
Salesforce CLI Command Reference

Salesforce, Summer '22



CONTENTS

- Salesforce CLI Command Reference 1
- sfdx 1
- sf 222
- CLI Deprecation Policy 283
- Discover Salesforce Plug-Ins 284

SALESFORCE CLI COMMAND REFERENCE

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

[sfdx](#)

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

[sf](#)

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

[CLI Deprecation Policy](#)

Salesforce deprecates CLI commands and parameters when, for example, the underlying API changes.

[Discover Salesforce Plug-Ins](#)

Check out these other plug-ins that work with specific Salesforce features.

sfdx

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

This version of the `sfdx` command reference includes details about version 7.155.0 of the `sfdx` executable of Salesforce CLI and the following plug-in versions:

- `salesforce-alm` version 54.5.0
- `@salesforce/plugin-alias` version 2.0.1
- `@salesforce/plugin-apex` version 0.13.0
- `@salesforce/plugin-auth` version 2.1.0
- `@salesforce/plugin-config` version 1.4.12
- `@salesforce/plugin-custom-metadata` version 2.0.0
- `@salesforce/plugin-data` version 2.0.3
- `@salesforce/plugin-limits` version 2.0.1
- `@salesforce/plugin-org` version 1.13.2
- `@salesforce/plugin-schema` version 2.1.1
- `@salesforce/plugin-templates` version 54.8.0
- `@salesforce/plugin-user` version 2.0.2
- `@salesforce/sfdx-plugin-lwc-test` version 0.1.7
- `@salesforce/plugin-source` version 1.10.2
- `@salesforce/plugin-info` version 2.0.1
- `@salesforce/plugin-community` version 2.0.0
- `@salesforce/plugin-signups` version 1.1.2

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 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.

[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.

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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

[access token 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.

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/.sfdx/sfdx.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.

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/.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

-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
```

access token Commands

Authorize an org using an access token.

[auth:access token:store](#)

Authorize an org using an existing Salesforce access token.

auth:access token:store

Authorize an org using an existing Salesforce access token.

Help for **auth:access token: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>!<access token>`"

Examples for **auth:access token:store**

```
sfdx auth:access token:store --instanceurl https://mycompany.my.salesforce.com
```

```
$ export SFDX_ACCESS_TOKEN=00Dxx0000000000!xxxxx
```

```
sfdx auth:access token:store --instanceurl https://dev-hub.my.salesforce.com --noprompt
```

Command Syntax

sfdx auth:access token: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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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 `sfdx 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/.sfdx/sfdx.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.

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`.

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
```

Command Syntax

sfdx auth:web: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/.sfdx/sfdx.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: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.

[config:list](#)

Lists the config variables that the Salesforce CLI uses for various commands and tasks.

[config:set](#)

Sets the configuration variables that the Salesforce CLI uses for various commands and tasks.

[config:unset](#)

Unsets the local and global configuration variables for the Salesforce CLI.

config:get

Get config var values for given names.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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

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 your scratch orgs, to deploy and retrieve source to and from non-source-tracked orgs, to see synchronization changes between your project and scratch 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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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. .

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.

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.

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.

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.

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 asynchronously.

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.

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 SUBJECT]

[-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/.sfdx/sfdx.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 | --subject SUBJECT

Optional

The sObject to create an Apex trigger on.

Type: string

Default value: SUBJECT

-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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

Examples for force:cmdt:generate

Generate a custom metadata type from an sObject in the default target org:

```
sfdx force:cmdt:generate --devname MyCMDT --subjectname 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 --subjectname 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 --subjectname 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"
--subjectname 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 --subjectname 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 --subjectname 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 SUBJECTNAME
[-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/.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

-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 | --subjectname SUBJECTNAME

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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.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

-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.

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/.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

-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.

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/.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

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: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.

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 OBJECTTYPE
[-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/.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 | --csvfile CSVFILE

Required
The path to the CSV file containing the ids of the records to delete.
Type: filepath

-s | --objecttype OBJECTTYPE

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.

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/.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

-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.

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 SUBJECTTYPE

[-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/.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 | --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 | --objecttype OBJECTTYPE

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.

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=7d1170000008U36AAE
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 SUBJECTTYPE
-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/.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

-s | --objecttype OBJECTTYPE

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.

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 001D000000Kv3d1
```

```
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 SUBJECTTYPE`

`[-i SUBJECTID]`

`[-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/.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

-s | --subjecttype SUBJECTTYPE

Required

The type of the record you're deleting.

Type: string

-i | --subjectid SUBJECTID

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.

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 001D000000Kv3d1
```

```
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 SUBJECTTYPE

[-i SUBJECTID]

[-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/.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

-s | --subjecttype SUBJECTTYPE

Required

The type of the record you're retrieving.

Type: string

-i | --subjectid SUBJECTID

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.

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 001D000000Kv3d1 -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 001D000000Kv3d1 -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 001D000000Kv3d1 -v "Name=NewAcme" --perflog --json
```

Command Syntax**sfdx force:data:record:update**

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-s SUBJECTTYPE
[-i SUBJECTID]
[-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/.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

-s | --subjecttype SUBJECTTYPE

Required

The sObject type of the record you're updating.

Type: string

-i | --subjectid SUBJECTID

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:query

Execute a SOQL query.

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
```

Command Syntax

sfdx force:data:soql:query

```
[--json]
[--loglevel LOGLEVEL]
[-u TARGETUSERNAME]
[--apiversion APIVERSION]
-q QUERY
[-t]
[-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/.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

-q | --query QUERY

Required

SOQL query to execute.

Type: string

-t | --usetoolingapi

Optional

Execute query with Tooling API.

Type: boolean

-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.

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/.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

-q | --query QUERY

Required

Soql query, or filepath of file containing a soql query, to retrieve records.

Type: string

-p | --plan

Optional

Generate multiple subject 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.

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_subject_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 SUBJECTTREEFILES]

[-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/.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 | --subjecttreefiles SUBJECTTREEFILES

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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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>.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.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:limits:recordcounts:display

Display record counts for the specified standard and custom objects.

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 '--subjecttype' 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 SUBJECTTYPE]

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

-s | --subjecttype SUBJECTTYPE

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:legacy:convert \(Deprecated\)](#)

The command `force:mdapi:legacy:convert` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:convert` instead. Converts metadata retrieved via Metadata API into the source format used in Salesforce DX projects.

[force:mdapi:legacy:deploy \(Deprecated\)](#)

The command `force:mdapi:legacy:deploy` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:deploy` instead. Deploys file representations of components into an org by creating or updating the components they represent. You can deploy and retrieve up to 10,000 files or 400 MB (39 MB compressed) at one time. The default target username is the admin user for the default scratch org.

[force:mdapi:legacy:deploy:report \(Deprecated\)](#)

The command `force:mdapi:legacy:deploy:report` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:deploy:report` instead. Checks the current status of an asynchronous metadata deployment.

[force:mdapi:legacy:retrieve \(Deprecated\)](#)

The command `force:mdapi:legacy:retrieve` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:retrieve` instead. 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.

[force:mdapi:legacy:retrieve:report \(Deprecated\)](#)

The command `force:mdapi:legacy:retrieve:report` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:retrieve:report` instead. Check the status of an asynchronous metadata retrieval.

