:record:insert`

```
[--json]
[--loglevel LOGLEVEL]
-f FILEPATH
-t TYPENAME
[-i INPUTDIR]
[-d OUTPUTDIR]
[-n NAMECOLUMN]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-f | --filepath FILEPATH`

Required

The path to the CSV file.

Type: string

`-t | --typename TYPENAME`

Required

The API Name of the custom metadata type. The '__mdt' suffix will be appended to the end of the name if it is omitted.

Type: string

`-i | --inputdir INPUTDIR`

Optional

The directory to pull the custom metadata type definition from.

Type: directory

Default value: force-app/main/default/objects

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store newly-created custom metadata record files.

Type: directory

Default value: force-app/main/default/customMetadata

`-n | --namecolumn NAMECOLUMN`

Optional

The column that is used to determine the name of the record.

Type: string

Default value: Name

`community` Commands

Use the community commands to create and publish an Experience Cloud site, and view a list of available templates in you org.

force:community:create

Creates an Experience Cloud site using a template.

force:community:publish

Publishes an Experience Builder site to make it live.

force:community:template:list

Retrieves the list of templates available in your org.

`force:community:create`

Creates an Experience Cloud site using a template.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style community create command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--templatename`. New name: `--template-name`.
- Changed flag name: Old name `--urlpathprefix`. New name: `--url-path-prefix`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:community:create`

See 'Which Experience Cloud Template Should I Use?' in Salesforce Help for more information about the different template types available for Experience Cloud.

When creating a site with the Build Your Own (LWR) template, you must also specify the AuthenticationType value using the format templateParams.AuthenticationType=value, where value is AUTHENTICATED, UNAUTHENTICATED, or AUTHENTICATED_WITH_PUBLIC_ACCESS. Name and values are case-sensitive. See 'ExperienceBundle' in the Metadata API Developer Guide for more information.

When you execute this command, it creates the site in preview status, which means that it isn't yet live. After you finish building your site, you can make it live.

If you have an Experience Builder site, publish the site using the sfdx force:community:publish command to make it live.

If you have a Salesforce Tabs + Visualforce site, activate the site to make it live by updating the status field of the Network type in the Metadata API. Alternatively, in Experience Workspaces, go to Administration | Settings, and click Activate.

For Experience Builder sites, activating the site just sends out a welcome email to site members.

Examples for `force:community:create`

```
 sfdx force:community:create --name 'My Customer Site' --templatename 'Customer Service'
--urlpathprefix customers --description 'My customer site'
```

```
 sfdx force:community:create -n partnercentral -t 'Partner Central' -p partners
```

```
 sfdx force:community:create -n lwrsite -t 'Build Your Own (LWR)' -p lwrsite
templateParams.AuthenticationType=UNAUTHENTICATED
```

Command Syntax

`sfdx force:community:create`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-n NAME
-t TEMPLATENAME
-p URLPATHPREFIX
[-d DESCRIPTION]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-n | --name NAME`

Required

The name of the site to create.

Type: string

`-t | --templatename TEMPLATENAME`

Required

The template to use to create the site, such as the Customer Service template. Run force:community:template:list to see which templates are available in your org.

Type: string

`-p | --urlpathprefix URLPATHPREFIX`

Required

The URL to append to the domain that you created when you enabled Digital Experiences for this org. For example, if your domain name is https://MyDomainName.my.site.com and you're creating a customer site, enter 'customers' to create the unique URL

https://MyDomainName.my.site.com/customers.

Type: string

`-d | --description DESCRIPTION`

Optional

The description of the site. The description displays in Digital Experiences - All Sites in Setup and helps with site identification.

Type: string

`force:community:publish`

Publishes an Experience Builder site to make it live.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style community publish command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:community:publish`

Each time you publish it, you update the live site with the most recent updates.

When you publish an Experience Builder site for the first time, you make the site's URL live and enable login access for site members.

Additionally, to send a welcome email to all site members, you must activate the site. (Activation is also required to successfully set up SEO for Experience Builder sites.) To activate a site, update the status field of the Network type in the Metadata API. Alternatively, in Experience Workspaces, go to Administration | Settings, and click Activate.

Subsequently, each time you publish the site, you update the live site with all changes made to the site since it was last published.

An email notification informs you when your changes are live.

Examples for `force:community:publish`

```
sfdx force:community:publish --name 'My Customer Site'
```

Command Syntax

`sfdx force:community:publish`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-n NAME
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-n | --name NAME`

Required

The name of the Experience Builder site that you want to publish.

Type: string

`force:community:template:list`

Retrieves the list of templates available in your org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style community list template command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:community:template:list`

See 'Which Experience Cloud Template Should I Use?' in Salesforce Help for more information about the different template types available for Experience Cloud.

Examples for `force:community:template:list`

```
sfdx force:community:template:list
```

Command Syntax

`sfdx force:community:template:list`

`[--json]`

```
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
```

Parameters

`--json`
: Optional

  Format output as JSON.

  Type: boolean

`--loglevel LOGLEVEL`
: Optional

  The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

  Type: enum

  Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

  Default value: warn

`-u | --targetusername TARGETUSERNAME`
: Optional

  A username or alias for the target org. Overrides the default target org.

  Type: string

`--apiversion APIVERSION`
: Optional

  Override the API version used for API requests made by this command.

  Type: string

`data` Commands

Use the data commands to manipulate records in your org. Commands are available to help you work with various APIs. Import CSV files with the Bulk API. Export and import data with the SObject Tree Save API. Perform simple CRUD operations on individual records with the REST API.

force:data:bulk:delete
Bulk delete records from a csv file.

force:data:bulk:status
View the status of a bulk data load job or batch.

force:data:bulk:upsert
Bulk upsert records from a CSV file.

force:data:record:create
Creates and inserts a record.

force:data:record:delete
Deletes a single record.

force:data:record:get
Displays a single record.

force:data:record:update
Updates a single record.

force:data:soql:bulk:report
View the status of a bulk query.

force:data:soql:query
Execute a SOQL query.

force:data:tree:export
Export data from an org.

force:data:tree:import
Import data into an org.

`force:data:bulk:delete`

Bulk delete records from a csv file.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using one of these equivalent `sf`-style commmands instead, depending on whether you want to use version 1.0 or 2.0 of Bulk API:

- force data bulk delete (Bulk API 1.0)
- data delete bulk (Bulk API 2.0)

Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjecttype`. New name: `--sobject`.
- Changed flag name: Old name `--csvfile`. New name: `--file`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:data:bulk:delete`

The file must be a CSV file with only one column: "Id".

One job can contain many batches, depending on the length of the CSV file.

Returns a job ID and a batch ID. Use these IDs to check job status with data:bulk:status.

Examples for `force:data:bulk:delete`

```
sfdx force:data:bulk:delete -s Account -f ./path/to/file.csv
```

```
sfdx force:data:bulk:delete -s MyObject__c -f ./path/to/file.csv
```

Command Syntax

`sfdx force:data:bulk:delete`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-f CSVFILE
-s SOBJECTTYPE
[-w WAIT]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-f | --csvfile CSVFILE`

Required

The path to the CSV file containing the ids of the records to delete.

Type: filepath

`-s | --sobjecttype SOBJECTTYPE`
Required

The sObject type of the records you're deleting.

Type: string

`-w | --wait WAIT`
Optional

The number of minutes to wait for the command to complete before displaying the results.

Type: minutes

`force:data:bulk:status`

View the status of a bulk data load job or batch.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using one of these equivalent `sf`-style commmands instead, depending on whether you want to use version 1.0 or 2.0 of Bulk API:

- force data bulk status (Bulk API 1.0)
- data delete resume (Bulk API 2.0)
- data upsert resume (Bulk API 2.0)

Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--batchid`. New name: `--batch-id`.
- Changed flag name: Old name `--jobid`. New name: `--job-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:data:bulk:status`

Run this command using the job ID or batch ID returned from the force:data:bulk:delete or force:data:bulk:upsert commands.

Examples for `force:data:bulk:status`

```
sfdx force:data:bulk:status -i 750xx000000005sAAA
```

```
sfdx force:data:bulk:status -i 750xx000000005sAAA -b 751xx000000005nAAA
```

Command Syntax

```
sfdx force:data:bulk:status
   [--json]
   [--loglevel LOGLEVEL]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   [-b BATCHID]
   -i JOBID
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-b | --batchid BATCHID`
Optional

The ID of the batch whose status you want to view.

Type: string

`-i | --jobid JOBID`
Required

The ID of the job you want to view or of the job whose batch you want to view.

Type: string

force:data:bulk:upsert

Bulk upsert records from a CSV file.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using one of these equivalent `sf`-style commmands instead, depending on whether you want to use version 1.0 or 2.0 of Bulk API:

- force data bulk upsert (Bulk API 1.0)
- data upsert bulk (Bulk API 2.0)

Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjecttype`. New name: `--sobject`.
- Changed flag name: Old name `--csvfile`. New name: `--file`.
- Changed flag name: Old name `--externalid`. New name: `--external-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for **`force:data:bulk:upsert`**

Inserts or updates records from a CSV file.

One job can contain many batches, depending on the length of the CSV file.

Returns a job ID and a batch ID. Use these IDs to check job status with data:bulk:status.

For information about formatting your CSV file, see "Prepare CSV Files" in the Bulk API Developer Guide.

By default, the job runs the batches in parallel. Specify --serial to run them serially.

Examples for **`force:data:bulk:upsert`**

```
sfdx force:data:bulk:upsert -s MyObject__c -f ./path/to/file.csv -i MyField__c
```

```
sfdx force:data:bulk:upsert -s MyObject__c -f ./path/to/file.csv -i Id -w 2
```

Command Syntax

`sfdx force:data:bulk:upsert`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-i EXTERNALID
-f CSVFILE
-s SOBJECTTYPE
```

```
[-w WAIT]
[-r]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-i | --externalid EXTERNALID`
Required

The column name of the external ID.

Type: string

`-f | --csvfile CSVFILE`
Required

The path to the CSV file that defines the records to upsert.

Type: filepath

`-s | --sobjecttype SOBJECTTYPE`
Required

The sObject type of the records you want to upsert.

Type: string

`-w | --wait WAIT`
Optional

The number of minutes to wait for the command to complete before displaying the results.

Type: minutes

-r | --serial

Optional

Run batches in serial mode.

Type: boolean

force:data:record:create

Creates and inserts a record.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style data create record command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Removed flag: `--perflog`
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjectype`. New name: `--sobject`.
- Changed flag name: Old name `--usetoolingapi`. New name: `--use-tooling-api`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for force:data:record:create

The format of a field-value pair is <fieldName>=<value>.

Enclose all field-value pairs in one set of double quotation marks, delimited by spaces.

Enclose values that contain spaces in single quotes.

To get data on API performance metrics, specify both --perflog and --json.

Examples for force:data:record:create

```
sfdx force:data:record:create -s Account -v "Name=Acme"
```

```
sfdx force:data:record:create -s Account -v "Name='Universal Containers'"
```

```
sfdx force:data:record:create -s Account -v "Name='Universal Containers'
Website=www.example.com"
```

```
sfdx force:data:record:create -t -s TraceFlag -v "DebugLevelId=7dl170000008U36AAE
StartDate=2017-12-01T00:26:04.000+0000 ExpirationDate=2017-12-01T00:56:04.000+0000
LogType=CLASS_TRACING TracedEntityId=01p17000000R6bLAAS"
```

```
sfdx force:data:record:create -s Account -v "Name=Acme" --perflog --json
```

Command Syntax

`sfdx force:data:record:create`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-s SOBJECTTYPE
-v VALUES
[-t]
[--perflog]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-s | --sobjecttype SOBJECTTYPE`

Required

The type of the record you’re creating.

Type: string

`-v | --values VALUES`

Required

The <fieldName>=<value> pairs you’re creating.

Type: string

`-t | --usetoolingapi`
Optional

Create the record with tooling api.

Type: boolean

`--perflog`
Optional

Get API performance data.

Type: boolean

`force:data:record:delete`

Deletes a single record.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style data delete record command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjectype`. New name: `--sobject`.
- Changed flag name: Old name `--usetoolingapi`. New name: `--use-tooling-api`.
- Changed flag name: Old name `--sobjectid`. New name: `--record-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:data:record:delete`

Specify an sObject type and either an ID or a list of <fieldName>=<value> pairs.

The format of a field-value pair is <fieldName>=<value>.

Enclose all field-value pairs in one set of double quotation marks, delimited by spaces.

Enclose values that contain spaces in single quotes.

To get data on API performance metrics, specify both --perflog and --json.

Examples for `force:data:record:delete`

```
sfdx force:data:record:delete -s Account -i 001D000000Kv3dl
```

```
sfdx force:data:record:delete -s Account -w "Name=Acme"
```

```
sfdx force:data:record:delete -s Account -w "Name='Universal Containers'"
```

```
sfdx force:data:record:delete -s Account -w "Name='Universal Containers' Phone='(123)
456-7890'"
```

```
sfdx force:data:record:delete -t -s TraceFlag -i 7tf170000009cU6AAI --perflog --json
```

Command Syntax

`sfdx force:data:record:delete`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`-s SOBJECTTYPE`

`[-i SOBJECTID]`

`[-w WHERE]`

`[-t]`

`[--perflog]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-s | --sobjecttype SOBJECTTYPE`

Required

The type of the record you're deleting.

Type: string

`-i | --sobjectid SOBJECTID`

Optional

The ID of the record you're deleting.

Type: id

`-w | --where WHERE`

Optional

A list of <fieldName>=<value> pairs to search for.

Type: string

`-t | --usetoolingapi`

Optional

Delete the record with Tooling API.

Type: boolean

`--perflog`

Optional

Get API performance data.

Type: boolean

`force:data:record:get`

Displays a single record.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style data get record command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjecttype`. New name: `--sobject`.
- Changed flag name: Old name `--usetoolingapi`. New name: `--use-tooling-api`.
- Changed flag name: Old name `--sobjectid`. New name: `--record-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:data:record:get`

Specify an sObject type and either an ID or a list of <fieldName>=<value> pairs.

The format of a field-value pair is <fieldName>=<value>.

Enclose all field-value pairs in one set of double quotation marks, delimited by spaces.

Enclose values that contain spaces in single quotes.

To get data on API performance metrics, specify both --perflog and --json.

Examples for `force:data:record:get`

```
sfdx force:data:record:get -s Account -i 001D000000Kv3dl
```

```
sfdx force:data:record:get -s Account -w "Name=Acme"
```

```
sfdx force:data:record:get -s Account -w "Name='Universal Containers'"
```

```
sfdx force:data:record:get -s Account -w "Name='Universal Containers' Phone='(123)
456-7890'"
```

```
sfdx force:data:record:get -t -s TraceFlag -i 7tf170000009cUBAAY --perflog --json
```

Command Syntax

`sfdx force:data:record:get`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-s SOBJECTTYPE
[-i SOBJECTID]
[-w WHERE]
[-t]
[--perflog]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-s | --sobjecttype SOBJECTTYPE`

Required

The type of the record you're retrieving.

Type: string

`-i | --sobjectid SOBJECTID`

Optional

The ID of the record you're retrieving.

Type: id

`-w | --where WHERE`

Optional

A list of <fieldName>=<value> pairs to search for.

Type: string

`-t | --usetoolingapi`

Optional

Retrieve the record with Tooling API.

Type: boolean

`--perflog`

Optional

Get API performance data.

Type: boolean

`force:data:record:update`

Updates a single record.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style data update record command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

- Removed flag: `--perflog`
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjectype`. New name: `--sobject`.
- Changed flag name: Old name `--usetoolingapi`. New name: `--use-tooling-api`.
- Changed flag name: Old name `--sobjectid`. New name: `--record-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:data:record:update`

The format of a field-value pair is <fieldName>=<value>.

Enclose all field-value pairs in one set of double quotation marks, delimited by spaces.

Enclose values that contain spaces in single quotes.

To get data on API performance metrics, specify both --perflog and --json.

Examples for `force:data:record:update`

```
sfdx force:data:record:update -s Account -i 001D000000Kv3dl -v "Name=NewAcme"
```

```
sfdx force:data:record:update -s Account -w "Name='Old Acme'" -v "Name='New Acme'"
```

```
sfdx force:data:record:update -s Account -i 001D000000Kv3dl -v "Name='Acme III'
Website=www.example.com"
```

```
sfdx force:data:record:update -t -s TraceFlag -i 7tf170000009cUBAAY -v
"ExpirationDate=2017-12-01T00:58:04.000+0000"
```

$sfdx force:data:record:update -s Account -i 001D000000Kv3dl -v "Name=NewAcme" --perflog --json

Command Syntax

```
sfdx force:data:record:update
   [--json]
   [--loglevel LOGLEVEL]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   -s SOBJECTTYPE
   [-i SOBJECTID]
   [-w WHERE]
   -v VALUES
   [-t]
   [--perflog]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-s | --sobjecttype SOBJECTTYPE`
Required

The sObject type of the record you're updating.

Type: string

`-i | --sobjectid SOBJECTID`
Optional

The ID of the record you're updating.

Type: id

`-w | --where WHERE`
Optional

A list of <fieldName>=<value> pairs to search for.

Type: string

`-v | --values VALUES`
Required

The <fieldName>=<value> pairs you're updating.

Type: string

`-t | --usetoolingapi`
Optional

Update the record with Tooling API.

Type: boolean

--perflog

Optional

Get API performance data.

Type: boolean

`force:data:soql:bulk:report`

View the status of a bulk query.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style data query resume command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--resultformat`. New name: `--result-format`.
- Changed flag name: Old name `--bulkqueryid`. New name: `--bulk-query-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:data:soql:bulk:report`

Run this command using the job ID returned from the force:data:soql:query --bulk command.

Examples for `force:data:soql:bulk:report`

```
sfdx force:data:soql:bulk:report -i 7500x000005BdFzXXX
```

Command Syntax

`sfdx force:data:soql:bulk:report`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`[-r RESULTFORMAT]`

`-i BULKQUERYID`

Parameters

--json

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-r | --resultformat RESULTFORMAT`

Optional

Result format emitted to stdout; --json flag overrides this parameter.

Type: enum

Permissible values are: human, csv, json

Default value: human

`-i | --bulkqueryid BULKQUERYID`

Required

The job ID of the bulk query.

Type: string

`force:data:soql:query`

Execute a SOQL query.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style data query command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- New flag: `--async`
- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

- Changed flag name: Old name `--usetoolingapi`. New name: `--use-tooling-api`.
- Changed flag name: Old name `--soqlqueryfile`. New name: `--file`.
- Changed flag name: Old name `--resultformat`. New name: `--result-format`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:data:soql:query`

When you execute this command in a project, it executes the query against the data in your default scratch org.

To get data on API performance metrics, specify both --perflog and --json.

Examples for `force:data:soql:query`

```
sfdx force:data:soql:query -q "SELECT Id, Name, Account.Name FROM Contact"
```

```
sfdx force:data:soql:query -q "SELECT Id, Name FROM Account WHERE ShippingState IN ('CA',
'NY')"
```

```
sfdx force:data:soql:query -q "SELECT Id, Name FROM Account WHERE ShippingState IN ('CA',
'NY')" --perflog --json
```

```
sfdx force:data:soql:query -q "SELECT Name FROM ApexTrigger" -t
```

```
sfdx force:data:soql:query --soqlqueryfile query.txt
```

```
sfdx force:data:soql:query --soqlqueryfile query.txt -t
```

Command Syntax

`sfdx force:data:soql:query`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-q QUERY]
[-f SOQLQUERYFILE]
[-t]
[-b]
[-w WAIT]
[-r RESULTFORMAT]
[--perflog]
```

Parameters

`--json`
: Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
: Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
: Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
: Optional

Override the API version used for API requests made by this command.

Type: string

`-q | --query QUERY`
: Optional

SOQL query to execute.

Type: string

`-f | --soqlqueryfile SOQLQUERYFILE`
: Optional

A SOQL query stored in a file.

Type: filepath

`-t | --usetoolingapi`
: Optional

Execute query with Tooling API.

Type: boolean

`-b | --bulk`
: Optional

Use the bulk 2.0 API to query data.

Type: boolean

`-w | --wait WAIT`
: Optional

Wait time for command to finish in minutes.

Type: minutes

`-r | --resultformat RESULTFORMAT`

Optional

Result format emitted to stdout; --json flag overrides this parameter.

Type: enum

Permissible values are: human, csv, json

Default value: human

`--perflog`

Optional

Get API performance data.

Type: boolean

`force:data:tree:export`

Export data from an org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style data export tree command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:data:tree:export`

Exports data from an org into sObject tree format for use with the force:data:tree:import command.

The query for export can return a maximum of 2,000 records. For more information, see the REST API Developer Guide: https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/resources_composite_sobject_tree.htm

Examples for `force:data:tree:export`

```
 sfdx force:data:tree:export -q "SELECT Id, Name, (SELECT Name, Address__c FROM
Properties__r) FROM Broker__c"
```

```
 sfdx force:data:tree:export -q <path to file containing soql query> -x export-demo -d
/tmp/sfdx-out -p
```

Command Syntax

`sfdx force:data:tree:export`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-q QUERY
[-p]
[-x PREFIX]
[-d OUTPUTDIR]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-q | --query QUERY`

Required

Soql query, or filepath of file containing a soql query, to retrieve records.

Type: string

`-p | --plan`

Optional

Generate mulitple sobject tree files and a plan definition file for aggregated import.

Type: boolean

-x | --prefix PREFIX

Optional

Prefix of generated files.

Type: string

-d | --outputdir OUTPUTDIR

Optional

Directory to store files'.

Type: directory

`force:data:tree:import`

Import data into an org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style data import tree command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjecttreefiles`. New name: `--files`.
- Changed flag name: Old name `--confighelp`. New name: `--config-help`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 19, 2023.

Help for `force:data:tree:import`

IMPORTANT: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

Imports data into an org using the SObject Tree Save API. This data can include master-detail relationships.

To generate JSON files for use with force:data:tree:import, run "sfdx force:data:tree:export".

The SObject Tree API supports requests that contain up to 200 records. For more information, see the REST API Developer Guide: https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/resources_composite_sobject_tree.htm

Examples for `force:data:tree:import`

```
sfdx force:data:tree:import -f Contact.json,Account.json -u me@my.org
```

```
sfdx force:data:tree:import -p Account-Contact-plan.json -u me@my.org
```

Command Syntax

`sfdx force:data:tree:import`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-f SOBJECTTREEFILES]
[-p PLAN]
[--confighelp]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-f | --sobjecttreefiles SOBJECTTREEFILES`

Optional

Comma-delimited, ordered paths of json files containing collection of record trees to insert.

