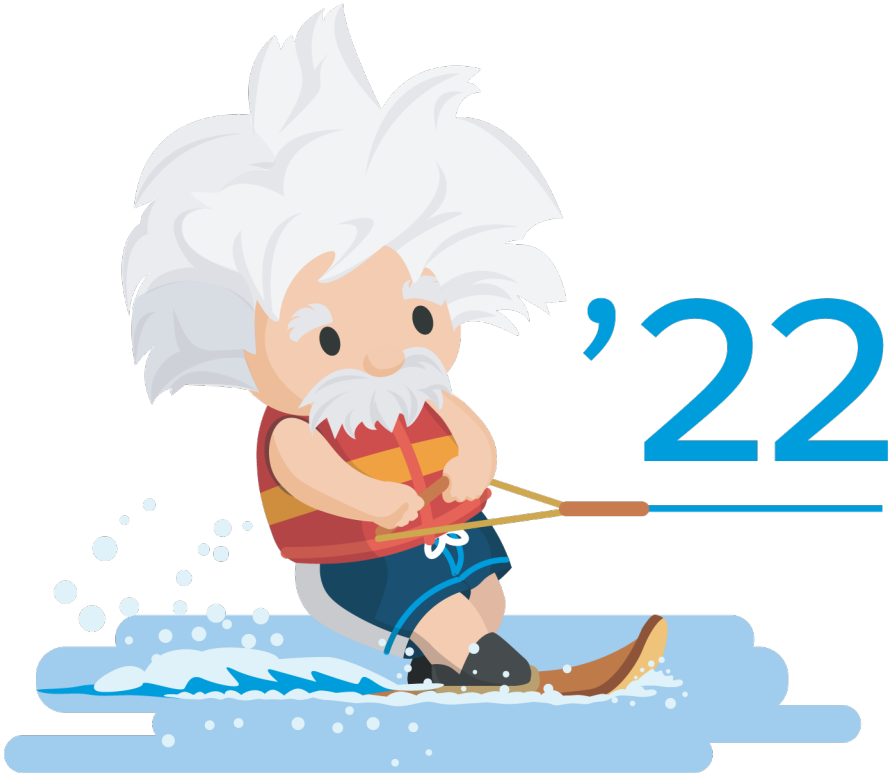
Salesforce CLI Setup Guide

Version 55.0, Summer '22



@salesforcedocs

Last updated: May 12, 2022

CONTENTS

CHAPTER 1 Before You Begin

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, synchronize source to and from orgs, create and install packages, and more.

Salesforce 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.

Salesforce CLI is a bundle of two executables: `sf` and `sfdx`. We first launched `sfdx` to provide you the ability to develop and test your apps more easily on Salesforce Platform. But if you want to work across *all* Salesforce clouds, `sfdx` doesn't provide all the commands you need. With `sf`, we're bringing together a cross-cloud set of commands that streamline how you build and deploy across Salesforce. See Get Started with Salesforce CLI Unification for more information about the `sf` executable, including how to install it.

We release new versions of the CLI and plug-ins weekly. Read the weekly release notes to learn about recent and upcoming changes.

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 8.1 and Windows Server 2012 (64-bit and 32-bit) or later
- Mac—macOS 10.11 or later (Intel and M1)
- 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.

Node.js

We bundle Node.js in each operating system-specific Salesforce CLI installer. We include the version of Node.js with Active LTS status and update it in tandem with the Node.js release schedule.

If you prefer to install Salesforce CLI using `npm`, we recommend you also use the Active LTS version of Node.js.

CLI Version Support

Salesforce supports only the most current version of Salesforce CLI. See the Salesforce CLI Release Notes for the latest version information.

SEE ALSO:

- *Salesforce Extensions for Visual Studio Code*
- *Salesforce DX Developer Guide*
- *Salesforce CLI Command Reference*
- oclif: The Open CLI Framework

CHAPTER 2 Install Salesforce CLI

In this chapter ...

Install the CLI on your computer using operating system-specific artifacts, such as `.pkg` on macOS, or with npm.

Choose one method to install on your computer. For example, don't install on macOs with both a `.pkg` and npm.

Install the CLI on macOS

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

Note: The macOS installer installs both the `sfdx` and `sf` executables.

1. Download the `.pkg` file.
2. Run the `.pkg` file, such as double-clicking it from Finder, and answer all the prompts.
3. After the installation completes, restart your Terminal windows or IDEs to make sure the Salesforce CLI executables are available.

SEE ALSO:

Verify Your Installation

Disable Automatic Update of the CLI

Install the CLI on Windows

Install Salesforce CLI on Windows with an `.exe` file.

Note: The Windows `.exe` installer installs both the `sfdx` and `sf` executables.

1. Download the `.exe` file.
2. Run the `.exe` file, such as double-clicking it from Windows Explorer, and answer all the prompts.
3. After the installation completes, restart your command prompts, PowerShell windows, or IDEs to make sure the Salesforce CLI executables are available.

Warning: Salesforce CLI works best within the native Windows command prompt (`cmd.exe`) and the Microsoft Windows PowerShell. We don't recommend using 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

Install the CLI with a TAR File

Salesforce CLI distributes TAR files that you can install on all supported operating systems. On Linux, the only way to install Salesforce CLI is with a TAR file.

Note: These TAR files bundle the `sfdx` and `sf` executables together.

Use this table to find the unversioned URLs for the TAR file (`.tar.gz` or `.tar.xz`) for your operating system. When we release a new version of Salesforce CLI every week, we also update these URLs so they point to the most up-to-date version. Unversioned URLs are especially useful for CI use cases. The table also includes manifest URLs that show the versioned URL for each file.

Operating System	Tar Files	Manifest
Linux	• sfdx-linux-x64.tar.gz • sfdx-linux-x64.tar.xz • sfdx-linux-arm.tar.gz • sfdx-linux-arm.tar.xz	• sfdx-linux-x64-buildmanifest • sfdx-linux-arm-buildmanifest
macOS	• sfdx-darwin-x64.tar.gz • sfdx-darwin-x64.tar.xz	• sfdx-darwin-x64-buildmanifest
Windows	• sfdx-win32-x64.tar.gz • sfdx-win32-x64.tar.xz • sfdx-win32-x86.tar.gz • sfdx-win32-x86.tar.xz	• sfdx-win32-x64-buildmanifest • sfdx-win32-x86-buildmanifest

Important: We highly recommended that you use the installers or npm to install Salesforce CLI on Windows. If, however, you decide to use the Windows TAR files, you must first install a separate program, such as 7Zip, to extract the file contents.

