

Package: sysreqr (via r-universe)

June 11, 2026

Title Preflight Checks for 'R' Package System Requirements

Version 0.1.0

Description Helps users on 'Linux' (and, where applicable, 'macOS') find the system packages they need before installing 'R' packages from source. Queries maintained system requirement sources, reports missing system packages, and generates installation commands, 'Dockerfile' snippets, 'GitHub Actions' steps, administrator request templates, and diagnostic reports from failed installation logs.

License GPL-3

URL <https://github.com/choxos/sysreqR>

BugReports <https://github.com/choxos/sysreqR/issues>

Depends R (>= 4.1)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Ahmad Sofi-Mahmudi [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-6829-0823>>)

Maintainer Ahmad Sofi-Mahmudi <a.sofimahmudi@gmail.com>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-06-11 11:50:02 UTC

RemoteUrl <https://github.com/cran/sysreqr>

RemoteRef HEAD

RemoteSha a48f59ec5a5da40804063d6094965c2d6e4e9048

Contents

admin_request	2
as_data_frame	3
as_install_plan	4
check_error	4
check_library	5
check_packages	6
check_ppm	7
check_project	8
detect_package_manager	9
detect_platform	9
detect_project_packages	10
diagnose_failed_packages	11
diagnose_log	12
dockerfile	13
explain	14
github_actions	15
install_command	16
is_sysreqr_plan	17
ppm_platforms	17
ppm_repo	18
ppm_sysreqs	19
print.sysreqr_plan	20
print.sysreqr_setup_advice	20
resolve_platform	21
setup_advice	22
use_ppm	23
write_dockerfile_snippet	24
write_install_script	25
write_json	25
write_report	26
Index	27

admin_request	<i>Create an administrator request</i>
---------------	--

Description

Produces a plain-text message that a user without root access can send to their system administrator. The message lists missing system packages, which R packages need them, and a suggested install command.

Usage

```
admin_request(x, platform = NULL, ...)
```

Arguments

x A sysreqr_plan or package vector.
 platform Platform specification accepted by [resolve_platform\(\)](#).
 ... Passed to [check_packages\(\)](#) when x is not a plan.

Value

A plain-text administrator request.

See Also

Other commands: [dockerfile\(\)](#), [github_actions\(\)](#), [install_command\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
cat(admin_request(plan))
```

as_data_frame

Coerce a plan to a plain data frame

Description

Strips the sysreqr_plan class and attributes, returning a plain data.frame. Equivalent to `as.data.frame()` on a plan, but provided as a verb form for users who prefer that style.

Usage

```
as_data_frame(x)
```

Arguments

x A sysreqr_plan.

Value

A data frame without the sysreqr_plan class.

See Also

Other plan: [is_sysreqr_plan\(\)](#), [print_sysreqr_plan\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
df <- as_data_frame(plan)
inherits(df, "sysreqr_plan")
inherits(df, "data.frame")
```

as_install_plan	<i>Return backend install plan data</i>
-----------------	---

Description

Converts a plan into a structured list suitable for downstream tooling (other R code, deployment scripts, or CI). The fields are platform, backend, pre_install, install, post_install, and packages.

Usage

```
as_install_plan(x)
```

Arguments

x A sysreqr_plan.

Value

A list with commands and plan data.

See Also

Other output: [write_dockerfile_snippet\(\)](#), [write_install_script\(\)](#), [write_json\(\)](#), [write_report\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
as_install_plan(plan)
```

check_error	<i>Check the most recent install error for likely system requirements</i>
-------------	---

Description

Convenience wrapper around [diagnose_log\(\)](#). When text is NULL (the default), the most recent R error message is read from base: `:geterrmessage()`.

Usage

```
check_error(  
  text = NULL,  
  platform = NULL,  
  backend = c("auto", "bundled", "ppm", "pak"),  
  repo = "cran",  
  check_installed = TRUE  
)
```

Arguments

text	Error text. When NULL, the most recent R error message is used.
platform	Platform specification accepted by resolve_platform() .
backend	One of "auto", "bundled", "ppm", or "pak" for failed package lookup.
repo	Repository name used by the PPM backend.
check_installed	Whether to check installed system packages on the current host when possible.

Value

A `sysreqr_plan`.

See Also

Other diagnose: [diagnose_failed_packages\(\)](#), [diagnose_log\(\)](#)

Examples