Type: array

`-p | --plan PLAN`

Optional

Path to plan to insert multiple data files that have master-detail relationships.

Type: filepath

`--confighelp`

Optional

Display schema information for the --plan configuration file to stdout; if you use this option, all other options except --json are ignored.

Type: boolean

`lightning` Commands

Use the lightning commands to create Aura components and Lightning web components. As of API version 45.0, you can build Lightning components using two programming models: Lightning Web Components, and the original model, Aura Components. Lightning web components and Aura components can coexist and interoperate on a page.

force:lightning:app:create

Creates a Lightning app bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

force:lightning:component:create

Creates a bundle for an Aura component or a Lightning web component in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

force:lightning:event:create

Creates a Lightning event bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

force:lightning:interface:create

Creates a Lightning interface bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

force:lightning:lwc:test:create

Creates a __tests__ directory in the specified directory. Creates a yourComponentName.test.js file with boilerplate code in the __tests__ directory.

force:lightning:lwc:test:run

Invokes Lightning Web Components Jest unit tests.

force:lightning:lwc:test:setup

Installs Jest unit testing tools for Lightning Web Components. For more information, see the Lightning Web Components Dev Guide: https://developer.salesforce.com/docs/component-library/documentation/lwc/lwc.testing.

force:lightning:test:create

Creates a Lightning test in the specified directory or the current working directory. The .resource file and associated metadata file are created.

`force:lightning:app:create`

Creates a Lightning app bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style lightning generate app command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--appname`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:lightning:app:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

If you don't specify an outputdir, we create a subfolder in your current working directory with the name of your bundle. For example, if the current working directory is force-app and your Lightning bundle is called myBundle, we create force-app/myBundle/ to store the files in the bundle.

Examples:

```
sfdx force:lightning:app:create -n myapp
```

```
sfdx force:lightning:app:create -n myapp -d aura
```

Command Syntax

`sfdx force:lightning:app:create`

```
[--json]
[--loglevel LOGLEVEL]
-n APPNAME
[-t TEMPLATE]
[-d OUTPUTDIR]
[--apiversion APIVERSION]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

-n | --appname APPNAME

Required

The Lightning app name. The name can be up to 40 characters and must start with a letter.

Type: string

-t | --template TEMPLATE

Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultLightningApp

Default value: DefaultLightningApp

-d | --outputdir OUTPUTDIR

Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

--apiversion APIVERSION

Optional

Override the API version used for API requests made by this command.

Type: string

`force:lightning:component:create`

Creates a bundle for an Aura component or a Lightning web component in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style lightning generate component command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--componentname`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:lightning:component:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

If you don't specify an outputdir, we create a subfolder in your current working directory with the name of your bundle. For example, if the current working directory is force-app and your Lightning bundle is called myBundle, we create force-app/myBundle/ to store the files in the bundle.

To create a Lightning web component, pass --type lwc to the command. If you don't include a --type value, Salesforce CLI creates an Aura component by default.

Examples:

```
sfdx force:lightning:component:create -n mycomponent
```

```
sfdx force:lightning:component:create -n mycomponent --type lwc
```

```
sfdx force:lightning:component:create -n mycomponent -d aura
```

```
sfdx force:lightning:component:create -n mycomponent --type lwc -d lwc
```

Command Syntax

`sfdx force:lightning:component:create`

```
[--json]
[--loglevel LOGLEVEL]
-n COMPONENTNAME
[-t TEMPLATE]
[-d OUTPUTDIR]
[--apiversion APIVERSION]
[--type TYPE]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-n | --componentname COMPONENTNAME`

Required

The Lightning component name. The name can be up to 40 characters and must start with a letter.

Type: string

-t | --template TEMPLATE

Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: default, analyticsDashboard, analyticsDashboardWithStep

Default value: default

-d | --outputdir OUTPUTDIR

Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

--apiversion APIVERSION

Optional

Override the API version used for API requests made by this command.

Type: string

--type TYPE

Optional

The type of the new Lightning component.

Type: string

Permissible values are: aura, lwc

Default value: aura

`force:lightning:event:create`

Creates a Lightning event bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style lightning generate event command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--eventname`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:lightning:event:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

If you don't specify an outputdir, we create a subfolder in your current working directory with the name of your bundle. For example, if the current working directory is force-app and your Lightning bundle is called myBundle, we create force-app/myBundle/ to store the files in the bundle.

Examples:

```
sfdx force:lightning:event:create -n myevent
```

```
sfdx force:lightning:event:create -n myevent -d aura
```

Command Syntax

`sfdx force:lightning:event:create`

`[--json]`

`[--loglevel LOGLEVEL]`

`-n EVENTNAME`

`[-t TEMPLATE]`

`[-d OUTPUTDIR]`

`[--apiversion APIVERSION]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-n | --eventname EVENTNAME`

Required

The Lightning event name. The name can be up to 40 characters and must start with a letter.

Type: string

`-t | --template TEMPLATE`

Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultLightningEvt

Default value: DefaultLightningEvt

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`force:lightning:interface:create`

Creates a Lightning interface bundle in the specified directory or the current working directory. The bundle consists of multiple files in a folder with the designated name.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style lightning generate interface command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--interfacename`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:lightning:interface:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

If you don't specify an outputdir, we create a subfolder in your current working directory with the name of your bundle. For example, if the current working directory is force-app and your Lightning bundle is called myBundle, we create force-app/myBundle/ to store the files in the bundle.

Examples:

```
sfdx force:lightning:interface:create -n myinterface
```

```
sfdx force:lightning:interface:create -n myinterface -d aura
```

Command Syntax

`sfdx force:lightning:interface:create`

```
[--json]
[--loglevel LOGLEVEL]
-n INTERFACENAME
[-t TEMPLATE]
[-d OUTPUTDIR]
[--apiversion APIVERSION]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-n | --interfacename INTERFACENAME`

Required

The Lightning interface name. The name can be up to 40 characters and must start with a letter.

Type: string

`-t | --template TEMPLATE`

Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultLightningIntf

Default value: DefaultLightningIntf

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`force:lightning:lwc:test:create`

Creates a __tests__ directory in the specified directory. Creates a yourComponentName.test.js file with boilerplate code in the __tests__ directory.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style force lightning lwc test create command instead.

Examples for `force:lightning:lwc:test:create`

```
sfdx force:lightning:lwc:test:create -f force-app/main/default/lwc/myButton/myButton.js
```

Command Syntax

`sfdx force:lightning:lwc:test:create`

`[--json]`

`[--loglevel LOGLEVEL]`

`-f FILEPATH`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-f | --filepath FILEPATH`

Required

Path to Lightning web component .js file to create a test for.

Type: string

`force:lightning:lwc:test:run`

Invokes Lightning Web Components Jest unit tests.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style force lightning lwc test run command instead.

Examples for `force:lightning:lwc:test:run`

```
sfdx force:lightning:lwc:test:run
```

```
sfdx force:lightning:lwc:test:run -w
```

Command Syntax

`sfdx force:lightning:lwc:test:run`

```
[--json]
[--loglevel LOGLEVEL]
[-d]
[--watch]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-d` | `--debug`

Optional

Runs tests in a Node process that an external debugger can connect to. The run pauses until the debugger is connected. For more information, see: https://jestjs.io/docs/en/troubleshooting

Type: boolean

`--watch`

Optional

Runs tests when a watched file changes. Watched files include the component under test and any files it references.

Type: boolean

`force:lightning:lwc:test:setup`

Installs Jest unit testing tools for Lightning Web Components. For more information, see the Lightning Web Components Dev Guide: https://developer.salesforce.com/docs/component-library/documentation/lwc/lwc.testing.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style force lightning lwc test setup command instead.

Examples for `force:lightning:lwc:test:setup`

```
sfdx force:lightning:lwc:test:setup
```

Command Syntax

`sfdx force:lightning:lwc:test:setup`

`[--json]`

`[--loglevel LOGLEVEL]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`force:lightning:test:create`

Creates a Lightning test in the specified directory or the current working directory. The .resource file and associated metadata file are created.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style lightning generate test command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--testname`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:lightning:test:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
sfdx force:lightning:test:create -n MyLightningTest
```

```
sfdx force:lightning:test:create -n MyLightningTest -d lightningTests
```

Command Syntax

`sfdx force:lightning:test:create`

`[--json]`

`[--loglevel LOGLEVEL]`

`-n TESTNAME`

`[-t TEMPLATE]`

`[-d OUTPUTDIR]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-n | --testname TESTNAME`

Required

The name of the new Lightning test. The name can be up to 40 characters and must start with a letter.

Type: string

`-t | --template TEMPLATE`

Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultLightningTest

Default value: DefaultLightningTest

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`limits` Commands

Display current org's limits.

force:limits:api:display

Display current org's limits.

force:limits:recordcounts:display

Display record counts for the specified standard and custom objects.

`force:limits:api:display`

Display current org's limits.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style limits api display command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 12, 2023.

Help for `force:limits:api:display`

When you execute this command in a project, it provides limit information for your default scratch org.

Examples for `force:limits:api:display`

```
sfdx force:limits:api:display
```

```
sfdx force:limits:api:display -u me@my.org
```

Command Syntax

`sfdx force:limits:api:display`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`force:limits:recordcounts:display`

Display record counts for the specified standard and custom objects.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style limits recordcounts display command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjecttype`. New name: `--sobject`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 12, 2023.

Help for `force:limits:recordcounts:display`

Use this command to get an approximate count of the records in standard or custom objects in your org. These record counts are the same as the counts listed in the Storage Usage page in Setup. The record counts are approximate because they're calculated asynchronously and your org's storage usage isn't updated immediately. To display all available record counts, run the command without the '--sobjecttype' parameter.

Examples for `force:limits:recordcounts:display`

```
sfdx force:limits:recordcounts:display
```

```
sfdx force:limits:recordcounts:display -s Account,Contact,Lead,Opportunity
```

```
sfdx force:limits:recordcounts:display -s Account,Contact -u me@my.org
```

Command Syntax

`sfdx force:limits:recordcounts:display`

```
    [--json]
    [--loglevel LOGLEVEL]
    [-u TARGETUSERNAME]
    [--apiversion APIVERSION]
    [-s SOBJECTTYPE]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-s | --sobjecttype SOBJECTTYPE`

Optional

Comma-separated list of API names of standard or custom objects for which to display record counts.

Type: array

`mdapi` Commands

Use the mdapi commands to retrieve and deploy Metadata API–formatted files that represent components in an org, or to convert Metadata API–formatted metadata into the source format used in Salesforce DX projects.

force:mdapi:convert
Convert metadata from the Metadata API format into the source format.

force:mdapi:deploy
Deploy metadata to an org using Metadata API.

force:mdapi:deploy:cancel
Cancel a metadata deployment .

force:mdapi:deploy:report
Check the status of a metadata deployment.

force:mdapi:describemetadata
Display details about the metadata types enabled for your org.

force:mdapi:listmetadata
Display properties of metadata components of a specified type.

force:mdapi:retrieve
Retrieve metadata from an org using Metadata API.

force:mdapi:retrieve:report
Check the status of a metadata retrieval.

`force:mdapi:convert`

Convert metadata from the Metadata API format into the source format.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project convert mdapi command instead.

Use this table to map the flags between the old and new commands.

`force:mdapi:convert` **Flag**	**Equivalent** `project convert mdapi` **Flag**
`-p`, `--metadatapath`	`-p`, `--metadata-dir`
`-r`, `--rootdir`	`-r`, `--root-dir`
`-d`, `--outputdir`	`-d`, `--output-dir`
`-x`, `--manifest`	`-x`, `--manifest`
`-m`, `--metadata`	`-m`, `--metadata`
`-json`	`-json`
`-loglevel`	No equivalent. Use the SF_LOG_LEVEL environment variable instead.

Here's an example to help you update your old commands. This `sfdx`-style command:

```
sfdx force:mdapi:convert --rootdir path/to/metadata --outputdir path/to/outputdir
```

Looks like this using the equivalent `sf`-style command:

```
sf project convert mdapi --root-dir path/to/metadata --output-dir path/to/outputdir
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:mdapi:convert`

Converts metadata retrieved via Metadata API into the source format used in Salesforce DX projects.

To use Salesforce CLI to work with components that you retrieved via Metadata API, first convert your files from the metadata format to the source format using "sfdx force:mdapi:convert".

To convert files from the source format back to the metadata format, so that you can deploy them using "sfdx force:mdapi:deploy", run "sfdx force:source:convert".

Examples for `force:mdapi:convert`

```
sfdx force:mdapi:convert -r path/to/metadata
```

```
sfdx force:mdapi:convert -r path/to/metadata -d path/to/outputdir
```

Command Syntax

`sfdx force:mdapi:convert`

```
[--json]
[--loglevel LOGLEVEL]
-r ROOTDIR
[-d OUTPUTDIR]
[-x MANIFEST]
[-p METADATAPATH]
[-m METADATA]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-r | --rootdir ROOTDIR`

Required

The root directory that contains the metadata you retrieved using Metadata API.

Type: directory

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store your files in after they're converted to the source format. Can be an absolute or relative path.

Type: directory

`-x | --manifest MANIFEST`

Optional

The complete path to the manifest (package.xml) file that specifies the metadata types to convert.

If you specify this parameter, don't specify --metadata or --sourcepath.

Type: string

`-p | --metadatapath METADATAPATH`

Optional

A comma-separated list of paths to the local metadata files to convert. The supplied paths can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all metadata types in the directory and its sub-directories).

If you specify this parameter, don't specify --manifest or --metadata. If the comma-separated list you're supplying contains spaces, enclose the entire comma-separated list in one set of double quotes.

Type: array

`-m | --metadata METADATA`

Optional

A comma-separated list of metadata component names to convert.

Type: array

Aliases for `force:mdapi:convert`

```
force:mdapi:beta:convert
```

`force:mdapi:deploy`

Deploy metadata to an org using Metadata API.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project deploy start command instead.

Because the `project deploy start` command works for files in both source format and metadata format (`mdapi`), specify the `--metadata-dir` flag to mimic the `force:mdapi:deploy` behavior.

Use this table to map the flags between the old and new commands. The mapping isn't always one-to-one; see the Notes column for more information.

`force:mdapi:deploy` **Flag**	**Equivalent** `project deploy start` **Flag**	**Notes**
`-c`, `--checkonly`	`--dry-run`, no short name.	You can also use `project deploy validate --metadata-dir`
`-d`, `--deploydir`	`--metadata-dir`, no short name.	
`-f`, `--zipfile`	`--metadata-dir`, no short name.	
`-g`, `--ignorewarnings`	`-g`, `--ignore-warnings`	
`-l`, `--testlevel`	`-l`, `--test-level`	
`-o`, `--ignoreerrors`	`-r`, `--ignore-errors`	
`-u`, `--targetusername`	`-o`, `--target-org`	
`-q`, `--validateddeployrequestid`	No equivalent.	Use the `project deploy validate --metadata-dir` and `project deploy quick --job-id` commands.
`-r`, `--runtests`	`-t`, `--tests`	

force:mdapi:deploy **Flag**	**Equivalent** project deploy start **Flag**	**Notes**
-s, --singlepackage	--single-package, no short name.	
-w, --wait	-w, --wait	
--apiversion	-a, --api-version	
--concise	--concise	
--coverageformatters	--coverage-formatters	
--json	--json	
--junit	--junit	
--loglevel	No equivalent. Use the SF_LOG_LEVEL environment variable instead.	
--purgeondelete	--purge-on-delete	
--resultsdir	--results-dir	
--soapdeploy	No equivalent.	Deploys use SOAP API by default. To use REST API, set the org-metadata-rest-deploy config variable or SF_ORG_METADATA_REST_DEPLOY environment variable.
--verbose	--verbose	

Here are some examples to help you update your old commands. This sfdx-style command:

```
sfdx force:mdapi:deploy --deploydir some/path --wait 1000 --checkonly \
--testlevel RunAllTestsInOrg
```

Looks like this using the equivalent sf-style command:

```
sf project deploy start --metadata-dir some/path --wait 1000 --dry-run \
--test-level RunAllTestsInOrg
```

This sfdx-style command:

```
sfdx force:mdapi:deploy --zipfile stuff.zip --resultsdir
```

Looks like this using the equivalent sf-style command:

```
sf project deploy start --metadata-dir stuff.zip --results-dir
```

For background information about this change, read this blog post, which describes how we've updated many of the existing sfdx commands to use the improvements we made in sf. We improved this particular command on April 13, 2023.

Examples for **`force:mdapi:deploy`**

Return a job ID you can use to check the deploy status:

```
sfdx force:mdapi:deploy -d some/path
```

Deploy and poll for 1000 minutes:

```
sfdx force:mdapi:deploy -d some/path -w 1000
```

Deploy a ZIP file:

```
sfdx force:mdapi:deploy -f stuff.zip
```

Validate a deployment so the ID can be used for a quick deploy:

```
sfdx force:mdapi:deploy -d some/path -w 1000 -c --testlevel RunAllTestsInOrg
```

Quick deploy using a previously validated deployment:

```
sfdx force:mdapi:deploy -q MyValidatedId
```

Command Syntax

`sfdx force:mdapi:deploy`

```
  [--json]
  [--loglevel LOGLEVEL]
  [-u TARGETUSERNAME]
  [--apiversion APIVERSION]
  [-c]
  [-d DEPLOYDIR]
  [-w WAIT]
  [-l TESTLEVEL]
  [-r RUNTESTS]
  [-o]
  [-g]
  [-q VALIDATEDDEPLOYREQUESTID]
  [--verbose]
  [-f ZIPFILE]
  [-s]
  [--soapdeploy]
  [--purgeondelete]
  [--concise]
  [--resultsdir RESULTSDIR]
  [--coverageformatters COVERAGEFORMATTERS]
  [--junit]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-c | --checkonly`

Optional

IMPORTANT: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

Validates the deployed metadata and runs all Apex tests, but prevents the deployment from being saved to the org.

If you change a field type from Master-Detail to Lookup or vice versa, that change isn't supported when using the --checkonly parameter to test a deployment (validation). This kind of change isn't supported for test deployments to avoid the risk of data loss or corruption. If a change that isn't supported for test deployments is included in a deployment package, the test deployment fails and issues an error.

If your deployment package changes a field type from Master-Detail to Lookup or vice versa, you can still validate the changes prior to deploying to Production by performing a full deployment to another test Sandbox. A full deployment includes a validation of the changes as part of the deployment process.

Note: A Metadata API deployment that includes Master-Detail relationships deletes all detail records in the Recycle Bin in the following cases.

1. For a deployment with a new Master-Detail field, soft delete (send to the Recycle Bin) all detail records before proceeding to deploy the Master-Detail field, or the deployment fails. During the deployment, detail records are permanently deleted from the Recycle Bin and cannot be recovered.

2. For a deployment that converts a Lookup field relationship to a Master-Detail relationship, detail records must reference a master record or be soft-deleted (sent to the Recycle Bin) for the deployment to succeed. However, a successful deployment permanently deletes any detail records in the Recycle Bin.

Type: boolean

`-d | --deploydir DEPLOYDIR`

Optional

The root of the directory tree that contains the files to deploy. The root must contain a valid package.xml file describing the entities in the directory structure. Required to initiate a deployment if you don't use --zipfile. If you specify both --zipfile and --deploydir, a zip file of the contents of the --deploydir directory is written to the location specified by --zipfile.

Type: directory

`-w | --wait WAIT`

Optional

The number of minutes to wait for the command to complete. The default is 0 (returns immediately). 0

Type: minutes

Default value: 0 minutes

`-l | --testlevel TESTLEVEL`

Optional

Specifies which level of deployment tests to run. Valid values are:

NoTestRun—No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial orgs. This test level is the default for development environments.

RunSpecifiedTests—Runs only the tests that you specify in the --runtests option. Code coverage requirements differ from the default coverage requirements when using this test level. Executed tests must comprise a minimum of 75% code coverage for each class and trigger in the deployment package. This coverage is computed for each class and trigger individually and is different than the overall coverage percentage.

RunLocalTests—All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default for production deployments that include Apex classes or triggers.

RunAllTestsInOrg—All tests in your org are run, including tests of managed packages.

If you don't specify a test level, the default behavior depends on the contents of your deployment package. For more information, see "Running Tests in a Deployment" in the Metadata API Developer Guide.

Type: enum

Permissible values are: NoTestRun, RunSpecifiedTests, RunLocalTests, RunAllTestsInOrg

`-r | --runtests RUNTESTS`

Optional

Lists the Apex classes containing the deployment tests to run. Use this parameter when you set --testlevel to RunSpecifiedTests.

Type: array

`-o | --ignoreerrors`

Optional

Ignores the deploy errors, and continues with the deploy operation. The default is false. Keep this parameter set to false when deploying to a production org. If set to true, components without errors are deployed, and components with errors are skipped.

Type: boolean

`-g | --ignorewarnings`

Optional

If a warning occurs and ignoreWarnings is set to true, the success field in DeployMessage is true. When ignoreWarnings is set to false, success is set to false, and the warning is treated like an error.

This field is available in API version 18.0 and later. Prior to version 18.0, there was no distinction between warnings and errors. All problems were treated as errors and prevented a successful deployment.

Type: boolean

`-q | --validateddeployrequestid VALIDATEDDEPLOYREQUESTID`

Optional

Specifies the ID of a package with recently validated components to run a Quick Deploy. Deploying a validation helps you shorten your deployment time because tests aren't rerun. If you have a recent successful validation, you can deploy the validated components without running tests. A validation doesn't save any components in the org. You use a validation only to check the success or failure messages that you would receive with an actual deployment. To validate your components, add the -c | --checkonly flag when you run 'sfdx force:mdapi:deploy'. This flag sets the checkOnly='true' parameter for your deployment. Before deploying a recent validation, ensure that the following requirements are met:

1. The components have been validated successfully for the target environment within the last 10 days.

2. As part of the validation, Apex tests in the target org have passed.

3. Code coverage requirements are met.

- If all tests in the org or all local tests are run, overall code coverage is at least 75%, and Apex triggers have some coverage.

- If specific tests are run with the RunSpecifiedTests test level, each class and trigger that was deployed is covered by at least 75% individually.

Type: id

`--verbose`

Optional

Indicates that you want verbose output from the deploy operation.

Type: boolean

`-f | --zipfile ZIPFILE`

Optional

The path to the .zip file of metadata files to deploy. You must indicate this option or --deploydir.If you specify both --zipfile and --deploydir, a .zip file of the contents of the deploy directory is created at the path specified for the .zip file.

Type: filepath

`-s | --singlepackage`

Optional

Indicates that the specified .zip file points to a directory structure for a single package. By default, the CLI assumes the directory is structured for a set of packages.

Type: boolean

`--soapdeploy`

Optional

Deploy metadata with SOAP API instead of the default REST API. Because SOAP API has a lower .ZIP file size limit (400 MB uncompressed, 39 MB compressed), Salesforce recommends REST API deployment. This flag provides backwards compatibility with API version 50.0 and earlier when deploy used SOAP API by default.

Type: boolean

`--purgeondelete`

Optional

Specify that deleted components in the destructive changes manifest file are immediately eligible for deletion rather than being stored in the Recycle Bin.

Type: boolean

`--concise`

Optional

Emit brief command output to stdout.

Type: boolean

`--resultsdir RESULTSDIR`

Optional

Output directory for code coverage and JUnit results; defaults to the deploy ID.

Type: directory

`--coverageformatters COVERAGEFORMATTERS`

Optional

Format of the code coverage results.

Type: array

`--junit`

Optional

Output JUnit test results.

Type: boolean

Aliases for `force:mdapi:deploy`

```
force:mdapi:beta:deploy
```

`force:mdapi:deploy:cancel`

Cancel a metadata deployment .

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project deploy cancel command instead.

Use this table to map the flags between the old and new commands.

`force:mdapi:deploy:cancel` **Flag**	**Equivalent** `project deploy cancel` **Flag**
`-u`, `--targetusername`	No equivalent.
`-apiversion`	No equivalent.
`-w`, `--wait`	`-w`, `--wait`
`-i`, `--jobid`	`-i`, `--job-id`
`-json`	`-json`

`force:mdapi:deploy:cancel` **Flag**	**Equivalent** `project deploy cancel` **Flag**
`-loglevel`	No equivalent. Use the SF_LOG_LEVEL environment variable instead.

Here's an example to help you update your old commands. This `sfdx`-style command:

```
sfdx force:mdapi:deploy:cancel --wait 2 --jobid <jobid>
```

Looks like this using the equivalent `sf`-style command:

```
sf project deploy cancel --wait 2 --job-id <jobid>
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:mdapi:deploy:cancel`

Use this command to cancel a specified asynchronous metadata deployment. You can also specify a wait time (in minutes) to check for updates to the canceled deploy status.