In these examples it's assumed that you're installing Salesforce CLI on Linux and in the `sfdx` subdirectory of your home directory.

1. Download or `wget` one of these TAR files.

```
wget
https://developer.salesforce.com/media/salesforce-cli/sfdx/channels/stable/sfdx-linux-x64.tar.xz
```

2. Create the directory where you want to install Salesforce CLI.

```
mkdir ~/sfdx
```

3. Unpack the contents for your TAR file:

```
tar xJf sfdx-linux-x64.tar.xz -C ~/sfdx --strip-components 1
```

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

Note: This example shows just one possible set of flags for the `tar` command on Linux. For other options on your operating system, refer to the `tar` documentation.

4. Update your PATH environment variable to include the Salesforce CLI `bin` directory. For example, to set it for your current terminal session:

```
export PATH=~/sfdx/bin:$PATH
```

To update your PATH permanently, add the appropriate entry to your shell's configuration file. For example, if you use the Bash shell, add this line to your `~/.bashrc` or `~/.bash_profile` file:

```
PATH=~/sfdx/bin:$PATH
```

SEE ALSO:

Verify Your Installation

Disable Automatic Update of the CLI

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.

Note: When installing Salesforce CLI with `npm`, you install the `sfdx` and `sf` executables separately using the commands shown in the following steps. In contrast, the Salesforce CLI installers and TAR files bundle the two executables together.

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 to install the `sfdx` executable.

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

   Run this command to install the `sf` executable.

   ```
   npm install @salesforce/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

Install Older Versions of Salesforce CLI

We recommend that you always use the latest version or release candidate of Salesforce CLI. However, we also understand that sometimes you might require an older version of the CLI. For these use cases, we publish JSON files that list the download URLs for recent versions of the installers and TAR files for each supported operating system.

Each week we add the latest released version to the lists; versions remain on the list for 20 weeks. We keep the TAR and installer files themselves for 40 weeks minimum.

Note: We continue to keep all old versions of the sfdx-cli npm package.

Operating System	File Type	TAR Compression Type	Link to JSON File
Linux ARM	TAR	gz	sfdx-linux-arm-tar-gz.json
Linux ARM	TAR	xz	sfdx-linux-arm-tar-xz.json
Linux 64	TAR	gz	sfdx-linux-x64-tar-gz.json
Linux 64	TAR	xz	sfdx-linux-x64-tar-xz.json
Windows 64	TAR	gz	sfdx-win32-x64-tar-gz.json
Windows 64	TAR	xz	sfdx-win32-x64-tar-xz.json
Windows x86	TAR	gz	sfdx-win32-x86-tar-gz.json
Windows x86	TAR	xz	sfdx-win32-x86-tar-xz.json
macOS	TAR	gz	sfdx-darwin-x64-tar-gz.json
macOS	TAR	xz	sfdx-darwin-x64-tar-xz.json
Windows 64	Installer		sfdx-x64-exe.json
Windows x86	Installer		sfdx-x86-exe.json
macOS	Installer		sfdx-pkg.json

Verify Your Installation

Verify your Salesforce CLI installation to ensure you've installed it correctly.

Run this command to verify the Salesforce CLI version:

```
sfdx --version
sfdx-cli/7.144.0 darwin-x64 node-v16.14.2
```

Run this command to view the installed plug-ins and their versions:

```
sfdx plugins --core

@oclif/plugin-autocomplete 0.3.0 (core)
@oclif/plugin-commands 1.3.0 (core)
@oclif/plugin-help 3.3.1 (core)
@oclif/plugin-not-found 1.2.6 (core)
@oclif/plugin-plugins 1.10.11 (core)
@oclif/plugin-update 1.5.0 (core)
@oclif/plugin-warn-if-update-available 2.0.4 (core)
@oclif/plugin-which 1.0.4 (core)
@salesforce/sfdx-plugin-lwc-test 0.1.7 (core)
alias 1.2.1 (core)
apex 0.11.0 (core)
auth 1.8.1 (core)
community 1.1.4 (core)
config 1.3.23 (core)
```

```
custom-metadata 1.0.12 (core)
data 0.6.10 (core)
generator 1.2.2 (core)
info 1.3.1 (core)
limits 1.3.0 (core)
org 1.11.2 (core)
salesforce-alm 54.0.2 (core)
schema 1.1.0 (core)
sfdx-cli 7.144.0 (core)
source 1.9.0 (core)
telemetry 1.4.0 (core)
templates 54.3.0 (core)
user 1.7.1 (core)
```

Run this command to view all available Salesforce CLI commands:

```
sfdx commands
```

Run this command to display the release notes for your current Salesforce CLI version:

```
sfdx whatsnew
```

SEE ALSO:

Salesforce CLI Plug-ins

CHAPTER 3 Install the CLI Release Candidate

We release a new version of the CLI weekly. At the same time we also publish a release candidate of the CLI that contains changes that we plan to include in the next weekly release. Think of the release candidate as the CLI-version of the Salesforce sandbox preview. You can install a release candidate if you want to check out upcoming features. Or stay on the current and official release. Or go back and forth. It's up to you!

We recommend you run your continuous integration (CI) jobs against both the current release and the release candidate to identify potential breaking changes before they happen. The current release has the `stable` tag; the release candidate has the `stable-rc` tag.

If you've already installed the CLI and are using the current release, run this command to switch to the release candidate.

```
sfdx update stable-rc
```

To uninstall the release candidate and return to the current version, run this command.

```
sfdx update stable
```

Install Using npm

Run this command to install the release candidate using npm.

```
npm install --global sfdx-cli@latest-rc
```

To uninstall the release candidate and return to the current version, run this command.

```
npm install --global sfdx-cli@latest
```

Install from a TAR File

Salesforce CLI distributes TAR files for the release candidate that you can install on all supported operating systems. The download URLs are similar to the URLs for installing the current release, but use the `stable-rc` channel rather than the `stable` channel.

For example, to `wget` the Linux TAR file for the release candidate, run this command, which downloads from the `stable-rc` channel.