```
check_error(
  text = "ERROR: configuration failed for package 'xml2'",
  platform = "ubuntu-22.04",
  backend = "bundled"
)
```

check_library	<i>Check system requirements for installed R packages</i>
---------------	---

Description

Reads the package names from one or more R library paths and resolves their system requirements. Useful for auditing an existing R installation.

Usage

```
check_library(
  packages = NULL,
  library = .libPaths()[1],
  platform = NULL,
  backend = c("bundled", "auto", "ppm", "pak")
)
```

Arguments

packages	Optional installed R package names.
library	Library path or paths.
platform	Platform specification accepted by resolve_platform() .
backend	One of "bundled", "auto", "ppm", or "pak".

Value

A `sysreqr_plan`.

See Also

Other preflight: [check_packages\(\)](#), [check_project\(\)](#), [detect_project_packages\(\)](#)

Examples

```
# Offline, with an explicit package list and the bundled backend:
check_library(
  packages = c("xml2", "curl"),
  platform = "ubuntu-22.04",
  backend = "bundled"
)
```

check_packages

Check system requirements for R packages

Description

Resolves the system packages an R package needs on a given Linux platform and returns them in a structured plan. The "auto" backend prefers the offline bundled database on apt platforms, then Posit Package Manager, then pak.

Usage

```
check_packages(
  packages,
  platform = NULL,
  backend = c("auto", "bundled", "ppm", "pak"),
  repo = "cran",
  dependencies = NA,
  upgrade = TRUE,
  check_installed = TRUE
)
```

Arguments

<code>packages</code>	Package names or package references.
<code>platform</code>	Platform specification accepted by resolve_platform() .
<code>backend</code>	One of "auto", "bundled", "ppm", or "pak".
<code>repo</code>	Repository name used in notes and by the PPM backend.
<code>dependencies</code>	Dependency policy passed to <code>pak::pkg_sysreqs()</code> when the pak backend is used.
<code>upgrade</code>	Upgrade policy passed to <code>pak::pkg_sysreqs()</code> when the pak backend is used.
<code>check_installed</code>	Whether to check installed system packages on the current host when possible.

Value

A `sysreqr_plan`.

See Also

[check_project\(\)](#), [check_library\(\)](#), [diagnose_log\(\)](#).

Other preflight: [check_library\(\)](#), [check_project\(\)](#), [detect_project_packages\(\)](#)

Examples

```
plan <- check_packages(c("xml2", "curl"), platform = "ubuntu-22.04")
plan

install_command(plan)
```

check_ppm

Check Posit Package Manager support

Description

Reports whether a platform is currently served by Posit Package Manager and whether it has system requirement metadata and binary packages.

Usage

```
check_ppm(platform = NULL, base_url = ppm_default_base_url())
```

Arguments

`platform` Platform specification accepted by [resolve_platform\(\)](#).
`base_url` Posit Package Manager base URL.

Value

A list with the matched platform and Package Manager status.

See Also

Other ppm: [ppm_platforms\(\)](#), [ppm_repo\(\)](#), [ppm_sysreqs\(\)](#), [use_ppm\(\)](#)

Examples

```
check_ppm("ubuntu-22.04")
check_ppm("fedora-40")
```

check_project	<i>Check system requirements for a project</i>
---------------	--

Description

Convenience wrapper around [detect_project_packages\(\)](#) and [check_packages\(\)](#).

Usage

```
check_project(
  path = ".",
  include_suggests = FALSE,
  platform = NULL,
  backend = c("auto", "bundled", "ppm", "pak"),
  ...
)
```

Arguments

path	Project path.
include_suggests	Whether to include Suggests from DESCRIPTION.
platform	Platform specification accepted by resolve_platform() .
backend	One of "auto", "ppm", or "pak".
...	Passed to check_packages() .

Value

A `sysreqr_plan`.

See Also

Other preflight: [check_library\(\)](#), [check_packages\(\)](#), [detect_project_packages\(\)](#)

Examples

```
project <- file.path(tempdir(), "demo-project")
dir.create(project, showWarnings = FALSE)
writeLines(
  c("Package: demo", "Imports: xml2"),
  file.path(project, "DESCRIPTION")
)
check_project(project, platform = "ubuntu-22.04", backend = "bundled")
```

`detect_package_manager`*Detect the platform package manager*

Description

Returns the name of the operating system package manager for a platform: "apt", "dnf", "yum", "zypper", "apk", or "brew".

Usage

