# RStudio Workbench Administration TESTING 2021-08.9.1444-490

**Administration Guide** 

2021-08-09

# 1. RStudio Workbench TESTING 2021-08.9.1444-490

RStudio Server enables you to provide a browser based interface (the RStudio IDE) to a version of R running on a remote Linux server. Deploying R and RStudio on a server has a number of benefits, including:

- The ability to access R sessions from any computer in any location;
- Easy sharing of code, data, and other files with colleagues;
- Allowing multiple users to share access to the more powerful compute resources (memory, processors, etc.) available on a well equipped server; and
- Centralized installation and configuration of R, R packages, TeX, and other supporting libraries.

This manual describes *RStudio Workbench*, which adds many enhancements to the open-source version of RStudio Server, including:

- The ability to run multiple concurrent RStudio IDE sessions per-user.
- Flexible use of multiple versions of R on the same server.
- The ability to run sessions on external cluster nodes, such as Kubernetes or Slurm.
- Support for Jupyter, JupyterLab, and VS Code editor sessions.
- Project sharing for easy collaboration within workgroups.
- Load balancing for increased capacity and higher availability.
- An administrative dashboard that provides insight into active sessions, server health, and monitoring of system-wide and per-user performance and resource metrics;
- Authentication using system accounts, ActiveDirectory, LDAP, SAML, OpenID, or Google Accounts;
- Full support for PAM (including PAM sessions for dynamically provisioning user resources);
- Ability to establish per-user or per-group CPU priorities and memory limits;
- HTTP enhancements including support for SSL and keep-alive for improved performance;
- Ability to restrict access to the server by IP:
- Customizable server health checks; and
- Suspend, terminate, or assume control of user sessions; Impersonate users for assistance and troubleshooting.

The documentation applies to both RStudio Server (Open Source) and RStudio Workbench (Pro), unless the section is specifically marked Pro, in which case it only applies to RStudio Workbench.

# Part I. Getting Started

### 2. Installation

#### 2.0.1. Prerequisites

#### System and Platform Requirements

RStudio Workbench runs on most modern Linux distributions, and can be accessed in most modern browsers. See the following link for detailed information on supported server and browser platforms as well as recommended minimal CPU, RAM, and disk storage requirements.

RStudio Workbench System Requirements

#### **R Versions**

RStudio requires a previous installation of R version 3.0.1 or higher; see below for instructions on installing R on your specific Linux distribution.

We recommend installing multiple versions of R. An environment with multiple versions of R helps you provide a stable, reproducible environment for your R developers. Install R using the directions at https://docs.rstudio.com/resources/install-r/.

#### **User Home Directories**

RStudio interacts frequently with user home directories. If you mount home directories with NFS, we recommend using the async mount option along with a modern, high-throughput network connection that can support many simultaneous clients. If you'd like your users to be able to share their projects with each other, see the section on Project Sharing for additional NFS requirements.

#### i Note

Due to its high latencies, we strongly discourage the use of EFS (Elastic File System) for home and shared directories within AWS. If EFS is used, RStudio will experience highly degraded performance. We recommend using a traditional NFSv3 or NFSv4 mount instead.

#### **Database Connectivity**

Since version 1.4, RStudio also requires database connectivity. In most cases, no additional configuration is needed - a SQLite database will be created and used automatically. However, if you are running a load-balanced setup, you will be required to set up a PostgreSQL database for RStudio Workbench to connect to; if one is not configured, the service will refuse to start. If you are using load balancing, make sure that you have an available PostgreSQL database to use **before** installation! See the Database section for more information.

# 3. Management Script

RStudio management tasks are performed using the rstudio-server utility (installed under /usr/sbin). This utility enables the stopping, starting, and restarting of the server, enumeration and suspension of user sessions, taking the server offline, as well as the ability to hot upgrade a running version of the server.

For example, to restart the server you can use the following command:

\$ sudo rstudio-server restart

Note that on some systems (including RedHat/CentOS 5 and SLES 11) the sudo utility doesn't have the /usr/sbin directory in it's path by default. For these systems you can use a full path to the management script. For example:

\$ sudo /usr/sbin/rstudio-server restart

## 4. Accessing the Server

#### 4.0.1. Logging In

By default RStudio runs on port 8787 and accepts connections from all remote clients. After installation you should therefore be able to navigate a web browser to the following address to access the server:

#### http://<server-ip>:8787

RStudio will prompt for a username and password and will authenticate access using the PAM authentication scheme configured for the server. Some notes related to user authentication:

- RStudio will not permit logins by system users (those with ids < 100).
- By default on Debian/Ubuntu the system default PAM profile (/etc/pam.d/other) will be used (this can be customized by creating an RStudio PAM profile at /etc/pam.d/rstudio).
- By default on RedHat/CentOS and SLES an RStudio PAM profile (/etc/pam.d/rstudio) that authenticates using the system username/password database will be used (this can be customized by editing the profile as appropriate).
- User credentials are encrypted using RSA as they travel over the network.

Additional details on customizing RStudio authentication are provided in Authenticating Users. Details on customizing the port and enabling SSL are covered in Access and Security.

#### 4.0.2. Troubleshooting Problems

If you are unable to access the server after installation, you should run the verify-installation command to output additional diagnostics:

#### \$ sudo rstudio-server verify-installation

This command will start the server and run and connect to an R session. Note that this will test the correct installation of RStudio and ensure that it can connect to a locally installed version of R. However, it won't test whether networking or authentication problems are preventing access to the server.

