Package 'vmr'

May 8, 2026

Type Package

Encoding UTF-8

Title Virtual Machines for R

Version 0.0.6

Date 2023-03-07

Maintainer Jean-François Rey <jf.rey.public@gmail.com>

Description Manage, provision and use Virtual Machines pre-configured for R.
Develop, test and build package in a clean environment.
'Vagrant' tool and a provider (such as 'Virtualbox') have to be installed.

URL https://gitlab.com/rstuff/vmr, https://rstuff.gitlab.io/vmr

BugReports https://gitlab.com/rstuff/vmr/-/issues

License GPL (>= 3)

BuildVignettes true

NeedsCompilation no

Biarch true

SystemRequirements Vagrant <https://www.vagrantup.com>

Depends utils, R (>= 3.3.0)

Imports jsonlite, curl

Collate 'vmr.R' 'package.R' 'virtualbox.R' 'vagrantcloudAPI.R'
'vagrant.R' 'vmr-methods.R'

RoxygenNote 7.2.3

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

Config/testthat/edition 3

VignetteBuilder knitr

Author Jean-François Rey [cre, aut]

Repository CRAN

Date/Publication 2023-03-07 21:00:02 UTC

Contents

vmr-package *Virtual Machines for R*

Description

Manage, provision and use Virtual Machines pre-configured for R. Develop, test and build package in a clean environment. 'Vagrant' tool and a provider (such as 'Virtualbox') have to be installed.

Details

Package:	vmr
Type:	Package
Version:	0.0.6
Date:	2023-03-07
License:	GPL (>=3)

This package is a wrap of the Vagrant tool and more. It allows to manage, provision and use Virtual Machines pre-configured for R. It currently only uses 'Virtualbox' (>= 6.1.14) as provider. Vagrant tool have to be installed too. Used VMs come from https://app.vagrantup.com/VMR repository and the sources use to generate them can be found at https://gitlab.com/rstuff/vms. See vignettes for the documentations `browseVignette("vmr")`.

Author(s)

Jean-François Rey <jf.rey.public@gmail.com>

Maintainer: Jean-François Rey <jf.rey.public@gmail.com>

See Also

Useful links:

- https://gitlab.com/rstuff/vmr
- https://rstuff.gitlab.io/vmr
- Report bugs at https://gitlab.com/rstuff/vmr/-/issues

Examples

```
## Not run:
library("vmr")

## End(Not run)
```

getProviderOptions *List provider options*

Description

List a provider available options.

Usage

```
getProviderOptions(provider = "virtualbox", details = FALSE)
```

Arguments

provider	a provider name
details	if TRUE print options, otherwise return default options

Details

It return a list of options name and value for a specific provider. To get the help page do ?<provider_name>Options(), for example [virtualboxOptions()].

Value

a list of options

Examples

```
vbOpts <- getProviderOptions(provider = "virtualbox")
print(vbOpts)
```

print.vmr *Print* **vmr** *object information*

Description

print information from a **vmr** object

Usage

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

Arguments

x	a **vmr** object
...	optional print arguments

Value

the **vmr** object (via invisible(x))

`summary.vmr` *Summary* **vmr** *object information*

Description

print information from a **vmr** object

Usage

```
## S3 method for class 'vmr'
summary(object, ...)
```

Arguments

`object`	a **vmr** object
`...`	optional print arguments

Value

the **vmr** object (via invisible(x))

`virtualboxGitlabRunner` *Configure the guest VM to be use as a Gitlab-Runner*

Description

Configure the guest VM to be use as a GitLab Runner and return the command to run in shell to register it.

Usage

```
virtualboxGitlabRunner(
  vmr,
  gitlab_url,
  gt_token,
  snapshot_name = "",
  vm_name = ""
)
```

Arguments

vmr	a **vmr** object
gitlab_url	a GitLab URL with protocol (http or https)
gt_token	a GitLab registration token
snapshot_name	name of a snapshot to use if any
vm_name	the 'VitualBox' VM name if not specified in 'vmr' object provider_options.