```
detect_package_manager(platform = NULL)
```

Arguments

`platform` A platform specification accepted by [resolve_platform\(\)](#).

Value

A package manager name such as "apt" or "dnf".

See Also

Other platform: [detect_platform\(\)](#), [resolve_platform\(\)](#)

Examples

```
detect_package_manager("ubuntu-22.04")
detect_package_manager("fedora-40")
detect_package_manager("opensuse156")
```

`detect_platform`*Detect the current platform*

Description

Detects the host operating system and, on Linux, parses `/etc/os-release`. On macOS, queries `sw_vers`. Returns a structured platform object used by the rest of the package. When `os_release` is supplied and exists, the file is parsed even on macOS or Windows; this makes fixture-driven testing practical.

Usage

```
detect_platform(os_release = "/etc/os-release")
```

Arguments

`os_release` Path to an os-release file. Defaults to the system file `/etc/os-release`. Mainly useful for tests and reproducible builds.

Value

An object of class `sysreqr_platform` (a list).

See Also

Other platform: [detect_package_manager\(\)](#), [resolve_platform\(\)](#)

Examples

```
platform <- detect_platform()
platform$distro

# Reproducible reads from a fixture file:
fixture <- system.file(
  "extdata", "os-release-fedora-40",
  package = "sysreqr",
  mustWork = FALSE
)
if (nzchar(fixture)) detect_platform(os_release = fixture)
```

`detect_project_packages`

Detect R packages used by a project

Description

Inspects a project directory for the R packages it uses. The detection priority is:

Usage

```
detect_project_packages(path = ".", include_suggests = FALSE)
```

Arguments

`path` Project path.

`include_suggests` Whether to include Suggests from DESCRIPTION.

Details

1. `renv.lock` (the Packages map),
2. DESCRIPTION (Depends, Imports, LinkingTo, and optionally Suggests),
3. `.R`, `.Rmd`, `.qmd`, and `NAMESPACE` files (looking for `library()`, `require()`, `requireNamespace()`, and `pkg:::fun` references; line comments are ignored).

Value

A character vector of package names.

See Also

Other preflight: [check_library\(\)](#), [check_packages\(\)](#), [check_project\(\)](#)

Examples

```
project <- file.path(tempdir(), "demo-project")
dir.create(project, showWarnings = FALSE)
writeLines(
  c("Package: demo", "Imports: xml2, curl"),
  file.path(project, "DESCRIPTION")
)
detect_project_packages(project)
```

diagnose_failed_packages

Diagnose failed R packages

Description

Resolves the system requirements for a set of R packages that the user already knows failed to install. Useful when an install was attempted outside R, or when the log was lost.

Usage

```
diagnose_failed_packages(
  packages,
  platform = NULL,
  backend = c("auto", "bundled", "ppm", "pak"),
  repo = "cran",
  check_installed = TRUE
)
```

Arguments

packages	R package names inferred from failed installation output.
platform	Platform specification accepted by resolve_platform() .
backend	One of "auto", "bundled", "ppm", or "pak" for failed package lookup.
repo	Repository name used by the PPM backend.
check_installed	Whether to check installed system packages on the current host when possible.

Value

A `sysreqr_plan`.

See Also

Other diagnose: [check_error\(\)](#), [diagnose_log\(\)](#)

Examples

```
diagnose_failed_packages(  
  c("xml2", "curl"),  
  platform = "ubuntu-22.04",  
  backend = "bundled"  
)
```

diagnose_log

Diagnose an R package installation log

Description

Scans an installation log for common compiler and linker errors and for R-level "configuration failed" and "non-zero exit status" patterns, then resolves those failed packages back to their system requirements.

Usage

```
diagnose_log(  
  path = NULL,  
  text = NULL,  
  platform = NULL,  
  backend = c("auto", "bundled", "ppm", "pak"),  
  repo = "cran",  
  check_installed = TRUE  
)
```