Examples for `force:mdapi:deploy:cancel`

Deploy a directory of files to the org

```
sfdx force:mdapi:deploy -d <directory>
```

Now cancel this deployment and wait two minutes

```
sfdx force:mdapi:deploy:cancel -w 2
```

If you have multiple deployments in progress and want to cancel a specific one, specify the job ID

```
sfdx force:mdapi:deploy:cancel -i <jobid>
```

Check the status of the cancel job

```
sfdx force:mdapi:deploy:report
```

Command Syntax

```
sfdx force:mdapi:deploy:cancel
   [--json]
   [--loglevel LOGLEVEL]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   [-w WAIT]
   [-i JOBID]
```

Parameters

`--json`
: Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
: Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u` | `--targetusername TARGETUSERNAME`
: Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
: Optional

Override the API version used for API requests made by this command.

Type: string

`-w` | `--wait WAIT`
: Optional

Number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you. The default is 33 minutes.

Type: minutes

Default value: 33 minutes

`-i` | `--jobid JOBID`
: Optional

Job ID of the deployment you want to cancel; defaults to your most recent CLI deployment if not specified.

Type: id

`force:mdapi:deploy:report`

Check the status of a metadata deployment.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style commands.

This command (`force:mdapi:deploy:report`) does more than just report: it also resumes a deployment, which is confusing. For this reason, we now provide these two new commands for each task, which is much more intuitive:

- project deploy report
- project deploy resume

Use this table to map the flags between the old and new commands.

`force:mdapi:deploy:report` **Flag**	**Equivalent** `project deploy report` **or** `project deploy resume` **Flag**	**Notes**
`-u,--targetusername`	`-o,--target-org`	
`--apiversion`	`--api-version`	
`-w, --wait`	`-w, --wait`	The new commands don't support the `--wait -1` flag (which means "wait forever"). Instead, specify a very large number with the new commands.
`-i, --jobid`	`-i, --job-id`	
`--concise`	`--concise`	
`--verbose`	`--verbose`	
`--resultsdir`	`--results-dir`	
`--coverageformatters`	`--coverage-formatters`	
`--junit`	`--junit`	
`-json`	`-json`	
`-loglevel`	No equivalent. Use the SF_LOG_LEVEL environment variable instead.	

Here are some examples to help you update your old commands. This `sfdx`-style command:

```
sfdx force:mdapi:deploy:report --jobid 1234 --wait 10
```

Looks like this using the equivalent `sf`-style command:

```
sf project deploy report --job-id 1234 --wait 10
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Examples for `force:mdapi:deploy:report`

Check the status of the most recent deployment

```
sfdx force:mdapi:deploy:report
```

Check the status of a deploy with job ID 1234 and wait for 10 minutes for the result:

```
sfdx force:mdapi:deploy:report -i 1234 -w 10
```

Command Syntax

```
sfdx force:mdapi:deploy:report
   [--json]
   [--loglevel LOGLEVEL]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   [-w WAIT]
   [-i JOBID]
   [--verbose]
   [--concise]
   [--resultsdir RESULTSDIR]
   [--coverageformatters COVERAGEFORMATTERS]
   [--junit]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-w | --wait WAIT`

Optional

The number of minutes to wait for the command to complete. The default is –1 (no limit).

Type: minutes

Default value: 0 minutes

`-i | --jobid JOBID`

Optional

The job ID (id field value for AsyncResult) of the deployment you want to check. The job ID is required if you haven't previously deployed using Salesforce CLI. If you deploy using Salesforce CLI and don't specify a job ID, we use the ID of the most recent metadata deployment.

Type: id

`--verbose`

Optional

Indicates that you want verbose output for deploy results.

Type: boolean

`--concise`

Optional

Emit brief command output to stdout.

Type: boolean

`--resultsdir RESULTSDIR`

Optional

Output directory for code coverage and JUnit results; defaults to the deploy ID.

Type: directory

`--coverageformatters COVERAGEFORMATTERS`

Optional

Format of the code coverage results.

Type: array

`--junit`

Optional

Output JUnit test results.

Type: boolean

Aliases for `force:mdapi:deploy:report`

```
force:mdapi:beta:deploy:report
```

`force:mdapi:describemetadata`

Display details about the metadata types enabled for your org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org list metadata-types command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--metadatatype`. New name: `--metadata-type`.
- Changed flag name: Old name `--resultfile`. New name: `--output-file`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 23, 2023.

Help for `force:mdapi:describemetadata`

Use this information to identify the syntax needed for a <name> element in package.xml. The most recent API version is the default, or you can specify an older version.

The default target username is the admin user for the default scratch org. The username must have the Modify All Data permission or the Modify Metadata permission (Beta). For more information about permissions, see Salesforce Help.

Examples for `force:mdapi:describemetadata`

```
sfdx force:mdapi:describemetadata -a 43.0
```

```
sfdx force:mdapi:describemetadata -u me@example.com
```

```
sfdx force:mdapi:describemetadata -f /path/to/outputfilename.txt
```

```
sfdx force:mdapi:describemetadata -u me@example.com -f /path/to/outputfilename.txt
```

Command Syntax

`sfdx force:mdapi:describemetadata`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[-a APIVERSION]`

`[-f RESULTFILE]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`-a | --apiversion APIVERSION`

Optional

The API version to use. The default is the latest API version

Type: string

`-f | --resultfile RESULTFILE`

Optional

The path to the file where the results of the command are stored. Directing the output to a file makes it easier to extract relevant information for your package.xml manifest file. The default output destination is the console.

Type: filepath

`force:mdapi:listmetadata`

Display properties of metadata components of a specified type.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org list metadata command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--metadatatype`. New name: `--metadata-type`.
- Changed flag name: Old name `--resultfile`. New name: `--output-file`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short flag name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 23, 2023.

Help for `force:mdapi:listmetadata`

This command is useful when you want to identify individual components in your manifest file or if you want a high-level view of particular components in your organization. For example, you could use this target to return a list of names of all Layout components in your org, then use this information in a retrieve operation that returns a subset of these components.

Examples for `force:mdapi:listmetadata`

```
sfdx force:mdapi:listmetadata -m CustomObject
```

```
sfdx force:mdapi:listmetadata -m CustomObject -a 43.0
```

```
sfdx force:mdapi:listmetadata -m CustomObject -u me@example.com
```

```
sfdx force:mdapi:listmetadata -m CustomObject -f /path/to/outputfilename.txt
```

```
sfdx force:mdapi:listmetadata -m Dashboard --folder foldername
```

```
sfdx force:mdapi:listmetadata -m Dashboard --folder foldername -a 43.0
```

```
sfdx force:mdapi:listmetadata -m Dashboard --folder foldername -u me@example.com
```

```
sfdx force:mdapi:listmetadata -m Dashboard --folder foldername -f
/path/to/outputfilename.txt
```

```
sfdx force:mdapi:listmetadata -m CustomObject -u me@example.com -f
/path/to/outputfilename.txt
```

Command Syntax

`sfdx force:mdapi:listmetadata`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[-a APIVERSION]`

`[-f RESULTFILE]`

`-m METADATATYPE`

`[--folder FOLDER]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`-a | --apiversion APIVERSION`

Optional

The API version to use. The default is the latest API version

Type: string

`-f | --resultfile RESULTFILE`

Optional

The path to the file where the results of the command are stored. The default output destination is the console.

Type: filepath

`-m | --metadatatype METADATATYPE`

Required

The metadata type to be retrieved, such as CustomObject or Report. The metadata type value is case-sensitive.

Type: string

`--folder FOLDER`

Optional

The folder associated with the component. This parameter is required for components that use folders, such as Dashboard, Document, EmailTemplate, or Report. The folder name value is case-sensitive.

Type: string

`force:mdapi:retrieve`

Retrieve metadata from an org using Metadata API.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project retrieve start command instead.

Because the `project retrieve start` command works for files in both source format and metadata format (`mdapi`), specify the `--target-metadata-dir` flag to mimic the `force:mdapi:retrieve` behavior.

Use this table to map the flags between the old and new commands.

`force:mdapi:retrieve` **Flag**	**Equivalent** `project retrieve start` **Flag**
`-r`, `--retrievetargetdir`	`-t`, `--target-metadata-dir`
`-k`, `--unpackaged`	`-x`, `--manifest`
`-d`, `--sourcedir`	`-d`, `--source-dir`
`-p`, `--packagenames`	`-n`, `--package-name`
`-n`, `--zipfilename`	`--zip-file-name`, no short name.

`force:mdapi:retrieve` **Flag**	**Equivalent** `project retrieve start` **Flag**
`-u, --targetusername`	`-o, --target-org`
`-z, --unzip`	`-z, --unzip`
`-s, --singlepackage`	`--single-package`, no short name.
`-w, --wait`	`-w, --wait`
`--apiversion`	`-a, --api-version`
`--json`	`--json`
`--loglevel`	`No equivalent. Use the SF_LOG_LEVEL environment variable instead.`
`--verbose`	No equivalent

Here are some examples to help you update your old commands. This `sfdx`-style command:

```
sfdx force:mdapi:retrieve --retrievetargetdir path/to/retrieve/dir --unpackaged
package.xml
```

Looks like this using the equivalent `sf`-style command:

```
sf project retrieve start --target-metadata-dir path/to/retrieve/dir --manifest
package.xml
```

This `sfdx`-style command:

```
sfdx force:mdapi:retrieve --sourcedir path/to/apexClasses \
--retrievetargetdir path/to/retrieve/dir --unzip --zipfilename apexClasses.zip
```

Looks like this using the equivalent `sf`-style command:

```
sf project retrieve start --source-dir path/to/apexClasses \
--target-metadata-dir path/to/retrieve/dir --unzip --zip-file-name apexClasses.zip
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for **`force:mdapi:retrieve`**

Uses Metadata API to retrieve a .zip of XML files that represent metadata from the targeted org. The default target username is the admin user for the default scratch org. You can retrieve and deploy up to 10,000 files or 400 MB (39 MB compressed) at one time.

Examples for **`force:mdapi:retrieve`**

Retrieve metadata in the default project directory into the target directory:

```
sfdx force:mdapi:retrieve -r path/to/retrieve/dir
```

Retrieve metadata defined in the specified manifest into the target directory:

```
sfdx force:mdapi:retrieve -r path/to/retrieve/dir -k package.xml
```

Retrieve metadata defined by the specified directory, name the retrieved zipfile and extract all contents

```
sfdx force:mdapi:retrieve -d path/to/apexClasses -r path/to/retrieve/dir --unzip
--zipfilename apexClasses.zip
```

Enqueue a retrieve request but do not wait for the metadata to be retrieved:

```
sfdx force:mdapi:retrieve -r path/to/retrieve/dir --wait 0
```

Command Syntax

`sfdx force:mdapi:retrieve`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[-a APIVERSION]`

`-r RETRIEVETARGETDIR`

`[-k UNPACKAGED]`

`[-d SOURCEDIR]`

`[-p PACKAGENAMES]`

`[-s]`

`[-n ZIPFILENAME]`

`[-z]`

`[-w WAIT]`

`[--verbose]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

-a | --apiversion APIVERSION
Optional

Use to override the default, which is the latest version supported by your CLI plug-in, with the version in your package.xml file.

Type: string

-r | --retrievetargetdir RETRIEVETARGETDIR
Required

The root of the directory structure where the retrieved .zip or metadata files are put.

Type: directory

-k | --unpackaged UNPACKAGED
Optional

The complete path for the manifest file that specifies the components to retrieve.

Type: filepath

-d | --sourcedir SOURCEDIR
Optional

The source directory to use instead of the default package directory specified in sfdx-project.json

Type: directory

-p | --packagenames PACKAGENAMES
Optional

A comma-separated list of package names to retrieve.

Type: array

-s | --singlepackage
Optional

Indicates that the specified .zip file points to a directory structure for a single package. By default, the CLI assumes the directory is structured for a set of packages.

Type: boolean

-n | --zipfilename ZIPFILENAME
Optional

The file name to use for the retrieved zip file.

Type: string

-z | --unzip
Optional

Extract all files from the retrieved zip file.

Type: boolean

-w | --wait WAIT
Optional

The number of minutes to wait for the command to complete.

Type: minutes

Default value: 1440 minutes

`--verbose`

Optional

Indicates that you want verbose output from the retrieve operation.

Type: boolean

Aliases for `force:mdapi:retrieve`

```
force:mdapi:beta:retrieve
```

`force:mdapi:retrieve:report`

Check the status of a metadata retrieval.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, there is no equivalent `sf`-style command.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:mdapi:retrieve:report`

Specify the job ID and a target directory for the retrieve you want to check. You can also specify a wait time (minutes) to check for updates to the retrieve status. If the retrieve was successful, the resulting zip file will be saved to the location passed in with the retrieve target parameter.

Examples for `force:mdapi:retrieve:report`

Poll until the metadata is retrieved (or timeout is reached) using data from the last force:mdapi:retrieve command:

sfdx force:mdapi:retrieve:report

Report the current status of the last retrieve command. If the retrieve is complete the zip file of metadata is written to the target directoy:

sfdx force:mdapi:retrieve:report -r path/to/retrieve/dir -w 0

Poll until the metadata is retrieved (or timeout is reached) using the provided RetrieveID, naming the zip file and extracting all contents:

sfdx force:mdapi:retrieve:report -i retrieveId -r path/to/retrieve/dir --unzip --zipfilename apexClasses.zip

Command Syntax

`sfdx force:mdapi:retrieve:report`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-r RETRIEVETARGETDIR]
```

```
[-i JOBID]
[-n ZIPFILENAME]
[-z]
[-w WAIT]
[--verbose]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-r | --retrievetargetdir RETRIEVETARGETDIR`

Optional

The root of the directory structure where the retrieved .zip or metadata files are put.

Type: directory

`-i | --jobid JOBID`

Optional

The job ID (asyncId) of the retrieve you want to check. If not specified, the default value is the ID of the most recent metadata retrieval you ran using Salesforce CLI. You must specify a --retrievetargetdir. Use with --wait to resume waiting.

Type: id

`-n | --zipfilename ZIPFILENAME`

Optional

The file name to use for the retrieved zip file.

Type: string

-z | --unzip
Optional

Extract all files from the retrieved zip file.

Type: boolean

-w | --wait WAIT
Optional

The number of minutes to wait for the command to complete.

Type: minutes

Default value: 1440 minutes

--verbose
Optional

Indicates that you want verbose output from the retrieve operation.

Type: boolean

Aliases for `force:mdapi:retrieve:report`

```
force:mdapi:beta:retrieve:report
```

`org` Commands

Use the org commands to manage the orgs you use with Salesforce CLI. Create and delete scratch orgs, list your created and authorized orgs, and open orgs in your browser.

force:org:clone
Clone a sandbox org.

force:org:create
Create a scratch or sandbox org .

force:org:delete
Mark a scratch or sandbox org for deletion .

force:org:display
Get the description for the current or target org.

force:org:list
List all orgs you've created or authenticated to.

force:org:open
Open your default scratch org, or another specified org.

force:org:shape:create
Create a scratch org configuration (shape) based on the specified source org.

force:org:shape:delete
Delete all org shapes for a target org.

force:org:shape:list
List all org shapes you've created.

force:org:snapshot:create (Pilot)
Create a snapshot of a scratch org.

force:org:snapshot:delete (Pilot)
Delete a scratch org snapshot.

force:org:snapshot:get (Pilot)
Get details about a scratch org snapshot.

force:org:snapshot:list (Pilot)
List scratch org snapshots.

force:org:status
Report status of sandbox creation or clone and authenticate to it.

`force:org:clone`

Clone a sandbox org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org create sandbox command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.
- Changed flag name: Old name `--definitionfile`. New name: `--definition-file`.
- Changed flag name: Old name `--setalias`. New name: `--alias`.
- Changed flag name: Old name `--setdefaultusername`. New name: `--set-default`, with new short name `-d`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 9, 2023.

Help for `force:org:clone`

There are two ways to clone a sandbox: either specify a sandbox definition file or provide key=value pairs at the command line. Key-value pairs at the command-line override their equivalent sandbox definition file values. In either case, you must specify both the "SandboxName" and "SourceSandboxName" options to set the names of the new sandbox and the one being cloned, respectively.

Set the --targetusername (-u) parameter to a production org with sandbox licenses. The --type (-t) parameter is required and must be set to "sandbox".

Examples for `force:org:clone`

```
 sfdx force:org:clone -t sandbox -f config/dev-sandbox-def.json -u prodOrg -a MyDevSandbox
```

```
 sfdx force:org:clone -t sandbox SandboxName=NewClonedSandbox
SourceSandboxName=ExistingSandbox -u prodOrg -a MyDevSandbox
```

Command Syntax

`sfdx force:org:clone`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-t TYPE
[-f DEFINITIONFILE]
[-s]
[-a SETALIAS]
[-w WAIT]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-t | --type TYPE`

Required

Type of org to create.

Type: enum

Permissible values are: sandbox

-f | --definitionfile DEFINITIONFILE

Optional

Path to the sandbox definition file.

Type: filepath

-s | --setdefaultusername

Optional

Set the cloned org as your default.

Type: boolean

-a | --setalias SETALIAS

Optional

Alias for the cloned org.

Type: string

-w | --wait WAIT

Optional

Sets the streaming client socket timeout, in minutes. If the streaming client socket has no contact from the server for a number of minutes, the client exits. Specify a longer wait time if timeouts occur frequently.

Type: minutes

Default value: 6 minutes

`force:org:create`

Create a scratch or sandbox org .

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using one of these two equivalent `sf`-style commmands instead (one for scratch orgs and one for sandboxes):

- org create scratch
- org create sandbox

Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.
- Changed flag name: Old name `--clientid`. New name: `--client-id`.

- Changed flag name: Old name `--definitionfile`. New name: `--definition-file`.
- Changed flag name: Old name `--durationdays`. New name: `--duration-days`, with new short name `-y`.
- Changed flag name: Old name `--noancestors`. New name: `--no-ancestors`.
- Changed flag name: Old name `--nonamespace`. New name: `--no-namespace`, with new short name `-m`.
- Changed flag name: Old name `--setalias`. New name: `--alias`.
- Changed flag name: Old name `--setdefaultusername`. New name: `--set-default`, with new short name `-d`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 9, 2023.

Help for `force:org:create`

Creates a scratch org or a sandbox org using the values specified in a configuration file or key=value pairs that you specify on the command line. Values specified on the command line override values in the configuration file. Specify a configuration file or provide key=value pairs while creating a scratch org or a sandbox. When creating scratch orgs, —targetdevhubusername (-v) must be a Dev Hub org. When creating sandboxes, the --targetusername (-u) must be a production org with sandbox licenses. The —type (-t) is required if creating a sandbox.

Examples for `force:org:create`

```
sfdx force:org:create -f config/enterprise-scratch-def.json -a MyScratchOrg
```

```
sfdx force:org:create edition=Developer -a MyScratchOrg -s -v devHub
```

```
 sfdx force:org:create -f config/enterprise-scratch-def.json -a ScratchOrgWithOverrides
username=testuser1@mycompany.org
```

```
sfdx force:org:create -t sandbox -f config/dev-sandbox-def.json -a MyDevSandbox -u prodOrg
```

Command Syntax

`sfdx force:org:create`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-t TYPE]
[-f DEFINITIONFILE]
[-n]
[-c]
[-i CLIENTID]
[-s]
[-a SETALIAS]
[-w WAIT]
```

```
[-d DURATIONDAYS]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`
Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-t | --type TYPE`
Optional

Type of org to create.

Type: enum

Permissible values are: scratch, sandbox

Default value: scratch

`-f | --definitionfile DEFINITIONFILE`
Optional

Path to an org definition file.

Type: filepath

`-n | --nonamespace`
Optional

Create the scratch org with no namespace.

Type: boolean

`-c | --noancestors`
Optional

Do not include second-generation package ancestors in the scratch org.

Type: boolean

`-i | --clientid CLIENTID`
Optional

Connected app consumer key; not supported for sandbox org creation.

Type: string

`-s | --setdefaultusername`
Optional

Set the created org as the default username.

Type: boolean

`-a | --setalias SETALIAS`
Optional

Alias for the created org.

Type: string

`-w | --wait WAIT`
Optional

The streaming client socket timeout (in minutes).

Type: minutes

Default value: 6 minutes

`-d | --durationdays DURATIONDAYS`
Optional

Duration of the scratch org (in days) (default:7, min:1, max:30).

Type: integer

Default value: 7

Aliases for `force:org:create`