If problems persist, you can consult the system log to see if there are additional messages there. On Debian/Ubuntu systems this will typically be located at:

#### /var/log/syslog

On RedHat/CentOS systems this will typically be located at:

#### /var/log/messages

For RStudio Workbench, you can also consult the server log to see if there are any errors reported there. This log is located at:

/var/lib/rstudio-server/monitor/log/rstudio-server.log

# 5. Support PRO

#### **5.0.1. Community Resources**

RStudio provides a community website where open-source RStudio users can go to get help with issues related to RStudio Server, R, R code, and the various R packages available in the R ecosystem. If you have such an issue, please visit <a href="https://community.rstudio.com">https://community.rstudio.com</a> and post a message on the community forum, where other users and RStudio employees can help you.

#### 5.0.2. Professional Support

RStudio provides email support to our professional products customers to help troubleshoot issues and answer questions about the software. To focus on building the best products, RStudio does not provide installation or professional services. If you require such assistance, please refer to our certified partners. For more information about the support we provide, please see the Support Agreement.

RStudio support is limited to our professional products. If your question is related to the open-source RStudio Server, R, R code, or specific R packages, please visit https://community.rstudio.com.

If your question is related to RStudio Workbench, you can open a support ticket by:

1. Running the diagnostics script:

sudo rstudio-server run-diagnostics

Note

If you have not been able to download the software, skip this step.

- 2. Email support@rstudio.com with:
- Your name and company
- What you are trying to do
- What you have tried
- Any errors you have received
- The diagnostics file produced from step 1

# Part II. Server Management

# 6. Overview

### 6.1. Server Management

In this section, you'll configure core server settings, primarily via config files. This includes settings for Linux service configuration, specifying various directories, and instructions for basic server administration. It also includes setting up logging, the administrative dashboard, and automated crash reporting.

# 7. Core Administrative Tasks

#### 7.0.1. Configuration Files

RStudio uses several configuration files all located within the /etc/rstudio directory. Configuration files and folders include:

rserver.conf	Core server settings
rsession.conf	Settings related to individual R sessions
notifications.conf	Notifications to be delivered to user sessions Pro
logging.conf	Configuration of the logging system including logger type,
	location, and level
crash-handler.conf	Configuration of the crash handling system (see Automated
	Crash Reporting)
database.conf	Configuration of the database that RStudio will store data in
	(see Database)
ngnix.http.conf	Extra HTTP configuration for nginx Pro
ngnix.server.conf	Extra server configuration for nginx Pro
ngnix.site	Extra site configuration for nginx Pro
profiles	User and group resource limits Pro
r-versions	Manual specification of additional versions of R Pro
ip-rules	IP access rules (allow or deny groups of IP addresses) Pro
load-balancer	Load balancing configuration Pro
health-check	Template for content to return for server health checks Pro
google-accounts	Mappings from Google accounts to local accounts Pro
file-locks	Configuration for file locking
env-vars	Additional environment variables to set during server startup
$\log$ in.html	Custom HTML for login page
themes/	Custom editor color schemes
fonts/	Fonts for RStudio's R console and code editor
keybindings/	Custom IDE keybindings
snippets/	Editor snippet files
templates/	Default templates for new files created in the IDE
dictionaries/	Custom spelling languages and dictionaries

The rserver.conf and rsession.conf files are created by default during installation however

the other config files are optional so need to be created explicitly. It should be noted that the rsession.conf file must be readable by each RStudio user, so it should be given appropriate permissions (e.g. 644).

The notifications.conf and r-versions files are created, but their entries are commented out as an example.

Whenever making changes to configuration files you need to restart the server for them to take effect. You can do this using the **restart** command of the server management utility:

```
$ sudo rstudio-server restart
```

#### 7.0.2. Alternate Configuration File Location

RStudio can be instructed to use a directory other than /etc/rstudio for hosting configuration files using the XDG standard environment variable XDG\_CONFIG\_DIRS. This can be useful when running RStudio in a container and mounting configuration at runtime. It can also be helpful for setting up alternate configurations for testing or troubleshooting purposes without running the risk of corrupting a known-good production configuration.

For the example below, presume that you'd like RStudio's configuration to live in /mnt/config/rstudio.

#### **Create the Directory**

First, create the directory that needs to host configuration (this can of course be skipped when mounting). Make sure that the restudio-server service account can read content in this directory.

```
$ mkdir -p /mnt/config/rstudio
$ chmod 755 /mnt/config/rstudio
```

#### **Copy Configuration**

Presuming that you'd like to start with your existing configuration, copy all of the configuration files and folders from your existing configuration set to your new configuration. You can do this as follows:

```
$ cp -r /etc/rstudio/* /mnt/config/rstudio/
```

#### **Configure Service**

Because RStudio runs as a system service, you must use your system's service manager to change its environment. If your Linux distribution uses the systemd init system, run sudo systemctl edit rstudio-server. In the editor, add the following section to the file (replacing /mnt/config with your choice of root). Note that the rstudio folder is not included in this path; this is a configuration root directory that will be respected by other applications that use the XDG standard.

```
[Service]
Environment="XDG_CONFIG_DIRS=/mnt/config"
```

If you wish to set RStudio's configuration folder directly, use the RSTUDIO\_CONFIG\_DIR environment variable instead. For example, to use /mnt/config/rstudio as the configuration folder:

```
[Service]
Environment="RSTUDIO_CONFIG_DIR=/mnt/config/rstudio"
```

RSTUDIO\_CONFIG\_DIR is also useful if you do not wish other XDG-compliant applications to be affected by the environment variable. If set, it takes precedence over XDG\_CONFIG\_DIRS.

If your Linux distribution does not use the systemd init system, consult the documentation for your Linux distribution to learn which init system it uses and the appropriate method for setting environment variables for the rstudio-server service.

#### **Change and Restart**

Finally, make any configuration changes you'd like in your new configuration folder, and then restart the server to use the new configuration files.

```
$ sudo rstudio-server restart
```

To return to the configuration in /etc/rstudio, just remove the Environment directive added above and restart the service.