[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.

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/.sfdx/sfdx.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.

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/.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 | --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).

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

Default value: NoTestRun

-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 .

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/.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

-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.

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]

[--resultsdire 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/.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

-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.

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/.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

-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:legacy:convert (Deprecated)

The command `force:mdapi:legacy:convert` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:convert` instead. Converts metadata retrieved via Metadata API into the source format used in Salesforce DX projects.

Help for force:mdapi:legacy:convert

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:legacy:convert".

To convert files from the source format back to the metadata format, so that you can deploy them using "sfdx force:mdapi:legacy:deploy", run "sfdx force:source:convert".

Examples:

```
sfdx force:mdapi:legacy:convert -r path/to/metadata
```

```
sfdx force:mdapi:legacy:convert -r path/to/metadata -d path/to/outputdir
```

Command Syntax

sfdx force:mdapi:legacy:convert

[--json]

[--loglevel LOGLEVEL]

-r ROOTDIR

[-d OUTPUTDIR]

[-x MANIFEST]

[-m METADATA]

[-p METADATAPATH]

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

-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

-m | --metadata METADATA

Optional

A comma-separated list of metadata component names to convert.

Type: array

-p | --metadatapath METADATAPATH

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. If the comma-separated list you're supplying contains spaces, enclose the entire comma-separated list in one set of double quotes.

Type: array

force:mdapi:legacy:deploy (Deprecated)

The command `force:mdapi:legacy:deploy` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:deploy` instead. Deploys file representations of components into an org by creating or updating the components they represent. You can deploy and retrieve up to 10,000 files or 400 MB (39 MB compressed) at one time. The default target username is the admin user for the default scratch org.

Help for force:mdapi:legacy: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.

Specify the location of the files to deploy as a .zip file or by the root of the directory tree containing the files. To check the status of a deployment, specify its job ID. To run quick deploy of a recently validated package, use `--validateddeployrequestid` with the validated ID.

To wait for the command to finish running no matter how long the deployment takes, set `--wait` to `-1`: `"sfdx force mdapi:deploy -w -1 ..."`.

If you don't specify `--wait`, the command returns immediately and the metadata deploys asynchronously. Run `force:mdapi:legacy:deploy:report` to check the status of the deploy.

Command Syntax

sfdx force:mdapi:legacy: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]`

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 | --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

-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 -1 (no limit). 0

Type: 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 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:legacy: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

force:mdapi:legacy:deploy:report (Deprecated)

The command `force:mdapi:legacy:deploy:report` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:deploy:report` instead. Checks the current status of an asynchronous metadata deployment.

Help for force:mdapi:legacy: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.

Command Syntax

```
sfdx force:mdapi:legacy:deploy:report  
  [--json]  
  [--loglevel LOGLEVEL]  
  [-u TARGETUSERNAME]  
  [--apiversion APIVERSION]  
  [-w WAIT]  
  [-i JOBID]  
  [--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/.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

-w | --wait WAIT

Optional

The number of minutes to wait for the command to complete. The default is -1 (no limit). 0

Type: 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

force:mdapi:legacy:retrieve (Deprecated)

The command `force:mdapi:legacy:retrieve` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:retrieve` instead. 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.

Help for force:mdapi:legacy:retrieve

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.

Command Syntax**sfdx force:mdapi:legacy:retrieve**

[--json]

[--loglevel LOGLEVEL]

[-u TARGETUSERNAME]

[-a APIVERSION]

[-w WAIT]

-r RETRIEVETARGETDIR

[-k UNPACKAGED]

[--verbose]

[-d SOURCEDIR]

[-p PACKAGENAMES]

[-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

-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

-w | --wait WAIT

Optional

The number of minutes to wait for the command to complete.

Type: minutes

-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

--verbose

Optional

Indicates that you want verbose output from the retrieve operation.

Type: boolean

-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

force:mdapi:legacy:retrieve:report (Deprecated)

The command `force:mdapi:legacy:retrieve:report` has been deprecated and will be removed in v55.0 or later. Use `force:mdapi:retrieve:report` instead. Check the status of an asynchronous metadata retrieval.

Help for force:mdapi:legacy: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 deploy status. If the retrieve was successful, the resulting zip file will be saved to the location passed in with the retrieve target parameter.

Command Syntax

```
sfdx force:mdapi:legacy:retrieve:report  
  [--json]  
  [--loglevel LOGLEVEL]  
  [-u TARGETUSERNAME]  
  [--apiversion APIVERSION]  
  [-w WAIT]  
  [-r RETRIEVETARGETDIR]  
  [--verbose]  
  [-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/.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

-w | --wait WAIT

Optional

The number of minutes to wait for the command to complete. -1

Type: minutes

-r | --retrieve targetdir RETRIEVETARGETDIR

Optional

The root of the directory structure where the retrieved .zip or metadata files are put.

Type: directory

--verbose

Optional

Indicates that you want verbose output from the retrieve operation.

Type: boolean

-i | --jobid JOBID

Optional

The job ID (asynclid) 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 --retrieve targetdir. Use with --wait to resume waiting.

Type: id

force:mdapi:listmetadata

Display properties of metadata components of a specified type.

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/.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

-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.

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/.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

-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.

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 directory:

```
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 RetrievalID, naming the zip file and extracting all contents:

```
sfdx force:mdapi:retrieve:report -i retrievalId -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/.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

-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:beta:create`

Create a scratch or sandbox org .

`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:beta:create

Create a scratch or sandbox org .

Help for **force:org:beta: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:beta: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:beta: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/.sfdx/sfdx.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

force:org:clone

Clone a sandbox org.

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.

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=DevSbx1 SourceSandboxName=Sbx2Clone -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/.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

-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.

Help for force:org:create

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:

```
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/.sfdx/sfdx.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

force:org:delete

Mark a scratch or sandbox org for deletion .

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/.sfdx/sfdx.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.

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/.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

--verbose
Optional
Emit additional command output to stdout.
Type: boolean

force:org:list

List all orgs you've created or authenticated to.

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/.sfdx/sfdx.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.

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/.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

-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.

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/.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:org:shape:delete

Delete all org shapes for a target org.

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/.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

-p | --noprompt

Optional

Do not prompt for confirmation.

Type: boolean

force:org:shape:list

List all org shapes you've created.

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/.sfdx/sfdx.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.

 **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 00Dxx00000000000 --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/.sfdx/sfdx.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.



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 00o...
```

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/.sfdx/sfdx.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 00o) of the snapshot to delete.

Type: string

force:org:snapshot:get (Pilot)

Get details about a scratch org snapshot.

