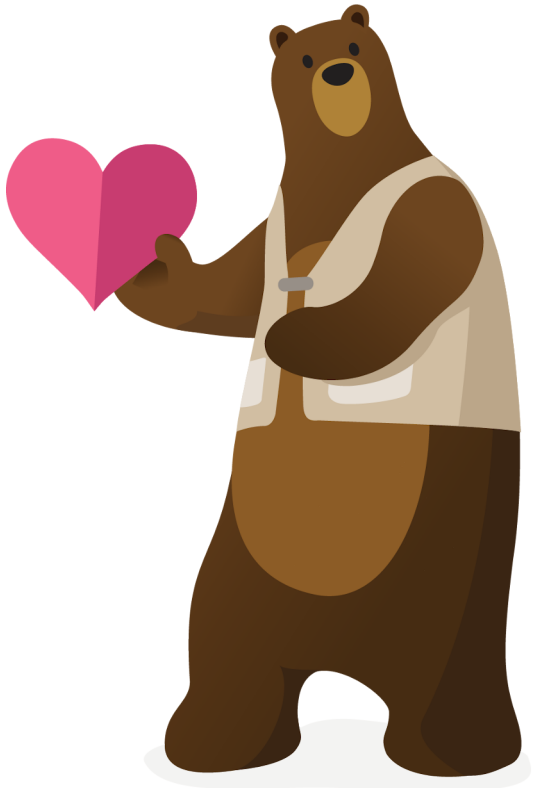
Salesforce CLI Setup Guide

Version 51.0, Spring '21



'21

@salesforcedocs

Last updated: November 24, 2020

CONTENTS

CHAPTER 1 Before You Begin

The Salesforce CLI is a command-line interface that simplifies development and build automation when working with your Salesforce org. Use it to create and manage orgs (scratch and sandboxes), synchronize source to and from orgs, create and install packages, and more.

Salesforce CLI consists of two parts:

- **CLI**: An npm package called `sfdx-cli`. The CLI is based on oclif, an open-source framework for building a command-line interface in Node.js. You run it on your local machine or continuous integration (CI) system. It supports the installation of custom plug-ins.
- **One or more plug-ins**. An npm package that you install into the CLI that contains commands. Most of the core functionality that Salesforce provides comes from plug-ins. These plug-ins are automatically installed when you install the CLI:
  - `salesforcedx`, the core Salesforce plug-in that contains commands that support source-driven development, such as creating and managing scratch orgs, synchronizing source code, creating second-generation packaging, and more.
  - Supporting plug-ins that contain commands that make it easier to use the CLI, such as setting configuration values or aliases.

  You can install more plug-ins, such as Einstein Analytics, to incorporate other Salesforce features into your development environment. You can also develop your own plug-in to add your custom functionality to Salesforce CLI.

System Requirements

Before you begin, review these system requirements to get the most out of Salesforce CLI and developer tools.

Operating Systems

Salesforce CLI supports the following operating systems.

- Windows—Windows 7 (64-bit and 32-bit) or later
- Mac—macOS 10.11 or later
- Linux—Ubuntu 14.0.4

Code Editor or IDE

You can use any code editor, including Salesforce Extensions for VS Code, a set of Visual Studio Code extensions that are designed for development on Salesforce Platform.

Note: If you're using Salesforce Extensions for VS Code, keep in mind that many of the installation commands are unavailable in the command palette. If you can't find a command in VS Code, run it in the integrated terminal.

Version Control System

You can use any version control system (VCS). We recommend that you use GitHub to take advantage of the samples in our GitHub repository.

CLI and Plug-In Versions

Salesforce supports only the most current versions of the Salesforce CLI and the `salesforcedx` plug-in. See the Salesforce CLI Release Notes for the latest version information.

To check the version of the Salesforce CLI installed on your computer, run `sfdx version`. To check the version of the installed `salesforcedx` plug-in, run `sfdx plugins --core`.

To upgrade to the current version, run `sfdx update`.

SEE ALSO:

Salesforce Extensions for Visual Studio Code

Salesforce DX Developer Guide

Salesforce CLI Command Reference

oclif: The Open CLI Framework

Update the CLI

CHAPTER 2 Install the Salesforce CLI and Plug-Ins

In this chapter ...

Use the CLI commands to create environments for development and testing, synchronize source code between your scratch orgs and version control system, and execute test suites.

Install the CLI on macOS

You install Salesforce CLI on macOS with a `.pkg` file.

1. Download the `.pkg` file.
2. Double-click the `.pkg` file.

SEE ALSO:

Verify Your Installation

Disable Automatic Update of the CLI and Plug-In

Install the CLI on Windows

You install the Salesforce CLI on Windows with an `.exe` file.

Download and run the Windows installer.

Warning: The Salesforce CLI works best within the native Windows command prompt (`cmd.exe`) and the Microsoft Powershell. We do not recommend using the Salesforce CLI with a Linux terminal emulator, such as Windows 10 Subsystem for Linux, cygwin, or MinGW, because support for bugs is limited.

SEE ALSO:

Verify Your Installation

Disable Automatic Update of the CLI and Plug-In

Install the CLI on Linux

The Linux version of Salesforce CLI is distributed as a tarball.

Find the download URL for your tarball from this manifest file. The `downloads` section lists unversioned URLs for the latest installers, which are especially useful for CI use cases. Select the URL based on your target platform.