#### 7.0.3. Configuring the Run-Time Data Directory

RStudio needs to write several temporary files while running to function properly. The directory at which these files is written can be set by the server-data-dir configuration option by modifying /etc/rstudio/rserver.conf like so:

```
server-data-dir=/var/run/rstudio-server
```

The data directory defaults to /var/run/rstudio-server but you can change it to any directory. The specified location must be readable by any users of RStudio.

#### 7.0.4. Setting Environment Variables

You can set environment variables for RStudio's server process using the <code>env-vars</code> configuration file. This is an alternative to setting the environment variables using your system's service manager. For example, to set the <code>HTTP\_PROXY</code> and <code>XDG\_DATA\_HOME</code> environment variables for the server process:

```
# /etc/rstudio/env-vars
# Set proxy for outbound HTTP requests
HTTP_PROXY=http://192.168.1.1

# Store user data on mounted external storage
XDG_DATA_HOME=/mnt/storage/$USER
```

The env-vars file is reloaded, and the environment variables set again, when the server process receives a SIGHUP signal. See Reloading Configuration for an example.

#### Note

This technique cannot be used to set the specific environment variables XDG\_CONFIG\_DIRS or RSTUDIO\_CONFIG\_DIR, because those variables control where configuration files are loaded from, and env-vars is *itself* a configuration file. Use your system's service manage to set those variables as described in Alternate Configuration File Location.

#### Note

With the exception of XDG variables, environment variables set for the server process are not generally forwarded to individual R sessions. To set environment variables such as HTTP\_PROXY for all R sessions on the server, use Renviron.site or set them in one of the scripts executed when R sessions are initialized (see Profile Script Execution).

#### 7.0.5. Stopping and Starting

During installation RStudio is automatically registered as a daemon which starts along with the rest of the system. The exact nature of this will depend on the init system in use on your system:

- On systems using systemd (such as Debian 7, Ubuntu 15, and RedHat/CentOS 7), this registration is performed as a systemd script at /etc/systemd/system/rstudio-server.service.
- On systems using Upstart (such as older versions of Debian and Ubuntu, and RedHat/CentOS 6), this registration is performed using an Upstart script at /etc/init/rstudio-server.conf.
- On systems using init.d, including RedHat/CentOS 5, an init.d script is installed at /etc/init.d/rstudio-server.

To manually stop, start, and restart the server you use the following commands:

```
$ sudo rstudio-server stop
$ sudo rstudio-server start
$ sudo rstudio-server restart
```

To check the current stopped/started status of the server:

```
$ sudo rstudio-server status
```

#### 7.0.6. Reloading Configuration Values

To reload the server's configuration without restarting it, use the reload command:

```
$ sudo rstudio-server reload
```

Alternately, you can send a SIGHUP to the rserver process, using a command like kill -s SIGHUP \$PID, where \$PID is the process ID of the rserver process.

Note that most configuration values cannot be applied without a full restart. The following are the values and settings that will be reloaded when you send SIGHUP or execute the reload command:

- 1. Logging configuration (logging.conf), as described in Logging.
- 2. Environment variables (env-vars), as described in Setting Environment Variables.
- 3. Load balancing settings (load-balancer), as described in Load Balancing.
- 4. **nginx configuration** (nginx.\*.conf), as described in Customizing Default Proxy.
- 5. Custom R version settings (r-versions), as described in Extended R Version Definitions.
- 6. Product license data, as described in License Management.

#### 7.0.7. Managing Active Sessions

There are a number of administrative commands which allow you to see what sessions are active and request suspension of running sessions.

To list all currently active sessions:

```
$ sudo rstudio-server active-sessions
```

#### **Suspending Sessions**

When R sessions have been idle (no processing or user interaction) for a specified period of time (2 hours by default) RStudio suspends them to disk to free up server resources. When the user next interacts with their session it is restored from disk and the user resumes right back where they left off. This is all done seamlessly such that users aren't typically aware that a suspend and resume has occurred.

To manually suspend an individual session:

```
$ sudo rstudio-server suspend-session <pid>
```

To manually suspend all running sessions:

```
$ sudo rstudio-server suspend-all
```

The suspend commands also have a "force" variation which will send an interrupt to the session to request the termination of any running R command:

```
$ sudo rstudio-server force-suspend-session <pid>
$ sudo rstudio-server force-suspend-all
```

The force-suspend-all command should be issued immediately prior to any reboot so as to preserve the data and state of active R sessions across the restart.

#### Killing Sessions

If you are for any reason unable to cooperatively suspend an R session using the commands described above you may need to force kill the session. Force killing a session results in SIGKILL being sent to the process, causing an immediate termination.

To force kill an individual session:

#### \$ sudo rstudio-server kill-session <pid>

To force kill all running sessions:

#### \$ sudo rstudio-server kill-all

Note that these commands should be exclusively reserved for situations where suspending doesn't work as force killing a session can cause user data loss (e.g. unsaved source files or R workspace content).

#### 7.0.8. Listing users

You can list all of the users that have signed in to RStudio Workbench by running the following command:

```
$ sudo rstudio-server list-users
```

#### 7.0.9. Adding users to the user database

Normally, when users sign in to RStudio Workbench for the first time, they are automatically added to the RStudio user database. However, you can manually add them via script if you need to. This is useful for designating specific users as administrators to allow them access to the administration panel without requiring them to belong to the administration group. This can be done by running the following command:

```
$ sudo rstudio-server add-user <username> <0 or 1>
```

Specifying a 0 in the above command will add the user without admin privilege; specify a 1 to grant admin privilege.

#### i Note

Users set as administrators in this way will be Admin Superusers. See Administrator Superusers.