Value

Character command to run in shell to register it

Examples

```
## Not run:
cmd <- virtualboxGitLabRunner(vmr, "gitlab.com", "mytoken")
system(cmd)

## End(Not run)
```

---

virtualboxOptions *List 'VirtualBox' options available*

Description

List available options for 'VirtualBox' provider

Usage

```
virtualboxOptions(details = TRUE)
```

Arguments

details	if TRUE print options (default), otherwise only return default options

Details

Get the 'VirtualBox' default options. It return a list as follow:

```
list(
gui = TRUE,
name = NULL,
nic_type = NULL,
linked_clone = FALSE,
check_guest_additions = TRUE,
modifyvm = list(cpus = "2", memory = "4096")
)
```

- **gui**: if TRUE show the GUI, otherwise headless mode is actived
- **name**: the 'VirtualBox' instance name
- **nic_type**: the NIC type for the network interface to use, by default use the default one. see VirtualBox Networking
- **linked_clone**: if TRUE, linked clones are based on a master VM, which is generated by importing the base box only once the first time it is required. For the linked clones only differencing disk images are created where the parent disk image belongs to the master VM. (Be careful, master VM can't be remove until linked_clone still exists)
- **check_guest_additions**: If TRUE (default) check if guest have guest additions installed.
- **modifyvm**: list of 'VirtualBox' properties for the guest VM (such as number of cpus, memory size,...). see 'VirtualBox' modifyvm

Value

A default list of options

```
list(
gui = TRUE,
name = NULL,
nic_type = NULL,
linked_clone = FALSE,
check_guest_additions = TRUE,
modifyvm = list(cpus = "2", memory = "4096")
)
```

Examples

```
## Not run:
vb.opts <- virtualboxOptions(details = FALSE)
vb.opts$modifyvm$cpus <- "4"
vb.opts$modifyvm$memory <- "8192"
vb.opts

## End(Not run)
```

---

vmrBoxDownload *Download a Box*

Description

Download a box from a **vmr** object.

Usage

```
vmrBoxDownload(vmr)
```

Arguments

vmr	a **vmr** object

Value

a **vmr** object

---

vmrConfigSSH *Configure ssh*

Description

Configure ssh credential.

Usage

```
vmrConfigSSH(
  vmr,
  ssh_user = "vagrant",
  ssh_pwd = "vagrant",
  port = "",
  ssh_private_key_path = ""
)
```

Arguments

vmr	a **vmr** object
ssh_user	the ssh user (default 'vagrant')
ssh_pwd	the ssh pwd if any (default 'vagrant')
port	the ssh port (default empty)
ssh_private_key_path	path to the private ssh key to use (default empty, use insecure vagrant key)

Details

by default **vmr** use vagrant as user/password and insecure key for ssh connection. This behavior can be change here, by setting an another user and/or ssh keys. Calling with no arguments will disable this option. Be careful, ssh using only password may result of *vmr* functions bugs.

Value

an updated **vmr** object

Examples

```
## Not run:
vmr <- vmrConfigSSH(ssh_user = "John", ssh_pwd = "d0e", port = "22")
vmr <- vmrConfigSSH(ssh_user = "John", private_key_path = "/path/to/private/key/")

## End(Not run)
```

`vmrConnect` *Open a ssh connection to guest machine*

Description

Open a ssh connection to guest machine

Usage

```
vmrConnect(vmr)
```

Arguments

`vmr`	a **vmr** object

Details

To open a ssh connection 'ssh' package have to be installed.

Value

a **vmr** object

`vmrCreate` *Create a* **vmr** *environment class*

Description

Create a **vmr** object.