1. Download or `wget` one of these tarballs.

```
wget https://developer.salesforce.com/media/salesforce-cli/sfdx-linux-amd64.tar.xz
```

2. Create an `sfdx` directory.

```
mkdir sfdx
```

3. Unpack the contents for your tarball version:

```
tar xJf sfdx-linux-amd64.tar.xz -C sfdx --strip-components 1
```

`-C` unpacks the contents in the `sfdx` directory, while `--strip-components 1` removes the root path component.

4. Run the install script.

```
./sfdx/install
```

The Salesforce CLI is installed in `/usr/local/bin/sfdx`. The installers are designed to provide appropriate permissions to the installation directories. If you receive permission or access errors that you can't address by using `sudo` or `chmod`, try installing the CLI using npm on page 5.

SEE ALSO:

Verify Your Installation

Disable Automatic Update of the CLI and Plug-In

Install the CLI with `npm`

If you've installed `Node.js` on your computer, you can use `npm` to install Salesforce CLI. This method lets you install Salesforce CLI from the command line and can be especially useful for continuous integration (CI) use cases.

This installation method is a good option if you don't have administrator permissions on your workstation, or if group policy blocks CLI installation and updates. Installing the CLI with `npm` doesn't require root permissions.

1. Ensure that the long-term support (Active LTS) version of Node.js is installed on your computer. To install the LTS version, go to https://nodejs.org/en/download/. To check your version number, run:

   ```
   node --version
   ```

2. Run this command.

   ```
   npm install sfdx-cli --global
   ```

   If you receive a permission error when installing the CLI using npm, we recommend not using sudo. See Fixing npm permissions.

SEE ALSO:

Verify Your Installation

npm Documentation

Verify Your Installation

Verify your Salesforce CLI installation and plug-in versions.

Run this command to verify the Salesforce CLI version:

```
$ sfdx --version
sfdx-cli/7.76.1-f301d5641f darwin-x64 node-v12.18.3
```

Run this command to verify the `salesforcedx` plug-in version:

```
$ sfdx plugins --core
@oclif/plugin-autocomplete 0.1.5 (core)
@oclif/plugin-commands 1.3.0 (core)
@oclif/plugin-help 3.2.0 (core)
@oclif/plugin-not-found 1.2.4 (core)
@oclif/plugin-plugins 1.9.1 (core)
@oclif/plugin-update 1.3.10 (core)
@oclif/plugin-warn-if-update-available 1.7.0 (core)
```

```
@oclif/plugin-which 1.0.3 (core)
@salesforce/sfdx-trust 3.4.3 (core)
alias 1.1.2 (core)
analytics 1.12.1 (core)
config 1.1.9 (core)
generator 1.1.3 (core)
salesforcedx 50.1.1 (core)                          //salesforcedx plug-in version
|- templates 50.1.0 (core)
|- @salesforce/sfdx-plugin-lwc-test 0.1.7 (core)
|- apex 0.1.1 (core)
|- custom-metadata 1.0.10 (core)
|- salesforce-alm 50.1.1 (core)
sfdx-cli 7.76.1 (core)                              //CLI version
```

This command returns a list of the other plug-ins installed in the CLI:

```
$ sfdx plugins
```

The core `salesforcedx` plug-in is not included in the preceding list unless you installed a newer version with the `sfdx plugins:install` command.

Run this command to return a list of the command families (topics) in the `force` namespace:

```
$ sfdx force --help
```

Run this command to view all available Salesforce CLI commands:

```
$ sfdx commands
```

SEE ALSO:

Install a Specific Version of the salesforcedx Plug-In

Install a Specific Version of the `salesforcedx` Plug-In

By default, the latest version of the `salesforcedx` plug-in is installed when you install Salesforce CLI for the first time. Sometimes, however, you want to use a specific version of the `salesforcedx` plug-in.

To determine which Salesforce CLI (`sfdx-cli`) and `salesforcedx` versions you have installed, run:

```
$ sfdx plugins --core

@oclif/plugin-autocomplete 0.1.5 (core)
@oclif/plugin-commands 1.3.0 (core)
@oclif/plugin-help 3.2.0 (core)
@oclif/plugin-not-found 1.2.4 (core)
@oclif/plugin-plugins 1.9.0 (core)
@oclif/plugin-update 1.3.10 (core)
@oclif/plugin-warn-if-update-available 1.7.0 (core)
@oclif/plugin-which 1.0.3 (core)
@salesforce/sfdx-trust 3.4.3 (core)
alias 1.1.2 (core)
analytics 1.12.1 (core)
config 1.1.8 (core)
generator 1.1.3 (core)
```

```
salesforcedx 49.13.1 (core)                          // salesforcedx plug-in version
|- custom-metadata 1.0.10 (core)
|- @salesforce/sfdx-plugin-lwc-test 0.1.7 (core)
|- templates 49.9.1 (core)
|- apex 0.1.1 (core)
|- salesforce-alm 49.12.2 (core)
sfdx-cli 7.75.1 (core)                               // CLI version
```

The output includes all installed plug-ins. If a plug-in has `(core)` next to its name, it's the version bundled with the CLI. If you install a specific version of the plug-in, its version number or tag is displayed instead.