```
wget
https://developer.salesforce.com/media/salesforce-cli/sfdx/channels/stable-rc/sfdx-linux-x64.tar.xz
```

Other than using a different channel, the instructions for installing the release candidate from a TAR file are the same as the instructions for installing the current release.

SEE ALSO:

Trailhead: Get Early Access with the Sandbox Preview

CHAPTER 4 Run Salesforce CLI Using a Docker Image

Salesforce publishes Docker container images for Salesforce CLI on Docker Hub. We follow the same release process as our installers and npm packages. Each week we publish a Docker container image for that week's release candidate (`latest-rc`). The following week we retag the image as `latest`. You can run the `latest` or `latest-rc` CLI versions, or a specific numbered version.

For each Salesforce CLI version, we provide two flavors:

- slim—The CLI installed on Linux with a TAR file, plus OpenJDK 11.
- full—The CLI installed on Linux with npm on a full Node.js installation, plus OpenJDK 11 and additional utilities such as jq.

Refer to this Web page or the following table to determine the name of the image you want to use.

Salesforce CLI Version Type	Docker Hub Image Name
Slim latest release	`salesforce/salesforcedx:latest-slim`
Full latest release	`salesforce/salesforcedx:latest-full`
Slim release candidate	`salesforce/salesforcedx:latest-rc-slim`
Full release candidate	`salesforce/salesforcedx:latest-rc-full`
Slim specific version, such as 7.112.0	`salesforce/salesforcedx:7.112.0-slim`
Full specific version, such as 7.112.0	`salesforce/salesforcedx:7.112.0-full`

For example, to pull and run the slim CLI release candidate image:

```
docker pull salesforce/salesforcedx:latest-rc-slim
docker run -it salesforce/salesforcedx:latest-rc-slim
```

Then you can run Salesforce CLI commands, such as:

```
sfdx version
```

To exit the Salesforce CLI Docker container:

```
exit
```

You can also remotely execute commands from outside the container after you have it running and know the container ID:

```
docker exec -it 8b1e2696a243 bin/bash sfdx version
```

SEE ALSO:

Docker Hub: The official Dockerfile for Salesforce DX.

CHAPTER 5 Update Salesforce CLI

In this chapter ...

If you want to ensure that you're 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 and plug-in versions, run:

```
sfdx update

sfdx-cli: Updating CLI from 7.105.1-32db2396ed to
7.106.1-2fb9e41053... done
sfdx-cli: Updating CLI... done
```

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

When you update Salesforce CLI, we automatically display the release notes for the version you're updating to so you can learn about the new, changed, and fixed features. To silence the display, set the `SFDX_HIDE_RELEASE_NOTES` and `SFDX_HIDE_RELEASE_NOTES_FOOTER` environment variables to `true`.

If You Installed Salesforce CLI Using npm

The auto-update option isn't available. When a new version of the CLI is available, run this command::

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

Determine How You Installed Salesforce CLI

Because the method to update Salesforce CLI differs depending on whether you used the installers or npm, you must know how you installed before you can update. In case you forgot, here are two ways to determine how you installed Salesforce CLI:

- Run `sfdx update`. If Salesforce CLI updates, then you installed with the installers. If the command returns this or similar warning, then you installed with npm:

  ```
  sfdx update
   ›   Warning: Use "npm update --global sfdx-cli" to update npm-based
   installations.
  sfdx-cli: Updating CLI... not updatable
  ```

- If you've installed Node.js, run `npm list -g --depth 0`. If the displayed list includes the entry `sfdx-cli@<version>`, then you installed Salesforce CLI with npm.

Disable Automatic Update of the CLI

When you run a command, Salesforce CLI checks to see if you have the latest version. If not, the CLI automatically updates itself. 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 6 Salesforce CLI Plug-ins

In this chapter ...

Salesforce CLI consists of an npm package called `sfdx-cli` and multiple plug-ins—also npm packages—that contain commands. Most of the core functionality that Salesforce CLI provides comes from plug-ins.

Some plug-ins are automatically installed when you install the CLI. These plug-ins contain commands that support source-driven development, such as creating and managing scratch orgs, synchronizing source code, creating second-generation packaging, and more. These commands are in the `force` namespace, such as `force:project:create`. Other core plug-ins contain commands that make it easier to use the CLI and are in their own namespace. For example, commands for setting configuration values (`config`) or aliases (`alias`) and authorizing orgs (`auth`).

See the Salesforce CLI Status page for a list of all the core CLI plug-ins, their GitHub repos, and their status.

You can install more plug-ins, such as 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. See **Salesforce CLI Plug-In Developer Guide**.

By default, the latest versions of the core plug-ins are installed when you install Salesforce CLI for the first time. Similarly, when you update the CLI to the latest version, the core plug-ins are also updated.

To determine the versions of the plug-ins currently installed in your CLI, run:

```
sfdx plugins --core

@oclif/plugin-autocomplete 0.3.0 (core)
@oclif/plugin-commands 1.3.0 (core)
@oclif/plugin-help 3.3.1 (core)
@oclif/plugin-not-found 1.2.6 (core)
@oclif/plugin-plugins 1.10.11 (core)
@oclif/plugin-update 1.5.0 (core)
@oclif/plugin-warn-if-update-available 1.7.3 (core)
@oclif/plugin-which 1.0.4 (core)
@salesforce/sfdx-plugin-lwc-test 0.1.7 (core)
alias 1.2.1 (core)
apex 0.8.0 (core)
auth 1.8.1 (core)
config 1.3.19 (core)
custom-metadata 1.0.12 (core)
data 0.6.8 (core)
generator 1.2.2 (core)
limits 1.3.0 (core)
org 1.11.1 (core)
salesforce-alm 53.9.0 (core)
schema 1.1.0 (core)
sfdx-cli 7.137.1 (core)
source 1.8.11 (core)
```