Usage

```
vmrCreate(
  name,
  provider = "virtualbox",
  version = "latest",
  provider.options = virtualboxOptions(FALSE)
)
```

Arguments

name	a box name
provider	the box provider (default: "virtualbox")
version	the box version (default : "latest")
provider.options	provider options (call [getProviderOptions()] to get values)

Details

Create a S3 **vmr** object (a simple list). The object contains all information needed to configure and manage a **vmr** environment (a vagrant environment).

A **vmr** environment need mostly a box *name* and a *provider*. The environment is attached to the current working directory.

vmr object main attributs:

- **path**: working directory
- **org**: Vagrant cloud user/organization name 'VMR'
- **box**: the box name
- **version**: the box version
- **provider**: the provider
- **provider_options**: the provider options (see [getProviderOptions()])
- **vagrantName**: Vagrant environment name
- **ID** <- Vagrant environment ID
- **synced_folder**: a list with source and destination
- **ssh_user**: the ssh user
- **ssh_pwd**: the ssh user password
- **ssh_port**: the ssh port
- **ssh_private_key_path**: the private ssh key path

Value

a **vmr** object (see details)

Examples

```
## Not run:
# List boxes available
boxes <- vmrList()
# Create a vmr object
vmr <- vmrCreate(boxes$Name[1])

# to customize the guest machine for virtualbox
virtualboxOpts <- getProviderOptions(provider = "virtualbox")
virtualboxOpts$modifyvm <- list(cpus = 4, memory = 4096)
virtualboxOpts$name <- "My VM Cool Name"
```

```
# To specify a provider and version
vmr <- vmrCreate(
  name = boxes$Name[1],
  provider = "virtualbox",
  version = boxes$Version[1],
  provider.options = virtualboxOpts
)

## End(Not run)
```

---

vmrDestroy *Remove all resources created in a* **vmr** *environment*

Description

Remove all resources created by vmrStart()

Usage

```
vmrDestroy(id = "", force = FALSE)
```

Arguments

id	a **vmr** environment id (default : "" id from the current environment)
force	if TRUE force to remove

Details

Will by default remove all resources created from the current **vmr** environment. By specifying the *id* any environment with this *id* will be remove.

Value

the vagrant environment id

Examples

```
## Not run:
vmrStop()
vmrDestroy()

## End(Not run)
```

vmrDisconnect *Disconnect ssh connection to guest machine*

Description

Close a ssh connection to the guest machine

Usage

```
vmrDisconnect(vmr)
```

Arguments

vmr	a **vmr** object

Details

'ssh' package need to be installed.

Value

a **vmr** object

vmrExec *Execute R methods into guest machine*

Description

Run R method into guest machine.

Usage

```
vmrExec(cmd = c())
```

Arguments

cmd	list of R command

Details

call Rscript -e "cmd" into the guest machine from the current **vmr** environment. Command are independents and do not keep memory of past commands.

Value

NULL

Examples

```
## Not run:
cmd <- c("Sys.info()", 'print("Hello World!")')
vmrExec(cmd)

## End(Not run)
```

---

vmrInfo *Get guest machine information*

Description

Get guest machine information. Print OS, R, R-devel and R packages information. Still in development.

Usage

```
vmrInfo()
```

Value

NULL

Examples

```
## Not run:
boxes <- vmrList()
vmr <- vmrCreate(boxes$Name[1])
vmr <- vmrInitEnv(vmr)
vmrStart()
vmrInfo()

## End(Not run)
```

---

vmrInitEnv *Initialize the* **vmr** *environment*

Description

Create **vmr** environment in the current directory. Set configuration into a template file name "Vagrantfile" and download the box if needed.

Usage

```
vmrInitEnv(vmr, force.vagrantfile = FALSE, force.download = FALSE)
```

Arguments

vmr	a **vmr** object
force.vagrantfile	if TRUE force to overwrite environment configuration (default FALSE)
force.download	if TRUE force to download the box, otherwise do not (default FALSE).

