R SUITE

R Suite CLI

reference manual

GETTING HELP

```
rsuite help
```

INSTALLING R SUITE

```
rsuite install
```

advanced options

```
rsuite install -v
```

some specific repository

```
rsuite install -u
http://url.to.your.repository
```

all supported options

```
rsuite install -h
```

The RSuiteRStudio package is an RStudio addin

```
rsuite install --rstudio-addin
```

Updating RSuite CLI

```
rsuite update
```

PROJECT MANAGEMENT

Starting project

```
rsuite proj start -n MyProject
```

If you do not want adding project under revision control

```
rsuite proj start -n MyProject --skip_rc
```

Creating package

```
rsuite proj pkgadd -n MyPackage
```

If you do not want adding package under revision control

```
rsuite proj pkgadd -n MyPackage --skip_rc
```

Project environment locking

```
rsuite proj lock
```

To safely unlock

```
rsuite proj unlock
```

Building project local enviroment

```
rsuite proj depsinst
```

short for --clean

```
rsuite proj depsinst -c
```

to clean up internal project environment before installing all required packages

short for --relock

```
rsuite proj depsinst -r
```

to allow env.lock file updating in case of removed or updated dependencies

Building project packages

```
rsuite proj build
```

to enforce rebuilding your project packages

```
rsuite proj build -f
```

Cleaning unused project dependencies

```
rsuite proj depsclean
```

Building deployment package

```
rsuite proj zip
```

```
rsuite proj zip -p /path/to/put/deployment/package
```

to put created deployment package

```
rsuite proj zip --version=1.0
```

If you do not want project consistency check for some reason

Testing project

```
rsuite proj test
```

```
rsuite proj test -d
path/to/test/folder/instide/your/project/folder/tree
```

If your tests are located elsewhere in project folder you can inform the command on that following way

BUILDING PKGZIP PACKAGES

Building PKGZIP containing project packages

```
rsuite pkgzip proj
```

You can specify which packages should be included following way

```
rsuite pkgzip proj -n Package1,Package2
```

If for some reason you do not want check project source consistency you can enforce PKGZIP version following way:

```
rsuite pkgzip proj --version=1.0
```

You can also decide which kind of packages will be built and included in PKGZIP (source or binary) with -b (short for --binary) option:

```
rsuite pkgzip proj -b TRUE
```

If you want to include also all dependencies in PKGZIP pass --with-deps option:

```
rsuite pkgzip proj -b FALSE --with-deps
```

In case you are building repository of packages (which you have access to while building PKGZIP) you probably would want to include into PKGZIP only dependencies which could not be satisfied by current content of the repository

```
rsuite pkgzip proj -b FALSE --with-deps --filter-repo
http://url.to.your.repository
```

Building PKGZIP containing in file packages

```
rsuite pkgzip proj -f
/path/to/file1.tar.gz,/path/to/file2.tar.gz
```

Building PKGZIP containing external packages

```
rsuite pkgzip ext -n package1,package2
```

You can specify that you want to include source (or binary) version of packages in PKGZIP with -b (short for --binary) option:

```
rsuite pkgzip ext -n package1,package2 -b TRUE
```

If you want to include also all dependencies in PKGZIP pass --with-deps option:

```
rsuite pkgzip ext -n package1,package2 -b FALSE --with-deps
```

In case you are building repository of packages (which you have access to while building PKGZIP) you probably would want to include into PKGZIP only dependencies which could not be satisfied by current content of the repository

```
rsuite pkgzip ext -n package1,package2 --with-deps --filter
-repo  http://url.to.your.repository
```

REPOSITORY MANAGEMENT

All repo sub-commands accept beside standard -v and -h following options

```
-d (short for --dir)
```

which takes as parameter path to local (in directory) repository

```
-s (short for --s3_url)
```

which takes as parameter url to S3 repository

Initializing repository

To create repository structure execute following

```
rsuite repo init -s http://your-s3-
bucket.s3.amazonaws.com/path
```

You can specify that you want (or not) initialize it for binary packages with following command

```
rsuite repo init -s http://your-s3-
bucket.s3.amazonaws.com/path -b TRUE
```

You can also force repository structure creation for different R version (which is important for binary packages) with following command

```
rsuite repo init -d /path/to/your/repository --rver 3.4
```

Adding project packages to repository

To add project packages to a repository execute following

```
rsuite repo addproj -s http://your-s3-
bucket.s3.amazonaws.com/path
```