 **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 00o...
```

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/.sfdx/sfdx.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 00o) of the snapshot to retrieve.

Type: string

force:org:snapshot:list (Pilot)

List scratch org snapshots.

 **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/.sfdx/sfdx.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.

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/.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
- 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](#)

Creates a package.

[force:package:delete](#)

Delete unlocked and second-generation managed packages. Before you delete a package, first delete all associated package versions.

[force:package:install](#)

Installs a package in the target org.

[force:package:install:report](#)

Retrieves the status of a package installation request.

[force:package:installed:list](#)

Lists all packages installed in the target org.

[force:package:list](#)

Lists all packages in the Dev Hub org.

[force:package:uninstall](#)

Uninstalls a second-generation package from the target org. To uninstall a first-generation package, use the Salesforce user interface.

[force:package:uninstall:report](#)

Retrieves the status of a package uninstall request.

[force:package:update](#)

Updates details about a package. Does not create a package version.

[force:package:version:create](#)

Creates a package version in the Dev Hub org.

[force:package:version:create:list](#)

Lists all requests to create second-generation package versions in the Dev Hub org.

[force:package:version:create:report](#)

Retrieves details about a package version creation request in the Dev Hub org.

[force:package:version:delete](#)

Delete unlocked and second-generation managed package versions.

[force:package:version:displayancestry](#)

Displays the ancestry tree for a 2GP managed package version.

[force:package:version:list](#)

Lists all package versions in the Dev Hub org.

[force:package:version:promote](#)

Promotes a package version to released status.

[force:package:version:report](#)

Retrieves details about a package version in the Dev Hub org.

[force:package:version:update](#)

Updates a second-generation package version in the Dev Hub org.

force:package:create

Creates a package.

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.

Examples:

```
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
```

Run 'sfdx force:package:list' to list all packages in the Dev Hub org.

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/.sfdx/sfdx.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

force:package:delete

Delete unlocked and second-generation managed packages. Before you delete a package, first delete all associated package versions.

Help for force:package:delete

Specify the ID or alias of the package you want to delete.

Examples:

```
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/.sfdx/sfdx.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

force:package:install

Installs a package in the target org.

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:

```
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/.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

-w | --wait WAIT

Optional

Maximum number of minutes to wait for installation status. The default is 0.

Type: 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

-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

Optional

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.

Deprecation notice: The `--securitytype` parameter's default value will change from `AllUsers` to `AdminsOnly` in an upcoming release (v47.0 or later).

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`

force:package:install:report

Retrieves the status of a package installation request.

Help for **force:package:install:report**

Examples:

```
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/.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 | --requestid REQUESTID

Required

The ID of the package install request you want to check.

Type: id

force:package:installed:list

Lists all packages installed in the target org.

Help for **force:package:installed:list**

Examples:

```
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/.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:package:list

Lists all packages in the Dev Hub org.

Help for force:package:list

You can view the namespace, IDs, and other details for each package.

Examples:

```
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/.sfdx/sfdx.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

force:package:uninstall

Uninstalls a second-generation package from the target org. To uninstall a first-generation package, use the Salesforce user interface.

Help for **force:package:uninstall**

Specify the package ID for a second-generation package.

Examples:

```
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"
```

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.

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/.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

-w | --wait WAIT

Optional

Maximum number of minutes to wait for uninstall status. The default is 0.

Type: minutes

-p | --package PACKAGE

Optional

The ID (starts with 04t) or alias of the package version to uninstall.

Type: string

force:package:uninstall:report

Retrieves the status of a package uninstall request.

Help for **force:package:uninstall:report**

Examples:

```
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/.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 | --requestid REQUESTID

Required

The ID of the package uninstall request you want to check.

Type: id

force:package:update

Updates details about a package. Does not create a package version.

Help for **force:package:update**

Specify a new value for each option you want to update.

Examples:

```
sfdx force:package:update -p "Your Package Alias" -n "New Package Name"
```

```
sfdx force:package:update -p 0Ho... -d "New Package Description"
```

Run "sfdx force:package:list" to list all packages in the Dev Hub org.

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/.sfdx/sfdx.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

force:package:version:create

Creates a package version in the Dev Hub org.

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:

```
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]
[-p PACKAGE]
[-d PATH]
[-f DEFINITIONFILE]
[-b BRANCH]
[-t TAG]
[-k INSTALLATIONKEY]
[-x]
[-w WAIT]
[-a VERSIONNAME]
[-n VERSIONNUMBER]
[-e VERSIONDESCRIPTION]
[-c]
[--releasenotesurl RELEASENOTESURL]
[--postinstallurl POSTINSTALLURL]
[--postinstallscript POSTINSTALLSCRIPT]
[--uninstallscript UNINSTALLSCRIPT]
[--skipvalidation]
[--skipancestorcheck]

```

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

-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

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

-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

-b | --branch BRANCH

Optional

Name of the branch in your source control system that the package version is based on.

Type: string

-t | --tag TAG

Optional

The package version's tag.

Type: string

-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

-w | --wait WAIT

Optional

The number of minutes to wait for the package version to be created.

Type: minutes

Default value: 0 minutes

-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

-e | --versiondescription VERSIONDESCRIPTION

Optional

The description of the package version to be created. Overrides the sfdx-project.json value.

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

--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

--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

--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

--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

--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

--skipancestorcheck

Optional

Override ancestry requirements, which allows you to specify a package ancestor that isn't the highest released package version.

Type: boolean

force:package:version:create:list

Lists all requests to create second-generation package versions in the Dev Hub org.

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:

```
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/.sfdx/sfdx.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

force:package:version:create:report

Retrieves details about a package version creation request in the Dev Hub org.

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.

Examples:

```
sfdx force:package:version:create:report -i 08c...
```

```
sfdx force:package:version:create:report -i 08c... -v devhub@example.com
```

To show all requests in the org, run "sfdx force:package:version:create:list".

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/.sfdx/sfdx.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

force:package:version:delete

Delete unlocked and second-generation managed package versions.

Help for **force:package:version:delete**

Specify the ID or alias of the package version you want to delete.

Examples:

```
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/.sfdx/sfdx.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

force:package:version:displayancestry

Displays the ancestry tree for a 2GP managed package version.

Help for **force:package:version:displayancestry**

Examples:

```
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/.sfdx/sfdx.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

force:package:version:list

Lists all package versions in the Dev Hub org.

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:

```
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/.sfdx/sfdx.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

force:package:version:promote

Promotes a package version to released status.

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:

```
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/.sfdx/sfdx.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

force:package:version:report

Retrieves details about a package version in the Dev Hub org.

Help for **force:package:version:report**

Examples:

```
sfdx force:package:version:report -p 04t...
```

```
sfdx force:package:version:report -p "Your Package Alias"
```

To update package version values, run "sfdx force:package:version:update".

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/.sfdx/sfdx.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

force:package:version:update

Updates a second-generation package version in the Dev Hub org.

Help for force:package:version:update

Specify a new value for each option you want to update.

Examples:

```
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"
```

To display details about a package version, run "sfdx force:package:version:report".

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/.sfdx/sfdx.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

package1 Commands

Use the package1 commands to create and view first-generation package versions in your Dev Hub org.

[force:package1:version:create](#)

Creates a first-generation package version in the release org.

[force:package1:version:create:get](#)

Retrieves the status of a package version creation request.

[force:package1:version:display](#)

Displays detailed information about an individual first-generation package version.

[force:package1:version:list](#)

Lists the versions for the specified package or all first-generation packages in the org.

force:package1:version:create

Creates a first-generation package version in the release org.

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/.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 | --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: number

force:package1:version:create:get

Retrieves the status of a package version creation request.