#### 7.0.10. Changing the admin status of a user

You can also change the admin status of a particular existing user:

```
$ sudo rstudio-server set-admin <username> <0 or 1>
```

#### 7.0.11. Locking and unlocking users

If you are using RStudio Workbench under a named user license, you can lock and unlock specific users to disallow or reallow access to the server. Locking a user will prevent them from signing in to RStudio, but will preserve their files. Locked users do not count against the named user limit on your license.

This feature may be used when a user leaves the organization or otherwise no longer needs access to RStudio Workbench on a permanent basis.

To lock a user, click on the user on the *Users* admin page, and then click the *Lock* button in the upper right-hand corner of the screen.

You can also lock users via the server management utility:

```
$ sudo rstudio-server lock-user <username>
$ sudo rstudio-server unlock-user <username>
```

Note that RStudio's Software License Descriptions only allow this feature to be used to permanently terminate a named user's access.

#### 7.0.12. Taking the Server Offline

If you need to perform system maintenance and want users to receive a friendly message indicating the server is offline you can issue the following command:

```
$ sudo rstudio-server offline
```

When the server is once again available you should issue this command:

```
$ sudo rstudio-server online
```

When the server is taken offline all R sessions will be suspended and no new sessions can be started. Session data will be saved but any running computations will be terminated.

#### 7.0.13. Upgrading to a New Version

If you perform an upgrade of RStudio and an existing version of the service is currently running, then the upgrade process will also ensure that active sessions are immediately migrated to the new version. This includes the following behavior:

- Running R sessions are suspended so that future interactions with the server automatically launch the updated R session binary
- Currently connected browser clients are notified that a new version is available and automatically refresh themselves.
- The core server binary is restarted

When load balancing is configured upgrading multiple nodes may cause brief glitches if you upgrade each server one at a time. This is due to the possibility of two servers with different versions trying to coordinate. If some downtime is acceptable, we recommend taking all nodes offline before upgrading.

To upgrade to a new version of RStudio you simply install the new version:

#### Debian/Ubuntu

```
$ sudo gdebi <rstudio-package.deb>
```

#### RedHat/CentOS:

```
$ sudo yum install <rstudio-package.rpm>
```

#### openSUSE/SLES:

```
$ sudo zypper install <rstudio-package.rpm>
```

# 8. Logging

RStudio provides the ability to change various facets of its logging functionality, such as changing the logger type (syslog vs file) on a per-binary or per-section basis. In most configurations, it is recommended that you do not change the logging configuration, but in some circumstances it is desirable to turn on debug logging to help troubleshoot issues.

#### 8.0.1. Configuration File

The logging configuration file is located at /etc/rstudio/logging.conf. It allows you to specify logging options in a hierarchy of three different types:

- 1) Global ([\*])
- 2) Per-binary ([@binary name])
- 3) Per log section ([log section name])

For example:

```
# /etc/rstudio/logging.conf
[*]
log-level=info
logger-type=syslog

[@rserver]
log-level=debug
logger-type=file
max-size-mb=10

[file-locking]
log-dir=/var/log/file-locking
log-file-mode=600
```

This configuration specifies that by default, all logging should be sent to syslog at info level or higher. Debug logging is enabled for the rserver binary specifically, where logs are written to file with a maximum log file size of 10MB, in the default /var/log/rstudio-server directory. File locking logging is also specifically redirected to a different logging directory,

/var/log/file-locking, with a specific file creation mode of 600. All logging sections named file-locking which occur in the rserver process will be logged to file, whereas the same logging sections in other binaries will continue to be logged to syslog.

The following settings may be specified in /etc/rstudio/logging.conf:

Config Option	Description	Default Value
log-level	The minimum log level to capture. Can be one of debug,	warn
logger-type	info, warn, or error.  The type of logger to use. Can be one of stderr, syslog, or file.	syslog

When using the file logger, the following can also be specified:

Config Option	Description	Default Value
log-dir	The log directory to store log files in. The resulting log file name is based on the binary name (and optionally the process ID).	/var/log/rstudio-server
log-file-mode	The filemode to use when creating the log file. Must be a valid POSIX octal file mode.	666 (All read/write)
log-file-	Whether to include the process id in the filename. Useful	0 (false)
include-pid	for differentiating between multiple processes of the same name. Can be 0 (false) or 1 (true)	
rotate	Whether or not to rotate the log file as it reaches maximum size. Can be 0 (false) or 1 (true)	1 (true)
max-size-mb	Maximum allowable size of the file before it is rotated. Only applicable if rotate is enabled.	2 (MB)

#### List of Logging Sections

The following is a list of logging sections that may be used within the logging configuration file.

Section Name	Description
file-locking	Logging of file locking operations, such as processes acquiring and releasing locks.

#### **Reloading Configuration**

In order to reload the logging settings during run-time of a process, simply send the SIGHUP signal to the desired process, and the logging configuration will be reloaded for that binary.

For example, to send the  $\mathtt{SIGHUP}$  signal to the  $\mathtt{rserver}$  process after changing the configuration file:

pidof rserver | sudo xargs kill -s SIGHUP

# 9. Administrative Dashboard PRO

RStudio Workbench includes an administrative dashboard with the following features:

- 1) Monitoring of active sessions and their CPU and memory utilization;
- 2) The ability to suspend, forcibly terminate, or assume control of any active session;
- 3) Historical usage data for individual server users (session time, memory, CPU, logs);
- 4) Historical server statistics (CPU, memory, active sessions, system load); and
- 5) Searchable server log (view all messages or just those for individual users)
- 6) The ability to lock users, preventing them from signing in to RStudio

The dashboard can be an invaluable tool in understanding server usage and capacity as well as to diagnose and resolve problems.

Note that at this time, historical monitoring is not available for sessions spawned via the Job Launcher.