```
telemetry 1.4.0 (core)
templates 53.5.0 (core)
user 1.7.1 (core)
```

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.

Install Other Versions of Salesforce CLI Plug-ins

Sometimes you want to use a specific version of a plug-in. For example, let's say Salesforce fixed a bug in the `alias` plug-in but we haven't yet released the fix in the current CLI. But you want to test the bug fix in your local development environment. Follow these steps to install the version of the plug-in that has the fix.

1. To find a specific version of a plug-in, first find it on the Salesforce CLI Status page. Then navigate to its GitHub repo, such as salesforcecli/plugin-alias, which lists all the releases and tags.
2. Install the version that contains the bug fix. For example, to install version 1.1.10 of the `alias` plug-in, run this command:

   ```
   sfdx plugins:install alias@1.1.10
   ```

   The preceding example uses the plug-in's short name, which is shown in the output of `sfdx plugins --core`. You can also use the plug-in's long name, which is the `name` property in the plug-in's package.json file.

   ```
   sfdx plugins:install @salesforce/plugin-alias@1.1.10
   ```

3. When you're finished testing, go back to using the current version of the plug-in by uninstalling the tagged version.

   ```
   sfdx plugins:uninstall alias
   ```

   Important: When you install a specific plug-in version using a tag, such as `1.1.10`, you stay with that tag until you uninstall it.

SEE ALSO:

Discover Salesforce Plug-Ins

Salesforce CLI Plug-In Developer Guide

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",
    ...
]
```

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.

CHAPTER 7 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 isn't currently available on computers running Windows.

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

1. Run `sfdx autocomplete`.
2. Follow the displayed instructions.

If autocomplete doesn't 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're running the command and all global values.

```
sfdx config:list
```

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

apiVersion            54.0                  Local
defaultdevhubusername my-dev-hub@force.com  Global
```

To return one or more previously set configuration values, use `config:get`. It's 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.

This example sets the API version for all projects (globally) to 51.0.

```
sfdx config:set apiVersion=51.0 --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=51.0
```

customOrgMetadataTemplates

Specifies either a local directory or a cloned GitHub repository that contains the default custom code templates used by the `force:project:create` command. The GitHub URL points to either the root directory that contains your templates or to a subdirectory on a branch in the repo that contains your templates. For example:

```
sfdx config:set
customOrgMetadataTemplates=https://github.com/mygithubacct/salesforcedx-templates
```

Environment variable: SFDX_CUSTOM_ORG_METADATA_TEMPLATES

```
SFDX_CUSTOM_ORG_METADATA_TEMPLATES=https://github.com/mygithubacct/salesforcedx-templates
```

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`. We recommend that you set this value to the My Domain login URL for your org. You can find the My Domain login URL on the My Domain page in Setup.

```
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

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=50.0 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_ACCESS_TOKEN

Specifies an access token when using the `auth:accesstoken:store` command. If you don't set this environment variable, the command prompts you for the access token. Useful for CI/CD scripts.

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 Dev Hub.

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, `https://`***`MyDomainName`***`.my.salesforce.com` or `https://login.salesforce.com` for a production org, and `https://`***`MyDomainName--SandboxName`***`.sandbox.my.salesforce.com`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

When set to `JSON`, specifies that all CLI commands output results in JSON format. If you set the environment variable to any other value, or unset it, the CLI commands output their results as specified by the parameters.

SFDX_CUSTOM_ORG_METADATA_TEMPLATES

Specifies either a local directory or a cloned GitHub repository that contains the default custom code templates used by the `force:project:create` command. The GitHub URL points to either the root directory that contains your templates or to a subdirectory on a branch in the repo that contains your templates.

Example:

```
SFDX_CUSTOM_ORG_METADATA_TEMPLATES=https://github.com/mygithubacct/salesforcedx-templates
```

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_DNS_TIMEOUT

Specifies the number of seconds that the `force:org:*` commands wait for a response when checking whether an org is connected. If the commands don't receive a response in that time, they time out. Default value is 3.

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_HIDE_RELEASE_NOTES

Set to `true` to silence the automatic display of the release notes when you run `sfdx update`. Default value is `false`.

Example:

```
SFDX_HIDE_RELEASE_NOTES=true
```

SFDX_HIDE_RELEASE_NOTES_FOOTER

Set to `true` to silence the boilerplate footer about displaying the release notes when you run `sfdx update`. Default value is `false`.

Example:

```
SFDX_HIDE_RELEASE_NOTES_FOOTER=true
```

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`. We recommend that you set this value to the My Domain login URL for your org. You can find the My Domain login URL on the My Domain page in Setup.

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_LOG_ROTATION_PERIOD

Time period after which Salesforce CLI rotates the log file. Rotating the log file means making a backup copy of the file and then clearing out the current log file to start afresh. For example, if set to `1d`, Salesforce CLI rotates the log file daily at midnight. If set to `2w`, the file is rotated every 2 weeks. See the `period` entry in this table for other time period options. Default value is `1d`.

Example:

```
SFDX_LOG_ROTATION_PERIOD=2w
```

SFDX_LOG_ROTATION_COUNT

Number of backup files to keep when rotating the log file. Default value is `2`. See SFDX_LOG_ROTATION_PERIOD for more information

Example:

```
SFDX_LOG_ROTATION_COUNT=10
```

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_REST_DEPLOY

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

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

To allow Node.js to use the self-signed certificate in the certificate chain, indicate *0*.

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. Don't 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 doesn't 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.

Rotating Log Files

Salesforce CLI uses rotating log files. By default, every day at midnight the CLI makes a backup copy of the log file and then clears out its entries to start afresh. We keep backups for the past two days along with the current day's logs. This behavior ensures that the current log file doesn't get too large.

You can change the default behavior with these environment variables:

- SFDX_LOG_ROTATION_PERIOD: How often a new log file is created, such as the default value of one day (`1d`) or two weeks `2w`.
- SFDX_LOG_ROTATION_COUNT: Number of backup files to keep. Default value is 2.

For example, let's say you choose a rotation period of 2 weeks and a count of 2. These values ensure that you have backup logs for the four weeks before the first entry in the current log.

SEE ALSO:

Environment Variables

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're 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 --global
```

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

CHAPTER 8 Uninstall Salesforce CLI or Plug-ins

Uninstalling Salesforce 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
sudo rm -rf /usr/local/sf
sudo rm -rf /usr/local/bin/sf
```

Windows

1. Select **Start > Control Panel > Programs > Programs and Features**.
2. Select **SFDX CLI**, and click **Uninstall**.
3. Inside your home directory, delete these two directories:
   - `.config\sfdx`
   - `.config\sf`

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 a Plug-In

Use the `plugins:uninstall` command to uninstall a plug-in you've previously installed.