Help for force:package1:version:create:get

Examples:

```
sfdx force:package:version:create:report -i 08c...
```

```
sfdx force:package:version:create:report -i 08c... -v 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/.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 | --requestid REQUESTID

Required

The ID of the PackageUploadRequest.

Type: id

force:package1:version:display

Displays detailed information about an individual first-generation package version.

Help for force:package1:version:display

Display detailed information about an individual package version, including metadata package ID, name, the release state, and build number.

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/.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 | --packageversionid PACKAGEVERSIONID

Required

ID (starts with 04t) of the metadata package version whose details you want to display.

Type: id

force:packagel:version:list

Lists the versions for the specified package or all first-generation packages in the org.

Help for force:packagel:version:list

If a metadata package ID is specified, lists all versions of the specified package. Otherwise, lists all package versions for the org. For each package version, the list includes the package version ID, metadata package ID, name, version number, and release state.

Command Syntax

sfdx force:packagel: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/.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 | --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

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.

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/.sfdx/sfdx.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:object:describe](#)

Displays the metadata for a standard or custom object.

[force:schema:object:list](#)

List all objects of a specified category.

force:schema:object:describe

Displays the metadata for a standard or custom object.

Examples for **force:schema:object:describe**

```
sfdx force:schema:object:describe -s Account
```

```
sfdx force:schema:object:describe -s MyObject__c
```

```
sfdx force:schema:object:describe -s ApexClass -t
```

Command Syntax

sfdx force:schema:object:describe

[--json]

[--loglevel LOGLEVEL]

[-u TARGETUSERNAME]

[--apiversion APIVERSION]

-s OBJECTTYPE

[-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/.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

-s | --subjecttype SUBJECTTYPE

Required

The API name of the object to describe.

Type: string

-t | --usetoolingapi

Optional

Execute with Tooling API.

Type: boolean

force:schema:subject:list

List all objects of a specified category.

Examples for force:schema:subject:list

```
sfdx force:schema:subject:list -c all
```

```
sfdx force:schema:subject:list -c custom
```

```
sfdx force:schema:subject:list -c standard
```

Command Syntax**sfdx force:schema:subject:list**

[--json]

[--loglevel LOGLEVEL]

[-u TARGETUSERNAME]

[--apiversion APIVERSION]

`[-c SUBJECTTYPECATEGORY]`

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 | --objecttypecategory SUBJECTTYPECATEGORY

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 your scratch orgs, to deploy and retrieve source to and from non-source-tracked orgs, to see synchronization changes between your project and scratch 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:legacy:pull \(Deprecated\)](#)

The command `force:source:legacy:pull` has been deprecated and will be removed in v55.0 or later. Use `force:source:pull` instead. Pulls changed source from the scratch org to your project to keep them in sync.

[force:source:legacy:push \(Deprecated\)](#)

The command `force:source:legacy:push` has been deprecated and will be removed in v55.0 or later. Use `force:source:push` instead. Pushes changed source from your project to a scratch org to keep them in sync.

[force:source:legacy:status \(Deprecated\)](#)

The command `force:source:legacy:status` has been deprecated and will be removed in v55.0 or later. Use `force:source:status` instead. Lists changes that have been made locally, in a scratch org, or both.

[force:source:legacy:tracking:clear \(Deprecated\)](#)

The command `force:source:legacy:tracking:clear` has been deprecated and will be removed in v55.0 or later. Use `force:source:tracking:clear` instead. Clear all local source tracking information.

[force:source:legacy:tracking:reset \(Deprecated\)](#)

The command `force:source:legacy:tracking:reset` has been deprecated and will be removed in v55.0 or later. Use `force:source:tracking:reset` instead. Reset local and remote source tracking.

[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 .

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/.sfdx/sfdx.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_1654811864449

-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.

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/.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 | --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.

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/.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 | --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

Default value: NoTestRun

-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.

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/.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

-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 .

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]

[--resultsdire 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/.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

-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 (asynclid) 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

--resultsdire RESULTS DIR

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.

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/.sfdx/sfdx.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 forceignored files.

Type: filepath

force:source:legacy:pull (Deprecated)

The command `force:source:legacy:pull` has been deprecated and will be removed in v55.0 or later. Use `force:source:pull` instead. Pulls changed source from the scratch org to your project to keep them in sync.

Help for force:source:legacy: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:legacy:pull**

[--json]

[--loglevel LOGLEVEL]

[-u TARGETUSERNAME]

[--apiversion APIVERSION]

`[-w WAIT]`

`[-f]`

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

-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

-f | --forceoverwrite

Optional

Runs the pull command even if conflicts exist. Changes in the scratch org overwrite changes in the project.

Type: boolean

force:source:legacy:push (Deprecated)

The command `force:source:legacy:push` has been deprecated and will be removed in v55.0 or later. Use `force:source:push` instead. Pushes changed source from your project to a scratch org to keep them in sync.

Help for **force:source:legacy: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:legacy:push

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[--apiversion APIVERSION]`

`[-f]`

`[-g]`

`[-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/.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 | --forceoverwrite

Optional

Runs the push command even if conflicts exist. Changes in the project overwrite changes in the scratch org.

Type: boolean

-g | --ignorewarnings

Optional

Completes the deployment even if warnings are generated.

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

force:source:legacy:status (Deprecated)

The command `force:source:legacy:status` has been deprecated and will be removed in v55.0 or later. Use `force:source:status` instead. Lists changes that have been made locally, in a scratch org, or both.

Help for **force:source:legacy:status**

Examples:

```
sfdx force:source:legacy:status -l
```

```
sfdx force:source:legacy:status -r
```

```
sfdx force:source:legacy:status -a
```

```
sfdx force:source:legacy:status -a -u me@example.com --json
```

Command Syntax

sfdx force:source:legacy:status

[--json]

[--loglevel LOGLEVEL]

[-u TARGETUSERNAME]

[--apiversion APIVERSION]

[-l]

[-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/.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

-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

force:source:legacy:tracking:clear (Deprecated)

The command `force:source:legacy:tracking:clear` has been deprecated and will be removed in v55.0 or later. Use `force:source:tracking:clear` instead. Clear all local source tracking information.

Help for **force:source:legacy: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:legacy: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:legacy: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/.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

-p | --noprompt

Optional

Do not prompt for source tracking override confirmation.

Type: boolean

force:source:legacy:tracking:reset (Deprecated)

The command `force:source:legacy:tracking:reset` has been deprecated and will be removed in v55.0 or later. Use `force:source:tracking:reset` instead. Reset local and remote source tracking.

Help for **force:source:legacy: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:legacy: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:legacy: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/.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

-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

force:source:manifest:create

Create a project manifest that lists the metadata components you want to deploy or retrieve .

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/.sfdx/sfdx.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.

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/.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 | --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.

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/.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 | --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.

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/.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 | --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 .

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
```

Command Syntax

sfdx force:source:retrieve

`[--json]`

`[--loglevel LOGLEVEL]`

`[-u TARGETUSERNAME]`

`[-a APIVERSION]`

`[-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/.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

-a | --apiversion APIVERSION

Optional

Override the API version used for API requests made by this command.

Type: string

-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.