```
force:org:beta:create
```

`force:org:delete`

Mark a scratch or sandbox org for deletion .

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using one of these two equivalent `sf`-style commmands instead (one for scratch orgs and one for sandboxes):

- org delete scratch

- org delete sandbox

Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 9, 2023.

Help for `force:org:delete`

To mark the org for deletion without being prompted to confirm, specify --noprompt.

Examples for `force:org:delete`

```
sfdx force:org:delete -u me@my.org
```

```
sfdx force:org:delete -u MyOrgAlias -p
```

Command Syntax

`sfdx force:org:delete`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-p]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-p | --noprompt`

Optional

No prompt to confirm deletion.

Type: boolean

`force:org:display`

Get the description for the current or target org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org display command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 9, 2023.

Help for `force:org:display`

Output includes your access token, client Id, connected status, org ID, instance URL, username, and alias, if applicable.

Use --verbose to include the SFDX auth URL. WARNING: The SFDX auth URL contains sensitive information, such as a refresh token that can be used to access an org. Don't share or distribute this URL or token.

Including --verbose displays the sfdxAuthUrl property only if you authenticated to the org using auth:web:login (not auth:jwt:grant)

Examples for `force:org:display`

```
sfdx force:org:display
```

```
sfdx force:org:display -u me@my.org
```

```
sfdx force:org:display -u TestOrg1 --json
```

```
sfdx force:org:display -u TestOrg1 --json > tmp/MyOrgDesc.json
```

Command Syntax

`sfdx force:org:display`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[--verbose]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`--verbose`

Optional

Emit additional command output to stdout.

Type: boolean

`force:org:list`

List all orgs you've created or authenticated to.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org list command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.
- Changed flag name: Old name `--skipconnectionstatus`. New name: `--skip-connection-status`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 9, 2023.

Examples for `force:org:list`

```
sfdx force:org:list
```

```
sfdx force:org:list --verbose --json
```

```
sfdx force:org:list --verbose --json > tmp/MyOrgList.json
```

Command Syntax

`sfdx force:org:list`

`[--json]`

`[--loglevel LOGLEVEL]`

`[--verbose]`

`[--all]`

`[--clean]`

`[-p]`

`[--skipconnectionstatus]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`--verbose`

Optional

Emit additional command output to stdout.

Type: boolean

`--all`

Optional

Include expired, deleted, and unknown-status scratch orgs.

Type: boolean

`--clean`

Optional

Remove all local org authorizations for non-active scratch orgs. Use auth:logout to remove non-scratch orgs.

Type: boolean

`-p | --noprompt`

Optional

Do not prompt for confirmation.

Type: boolean

`--skipconnectionstatus`

Optional

Skip retrieving the connection status of non-scratch orgs.

Type: boolean

`force:org:open`

Open your default scratch org, or another specified org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org open command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.

- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.
- Changed flag name: Old name `--urlonly`. New name: `--url-only`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 9, 2023.

Help for `force:org:open`

To open a specific page, specify the portion of the URL after "https://MyDomainName.my.salesforce.com/" as --path.

For example, specify "--path lightning" to open Lightning Experience, or specify "--path /apex/YourPage" to open a Visualforce page.

To generate a URL but not launch it in your browser, specify --urlonly.

To open in a specific browser, use the --browser parameter. Supported browsers are "chrome", "edge", and "firefox". If you don't specify --browser, the org opens in your default browser.

Examples for `force:org:open`

```
sfdx force:org:open
```

```
sfdx force:org:open -u me@my.org
```

```
sfdx force:org:open -u MyTestOrg1
```

```
sfdx force:org:open -r -p lightning
```

```
sfdx force:org:open -u me@my.org -b firefox
```

Command Syntax

`sfdx force:org:open`

```
    [--json]
    [--loglevel LOGLEVEL]
    [-u TARGETUSERNAME]
    [--apiversion APIVERSION]
    [-b BROWSER]
    [-p PATH]
    [-r]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-b | --browser BROWSER`

Optional

Browser where the org opens.

Type: string

Permissible values are: chrome, edge, firefox

`-p | --path PATH`

Optional

Navigation URL path.

Type: string

`-r | --urlonly`

Optional

Display navigation URL, but don't launch browser.

Type: boolean

`force:org:shape:create`

Create a scratch org configuration (shape) based on the specified source org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org create shape command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 26, 2023.

Examples for `force:org:shape:create`

```
sfdx force:org:shape:create -u me@my.org
```

```
sfdx force:org:shape:create -u me@my.org --json --loglevel debug
```

Command Syntax

`sfdx force:org:shape:create`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`force:org:shape:delete`

Delete all org shapes for a target org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org delete shape command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 26, 2023.

Examples for `force:org:shape:delete`

```
sfdx force:org:shape:delete -u me@my.org
```

```
sfdx force:org:shape:delete -u MyOrgAlias -p
```

```
sfdx force:org:shape:delete -u me@my.org --json
```

```
sfdx force:org:shape:delete -u me@my.org -p --json > tmp/MyOrgShapeDelete.json
```

Command Syntax

`sfdx force:org:shape:delete`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-p]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

-u | --targetusername TARGETUSERNAME

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

--apiversion APIVERSION

Optional

Override the API version used for API requests made by this command.

Type: string

-p | --noprompt

Optional

Do not prompt for confirmation.

Type: boolean

`force:org:shape:list`

List all org shapes you've created.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org list shape command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Removed flag: `--verbose`
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 26, 2023.

Examples for `force:org:shape:list`

```
sfdx force:org:shape:list
```

```
sfdx force:org:shape:list --json
```

```
sfdx force:org:shape:list --json > tmp/MyOrgShapeList.json
```

Command Syntax

`sfdx force:org:shape:list`

[`--json`]

[`--loglevel LOGLEVEL`]

[`--verbose`]

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`--verbose`

Optional

Emit additional command output to stdout.

Type: boolean

`force:org:snapshot:create` (Pilot)

Create a snapshot of a scratch org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org create snapshot command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.
- Changed flag name: Old name `--snapshotname`. New name: `--name`.
- Changed flag name: Old name `--sourceorg`. New name: `--source-org`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 26, 2023.

Note: We provide the `force:org:snapshot:create` command to selected customers through an invitation-only pilot program that requires agreement to specific terms and conditions. Pilot programs are subject to change, and we can't guarantee acceptance. The `force:org:snapshot:create` command isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features. You can provide feedback and suggestions for the `force:org:snapshot:create` command in the W19 Pilot: Scratch Org Snapshots group in the Trailblazer Community.

Help for `force:org:snapshot:create`

A snapshot is a point-in-time copy of a scratch org. The copy is stored in Salesforce and referenced by its unique name in a scratch org definition file.

Use "sfdx force:org:snapshot:get" to get details, including status, about a snapshot creation request.

To create a scratch org from a snapshot, include the "snapshot" entry (instead of "edition") in the scratch org definition file and set it to the name of the snapshot. Then use "sfdx force:org:create" to create the scratch org.

Examples for `force:org:snapshot:create`

Create a snapshot called "Dependencies" using the source scratch org ID:

```
 sfdx force:org:snapshot:create --sourceorg 00Dxx0000000000 --snapshotname Dependencies
--description 'Contains PackageA v1.1.0'
```

Create a snapshot called "NightlyBranch" using the source scratch org username:

```
 sfdx force:org:snapshot:create -o myuser@myorg -n NightlyBranch -d 'Contains PkgA v2.1.0
 and PkgB 3.3.0'
```

Command Syntax

`sfdx force:org:snapshot:create`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-v TARGETDEVHUBUSERNAME]`

`[--apiversion APIVERSION]`

`-o SOURCEORG`

`-n SNAPSHOTNAME`

`[-d DESCRIPTION]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-o | --sourceorg SOURCEORG`

Required

The org ID, or a locally authenticated username or alias, of the scratch org to snapshot.

Type: string

`-n | --snapshotname SNAPSHOTNAME`

Required

The unique name of the snapshot. Use this name to create scratch orgs from the snapshot.

Type: string

`-d | --description DESCRIPTION`

Optional

A description of the snapshot. Use this description to document the contents of the snapshot.

We suggest that you include a reference point, such as a version control system tag or commit ID.'

Type: string

`force:org:snapshot:delete` (Pilot)

Delete a scratch org snapshot.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org delete snapshot command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 26, 2023.

Note: We provide the `force:org:snapshot:delete` command to selected customers through an invitation-only pilot program that requires agreement to specific terms and conditions. Pilot programs are subject to change, and we can't guarantee acceptance. The `force:org:snapshot:delete` command isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features. You can provide feedback and suggestions for the `force:org:snapshot:delete` command in the W19 Pilot: Scratch Org Snapshots group in the Trailblazer Community.

Help for `force:org:snapshot:delete`

Dev Hub admins can delete any snapshot, while users can delete only theirs unless a Dev Hub admin gives the user Modify All permissions.

Examples for `force:org:snapshot:delete`

Delete a snapshot from the default Dev Hub using the snapshot ID:

```
sfdx force:org:snapshot:delete --snapshot 0Oo...
```

Delete a snapshot from the specified Dev Hub using the snapshot name:

```
sfdx force:org:snapshot:delete -s BaseSnapshot -v SnapshotDevHub
```

Command Syntax

`sfdx force:org:snapshot:delete`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
-s SNAPSHOT
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-s | --snapshot SNAPSHOT`

Required

The name or ID (starts with 0Oo) of the snapshot to delete.

Type: string

`force:org:snapshot:get` (Pilot)

Get details about a scratch org snapshot.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org get snapshot command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 26, 2023.

Note: We provide the `force:org:snapshot:get` command to selected customers through an invitation-only pilot program that requires agreement to specific terms and conditions. Pilot programs are subject to change, and we can't guarantee acceptance. The `force:org:snapshot:get` command isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features. You can provide feedback and suggestions for the `force:org:snapshot:get` command in the W19 Pilot: Scratch Org Snapshots group in the Trailblazer Community.

Help for `force:org:snapshot:get`

Snapshot creation can take a while. Use this command with the snapshot name or ID to check its creation status. Once the status changes to Active, you can use the snapshot to create scratch orgs.

To create a snapshot, use the "sfdx force:org:snapshot:create" command. To retrieve a list of all snapshots, use "sfdx force:org:snapshot:list".

Examples for `force:org:snapshot:get`

Get snapshot details using its ID:

```
sfdx force:org:snapshot:get --snapshot 0Oo...
```

Get snapshot details using its name:

```
sfdx force:org:snapshot:get -s Dependencies
```

Command Syntax

`sfdx force:org:snapshot:get`

```
[--json]
[--loglevel LOGLEVEL]
```

```
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
-s SNAPSHOT
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`
Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-s | --snapshot SNAPSHOT`
Required

The name or ID (starts with 0Oo) of the snapshot to retrieve.

Type: string

`force:org:snapshot:list` (Pilot)

List scratch org snapshots.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org list snapshot command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 26, 2023.

Note: We provide the `force:org:snapshot:list` command to selected customers through an invitation-only pilot program that requires agreement to specific terms and conditions. Pilot programs are subject to change, and we can't guarantee acceptance. The `force:org:snapshot:list` command isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can't guarantee general availability within any particular time frame or at all. Make your purchase decisions only on the basis of generally available products and features. You can provide feedback and suggestions for the `force:org:snapshot:list` command in the W19 Pilot: Scratch Org Snapshots group in the Trailblazer Community.

Help for `force:org:snapshot:list`

You can view all the snapshots in a Dev Hub that you have access to. If you're an admin, you can see all snapshots associated with the Dev Hub org. If you're a user, you can see only your snapshots unless a Dev Hub admin gives you View All permissions.

To create a snapshot, use the "sfdx force:org:snapshot:create" command. To get details about a snapshot request, use "sfdx force:org:snapshot:get".

Examples for `force:org:snapshot:list`

List snapshots in the default Dev Hub:

```
sfdx force:org:snapshot:list
```

List snapshots in the Dev Hub with the specified username:

```
sfdx force:org:snapshot:list -v OtherDevHub@example.com
```

Command Syntax

`sfdx force:org:snapshot:list`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

-v | --targetdevhubusername TARGETDEVHUBUSERNAME
Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

--apiversion APIVERSION
Optional

Override the API version used for API requests made by this command.

Type: string

force:org:status

Report status of sandbox creation or clone and authenticate to it.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org resume sandbox command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-hub-org`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.
- Changed flag name: Old name `--setdefaultusername`. New name: `--set-default`, with new short name `-d`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 9, 2023.

Help for force:org:status

Use this command to check the status of your sandbox creation or clone and, if the sandbox is ready, authenticate to it.

Use the --wait (-w) parameter to specify the number of minutes that the command waits for the sandbox creation or clone to complete before returning control of the terminal to you.

Set the --targetusername (-u) parameter to the username or alias of the production org that contains the sandbox license.

Examples for force:org:status

```
sfdx force:org:status --sandboxname DevSbx1 --setalias MySandbox -u prodOrg
```

```
sfdx force:org:status --sandboxname DevSbx1 --wait 45 --setdefaultusername -u prodOrg
```

Command Syntax

```
sfdx force:org:status
   [--json]
   [--loglevel LOGLEVEL]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   -n SANDBOXNAME
   [-s]
   [-a SETALIAS]
   [-w WAIT]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-n | --sandboxname SANDBOXNAME`
Required

Name of the sandbox org to check status for.

Type: string

`-s | --setdefaultusername`
Optional

Set the created or cloned org as your default.

Type: boolean

`-a | --setalias SETALIAS`

Optional

Alias for the created or cloned org.

Type: string

`-w | --wait WAIT`

Optional

Number of minutes to wait while polling for status.

Type: minutes

Default value: 6 minutes

`package` Commands

Use the package commands to develop and install packages.

force:package:create

Create a package.

force:package:delete

Delete a package.

force:package:install

Install a package in the target org.

force:package:install:report

Retrieve the status of a package installation request.

force:package:installed:list

List the org's installed packages.

force:package:list

List all packages in the Dev Hub org.

force:package:uninstall

Uninstall a second-generation package from the target org.

force:package:uninstall:report

Retrieve status of package uninstall request.

force:package:update

Update package details.

force:package:version:create

Creates a package version in the Dev Hub org.

force:package:version:create:list

List package version creation requests.

force:package:version:create:report

Retrieve details about a package version creation request.

force:package:version:delete

Delete a package version.

force:package:version:displayancestry
Display the ancestry tree for a 2GP managed package version.

force:package:version:list
List all package versions in the Dev Hub org.

force:package:version:promote
Promote a package version to released.

force:package:version:report
Retrieve details about a package version in the Dev Hub org.

force:package:version:update
Update a package version.

`force:package:create`

Create a package.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package create command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--errornotificationusername`. New name: `--error-notification-username`.
- Changed flag name: Old name `--nonamespace`. New name: `--no-namespace`.
- Changed flag name: Old name `--orgdependent`. New name: `--org-dependent`.
- Changed flag name: Old name `--packagetype`. New name: `--package-type`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:create`

First, use this command to create a package. Then create a package version.

If you don't have a namespace defined in your sfdx-project.json file, use --nonamespace.

Your --name value must be unique within your namespace.

Run 'sfdx force:package:list' to list all packages in the Dev Hub org.

Examples for `force:package:create`

```
sfdx force:package:create -n YourPackageName -t Unlocked -r force-app
```

```
sfdx force:package:create -n YourPackageName -d "Your Package Descripton" -t Unlocked -r
force-app
```

Command Syntax

`sfdx force:package:create`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
-n NAME
-t PACKAGETYPE
[-d DESCRIPTION]
[-e]
-r PATH
[--orgdependent]
[-o ERRORNOTIFICATIONUSERNAME]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

-n | --name NAME

Required

Name of the package to create.

Type: string

-t | --packagetype PACKAGETYPE

Required

Package type for the package.

The options for package type are Managed and Unlocked (Managed=DeveloperManagedSubscriberManaged, Unlocked=DeveloperControlledSubscriberEditable).

These options determine upgrade and editability rules.

Type: enum

Permissible values are: Managed, Unlocked

-d | --description DESCRIPTION

Optional

Description of the package.

Type: string

-e | --nonamespace

Optional

Creates the package with no namespace. Available only for unlocked packages. Useful when you're migrating an existing org to packages. But, use a namespaced package for new metadata.

Type: boolean

-r | --path PATH

Required

The path to the directory that contains the contents of the package.

Type: directory

--orgdependent

Optional

Package depends on unpackaged metadata in the installation org. Applies to unlocked packages only.

Use Source Tracking in Sandboxes to develop your org-dependent unlocked package.

For more information, see "Create Org-Dependent Unlocked Packages" in the Salesforce DX Developer Guide.

Type: boolean

-o | --errornotificationusername ERRORNOTIFICATIONUSERNAME

Optional

An active Dev Hub org user designated to receive email notifications for unhandled Apex exceptions, and install, upgrade, or uninstall failures associated with your package.

Type: string

Aliases for `force:package:create`

```
force:package:beta:create
```

`force:package:delete`

Delete a package.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package delete command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:delete`

Specify the ID or alias of the package you want to delete.

Delete unlocked and second-generation managed packages. Before you delete a package, first delete all associated package versions.

Examples for `force:package:delete`

```
sfdx force:package:delete -p "Your Package Alias"
```

```
sfdx force:package:delete -p 0Ho...
```

Command Syntax

`sfdx force:package:delete`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
[-n]
-p PACKAGE
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

-v | --targetdevhubusername TARGETDEVHUBUSERNAME

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

--apiversion APIVERSION

Optional

Override the API version used for API requests made by this command.

Type: string

-n | --noprompt

Optional

Don't prompt before deleting the package.

Type: boolean

-p | --package PACKAGE

Required

The ID (starts with 0Ho) or alias of the package to delete.

Type: string

Aliases for `force:package:delete`

```
force:package:beta:delete
```

`force:package:install`

Install a package in the target org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package install command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--apexcompile`. New name: `--apex-compile`.
- Changed flag name: Old name `--installationkey`. New name: `--installation-key`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.

- Changed flag name: Old name `--publishwait`. New name: `--publish-wait`.
- Changed flag name: Old name `--securitytype`. New name: `--security-type`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--upgradetype`. New name: `--upgrade-type`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:install`

Supply the ID of the package version to install. The package installs in your default target org unless you supply the username for a different target org.

For package upgrades, to specify options for component deprecation or deletion of removed components, include an --upgradetype value. To delete components that can be safely deleted and deprecate the others, specify --upgradetype Mixed (the default). To deprecate all removed components, specify --upgradetype DeprecateOnly. To delete all removed components, except for custom objects and custom fields, that don't have dependencies, specify --upgradetype Delete. (Note: This option can result in the loss of data that is associated with the deleted components.) The default is Mixed.

Examples for `force:package:install`

```
sfdx force:package:install --package 04t... -u me@example.com
```

```
sfdx force:package:install --package awesome_package_alias
```

```
sfdx force:package:install --package "Awesome Package Alias"
```

```
sfdx force:package:install --package 04t... -t DeprecateOnly
```

Command Syntax

`sfdx force:package:install`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-w WAIT]
[-k INSTALLATIONKEY]
[-b PUBLISHWAIT]
[-r]
-p PACKAGE
[-a APEXCOMPILE]
[-s SECURITYTYPE]
[-t UPGRADETYPE]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-w | --wait WAIT`
Optional

Maximum number of minutes to wait for installation status. The default is 0.

Type: minutes

Default value: 0 minutes

`-k | --installationkey INSTALLATIONKEY`
Optional

Installation key for installing a key-protected package. The default is null.

Type: string

`-b | --publishwait PUBLISHWAIT`
Optional

Maximum number of minutes to wait for the Subscriber Package Version ID to become available in the target org before canceling the install request. The default is 0.

Type: minutes

Default value: 0 minutes

`-r | --noprompt`
Optional

Allows the following without an explicit confirmation response: 1) Remote Site Settings and Content Security Policy websites to send or receive data, and 2) --upgradetype Delete to proceed.

Type: boolean

`-p | --package PACKAGE`
Required

The ID (starts with 04t) or alias of the package version to install.

Type: string

`-a | --apexcompile APEXCOMPILE`
Optional

Applies to unlocked packages only. Specifies whether to compile all Apex in the org and package, or only the Apex in the package.

For package installs into production orgs, or any org that has Apex Compile on Deploy enabled, the platform compiles all Apex in the org after the package install or upgrade operation completes.

This approach assures that package installs and upgrades don't impact the performance of an org, and is done even if --apexcompile package is specified.

Type: enum

Permissible values are: all, package

Default value: all

`-s | --securitytype SECURITYTYPE`
Optional

Security access type for the installed package.

Type: enum

Permissible values are: AllUsers, AdminsOnly

Default value: AdminsOnly

`-t | --upgradetype UPGRADETYPE`
Optional

For package upgrades, specifies whether to mark all removed components as deprecated (DeprecateOnly), to delete removed components that can be safely deleted and deprecate the others (Mixed), or to delete all removed components, except for custom objects and custom fields, that don't have dependencies (Delete). The default is Mixed. Can specify DeprecateOnly or Delete only for unlocked package upgrades.

Type: enum

Permissible values are: DeprecateOnly, Mixed, Delete

Default value: Mixed

Aliases for `force:package:install`

```
force:package:beta:install
```

`force:package:install:report`

Retrieve the status of a package installation request.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package install report command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--requestid`. New name: `--request-id`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Examples for `force:package:install:report`

```
sfdx force:package:install:report -i 0Hf...
```

```
sfdx force:package:install:report -i 0Hf... -u me@example.com
```

Command Syntax

`sfdx force:package:install:report`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-i REQUESTID
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-i | --requestid REQUESTID`

Required

The ID of the package install request you want to check.