```
diagnose_install_log(  
  path = NULL,  
  text = NULL,  
  platform = NULL,  
  backend = c("auto", "bundled", "ppm", "pak"),  
  repo = "cran",  
  check_installed = TRUE  
)
```

Arguments

path	Path to an installation log.
text	Log text. Used when path is NULL.
platform	Platform specification accepted by resolve_platform() .
backend	One of "auto", "bundled", "ppm", or "pak" for failed package lookup.
repo	Repository name used by the PPM backend.
check_installed	Whether to check installed system packages on the current host when possible.

Details

The function combines two paths:

1. Direct log-pattern matching against a curated list of header and linker error messages.
2. Failed package extraction plus a back-end lookup of those package names.

Confidence levels are "high" for direct pattern matches and pak/PPM lookups, and "medium" for bundled fallback data and inferred package requirements.

Value

A `sysreqr_plan` with likely system package fixes.

See Also

Other diagnose: [check_error\(\)](#), [diagnose_failed_packages\(\)](#)

Examples

```
log <- paste(
  "fatal error: libxml/parser.h: No such file or directory",
  "ERROR: configuration failed for package 'xml2'",
  sep = "\n"
)
plan <- diagnose_log(text = log, platform = "ubuntu-22.04")
plan
```

 dockerfile

Generate Dockerfile lines

Description

Produces a Dockerfile RUN snippet that installs the system packages a plan needs. On apt platforms the output uses the standard `apt-get update && apt-get install -y --no-install-recommends ... && rm -rf /var/lib` pattern.

Usage

```
dockerfile(x, platform = NULL, missing_only = TRUE, ...)
```

Arguments

x	A <code>sysreqr_plan</code> or package vector.
platform	Platform specification accepted by <code>resolve_platform()</code> .
missing_only	Whether to include only packages not known to be installed.
...	Passed to <code>check_packages()</code> when x is not a plan.

Value

A single Dockerfile snippet.

See Also

Other commands: `admin_request()`, `github_actions()`, `install_command()`

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
cat(dockerfile(plan))
```

explain

Explain system requirements

Description

Prints a short, friendly explanation for each system package an R package needs and the command to install it. Useful for teaching and for emails to less-experienced collaborators.

Usage

```
explain(x, platform = NULL, ...)
```

Arguments

x	A package name, package vector, or <code>sysreqr_plan</code> .
platform	Platform specification accepted by <code>resolve_platform()</code> .
...	Passed to <code>check_packages()</code> when x is not a plan.

Value

A character vector of explanation lines, invisibly.

See Also

Other setup: [print.sysreqr_setup_advice\(\)](#), [setup_advice\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
explain(plan)
```

github_actions

Generate a GitHub Actions snippet

Description

Produces a GitHub Actions YAML step that installs the system packages a plan needs. `gha()` is a short alias.

Usage

```
github_actions(x, platform = NULL, missing_only = TRUE, ...)
```

```
gha(x, platform = NULL, missing_only = TRUE, ...)
```

Arguments

<code>x</code>	A <code>sysreqr_plan</code> or package vector.
<code>platform</code>	Platform specification accepted by resolve_platform() .
<code>missing_only</code>	Whether to include only packages not known to be installed.
<code>...</code>	Passed to check_packages() when <code>x</code> is not a plan.

Value

A YAML snippet.

See Also

Other commands: [admin_request\(\)](#), [dockerfile\(\)](#), [install_command\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
cat(github_actions(plan))
identical(gha(plan), github_actions(plan))
```

install_command	<i>Generate an installation command</i>
-----------------	---

Description

Translates a plan (or a package vector that can be resolved to a plan) into shell commands appropriate for the platform's package manager.

Usage

```
install_command(  
  x,  
  platform = NULL,  
  sudo = TRUE,  
  update = TRUE,  
  missing_only = TRUE,  
  ...  
)
```

Arguments

x	A <code>sysreqr_plan</code> or package vector.
platform	Platform specification accepted by <code>resolve_platform()</code> .
sudo	Whether to prefix commands with <code>sudo</code> .
update	Whether to include the package manager update command when appropriate.
missing_only	Whether to include only packages not known to be installed.
...	Passed to <code>check_packages()</code> when x is not a plan.

Value

A character vector of shell commands.

See Also

Other commands: `admin_request()`, `dockerfile()`, `github_actions()`

Examples

```
plan <- check_packages(c("xml2", "curl"), platform = "ubuntu-22.04")  
install_command(plan)  
install_command(plan, sudo = FALSE, update = FALSE)
```

is_sysreqr_plan	<i>Test whether an object is a sysreqr plan</i>
-----------------	---

Description

Test whether an object is a sysreqr plan

Usage

```
is_sysreqr_plan(x)
```

Arguments

x An object.

Value

TRUE if x inherits from "sysreqr_plan".