#### 9.0.1. Enabling the Dashboard

The administrative dashboard is accessed at the following URL:

```
http://<server-address>/admin
```

The administrative dashboard is disabled by default. To enable it you set the admin-enabled option. You can also specify that only users of certain group have access to the dashboard using the admin-group option. For example:

```
# /etc/rstudio/rserver.conf
admin-enabled=1
admin-group=rstudio-admins
```

You can specify a single group as the above example does or a comma-delimited list of groups. For example:

```
# /etc/rstudio/rserver.conf
admin-group=server-admins,rstudio-admins,domain-admins
```

Note that changes to the configuration will not take effect until the server is restarted.

#### 9.0.2. Administrator Superusers

You can further designate a certain user or group of users as administrative "superusers". Superusers have the following additional privileges:

- 1) Suspend or terminate active sessions
- 2) Assume control of active sessions (e.g. for troubleshooting)
- 3) Login to RStudio as any other server user

Administrative superusers do not have root privilege on the system, but rather have a narrow set of delegated privileges that are useful in managing and supporting the server. You can define the users with this privilege using the admin-superuser-group setting. For example:

```
# /etc/rstudio/rserver.conf
admin-superuser-group=rstudio-superuser-admins
```

Note that as with the admin groups above, you can specify a single group as the above example does or a comma-delimited list of groups. For example:

```
# /etc/rstudio/rserver.conf
admin-superuser-group=rstudio-superuser-admins,domain-admins
```

Changes to the configuration will not take effect until the server is restarted. Admin superusers can also be added via the command line. See Adding users to the user database.

#### **User Impersonation Restrictions**

Note that the ability to login as other users and assume control of existing sessions is not available if you are authenticating with SAML SSO, Google Accounts or proxied authentication. This is because these authentication mechanisms use a different user-identity mechanism which isn't compatible with the way that user session impersonation is implemented.

#### 9.0.3. Server Log Time Zone

You can control the time zone in which the server logs are displayed in the admin dashboard by the use of the admin-monitor-log-use-server-time-zone option. For example:

```
# /etc/rstudio/rserver.conf
admin-monitor-log-use-server-time-zone=1
```

Setting this option to 1 will display the server logs in the server's time zone. The default value of 0 will display the log times in UTC.

#### 9.0.4. Licensing Considerations

If you have been granted a license which has a limit on the total number of users that may use RStudio (i.e. named user licensing), you will need to control user access to the server. Each unique user that signs in to RStudio will count against your available user limit. If too many users sign in and attempt to use the system, new users will be denied, as the license limit will be reached.

If this occurs, please contact sales@rstudio.com to purchase additional users.

For more information on licensing, see License Management.

# 10. Automated Crash Reporting

RStudio allows you to automatically capture crash dumps for all RStudio processes and upload them to our crash database for easy reporting, allowing us to more easily determine the exact cause of crashes without adding administrative overhead to your team by requiring you to manually collect crash dumps. As user privacy is of utmost concern, these crash dumps **only** contain stack information, such as the names of the functions that were on the callstack at the time of the crash. **No** heap information is stored within the dumps, so no sensitive user data is captured in the crash dumps. The IP address of the machine uploading the reports is captured and used **only** to determine the unique amount of users experiencing a crash.

#### 10.0.1. Crash Handler Configuration

A separate RStudio process is responsible for handling crashes, and may be configured similar to other RStudio processes by modifying the config file /etc/rstudio/crash-handler.conf. Automated crash handling is disabled by default, though we recommend that you enable it. The following shows a table of available configuration options for /etc/rstudio/crash-handler.conf:

Config		Require	ed
Option	Description	(Y/N)	Default Value
crash- handling- enabled	Enables/disables automatic capturing of crash dumps for all RStudio processes.	N	0 (disabled)
enabled crash-db- path	Location of the path where crash dumps should be stored on disk. This path must be readable and writeable by all RStudio users.	N	/tmp/crashpad_databas
uploads- enabled	Enables/disables automatic uploading of crash dumps to our crash reporting service. If this is disabled, you will have to manually upload crash dumps.	N	1 (enabled)
upload-url	Web URL where crash reports are uploaded. You should likely not change this setting unless RStudio Support instructs you to do so.	N	

Config		Require	<del>1</del>
Option	Description	(Y/N)	Default Value
upload- proxy	Proxy server to connect to when submitting the minidump. Only applicable on Linux - uses the default system-wide setting on OSX. If left blank, any system-wide setting specified will be used. This should be in the form of [scheme]://[host]:[port], where scheme may be one of the following: http https://schemesch	<b>N</b> ml	

For most RStudio installations, it is sufficient to merely set crash-handling-enabled=1 in the config file , like so:

```
# /etc/rstudio/crash-handler.conf
crash-handling-enabled=1
```

#### 10.0.2. Manually Uploading Crash Dumps

In some cases, you may be unable to automatically upload crash dumps to the crash reporting service because of lack of internet connectivity or simply because you want to manually upload crash dumps. RStudio allows you to manually upload crash dumps at any time by invoking the following command:

rstudio-server upload-minidump /path/to/minidump

# Part III. **Authenticating Users**

# 11. Overview

#### 11.1. Authenticating Users

R users require local system accounts regardless of what RStudio authentication method you use. You must set up local system accounts manually or programmatically and then map authenticating users to these accounts.

For user identification, authentication, and authorization using local system accounts, RStudio relies heavily on Linux Pluggable Authentication Module (PAM). PAM can be used by itself to authenticate users or along with other external authentication mechanisms (e.g., Web Single Sign-On) to authorize existing local system accounts.

#### i Note

Not all RStudio products require local system accounts or PAM. For example, RStudio Connect and Shiny Server rely on their own authentication engines and on a single system account for doing their work in most cases, not requiring individualized development environments like the ones offered by RStudio Workbench.