Type: id

Aliases for `force:package:install:report`

```
force:package:beta:install:report
```

`force:package:installed:list`

List the org's installed packages.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package installed list command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Examples for `force:package:installed:list`

```
sfdx force:package:installed:list
```

```
sfdx force:package:installed:list -u me@example.com
```

Command Syntax

`sfdx force:package:installed:list`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

Aliases for `force:package:installed:list`

```
force:package:beta:installed:list
```

`force:package:list`

List all packages in the Dev Hub org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package list command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:list`

You can view the namespace, IDs, and other details for each package.

Examples for `force:package:list`

```
sfdx force:package:list -v devhub@example.com
```

```
sfdx force:package:list -v devhub@example.com --verbose
```

Command Syntax

`sfdx force:package:list`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-v TARGETDEVHUBUSERNAME]`

`[--apiversion APIVERSION]`

`[--verbose]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v` | `--targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`--verbose`

Optional

Displays extended package details.

Type: boolean

Aliases for `force:package:list`

```
force:package:beta:list
```

`force:package:uninstall`

Uninstall a second-generation package from the target org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package uninstall command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:uninstall`

Specify the package ID for a second-generation package.

To list the org's installed packages, run "sfdx force:package:installed:list".

To uninstall a first-generation package, from Setup, enter Installed Packages in the Quick Find box, then select Installed Packages.

Examples for `force:package:uninstall`

```
sfdx force:package:uninstall -p 04t... -u me@example.com
```

```
sfdx force:package:uninstall -p undesirable_package_alias
```

```
sfdx force:package:uninstall -p "Undesirable Package Alias"
```

Command Syntax

`sfdx force:package:uninstall`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-w WAIT]
-p PACKAGE
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-w | --wait WAIT`

Optional

Maximum number of minutes to wait for uninstall status. The default is 0.

Type: minutes

Default value: 0 minutes

`-p | --package PACKAGE`

Required

The ID (starts with 04t) or alias of the package version to uninstall.

Type: string

Aliases for `force:package:uninstall`

```
force:package:beta:uninstall
```

`force:package:uninstall:report`

Retrieve status of package uninstall request.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package uninstall report command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--requestid`. New name: `--request-id`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Examples for `force:package:uninstall:report`

```
sfdx force:package:uninstall:report -i 06y...
```

```
sfdx force:package:uninstall:report -i 06y... -u me@example.com
```

Command Syntax

`sfdx force:package:uninstall:report`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`-i REQUESTID`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-i | --requestid REQUESTID`

Required

The ID of the package uninstall request you want to check.

Type: id

Aliases for `force:package:uninstall:report`

```
force:package:beta:uninstall:report
```

`force:package:update`

Update package details.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package update command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--errornotificationusername`. New name: `--error-notification-username`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:update`

Specify a new value for each option you want to update.

Run "sfdx force:package:list" to list all packages in the Dev Hub org.

Examples for `force:package:update`

```
sfdx force:package:update -p "Your Package Alias" -n "New Package Name"
```

```
sfdx force:package:update -p 0Ho... -d "New Package Description"
```

Command Syntax

`sfdx force:package:update`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
-p PACKAGE
[-n NAME]
[-d DESCRIPTION]
[-o ERRORNOTIFICATIONUSERNAME]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`
Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-p | --package PACKAGE`
Required

The ID (starts with 0Ho) or alias of the package to update.

Type: string

`-n | --name NAME`
Optional

New name of the package.

Type: string

`-d | --description DESCRIPTION`
Optional

New description of the package.

Type: string

`-o | --errornotificationusername ERRORNOTIFICATIONUSERNAME`
Optional

An active Dev Hub org user designated to receive email notifications for unhandled Apex exceptions, and install, upgrade, or uninstall failures associated with your package.

Type: string

Aliases for `force:package:update`

```
force:package:beta:update
```

`force:package:version:create`

Creates a package version in the Dev Hub org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package version create command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--codecoverage`. New name: `--code-coverage`.
- Changed flag name: Old name `--definitionfile`. New name: `--definition-file`.
- Changed flag name: Old name `--installationkey`. New name: `--installation-key`.
- Changed flag name: Old name `--installationkeybypass`. New name: `--installation-key-bypass`.
- Changed flag name: Old name `--postinstallscript`. New name: `--post-install-script`.
- Changed flag name: Old name `--postinstallurl`. New name: `--post-install-url`.
- Changed flag name: Old name `--releasenotesurl`. New name: `--releasenotes-url`.
- Changed flag name: Old name `--skipancestorcheck`. New name: `--skip-ancestor-check`.
- Changed flag name: Old name `--skipvalidation`. New name: `--skip-validation`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.
- Changed flag name: Old name `--uninstallscript`. New name: `--uninstall-script`.
- Changed flag name: Old name `--versiondescription`. New name: `--version-description`.
- Changed flag name: Old name `--versionname`. New name: `--version-name`.
- Changed flag name: Old name `--versionnumber`. New name: `--version-number`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:version:create`

The package version is based on the package contents in the specified directory.

To retrieve details about a package version create request, including status and package version ID (04t), run "sfdx force:package:version:create:report -i 08c...".

We recommend that you specify the --installationkey parameter to protect the contents of your package and to prevent unauthorized installation of your package.

To list package version creation requests in the org, run "sfdx force:package:version:create:list".

To promote a package version to released, you must use the --codecoverage parameter. The package must also meet the code coverage requirements. This requirement applies to both managed and unlocked packages.

We don't calculate code coverage for org-dependent unlocked packages, or for package versions that specify --skipvalidation.

Examples for `force:package:version:create`

```
sfdx force:package:version:create -d common -k password123
```

```
sfdx force:package:version:create -p "Your Package Alias" -k password123
```

```
sfdx force:package:version:create -p 0Ho... -k password123
```

```
sfdx force:package:version:create -d common -k password123 --skipvalidation
```

Command Syntax

`sfdx force:package:version:create`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-v TARGETDEVHUBUSERNAME]`

`[--apiversion APIVERSION]`

`[-b BRANCH]`

`[-c]`

`[-f DEFINITIONFILE]`

`[-k INSTALLATIONKEY]`

`[-x]`

`[-p PACKAGE]`

`[-d PATH]`

`[--postinstallscript POSTINSTALLSCRIPT]`

`[--postinstallurl POSTINSTALLURL]`

`[--releasenotesurl RELEASENOTESURL]`

`[--skipancestorcheck]`

`[--skipvalidation]`

`[-t TAG]`

`[--uninstallscript UNINSTALLSCRIPT]`

`[-e VERSIONDESCRIPTION]`

`[-a VERSIONNAME]`

`[-n VERSIONNUMBER]`

`[-w WAIT]`

`[--language LANGUAGE]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

--loglevel LOGLEVEL

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

-v | --targetdevhubusername TARGETDEVHUBUSERNAME

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

--apiversion APIVERSION

Optional

Override the API version used for API requests made by this command.

Type: string

-b | --branch BRANCH

Optional

Name of the branch in your source control system that the package version is based on.

Type: string

-c | --codecoverage

Optional

Calculate and store the code coverage percentage by running the Apex tests included in this package version. Before you can promote and release a managed or unlocked package version, the Apex code must meet a minimum 75% code coverage requirement. We don't calculate code coverage for org-dependent unlocked packages or for package versions that specify --skipvalidation.

Type: boolean

-f | --definitionfile DEFINITIONFILE

Optional

The path to a definition file similar to scratch org definition file that contains the list of features and org preferences that the metadata of the package version depends on.

Type: filepath

-k | --installationkey INSTALLATIONKEY

Optional

Installation key for creating the key-protected package. Either an --installationkey value or the --installationkeybypass flag is required.

Type: string

-x | --installationkeybypass

Optional

Bypasses the installation key requirement. If you bypass this requirement, anyone can install your package. Either an --installationkey value or the --installationkeybypass flag is required.

Type: boolean

`-p | --package PACKAGE`

Optional

The ID (starts with 0Ho) or alias of the package to create a version of.

Type: string

`-d | --path PATH`

Optional

The path to the directory that contains the contents of the package.

Type: directory

`--postinstallscript POSTINSTALLSCRIPT`

Optional

Applies to managed packages only. The post-install script name. The post-install script is an Apex class within this package that is run in the installing org after installations or upgrades of this package version.

Type: string

`--postinstallurl POSTINSTALLURL`

Optional

The post-install instructions URL. The contents of the post-installation instructions URL are displayed in the UI after installation of the package version.

Type: url

`--releasenotesurl RELEASENOTESURL`

Optional

The release notes URL. This link is displayed in the package installation UI to provide release notes for this package version to subscribers.

Type: url

`--skipancestorcheck`

Optional

Override ancestry requirements, which allows you to specify a package ancestor that isn't the highest released package version.

Type: boolean

`--skipvalidation`

Optional

Skips validation of dependencies, package ancestors, and metadata during package version creation. Skipping validation reduces the time it takes to create a new package version, but you can promote only validated package versions. Skipping validation can suppress important errors that can surface at a later stage. You can specify skip validation or code coverage, but not both. Code coverage is calculated during validation.

Type: boolean

`-t | --tag TAG`

Optional

The package version's tag.

Type: string

`--uninstallscript UNINSTALLSCRIPT`

Optional

Applies to managed packages only. The uninstall script name. The uninstall script is an Apex class within this package that is run in the installing org after uninstallations of this package.

Type: string

`-e | --versiondescription VERSIONDESCRIPTION`

Optional

The description of the package version to be created. Overrides the sfdx-project.json value.

Type: string

`-a | --versionname VERSIONNAME`

Optional

The name of the package version to be created. Overrides the sfdx-project.json value.

Type: string

`-n | --versionnumber VERSIONNUMBER`

Optional

The version number of the package version to be created. Overrides the sfdx-project.json value.

Type: string

`-w | --wait WAIT`

Optional

The number of minutes to wait for the package version to be created.

Type: minutes

Default value: 0 minutes

`--language LANGUAGE`

Optional

The language for the package. Specify the language using a language code listed under "Supported Languages" in Salesforce Help.

If no language is specified, the language defaults to the language of the Dev Hub user who created the package.

Only applies to orgs running API version 57.0 or higher.

Type: string

Aliases for `force:package:version:create`

```
force:package:beta:version:create
```

`force:package:version:create:list`

List package version creation requests.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package version create list command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--createdlastdays`. New name: `--created-last-days`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:version:create:list`

Shows the details of each request to create a package version in the Dev Hub org.

All filter parameters are applied using the AND logical operator (not OR).

To get information about a specific request, run "sfdx force:package:version:create:report" and supply the request ID.

Examples for `force:package:version:create:list`

```
sfdx force:package:version:create:list
```

```
sfdx force:package:version:create:list --createdlastdays 3
```

```
sfdx force:package:version:create:list --status Error
```

```
sfdx force:package:version:create:list -s InProgress
```

```
sfdx force:package:version:create:list -c 3 -s Success
```

Command Syntax

`sfdx force:package:version:create:list`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
[-c CREATEDLASTDAYS]
[-s STATUS]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v` | `--targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-c` | `--createdlastdays CREATEDLASTDAYS`

Optional

Filters the list based on the specified maximum number of days since the request was created (starting at 00:00:00 of first day to now; 0 for today).

Type: number

`-s` | `--status STATUS`

Optional

Filters the list based on the status of version creation requests.

Type: enum

Permissible values are: Queued, InProgress, Success, Error

Aliases for `force:package:version:create:list`

```
force:package:beta:version:create:list
```

`force:package:version:create:report`

Retrieve details about a package version creation request.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package version create report command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--packagecreaterequestid`. New name: `--package-create-request-id`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:version:create:report`

Specify the request ID for which you want to view details. If applicable, the command displays errors related to the request.

To show all requests in the org, run "sfdx force:package:version:create:list".

Examples for `force:package:version:create:report`

```
sfdx force:package:version:create:report -i 08c...
```

```
sfdx force:package:version:create:report -i 08c... -v devhub@example.com
```

Command Syntax

`sfdx force:package:version:create:report`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-v TARGETDEVHUBUSERNAME]`

`[--apiversion APIVERSION]`

`-i PACKAGECREATEREQUESTID`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-i | --packagecreaterequestid PACKAGECREATEREQUESTID`

Required

The ID (starts with 08c) of the package version creation request you want to display.

Type: id

Aliases for `force:package:version:create:report`

```
force:package:beta:version:create:report
```

`force:package:version:delete`

Delete a package version.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package version delete command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:version:delete`

Specify the ID or alias of the package version you want to delete.

Examples for `force:package:version:delete`

```
sfdx force:package:version:delete -p "Your Package Alias"
```

```
sfdx force:package:version:delete -p 04t...
```

Command Syntax

`sfdx force:package:version:delete`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
[-n]
-p PACKAGE
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`
Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-n | --noprompt`
Optional

don't prompt before deleting the package version

Type: boolean

`-p | --package PACKAGE`
Required

The ID (starts with 04t) or alias of the package version to delete.

Type: string

Aliases for `force:package:version:delete`

```
force:package:beta:version:delete
```

`force:package:version:displayancestry`

Display the ancestry tree for a 2GP managed package version.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package version displayancestry command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--dotcode`. New name: `--dot-code`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Examples for **`force:package:version:displayancestry`**

```
sfdx force:package:version:displayancestry -p package_version_alias
```

```
sfdx force:package:version:displayancestry -p package_version_alias --dotcode
```

```
sfdx force:package:version:displayancestry -p OHo...
```

```
sfdx force:package:version:displayancestry -p 04t...
```

Command Syntax

`sfdx force:package:version:displayancestry`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
-p PACKAGE
[--dotcode]
[--verbose]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-p | --package PACKAGE`

Required

The ID or alias of the package or package version to display ancestry for. If you specify a package ID (starts with 0Ho) or alias, the ancestor tree for every package version associated with the package ID is displayed.

If you specify a package version (starts with 04t) or alias, the ancestry tree of the specified package version is displayed.

Type: string

`--dotcode`

Optional

Displays the ancestry tree in DOT code. You can use the DOT code output in graph visualization software to create tree visualizations.

Type: boolean

`--verbose`

Optional

Displays both the package version ID (starts with 04t) and the version number (major.minor.patch.build) in the ancestry tree.

Type: boolean

Aliases for `force:package:version:displayancestry`

```
force:package:beta:version:displayancestry
```

`force:package:version:list`

List all package versions in the Dev Hub org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package version list command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--createdlastdays`. New name: `--created-last-days`.
- Changed flag name: Old name `--modifiedlastdays`. New name: `--modified-last-days`.
- Changed flag name: Old name `--orderby`. New name: `--order-by`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:version:list`

Displays details of each package version in the org.

Use --concise or --verbose to display limited or additional details, respectively.

All filter parameters are applied using the AND logical operator (not OR).

Examples for `force:package:version:list`

```
 sfdx force:package:version:list --verbose --createdlastdays 3 --released --orderby
PatchVersion
```

```
 sfdx force:package:version:list --packages 0Ho000000000000,0Ho000000000001 --released
--modifiedlastdays 0
```

```
 sfdx force:package:version:list --released
```

```
 sfdx force:package:version:list --concise --modifiedlastdays 0
```

```
 sfdx force:package:version:list --concise -c 3 -r
```

```
 sfdx force:package:version:list --packages exp-mgr,exp-mgr-util --released
--modifiedlastdays 0
```

Command Syntax

`sfdx force:package:version:list`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-v TARGETDEVHUBUSERNAME]`

`[--apiversion APIVERSION]`

`[-c CREATEDLASTDAYS]`

`[--concise]`

`[-m MODIFIEDLASTDAYS]`

`[-p PACKAGES]`

`[-r]`

`[-o ORDERBY]`

`[--verbose]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`
Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-c | --createdlastdays CREATEDLASTDAYS`
Optional

Filters the list based on the specified maximum number of days since the request was created (starting at 00:00:00 of first day to now; 0 for today).

Type: number

`--concise`
Optional

Displays limited package version details.

Type: boolean

`-m | --modifiedlastdays MODIFIEDLASTDAYS`
Optional

Lists the items modified in the specified last number of days, starting at 00:00:00 of first day to now. Use 0 for today.

Type: number

`-p | --packages PACKAGES`
Optional

Filters results on the specified comma-delimited packages (aliases or 0Ho IDs).

Type: array

`-r | --released`
Optional

Displays released versions only (IsReleased=true).

Type: boolean

`-o | --orderby ORDERBY`
Optional

Orders the list by the specified package version fields.

Type: array

`--verbose`
Optional

Displays extended package version details.

Type: boolean

Aliases for `force:package:version:list`

```
force:package:beta:version:list
```

`force:package:version:promote`

Promote a package version to released.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package version promote command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:version:promote`

Supply the ID or alias of the package version you want to promote. Promotes the package version to released status.

Examples for `force:package:version:promote`

```
sfdx force:package:version:promote -p 04t...
```

```
sfdx force:package:version:promote -p awesome_package_alias
```

```
sfdx force:package:version:promote -p "Awesome Package Alias"
```

Command Syntax

```
sfdx force:package:version:promote
   [--json]
   [--loglevel LOGLEVEL]
   [-v TARGETDEVHUBUSERNAME]
   [--apiversion APIVERSION]
   -p PACKAGE
   [-n]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-p | --package PACKAGE`

Required

The ID (starts with 04t) or alias of the package version to promote.

Type: string

`-n | --noprompt`

Optional

Do not prompt to confirm setting the package version as released.

Type: boolean

Aliases for `force:package:version:promote`

```
force:package:beta:version:promote
```

`force:package:version:report`

Retrieve details about a package version in the Dev Hub org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package version report command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for **`force:package:version:report`**

To update package version values, run "sfdx force:package:version:update".

Examples for **`force:package:version:report`**

```
sfdx force:package:version:report -p 04t...
```

```
sfdx force:package:version:report -p "Your Package Alias"
```

Command Syntax

`sfdx force:package:version:report`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
-p PACKAGE
[--verbose]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-p | --package PACKAGE`
Required

The ID (starts with 04t) or alias of the package to retrieve details for.

Type: string

`--verbose`
Optional

Displays extended package version details.

Type: boolean

Aliases for `force:package:version:report`

```
force:package:beta:version:report
```

`force:package:version:update`

Update a package version.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package version update command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--installationkey`. New name: `--installation-key`.
- Changed flag name: Old name `--targetdevhubusername`. New name: `--target-dev-hub`.
- Changed flag name: Old name `--versiondescription`. New name: `--version-description`.
- Changed flag name: Old name `--versionname`. New name: `--version-name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package:version:update`

Specify a new value for each option you want to update.

To display details about a package version, run "sfdx force:package:version:report".

Examples for `force:package:version:update`

```
sfdx force:package:version:update -p "Your Package Alias" -k password123
```

```
sfdx force:package:version:update -p 04t... -b main -t 'Release 1.0.7'
```

```
sfdx force:package:version:update -p 04t... -e "New Package Version Description"
```

Command Syntax

`sfdx force:package:version:update`

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[--apiversion APIVERSION]
-p PACKAGE
[-a VERSIONNAME]
[-e VERSIONDESCRIPTION]
[-b BRANCH]
[-t TAG]
[-k INSTALLATIONKEY]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-p | --package PACKAGE`
Required

The ID (starts with 04t) or alias of the package to update a version of.

Type: string

`-a | --versionname VERSIONNAME`
Optional

The new package version name.

Type: string

`-e | --versiondescription VERSIONDESCRIPTION`
Optional

The new package version description.

Type: string

`-b | --branch BRANCH`
Optional

The new package version branch.

Type: string

`-t | --tag TAG`
Optional

The new package version tag.

Type: string

`-k | --installationkey INSTALLATIONKEY`
Optional

The new installation key for the key-protected package. The default is null.

Type: string

Aliases for `force:package:version:update`

```
force:package:beta:version:update
```

`package1` Commands

Use the package1 commands to create and view first-generation package versions in your Dev Hub org.

force:package1:version:create
Create a first-generation package version in the release org.

force:package1:version:create:get
Retrieve the status of a package version creation request.

force:package1:version:display
Display details about a first-generation package version.

force:package1:version:list
List package versions for the specified first-generation package or for the org.

`force:package1:version:create`

Create a first-generation package version in the release org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package1 version create command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--installationkey`. New name: `--installation-key`.
- Changed flag name: Old name `--managedrelease`. New name: `--managed-release`.
- Changed flag name: Old name `--packageid`. New name: `--package-id`.
- Changed flag name: Old name `--packageversionid`. New name: `--package-version-id`.
- Changed flag name: Old name `--postinstallurl`. New name: `--post-install-url`.
- Changed flag name: Old name `--releasenotesurl`. New name: `--release-notes-url`.
- Changed flag name: Old name `--requestid`. New name: `--request-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:package1:version:create`

The package version is based on the contents of the specified metadata package. Omit -m if you want to create an unmanaged package version.

Command Syntax

```
sfdx force:package1:version:create
   [--json]
   [--loglevel LOGLEVEL]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   -i PACKAGEID
   -n NAME
   [-d DESCRIPTION]
   [-v VERSION]
   [-m]
   [-r RELEASENOTESURL]
   [-p POSTINSTALLURL]
   [-k INSTALLATIONKEY]
```

```
[-w WAIT]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-i | --packageid PACKAGEID`

Required

ID of the metadata package (starts with 033) of which you're creating a new version.

Type: id

`-n | --name NAME`

Required

Package version name.

Type: string

`-d | --description DESCRIPTION`

Optional

Package version description.

Type: string

`-v | --version VERSION`

Optional

Package version in major.minor format, for example, 3.2.

Type: string

`-m | --managedreleased`

Optional

Creates a managed package version. To create a beta version, don’t include this parameter.

Type: boolean

`-r | --releasenotesurl RELEASENOTESURL`

Optional

The release notes URL. This link is displayed in the package installation UI to provide release notes for this package version to subscribers.

Type: url

`-p | --postinstallurl POSTINSTALLURL`

Optional

The post-install instructions URL. The contents of the post-installation instructions URL are displayed in the UI after installation of the package version.

Type: url

`-k | --installationkey INSTALLATIONKEY`

Optional

Installation key for creating the key-protected package. The default is null.

Type: string

`-w | --wait WAIT`

Optional

Minutes to wait for the package version to be created. The default is 2 minutes.