Run this command to install a specific version of the plug-in, such as 49.14.0:

```
sfdx plugins:install salesforcedx@49.14.0
```

During sandbox preview, we provide a pre-release of the `salesforcedx` plug-in that contains updates that work only with the new features in the preview. Run this command to install the pre-release:

```
sfdx plugins:install salesforcedx@pre-release
```

To install the early release candidate of the weekly `salesforcedx` plug-in release, run this command:

```
sfdx plugins:install salesforcedx@latest-rc
```

Important: When you install a plug-in version using a tag, such as `@pre-release` or `latest-rc`, you stay with that tag until you uninstall the plug-in. To stop being tied to this release tag and go back to using the core `salesforcedx` plug-in that's bundled with the CLI, run this command:

```
sfdx plugins:uninstall salesforcedx
```

Discover Salesforce Plug-Ins

After you've installed Salesforce CLI and the core `salesforcedx` plug-in, 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.

Einstein Analytics Plug-In

Salesforce Einstein 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 Einstein 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 Salesforce Analytics REST API endpoints to manage Analytics assets programmatically. Create and iteratively develop 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.

Install Trusted Unsigned Plug-ins Automatically

When you install a plug-in with the `sfdx plugins:install` command, Salesforce CLI first verifies its digital signature. If the plug-in provides a valid signature, the CLI installs it. Otherwise, Salesforce CLI doesn't install it until you answer a warning prompt and acknowledge that you understand the risks. This process works well when you install a plug-in interactively at the command line, but can prevent a batch CI/CD job from completing. To automatically install a plug-in without prompting, even when unsigned, create an allowlist file on your local file system and add the plug-ins you trust.

Warning: After you install a plug-in and run one of its commands in a terminal, the command runs with your user privileges. As a result, the command can read encrypted data, communicate with any Salesforce org you authenticated to, or remove files in your home directory. Install only unsigned and unverified plug-ins that you trust.

1. Create a file called `unsignedPluginAllowList.json` and put it in one of these directories:
   - (Linux and macOS): `$HOME/.config/sfdx`
   - (Windows) Depending on your Windows configuration, either `C:\Users\username\.config\sfdx` or `%LOCALAPPDATA%\sfdx`
2. Add the names of the plug-ins you trust to the JSON file in a simple array of strings. For example:

```
[
    "sfdx-templates",
    "salesforce-cmdt",
    ...
]
```

CHAPTER 3 Salesforce CLI Configuration and Tips

In this chapter ...

Use Salesforce command-line interface (CLI) for most development and testing tasks. These tasks include authorizing a Dev Hub org, creating a scratch org, synchronizing source code between your scratch orgs and VCS, and running tests.

You can start using the CLI right after you install it.

The CLI commands are grouped into top-level topics. For example, the `force` top-level topic is divided into topics that group commands by functionality, such as the `force:org` commands to manage your orgs. The `config` top-level topic contains commands for managing configuration values.

Run `--help` at each level to get more information.

```
sfdx --help                    // lists all top-level topics
sfdx force --help              // lists all the topics under force
sfdx force:org --help          // lists all the commands in the topic
 force:org
sfdx force:org:open --help     // detailed info about the
force:org:open command
```

Run this command to view all available Salesforce CLI commands:

```
sfdx commands
```

To see all commands with their parameters and flags, run the command with the `--json` flag:

```
sfdx commands --json
```

Autocomplete CLI Commands, Parameters, and File Names

Partially type a Salesforce CLI command and then press Tab to autocomplete it, or press Tab twice to see all the available commands. The autocomplete feature also works on Salesforce CLI parameters and file names.

Note: Autocomplete is not currently available on computers running Windows.

Before you can use the autocomplete feature, install it using these steps.

1. Make sure you're on the latest version of the CLI. Run `sfdx plugins:install salesforcedx@latest`.
2. Run `sfdx plugins:install @oclif/plugin-autocomplete`.
3. Run `sfdx autocomplete`.
4. Follow the displayed instructions.

If autocomplete does not work immediately after installation, run `sfdx autocomplete --refresh-cache`. Then open a new terminal window.

Use the Salesforce CLI from Behind a Company Firewall or Web Proxy

If you install or update the Salesforce CLI on a computer that's behind a company firewall or web proxy, you sometimes receive error messages. In this case, you must further configure your system.

You get an error similar to the following when you run a command after installing the CLI binary behind a firewall or web proxy. This error is from a Linux computer, but Windows and macOS users sometimes see a similar error.

```
sfdx-cli: Updating CLI... !
   'ECONNRESET': tunneling socket could not be established, cause=connect EHOSTUNREACH
0.0.23.221:8080 - Local (10.126.148.39:53107)
```

To address this issue, run these commands from your terminal or Windows command prompt, replacing *username:pwd* with your web proxy username and password. If your proxy doesn't require these values, omit them. Also replace *proxy.company.com:8080* with the URL and port of your company proxy.

```
npm config set https-proxy https://username:pwd@proxy.company.com:8080
npm config set proxy https://username:pwd@proxy.company.com:8080
```

Then set the HTTP_PROXY or HTTPS_PROXY environment variable to the full URL of the proxy. For example, on UNIX:

```
export HTTP_PROXY=https://username:pwd@proxy.company.com:8080
```

```
export HTTPS_PROXY=https://username:pwd@proxy.company.com:8080
```

On a Windows machine:

```
set HTTP_PROXY=https://username:pwd@proxy.company.com:8080
```

```
set HTTPS_PROXY=https://username:pwd@proxy.company.com:8080
```

If You Still See an Error

Your Proxy Requires an Extra Certificate Authority

If you set the proxy environment variable, and you still see error messages, it's possible that your proxy requires an extra certificate authority (CA). Ask your IT department where to find or download the certificates.

Set this environment variable to point to the CA file: NODE_EXTRA_CA_CERTS.