Examples for force:source:status

```
sfdx force:source:status -l
```

```
sfdx force:source:status -r
```

```
sfdx force:source:status -a
```

```
sfdx force:source:status -a -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/.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

-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.

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/.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

-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.

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/.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

-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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.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

-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.

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/.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

-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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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/.sfdx/sfdx.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.

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*.

sf

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

This version of the `sf` command reference includes details about version 1.32.0 of the `sf` executable of Salesforce CLI and the following plug-in versions:

- `@salesforce/plugin-login` version 1.0.9
- `@salesforce/plugin-env` version 1.3.2
- `@salesforce/plugin-deploy-retrieve` version 1.4.3
- `@salesforce/plugin-config` version 2.3.2
- `@salesforce/plugin-functions` version 1.7.1
- `@salesforce/plugin-generate` version 1.0.14
- `@salesforce/plugin-info` version 2.0.1

For information about installing Salesforce CLI, see the [Salesforce CLI Setup Guide](#).

For information about Salesforce CLI changes, see the [Salesforce CLI Release Notes](#).

[config Commands](#)

Commands to configure Salesforce CLI.

[deploy Commands](#)

Commands to deploy artifacts to an environment.

[env Commands](#)

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

[generate Commands](#)

Commands to generate a project, create a function, and more.

[info Commands](#)

access cli info from the command line

[login Commands](#)

Commands to log in to an environment.

[logout Commands](#)

Commands to log out of an environment.

[retrieve Commands](#)

Commands to retrieve artifacts from an environment.

[run Commands](#)

Commands to run a function.

[whoami Commands](#)

Commands to show information about yourself or your account.

[Help for sf Commands](#)

The `-h` and `--help` flags show details about `sf` topics and their commands.

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

config list

List the configuration variables that you've previously set.

Description for **config list**

Global configuration variables apply to any directory 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

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 directory.

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
```

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 directory.

Type: boolean

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 directory.

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 directory.

Type: boolean

deploy Commands

Commands to deploy artifacts to an environment.

[deploy](#)

Deploy a project interactively to any Salesforce environment.

[deploy functions](#)

[deploy metadata \(Beta\)](#)

Deploy metadata in source format to an org from your local project.

[deploy metadata cancel \(Beta\)](#)

Cancel a deploy operation.

[deploy metadata quick \(Beta\)](#)

Quickly deploy a validated deployment to an org.

[deploy metadata report \(Beta\)](#)

Check the status of a deploy operation.

[deploy metadata resume \(Beta\)](#)

Resume watching a deploy operation.

[deploy metadata validate \(Beta\)](#)

Validate a metadata deployment without actually executing it.

deploy

Deploy a project interactively to any Salesforce environment.

Description for **deploy**

This command must be run from within a project.

The command first analyzes your project, your active or logged-into environments, and local defaults to determine what to deploy and where to deploy it. The command then prompts you for information about this particular deployment and provides intelligent choices based on its analysis.

For example, if your local project contains a source directory with metadata files in source format, the command asks if you want to deploy that Salesforce app to an org. The command lists your connected orgs and asks which one you want to deploy to. The list of orgs starts with scratch orgs, ordered by expiration date with the most recently created one first, and then Dev Hub and production orgs ordered by name. If the command finds Apex tests, it asks if you want to run them and at which level.

The command stores your responses in the "deploy-options.json" file in your local project directory and uses them as defaults when you rerun the command. Specify `--interactive` to force the command to reprompt.

Use this command for quick and simple deploys. For more complicated deployments, use the environment-specific commands, such as "sf deploy metadata", that provide additional flags.

Examples for **deploy**

Deploy a project and use stored values from a previous command run:

```
sf deploy
```

Reprompt for all deployment inputs:

```
sf deploy --interactive
```

Usage

```
sf deploy  
  [--interactive]
```

Flags

```
--interactive  
  Optional  
  Force the CLI to prompt for all deployment inputs.  
  Type: boolean
```

deploy functions

Usage

```
sf deploy functions  
  -o CONNECTED-ORG  
  [-b BRANCH]  
  [--force]  
  [-q]
```

Flags

-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

deploy metadata (Beta)

Deploy metadata in source format to an org from your local project.



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 **deploy metadata**

You must run this command from within a project.

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.

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 **deploy metadata**

Deploy the source files in a directory:

```
sf deploy metadata --source-dir path/to/source
```

Deploy a specific Apex class and the objects whose source is in a directory (both examples are equivalent):

```
sf deploy metadata --source-dir path/to/apex/classes/MyClass.cls path/to/source/objects
```

```
sf deploy metadata --source-dir path/to/apex/classes/MyClass.cls --source-dir path/to/source/objects
```


Deploy all Apex classes:

```
sf deploy metadata --metadata ApexClass
```

Deploy a specific Apex class:

```
sf deploy metadata --metadata ApexClass:MyApexClass
```

Deploy all custom objects and Apex classes (both examples are equivalent):

```
sf deploy metadata --metadata CustomObject ApexClass
```

```
sf deploy metadata --metadata CustomObject --metadata ApexClass
```

Deploy all Apex classes and a profile that has a space in its name:

```
sf deploy metadata --metadata ApexClass --metadata "Profile:My Profile"
```

Deploy all components listed in a manifest:

```
sf deploy metadata --manifest path/to/package.xml
```

Run the tests that aren't in any managed packages as part of a deployment:

```
sf deploy metadata --metadata ApexClass --test-level RunLocalTests
```

Usage

sf deploy metadata

[--json]

[-a API-VERSION]

[--async]

[--concise]

[--dry-run]

[-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]

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, which is the latest version supported the CLI, with the API version of your package.xml file.

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 deploy metadata resume". To check the status of the deployment, run "sf deploy metadata 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

-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.

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

Optional

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`.

Separate multiple test names with commas, and enclose the entire flag value in double quotes if a test contains a 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 `--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

Default value: NoTestRun

--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 deploy metadata resume". To check the status of the deployment, run "sf deploy metadata report".

Type: option

Default value: 33 minutes

deploy metadata cancel (Beta)

Cancel a 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 deploy metadata 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 deploy metadata cancel

Cancel a deploy operation using a job ID:

```
sf deploy metadata cancel --job-id 0Af0x000017yLUFCA2
```

Cancel the most recent deploy operation:

```
sf deploy metadata cancel --use-most-recent
```

Usage

```
sf deploy metadata 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 deploy metadata resume". To check the status of the cancellation, run "sf deploy metadata 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 deploy metadata
- sf deploy metadata validate
- sf deploy metadata quick
- sf deploy metadata 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 deploy metadata resume". To check the status of the cancellation, run "sf deploy metadata report".

Type: option

deploy metadata 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 deploy metadata quick

Before you run this command, first create a validated deployment with the "sf deploy metadata 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 deploy metadata quick

Run a quick deploy to your default org using a job ID:

```
sf deploy metadata quick --job-id 0Af0x000017yLUFCA2
```

Asynchronously run a quick deploy of the most recently validated deployment to an org with alias "my-prod-org":