Type: minutes

Aliases for `force:package1:version:create`

```
force:package1:beta:version:create
```

`force:package1:version:create:get`

Retrieve the status of a package version creation request.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, command and its flags continue to work the same as before, and any scripts that use the command won’t break. However, we recommend that you start using the equivalent `sf`-style package1 version create get command instead. Here’s how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--requestid`. New name: `--request-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Examples for `force:package1:version:create:get`

```
sfdx force:package1:version:create:get -i 0HD...
```

```
sfdx force:package1:version:create:get -i 0HD... -u devhub@example.com
```

Command Syntax

`sfdx force:package1:version:create:get`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`-i REQUESTID`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u` | `--targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-i` | `--requestid REQUESTID`

Required

The ID of the PackageUploadRequest.

Type: id

Aliases for `force:package1:version:create:get`

```
force:package1:beta:version:create:get
```

`force:package1:version:display`

Display details about a first-generation package version.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package1 version display command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--packageversionid`. New name: `--package-version-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Command Syntax

```
sfdx force:package1:version:display
    [--json]
    [--loglevel LOGLEVEL]
    [-u TARGETUSERNAME]
    [--apiversion APIVERSION]
    -i PACKAGEVERSIONID
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-i | --packageversionid PACKAGEVERSIONID`

Required

ID (starts with 04t) of the metadata package version whose details you want to display.

Type: id

Aliases for `force:package1:version:display`

```
force:package1:beta:version:display
```

`force:package1:version:list`

List package versions for the specified first-generation package or for the org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style package1 version list command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--packageid`. New name: `--package-id`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Command Syntax

`sfdx force:package1:version:list`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`[-i PACKAGEID]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-i | --packageid PACKAGEID`

Optional

Metadata package ID (starts with 033) whose package versions you want to list. If not specified, shows all versions for all packages (managed and unmanaged) in the org.

Type: id

Aliases for `force:package1:version:list`

```
force:package1:beta:version:list
```

`project` Commands

Use the project commands to set up a Salesforce DX project.

force:project:create

Creates a Salesforce DX project in the specified directory or the current working directory. The command creates the necessary configuration files and folders.

`force:project:create`

Creates a Salesforce DX project in the specified directory or the current working directory. The command creates the necessary configuration files and folders.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project generate command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--projectname`. New name: `--name`.
- Changed flag name: Old name `--defaultpackagedir`. New name: `--default-package-dir`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for **`force:project:create`**

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
sfdx force:project:create --projectname mywork
```

```
sfdx force:project:create --projectname mywork --defaultpackagedir myapp
```

```
sfdx force:project:create --projectname mywork --defaultpackagedir myapp --manifest
```

```
sfdx force:project:create --projectname mywork --template empty
```

Command Syntax

```
sfdx force:project:create
    [--json]
    [--loglevel LOGLEVEL]
    -n PROJECTNAME
    [-t TEMPLATE]
    [-d OUTPUTDIR]
    [-s NAMESPACE]
    [-p DEFAULTPACKAGEDIR]
    [-x]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-n | --projectname PROJECTNAME`
Required

The name for the new project. Any valid folder name is accepted.

Type: string

`-t | --template TEMPLATE`
Optional

The template to use to create the project. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: standard, empty, analytics

Default value: standard

`-d | --outputdir OUTPUTDIR`
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`-s | --namespace NAMESPACE`
Optional

The namespace associated with this project and any connected scratch orgs.

Type: string

`-p | --defaultpackagedir DEFAULTPACKAGEDIR`
Optional

The default package directory name. Metadata items such as classes and Lightning bundles are placed inside this folder.

Type: string

Default value: force-app

`-x | --manifest`
Optional

Generates a default manifest (package.xml) for fetching Apex, Visualforce, Lightning components, and static resources.

Type: boolean

`schema` Commands

Use the schema commands to view information about the standard and custom objects in your org.

force:schema:sobject:describe
Displays the metadata for a standard or custom object.

force:schema:sobject:list
List all objects of a specified category.

`force:schema:sobject:describe`

Displays the metadata for a standard or custom object.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style subject describe command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjecttype`. New name: `--sobject`.
- Changed flag name: Old name `--usetoolingapi`. New name: `--use-tooling-api`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 12, 2023.

Examples for `force:schema:sobject:describe`

```
sfdx force:schema:sobject:describe -s Account
```

```
sfdx force:schema:sobject:describe -s MyObject__c
```

```
sfdx force:schema:sobject:describe -s ApexClass -t
```

Command Syntax

`sfdx force:schema:sobject:describe`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`-s SOBJECTTYPE`

`[-t]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-s | --sobjecttype SOBJECTTYPE`

Required

The API name of the object to describe.

Type: string

`-t | --usetoolingapi`

Optional

Execute with Tooling API.

Type: boolean

`force:schema:sobject:list`

List all objects of a specified category.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style subject list command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sobjecttype`. New name: `--sobject`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 12, 2023.

Examples for `force:schema:sobject:list`

```
sfdx force:schema:sobject:list -c all
```

```
sfdx force:schema:sobject:list -c custom
```

```
sfdx force:schema:sobject:list -c standard
```

Command Syntax

`sfdx force:schema:sobject:list`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-c SOBJECTTYPECATEGORY]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-c | --sobjecttypecategory SOBJECTTYPECATEGORY`

Optional

The type of objects to list (all|custom|standard).

Type: string

Default value: ALL

`source` Commands

Use the source commands to push and pull source to and from source-tracked orgs, to deploy and retrieve source to and from orgs, to see synchronization changes between your project and source-tracked orgs, and to convert your source to the metadata format for Metadata API deployments.

force:source:convert
Convert source into Metadata API format .

force:source:delete
Delete source from your project and from a non-source-tracked org.

force:source:deploy
Deploy source to an org.

force:source:deploy:cancel
Cancel a source deployment.

force:source:deploy:report
Check the status of a metadata deployment .

force:source:ignored:list
Check your local project package directories for forceignored files.

force:source:manifest:create
Create a project manifest that lists the metadata components you want to deploy or retrieve .

force:source:open
Edit a Lightning Page with Lightning App Builder.

force:source:pull
Pull source from the scratch org to the project.

force:source:push
Push source to a scratch org from the project.

force:source:retrieve
Retrieve source from an org .

force:source:status
List local changes and/or changes in a scratch org.

force:source:tracking:clear
Clear all local source tracking information.

force:source:tracking:reset
Reset local and remote source tracking.

`force:source:convert`

Convert source into Metadata API format .

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project convert source command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- New flag: `--api-version`
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--packagename`. New name: `--package-name`.
- Changed flag name: Old name `--rootdir`. New name: `--root-dir`.
- Changed flag name: Old name `--sourcepath`. New name: `--source-dir`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for **`force:source:convert`**

Converts source-formatted files into metadata that you can deploy using Metadata API.

To convert source-formatted files into the metadata format, so that you can deploy them using Metadata API,

run "sfdx force:source:convert". Then deploy the metadata using "sfdx force:mdapi:deploy".

To convert Metadata API–formatted files into the source format, run "sfdx force:mdapi:convert".

To specify a package name that includes spaces, enclose the name in single quotes.

Examples for **`force:source:convert`**

```
sfdx force:source:convert -r path/to/source
```

```
sfdx force:source:convert -r path/to/source -d path/to/outputdir -n 'My Package'
```

Command Syntax

```
sfdx force:source:convert
   [--json]
   [--loglevel LOGLEVEL]
   [-r ROOTDIR]
   [-d OUTPUTDIR]
   [-n PACKAGENAME]
   [-x MANIFEST]
   [-p SOURCEPATH]
   [-m METADATA]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-r | --rootdir ROOTDIR`

Optional

A source directory other than the default package to convert.

Type: directory

`-d | --outputdir OUTPUTDIR`

Optional

Output directory to store the Metadata API–formatted files in.

Type: directory

Default value: metadataPackage_1676579552285

`-n | --packagename PACKAGENAME`

Optional

Name of the package to associate with the metadata-formatted files.

Type: string

`-x | --manifest MANIFEST`

Optional

The complete path to the manifest (package.xml) file that specifies the metadata types to convert.

If you specify this parameter, don't specify --metadata or --sourcepath.

Type: string

`-p | --sourcepath SOURCEPATH`

Optional

A comma-separated list of paths to the local source files to convert. The supplied paths can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all metadata types in the directory and its sub-directories).

If you specify this parameter, don't specify --manifest or --metadata.

Type: array

`-m | --metadata METADATA`

Optional

Comma-separated list of metadata component names to convert.

Type: array

`force:source:delete`

Delete source from your project and from a non-source-tracked org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project delete source command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--checkonly`. New name: `--check-only`.
- Changed flag name: Old name `--forceoverwrite`. New name: `--force-overwrite`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.
- Changed flag name: Old name `--sourcepath`. New name: `--source-dir`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--testlevel`. New name: `--test-level`.
- Changed flag name: Old name `--tracksource`. New name: `--track-source`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:source:delete`

IMPORTANT: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

Use this command to delete components from orgs that don't have source tracking.

To remove deleted items from scratch orgs, which have change tracking, use "sfdx force:source:push".

Examples for `force:source:delete`

```
sfdx force:source:delete -m <metadata>
```

```
sfdx force:source:delete -p path/to/source
```

Command Syntax

`sfdx force:source:delete`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-c]
```

```
[-w WAIT]
[-l TESTLEVEL]
[-r]
[-m METADATA]
[-p SOURCEPATH]
[-t]
[-f]
[--verbose]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-c | --checkonly`

Optional

Validates the deleted metadata and runs all Apex tests, but prevents the deletion from being saved to the org.

If you change a field type from Master-Detail to Lookup or vice versa, that change isn't supported when using the --checkonly parameter to test a deletion (validation). This kind of change isn't supported for test deletions to avoid the risk of data loss or corruption. If a change that isn't supported for test deletions is included in a deletion package, the test deletion fails and issues an error.

If your deletion package changes a field type from Master-Detail to Lookup or vice versa, you can still validate the changes prior to deploying to Production by performing a full deletion to another test Sandbox. A full deletion includes a validation of the changes as part of the deletion process.

Note: A Metadata API deletion that includes Master-Detail relationships deletes all detail records in the Recycle Bin in the following cases.

1. For a deletion with a new Master-Detail field, soft delete (send to the Recycle Bin) all detail records before proceeding to delete the Master-Detail field, or the deletion fails. During the deletion, detail records are permanently deleted from the Recycle Bin and cannot be recovered.

2. For a deletion that converts a Lookup field relationship to a Master-Detail relationship, detail records must reference a master record or be soft-deleted (sent to the Recycle Bin) for the deletion to succeed. However, a successful deletion permanently deletes any detail records in the Recycle Bin.

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you.

Type: minutes

Default value: 33 minutes

`-l | --testlevel TESTLEVEL`

Optional

Specifies which level of deployment tests to run. Valid values are:

NoTestRun—No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial orgs. This test level is the default for development environments.

RunLocalTests—All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default for production deployments that include Apex classes or triggers.

RunAllTestsInOrg—All tests in your org are run, including tests of managed packages.

If you don't specify a test level, the default behavior depends on the contents of your deployment package. For more information, see "Running Tests in a Deployment" in the Metadata API Developer Guide.

Type: enum

Permissible values are: NoTestRun, RunLocalTests, RunAllTestsInOrg

Default value: NoTestRun

`-r | --noprompt`

Optional

Do not prompt for delete confirmation.

Type: boolean

`-m | --metadata METADATA`

Optional

A comma-separated list of names of metadata components to delete from your project and your org.

If you specify this parameter, don't specify --sourcepath.

Type: array

`-p | --sourcepath SOURCEPATH`

Optional

A comma-separated list of paths to the local metadata to delete. The supplied paths can be a single file (in which case the operation is applied to only one file) or a folder (in which case the operation is applied to all metadata types in the directory and its sub-directories).

If you specify this parameter, don't specify --metadata.

Type: array

`-t | --tracksource`

Optional

If the delete succeeds, update the source tracking information, similar to push.

Type: boolean

`-f | --forceoverwrite`

Optional

Ignore conflict warnings and overwrite changes to the org.

Type: boolean

`--verbose`

Optional

Emit additional command output to stdout.

Type: boolean

`force:source:deploy`

Deploy source to an org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project deploy start command instead.

Use this table to map the flags between the old and new commands. The mapping isn't always one-to-one; see the Notes column for more information.

`force:source:deploy` **Flag**	**Equivalent** `project deploy start` **Flag**	**Notes**
`-c`, `--checkonly`	`--dry-run`, no short name.	You can also use `project deploy validate`
`-f`, `--forceoverwrite`	`-c`, `--ignore-conflicts`	Note the new long and short flag name.
`-x`, `--manifest`	`-x`, `--manifest`	
`-m`, `--metadata`	`-m`, `--metadata`	
`-o`, `--ignoreerrors`	`-r`, `--ignore-errors`	Note the new short flag name.
`-g`, `--ignorewarnings`	`-g`, `--ignore-warnings`	
`-p`, `--sourcepath`	`-d`, `--source-dir`	Note the new short flag name.
`-l`, `--testlevel`	`-l`, `--test-level`	
`-u`, `--targetusername`	`-o`, `--target-org`	Note the new short flag name.
`-t`, `--tracksource`	No equivalent.	The `project deploy start` command always keeps track of your

`force:source:deploy` **Flag**	**Equivalent** `project deploy start` **Flag**	**Notes**
		source if the org is enabled for source-tracking. If you don't want to use source tracking, create an org that doesn't have source tracking enabled.
`-q`, `--validateddeployrequestid`	No equivalent.	Use the `project deploy validate` and `project deploy quick --job-id` commands.
`-r`, `--runtests`	`-t`, `--tests`	
`-w`, `--wait`	`-w`, `--wait`	
`--apiversion`	`-a`, `--api-version`	
`--concise`	`--concise`	
`--coverageformatters`	`--coverage-formatters`	
`--json`	`--json`	
`--junit`	`--junit`	
`--loglevel`	`No equivalent. Use the SF_LOG_LEVEL environment variable instead.`	
`--predestructivechanges`	`--pre-destructive-changes`	
`--postdestructivechanges`	`--post-destructive-changes`	
`--purgeondelete`	`--purge-on-delete`	
`--resultsdir`	`--results-dir`	
`--soapdeploy`	`No equivalent.`	Deploys use SOAP API by default. To use REST API, set the `org-metadata-rest-deploy` config variable or SF_ORG_METADATA_REST_DEPLOY environment variable.
`--verbose`	`--verbose`	

Here are some examples to help you update your old commands. This `sfdx`-style command:

```
sfdx force:source:deploy --metadata "ApexClass,CustomObject" --testlevel
RunSpecifiedTests \
--runtests MyTests --targetusername my-scratch
```

Looks like this using the equivalent `sf`-style command:

```
sf project deploy start --metadata ApexClass --metadata CustomObject \
--test-level RunSpecifiedTests --tests MyTests --target-org my-scratch
```

This `sfdx`-style command:

```
sfdx force:source:deploy --manifest package.xml --predestructivechanges
destructiveChangesPre.xml \
--check-only
```

Looks like this using the equivalent `sf`-style command:

```
sf project deploy start --manifest package.xml --pre-destructive-changes
destructiveChangesPre.xml \
--dry-run
```

This `sfdx`-style command:

```
sfdx force:source:deploy \
--sourcepath "path/to/objects/MyCustomObject/fields/MyField.field-meta.xml,
path/to/apex/classes"
```

Looks like this using the equivalent `sf`-style command:

```
sf project deploy start --source-dir
path/to/objects/MyCustomObject/fields/MyField.field-meta.xml \
--source-dir path/to/apex/classes
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:source:deploy`

IMPORTANT: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

Use this command to deploy source (metadata that's in source format) to an org.

To take advantage of change tracking with scratch orgs, use "sfdx force:source:push".

To deploy metadata that's in metadata format, use "sfdx force:mdapi:deploy".

The source you deploy overwrites the corresponding metadata in your org. This command does not attempt to merge your source with the versions in your org.

To run the command asynchronously, set --wait to 0, which immediately returns the job ID. This way, you can continue to use the CLI.

To check the status of the job, use force:source:deploy:report.

If the comma-separated list you're supplying contains spaces, enclose the entire comma-separated list in one set of double quotes. On Windows, if the list contains commas, also enclose the entire list in one set of double quotes.

If you use the --manifest, --predestructivechanges, or --postdestructivechanges parameters, run the force:source:manifest:create command to easily generate the different types of manifest files.

Examples for `force:source:deploy`

To deploy the source files in a directory:

```
sfdx force:source:deploy -p path/to/source
```

To deploy a specific Apex class and the objects whose source is in a directory:

```
sfdx force:source:deploy -p "path/to/apex/classes/MyClass.cls,path/to/source/objects"
```

To deploy source files in a comma-separated list that contains spaces:

```
sfdx force:source:deploy -p "path/to/objects/MyCustomObject/fields/MyField.field-meta.xml,
path/to/apex/classes"
```

To deploy all Apex classes:

```
sfdx force:source:deploy -m ApexClass
```

To deploy a specific Apex class:

```
sfdx force:source:deploy -m ApexClass:MyApexClass
```

To deploy a specific Apex class and update source tracking files :

```
sfdx force:source:deploy -m ApexClass:MyApexClass --tracksource
```

To deploy all custom objects and Apex classes:

```
sfdx force:source:deploy -m "CustomObject,ApexClass"
```

To deploy all Apex classes and two specific profiles (one of which has a space in its name):

```
sfdx force:source:deploy -m "ApexClass, Profile:My Profile, Profile: AnotherProfile"
```

To deploy all components listed in a manifest:

```
sfdx force:source:deploy -x path/to/package.xml
```

To run the tests that aren't in any managed packages as part of a deployment:

```
sfdx force:source:deploy -m ApexClass -l RunLocalTests
```

To check whether a deployment would succeed (to prepare for Quick Deploy):

```
sfdx force:source:deploy -m ApexClass -l RunAllTestsInOrg -c
```

To deploy an already validated deployment (Quick Deploy):

```
sfdx force:source:deploy -q 0Af9A00000FTM6pSAH`
```

To run a destructive operation before the deploy occurs:

```
 sfdx force:source:deploy --manifest package.xml --predestructivechanges
destructiveChangesPre.xml
```

To run a destructive operation after the deploy occurs:

```
 sfdx force:source:deploy --manifest package.xml --postdestructivechanges
destructiveChangesPost.xml
```

Command Syntax

```
sfdx force:source:deploy
   [--json]
   [--loglevel LOGLEVEL]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   [-c]
   [--soapdeploy]
   [-w WAIT]
   [-l TESTLEVEL]
   [-r RUNTESTS]
   [-o]
   [-g]
   [--purgeondelete]
   [-q VALIDATEDDEPLOYREQUESTID]
   [--verbose]
   [-m METADATA]
   [-p SOURCEPATH]
   [-x MANIFEST]
   [--predestructivechanges PREDESTRUCTIVECHANGES]
   [--postdestructivechanges POSTDESTRUCTIVECHANGES]
   [-t]
   [-f]
   [--resultsdir RESULTSDIR]
   [--coverageformatters COVERAGEFORMATTERS]
   [--junit]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-c | --checkonly`

Optional

Validates the deployed metadata and runs all Apex tests, but prevents the deployment from being saved to the org.

If you change a field type from Master-Detail to Lookup or vice versa, that change isn't supported when using the --checkonly parameter to test a deployment (validation). This kind of change isn't supported for test deployments to avoid the risk of data loss or corruption. If a change that isn't supported for test deployments is included in a deployment package, the test deployment fails and issues an error.

If your deployment package changes a field type from Master-Detail to Lookup or vice versa, you can still validate the changes prior to deploying to Production by performing a full deployment to another test Sandbox. A full deployment includes a validation of the changes as part of the deployment process.

Note: A Metadata API deployment that includes Master-Detail relationships deletes all detail records in the Recycle Bin in the following cases.

1. For a deployment with a new Master-Detail field, soft delete (send to the Recycle Bin) all detail records before proceeding to deploy the Master-Detail field, or the deployment fails. During the deployment, detail records are permanently deleted from the Recycle Bin and cannot be recovered.

2. For a deployment that converts a Lookup field relationship to a Master-Detail relationship, detail records must reference a master record or be soft-deleted (sent to the Recycle Bin) for the deployment to succeed. However, a successful deployment permanently deletes any detail records in the Recycle Bin.

Type: boolean

`--soapdeploy`

Optional

Deploy metadata with SOAP API instead of REST API.

Type: boolean

`-w | --wait WAIT`

Optional

Number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you.

Type: minutes

Default value: 33 minutes

`-l | --testlevel TESTLEVEL`

Optional

Specifies which level of deployment tests to run. Valid values are:

NoTestRun—No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial orgs. This test level is the default for development environments.

RunSpecifiedTests—Runs only the tests that you specify in the --runtests option. Code coverage requirements differ from the default coverage requirements when using this test level. Executed tests must comprise a minimum of 75% code coverage for each class and trigger in the deployment package. This coverage is computed for each class and trigger individually and is different than the overall coverage percentage.

RunLocalTests—All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default for production deployments that include Apex classes or triggers.

RunAllTestsInOrg—All tests in your org are run, including tests of managed packages.

If you don't specify a test level, the default behavior depends on the contents of your deployment package. For more information, see "Running Tests in a Deployment" in the Metadata API Developer Guide.

Type: enum

Permissible values are: NoTestRun, RunSpecifiedTests, RunLocalTests, RunAllTestsInOrg

-r | --runtests RUNTESTS

Optional

Lists the Apex classes containing the deployment tests to run. Use this parameter when you set --testlevel to RunSpecifiedTests.

Type: array

-o | --ignoreerrors

Optional

Ignores the deploy errors, and continues with the deploy operation. The default is false. Keep this parameter set to false when deploying to a production org. If set to true, components without errors are deployed, and components with errors are skipped.

Type: boolean

-g | --ignorewarnings

Optional

If a warning occurs and ignoreWarnings is set to true, the success field in DeployMessage is true. When ignoreWarnings is set to false, success is set to false, and the warning is treated like an error.

Type: boolean

--purgeondelete

Optional

Specify that deleted components in the destructive changes manifest file are immediately eligible for deletion rather than being stored in the Recycle Bin.