See Also

Other plan: [as_data_frame\(\)](#), [print.sysreqr_plan\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
is_sysreqr_plan(plan)
is_sysreqr_plan(data.frame(x = 1))
```

ppm_platforms	<i>List Posit Package Manager platforms</i>
---------------	---

Description

Queries the Posit Package Manager `/__api__/status` endpoint and returns a data frame of supported distributions.

Usage

```
ppm_platforms(base_url = ppm_default_base_url())
```

Arguments

base_url Posit Package Manager base URL.

Value

A data frame of platform records reported by Package Manager.

See Also

Other ppm: [check_ppm\(\)](#), [ppm_repo\(\)](#), [ppm_sysreqs\(\)](#), [use_ppm\(\)](#)

Examples

```
ppm_platforms()
```

ppm_repo

Build a Posit Package Manager repository URL

Description

Constructs a Linux binary repository URL for the given platform, CRAN repository alias, and snapshot.

Usage

```
ppm_repo(
  platform = NULL,
  repo = "cran",
  snapshot = "latest",
  base_url = ppm_default_base_url()
)
```

Arguments

platform	Platform specification accepted by resolve_platform() .
repo	Repository name.
snapshot	Snapshot name or date.
base_url	Posit Package Manager base URL.

Value

A repository URL.

See Also

Other ppm: [check_ppm\(\)](#), [ppm_platforms\(\)](#), [ppm_sysreqs\(\)](#), [use_ppm\(\)](#)

Examples

```
ppm_repo(platform = "ubuntu-22.04")
ppm_repo(platform = "ubuntu-26.04", snapshot = "2026-04-01")
```

`ppm_sysreqs`*Query Package Manager system requirements*

Description

Queries the Posit Package Manager `/sysreqs` endpoint for the given packages and platform, and normalizes the response into a `sysreqr_plan`. If the API call fails, the function falls back to the bundled database and records the failure in the `"fallback_error"` attribute of the returned plan.

Usage

```
ppm_sysreqs(  
  packages = NULL,  
  all = FALSE,  
  platform = NULL,  
  repo = "cran",  
  base_url = ppm_default_base_url(),  
  check_installed = TRUE  
)
```

Arguments

<code>packages</code>	Package names. Required when <code>all = FALSE</code> .
<code>all</code>	Whether to return system requirements for the whole repository.
<code>platform</code>	Platform specification accepted by resolve_platform() .
<code>repo</code>	Repository name.
<code>base_url</code>	Posit Package Manager base URL.
<code>check_installed</code>	Whether to check installed system packages on the current host when possible.

Value

A `sysreqr_plan`.

See Also

Other ppm: [check_ppm\(\)](#), [ppm_platforms\(\)](#), [ppm_repo\(\)](#), [use_ppm\(\)](#)

Examples

```
ppm_sysreqs(c("xml2", "curl"), platform = "ubuntu-22.04")
```

print.sysreqr_plan *Print a sysreqr plan*

Description

Print a sysreqr plan

Usage

```
## S3 method for class 'sysreqr_plan'  
print(x, ...)
```

Arguments

x	A sysreqr_plan.
...	Unused.

Value

x, invisibly.

See Also

Other plan: [as_data_frame\(\)](#), [is_sysreqr_plan\(\)](#)

print.sysreqr_setup_advice
 Print setup advice

Description

Print setup advice

Usage

```
## S3 method for class 'sysreqr_setup_advice'  
print(x, ...)
```

Arguments

x	A sysreqr_setup_advice object.
...	Unused.

Value

x, invisibly.

See Also

Other setup: [explain\(\)](#), [setup_advice\(\)](#)

resolve_platform	<i>Resolve a platform specification</i>
------------------	---

Description

Accepts a NULL, a `sysreqr_platform` object, or a short string and returns a fully normalized platform object. Strings can be the `<distro>-<version>` shorthand ("ubuntu-22.04", "debian-12") or a codename alias ("jammy", "noble", "resolute", "bookworm", "trixie").

Usage

```
resolve_platform(platform = NULL)
```

Arguments

platform	NULL, a platform list returned by detect_platform() , or a character string such as "ubuntu-22.04", "debian-12", "jammy", or "bookworm".
----------	--

Value

A platform list with class "sysreqr_platform".

See Also

Other platform: [detect_package_manager\(\)](#), [detect_platform\(\)](#)

Examples