```
sf deploy metadata quick --async --use-most-recent --target-org my-prod-org
```

Usage

sf deploy metadata quick

[--json]

[--async]

[--concise]

[-i JOB-ID]

[-o TARGET-ORG]

[-r]

[--verbose]

`[-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 deploy, run "sf deploy metadata resume". To check the status of the deploy, run "sf deploy metadata 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 deploy metadata resume". To check the status of the deploy, run "sf deploy metadata report".

Type: option

Default value: 33 minutes

deploy metadata report (Beta)

Check the status of a 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 deploy metadata 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 deploy metadata report

Check the status using a job ID:

```
sf deploy metadata report --job-id 0Af0x000017yLUFCA2
```

Check the status of the most recent deploy operation:

```
sf deploy metadata report --use-most-recent
```

Usage

sf deploy metadata report

[--json]

[-i JOB-ID]

[-r]

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 deploy metadata
- sf deploy metadata validate
- sf deploy metadata quick
- sf deploy metadata 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

deploy metadata resume (Beta)

Resume watching a 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 **deploy metadata 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 **deploy metadata resume**

Resume watching a deploy operation using a job ID:

```
sf deploy metadata resume --job-id 0Af0x000017yLUFCA2
```

Resume watching the most recent deploy operation:

```
sf deploy metadata resume --use-most-recent
```

Usage

sf deploy metadata resume

- `[--json]`
- `[--concise]`
- `[-i JOB-ID]`
- `[-r]`

`[--verbose]`

`[-w WAIT]`

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 deploy metadata

- sf deploy metadata validate

- sf deploy metadata quick

- sf deploy metadata 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 deploy metadata report".

Type: option

deploy metadata validate (Beta)

Validate a metadata deployment without actually executing it.



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 **deploy metadata 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 deploy metadata", 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 deploy metadata 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 **deploy metadata validate**

NOTE: These examples focus on validating large deployments. See the help for "sf deploy metadata" 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 deploy metadata 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 deploy metadata 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 deploy metadata validate --manifest path/to/package.xml
```

Usage

sf deploy metadata 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]`

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, which is the latest version supported the CLI, with the API version in your package.xml file.

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 deploy metadata resume". To check the status of the validation, run "sf deploy metadata 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

Optional

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`.

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 `--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: 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 deploy metadata resume". To check the status of the validation, run "sf deploy metadata report".

Type: option

Default value: 33 minutes

env Commands

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

[env create compute](#)

Create a compute environment for use with Salesforce Functions.

[env create sandbox \(Beta\)](#)

Create a sandbox org.

[env create scratch \(Beta\)](#)

Create a scratch org.

[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 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 resume scratch](#)

Resume the creation of an incomplete scratch org.

[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 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

[-o CONNECTED-ORG]

[-a ALIAS]

Flags

-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 create sandbox (Beta)

Create a sandbox 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 env 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 env 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 env 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 env create sandbox --name mysandbox --license-type Developer --alias MyDevSandbox --target-org prodOrg --set-default
```

Usage

sf env 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]`

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 definition 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 env 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 env 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

Default value: Developer

-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

--no-prompt


Optional

Don't prompt for confirmation about the sandbox configuration.

Type: boolean

env create scratch (Beta)

Create a scratch 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 **env create scratch**

There are two ways to create a scratch org: specify a definition file that contains the options or use the `--edition` flag to specify the one required option. If you want to set options other than the edition, 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 **env create scratch**

Create a Developer edition scratch org using your default Dev Hub and give the scratch org an alias:

```
sf env create scratch --edition=developer --alias my-scratch-org
```

Specify the Dev Hub using its alias and a scratch org definition file. Set the scratch org as your default and specify that it expires in 3 days:

```
sf env create scratch --target-dev-hub=MyHub --definition-file  
config/project-scratch-def.json --set-default --duration-days 3
```

Usage

sf env 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]
```

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 env 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

Optional

Username or alias of the Dev Hub org.

Overrides the value of the target-dev-hub configuration variable, if set.

Type: option

Default value: [object Object]

-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.

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 `env 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

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  
  [-e TARGET-COMPUTE]  
  [--confirm CONFIRM]
```

Flags

```
-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. For example, scratch org details include the access token, alias, username of the associated Dev Hub, the creation and expiration date, the generated scratch org username, and more. Compute environment details include the alias, connected orgs, creation date, project name, and more.

Examples for **env display**

Display details about a scratch org with alias my-scratch-org:

```
sf env display --target-env=my-scratch-org
```

Specify a username instead of an alias:

```
sf env display --target-env=test-123456-abcdefg@example.com
```

Specify JSON format and redirect output into a file:

```
sf env display --target-env=my-scratch-org --json > tmp/MyOrdDesc.json
```

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. For orgs, active means unexpired scratch orgs and orgs you're currently logged into.

Output is displayed in multiple tables, one for each environment type. For example, the Salesforce Orgs table lists the non-scratch orgs you're logged into, such as sandboxes, Dev Hubs, production orgs, and so on. Scratch orgs and compute environments get their own tables.

The two org tables show similar information, such as aliases, information about the org, and how you authorized (logged into) it, such as with a web browser or JWT. The scratch org table also shows the expiration date. For non-scratch orgs, the Username column refers to the user you logged into the org with. For scratch orgs it refers to the username that was generated for you when you created the scratch org. Your default scratch org or Dev Hub org is indicated with the "target-org" or "target-dev-hub" configuration variable, respectively, in the Config column.

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
```

Filter the output to list only orgs you authorized using a web browser; "Auth Method" is the name of a column:

```
sf env list --filter "Auth Method=web"
```

Display only the Aliases column and sort the aliases in descending order:

```
sf env list --sort "-Aliases" --columns "Aliases"
```

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 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

[-e TARGET-COMPUTE]

Flags

-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  
  [-e TARGET-COMPUTE]  
  [-l DRAIN-URL]
```

Flags

```
-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  
  [-e TARGET-COMPUTE]  
  
  [-j]
```

Flags

```
-e | --target-compute TARGET-COMPUTE  
  Optional  
  Environment name.  
  Type: option  
  
-j | --json  
  Optional  
  Output list in JSON format.  
  Type: boolean
```

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  
  [-e TARGET-COMPUTE]  
  
  [-l DRAIN-URL]
```

Flags

```
-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**

You can open the following types of environments in a web browser: scratch orgs, sandboxes, Dev Hubs, and production orgs. Run "sf env list" to view your environments and their aliases and login usernames.

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, such as /apex/YourPage to open a Visualforce page.

Examples for **env open**

Open the Visualforce page /apex/StartHere in a scratch org with alias test-org:

```
sf env open --target-env test-org --path /apex/StartHere
```

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

```
sf env open --target-env test-org --path /apex/StartHere --url-only
```

Open the environment in the Google Chrome browser:

```
sf env open --target-env test-org --path /apex/StartHere --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.

Specify the login user or alias that's associated with the environment. For scratch orgs, the login user is generated by the command that created the scratch org. You can also set an alias for the scratch org when you create it.

For Dev Hubs, sandboxes, and production orgs, specify the alias you set when you logged into the org with "sf login".

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 resume scratch

Resume the creation of an incomplete scratch org.

Description for env resume scratch

When the original "sf env 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 env resume scratch

Resume a scratch org create with a job ID:

```
sf env resume scratch --job-id 2SR3u0000008fBDGAY
```

Resume your most recent incomplete scratch org:

```
sf env resume scratch --use-most-recent
```

Usage

sf env 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

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

[-e TARGET-COMPUTE]

[-j]

Flags

-e | --target-compute TARGET-COMPUTE

Optional

Environment name.

Type: option

-j | --json

Optional

Output list in JSON format.

Type: boolean

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  
  [-e TARGET-COMPUTE]  
  [-j]
```

Flags

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

```
-j | --json  
  Optional  
  Output list in JSON format.  
  Type: boolean
```

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  
  [-e TARGET-COMPUTE]
```

Flags

```
-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  
  [-e TARGET-COMPUTE]
```

Flags

```
-e | --target-compute TARGET-COMPUTE  
  Optional  
  Environment name.  
  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 project](#)

Generate a Salesforce DX project.

