SDKMAN! JVM SDK manager

START HERE

`sdk list java`
Available JDKs

`sdk current`
Versions in use

`sdk install java <id>`
Install a JDK

`sdk env init`
Pin in .sdkmanrc

`sdk use java <id>`
This shell only

`sdk env install`
Set up a project

`sdk default java <id>`
All new shells

`sdk upgrade`
Find outdated SDKs

Candidate java, maven, ... → `install` → **Installed version** candidates/<c>/<v> → `default` → **current link** new shells → `use` / `env` → **This shell** PATH and *_HOME

CONCEPTS

`source ~/.sdkman/bin/sdkman-init.sh`
Load sdk into the current shell

SDKMAN! keeps parallel versions of JVM SDKs, called candidates (java, maven, gradle, kotlin, ...). sdk is a shell function loaded by sdkman-init.sh.

sdk use affects only this shell; sdk default moves the current symlink that new shells pick up. JAVA_HOME follows the active java.

FIND AND INSTALL

`sdk list`
Browse all candidates
Opens in a pager; press q to quit.

`sdk list <sdk>`
List versions of one candidate

`sdk install <sdk>`
Install the default version
Asks whether to make it the default.

`sdk install <sdk> <version>`
Install a specific version

`sdk install <sdk> <version> <path>`
Register an existing local install
Version name must be new, e.g. 11-local.

`sdk uninstall <sdk> <version>`
Remove an installed version
For a local version only the link goes.

`sdk update`
Refresh the candidate list

Markers in sdk list <sdk>

Mark	Meaning
+	Local version
*	Installed
>	In use

SWITCH VERSIONS

`sdk use <sdk> <version>`
Use a version in this shell
Fails if not installed; install it first.

`sdk default <sdk> <version>`
Set default for new shells

SWITCH VERSIONS (cont.)

`sdk default <sdk>`
Reset to the SDKMAN default version

`sdk current`
Show versions in use

`sdk current <sdk>`
Show the version of one candidate

`sdk home <sdk> <version>`
Print the install path
No trailing newline: meant for $(...) in scripts.

`sdk upgrade`
List outdated candidates

`sdk upgrade <sdk>`
Upgrade one candidate

JAVA VENDORS

`sdk list java`
Vendor table with identifiers
Install by the Identifier column, not Version.

`sdk install java <version>-tem`
Install an Eclipse Temurin JDK

Identifier suffixes

Suffix	Distribution
-tem	Eclipse Temurin
-amzn	Amazon Corretto
-zulu	Azul Zulu
-librca	BellSoft Liberica
-graalce	GraalVM CE
-graal	Oracle GraalVM
-oracle	Oracle JDK
-ms	Microsoft OpenJDK

PROJECT ENV (.SDKMANRC)

`sdk env init`
Create .sdkmanrc with current java

`sdk env`
Switch to versions in .sdkmanrc

`sdk env install`
Install missing versions and switch

`sdk env clear`
Return to default versions

PROJECT ENV (.SDKMANRC) (cont.)

.sdkmanrc

```
# candidate=version, one per line
java=21.0.4-tem
maven=3.9.9
gradle=8.10
```

With sdkman_auto_env=true, cd into a folder with .sdkmanrc runs sdk env; leaving restores defaults.

MAINTENANCE

`sdk version`
Print script and native versions

`sdk help [<subcommand>]`
Show help, or help for one command

`sdk selfupdate`
Update SDKMAN! itself

`sdk selfupdate force`
Reinstall SDKMAN! even if current

`sdk flush`
Clear caches and leftover downloads
Use it instead of deleting ~/.sdkman/tmp by hand.

`sdk flush tmp|metadata|version`
Flush a single cache target

Short aliases

Alias	Command
i / rm	install / uninstall
ls / l	list
u / d	use / default
c / ug	current / upgrade
h / e / v	home / env / version

CONFIGURATION

`sdk config`
Edit ~/.sdkman/etc/config
Uses $EDITOR, else vi. Open a new shell afterwards.

Everyday keys (sdkman_ prefix)

Key	Effect
auto_answer	No prompts (CI)
auto_env	Apply .sdkmanrc on cd
selfupdate_feature	Prompt for updates
auto_complete	Shell completion
colour_enable	Coloured output

CONFIGURATION (cont.)

Network and debug keys

Key	Effect
healthcheck_enable	API check at startup
curl_connect_timeout	Connect timeout (s)
curl_max_time	Max transfer time (s)
checksum_enable	Verify downloads
beta_channel	Beta releases
debug_mode	Verbose output
insecure_ssl	Skip TLS checks (avoid)

There is no offline subcommand. Without network, commands that need the API fail with a warning and sdk list shows installed versions only; set healthcheck_enable=false in the config to skip the startup check.

RECIPES

Set up a new machine

```
curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk install java 21.0.4-tem
sdk install maven
sdk install gradle
```

Pin a project

```
sdk use java 21.0.4-tem
sdk env init
echo 'maven=3.9.9' >> .sdkmanrc
git add .sdkmanrc
```

Non-interactive CI

```
sed -i 's/auto_answer=false/auto_answer=true/' \
  ~/.sdkman/etc/config
sdk env install
export JAVA_HOME="$(sdk home java 21.0.4-tem)"
```

Tidy up old versions

```
sdk upgrade
ls ~/.sdkman/candidates/java
sdk uninstall java 17.0.12-tem
sdk flush
```