```
resolve_platform("ubuntu-22.04")
resolve_platform("resolute")
resolve_platform("bookworm")
```

`setup_advice`*Get beginner setup advice for R package installation on Linux*

Description

Produces a practical checklist for users who are new to R on GNU/Linux. It recommends binary package repositories where available, explains when R Project operating system repositories are relevant, lists source build tools, and can add package-specific system requirements. It never runs `sudo`, edits the system R configuration, or changes operating system repositories; the user remains in control.

Usage

```
setup_advice(  
  packages = NULL,  
  platform = NULL,  
  backend = c("auto", "bundled", "ppm", "pak"),  
  repo = "cran",  
  check_installed = TRUE,  
  include_r_project_repo = TRUE,  
  script = NULL  
)
```

Arguments

<code>packages</code>	Optional R package names to check.
<code>platform</code>	Platform specification accepted by resolve_platform() .
<code>backend</code>	One of "auto", "bundled", "ppm", or "pak".
<code>repo</code>	Repository name used for Posit Package Manager and <code>sysreq</code> lookup.
<code>check_installed</code>	Whether to check installed system packages on the current host when possible.
<code>include_r_project_repo</code>	Whether to include optional R Project operating system repository commands for supported Linux distributions.
<code>script</code>	Optional path. When supplied, an executable shell script is written with the safe setup commands.

Value

A `sysreqr_setup_advice` object.

See Also

Other setup: [explain\(\)](#), [print.sysreqr_setup_advice\(\)](#)

Examples

```

advice <- setup_advice(
  "xml2",
  platform = "ubuntu-22.04",
  backend = "bundled"
)
print(advice)

# Write a reviewable shell script:
setup_advice(
  c("xml2", "curl"),
  platform = "ubuntu-22.04",
  backend = "bundled",
  script = tempfile(fileext = ".sh")
)

```

use_ppm

Configure Package Manager repository options

Description

Emits or installs the R code lines that point options(repos) at a Posit Package Manager binary repository. With `dry_run = TRUE` (the default), the lines are returned without touching any file, so the user can review them before applying. When `dry_run = FALSE`, `path` must be supplied explicitly.

Usage

```

use_ppm(
  scope = c("user", "project"),
  platform = NULL,
  repo = "cran",
  dry_run = TRUE,
  path = NULL
)

```

Arguments

<code>scope</code>	"user" edits the user .Rprofile, "project" edits the current project .Rprofile.
<code>platform</code>	Platform specification accepted by resolve_platform() .
<code>repo</code>	Repository name.
<code>dry_run</code>	If TRUE, return the lines that would be written without editing files.
<code>path</code>	Explicit .Rprofile path used when <code>dry_run = FALSE</code> .

Value

The configuration lines, invisibly when written.

See Also

Other ppm: [check_ppm\(\)](#), [ppm_platforms\(\)](#), [ppm_repo\(\)](#), [ppm_sysreqs\(\)](#)

Examples

```
use_ppm("user", platform = "ubuntu-22.04", dry_run = TRUE)

# Write to a throwaway .Rprofile under tempdir():
use_ppm(
  "user",
  platform = "ubuntu-22.04",
  dry_run = FALSE,
  path = file.path(tempdir(), ".Rprofile")
)
```

```
write_dockerfile_snippet
```

Write a Dockerfile snippet

Description

Writes the output of [dockerfile\(\)](#) to a file so it can be appended to an existing Dockerfile or included verbatim.

Usage

```
write_dockerfile_snippet(plan, path)
```

Arguments

plan	A sysreqr_plan.
path	Output path.

Value

path, invisibly.

See Also

Other output: [as_install_plan\(\)](#), [write_install_script\(\)](#), [write_json\(\)](#), [write_report\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
write_dockerfile_snippet(plan, file.path(tempdir(), "Dockerfile.sysreqs"))
```

write_install_script *Write an install script*

Description

Writes a POSIX-shell install script. The script begins with `#!/usr/bin/env sh` and `set -eu` and is marked executable.

Usage

```
write_install_script(plan, path)
```

Arguments

plan	A sysreqr_plan.
path	Output path.

Value

path, invisibly.