Details

The **vmr** environment consist of a directory (the working directory) and a template file name *Vagrantfile*. If the box is not present in localhost it will be download.

Value

the **vmr** object

Examples

```
## Not run:
boxes <- vmrList()
vmr <- vmrCreate(boxes$Name[1])
vmr <- vmrInitEnv(vmr)

## End(Not run)
```

---

vmrInstallPackages *Install R packages into guest machine*

Description

Install a list of R packages into the guest machine of the current **vmr** environment.

Usage

```
vmrInstallPackages(pkgs = c())
```

Arguments

pkgs	list of R packages

Value

installed packages vector

Examples

```
## Not run:
vmrInstallPackages(c("vmr"))

## End(Not run)
```

vmrIsRunning *Is* **vmr** *environment running*

Description

Check if a guest machine in a **vmr** environment is running

Usage

```
vmrIsRunning()
```

Value

TRUE if running, otherwise FALSE

Examples

```
## Not run:
lboxes <- vmrList()
vmr <- vmrCreate(lboxes$Name[1])
vmr <- vmrInitEnv(vmr)
vmrStart()
vmrIsRunning()
vmrStop()
vmrIsRunning()

## End(Not run)
```

vmrList *List available boxes from VagrantCloud*

Description

List of available boxes from a VagrantCloud organization account.

Usage

```
vmrList(org = .VagrantCloudOrganization)
```

Arguments

org	Vagrant Cloud organization name (default : 'VMR')

Details

Default usage lists boxes preconfigurated with R from VMR organization account.

Value

a data.frame with Name, Provider, Version and Description of available boxes

---

`vmrListBox` *List all available version of a box*

Description

List all versions and providers available of a box.

Usage

```
vmrListBox(box_name, org = .VagrantCloudOrganization)
```

Arguments

`box_name`	the box name
`org`	Vagrant Cloud organization name (default : 'VMR')

Details

List information of a box from VagrantCloud. Default usage list information of a box preconfigurated with R from VMR organization account.

Value

a data.frame with "Name, "Version", "Description", "Provider" and "Date" of the box

Examples

```
## Not run:
# List Boxes
boxes <- vmrList()
# Box informaion
box_info <- vmrListBox(boxes$Name[1])
box_info

## End(Not run)
```

vmrListSnapshot *List snapshot of the guest machine*

Description

Print all snapshot name of the guest machine

Usage

```
vmrListSnapshot()
```

Value

NULL

vmrLoad *Load a* **vmr** *environment containing a Vagrant file*

Description

Load a **vmr** environment containing a VagrantFile and create a **vmr** object (see [vmrCreate()] for object details).

Usage

```
vmrLoad(dir = "./", vagrantfileName = "Vagrantfile")
```

Arguments

dir	the **vmr** environment directory (default: "./")
vagrantfileName	a Vagrantfile name (default: "Vagrantfile")

Details

It read a Vagrant file template with **vmr** compatible parameters. It's an experimental Vagrant file reading, some parameters may not be loaded.

Value

a **vmr** object

Examples

```
## Not run:
# load the Vagrantfile in the current directory
vmr <- vmrLoad(getwd())

## End(Not run)
```

`vmrLocalBoxList` *List downloaded boxes*

Description

List all boxes downloaded in localhost

Usage

```
vmrLocalBoxList()
```

Value

a data.frame with boxes Name, Providers and Version

Examples

```
## Not run:
localBoxes <- vmrLocalBoxList()
print(localBoxes)

## End(Not run)
```

`vmrLocalBoxPrune` *Remove old installed boxes*

Description

Removes old versions of installed boxes.

Usage

```
vmrLocalBoxPrune()
```

Value

a data.frame of still installed boxes (Name, Poviders and Version)

Examples

```
## Not run:
vmrLocalBoxPrune()

## End(Not run)
```

---

vmrLocalBoxRemove *Remove a box from localhost*

Description

Remove a specific box from localhost.