Let's say, for example, that you previously installed a specific version of the `auth` plug-in, but now you want to go back to the latest version. Uninstalling the plug-in takes you back to the core version that's bundled with the CLI. Enter this command from a terminal or Windows command prompt:

```
sfdx plugins:uninstall auth
```

If the plug-in is standalone and not bundled with the CLI, then uninstalling it removes it from the CLI.

CHAPTER 9 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      // Version of the CLI and all installed plug-ins
sfdx --version     // CLI version
```

Error: API Version Mismatch

If you update Salesforce CLI and try to push source from your local DX project to a scratch org, you see an API version error.

```
sfdx force:source:push
ERROR running force:source:push: The configured apiVersion 51.0 is not supported for this
 org. The max apiVersion is 50.0
```

What happened?

Answer: Your locally configured `apiVersion` is greater than your org's supported max `apiVersion`. To troubleshoot, first run this command to determine if your local `apiVersion` is overridden:

```
sfdx config:list
```

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

```
sfdx config:set apiVersion=50.0
```

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

```
sfdx config:set apiVersion=50.0 -g
```

To override the API version for a single CLI command execution, use the `--apiversion` parameter:

```
sfdx force:source:push --apiversion=50.0
```

Note: Not all commands support the `--apiversion` parameter.

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 10 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.

CHAPTER 11 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 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 lifecycle.
- *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.

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 website.

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 use 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 website.

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 sandbox.
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

CHAPTER 12 Get Started with Salesforce CLI Unification

In this chapter ...

Salesforce CLI is a bundle of two executables: `sf` and `sfdx`. The `sf` executable will eventually provide a single set of commands designed to develop and deploy applications across all Salesforce clouds. It currently supports Salesforce Platform and Salesforce Functions. The `sfdx` executable is the one you know and love, and have used to build applications for the Salesforce Platform.

We recommend that you start using `sf` commands in your daily work, just to play around with them and see how they work. If you create new CI/CD scripts, add the available `sf` commands. If you start using new features like Salesforce Functions that are available exclusively in `sf`, add those commands directly to your existing scripts if needed. Because `sf` and `sfdx` are bundled, adding new `sf` commands to existing scripts doesn't break them.

The two executables are aware of each other and some of their commands are interoperable. For example, if you create a scratch org with an `sfdx` command, you can deploy metadata to it with an `sf` command. See How sf and sfdx Work Together on page 43 for details.

But note that `sf` isn't an `sfdx` plug-in, and while some commands interoperate between the two executables, the two executables are separate entities. For example, if you install a plug-in into `sfdx` and use some of its commands, you don't also see those commands in `sf`. To use the commands in `sf` you must explicitly install the plug-in into `sf`.

How Does `sf` Improve on `sfdx`?

`sf` is similar to `sfdx` in that it's a command-line interface that simplifies Salesforce development and build automation. But it also includes some key new features and design changes.

- The `sf` command hierarchy reflects a typical developer's workflow rather than Salesforce brands, products, or features.

  For example, the top-level topics include configuring the CLI (`sf config`), logging into environments (`sf login`), deploying and retrieving (`sf deploy` and `sf retrieve`), and managing environments (`sf env`).

  As `sf` grows and embraces other Salesforce clouds, it will include their commands in this workflow-focused hierarchy instead of each product having its own command hierarchy. Because `sf` creates consistency across all these commands, it's easier and more intuitive to use, even when developing on a new cloud.

- `sf` provides interactive commands that actively prompt you for required information rather than passively accepting flag values. Now you don't have to remember all the flag names or which are required, which in turn reduces errors. For example, `sf deploy` prompts you for deployment environment and artifacts, level of testing, and so on.

Each interactive command has an environment-specific command for use in automated scripts. For example, `sf deploy metadata` has flags to specify the org you're deploying to, the metadata location, and so on.

- Improved `--help` output:
  - We've enhanced the command-line help to include additional information and examples. As a result, the help can get long, so we've also changed the behavior of the flags:
    - `-h`: Displays a subset of the full help: short command and flag descriptions and command usage. Great for quick reference information.
    - `--help`: Displays the `-h` content plus longer command and flag descriptions, examples, and the configuration and environment variables that affect the command.
  - The long `--help` output contains new CONFIGURATION VARIABLES and ENVIRONMENT VARIABLES sections that list the configuration and environment variables that affect the command.
  - The short flag descriptions are grouped into logical groups for easier reading, which is especially useful when a command has many flags. All examples have brief explanations.
  - The long flag descriptions are displayed at the end of the `--help` output. In `sfdx`, the long flag descriptions don't display in the `--help` output at all. They're documented only in the Salesforce CLI Command Reference.
- It's easy to find the command you want, even if you can't remember its exact syntax. Simply type the command fragments you remember, in any order, and `sf` either displays a list of possible commands or automatically runs the command if there's only one choice. For example, if you type `sf list`, you see this handy little dialogue where you can choose the command you want:

```
sf list
? Which of these commands do you mean (Use arrow keys)
> config list
> env list
> env logdrain list
> env var list
```

- The command output has been standardized and improved for readability.

Look Out: Differences Between `sf` and `sfdx`

- The commands to deploy and retrieve org metadata (`sf deploy|retrieve metadata`) don't yet include source-tracking functionality. If you want to use source-tracking in scratch orgs or sandboxes, keep using the `sfdx force:source:push|pull` or `force:source:deploy|retrieve` commands for now.
- `sf` uses spaces between topics, commands, and subcommands rather than colons. For example, the `sf` command to get a configuration variable is `sf config get`; in `sfdx`, the command is `sfdx config:get`.
- Where appropriate, we've renamed `sfdx` flags in `sf` to include dashes in its name for better readability. For example, `--apiversion` in `sfdx` is now `--api-version` in `sf`.
- In `sf`, you have two ways to use flags that take multiple values:

  - Specify the flag multiple times, with each flag taking a different single value. For example:

    ```
    sf deploy metadata --metadata ApexClass \
                       --metadata CustomObject \
                       --metadata AnotherCustomObject
                       --metadata "My Apex Class"
    ```

  - Specify the flag one time, but separate all the values with a space:

    ```
    sf deploy metadata --metadata ApexClass CustomObject
    AnotherCustomObject "My Apex Class"
    ```

  In `sfdx` you specify the flag one time and separate the values with commas:

  ```
  sfdx force:source:deploy --metadata
  ApexClass,CustomObject,AnotherCustomObject
  ```

- While equivalent `sf` and `sfdx` commands, such as `sf config set` and `sfdx config:set`, are interoperable and aware of each other, they produce different JSON output.

New and Changed Terminology

A quick word about these `sf` terms:

- **Environments**: A general term for the *thing* you use the CLI to interact with, such as the org to which you deploy metadata. We currently support Salesforce orgs and compute environments to which you deploy Salesforce Functions.

  Commands for managing environments are grouped under the `sf env` topic, such as `sf env list`, which lists all the environments you've created or logged into.

- **Logging in:** In `sf` you log into an environment, which authorizes the CLI to run other commands that interact with that environment. In `sfdx` we use the term authorize.
- **Flags:** In `sf` we consistently use the term flag in the documentation, such as the --json flag of a command. In `sfdx` we use the terms parameter, flag, and option in the documentation, sometimes interchangeably.
- **Configuration variables:** In `sf`, you use configuration variables to configure aspects of the CLI. In `sfdx` we call them configuration values.
- **Target:** In `sf`, we use the term *target* when referring to an environment, such as the `target-org` or `target-dev-hub` configuration variables or `--target-env` flag. We're moving away from the terms `defaultusername` and `defaultdevhubusername`, which we use in `sfdx`.

SEE ALSO:

Salesforce Functions

Salesforce DX Developer Guide

Salesforce CLI Command Reference

How `sf` and `sfdx` Work Together

The `sf` and `sfdx` commands play nicely together.

Environments

Currently, the only environments you can interact with are Salesforce orgs (scratch orgs, Dev Hubs, production orgs, sandboxes, and so on) and compute environments (where you deploy Salesforce Functions). Org authorizations are interoperable between `sf` and `sfdx`.

Specifically, when you create a scratch org, authorize an org, or log out of an org with `sfdx`, the `sf` commands respect these actions. The reverse is also true: if you log in or out of an environment in `sf`, the `sfdx` commands respect it.

For example, let's say you create a scratch org with `sfdx force:org:create`. When you run `sf env list` in the same project, the new scratch org is displayed.

Similarly, if you log into an org with `sf login org`, then run `sfdx force:org:list`, the org you logged into is displayed.

Aliases

Aliases are interoperable between `sf` and `sfdx`.

For example, if you run `sfdx force:org:create --setalias MyScratchOrg`, you can use this alias in `sf`, such as `sf env open --target-env MyScratchOrg`.

Similarly, if you run `sf login org --alias MyDevHub`, you can use this alias in `sfdx`, such as `sfdx force:org:open --targetusername MyDevHub`.

Configuration Variables

Configuration variables are interoperable between `sf` and `sfdx`. But their names are different, as shown in this table, which lists all available configuration variables in both `sf` and `sfdx`. It also lists the equivalent `sf` environment variable; see Environment Variables for the full list.

sf Configuration Variable	Equivalent sfdx Configuration Value	sf Environment Variable
`disable-telemetry`	`disableTelemetry`	SF_DISABLE_TELEMETRY
`org-api-version`	`apiVersion`	SF_ORG_API_VERSION
`org-custom-metadata-templates`	`customOrgMetadataTemplates`	SF_ORG_CUSTOM_METADATA_TEMPLATES
`org-instance-url`	`instanceUrl`	SF_ORG_INSTANCE_URL
`org-max-query-limit`	`maxQueryLimit`	SF_ORG_MAX_QUERY_LIMIT
`org-metadata-rest-deploy`	`restDeploy`	SF_ORG_METADATA_REST_DEPLOY
`target-dev-hub`	`defaultdevhubusername`	SF_TARGET_DEV_HUB
`target-org`	`defaultusername`	SF_TARGET_ORG

Let's run through a few use cases to see how the interoperability works, using `target-org` and `defaultusername` as examples. The same behavior applies to the rest of the configuration variables. Here's the general rule: if you set `target-org` to a value, then `defaultusername` is also automatically set to the same value. And vice versa.

Action: Use this `sfdx` command to create a scratch org:

```
sfdx force:org:create --setdefaultusername -f config/project-scratch-def.json --setalias
 my-scratch-org
```

Then run `sfdx config:list` and `sf config list`.

Result: Both `defaultusername` and `target-org` are set to my-scratch-org.

Action: Use this `sf` command to set the target-org configuration variable to an org with alias `uat-testing-org`:

```
sf config set target-org=uat-testing-org
```

Then run `sfdx config:list` and `sf config list`.

Result: Both `defaultusername` and `target-org` are set to `uat-testing-org`.

Action: Use this `sfdx` command to set the `defaultusername` configuration variable to an org with alias `production-org`:

```
sfdx config:set defaultusername=production-org
```

Then run `sfdx config:list` and `sf config list`.

Result: Both `defaultusername` and `target-org` are set to `production-org`.

Action: Use this `sf` command to set `defaultusername`:

```
sf config set defaultusername=other-org
```

Result: Error, because you can't use `sf` commands to directly set `defaultusername`.

Action: Use this `sfdx` command to set target-org:

```
sfdx config:set target-org=other-org
```

Result: Error, because you can't use `sfdx` commands to directly set `target-org`.

Install an `sfdx` Plug-In into `sf`

Because both `sf` and `sfdx` are built on the open-source CLI framework (oclif), you can successfully install and use plug-ins originally built for `sfdx` in `sf`. However, you see these discrepancies in the `--help` and command output:

- The `--help` content likely assumes you're running the commands in `sfdx`. So the command examples show using `sfdx` instead of `sf`, use colon separators instead of spaces, and so on.
- Similarly, the command output reflects the `sfdx` style. For example, if you install the `@oclif/plugin-commands` plug-in and then run `sf commands`, all the commands are correctly listed. However, the commands are displayed with colon separators and their description is the first line of the long command description rather than the short summary.

Install and Update `sf`

Because `sf` is bundled with `sfdx`, installing `sf` is easy.

Existing `sfdx` Users

If you've already installed Salesforce CLI and are currently using `sfdx` commands, you probably already have the `sf` executable. Update to the latest version of `sfdx` then start using `sf`.