Here are the various authentication mechanisms supported by RStudio:

Authentication	RStudio Configuration
Local Accounts	PAM Authentication (via pam_unix)
LDAP or Active Directory	PAM Authentication (via pam_sss or pam_ldap in older systems)
Kerberos	PAM Authentication (via pam_sss or pam_krb5 in older systems)
Web Single Sign-On (SSO)	SAML Single Sign-On Authentication or OpenID Connect Authentication
Others (client-server, e.g., RADIUS)	As supported by various PAM modules
Others (browser-based, e.g., Kerberos SPNEGO SSO)	Proxied Authentication

### i Note

SAML, OpenID, and Proxied authentication still require PAM Sessions to automatically create local system accounts. Without it, local system accounts have to be provisioned manually one-by-one.

# 12. Customizing the Sign-In Page

You can customize the content and appearance of the RStudio sign-in page by including custom HTML within the page. This is accomplished by either:

- Providing a file at /etc/rstudio/login.html that includes additional HTML to include within the login page; or
- Specifying the auth-login-page-html option within the rserver.conf config file which points to an alternate location for the login HTML file. For example, the following specifies that the file located at /opt/config/rstudio-login.html should be included within the login page:

```
# /etc/rstudio/rserver.conf
auth-login-page-html=/opt/config/rstudio-login.html
```

The contents of the specified HTML file will be included after the standard login header and login username/password form. If you want to modify the appearance of the header and/or add content above the username/password form, you can use CSS and JavaScript within your login.html file to modify the page after it loads.

The same can be done to tweak the display of the sign in page for remote RDP sessions as well, with either the /etc/rstudio/rdplogin.html file, or the auth-rdp-login-page-html setting within rserver.conf.

# 13. User Provisioning

It is a common practice in Linux environments to configure **sssd** to fetch users and groups from an LDAP or Active Directory server to automate the creation (provisioning) of local system accounts.

In addition to user creation, sssd can also be configured to authenticate or authorize users via PAM using the pam\_sss module.

#### Note

nss is an older alternative to sssd that also has LDAP synchronization capabilities. However, differently from sssd, it offers no support for authentication.

#### ! Important

When a user has an active session in RStudio, changes to his or her local account name (username) or uid are not supported and it can lead to unexpected behaviors in RStudio.

# 14. Restricting Access to Specific Users

#### 14.0.1. Minimum User Id

By default RStudio only allows normal (as opposed to system) users to successfully authenticate. The minimum user id is determined by reading the UID\_MIN value from the /etc/login.defs file. If the file doesn't exist or UID\_MIN isn't defined within it then a default value of 1000 is used.

You change the minimum user id by specifying the auth-minimum-user-id option. For example:

```
# /etc/rstudio/rserver.conf
auth-minimum-user-id=100
```

Note that it's possible that your PAM configuration is also applying a constraint on user-ids (see the Default PAM Configuration section above for an example). In this case you should ensure that the auth-minimum-user-id is consistent with the value specified in your PAM configuration.

If your users are using very large UIDs (above 1048575/0xFFFFF), it is *strongly* recommended to set the auth-minimum-user-id value to enable RStudio to make better assumptions when mapping user identifiers to projects.

#### 14.0.2. Restricting by Group

You can specify that only users of certain groups are allowed to access RStudio. To do this you use the auth-required-user-group setting. For example:

```
# /etc/rstudio/rserver.conf
auth-required-user-group=rstudio-users
```

You can specify a single group as the above example does or a comma-delimited list of groups. For example:

```
# /etc/rstudio/rserver.conf
auth-required-user-group=analysts,admins,rstudio-users
```

Note that this change will not take effect until the server is restarted.

#### Creating and Managing Group Membership

To create a new group you use the groupadd command:

```
$ sudo groupadd <groupname>
```

To add a user to an existing group you use the usermod command:

```
$ sudo usermod -a -G <groupname> <username>
```

Note that it's critical that you include the -a flag as that indicates that the group should be added to the user rather than replace the user's group list in it's entirety.

## 15. Authorization Timeout

#### 15.0.1. Inactivity Timeout

By default, user authorization will expire after 60 minutes of inactivity, requiring the user to sign in again to continue using their sessions. This is configurable by changing the auth-timeout-minutes setting in /etc/rstudio/rserver.conf. For example, to sign users out after 20 minutes of inactivity instead:

```
# /etc/rstudio/rserver.conf
auth-timeout-minutes=20
```

This setting supersedes the auth-stay-signed-in-days setting discussed below in Stay Signed In, as it provides additional security by ensuring that users that are not actively using the system do not stay signed in. However, if you wish to allow users to stay signed in for many days at a time and disable the authorization timeout completely, set auth-timeout-minutes to 0, which will cause the auth-stay-signed-in-days setting's behavior to be used instead.

Whenever a user is signed out, the database will be updated to store recently logged out/invalidated cookies, preventing the use of credential replay attacks.

#### 15.0.2. Stay Signed In

Users have an option to stay signed in across browser sessions when using PAM or Google authentication methods. By default when choosing the stay signed in option users will remain signed in for 30 days. You can modify this behavior using the auth-stay-signed-in-days setting. For example:

```
# /etc/rstudio/rserver.conf
auth-stay-signed-in-days=7
```



This setting is deprecated in favor of the auth-timeout-minutes setting for additional security. See Inactivity Timeout above for more information.

Note that for this setting to take effect, auth-timeout-minutes must be set to 0.

If you want to prevent users from being able to stay signed in, you can prevent the "Stay Signed In" option from being shown to them by using the auth-stay-signed-in setting. For example:

```
# /etc/rstudio/rserver.conf
auth-stay-signed-in=0
```

Setting this option to 0 will result in users being prompted to log in each time they start a new browser session (i.e., logins will only be valid as long as the browser process in which they originated in remains running).

## 16. PAM Authentication

#### 16.0.1. PAM Basics

Local system accounts used by PAM must follow the system conventions for usernames. For example, usernames are case-sensitive.