Type: boolean

-q | --validateddeployrequestid VALIDATEDDEPLOYREQUESTID

Optional

Specifies the ID of a package with recently validated components to run a Quick Deploy. Deploying a validation helps you shorten your deployment time because tests aren't rerun. If you have a recent successful validation, you can deploy the validated components without running tests. A validation doesn't save any components in the org. You use a validation only to check the success or failure messages that you would receive with an actual deployment. To validate your components, add the -c | --checkonly flag when you run "sfdx force:mdapi:deploy". This flag sets the checkOnly="true" parameter for your deployment. Before deploying a recent validation, ensure that the following requirements are met:

1. The components have been validated successfully for the target environment within the last 10 days.

2. As part of the validation, Apex tests in the target org have passed.

3. Code coverage requirements are met.

- If all tests in the org or all local tests are run, overall code coverage is at least 75%, and Apex triggers have some coverage.

- If specific tests are run with the RunSpecifiedTests test level, each class and trigger that was deployed is covered by at least 75% individually.

Type: id

`--verbose`

Optional

Emit additional command output to stdout.

Type: boolean

`-m | --metadata METADATA`

Optional

A comma-separated list of names of metadata components to deploy to the org.

If you specify this parameter, don't specify --manifest or --sourcepath.

Type: array

`-p | --sourcepath SOURCEPATH`

Optional

A comma-separated list of paths to the local source files to deploy. The supplied paths can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all metadata types in the directory and its sub-directories).

If you specify this parameter, don't specify --manifest or --metadata.

Type: array

`-x | --manifest MANIFEST`

Optional

The complete path for the manifest (package.xml) file that specifies the components to deploy. All child components are included.

If you specify this parameter, don't specify --metadata or --sourcepath.

Type: filepath

`--predestructivechanges PREDESTRUCTIVECHANGES`

Optional

File path for a manifest (destructiveChangesPre.xml) of components to delete before the deploy.

Type: filepath

`--postdestructivechanges POSTDESTRUCTIVECHANGES`

Optional

File path for a manifest (destructiveChangesPost.xml) of components to delete after the deploy.

Type: filepath

`-t | --tracksource`

Optional

If the deploy succeeds, update source tracking information; doesn't delete locally deleted files from org unless you also specify --predestructivechanges or --postdestructivechanges.

Type: boolean

`-f | --forceoverwrite`

Optional

Ignore conflict warnings and overwrite changes to the org.

Type: boolean

`--resultsdir RESULTSDIR`

Optional

Output directory for code coverage and JUnit results; defaults to the deploy ID.

Type: directory

`--coverageformatters COVERAGEFORMATTERS`

Optional

Format of the code coverage results.

Type: array

`--junit`

Optional

Output JUnit test results.

Type: boolean

`force:source:deploy:cancel`

Cancel a source deployment.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project deploy cancel command instead.

Use this table to map the flags between the old and new commands.

`force:source:deploy:cancel` **Flag**	**Equivalent** `project deploy cancel` **Flag**	**Notes**
`-i`, `--jobid`	`-i`, `--job-id`	
`-u`, `--targetusername`	`-o`, `--target-org`	Note the new short flag name.
`-w`, `--wait`	`-w`, `--wait`	
`--apiversion`	`-a`, `--api-version`	
`--json`	`--json`	
`--loglevel`	`No equivalent. Use the SF_LOG_LEVEL environment variable instead.`	

Here's an example to help you update your old commands. This `sfdx`-style command:

```
sfdx force:source:deploy:cancel --wait 2 --jobid 1234
```

Looks like this using the equivalent `sf`-style command:

```
sf project deploy cancel --wait 2 --job-id 1234
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:source:deploy:cancel`

Use this command to cancel a specified asynchronous source deployment. You can also specify a wait time (in minutes) to check for updates to the canceled deploy status.

To run the command asynchronously, set --wait to 0, which immediately returns the job ID. This way, you can continue to use the CLI.

To check the status of the job, use force:source:deploy:report.

Examples for `force:source:deploy:cancel`

Deploy a directory of files to the org

```
sfdx force:source:deploy -d <directory>
```

Now cancel this deployment and wait two minutes

```
sfdx force:source:deploy:cancel -w 2
```

If you have multiple deployments in progress and want to cancel a specific one, specify the job ID

```
sfdx force:source:deploy:cancel -i <jobid>
```

Check the status of the cancel job

```
sfdx force:source:deploy:report
```

Command Syntax

`sfdx force:source:deploy:cancel`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-w WAIT]
[-i JOBID]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-w | --wait WAIT`

Optional

Number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you.

Type: minutes

Default value: 33 minutes

`-i | --jobid JOBID`

Optional

Job ID of the deployment you want to cancel; defaults to your most recent CLI deployment if not specified.

Type: id

`force:source:deploy:report`

Check the status of a metadata deployment .

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style commands.

This command (`force:source:deploy:report`) does more than just report: it also resumes a deployment, which is confusing. For this reason, we now provide these two new commands for each task, which is much more intuitive:

- project deploy report
- project deploy resume

Use this table to map the flags between the old and new commands.

`force:source:deploy:report` **Flag**	**Equivalent** `project deploy report` **or** `project deploy resume` **Flag**	**Notes**
`-i`, `--jobid`	`-i`, `--job-id`	
`-u`, `--targetusername`	No equivalent.	The `project deploy report` and `project deploy resume` commands use a job ID so a target org isn't needed.
`-w`, `--wait`	`-w`, `--wait`	
`--apiversion`	No equivalent.	API version isn't needed for this command.
`--coverageformatters`	`--coverage-formatters`	
`--json`	`--json`	
`--junit`	`--junit`	
`--loglevel`	`No equivalent. Use the SF_LOG_LEVEL environment variable instead.`	
`--resultsdir`	`--results-dir`	
`--verbose`	`--verbose`	

Here's an example to help you update your old commands. This `sfdx`-style command:

```
sfdx force:source:deploy:report --jobid 1234 --wait 10
```

Looks like this using the equivalent `sf`-style command:

```
sf project deploy report --job-id 1234 --wait 10
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:source:deploy:report`

Specify the job ID for the deploy you want to check. You can also specify a wait time (minutes) to check for updates to the deploy status.

Examples for `force:source:deploy:report`

Deploy a directory of files to the org

```
sfdx force:source:deploy -d <directory>
```

Now cancel this deployment and wait two minutes

```
sfdx force:source:deploy:cancel -w 2
```

If you have multiple deployments in progress and want to cancel a specific one, specify the job ID

```
sfdx force:source:deploy:cancel -i <jobid>
```

Check the status of the cancel job

```
sfdx force:source:deploy:report
```

Command Syntax

`sfdx force:source:deploy:report`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`[-w WAIT]`

`[-i JOBID]`

`[--verbose]`

`[--resultsdir RESULTSDIR]`

`[--coverageformatters COVERAGEFORMATTERS]`

`[--junit]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

-w | --wait WAIT

Optional

Number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you.

Type: minutes

Default value: 33 minutes

-i | --jobid JOBID

Optional

The job ID (asyncId) of the deployment you want to check. If not specified, the default value is the ID of the most recent metadata deployment you ran using Salesforce CLI. Use with -w to resume waiting.

Type: id

--verbose

Optional

Emit additional command output to stdout.

Type: boolean

--resultsdir RESULTSDIR

Optional

Output directory for code coverage and JUnit results; defaults to the deploy ID.

Type: directory

--coverageformatters COVERAGEFORMATTERS

Optional

Format of the code coverage results.

Type: array

--junit

Optional

Output JUnit test results.

Type: boolean

`force:source:ignored:list`

Check your local project package directories for forceignored files.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project list ignored command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--sourcepath`. New name: `--source-dir`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Command Syntax

`sfdx force:source:ignored:list`

```
[--json]
[--loglevel LOGLEVEL]
[-p SOURCEPATH]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-p | --sourcepath SOURCEPATH`

Optional

File or directory of files that the command checks for foreceignored files.

Type: filepath

`force:source:manifest:create`

Create a project manifest that lists the metadata components you want to deploy or retrieve .

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project generate manifest command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--fromorg`. New name: `--from-org`.
- Changed flag name: Old name `--includepackages`. New name: `--include-packages`.
- Changed flag name: Old name `--manifestname`. New name: `--name`.
- Changed flag name: Old name `--manifesttype`. New name: `--type`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--sourcepath`. New name: `--source-dir`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:source:manifest:create`

Create a manifest from a list of metadata components (--metadata) or from one or more local directories that contain source files (--sourcepath). You can specify either of these parameters, not both.

Use --manifesttype to specify the type of manifest you want to create. The resulting manifest files have specific names, such as the standard package.xml or destructiveChanges.xml to delete metadata. Valid values for this parameter, and their respective file names, are:

package : package.xml (default)

pre : destructiveChangesPre.xml

post : destructiveChangesPost.xml

destroy : destructiveChanges.xml

See https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/meta_deploy_deleting_files.htm for information about these destructive manifest files.

Use --manifestname to specify a custom name for the generated manifest if the pre-defined ones don’t suit your needs. You can specify either --manifesttype or --manifestname, but not both.

Examples for `force:source:manifest:create`

```
sfdx force:source:manifest:create -m ApexClass
```

```
sfdx force:source:manifest:create -m ApexClass:MyApexClass --manifesttype destroy
```

```
sfdx force:source:manifest:create --sourcepath force-app --manifestname myNewManifest
```

```
sfdx force:source:manifest:create --fromorg test@myorg.com --includepackages unlocked
```

Command Syntax

```
sfdx force:source:manifest:create
   [--json]
   [--loglevel LOGLEVEL]
   [--apiversion APIVERSION]
   [-m METADATA]
   [-p SOURCEPATH]
   [-n MANIFESTNAME]
   [-t MANIFESTTYPE]
   [-c INCLUDEPACKAGES]
   [--fromorg FROMORG]
   [-o OUTPUTDIR]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-m | --metadata METADATA`
Optional

Comma-separated list of names of metadata components to include in the manifest.

Type: array

`-p | --sourcepath SOURCEPATH`
Optional

Comma-separated list of paths to the local source files to include in the manifest.

Type: array

`-n | --manifestname MANIFESTNAME`
Optional

Name of a custom manifest file to create.

Type: string

`-t | --manifesttype MANIFESTTYPE`
Optional

Type of manifest to create; the type determines the name of the created file.

Type: enum

Permissible values are: pre, post, destroy, package

`-c | --includepackages INCLUDEPACKAGES`
Optional

Comma-separated list of package types (managed, unlocked) whose metadata is included in the manifest; by default, metadata in packages is ignored.

Type: array

`--fromorg FROMORG`
Optional

Username or alias of the org that contains the metadata components from which to build a manifest.

Type: string

`-o | --outputdir OUTPUTDIR`

Optional

Directory to save the created manifest.

Type: string

`force:source:open`

Edit a Lightning Page with Lightning App Builder.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org open --source-file command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--sourcefile`. New name: `--source-file`.
- Changed flag name: Old name `--urlonly`. New name: `--url-only`.

Here's an example to help you update your old commands. This `sfdx`-style command:

```
sfdx force:source:open --source-file
force-app/main/default/flexipages/Hello.flexipage-meta.xml \
--urlonly --targetusername myscratch
```

Looks like this using the equivalent `sf`-style command:

```
sf org open --source-path force-app/main/default/flexipages/Hello.flexipage-meta.xml
\
--url-only --target-org myscratch
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on March 23, 2023.

Help for `force:source:open`

Opens the specified Lightning Page in Lightning App Builder. Lightning Page files have the suffix .flexipage-meta.xml, and are stored in the flexipages directory.

If you specify a Visualforce page, which has a .page suffix, the page opens in your browser so you can preview it. If you specify a different type of file, this command opens your org's home page.

The file opens in your default browser.

If no browser-based editor is available for the selected file, this command opens your org's home page.

To generate a URL for the browser-based editor but not open the editor, use --urlonly.

Examples for `force:source:open`

```
sfdx force:source:open -f path/to/source
```

```
sfdx force:source:open -r -f path/to/source
```

```
sfdx force:source:open -f path/to/source -u my-user@my-org.com
```

Command Syntax

`sfdx force:source:open`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-f SOURCEFILE
[-r]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-f | --sourcefile SOURCEFILE`

Required

File to edit.

Type: filepath

-r | --urlonly

Optional

Generate a navigation URL; don't launch the editor.

Type: boolean

force:source:pull

Pull source from the scratch org to the project.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project retrieve start command instead.

Use this table to map the flags between the old and new commands.

`force:source:pull` Flag	Equivalent `project retrieve start` Flag	Notes
`-f, --forceoverwrite`	`-c, --ignore-conflicts`	Note the new long and short flag names.
`-u, --targetusername`	`-o, --target-org`	Note the new short flag name.
`-w, --wait`	`-w, --wait`	
`--apiversion`	`-a, --api-version`	
`--json`	`--json`	
`--loglevel`	`No equivalent. Use the SF_LOG_LEVEL environment variable instead.`	

Here's an example to help you update your old commands. This `sfdx`-style command:

```
sfdx force:source:pull --targetusername myscratch --forceoverwrite --wait 10
```

Looks like this using the equivalent `sf`-style command:

```
sf project retrieve start --target-org myscratch --ignore-conflicts --wait 10
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for **force:source:pull**

If the command detects a conflict, it displays the conflicts but does not complete the process. After reviewing the conflict, rerun the command with the --forceoverwrite parameter.

Command Syntax

`sfdx force:source:pull`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-f]
[-w WAIT]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-f | --forceoverwrite`

Optional

Ignore conflict warnings and overwrite changes to the project.

Type: boolean

`-w | --wait WAIT`

Optional

The number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you. The default is 33 minutes.

Type: minutes

Default value: 33 minutes

Aliases for `force:source:pull`

```
force:source:beta:pull
```

`force:source:push`

Push source to a scratch org from the project.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project deploy start command instead.

The `project deploy start` command doesn't support the `pushPackageDirectoriesSequentially` property of `sfdx-project.json`. The `force:source:push` command uses this property to deploy packages sequentially. If you need to deploy packages sequentially and in a specific order, use multiple `project deploy start` commands in the desired order.

Use this table to map the flags between the old and new commands.

`force:source:push` **Flag**	**Equivalent** `project retrieve start` **Flag**	**Notes**
`-f`, `--forceoverwrite`	`-c`, `--ignore-conflicts`	Note the new long and short flag names.
`-g`, `--ignorewarnings`	`-g`, `--ignore-warnings`	Note the new long and short flag names.
`-u`, `--targetusername`	`-o`, `--target-org`	Note the new short flag name.
`-w`, `--wait`	`-w`, `--wait`	
`--apiversion`	`-a`, `--api-version`	
`--json`	`--json`	
`--loglevel`	`No equivalent. Use the SF_LOG_LEVEL environment variable instead.`	
`--quiet`	No equivalent.	

Here's an example to help you update your old commands. This `sfdx`-style command:

```
sfdx force:source:push --targetusername myscratch --forceoverwrite --wait 10
```

Looks like this using the equivalent `sf`-style command:

```
sf project deploy start --target-org myscratch --ignore-conflicts --wait 10
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:source:push`

If the command detects a conflict, it displays the conflicts but does not complete the process. After reviewing the conflict, rerun the command with the --forceoverwrite parameter.

Command Syntax

`sfdx force:source:push`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-f]
[-w WAIT]
[-g]
[--quiet]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-f | --forceoverwrite`

Optional

Runs the push command even if conflicts exist. Changes in the project overwrite changes in the scratch org.

Type: boolean

-w | --wait WAIT

Optional

Number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you. The default is 33 minutes.

Type: minutes

Default value: 33 minutes

-g | --ignorewarnings

Optional

Completes the deployment even if warnings are generated.

Type: boolean

--quiet

Optional

Command does not output to stdout.

Type: boolean

Aliases for `force:source:push`

```
force:source:beta:push
```

`force:source:retrieve`

Retrieve source from an org .

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project retrieve start command instead.

Use this table to map the flags between the old and new commands. The mapping isn't always one-to-one; see the Notes column for more information.

`force:source:retrieve` **Flag**	**Equivalent** `project retrieve start` **Flag**	**Notes**
`-f, --forceoverwrite`	`-c, --ignore-conflicts`	Note the new long and short flag name.
`-x, --manifest`	`-x, --manifest`	
`-m, --metadata`	`-m, --metadata`	
`-n, --packagenames`	`-n, --package-name`	
`-r, --retrievetargetdir`	`-r, --output-dir`	
`-p, --sourcepath`	`-d, --source-dir`	Note the new short flag name.
`-u, --targetusername`	`-o, --target-org`	Note the new short flag name.

`force:source:retrieve` **Flag**	**Equivalent** `project retrieve start` **Flag**	**Notes**
`-t`, `--tracksource`	No equivalent.	The `project retrieve start` command always keeps track of your source if the org is enabled for source-tracking. If you don't want to use source tracking, create an org that doesn't have source tracking enabled.
`-w`, `--wait`	`-w`, `--wait`	
`--apiversion`	`-a`, `--api-version`	
`--json`	`--json`	
`--loglevel`	`No equivalent. Use the SF_LOG_LEVEL environment variable instead.`	
`--verbose`	`--verbose`	

Here are some examples to help you update your old commands. This `sfdx`-style command:

```
sfdx force:source:retrieve --metadata "ApexClass,CustomObject" --targetusername
my-scratch
```

Looks like this using the equivalent `sf`-style command:

```
sf project retrieve start --metadata ApexClass --metadata CustomObject --target-org
my-scratch
```

This `sfdx`-style command:

```
sfdx force:source:retrieve --packagenames MyPackage --manifest package.xml
```

Looks like this using the equivalent `sf`-style command:

```
sf project retrieve start --package-name MyPackage --manifest package.xml
```

This `sfdx`-style command:

```
sfdx force:source:retrieve \
--sourcepath "path/to/objects/MyCustomObject/fields/MyField.field-meta.xml,
path/to/apex/classes"
```

Looks like this using the equivalent `sf`-style command:

```
sf project retrieve start --source-dir
path/to/objects/MyCustomObject/fields/MyField.field-meta.xml \
--source-dir path/to/apex/classes
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:source:retrieve`

Use this command to retrieve source (metadata that's in source format) from an org.

To take advantage of change tracking with scratch orgs, use "sfdx force:source:pull".

To retrieve metadata that's in metadata format, use "sfdx force:mdapi:retrieve".

The source you retrieve overwrites the corresponding source files in your local project. This command does not attempt to merge the source from your org with your local source files.

If the comma-separated list you're supplying contains spaces, enclose the entire comma-separated list in one set of double quotes. On Windows, if the list contains commas, also enclose it in one set of double quotes.

Examples for `force:source:retrieve`

To retrieve the source files in a directory:

```
sfdx force:source:retrieve -p path/to/source
```

To retrieve a specific Apex class and the objects whose source is in a directory:

```
sfdx force:source:retrieve -p "path/to/apex/classes/MyClass.cls,path/to/source/objects"
```

To retrieve source files in a comma-separated list that contains spaces:

```
sfdx force:source:retrieve -p "path/to/objects/MyCustomObject/fields/MyField.field-meta.xml,
path/to/apex/classes"
```

To retrieve all Apex classes:

```
sfdx force:source:retrieve -m ApexClass
```

To retrieve a specific Apex class:

```
sfdx force:source:retrieve -m ApexClass:MyApexClass
```

To retrieve a specific Apex class and update source tracking files:

```
sfdx force:source:retrieve -m ApexClass:MyApexClass -t
```

To retrieve all custom objects and Apex classes:

```
sfdx force:source:retrieve -m "CustomObject,ApexClass"
```

To retrieve all Apex classes and two specific profiles (one of which has a space in its name):

```
sfdx force:source:retrieve -m "ApexClass, Profile:My Profile, Profile: AnotherProfile"
```

To retrieve all metadata components listed in a manifest:

```
sfdx force:source:retrieve -x path/to/package.xml
```

To retrieve metadata from a package or multiple packages:

```
sfdx force:source:retrieve -n MyPackageName
```

```
sfdx force:source:retrieve -n "Package1, PackageName With Spaces, Package3"
```

To retrieve all metadata from a package and specific components that aren't in the package, specify both -n | --packagenames and one other scoping parameter:

```
sfdx force:source:retrieve -n MyPackageName -p path/to/apex/classes
```

```
sfdx force:source:retrieve -n MyPackageName -m ApexClass:MyApexClass
```

```
sfdx force:source:retrieve -n MyPackageName -x path/to/package.xml
```

To retrieve source files to a given directory instead of the default package directory specified in sfdx-project.json:

```
sfdx force:source:retrieve -m "StandardValueSet:TaskStatus" -r path/to/unpackaged
```

Command Syntax

`sfdx force:source:retrieve`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[-a APIVERSION]
[-r RETRIEVETARGETDIR]
[-p SOURCEPATH]
[-w WAIT]
[-x MANIFEST]
[-m METADATA]
[-n PACKAGENAMES]
[-t]
[-f]
[--verbose]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`-a | --apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-r | --retrievetargetdir RETRIEVETARGETDIR`

Optional

The root of the directory structure into which the source files are retrieved.

If the target directory matches one of the package directories in your sfdx-project.json file, the command fails.

Running the command multiple times with the same target adds new files and overwrites existing files.

Type: directory

`-p | --sourcepath SOURCEPATH`

Optional

A comma-separated list of file paths for source to retrieve from the org. The supplied paths can be to a single file (in which case the operation is applied to only one file) or to a folder (in which case the operation is applied to all source files in the directory and its sub-directories).

If you specify this parameter, don't specify --manifest or --metadata.

Type: array

`-w | --wait WAIT`

Optional

Number of minutes to wait for the command to complete and display results to the terminal window. If the command continues to run after the wait period, the CLI returns control of the terminal window to you.

Type: minutes

Default value: 33 minutes

`-x | --manifest MANIFEST`

Optional

The complete path for the manifest (package.xml) file that specifies the components to retrieve.

If you specify this parameter, don't specify --metadata or --sourcepath.

Type: filepath

`-m | --metadata METADATA`

Optional

A comma-separated list of names of metadata components to retrieve from the org.

If you specify this parameter, don't specify --manifest or --sourcepath.

Type: array

`-n | --packagenames PACKAGENAMES`

Optional

A comma-separated list of packages to retrieve.

Type: array

-t | --tracksource

Optional

If the retrieve succeeds, update source tracking information; doesn't delete local files that were deleted in the org.

Type: boolean

-f | --forceoverwrite

Optional

Ignore conflict warnings and overwrite changes to the project.

Type: boolean

--verbose

Optional

Emit additional command output to stdout.

Type: boolean

force:source:status

List local changes and/or changes in a scratch org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style commands instead.