Usage

```
vmrLocalBoxRemove(name, provider = "", version = "", force = FALSE)
```

Arguments

name	the box name
provider	the box provider (default: first provider found)
version	the box version (default: version available)
force	if TRUE force to remove

Value

execution code or message

Examples

```
## Not run:
lboxes <- vmrLocalBoxList()
vmrLocalBoxRemove(lboxes$Name[[1]])
# if multiple providers and versions
vmrLocalBoxRemove(lboxes$Name[[1]], lboxes$Provider[[1]], lboxes$Version[[1]])

## End(Not run)
```

vmrLocalBoxUpdate *Update local box version*

Description

Download the latest version of the box use in the current **vmr** environment.

Usage

```
vmrLocalBoxUpdate()
```

Value

execution code or message

vmrMountDir *Mount a host directory to guest*

Description

Mount a host directory to the guest machine.

Usage

```
vmrMountDir(vmr, src = "", dest = "")
```

Arguments

vmr	a **vmr** object
src	a host directory
dest	a destination guest directory

Details

If the option of mounting a directory is available in the guest provider, it will mount *src* to *destination* directory. Calling with no arguments will disable this option.

Value

a **vmr** object

Examples

```
## Not run:
boxes <- vmrList()
vmr <- vmrCreate(boxes$Name[1])
vmr <- vmrMountDir(vmr, src = getwd(), dest = "/vmr")
vmr <- vmrInitEnv(vmr)
vmrStart()

## End(Not run)
```

---

vmrPackageBuild *Build a package in the guest machine*

Description

Build a package bundle or binary into the guest machine.

Usage

```
vmrPackageBuild(pkg = "./", binary = FALSE)
```

Arguments

pkg	a package directory or a tar.gz file
binary	if TRUE build binary package otherwise FALSE

Details

upload the package and run devtools::build() (build available in $HOME/vmr/package/pkg) in the current **vmr** environment.

Value

NULL

---

vmrPackageCheck *Perform a package check on guest*

Description

Perform a package check into the guest

Usage

```
vmrPackageCheck(pkg = "./")
```

Arguments

pkg	a package directory or a tar.gz file

Details

upload the package and run devtools::check() into the guest machine. (check available in $HOME/vmr/package/pkg). Checking a directory with multiple files may slower upload, prefer tar.gz file

Value

NULL

Examples

```
## Not run:
vmrPackageCheck("vmr_package.tar.gz")

## End(Not run)
```

---

vmrPackageTest *Test a package into a guest machine*

Description

Test a package into a guest machine

Usage

```
vmrPackageTest(pkg = "./")
```

Arguments

pkg	a package directory or tar.gz

Details

Perform a package check into the guest machine of the current **vmr** environment using devtools::test(). (tests are available in $HOME/vmr/package/pkg)

Value

NULL

vmrProvision *Provision a* **vmr** *environment*

Description

Provision a **vmr** environment.

Usage

```
vmrProvision(cmd = c(), elts = c(), dest = "")
```

Arguments

cmd	list of shell commands
elts	list of files and/or directories
dest	destination of elts (default HOME/vmr)

Details

Upload 'elts' files and/or directories to the guest machine 'dest' from the current **vmr** environment. And finaly run shell commands 'cmd' in the guest machine.

Value

NULL

vmrRemoveSnapshot *remove a snapshot of the guest machine*

Description

remove a snapshot of the guest machine

Usage

```
vmrRemoveSnapshot(snap_name)
```

Arguments

snap_name	the snapshot name

Value

NULL

vmrRestoreSnapshot *Restore a snapshot of the guest machine*

Description

Restore a snapshot of the guest machine.

Usage

```
vmrRestoreSnapshot(snap_name)
```

Arguments

snap_name	the snapshot name

Value

the snapshot name

Examples

```
## Not run:
vmrRestoreSnapshot("my snapshot")

## End(Not run)
```

vmrResume *Resume a stopped guest machine*

Description

Resume a stopped guest machine.