You can specify which packages should be added following way

```
rsuite repo addproj -s http://your-s3-
bucket.s3.amazonaws.com/path -n Package1,Package2
```

If for some reason you do not want check project source consistency while rebuilding project packages you can skip RC checks:

```
rsuite repo addproj -s http://your-s3-
bucket.s3.amazonaws.com/path --skip_rc
```

You can also decide which kind of packages will be built and added to repository (source or binary) with -b (short for --binary) option:

```
rsuite repo addproj -s http://your-s3-
bucket.s3.amazonaws.com/path -b FALSE
```

If you want to add also all dependencies which are not currently present in the repository pass --with-deps option:

```
rsuite repo addproj -s http://your-s3-
bucket.s3.amazonaws.com/path -b TRUE --with-deps
```

Adding in file packages to repository

```
rsuite repo addfile -d /path/to/your/repository -f
/path/to/file1.tar.gz,/path/to/file2.tar.gz
```

You can specify that you want to add source (or binary) version of packages to repository with -b (short for --binary) option:

```
rsuite repo addext -d /path/to/your/repository -n
package1,package2 -b TRUE
```

If you want to add also all dependencies which are not currently present in the repository pass --with-deps option:

```
rsuite repo addext -d /path/to/your/repository -n
package1,package2 -b TRUE --with-deps
```

Adding content of PKGZIP to repository

```
rsuite repo addpkgzip -s http://your-s3-
bucket.s3.amazonaws.com/path -z /path/to/pkgzip.zip
```

Adding package from GitHub to repository

```
rsuite repo addgithub -d /path/to/your/repository -r
github/ProjectName
```

GitHub repository can be specified in format

```
username/repo[/subdir][@ref|#pull]
```

You can also specify following options to addgithub:

```
-H (short for --host)
```

which GitHub API host to use. Use it to override with your GitHub enterprise hostname, for example, 'github.hostname.com/api/v3'.

```
-b (short for --binary)
```

which takes as parameter logical value (T/F/TRUE/FALSE). It specifies what kind of package will be added to the repository: system specific binary of source.

```
--rver
```

which takes R version number to target built package for (important for binary packages).

```
--with-deps
```

If passed will upload also dependencies which are not currently present in the repository.

List contents of repository

```
rsuite repo list -s http://your-s3-
       bucket.s3.amazonaws.com/path
```

Remove packages from repository

```
rsuite repo remove -s http://your-s3-
bucket.s3.amazonaws.com/path -r
Package1==Version1,Package2==Version2
```

TEMPLATE MANAGEMENT

Creating custom project and package templates

To create a new template use the following command

```
rsuite tmpl start  -n MyTemplate
```

This command creates a folder called MyTemplate in your working directory. All templates have the following file structure

```
2018-06-22  07:37    <DIR>          .
2018-06-22  07:37    <DIR>          ..
2018-06-21  07:43    <DIR>          package
2018-06-22  07:38    <DIR>          project
               0 File(s)              0 bytes
               4 Dir(s)  260 126 388 224 bytes free
```

The project and package directories contain the default R Suite project and package files. You can add, delete and modify those files according to your preferences. If you want to create a template containing only a project or package directory, you can do it by using the --prj and --pkg options accordingly.

```
rsuite tmpl start  -n MyTemplate --prj
```

```
rsuite tmpl start  -n MyTemplate --pkg
```

If you want to create a template in a different location just use the -p (short for --path) option

```
rsuite tmpl start -n MyTemplate -p /path/to/create/template
```

Important: Templates have specific requirements in case of projects they have to contain a PARAMETERS file as for packages they have to contain a DESCRIPTION file. R Suite templates support the usage of markers - special keywords which will be replaced while creating a project/package from a custom template. All markers have the following form: __<word>__ for example __ProjectName__. The following markers are supported:

```
__ProjectName__ - will be replaced with the name of the
project
__PackageName__ - will be replaced with the name of the
package
__RSuiteVersion__ - will be replaced with the used
RSuite version
__RVersion__ - will be replaced with the used R version
__Date__ - will be replaced with the current date
__User__ - will be replaced with the username
__LatestMRAN__ - will be replaced with MRAN[<Date of latest
working MRAN snapshot from the last 2 weeks>]
```

Registering custom templates

To register a template created outside of the default template directory use the following command

```
rsuite tmpl register -p /path/to/created/template
```