Your Corporate Network Is Blocking Salesforce Hosts

It's possible that your corporate network is blocking the Salesforce hosts for updating or installing Salesforce CLI. Contact your IT department add these domains to your allowlist:

- https://developer.salesforce.com/media/salesforce-cli
- https://registry.npmjs.org

Windows Performance Suggestions

Follow these suggestions to improve the performance of Salesforce CLI on Windows.

Warning: We recommend that you consult your security administrator before making any of these suggested configuration changes.

Use a local file system for your Salesforce DX project rather than a cloud-based one.

Salesforce CLI performs better when your Salesforce DX project and associated files are on a local file system. Cloud-based file systems, such as OneDrive, Google Drive, and Dropbox, constantly watch all the files and directories in the file system. As a result, if you create your Salesforce DX project in one of these file systems, it can limit the performance of the Salesforce CLI. To avoid this issue, move your project directory away from these systems.

Install Salesforce CLI with the official installer and exclude the `sfdx` executable from Windows Defender.

- Windows Defender continually rescans executables for potential threats. This scanning can have a noticeable performance impact on slower machines.
- To exclude the CLI, use the `sfdx` executable installed from developer.salesforce.com and follow these steps:
  1. Add an exclusion to Windows Security.
  2. When prompted to select a folder, select *`C:\Program Files\Salesforce CLI`*.

Exclude the project folder from Windows Defender.

It's also possible that Windows Defender keeps rescanning your project folder, causing negative performance. To exclude your project folder, follow these steps.

Exclude the `sfdx` executable from other security software.

Some companies use more extensive security software than Windows Defender, and this security software can cause Salesforce CLI to perform slowly. Work with your internal IT department to exclude the `sfdx` executable from all security software.

Close memory intensive programs.

Salesforce CLI can be performing slowly because other programs such as Google Chrome or VS Code are using too much memory. Try restarting these programs to free up memory.

CLI Runtime Configuration Values

You can set CLI runtime configuration values for your current project or for all projects. You can set two kinds of configuration values: global and local. Global values apply to all projects on your computer. Local values apply to a specific project. Local values override global values when commands are run from within a Salesforce DX project directory.

To set a configuration value for the current project:

```
sfdx config:set name=<value>
```

For local configuration values, you must issue this command from within the Salesforce DX project directory.

To set the value for all your projects:

```
sfdx config:set name=<value> --global
```

You can issue global commands anywhere or within any project, yet they apply to all the Salesforce CLI commands you run.

You can view the local and global configuration values that you have set. The output lists the local values for the project directory from which you are running the command and all global values.

```
sfdx config:list
```

```
=== Config
Name                  Value                 Location
____________________  ____________________  ________

apiVersion            50.0                  Local
defaultdevhubusername my-dev-hub@force.com  Global
```

To return one or more previously set configuration values, use `config:get`. It is often useful to specify JSON output for this command for easier parsing in a continuous integration (CI) environment. For example, to return the value of `defaultusername` and `defaultdevhubusername`:

```
sfdx config:get defaultusername defaultdevhubusername --json
```

To unset a configuration value, run the `config:unset` command. For example, to unset the `instanceUrl` configuration value:

```
sfdx config:unset instanceUrl
```

Note: Alternately, you can set all CLI configuration values as environment variables. Environment variables override configuration values.

You can set these CLI configuration values.

apiVersion

The API version for a specific project or all projects. Normally, the CLI assumes that you're using the same version of the CLI as the Dev Hub org.

Let's say you decide to use the pre-release version of the CLI (v43 in Summer '18). But your Dev Hub org is running the current API version (v42 in Spring '18). In this case, set this value to match the API version of your Dev Hub org (v42).

This example sets the API version for all projects (globally).

```
sfdx config:set apiVersion=42.0 --global
```

Be sure not to confuse this CLI configuration value with the `sourceApiVersion` project configuration value, which has a similar name.

Environment variable: SFDX_API_VERSION

```
SFDX_API_VERSION=42.0
```

defaultusername

The username for an org that all commands run against by default.

```
sfdx config:set defaultusername=test-scratch-org@example.com
```

Environment variable: SFDX_DEFAULTUSERNAME

```
SFDX_DEFAULTUSERNAME=test-scratch-org@example.com
```

defaultdevhubusername

The username for your default Dev Hub org.

```
sfdx config:set defaultdevhubusername=my-dev-hub@devhub.org
```

Environment variable: SFDX_DEFAULTDEVHUBUSERNAME

```
SFDX_DEFAULTDEVHUBUSERNAME=my-dev-hub@devhub.org
```

disableTelemetry

By default, the CLI collects usage information, user environment information, and crash reports. This option enables you to opt out.

```
sfdx config:set disableTelemetry=true
```

Environment variable: SFDX_DISABLE_TELEMETRY

instanceUrl

The URL of the Salesforce instance that is hosting your org. Default value is `https://login.salesforce.com`.

```
sfdx config:set instanceUrl=https://yoda.my.salesforce.com
```

Environment variable: SFDX_INSTANCE_URL

```
SFDX_INSTANCE_URL=https://yoda.my.salesforce.com
```

maxQueryLimit

The maximum number of Salesforce records returned by a CLI command. Default value is 10,000.

