# Installation

Installing SDKMAN! on UNIX-like platforms is as easy as ever. SDKMAN! installs smoothly on Mac OSX, Linux, Cygwin, Solaris and FreeBSD. We also support Bash and ZSH shells.  
Simply open a new terminal and enter:

$ curl -s "https://get.sdkman.io" | bash

Follow the instructions on-screen to complete installation.  
Next, open a new terminal **or** enter:

$ source "$HOME/.sdkman/bin/sdkman-init.sh"

Lastly, run the following code snippet to ensure that installation succeeded:

$ sdk version

If all went well, the version should be displayed. Something like:

sdkman 5.0.0+51

### Beta Channel

For the more adventurous among us, we have a beta channel. All new CLI features will first be rolled out to this group of users for trial purposes. Beta versions can be considered stable for the most part, but might occasionally break. To join the beta program, simply update the the ~/.sdkman/etc/config file as follows:

sdkman\_beta\_channel=true

Next, open a new terminal and perform a forced update with:

$ sdk selfupdate force

To leave the beta channel, simply set the above config back to false and follow the same procedure.

### Uninstallation

In the unlikely event that you would like to uninstall SDKMAN!, we don't have an automated way of doing this yet. If you really do want to remove it from your system, it is very easy to do so.The following will guide you through backing up, then removing the entire installation from your system.

tar zcvf ~/sdkman-backup\_$(date +%F-%kh%M).tar.gz -C ~/ .sdkman

$ rm -rf ~/.sdkman

The last step involves editing and removing the initialisation snippet from your .bashrc, .bash\_profile and/or .profile files. If you use ZSH, remove it from the .zshrc file. The snippet of code to be removed looks something like this:

#THIS MUST BE AT THE END OF THE FILE FOR SDKMAN TO WORK!!!

[[ -s "/home/dudette/.sdkman/bin/sdkman-init.sh" ]] && source "/home/dudette/.sdkman/bin/sdkman-init.sh"

Once removed, you have successfully uninstalled SDKMAN! from your machine.

### Installing to a Custom Location

It is possible to install SDKMAN! to a custom location other than $HOME/.sdkman. This can be achieved by exporting your custom location as SDKMAN\_DIR prior to installing.  
Simply open a new terminal and enter:

$ export SDKMAN\_DIR="/usr/local/sdkman" && curl -s "https://get.sdkman.io" | bash

For this to work it is vital that your user has full access rights to this folder.It is also important that the folder does not exist as SDKMAN! will attempt to create it.

That's all there is to it! Next we will look at [Usage](http://sdkman.io/usage.html).

**Usage**

**Installing an SDK**

**Latest Stable**

Install the **latest stable** version of your SDK of choice (say, Java JDK) by running the following command:

$ sdk install java

You will see something like the following output:

Downloading: java 8u111

In progress...

######################################################################## 100.0%

Installing: java 8u111

Done installing!

Now you will be prompted if you want this version to be set as **default.**

Do you want java 8u111 to be set as default? (Y/n):

Answering *yes* (or *hitting enter*) will ensure that all subsequent shells opened will have this version of the SDK in use by default.

Setting java 8u111 as default.

**Specific Version**

Need a **specific** version of an SDK? Simply qualify the version you require:

$ sdk install scala 2.12.1

All subsequent steps same as above.

**Install Local Version(s)**

Need a snapshot? Already have a local installation? Setup a local version:

$ sdk install groovy 3.0.0-SNAPSHOT /path/to/groovy-3.0.0-SNAPSHOT

**Remove Version**

Remove an installed version.

$ sdk uninstall scala 2.11.6

**List Candidates**

To get a listing of available Candidates:

$ sdk list

This will render a searchable alphabetic list with name, current stable default version, website URL, description and easy install command for each Candidate. The output is piped to less so standard keyboard shortcuts may be used with q to exit.

================================================================================

Available Candidates

================================================================================

q-quit /-search down

j-down ?-search up

k-up h-help

--------------------------------------------------------------------------------

Groovy (2.4.5) http://www.groovy-lang.org/

Groovy is a powerful, optionally typed and dynamic language, with static-typing

and static compilation capabilities, for the Java platform aimed at multiplying

developers’ productivity thanks to a concise, familiar and easy to learn syntax.

It integrates smoothly with any Java program, and immediately delivers to your

application powerful features, including scripting capabilities, Domain-Specific

Language authoring, runtime and compile-time meta-programming and functional

programming.

$ sdk install groovy

--------------------------------------------------------------------------------

Scala (2.11.7) http://www.scala-lang.org/

...

**List Versions**

To get a listing of Candidate Versions:

$ sdk list groovy

This will result in a list view showing the available, local, installed and current versions of the SDK.