Linux users can also register templates in the global template directory (/etc/.rsuite/templates) using the -g (short for --global) option

```
rsuite tmpl register -p /path/to/created/template -g
```

Important: to register a template in the global template directory extended permissions are required

All registered templates can be listed using the following command

```
rsuite tmpl list
```

Starting projects/packages using created templates

```
rsuite proj start -n MyProject -t MyTemplate
```

Unregistered templates can also be provided to the -t option. You simply have to provide the path to the template. Important: the path has to point to the directory containing the template files e.g. <path>/MyTemplate/project

```
rsuite proj start -n MyProject -t <path to template>
```

Similarly package templates can be used

```
rsuite proj pkgadd -n MyPackage -t MyTemplate
```

```
rsuite proj pkgadd -n MyPackage -t <path to template>
```

Important: The default RSuite template is called builtin and can also be provided to the -t option.

DOCKER INTEGRATION

Building project under different platform in docker container

```
rsuite docker zip -p centos
```

You can provide following options to rsuite docker zip:

```
-p (short for --platform)
```

takes platform name to build project under. centos, ubuntu and debian platforms are supported (ubuntu is default).

```
-d (short for --dest)
```

takes location to put generated zip into. Default is current folder.

```
--sh
```

accepts additional shell commands to execute inside container before building the project. You can pass commands to install additional system packages to required for project to build properly.

For example following command will install libglpk-dev on container before building project:

```
rsuite docker zip --sh "apt-get install -y libglpk-dev"
```

```
--dont-rm
```

If passed, container used to build the project will not be removed after command finished. If not passed container used to build project will be removed even if project building fails. Option is useful if you want to detect reasons for failure.

```
--packages
```

Accepts comma separated list of project packages to include in project build. If not passed all project packages will be included. Option is useful if your solution consists of number of images and only some packages must be present on some of them.

```
--exc-master
```

If passed, generated deployment zip will not contain any master scripts. It is useful if master scripts depend on some glue packages which you do not want to be present in deployment zip package.

```
--version
```

Accepts version number to tag deployment zip package with. Version number should be in form DD.DD. If not passed standard algorithm for tag selection is applied (ZipVersion from project PARAMETERS + revision from RC). If version is not enforced R Suite will check for source codes consistency against RC.

```
-i (short for --image)
```

takes name of docker image to use for building zip instead of default rsuite image.

It also accepts standard options -v and -h.

Building docker image for solution

```
rsuite docker img -t <my/image:tag>
```

You can provide following options to rsuite docker img:

```
--tag-latest
```

if passed image build will also tagged as latest.

```
-p (short for --platform)
```

takes platform name to deploy solution under. centos and ubuntu platforms are supported (ubuntu is default).

```
-f (short for --from)
```

accepts image name to use as base image. If not passed default base images will be used: for appropriate platform and R version.

```
-z (short for --zip)
```

accepts project deployment zip package file (built probably with 'rsuite docker zip' command). If passed project deployment zip package will not be built. Passed zip package will be used instead.

```
--templ
```

accepts Dockerfile template file to use for building image. It is regular Dockerfile which can contain tags to be replaced by RSuite CLI with automatically generated commands

Following tags are accepted:

<From>

will be replaced with base image name

<DeployProject>

will be replaced with commands for deploying project out of project deployment zip package.

If not passed default Dockerfile template will be generated of following content:

```
FROM <From>
<DeployProject>
```

```
--templ-ctx
```

accepts comma separated list of folders to add to image building context. You will be able to copy files from these folders inside your image while building with COPY command in Dockerfile.

```
--sh
```

accepts additional shell commands to pass to rsuite docker zip command then building project deployment zip package.

```
--packages
```

accepts comma separated list of project packages to pass to rsuite docker zip command then building project deployment zip package.

```
--exc-master
```

option passed to rsuite docker zip command then building project deployment zip package.

```
--version
```

Accepts version number to pass to rsuite docker zip command then building project deployment zip package.

```
-i (short for --image)
```

takes name of docker image to use for building zip (if not passed) instead of default rsuite image.

SYSTEM REQUIREMENTS MANAGEMENT

Collecting system requirements

```
rsuite sysreqs collect
```

Checking system against requirements

```
rsuite sysreqs check
```

Installing system requirements

```
rsuite sysreqs install
```

Create system update script

```
rsuite sysreqs script
```

FULL documentation: http://rsuite.io/RSuite_Tutorial.php