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

[-n FUNCTION-NAME]

-l LANGUAGE

Flags

-n | --function-name FUNCTION-NAME

Optional

Function name. Must start with a capital letter.

Type: option

-l | --language LANGUAGE

Required

Language. Can be one of: javascript, typescript, java.

Type: option

Permissible values are: javascript, typescript, java

generate project

Generate a Salesforce DX project.

Description for generate project

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 basic scaffolding 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=53.0 --global
```

Examples for **generate project**

Generate a project called MyProject:

```
sf generate project --name MyProject
```

Generate the minimum number of files and directories:

```
sf generate project --name MyProject --template empty
```

Generate the project in /Users/jdoe/sf-projects rather than the current directory:

```
sf generate project --name MyProject --template empty --output-dir /Users/jdoe/sf-projects
```

Usage

sf generate project

[--json]

[-p DEFAULT-PACKAGE-DIR]

[-x]

-n NAME

[-s NAMESPACE]

[-d OUTPUT-DIR]

[-t TEMPLATE]

Flags

--json

Optional

Format output as json.

Type: boolean

-p | --default-package-dir DEFAULT-PACKAGE-DIR

Optional

Default package directory name.

The 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

-n | --name NAME

Required

Name of the generated project.

Creates a project directory with this name. Also sets the "name" property in the sfdx-project.json file to this name.

Type: option

-s | --namespace NAMESPACE

Optional

Project associated namespace.

The namespace associated with this project and any connected scratch orgs.

Type: option

-d | --output-dir OUTPUT-DIR

Optional

Directory to store the newly created project files.

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

Type: option

Default value: .

-t | --template TEMPLATE

Optional

Template to use to create the project.

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

info Commands

access cli info from the command line

[info releasenotes display](#)

info releasenotes display

Description for info releasenotes display

Display Salesforce CLI release notes on the command line.

Examples for info releasenotes display

Display release notes for the currently installed CLI version:

```
$ sf info releasenotes display
```

Display release notes for CLI version 7.120.0:

```
$ sf 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):

```
$ sf info releasenotes display --version latest
```

Usage

sf info releasenotes display

[--json]

[--loglevel LOGLEVEL]

[-v VERSION]

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

-v | --version VERSION

Optional

CLI version or tag for which to display release notes.

Type: string

Aliases for **info** **releasenotes** **display**

```
whatsnew
```

login Commands

Commands to log in to an environment.

[login](#)

Log interactively into an environment, such as a Salesforce org.

[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 org](#)

Log in to a Salesforce org using the web server flow.

[login org jwt](#)

Log in to a Salesforce org using a JSON web token (JWT).

login

Log interactively into an environment, such as a Salesforce org.

Description for **login**

Logging into an environment authorizes the CLI to run other commands that connect to that environment, such as deploying or retrieving metadata to and from an org.

The command first prompts you to choose an environment from a list of available ones. It then opens a browser to the appropriate login URL, such as <https://login.salesforce.com> for an org. Then, depending on the environment you choose, the command prompts for other actions, such as giving the environment an alias or setting it as your default.

This command is fully interactive and has no flags other than displaying the command-line help. Each environment has its own specific login command, such as "sf login org", which usually provide more flags than this interactive one. For more information about the interactive prompts from this command, see the help for the environment-specific command, such as "sf login org --help".

Examples for **login**

Log in interactively:

```
sf login
```

Usage

sf login

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

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

-u USERNAME

-f KEYFILE

-i CLIENTID

[-l INSTANCE-URL]

[--json]

`[-a ALIAS]`

`[-d]`

`[-v]`

Flags

-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

--json

Optional

Format output as JSON.

Type: boolean

-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

login org

Log in to a Salesforce org using the web server flow.

Description for login org

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 a project. 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 client id) that's generated for you. Then specify the consumer key with the `--clientid` flag.

Examples for login org

Run the command with no flags to open the default Salesforce login page (<https://login.salesforce.com>):

```
sf login org
```

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 login org --set-default-dev-hub --alias dev-hub
```

Log in to a sandbox and set it as your default org:

```
sf login org --instance-url https://MyDomainName--SandboxName.sandbox.my.salesforce.com  
--set-default
```

Use `--browser` to specify a specific browser, such as Google Chrome:

```
sf login org --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 login org --instance-url https://MyDomainName--SandboxName.sandbox.my.salesforce.com  
--set-default --browser chrome --clientid 04580y4051234051
```

Usage

sf login org

`[--json]`

`[-a ALIAS]`

`[-b BROWSER]`

`[-i CLIENTID]`

`[-l INSTANCE-URL]`

`[-d]`

`[-v]`

Flags

--json

Optional

Format output as json.

Type: boolean

-a | --alias ALIAS

Optional

Alias for the org.

Type: option

-b | --browser BROWSER

Optional

Browser in which to open the org.

You can log in to an org with one of the following browsers: Firefox, Safari, Google Chrome, or Windows Edge. 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

-i | --clientid CLIENTID

Optional

OAuth client id (also called consumer key) of your custom connected app.

Type: option

-l | --instance-url INSTANCE-URL

Optional

URL of the instance that the org lives on. (defaults to `https://login.salesforce.com`)

If you specify `--instance-url`, the 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

Default value: `https://login.salesforce.com`

-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

login org jwt

Log in to a Salesforce org using a JSON web token (JWT).

Description for login org 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 `--keyfile` 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 `--clientid` 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 login org jwt

Log into an org with username `jdoo@example.org` and on the default instance URL (<https://login.salesforce.org>). The private key is stored in the file `/Users/jdoo/JWT/server.key` and the command uses the connected app with consumer key (client id) `04580y4051234051`.