For example, let's say you run `sfdx force:mdapi:listmetadata -m Role` on a Salesforce org that has 15,000 roles. By default the command displays only 10,000 roles. A message warns you that the command retrieved only some of the roles. To see all of them, set this config value to a larger number.

```
sfdx config:set maxQueryLimit=20000
```

Environment variable: SFDX_MAX_QUERY_LIMIT

```
SFDX_MAX_QUERY_LIMIT=200000
```

restDeploy

If `true`, the CLI uses Metadata REST API for deployments. By default, Salesforce CLI uses SOAP. Deployments using REST aren't bound by the 39 MB `.zip` file size limit that applies to SOAP deployments.

```
sfdx config:set restDeploy=true
```

Environment variable: SFDX_REST_DEPLOY

```
SFDX_REST_DEPLOY=true
```

SEE ALSO:

Disable CLI Data Collection and Metrics

CLI Parameter Resolution Order

Environment Variables

You can set environment variables to configure certain values that Salesforce CLI and Salesforce DX tooling use.

Salesforce CLI Environment Variables

Environment variables override CLI runtime configuration values. To set an environment variable for only the command you're running, append the variable, like this:

```
SFDX_API_VERSION=44.0 sfdx force:org:create -<options>
```

FORCE_OPEN_URL
Specifies the web page that opens in your browser when you run `force:org:open`. For example, to open Lightning Experience, set to `lightning`.

Equivalent to the `--path` parameter of `force:org:open`.

FORCE_SHOW_SPINNER
Set to `true` to show a spinner animation on the command line when running asynchronous CLI commands. Default is `false`.

FORCE_SPINNER_DELAY
Specifies the speed of the spinner in milliseconds. Default is 60.

SFDX_API_VERSION
The API version for a specific project or all projects. Normally, the Salesforce CLI assumes that you're using the same version of the CLI as your production org. However, let's say you decide to use the pre-release version of the CLI (v43 in Summer '18), but your production org is running the current API version (v42 in Spring '18). In this case, you'd want to set this value to match the API version of your production org (v42).

SFDX_AUDIENCE_URL
Overrides the **aud** (audience) field used for JWT authentication so that it matches the expected value of the authorization server URL for the org you're logging into. For example, `http://login.salesforce.com` for a production org or `https://test.salesforce.com` for a sandbox.

SFDX_CODE_COVERAGE_REQUIREMENT
Specifies the code coverage percentages that are displayed in green when you run `force:apex:test:run` or `force:apex:test:report` with the `--codecoverage` parameter.

If the code coverage percentage for an Apex test is equal to or higher than this setting, it's displayed in green. If the percent is lower, it's displayed in red. Applies only to human-readable output. Default is 70%.

SFDX_CONTENT_TYPE
All CLI commands output results in JSON format.

SFDX_DEFAULTDEVHUBUSERNAME
Specifies the username of your default Dev Hub org so you don't have to use the `--targetdevhubusername` CLI parameter. Overrides the value of the `defaultdevhubusername` runtime configuration value.

SFDX_DEFAULTUSERNAME
Specifies the username of your default org so you don't have to use the `--targetusername` CLI parameter. Overrides the value of the `defaultusername` runtime configuration value.

SFDX_DISABLE_AUTOUPDATE or SFDX_AUTOUPDATE_DISABLE (either var works)
Set to `true` to disable the auto-update feature of the CLI. By default, the CLI periodically checks for and installs updates.

SFDX_DISABLE_SOURCE_MEMBER_POLLING
Set to `true` to disable polling of your org's SourceMember object when you run the `force:source:push|pull` commands.

The commands poll the SourceMember object to track what's changed between your local source and the org after the push or pull completes. If you have a large metadata deployment, however, the polling can take a while, or even time out. Sometimes you don't require source tracking at all, such as in a CI/CD job. These use cases are good candidates for setting this environment variable.

The environment variable works with both scratch orgs and sandboxes.

Warning: When you disable SourceMember polling, the CLI's internal tracking of what's changed between your local source and org metadata gets out of sync. As a result, subsequent runs of the `force:source:push|pull|status` commands are unreliable, and it's up to you to synchronize your source. To reset source tracking, use the `force:source:tracking:reset` command.

SFDX_DISABLE_TELEMETRY

Set to `true` to disable the CLI from collecting usage information, user environment information, and crash reports.

SFDX_DOMAIN_RETRY

Specifies the time, in seconds, that the CLI waits for the Lightning Experience custom domain to resolve and become available in a newly created scratch org.

The default value is 240 (4 minutes). Set the variable to 0 to bypass the Lightning Experience custom domain check entirely.

SFDX_IMPROVED_CODE_COVERAGE

Scopes Apex test results to the classes entered during a test run when running `force:apex:test:run` and `force:apex:test:report`. Set to `true` to improve code coverage.

SFDX_INSTANCE_URL

The URL of the Salesforce instance that is hosting your org. Default value is `https://login.salesforce.com`.

SFDX_JSON_TO_STDOUT

Sends messages when Salesforce CLI commands fail to stdout instead of stderr. Setting this environment variable to `true` is helpful for scripting use cases.

Example:

```
SFDX_JSON_TO_STDOUT=true
```

SFDX_LOG_LEVEL

Sets the level of messages that the CLI writes to the log file.

Example:

```
SFDX_LOG_LEVEL=debug
```

SFDX_MAX_QUERY_LIMIT

The maximum number of Salesforce records returned by a CLI command. Default value is 10,000.