PAM profiles are located in the /etc/pam.d directory. Each application can have their own profile, and there is also a default profile used for applications without one (the default profile is handled differently depending on which version of Linux you are running).

To learn more about PAM and the many options and modules available for it see the following:

- https://access.redhat.com/documentation/en-us/red\_hat\_enterprise\_linux/7/html/sy stem-level\_authentication\_guide/pluggable\_authentication\_modules
- http://tldp.org/HOWTO/User-Authentication-HOWTO/x115.html
- http://linux.die.net/man/8/pam

#### 16.0.2. PAM & Provisioning

PAM recommendations for user provisioning:

- If you are using PAM authentication to connect to an external authentication provider, you can use a PAM module like pam\_mkhomedir to automatically create the users' home directories on login.
- User provisioning requires PAM sessions to be used is most cases. See the PAM Sessions section for details on how RStudio can be configured to use those.
- If you are using SAML, OpenID or Proxied authentication and local system accounts are automatically created by sssd, a PAM configuration with the module pam\_rootok must be used, since PAM sessions will be started by RStudio using root in the behalf of the actual user.

#### 16.0.3. PAM Authentication

By default, RStudio authenticates users via the Linux standard PAM API. PAM itself is typically configured by default to authenticate against the system user database (/etc/passwd) however it can also be configured to authenticate against a wide variety of other systems including Active Directory, LDAP, and Kerberos.

#### Note

PAM can be used for both authentication as well as to tailor the environment for users with PAM sessions. This section only describes PAM for the purposes of authentication.

#### 16.0.4. Default PAM Configuration

#### Debian / Ubuntu

On Debian and Ubuntu systems RStudio does not provide an RStudio specific PAM configuration file. As a result, RStudio uses the /etc/pam.d/other profile, which by default inherits from a set of common configuration files:

```
/* /etc/pam.d/other */
  @include common-auth
  @include common-account
  @include common-password
  @include common-session
```

If the /etc/pam.d/other profile reflects the authentication system and policies that you'd like RStudio to use then no further configuration is required. If you want to create a custom PAM profile for RStudio you would create a file named /etc/pam.d/rstudio and specify whatever settings are appropriate.

#### RedHat / CentOS / SUSE

On RedHat, CentOS and SUSE systems applications without their own PAM profiles are denied access by default. Therefore to ensure that RStudio is running and available after installation a default PAM profile is installed at /etc/pam.d/rstudio. This profile is configured to require a user-id greater than 500 and to authenticate users against local system accounts:

```
; /etc/pam.d/rstudio
auth requisite pam_succeed_if.so uid >= 500 quiet
auth required pam_unix.so nodelay
account required pam_unix.so
```

This default PAM profile may not reflect the authentication behavior that you want for RStudio. In that case, some customization may be required. If you've already set up another PAM profile (e.g. /etc/pam.d/login) with the desired behavior then it may be enough to simply copy that profile over the RStudio one. For example:

\$ sudo cp /etc/pam.d/login /etc/pam.d/rstudio

#### 16.0.5. Diagnosing PAM Authentication Problems

If you are unable to log into RStudio there may be an underlying problem with the PAM configuration. The best way to diagnose PAM configuration problems is to use the pamtester utility (which is bundled with RStudio). Using pamtester enables you to test authentication in an isolated environment as well as to see much more detailed diagnostics.

The pamtester utility is located at /usr/lib/rstudio-server/bin/pamtester. To invoke it you pass several arguments indicating the PAM profile to test, the user to test for, and whether you want verbose output. For example:

\$ sudo /usr/lib/rstudio-server/bin/pamtester --verbose rstudio <username> authenticate

You can find more detailed documentation on using pamtester here: http://linux.die.net/man/1/pamtester.

Additionally, RStudio expects the PAM password prompt text when logging in to be Password:. If you use a nonstandard password prompt, you must disable strict prompt matching by modifying /etc/rstudio/rserver.conf like so:

auth-pam-require-password-prompt=0

## 17. SAML Single Sign-On Authentication PRO

RStudio Workbench can be configured to authenticate users via SAML. This enables users to log in with their existing Single Sign-On (SSO) credentials and to be automatically authenticated to RStudio whenever they are already logged into their Identity Provider (IdP) account.

#### 17.0.1. Enabling SAML SSO

To enable authentication with SAML you add the auth-saml option to the RStudio configuration file /etc/rstudio/rserver.conf:

# /etc/rstudio/rserver.conf
auth-saml=1

#### ! Important

Once you enable authentication with SAML that becomes the exclusive means of authentication - you can't concurrently use both PAM and SAML authentication.

#### ! Important

SAML authentication still requires PAM Sessions and sssd to automatically create local system accounts. Without them, local system accounts have to be provisioned manually one-by-one.

#### 17.0.2. Configuring SAML

To use SAML authentication in RStudio you should configure the parts involved in this order:

- Configuring RStudio as a Service Provider
- Configuring your Identity Provider with RStudio

#### 17.0.3. Configuring RStudio as a Service Provider

RStudio Workbench needs to be configured in order to trust the SAML assertions sent by your IdP.

#### **Identity Provider Metadata Setup**

#### Note

RStudio Workbench expects the IdP metadata to contain the service name (EntityID), the Single Sign-On (SSO) URL and the signing certificate.

The easiest way to configure RStudio Workbench is to point it to the IdP metadata URL. This can be done by adding the option auth-saml-metadata-url. This option automatically renews the metadata when it expires but it requires direct connectivity between your server and the IdP.

```
# /etc/rstudio/rserver.conf
auth-saml-metadata-url=https://idp.example.com/saml/metadata
```

If you want to avoid direct connectivity between your server and the IdP or if that setup is not possible you should use an offline setup. In this scenario, you should download the metadata from your IdP and upload it to your server. Then, add the option auth-saml-metadata-path pointing to the file location within your server. This option requires manual intervention if the metadata or the signing certificate expires.

```
# /etc/rstudio/rserver.conf
auth-saml-metadata-path=/path/to/saml/metadata.xml
```

#### Warning

The metadata URL option has precedence over the metadata file path option. You must remove the URL option first before using the file option.