```
sf login org jwt --username jdoo@example.org --keyfile /Users/jdoo/JWT/server.key --clientid 04580y4051234051
```

Set the org as the default and give it an alias:

```
sf login org jwt --username jdoo@example.org --keyfile /Users/jdoo/JWT/server.key --clientid 04580y4051234051 --alias ci-org --set-default
```

Set the org as the default Dev Hub and give it an alias:

```
sf login org jwt --username jdoo@example.org --keyfile /Users/jdoo/JWT/server.key --clientid 04580y4051234051 --alias ci-dev-hub --set-default-dev-hub
```

Log in to a sandbox using URL `https://MyDomainName--SandboxName.sandbox.my.salesforce.com`:

```
sf login org jwt --username jdoo@example.org --keyfile /Users/jdoo/JWT/server.key --clientid 04580y4051234051 --alias ci-org --set-default --instance-url https://MyDomainName--SandboxName.sandbox.my.salesforce.com
```

Usage

```
sf login org jwt  
  [--json]
```

```

[-a ALIAS]
[-i CLIENTID]
[-l INSTANCE-URL]
[-f KEYFILE]
[-d]
[-v]
[-u USERNAME]

```

Flags

--json

Optional

Format output as json.

Type: boolean

-a | --alias ALIAS

Optional

Alias for the org.

Type: option

-i | --clientid CLIENTID

Optional

OAuth client id (also called consumer key) of your custom connected app.

Type: option

-l | --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

Default value: <https://login.salesforce.com>

-f | --keyfile KEYFILE

Optional

Path to a file containing the private key.

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

-u | --username USERNAME

Optional

Username of the user logging in.

Type: option

logout Commands

Commands to log out of an environment.

[logout](#)

Log out interactively from environments, such as Salesforce orgs and compute environments.

[logout functions](#)

Log out of your Salesforce Functions account.

[logout org](#)

Log out of a specified Salesforce org.

logout

Log out interactively from environments, such as Salesforce orgs and compute 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.

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 `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

logout org

Log out of a specified Salesforce org.

Description for logout org

By default, the command prompts you to confirm that you want to log out of the specified org. Use `--no-prompt` to not be prompted.

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 logout org

Log out of an org with alias "ci-org":

```
sf logout org --target-org ci-org
```

If your org doesn't have an alias, specify the username that you used when you logged into it:

```
sf logout org --target-org jdoe@example.org
```

Usage

sf logout org

[`--json`]

```
-o TARGET-ORG  
[--no-prompt]
```

Flags

--json

Optional

Format output as json.

Type: boolean

-o | --target-org TARGET-ORG

Required

Org alias or username to log out of.

Type: option

--no-prompt

Optional

Don't prompt for confirmation.

Type: boolean

retrieve Commands

Commands to retrieve artifacts from an environment.

[retrieve metadata \(Beta\)](#)

Retrieve metadata in source format from an org to your local project.

retrieve metadata (Beta)

Retrieve metadata in source format from an org to your local project.



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 **retrieve metadata**

You must run this command from within a project.

This command doesn't support source-tracking. The source you retrieve overwrites the corresponding source files in your local project. This command doesn't attempt to merge the source from your org with your local source files.

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 **retrieve metadata**

Retrieve the source files in a directory:

```
sf retrieve metadata --source-dir path/to/source
```

Retrieve a specific Apex class and the objects whose source is in a directory (both examples are equivalent):

```
sf retrieve metadata --source-dir path/to/apex/classes/MyClass.cls path/to/source/objects
```

```
sf retrieve metadata --source-dir path/to/apex/classes/MyClass.cls --source-dir  
path/to/source/objects
```

Retrieve all Apex classes:

```
sf retrieve metadata --metadata ApexClass
```

Retrieve a specific Apex class:

```
sf retrieve metadata --metadata ApexClass:MyApexClass
```

Retrieve all custom objects and Apex classes (both examples are equivalent):

```
sf retrieve metadata --metadata CustomObject ApexClass
```

```
sf retrieve metadata --metadata CustomObject --metadata ApexClass
```

Retrieve all metadata components listed in a manifest:

```
sf retrieve metadata --manifest path/to/package.xml
```

Retrieve metadata from a package:

```
sf retrieve metadata --package-name MyPackageName
```

Retrieve metadata from multiple packages, one of which has a space in its name (both examples are equivalent):

```
sf retrieve metadata --package-name Package1 "PackageName With Spaces" Package3
```

```
sf retrieve metadata --package-name Package1 --package-name "PackageName With Spaces"  
--package-name Package3
```

Usage

sf retrieve metadata

[--json]

[-a API-VERSION]

[-x MANIFEST]

[-m METADATA]

[-n PACKAGE-NAME]

[-d SOURCE-DIR]

[-o TARGET-ORG]

[-w WAIT]

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

-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.

Type: option

-n | --package-name PACKAGE-NAME

Optional

Package names to retrieve.

Type: option

-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

-o | --target-org TARGET-ORG

Optional

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

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](#)

Build and run a Salesforce Function in a 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

[-l FUNCTION-URL]

[-H HEADERS]

[-p PAYLOAD]

[-s]

[-o CONNECTED-ORG]

Flags

-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.

Previously, this command ran functions in a container. Container mode is still supported via the `container` subcommand. Arguments relevant to container mode are still accepted, but are deprecated, ignored, and will be dropped in a future release.

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: javascript, typescript, java, auto

Default value: auto

-v | --verbose

Optional

Output additional logs.

Type: boolean

run function start container

Build and run a Salesforce Function in a container.

Description for **run function start container**

Run this command from the directory of your Salesforce Functions project.

Examples for **run function start container**

Build and run a function:

```
sf run function start container
```

Run a function on a specific port with additional logs:

```
sf run function start container --port 5000 --verbose
```

Add environment variables and specify a network:

```
sf run function start container --env KEY=VALUE --network host
```

Usage

sf run function start container

[-p PORT]
[-b DEBUG-PORT]
[--clear-cache]
[--no-pull]
[-e ENV]
[--network NETWORK]
[-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

--clear-cache

Optional

Clear associated cache before executing.

Type: boolean

--no-pull

Optional

Skip pulling builder image before use.

Type: boolean

-e | --env ENV

Optional

Set environment variables (provided during build and run).

Type: option

--network NETWORK

Optional

Connect and build containers to a network. This can be useful to build containers which require a local resource.

Type: option

-v | --verbose

Optional

Output additional logs.

Type: boolean

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 debbuging the function.

Type: option

Default value: 9229

-l | --language LANGUAGE

Optional

The language that the function runs in.

Type: option

Permissible values are: javascript, typescript, java, auto

Default value: auto

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

[-j]

Flags

-j | --json

Optional

Output list in JSON format.

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.

CLI Deprecation Policy

Salesforce deprecates CLI commands and parameters when, for example, the underlying API changes.

The Salesforce CLI deprecation policy is:

- Salesforce can deprecate a command or parameter 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 parameter 4 months, or more, after the deprecation warning first appears.
- If you use a command or parameter 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 plug-in version of when the command or parameter will be removed. The command help also includes deprecation information when appropriate.

- When possible, Salesforce provides a functional alternative to the deprecated command or parameter.
- Salesforce announces new and upcoming deprecated commands and parameters in the release notes.

Discover Salesforce Plug-Ins

Check out these other plug-ins that work with specific Salesforce features.

ISV Technical Enablement Plug-In

The ISVTE Plug-in 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 plug-in, you're asked to confirm that it's unsigned. Answer `yes`. This behavior is expected.

See [GitHub](#) for documentation and more information.

CRM Analytics Plug-In

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 plug-in to create scratch orgs with Analytics Studio, which you can use to develop and test source code. The plug-in 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 Plug-In

The Salesforce Code Analyzer plug-in 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 plug-in creates "rule violations" when the scanner identifies issues. Developers use this information as feedback to fix their code. Integrate this plug-in into your CI/CD solution to continually enforce the rules and ensure high-quality code.

See [GitHub](#) for documentation and more information.