We found that this command (`force:source:status`), which shows either the local and remote changes, not very intuitive. So instead we now provide two separate commands to preview what a deploy or a retrieve will do:

- project deploy preview
- project retrieve preview

These `preview` commands have the same flags as their non-preview commands, such as `project deploy start`.

Use this table to map the flags between the old and new commands.

`force:source:status` **Flag**	**Equivalent** `project deploy preview` **or** `project retrieve preview` **Flag**	**Notes**
`-l`, `--local`	No equivalent.	We now provide two separate commands to preview what a deploy or a retrieve will do, which is more intuitive.
`-r`, `--remote`	No equivalent	We now provide two separate commands to preview what a deploy or a retrieve will do, which is more intuitive.
`-u`, `--targetusername`	`-o`, `--target-org`	Note the new short and long flag name.
`--apiversion`	`-a`, `--api-version`	
`--concise`	`No equivalent.`	

`force:source:status` **Flag**	**Equivalent** `project deploy preview` **or** `project retrieve preview` **Flag**	**Notes**
`--json`	`--json`	
`--loglevel`	`No equivalent. Use the SF_LOG_LEVEL environment variable instead.`	

Here's an example to help you update your old commands. This `sfdx`-style command:

```
sfdx force:source:status --targetusername --remote
```

Looks like this using the equivalent `sf`-style command:

```
sf project retrieve preview --target-org myscratch --source-dir path/to/source/objects
```

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Examples for `force:source:status`

```
sfdx force:source:status -l
```

```
sfdx force:source:status -r
```

```
sfdx force:source:status
```

```
sfdx force:source:status -u me@example.com --json
```

Command Syntax

`sfdx force:source:status`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`[-l]`

`[-r]`

`[--concise]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-l | --local`

Optional

Lists the changes that have been made locally.

Type: boolean

`-r | --remote`

Optional

Lists the changes that have been made in the scratch org.

Type: boolean

`--concise`

Optional

Emit brief command output to stdout.

Type: boolean

Aliases for `force:source:status`

```
force:source:beta:status
```

`force:source:tracking:clear`

Clear all local source tracking information.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project delete tracking command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:source:tracking:clear`

WARNING: This command deletes or overwrites all existing source tracking files. Use with extreme caution.

Clears all local source tracking information. When you next run force:source:status, the CLI displays all local and remote files as changed, and any files with the same name are listed as conflicts.

Command Syntax

```
sfdx force:source:tracking:clear
    [--json]
    [--loglevel LOGLEVEL]
    [-u TARGETUSERNAME]
    [--apiversion APIVERSION]
    [-p]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

-p | --noprompt

Optional

Do not prompt for source tracking override confirmation.

Type: boolean

Aliases for `force:source:tracking:clear`

```
force:source:beta:tracking:clear
```

`force:source:tracking:reset`

Reset local and remote source tracking.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style project reset tracking command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--noprompt`. New name: `--no-prompt`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on April 13, 2023.

Help for `force:source:tracking:reset`

WARNING: This command deletes or overwrites all existing source tracking files. Use with extreme caution.

Resets local and remote source tracking so that the CLI no longer registers differences between your local files and those in the org. When you next run force:source:status, the CLI returns no results, even though conflicts might actually exist. The CLI then resumes tracking new source changes as usual.

Use the --revision parameter to reset source tracking to a specific revision number of an org source member. To get the revision number, query the SourceMember Tooling API object with the force:data:soql:query command. For example:

```
 sfdx force:data:soql:query -q "SELECT MemberName, MemberType, RevisionCounter FROM
SourceMember" -t
```

Command Syntax

`sfdx force:source:tracking:reset`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

```
[-r REVISION]
[-p]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-r | --revision REVISION`
Optional

Reset to a specific SourceMember revision counter number.

Type: integer

`-p | --noprompt`
Optional

Do not prompt for source tracking override confirmation.

Type: boolean

Aliases for `force:source:tracking:reset`

```
force:source:beta:tracking:reset
```

`staticresource` Commands

force:staticresource:create
Creates a static resource in the specified directory or the current working directory. The resource folder and associated metadata file are created.

`force:staticresource:create`

Creates a static resource in the specified directory or the current working directory. The resource folder and associated metadata file are created.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style static-resource generate command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--resourcename`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:staticresource:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
sfdx force:staticresource:create -n MyResource
```

```
sfdx force:staticresource:create -n MyResource --contenttype application/json
```

```
sfdx force:staticresource:create -n MyResource -d staticresources
```

Command Syntax

`sfdx force:staticresource:create`

```
[--json]
[--loglevel LOGLEVEL]
-n RESOURCENAME
[--contenttype CONTENTTYPE]
[-d OUTPUTDIR]
[--apiversion APIVERSION]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-n | --resourcename RESOURCENAME`
Required

The name of the new static resource. This name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Type: string

`--contenttype CONTENTTYPE`
Optional

The content type of the generated static resource. This must be a valid MIME type such as application/json, application/javascript, application/zip, text/plain, text/css, etc.

Type: string

Default value: application/zip

`-d | --outputdir OUTPUTDIR`
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`user` Commands

Commands that perform user-related admin tasks.

force:user:create
Create a user for a scratch org.

force:user:display
Displays information about a user of a scratch org.

force:user:list
List all authenticated users of an org.

force:user:password:generate
Generate a password for scratch org users.

force:user:permset:assign
Assign a permission set to one or more users of an org.

force:user:permsetlicense:assign
Assign a permission set license to one or more users of an org.

`force:user:create`

Create a user for a scratch org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org create user command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Removed flag: `--targetdevhubusername`
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--setalias`. New name: `--set-alias`.
- Changed flag name: Old name `--definitionfile`. New name: `--definition-file`.
- Changed flag name: Old name `--setuniqueusername`. New name: `--set-unique-username`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:user:create`

Create a user for a scratch org, optionally setting an alias for use by the CLI, assigning permission sets (e.g., permsets=ps1,ps2), generating a password (e.g., generatepassword=true), and setting User sObject fields.

Examples for `force:user:create`

```
sfdx force:user:create
```

```
sfdx force:user:create -a testuser1 -f config/project-user-def.json profileName='Chatter
Free User'
```

```
sfdx force:user:create username=testuser1@my.org email=me@my.org permsets=DreamHouse
```

```
sfdx force:user:create -f config/project-user-def.json email=me@my.org generatepassword=true
```

Command Syntax

`sfdx force:user:create`

```
[--json]
[--loglevel LOGLEVEL]
```

```
[-v TARGETDEVHUBUSERNAME]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
[-a SETALIAS]
[-f DEFINITIONFILE]
[-s]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-a | --setalias SETALIAS`

Optional

Set an alias for the created username to reference within the CLI.

Type: string

`-f | --definitionfile DEFINITIONFILE`

Optional

File path to a user definition.

Type: string

-s | --setuniqueusername
Optional

Force the username, if specified in the definition file or at the command line, to be unique by appending the org ID.

Type: boolean

force:user:display

Displays information about a user of a scratch org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the sf-style commands on page 1 instead, which we update regularly. We keep this reference information about the sfdx-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent sf-style org display user command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Removed flag: `--targetdevhubusername`
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing sfdx commands to use the improvements we made in sf. We improved this particular command on February 2, 2023.

Help for force:user:display

Output includes the profile name, org ID, access token, instance URL, login URL, and alias if applicable.

Examples for force:user:display

```
sfdx force:user:display
```

```
sfdx force:user:display -u me@my.org --json
```

Command Syntax

sfdx force:user:display

```
[--json]
[--loglevel LOGLEVEL]
[-v TARGETDEVHUBUSERNAME]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
```

Parameters

--json
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`force:user:list`

List all authenticated users of an org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org list users command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Removed flag: `--targetdevhubusername`
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:user:list`

The original scratch org admin is marked with "(A)"

Examples for **`force:user:list`**

```
sfdx force:user:list
```

```
sfdx force:user:list -u me@my.org --json
```

```
sfdx force:user:list --json > tmp/MyUserList.json
```

Command Syntax

`sfdx force:user:list`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-v TARGETDEVHUBUSERNAME]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`

Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`force:user:password:generate`

Generate a password for scratch org users.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org generate password command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Removed flag: `--targetdevhubusername`
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--onbehalfof`. New name: `--on-behalf-of`, with new short name `-b`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for `force:user:password:generate`

Generates and sets a random password for one or more scratch org users. Targets the usernames listed with the --onbehalfof parameter or the --targetusername parameter. Defaults to the defaultusername.

If you haven't set a default Dev Hub, or if your scratch org isn't associated with your default Dev Hub, --targetdevhubusername is required.

To change the password strength, set the --complexity parameter to a value between 0 and 5. Each value specifies the types of characters used in the generated password:

0 - lower case letters only

1 - lower case letters and numbers only

2 - lower case letters and symbols only

3 - lower and upper case letters and numbers only

4 - lower and upper case letters and symbols only

5 - lower and upper case letters and numbers and symbols only

To see a password that was previously generated, run "sfdx force:user:display".

Examples for `force:user:password:generate`

```
sfdx force:user:password:generate
```

```
sfdx force:user:password:generate -l 12
```

```
sfdx force:user:password:generate -c 3
```

```
sfdx force:user:password:generate -u me@my.org --json
```

```
sfdx force:user:password:generate -o "user1@my.org,user2@my.org,user3@my.org"
```

Command Syntax

```
sfdx force:user:password:generate
   [--json]
   [--loglevel LOGLEVEL]
   [-v TARGETDEVHUBUSERNAME]
   [-u TARGETUSERNAME]
   [--apiversion APIVERSION]
   [-o ONBEHALFOF]
   [-l LENGTH]
   [-c COMPLEXITY]
```

Parameters

`--json`
Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`
Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v | --targetdevhubusername TARGETDEVHUBUSERNAME`
Optional

A username or alias for the target Dev Hub org. Overrides the default Dev Hub org.

Type: string

`-u | --targetusername TARGETUSERNAME`
Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-o | --onbehalfof ONBEHALFOF`
Optional

Comma-separated list of usernames or aliases to assign the password to.

Type: array

-l | --length LENGTH

Optional

Number of characters in the generated password; valid values are between 8 and 1000.

Type: integer

Default value: 13

-c | --complexity COMPLEXITY

Optional

Level of password complexity or strength; the higher the value, the stronger the password.

Type: integer

Default value: 5

force:user:permset:assign

Assign a permission set to one or more users of an org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org assign permset command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Removed flag: `--targetdevhubusername`
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--onbehalfof`. New name: `--on-behalf-of`, with new short name `-b`.
- Changed flag name: Old name `--permsetname`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Help for force:user:permset:assign

To specify an alias for the -u or -o parameter, use the username alias you set with the "alias:set" CLI command, not the User.Alias value of the org user.

Examples for force:user:permset:assign

```
sfdx force:user:permset:assign -n "DreamHouse, LargeDreamHouse"
```

```
sfdx force:user:permset:assign -n DreamHouse -u me@my.org
```

```
sfdx force:user:permset:assign -n DreamHouse -o "user1@my.org,user2,user3"
```

Command Syntax

`sfdx force:user:permset:assign`

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-n PERMSETNAME
[-o ONBEHALFOF]
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-n | --permsetname PERMSETNAME`

Required

Comma-separated list of permission sets to assign.

Type: array

`-o | --onbehalfof ONBEHALFOF`

Optional

Comma-separated list of usernames or aliases to assign the permission set to.

Type: array

`force:user:permsetlicense:assign`

Assign a permission set license to one or more users of an org.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style org assign permsetlicense command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Removed flag: `--targetdevhubusername`
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--targetusername`. New name: `--target-org`, with new short name `-o`.
- Changed flag name: Old name `--onbehalfof`. New name: `--on-behalf-of`, with new short name `-b`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 2, 2023.

Examples for `force:user:permsetlicense:assign`

```
sfdx force:user:permsetlicense:assign -n DreamHouse
```

```
sfdx force:user:permsetlicense:assign -n DreamHouse -u me@my.org
```

```
sfdx force:user:permsetlicense:assign -n DreamHouse -o "user1@my.org,user2,user3"
```

Command Syntax

`sfdx force:user:permsetlicense:assign`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`-n NAME`

`[-o ONBEHALFOF]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-u | --targetusername TARGETUSERNAME`

Optional

A username or alias for the target org. Overrides the default target org.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-n | --name NAME`

Required

The name of the permission set license to assign.

Type: string

`-o | --onbehalfof ONBEHALFOF`

Optional

Comma-separated list of usernames or aliases to assign the permission set license to.

Type: array

`visualforce` Commands

Use the visualforce commands to create Visualforce pages and components.

force:visualforce:component:create

Creates a Visualforce component in the specified directory or the current working directory. The command creates the .component file and associated metadata file.

force:visualforce:page:create

Creates a Visualforce page in the specified directory or the current working directory. The command creates the .page file and associated metadata file.

`force:visualforce:component:create`

Creates a Visualforce component in the specified directory or the current working directory. The command creates the .component file and associated metadata file.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style visualforce generate component command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.

- Changed flag name: Old name `--componentname`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for **`force:visualforce:component:create`**

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
sfdx force:visualforce:component:create -n mycomponent -l mylabel
```

```
sfdx force:visualforce:component:create -n mycomponent -l mylabel -d components
```

Command Syntax

`sfdx force:visualforce:component:create`

```
[--json]
[--loglevel LOGLEVEL]
[-t TEMPLATE]
[-d OUTPUTDIR]
-n COMPONENTNAME
[--apiversion APIVERSION]
-l LABEL
```

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-t | --template TEMPLATE`

Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultVFComponent

Default value: DefaultVFComponent

`-d | --outputdir OUTPUTDIR`
Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`-n | --componentname COMPONENTNAME`
Required

The Visualforce component name. The name can be up to 40 characters and must start with a letter.

Type: string

`--apiversion APIVERSION`
Optional

Override the API version used for API requests made by this command.

Type: string

`-l | --label LABEL`
Required

The label saved in the metadata for the Visualforce component.

Type: string

`force:visualforce:page:create`

Creates a Visualforce page in the specified directory or the current working directory. The command creates the .page file and associated metadata file.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style visualforce generate page command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.
- Changed flag name: Old name `--apiversion`. New name: `--api-version`.
- Changed flag name: Old name `--outputdir`. New name: `--output-dir`.
- Changed flag name: Old name `--pagename`. New name: `--name`.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on February 16, 2023.

Help for `force:visualforce:page:create`

If not supplied, the apiversion, template, and outputdir use default values.

The outputdir can be an absolute path or relative to the current working directory.

Examples:

```
sfdx force:visualforce:page:create -n mypage -l mylabel
```

```
sfdx force:visualforce:page:create -n mypage -l mylabel -d pages
```

Command Syntax

`sfdx force:visualforce:page:create`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-t TEMPLATE]`

`[-d OUTPUTDIR]`

`-n PAGENAME`

`[--apiversion APIVERSION]`

`-l LABEL`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-t | --template TEMPLATE`

Optional

The template to use to create the file. Supplied parameter values or default values are filled into a copy of the template.

Type: string

Permissible values are: DefaultVFPage

Default value: DefaultVFPage

`-d | --outputdir OUTPUTDIR`

Optional

The directory to store the newly created files. The location can be an absolute path or relative to the current working directory. The default is the current directory.

Type: string

Default value: .

`-n | --pagename PAGENAME`

Required

The Visualforce page name. The name can be up to 40 characters and must start with a letter.

Type: string

`--apiversion APIVERSION`

Optional

Override the API version used for API requests made by this command.

Type: string

`-l | --label LABEL`

Required

The label saved in the metadata for the Visualforce page.

Type: string

`info` Namespace

Access cli info from the command line.

releasenotes Commands
Commands related to cli release notes.

`releasenotes` Commands

Commands related to cli release notes.

info:releasenotes:display
Display Salesforce CLI release notes on the command line.

`info:releasenotes:display`

Display Salesforce CLI release notes on the command line.

Warning: As of April 20, 2023, we no longer maintain this section of the command reference. Refer to the reference information about the `sf`-style commands on page 1 instead, which we update regularly. We keep this reference information about the `sfdx`-style commands as-is for historical reference only.

Don't worry, this command and its flags continue to work the same as before, and any scripts that use the command won't break. However, we recommend that you start using the equivalent `sf`-style info releasenotes display command instead. Here's how the flags changed between the old and new commands; if a flag isn't listed, the old and new names are the same:

- Removed flag: `--loglevel`. Use the SF_LOG_LEVEL environment variable instead.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx` commands to use the improvements we made in `sf`. We improved this particular command on January 12, 2023.

Examples for `info:releasenotes:display`

Display release notes for the currently installed CLI version:

```
sfdx info:releasenotes:display
```

Display release notes for CLI version 7.120.0:

```
sfdx info:releasenotes:display --version 7.120.0
```

Display release notes for the CLI version that corresponds to a tag (stable, stable-rc, latest, latest-rc, rc):

```
sfdx info:releasenotes:display --version latest
```

Command Syntax

`sfdx info:releasenotes:display`

`[--json]`

`[--loglevel LOGLEVEL]`

`[-v VERSION]`

Parameters

`--json`

Optional

Format output as JSON.

Type: boolean

`--loglevel LOGLEVEL`

Optional

The logging level for this command invocation. Logs are stored in $HOME/.sf/sf.log.

Type: enum

Permissible values are: trace, debug, info, warn, error, fatal, TRACE, DEBUG, INFO, WARN, ERROR, FATAL

Default value: warn

`-v` | `--version VERSION`

Optional

CLI version or tag for which to display release notes.

Type: string

Aliases for `info:releasenotes:display`

```
whatsnew
```

Help for `sfdx` Commands

The `-h` | `--help` parameter shows details about `sfdx` topics and their commands.

Warning: As of April 20, 2023, we no longer maintain this page.

For background information about this change, read this blog post, which describes how we've updated many of the existing `sfdx`-style commands to use the improvements we made in `sf`.

For namespaces, the `-h` | `--help` parameter lists all topics in the namespace. For example, to see names and descriptions of all topics in the `force` namespace, run `sfdx force -h`.

For topics, the `-h` | `--help` parameter lists the commands and their descriptions. For example, to see all commands in the `org` topic, run `sfdx force:org -h`.

For commands, adding the `-h` | `--help` parameter shows parameters and usage information. For example, to see help for the `org:create` command, run `sfdx force:org:create -h`.

Help for commands has four parts.

1. **Short Description of Command**

   At the top of the `--help` output (with no heading), a short description of the command is shown. For longer descriptions, see the *Salesforce CLI Command Reference*.

2. **Usage**

   The command signature on the Usage line uses the docopt format.

   - All available parameters are listed. Parameters that have short names are listed using their short names.
   - Parameters that take a value show the value's type (for example, `<string>`) in angle brackets immediately after the parameter's name.
   - Optional parameters are in square brackets (`[ … ]`).
   - Required parameters have no annotation.
   - For parameters that accept a limited set of values, the values are shown after the parameter name, separated by pipes (`--parametername value1|value2|value3`).
   - Mutually exclusive options are shown in parentheses, separated by a pipe (`( … | … )`).

   If the command takes varargs (name-value pairs that aren't parameters), the usage signature includes `name=value...`.

   Tip: To see all Salesforce CLI commands, run `sfdx commands`.

3. **Options**

   The Options section lists all the command's parameters, including their short name, long name, and purpose. For parameters that accept a value, the value name is written after an equals sign (=). The equals sign is optional when you run the command—for example, you could run `sfdx force:org:create` ***`-f=config/enterprise-scratch-def.json`*** `-a TestOrg1` or `sfdx force:org:create` ***`-f config/enterprise-scratch-def.json`*** `-a TestOrg1` with the same results.

   Parameters that accept a limited list of values include the values in parentheses, with the default value indicated by an asterisk (`*`).

   For more information about the parameters, see the *Salesforce CLI Command Reference*.

4. **Description**

   Usage notes and examples are below the list of parameters, in the Description section. This information is also available in the *Salesforce CLI Command Reference*.

CLI Deprecation Policy

Salesforce deprecates CLI commands and flags when, for example, the underlying API changes.

The Salesforce CLI deprecation policy is:

- Salesforce can deprecate a command or flag at any time.
- When you run the deprecated command, Salesforce provides a deprecation warning for a minimum of 4 months.
- Salesforce removes the deprecated command or flag 4 months, or more, after the deprecation warning first appears.
- If you use a command or flag that's been deprecated but not yet removed, you get a warning message in `stderr` when you specify human-readable output. If you specify JSON output, the warning is presented as a property. The message includes the plugin version of when the command or flag will be removed. The command help also includes deprecation information when appropriate.
- When possible, Salesforce provides a functional alternative to the deprecated command or flag.
- Salesforce announces new and upcoming deprecated commands and flags in the release notes.

Discover Salesforce Plugins

Check out these other plugins that work with specific Salesforce features.

ISV Technical Enablement Plugin

The ISVTE plugin is an on-demand Technical Evangelist. It scans your package metadata and code, and provides targeted feedback to help you improve and future-proof your app. The feedback includes a detailed metadata inventory, recommendations on features or technologies to consider using, enablement resources, and installation limitations. The feedback also includes best practices, partner alerts, guidance on improving your partner Trailblazer score, and more. While it's designed for ISV and OEM partners, anyone developing on the platform can use it.

When you install the plugin, you're asked to confirm that it's unsigned. Answer *`yes`*. This behavior is expected.

See GitHub for documentation and more information.

CRM Analytics Plugin

CRM Analytics is a cloud-based platform for connecting data from multiple sources, creating interactive views of that data, and sharing those views in apps.

Use the CRM Analytics CLI plugin to create scratch orgs with Analytics Studio, which you can use to develop and test source code. The plugin includes commands that call a subset of the Analytics REST API endpoints to manage CRM Analytics assets programmatically. Create and iteratively develop CRM Analytics templates. Update and delete apps, dashboards, lenses, and dataflows. Use history commands to restore previous versions of dashboards and dataflows. Manage the auto-install lifecycle for embedded templated apps.

See Develop with the Analytics Plugin for the Salesforce CLI for documentation and more information.

Salesforce Code Analyzer Plugin

The Salesforce Code Analyzer plugin is a unified tool for static analysis of source code, in multiple languages (including Apex), with a consistent command-line interface and report output. We currently support the PMD rule engine, ESLint, and RetireJS.

The plugin creates "rule violations" when the scanner identifies issues. Developers use this information as feedback to fix their code. Integrate this plugin into your continuous integration (CI) solution to continually enforce the rules and ensure high-quality code.

See GitHub for documentation and more information.