Example:

```
SFDX_MAX_QUERY_LIMIT=200000
```

SFDX_MDAPI_TEMP_DIR

Places the files (in metadata format) in the specified directory when you run some CLI commands, such as `force:source:<name>`. Retaining these files can be useful for several reasons. You can debug problems that occur during command execution. You can use the generated `package.xml` when running subsequent commands, or as a starting point for creating a manifest that includes all the metadata you care about.

```
SFDX_MDAPI_TEMP_DIR=/users/myName/myDXProject/metadata
```

SFDX_NPM_REGISTRY

Sets the URL to a private npm server, where all packages that you publish are private. We support only repositories that don't require authentication.

```
SFDX_NPM_REGISTRY=<full_URL>
```

Example:

```
SFDX_NPM_REGISTRY=http://mypkgs.myclient.com/npm/my_npm_pkg
```

Verdaccio is an example of a lightweight private npm proxy registry.

SFDX_PRECOMPILE_ENABLE

Set to `true` to enable Apex pre-compile before the tests are run. This variable works with the `force:apex:test:run` command. Default is `false`.

Important: The duration of an Apex test pre-compilation can be inconsistent. As a result, runs of the same Apex tests are sometimes quick and other times they time out. We recommend that you set this variable to `true` only if your Apex tests (without pre-compile) activate multiple concurrent Apex compilations that consume many system resources.

SFDX_PROJECT_AUTOUPDATE_DISABLE_FOR_PACKAGE_CREATE

For `force:package:create`, disables automatic updates to the `sfdx-project.json` file.

SFDX_PROJECT_AUTOUPDATE_DISABLE_FOR_PACKAGE_VERSION_CREATE

For `force:package:version:create`, disables automatic updates to the `sfdx-project.json` file.

SFDX_SOURCE_MEMBER_POLLING_TIMEOUT

Set to the number of seconds you want the `force:source:push` command to keep polling the SourceMember object before the command times out. The `force:source:push` command polls the SourceMember object to track what's changed between your local source and the org after the push completes. The CLI calculates a time-out for each `force:source:push` command run based on the number of components it deploys. Use this environment variable to override the calculated time-out.

For example, if the push times out after 3 minutes, try setting a time-out of 5 minutes (300 seconds):

```
SFDX_SOURCE_MEMBER_POLLING_TIMEOUT=300
```

SFDX_USE_GENERIC_UNIX_KEYCHAIN

(Linux and macOS only) Set to `true` if you want to use the generic UNIX keychain instead of the Linux `libsecret` library or macOS keychain. Specify this variable when using the CLI with ssh or "headless" in a CI environment.

SFDX_USE_PROGRESS_BAR

For `force:mdapi:deploy`, `force:source:deploy`, and `force:source:push`, set to `false` to disable the progress bar.

Examples:

To set globally: `SFDX_USE_PROGRESS_BAR=false`.

To set for a single command: `SFDX_USE_PROGRESS_BAR=false sfdx force:source:deploy`.

General Environment Variables

HTTP_PROXY

If you receive an error when you install or update the Salesforce CLI on a computer that's behind a firewall or web proxy, set this environment variable. Use the URL and port of your company proxy, for example:

```
http://username:pwd@proxy.company.com:8080
```

HTTPS_PROXY

If you receive an error when you install or update the Salesforce CLI on a computer that's behind a firewall or web proxy, set this environment variable. Use the URL and port of your company proxy, for example:

```
http://username:pwd@proxy.company.com:8080
```

NODE_EXTRA_CA_CERTS

Installs your self-signed certificate. Indicate the fully qualified path to the certificate file name. Then run `sfdx update`.

See NODE_EXTRA_CA_CERTS=file for more details.

NODE_TLS_REJECT_UNAUTHORIZED

Indicate `0` to allow Node.js to use the self-signed certificate in the certificate chain.

SEE ALSO:

Log Messages and Log Levels

Support for JSON Responses

CLI Parameter Resolution Order

Because you can specify parameters for a given CLI command in several ways, it's important to know the order of parameter resolution.

The CLI resolves command-line parameters and arguments, definition files, environment variables, and settings in this order:

1. Command-line parameters and arguments, such as `--loglevel`, `--targetusername`, or `sandboxName=FullSbx`.
2. Parameters and options listed in a file specified on the command line. An example is a scratch org definition in a file specified by the `--definitionfile` parameter of `force:org:create`. If you indicate a parameter or an argument on the command line that differs from what exists in the definition file, the command line takes precedence.
3. Environment variables, such as SFDX_LOG_LEVEL.
4. Local CLI configuration values, such as `defaultusername` or `defaultdevhubusername`. To view the local values, run `config:list` from your project directory.
5. Global CLI configuration values. To view the global values, run `config:list` from any directory.

Remember, command-line parameters are at the top of the precedence list. For example, if you set the SFDX_LOG_LEVEL environment variable to INFO but you specify `--loglevel DEBUG` when running a command, the log level is DEBUG.

If you specify the `--targetusername` parameter for a specific CLI command, the CLI command connects to an org with that username. It does not connect to the org previously set using `defaultusername`.

SEE ALSO:

CLI Runtime Configuration Values

Environment Variables

Support for JSON Responses

Salesforce CLI commands typically display their output to the console (stdout) in non-structured, human-readable format. Messages written to the log file (`stderr`) are always in JSON format.

To view the console output in JSON format, specify the `--json` parameter for a particular CLI command.

```
sfdx force:org:display --json
```

Most CLI commands support JSON output. To confirm, run the command with the `--help` parameter to view the supported parameters.