================================================================================

Available Groovy Versions

================================================================================

> \* 2.4.4 2.3.1 2.0.8 1.8.3

2.4.3 2.3.0 2.0.7 1.8.2

2.4.2 2.2.2 2.0.6 1.8.1

2.4.1 2.2.1 2.0.5 1.8.0

2.4.0 2.2.0 2.0.4 1.7.9

2.3.9 2.1.9 2.0.3 1.7.8

2.3.8 2.1.8 2.0.2 1.7.7

2.3.7 2.1.7 2.0.1 1.7.6

2.3.6 2.1.6 2.0.0 1.7.5

2.3.5 2.1.5 1.8.9 1.7.4

2.3.4 2.1.4 1.8.8 1.7.3

2.3.3 2.1.3 1.8.7 1.7.2

2.3.2 2.1.2 1.8.6 1.7.11

2.3.11 2.1.1 1.8.5 1.7.10

2.3.10 2.1.0 1.8.4 1.7.1

================================================================================

+ - local version

\* - installed

> - currently in use

================================================================================

**Use Version**

Choose to use a given version in the current terminal:

$ sdk use scala 2.12.1

It is important to realise that this will switch the candidate version **for the current shell only**. To make this change permanent, use the [default](http://sdkman.io/usage.html#default) command instead.

**Default Version**

Chose to make a given version the default:

$ sdk default scala 2.11.6

This will ensure that all subsequent shells will start with version 2.11.6 in use.

**Current Version(s)**

To see what is currently in use for a Candidate:

$ sdk current java

Using java version 8u111

To see what is currently in use for **all** Candidates:

$ sdk current

Using:

groovy: 2.4.7

java: 8u111

scala: 2.12.1

**Outdated Version(s)**

To see what is currently out of date for a Candidate on your system:

$ sdk outdated springboot

Outdated:

springboot (1.2.4.RELEASE, 1.2.3.RELEASE < 1.2.5.RELEASE)

To see what is outdated for **all** Candidates:

$ sdk outdated

Outdated:

gradle (2.3, 1.11, 2.4, 2.5 < 2.6)

grails (2.5.1 < 3.0.4)

springboot (1.2.4.RELEASE, 1.2.3.RELEASE < 1.2.5.RELEASE)

**SDKMAN! Version**

Display the current version of SDKMAN!:

$ sdk version

**Broadcast Messages**

Get the latest SDK release notifications on the command line:

$ sdk broadcast

==== BROADCAST =================================================================

\* 06/12/16: Scala 2.12.1 released on SDKMAN! #scala

\* 23/11/16: Gradle 3.2.1 released on SDKMAN! #gradle

\* 22/11/16: Ceylon 1.3.1 released on SDKMAN! #ceylonlang

================================================================================

It is also worth mentioning that whenever an SDK version is released on SDKMAN!, a notification will appear when next using the CLI. Every new broadcast is also pushed to Twitter.

**Offline Mode**

Initially called *Aeroplane Mode*, this allows SDKMAN! to function when working offline. It has a parameter that can be passed to *enable* or *disable* the offline mode.

$ sdk offline enable

Forced offline mode enabled.

$ sdk offline disable

Online mode re-enabled!

When operating in **offline** mode, most commands will still work even though they will operate in a scaled down capacity. An example is the list command, which will only display the currently installed and active version(s):

$ sdk list

------------------------------------------------------------

Offline Mode: only showing installed groovy versions

------------------------------------------------------------

> 2.4.4

\* 2.4.3

------------------------------------------------------------

\* - installed

> - currently in use

------------------------------------------------------------

The offline mode will also be disabled/enabled automatically when the internet becomes available/unavailable. Naturally, commands that require internet connectivity will not function but give a warning.

**Self-Update**

Installs a new version of SDKMAN! if available.

$ sdk selfupdate

If no new version is available an appropriate message will be displayed. Re-installation may be forced by passing the force parameter to the command:

$ sdk selfupdate force

Automatic daily checks for new versions of SDKMAN! will also be performed on the behalf of the user.

**Flush**

From time to time it may be necessary to flush SDKMAN!'s local state.The flush command helps with this and allows for the following to be performed:

**Candidates**

$ sdk flush candidates

Clears out the Candidate list. Opening a new terminal will fetch and store the latest list.This is usually required when a new Candidate is made available on SDKMAN!.

**Broadcast**

$ sdk flush broadcast

Clears out the Broadcast cache, downloading the latest available news on next command invocation.

**Archives**

$ sdk flush archives

Cleans the cache containing all downloaded SDK binaries. This can take up a lot of space so is worth clearing out from time to time!

**Temporary Folder**

$ sdk flush temp

Clears out the staging work folder used when installing new versions of candidates and SDKMAN! itself.