See Also

Other output: [as_install_plan\(\)](#), [write_dockerfile_snippet\(\)](#), [write_json\(\)](#), [write_report\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
write_install_script(plan, file.path(tempdir(), "install-sysreqs.sh"))
```

write_json *Write a sysreqr plan as JSON*

Description

Serializes the plan data frame to JSON.

Usage

```
write_json(plan, path)
```

Arguments

plan	A sysreqr_plan.
path	Output path.

Value

path, invisibly.

See Also

Other output: [as_install_plan\(\)](#), [write_dockerfile_snippet\(\)](#), [write_install_script\(\)](#), [write_report\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
write_json(plan, file.path(tempdir(), "sysreqs.json"))
```

write_report

Write a Markdown report

Description

Produces a human-readable Markdown report describing the platform, selected backend, R packages checked, system packages needed, and a suggested install command.

Usage

```
write_report(plan, path)
```

Arguments

plan	A sysreqr_plan.
path	Output path.

Value

path, invisibly.

See Also

Other output: [as_install_plan\(\)](#), [write_dockerfile_snippet\(\)](#), [write_install_script\(\)](#), [write_json\(\)](#)

Examples

```
plan <- check_packages("xml2", platform = "ubuntu-22.04")
write_report(plan, file.path(tempdir(), "SYSREQS.md"))
```

Index

- * **commands**
 - admin_request, 2
 - dockerfile, 13
 - github_actions, 15
 - install_command, 16
- * **diagnose**
 - check_error, 4
 - diagnose_failed_packages, 11
 - diagnose_log, 12
- * **output**
 - as_install_plan, 4
 - write_dockerfile_snippet, 24
 - write_install_script, 25
 - write_json, 25
 - write_report, 26
- * **plan**
 - as_data_frame, 3
 - is_sysreqr_plan, 17
 - print.sysreqr_plan, 20
- * **platform**
 - detect_package_manager, 9
 - detect_platform, 9
 - resolve_platform, 21
- * **ppm**
 - check_ppm, 7
 - ppm_platforms, 17
 - ppm_repo, 18
 - ppm_sysreqs, 19
 - use_ppm, 23
- * **preflight**
 - check_library, 5
 - check_packages, 6
 - check_project, 8
 - detect_project_packages, 10
- * **setup**
 - explain, 14
 - print.sysreqr_setup_advice, 20
 - setup_advice, 22
- admin_request, 2
- admin_request(), 14–16
- as_data_frame, 3
- as_data_frame(), 17, 20
- as_install_plan, 4
- as_install_plan(), 24–26
- check_error, 4
- check_error(), 12, 13
- check_library, 5
- check_library(), 7, 8, 11
- check_packages, 6
- check_packages(), 3, 6, 8, 11, 14–16
- check_ppm, 7
- check_ppm(), 18, 19, 24
- check_project, 8
- check_project(), 6, 7, 11
- detect_package_manager, 9
- detect_package_manager(), 10, 21
- detect_platform, 9
- detect_platform(), 9, 21
- detect_project_packages, 10
- detect_project_packages(), 6–8
- diagnose_failed_packages, 11
- diagnose_failed_packages(), 5, 13
- diagnose_install_log (diagnose_log), 12
- diagnose_log, 12
- diagnose_log(), 4, 5, 7, 12
- dockerfile, 13
- dockerfile(), 3, 15, 16, 24
- explain, 14
- explain(), 21, 22
- gha (github_actions), 15
- github_actions, 15
- github_actions(), 3, 14, 16
- install_command, 16
- install_command(), 3, 14, 15
- is_sysreqr_plan, 17

`is_sysreqr_plan()`, 3, 20

`ppm_platforms`, 17
`ppm_platforms()`, 7, 18, 19, 24
`ppm_repo`, 18
`ppm_repo()`, 7, 18, 19, 24
`ppm_sysreqs`, 19
`ppm_sysreqs()`, 7, 18, 24
`print_sysreqr_plan`, 20
`print_sysreqr_plan()`, 3, 17
`print_sysreqr_setup_advice`, 20
`print_sysreqr_setup_advice()`, 15, 22

`resolve_platform`, 21
`resolve_platform()`, 3, 5–11, 13–16, 18, 19, 22, 23

`setup_advice`, 22
`setup_advice()`, 15, 21

`use_ppm`, 23
`use_ppm()`, 7, 18, 19

`write_dockerfile_snippet`, 24
`write_dockerfile_snippet()`, 4, 25, 26
`write_install_script`, 25
`write_install_script()`, 4, 24, 26
`write_json`, 25
`write_json()`, 4, 24–26
`write_report`, 26
`write_report()`, 4, 24–26