To get JSON responses to all Salesforce CLI commands without specifying the `--json` option each time, set the SFDX_CONTENT_TYPE environment variable.

```
export SFDX_CONTENT_TYPE=JSON
```

Log Messages and Log Levels

Salesforce CLI writes all log messages to the `USER_HOME_DIR/.sfdx/sfdx.log` file. CLI invocations append log messages to this running log file. Only errors are output to the terminal or command window from which you run the CLI.

Important: The files in the `USER_HOME_DIR/.sfdx` directory are used internally by Salesforce CLI. Do not remove or edit them.

The default level of log messages is `warn`. You can set the log level to one of the following, listed in order of least to most information. The level is cumulative: for the `debug` level. The `--loglevel` parameter supports parameter values in only lowercase (due to the migration to oclif). To assist you with the transition, we support uppercase parameters in Spring '19 but plan to deprecate support for them in Summer '19.

- error
- warn
- info
- debug
- trace
- fatal

You can change the log level in two ways, depending on what you want to accomplish.

To change the log level for the execution of a single CLI command, use the `--loglevel` parameter. Changing the log level in this way does not affect subsequent CLI use. This example specifies debug-level log messages when you create a scratch org.

```
sfdx force:org:create --definitionfile config/project-scratch-def.json --loglevel debug
--setalias my-scratch-org
```

To globally set the log level for all CLI commands, set the SFDX_LOG_LEVEL environment variable. For example, on UNIX:

```
export SFDX_LOG_LEVEL=debug
```

Note: Salesforce CLI gathers diagnostic information about its use and reports it to Salesforce so that the development team can investigate issues. The type of information includes command duration and command invocation counts.

Disable CLI Data Collection and Metrics

Salesforce collects usage data and metrics (telemetry) to help improve Salesforce CLI. We collect anonymous information related to the use of the CLI and plug-ins, such as which commands and parameters were run, and performance and error data.

We use these data to improve the CLI by looking at trends in command executions and how the CLI is configured. We also research error data to improve the CLI and to create efficiencies in our work (and yours). You are automatically enrolled in telemetry when you use the CLI.

If you would prefer to opt out of telemetry, set the `disableTelemetry` configuration value to `true`.

```
sfdx config:set disableTelemetry=true
```

Alternatively, you can opt out via an environment variable: `SFDX_DISABLE_TELEMETRY=true`.

CHAPTER 4 Update the CLI

In this chapter ...

If you want to ensure that you are running the latest version of Salesforce CLI, you can manually update it.

If You Installed Salesforce CLI Using the Installer

To install the latest Salesforce CLI version, run:

```
sfdx update
sfdx-cli: Updating CLI from 6.0.0-0743bea5 to 6.0.0-aebbfd66
(alpha)... done
sfdx-cli: Updating CLI... already on latest version: 6.0.0-aebbfd66
sfdx-cli: Updating plugins... done
```

By default, the CLI periodically checks for and installs updates. To disable auto-update, set the `SFDX_AUTOUPDATE_DISABLE` environment variable to `true`.

If You Installed Salesforce CLI Using npm

The auto-update option is not available. To update Salesforce CLI using npm, run:

```
npm install --global sfdx-cli
```

Windows: Improve CLI Update Speed

Anti-virus software, such as Windows Defender, slows a CLI update because it scans all the files as they are downloaded. To improve the update speed, add the `%LOCALAPPDATA%\sfdx` directory to the list of directories that are excluded from the scan. We recommend that you consult your security administrator before making this configuration change.

Disable Automatic Update of the CLI and Plug-In

When you run a command, Salesforce CLI checks to see if you have the latest version. If not, the CLI automatically updates itself and the `salesforcedx` plug-in. You can disable this automatic update with an environment variable.

To remain on the current version of the CLI and disable automatic updates, set the `SFDX_AUTOUPDATE_DISABLE` environment variable to `true`. How you set an environment variable is different for different operating systems. See the operating system vendor's help for instructions on how to set environment variables.

CHAPTER 5 Uninstall the CLI Binary or Plug-In

Uninstalling the CLI removes it entirely from your computer.

macOS or Linux

Enter all these commands in a terminal:

```
sudo rm -rf /usr/local/sfdx
sudo rm -rf /usr/local/lib/sfdx
sudo rm -rf /usr/local/bin/sfdx
sudo rm -rf ~/.local/share/sfdx ~/.config/sfdx ~/.cache/sfdx
sudo rm -rf ~/Library/Caches/sfdx
```

Windows

1. Select **Start > Control Panel > Programs > Programs and Features**.
2. Select **SFDX CLI**, and click **Uninstall**.
3. Inside your home directory, delete the `.config\sfdx` directory.

If the CLI is still installed, delete the `%LOCALAPPDATA%\sfdx` directory in Program Files.

npm

If you installed Salesforce CLI with npm, uninstall it with this command:

```
npm uninstall sfdx-cli --global
```

Uninstall the `salesforcedx` Plug-In

Enter this command from a terminal or Windows command prompt:

```
sfdx plugins:uninstall salesforcedx
```

CHAPTER 6 Troubleshoot Salesforce CLI

In this chapter ...

Here's a list of Salesforce CLI errors and how to fix them.

CLI Version Information

Use these commands to view version information about Salesforce CLI.

```
sfdx plugins --core      // Versions for all installed plug-ins and sfdx-cli
sfdx plugins             // salesforcedx plug-in versions
sfdx --version     // sfdx-cli version
```

Error: API Version Mismatch

If you update the `salesforcedx` plug-in and try to push source from your local DX project to a scratch org, you see an API version error.