```
sfdx update
sf help
```

New Salesforce CLI Users

The operating system-specific Salesforce CLI installers, such as `*.pkg` on macOS, include both the `sfdx` and `sf` executables. If you haven't previously installed Salesforce CLI, install it now to get both executables. See Install Salesforce CLI.

After you install, update to the latest version of `sfdx` then start using `sf`.

```
sfdx update
sf help
```

Release Candidate

Similar to the `sfdx` executable, we publish a release candidate of the `sf` executable each week. The release candidate contains changes that we plan to include in the final weekly version.

To install the `sf` release candidate, update the `sfdx` executable to its release candidate:

```
sfdx update stable-rc
```

To return to the stable version for both executables:

```
sfdx update stable
```

Update to a Different Version

It's easy to update `sf` to any available version by specifying the `--version` flag of the `update` command.

```
sf update --version 1.14.0
```

To get a list of all available versions, use the `--available` flag.

```
sf update --available
```

The output also includes the location of the version, either local—because you've previously installed it—or remote. The update command downloads the remote version if necessary. Use the `--interactive` flag to choose a version from a list using your cursor.

Note: These update options are available only if you installed `sf` using its specific installer. The options aren't available if you installed `sf` with the `sfdx` installer or used npm.

Troubleshooting

If you run into issues installing `sf` as part of an `sfdx` installation, try uninstalling `sfdx` and then installing `sf` on its own using one of these methods:

- npm: Run `npm install @salesforce/cli --global`
- macOS installer: Download and install sf.pkg.

- Windows installer: Download and install sf-x86.exe.

SEE ALSO:

Uninstall Salesforce CLI or Plug-ins

Run Through the Dreamhouse Example

The quickest way to get going with `sf` is to clone the `dreamhouse-lwc` GitHub repo. Use its configuration files and Salesforce application to try out some of the new `sf` commands and see how they work with `sfdx`. In this example, we use `sf` commands when possible, but if `sf` doesn't provide certain functionality, we use the appropriate `sfdx` command instead.

1. Open a terminal or command prompt window, and clone the dreamhouse-lwc GitHub sample repo using HTTPS or SSH.

```
git clone https://github.com/trailheadapps/dreamhouse-lwc.git
 --or--
 git clone git@github.com:trailheadapps/dreamhouse-lwc.git
```

2. Change to the `dreamhouse-lwc` project directory.

```
cd dreamhouse-lwc
```

3. Log in to your Dev Hub org to authorize it to create scratch orgs. Set it as your default Dev Hub and assign it an alias.

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

Enter your Dev Hub org credentials in the browser that opens. After you log in successfully, you can close the browser.

4. Create a scratch org using the `config/project-scratch-def.json` file, set it as your default, and assign it an alias.

```
sf env create scratch --definition-file config/project-scratch-def.json --set-default
--alias my-scratch-org
```

The command uses the default Dev Hub you set with the `sf login` command in a previous step.

5. View the environments (in this case, orgs) that you've either created or logged into.

```
sf env list
```

The first table displays the Dev Hub you logged into and the second table displays the scratch org you created. The right-most Config column in both tables indicates the default scratch org and Dev Hub org with the target-org and target-dev-hub configuration variables, respectively. The left-most Aliases column displays the aliases you assigned each org. Here's some sample output.

```
Salesforce Orgs
=================================================================================================
| Aliases Username                Org ID   Instance Url                        Auth Method
 Config
| ------- ----------------------  ------   --------------------------------  -----------
 -------------
|         user@unicorn-qv0giu.com 00D5e123 https://unicorn.my.salesforce.com web
| DevHub  user@moose-qycot1.com   00D3h456 https://moose.my.salesforce.com   web
   target-dev-hub

Scratch Orgs
=================================================================================================
```

```
| Aliases        Username             Org ID    Instance Url                       Auth
 Method Config
  -------------  -------------------- --------- ---------------------------------
----------  ----------
| my-scratch-org test-0xo@example.com 00D1D0789 https://ability.my.salesforce.com web
        target-org
```

6. Deploy the Dreamforce app, whose source is in the `force-app` directory, to the scratch org.

```
sf deploy metadata --source-dir force-app
```

7. Assign the `dreamhouse` permission set to the default user.

```
sfdx force:user:permset:assign -n dreamhouse
```

8. Import sample data into the scratch org.

```
sfdx force:data:tree:import -p data/sample-data-plan.json
```

9. Run Apex tests.

```
sfdx force:apex:test:run --resultformat human
```

10. Open the scratch org and view the pushed metadata under Most Recently Used.

```
sf env open --target-env my-scratch-org
```

11. Optionally, in the App Launcher, find and open Dreamhouse, just to make sure it deployed correctly.
12. In the Salesforce UI, make a small cosmetic change to an item. For example, update a comment in the `GeocodingServiceTest` Apex class. This step is to later show how you retrieve metadata from an org.
13. Back in the terminal or command window, retrieve the small change you made in the org to your project.