Usage

```
vmrResume()
```

Details

In the current **vmr** environment, start a stopped ([vmrSuspend()]) guest machine.

Value

NULL

vmrSend *Send files and/or directories to guest machine*

Description

Send files and/or directories to the guest machine in the current **vmr** environment. They are upload into ~/vmr/ directory.

Usage

```
vmrSend(elt = c())
```

Arguments

elt	list of files and directories

Value

0 if OK, message otherwise

Examples

```
## Not run:
vmrSend(c("myfile"))

## End(Not run)
```

vmrSetVerbose *Set verbose level*

Description

Set verbose level for vmr package functions

Usage

```
vmrSetVerbose(verbose_mode = "Normal")
```

Arguments

verbose_mode	"None", "Normal" or "Full"

Details

Three verboses mode is available:

- "None" : print nothings
- "Normal" : print essential
- "Full" : print all

Value

invisible verbose value

`vmrStart` *Start a* **vmr** *environment*

Description

Start a guest virtual machine using the current **vmr** environment (directory and Vagrantfile template)

Usage

```
vmrStart()
```

Value

the vmr environment unique id

Examples

```
## Not run:
lboxes <- vmrList()
vmr <- vmrCreate(lboxes$Name[1])
vmr <- vmrInitEnv(vmr)
vmrStart()
vmrStop()

## End(Not run)
```

vmrStatus *Get the state of the guest machine*

Description

Print guest machine state in the current **vmr** environment.

Usage

```
vmrStatus()
```

Value

a data.frame with Name, Provider and state

vmrStop *Stop a* **vmr** *environement*

Description

Stop a guest virtual machine in the current **vmr** environment.

Usage

```
vmrStop(force = FALSE)
```

Arguments

force	if TRUE force to stop (powerOff), otherwise FALSE clean shutdown

Value

NULL

Examples

```
## Not run:
lboxes <- vmrList()
vmr <- vmrCreate(lboxes$Name[1])
vmr <- vmrInitEnv(vmr)
vmrStart()
vmrStop()

## End(Not run)
```

vmrSuspend *Save state and stop guest machine*

Description

Save the guest machine and stop it.

Usage

```
vmrSuspend()
```

Details

In the current **vmr** environment, save the state of the guest machine and stop it.

Value

NULL

vmrTakeSnapshot *Take a snapshot of the guest machine*

Description

Take a snapshot of the guest machine.

Usage

```
vmrTakeSnapshot(snap_name)
```

Arguments

snap_name	the name given to the snapshot

Value

the snapshot name (invisible)

Examples

```
## Not run:
vmrTakeSnapshot("my snapshot")

## End(Not run)
```

vmrUpdateEnvVersion *Update a* **vmr** *environment.*

Description

Force to use the latest box version of the current **vmr** environment.

Usage

```
vmrUpdateEnvVersion(vmr)
```

Arguments

vmr	a **vmr** object

Details

Put **vmr** object version to latest and update the Vagrant File template. Download the new box version if needed.

Value

a **vmr** object

Examples

```
## Not run:
boxes <- vmrList()
vmr <- vmrCreate(boxes$Name[1], version = "oldone")
vmr <- vmrInitEnv(vmr)

# update to latest
vmr <- vmrUpdateEnvVersion(vmr)
vmrStart()

## End(Not run)
```

vmrUpdatePackages *Update R packages installed*

Description

Updates R packages installed in the guest machine.

Usage

```
vmrUpdatePackages()
```

Details

Will perform a `update.packages()` in the guest machine of the current **vmr** environment.

Examples

```
## Not run:
lboxes <- vmrList()
vmr <- vmrCreate(lboxes$Name[1])
vmr <- vmrInitEnv(vmr)
vmrStart()
vmrUpdatePackages()

## End(Not run)
```

Index