**Help**

You can get basic help by running the following command:

$ sdk help

This should yield something like:

Usage: sdk [version]

sdk offline

commands:

install or i [version]

uninstall or rm

list or ls

use or u [version]

default or d [version]

current or c [candidate]

outdated or o [candidate]

version or v

broadcast or b

help or h

offline

selfupdate [force]

flush

candidate : ...

version : where optional, defaults to latest stable if not provided

eg: sdk install groovy

**Configuration**

Although configuration is limited, the list of configurable items will grow as required. Configuration can be found in the ~/.sdkman/etc/config file. Currently the following is configurable:

# make sdkman non-interactive, preferred for CI environments

sdkman\_auto\_answer=true|false

# perform automatic selfupdates

sdkman\_auto\_selfupdate=true|false

# disables SSL certificate verification

# https://github.com/sdkman/sdkman-cli/issues/327

# HERE BE DRAGONS....

sdkman\_insecure\_ssl=true|false

# disable GVM alias, for users of the Go Version Manager

sdkman\_disable\_gvm\_alias=true|false

# configure curl timeouts

sdkman\_curl\_connect\_timeout=5

sdkman\_curl\_max\_time=4

# subscribe to the beta channel

sdkman\_beta\_channel=true

# Vendors

SDKMAN! is unique in that it empowers SDK Vendors to publish their own Candidate releases on our platform. We provide a secure API that can be used to manage all aspects of a release on SDKMAN!. This includes **Releasing** a new Version, setting an existing Version as **Default** (Stable) and **Announcing** the release on the SDKMAN! CLI broadcast and Twitter feed.

### Operations

The API is a simple JSON REST API that allows several operations:

* Release a new Candidate Version
* Make a Version the Default for a given Candidate
* Broadcast a structured release message
* Broadcast a freeform message

### Access

This is a secured API, and requires appropriate credentials to perform the above operations. Access will be granted on a case-by-case basis to Vendors who are interested in making their technology available on SDKMAN!. If you want to publish your releases on SDKMAN!, please contact [Marco Vermeulen](mailto:marco@sdkman.io) to help getting you on board.

### Endpoints

The simplest way to call the API is by using cURL. Of course, any other client can be used to perform the API operations:

#### Release a new Candidate Version

This will perform a Minor Release on SDKMAN!. It will simply add the new Candidate Version, but will not make it the default version for the Candidate.

curl -X POST \

-H "Consumer-Key: CONSUMER\_KEY" \

-H "Consumer-Token: CONSUMER\_TOKEN" \

-H "Content-Type: application/json" \

-H "Accept: application/json" \

-d '{"candidate": "groovy", "version": "2.4.2", "url": "http://dl.bintray.com/groovy/maven/groovy-binary-2.4.2.zip"}' \

https://vendors.sdkman.io/release

#### Set existing Version as Default for Candidate

When calling this endpoint for an existing Candidate Version, it will make it the Default Version for that Candidate. This makes a Minor release a Major release!

curl -X PUT \

-H "Consumer-Key: CONSUMER\_KEY" \

-H "Consumer-Token: CONSUMER\_TOKEN" \

-H "Content-Type: application/json" \

-H "Accept: application/json" \

-d '{"candidate": "groovy", "version": "2.3.8"}' \

https://vendors.sdkman.io/default

#### Broadcast a Structured Message

This will result in a structured message announcement on social media and SDKMAN! CLI. The result will look something like: Grails 3.0.0 has been released on SDKMAN! #grailsfw. This message will be announced to the broadcast channel of SDKMAN! CLI, as well as on the [@sdkman\_](https://www.twitter.com/sdkman_) Twitter feed.

curl -X POST \

-H "Consumer-Key: CONSUMER\_KEY" \

-H "Consumer-Token: CONSUMER\_TOKEN" \

-H "Content-Type: application/json" \

-H "Accept: application/json" \

-d '{"candidate": "grails", "version": "3.0.0", "hashtag": "grailsfw"}' \

https://vendors.sdkman.io/announce/struct

#### Broadcast a Freeform Message

This endpoint is not used as much as the previous, but allows freeform messages to be announced.

curl -X POST \

-H "Consumer-Key: CONSUMER\_KEY" \

-H "Consumer-Token: CONSUMER\_TOKEN" \

-H "Content-Type: application/json" \

-H "Accept: application/json" \

-d '{"text": "SDKMAN! 2.4.0 rolling out. Broadcast and Offline checks optimised."}' \

https://vendors.sdkman.io/announce/freeform

### Gradle SDK Vendor Plugin

If fiddling with cURL (or HttpClient) isn’t your thing, you could consider using our Gradle plugin. The plugin allows the release to be done as a side effect of your CI build! It exposes several useful tasks like:

* sdkReleaseVersion
* sdkDefaultVersion
* sdkAnnounceVersion

It also exposes some convenience tasks that roll the above into single tasks:

* sdkMajorRelease: performs release, default and structured announce
* sdkMinorRelease: performs release and structured announce, no default

For more details of about this plugin, as well as how to configure it, please refer to the [Github Page](https://github.com/sdkman/gradle-sdkvendor-plugin) for the project.