```
sf retrieve metadata --source-dir force-app
```

Confirm that the small change you made in your org has been retrieved to your project by viewing the file `force-app/main/default/classes/GeocodingServiceTest.cls`.

List of `sf` Commands with `sfdx` Equivalents

This table shows the mapping between the currently available `sf` commands and their closest `sfdx` equivalent.

Note: The table doesn't include Salesforce Functions commands, such as `sf run function start`, which are new in `sf` and don't exist in `sfdx`.

`sf` **command**	**Similar** `sfdx` **command**
`sf autocomplete`	`sfdx autocomplete`
`sf config get`	`sfdx config:get`
`sf config list`	`sfdx config:list`
`sf config set`	`sfdx config:set`

`sf` **command**	**Similar** `sfdx` **command**
`sf config unset`	`sfdx config:unset`
`sf deploy` (interactive)	No equivalent.
`sf deploy metadata`	`sfdx force:source:deploy`
`sf deploy metadata cancel`	`sfdx force:source:deploy:cancel`
`sf deploy metadata quick`	`sfdx force:source:deploy --validateddeployrequestid`
`sf deploy metadata report`	`sfdx force:source:deploy:report --wait 0`
`sf deploy metadata resume`	`sfdx force:source:deploy:report`
`sf deploy metadata validate`	`sfdx force:source:deploy --checkonly --testlevel >NoTestRun`
`sf env create sandbox`	`sfdx force:org:create --type sandbox`
`sf env create scratch`	`sfdx force:org:create --type scratch`
`sf env delete sandbox`	`sfdx force:org:delete`
`sf env delete scratch`	`sfdx force:org:delete`
`sf env display`	`sfdx force:org:display`
`sf env list`	`sfdx force:org:list`, `sfdx auth:list`
`sf env open`	`sfdx force:org:open`
`sf env resume sandbox`	`sfdx force:org:status`
`sf env resume scratch`	No equivalent.
`sf generate project`	`sfdx force:project:create`
`sf help`	`sfdx help`
`sf info releasenotes display`	`sfdx info:releasenotes:display`
`sf login` (interactive)	No equivalent.
`sf login org`	`sfdx auth:web:login`
`sf login org jwt`	`sfdx auth:jwt:grant`
`sf logout` (interactive)	No equivalent.
`sf logout org`	`sfdx auth:logout`
`sf retrieve metadata`	`sfdx force:source:retrieve`
`sf version`	`sfdx version`

Environment Variables

Set environment variables to configure aspects of the `sf` executable of Salesforce CLI.

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

```
SF_ORG_API_VERSION=54.0 sf retrieve metadata --metadata ApexClass
```

SF_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, `https://MyDomainName.my.salesforce.com` or `https://login.salesforce.com` for a production org, and `https://MyDomainName--SandboxName.sandbox.my.salesforce.com`or `https://test.salesforce.com` for a sandbox.

Example:

```
SF_AUDIENCE_URL=https://MyDomainName.my.salesforce.com
```

SF_CONTENT_TYPE

When set to `JSON`, specifies that all CLI commands output results in JSON format. If you set the environment variable to any other value, or unset it, the CLI commands output their results as specified by the flags.

Example:

```
SF_CONTENT_TYPE=JSON
```

SF_DISABLE_AUTOUPDATE or SF_AUTOUPDATE_DISABLE (either var works)

Set to `true` to disable the auto-update feature of Salesforce CLI. By default, the CLI periodically checks for and installs updates.

SF_DISABLE_SOURCE_MEMBER_POLLING

Set to `true` to disable polling of your org's SourceMember object when you run the `sf deploy|retrieve metadata` commands.

The commands poll the SourceMember object to track what's changed between your local source and the org after the deploy or retrieve 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 `sf deploy|retrieve metadata` commands are unreliable, and it's up to you to synchronize your source.

SF_DISABLE_TELEMETRY

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

SF_DNS_TIMEOUT

Specifies the number of seconds that the `env create` commands wait for a response when checking whether an org is connected. If the commands don't receive a response in that time, they time out. Default value is 3.

SF_DOMAIN_RETRY

Specifies the time, in seconds, that Salesforce 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.

SF_LOG_LEVEL

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

Example:

```
SF_LOG_LEVEL=debug
```

SF_LOG_ROTATION_PERIOD

Time period after which Salesforce CLI rotates the log file. Rotating the log file means making a backup copy of the file and then clearing out the current log file to start afresh. For example, if set to `1d`, Salesforce CLI rotates the log file daily at midnight. If set to `2w`, the file is rotated every 2 weeks. See the `period` entry in this table for other time period options. Default value is `1d`.

Example:

```
SF_LOG_ROTATION_PERIOD=2w
```

SF_LOG_ROTATION_COUNT

Number of backup files to keep when rotating the log file. Default value is `2`. See SF_LOG_ROTATION_PERIOD for more information

Example:

```
SF_LOG_ROTATION_COUNT=10
```

SF_MDAPI_TEMP_DIR

Places the files (in metadata format) in the specified directory when you run some CLI commands, such as `sf metadata retrieve`. 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.

```
SF_MDAPI_TEMP_DIR=/users/myName/myDXProject/metadata
```

SF_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.

Example:

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

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

SF_ORG_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 Dev Hub.

SF_ORG_CUSTOM_METADATA_TEMPLATES

Specifies either a local directory or a cloned GitHub repository that contains the default custom code templates used by the `sf generate project` command. The GitHub URL points to either the root directory that contains your templates or to a subdirectory on a branch in the repo that contains your templates.

Example:

```
SF_ORG_CUSTOM_METADATA_TEMPLATES=https://github.com/mygithubacct/salesforcedx-templates
```

SF_ORG_INSTANCE_URL

The URL of the Salesforce instance that is hosting your org. Default value is `https://login.salesforce.com`. We recommend that you set this value to the My Domain login URL for your org. You can find the My Domain login URL on the My Domain page in Setup.

SF_ORG_MAX_QUERY_LIMIT

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

Example:

```
SF_ORG_MAX_QUERY_LIMIT=200000
```

SF_ORG_METADATA_REST_DEPLOY

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

SF_SOURCE_MEMBER_POLLING_TIMEOUT

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

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

```
SF_SOURCE_MEMBER_POLLING_TIMEOUT=300
```

SF_TARGET_DEV_HUB

Specifies the username or alias of your default Dev Hub org so you don't have to use the `--target-dev-hub` CLI flag. Overrides the value of the `target-dev-hub` configuration variable.

Example of setting it to an alias:

```
SF_TARGET_DEV_HUB=myDevHub
```

Example of setting it to a username:

```
SF_TARGET_DEV_HUB=mydevhuborg@example.com
```

SF_TARGET_ORG

Specifies the username or alias of your default org so you don't have to use the `--target-org` flag. Overrides the value of the `target-org` configuration variable.

Example of setting it to an alias:

```
SF_TARGET_ORG=myscratchorg
```

Example of setting it to a username:

```
SF_TARGET_ORG=test-xhquykly9fhl@example.com
```

SF_USE_PROGRESS_BAR

Set to `false` to disable the progress bar for the `sf deploy metadata` command.

Example:

```
SF_USE_PROGRESS_BAR=false
```