EDITIONS

Available in: All Editions

```
sfdx force:source:push
ERROR running force:source:push: The configured apiVersion 48.0 is not supported for this
 org. The max apiVersion is 47.0
```

What happened?

Answer: Your locally configured `apiVersion` is greater than your org's supported max `apiVersion`. To troubleshoot, run these commands:

- Run `sfdx force` to check what `salesforcedx` plug-in version you have installed locally.
- Run `sfdx config:list` to determine if the `apiVersion` is overridden.

To resolve the error for a specific project, set the `apiVersion` locally:

```
sfdx config:set apiVersion=47.0
```

To resolve the error for all Salesforce DX projects, set the `apiVersion` globally:

```
sfdx config:set apiVersion=47.0 -g
```

Remember, after you uninstall the pre-release version, update the CLI to the latest version and reset the global `apiVersion`:

```
sfdx plugins:install salesforcedx@latest
sfdx config:set apiVersion= -g
```

Run CLI Commands on macOS Sierra (Version 10.12)

Some users who upgrade to macOS Sierra can't execute CLI commands. This is a general problem and not isolated to Salesforce DX. To resolve the issue, reinstall your Xcode developer tools.

Execute this command in Terminal:

```
xcode-select --install
```

If you still can't execute CLI commands, download the **Command Line Tools (macOS sierra) for Xcode 8** package from the Apple Developer website.

SEE ALSO:

- Apple Developer Downloads
- Stack Overflow: Command Line Tools bash (git) not working - macOS Sierra Final Release Candidate

Error: Command Failed with ENOENT

After recently installing Salesforce CLI, you get this error when you run a command such as `force:project:create`.

```
ERROR running force:project:create:  Command failed with ENOENT: npm root -g --prefix
/Users/johndoe/Documents/sfdx_workspaces/.yo-repository --loglevel error
spawnSync npm ENOENT
```

Answer: Install Node.js.

CHAPTER 7 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 in any major update of the `salesforcedx` plug-in.
- Salesforce removes the deprecated command or parameter in the next major release of the `salesforcedx` plug-in. For example, if Salesforce deprecates a command in version 41, it does not appear in version 42.
- 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.

CHAPTER 8 Next Steps

Read on to learn what to do after you've installed Salesforce CLI.

Check out the examples in the Sample Gallery. The gallery contains sample apps, such as Dreamhouse, that show what you can build on the Salesforce platform. They're continuously updated to incorporate the latest features and best practices.

Ramp up quickly on Salesforce CLI with the Quick Start: Salesforce DX Trailhead project. Then dive right into development with the Build Apps Together with Package Development trail.

Looking for a more visual developer experience? We got you covered! Check out Salesforce Extensions for VS Code, which is built on Salesforce CLI.

Read the Salesforce DX documentation:

- *Salesforce DX Developer Guide* to learn how to manage and develop apps on the Salesforce Platform across their entire life cycle.
- *Salesforce CLI Command Reference* for the complete list of CLI commands and how to use them.
- *Salesforce CLI Plug-In Developer Guide* to learn how to develop your own plug-ins for Salesforce CLI.

Salesforce DX Development Pathways

Salesforce CLI is a powerful tool that you can use to develop applications in many different ways. Here are some common pathways, with the required steps to get you started and suggestions on what to do next.

Get Started: Use Scratch Orgs for Development

A scratch org is a source-driven and disposable deployment of Salesforce code and metadata. Scratch orgs drive developer productivity and collaboration during the development process, and facilitate automated testing and continuous integration.

1. As the Admin user, enable Dev Hub in your Developer Edition, trial, or production org (if you're a customer), or your business org (if you're an AppExchange partner).

   If you don't have an org, sign up for a free Developer Edition org on the Salesforce Developers web site.

2. If you want your dev team to create scratch orgs, add them to your Dev Hub org.
3. (Optional) Turn on Einstein Features in your Dev Hub to eliminate the manual steps for enabling the chatbot feature in scratch orgs.
4. Clone a sample Salesforce DX project from GitHub and try out the most common CLI commands. Then check out the *Salesforce DX Developer Guide* and learn about Salesforce DX project configuration, scratch orgs, synchronizing your code, and other developer topics.

Get Started: Develop Second-Generation Managed Packages

As an AppExchange partner, use second-generation managed packaging (2GP) to organize your source, build small modular packages, integrate with your version control system, and better utilize your custom Apex code.

1. As the Admin user, enable Dev Hub in your Developer Edition, trial, or production org (if you're a customer), or your business org (if you're an AppExchange partner).

   If you don't have an org, sign up for a free Developer Edition org on the Salesforce Developers web site.

2. In the Dev Hub, enable Second-Generation Packaging.
3. If you want your dev team to create 2GP managed packages add them to your Dev Hub org.
4. Read all about 2GP managed packages and how to create them in the *Salesforce DX Developer Guide*.

Get Started: Use a Sandbox

Sandboxes are copies of your Salesforce org that you can use for development, testing, and training, without compromising the data and applications in your production org. You can turn on source tracking in your production org so Developer and Developer Pro sandboxes automatically track changes between the production org and your local development workspace.

1. Enable source tracking in your production org.
2. Learn how to use Salesforce CLI to create, manage, and develop with sandboxes by reading the *Salesforce DX Developer Guide*.

SEE ALSO:

Salesforce DX Developer Guide

Salesforce CLI Command Reference

Salesforce CLI Plug-In Developer Guide