If your Identity Provider requires information about RStudio in order to provide metadata, see the Manual Service Provider Setup section below for a list of details you may need to provide about your RStudio Workbench server.

#### Manual Identity Provider Setup

If your IdP does not provide metadata or if the metadata has any problems, you should use a manual setup. The following required options must be added to your server configuration:

- auth-saml-idp-entity-id An URL to an HTTP(S) endpoint on the IdP, in general the location of its metadata. In very exceptional cases this may not be an URL.
- auth-saml-idp-sso-url An URL to an HTTP(S) endpoint on the IdP to where your server will send authentication requests.
- auth-saml-idp-sign-cert-path The path to a PEM file containing the public trust certificate for verifying the assertions' signatures.

```
# /etc/rstudio/rserver.conf
auth-saml-idp-entity-id=https://idp.example.com/saml/metadata
auth-saml-idp-sso-url=https://idp.example.com/saml/sso
auth-saml-idp-sign-cert-path=/path/to/saml.cert
```

#### Warning

The metadata URL and file path options have precedence over the individually configured options. You must remove the metadata options first before using individual settings.

#### 17.0.4. Configuring your Identity Provider with RStudio

In order to use SAML with RStudio Workbench you also need to register your server with your IdP.

#### **SAML Attributes**

#### Important

Your IdP must return in the assertion at least one attribute (or NameID) matching the user's local account username (lowercase).

By default, RStudio Workbench will look for an attribute called Username (case-sensitive). If you wish to use a different attribute or the assertion's NameID value, add the option auth-saml-sp-attribute-username with the appropriate value.

```
# /etc/rstudio/rserver.conf
auth-saml-sp-attribute-username=NameID
```

#### **Preconfigured Setups**

RStudio Workbench has preconfigured entries in Okta, OneLogin and Azure. In some cases, all you need to provide is the URL to your server. Please refer to the documentation on these vendors for more information.

#### Service Provider Metadata Setup

RStudio Workbench provides its own Service Provider (SP) SAML metadata at the /saml/metadata endpoint. For example, if your server is running at https://server.example.com, its metadata can be found at https://server.example.com/saml/metadata.

Your IdP may ask for the metadata URL or the metadata file. For the latter, you should download the metadata file and upload it to your IdP.

#### Manual Service Provider Setup

If your IdP requires a manual configuration, the basic information about your server is in the SAML metadata.

#### Note

The SAML metadata primarily contains information about the service name (EntityID) and the assertion consumer service (ACS) URL.

If you cannot start RStudio configured with SAML because your IdP must be configured first, you may be asked the following information:

- RStudio Entity ID: This value is the same URL as the metadata endpoint used for Service Provider Metadata Setup. For example, https://server.example.com/saml/metadata
- RStudio Assertion Consumer Service URL: RStudio expects SAML assertions at the /saml/acs endpoint. For example, https://server.example.com/saml/acs
- If encryption is used, you will need to provide the encryption certificate used by RStudio, see SAML Encryption below. RStudio supports most common forms of encryption used with SAML.
- If your Identity Provider expects signed requests from RStudio, you will need to provide the signing certificate used by RStudio, see SAML Request Signing below. Any signing algorithm you choose in your IdP must match RStudio's configuration.
- SAML Attributes as mentioned above.
- Also, information requested about Unsupported SAML Options in RStudio should be left blank.

#### 17.0.5. Advanced Options

Depending on your IdP capabilities you may need to add a few more options to your server:

- auth-saml-idp-post-binding: By default RStudio Workbench will redirect to your IdP for authentication requests. With the value 1, this option makes it use an HTTP POST instead. This option can also be used with a metadata file or URL if your IdP supports both redirect and POST.
- auth-saml-sso-initiation: By default RStudio Workbench will be able to initiate an authentication with SAML (SP-initiated) or to accept an ad-hoc assertion (IdP-initiated). If you prefer just one of these flows, use this options with either sp or idp values. When set to idp users will be sent to the configured IdP SSO URL if a SP-initiated flow is attempted.
- auth-saml-sp-name-id-format: By default RStudio Workbench will accept any NameID Format. Add this options with the values persistent, transient, emailaddress, or unspecified to make RStudio Workbench request and expect a particular format from the IdP.



#### Warning

auth-saml-sp-name-id-format=transient and auth-saml-sp-attribute-username=NameID will not be accepted as a valid combination. It would lead to undetermined usernames in each attempt.

Here are some examples of valid configurations of the aforementioned advanced options:

```
# /etc/rstudio/rserver.conf
auth-saml-idp-post-binding=1
auth-saml-idp-sso-url=https://idp.example.com/saml/sso
# /etc/rstudio/rserver.conf
auth-saml-sso-initiation=idp
auth-saml-idp-sso-url=https://idp.example.com/login
# /etc/rstudio/rserver.conf
auth-saml-sp-name-id-format=persistent
```

#### 17.0.6. SAML Encryption

To enable support for encrypted SAML assertions, you will need a key pair in the form of a public certificate file and a private RSA key, both in PEM format.

The following options should be added to your server:

- auth-saml-sp-encryption-key-path: The path to a PEM file containing the private RSA key for decrypting the assertion.
- auth-saml-sp-encryption-cert-path: The path to a PEM file containing the public certificate for encrypting the assertion. The contents of this file will be present in your server metadata after configured. You may also be asked to upload this certificate to the IdP instead.

```
# /etc/rstudio/rserver.conf
auth-saml-sp-encryption-key-path=/path/to/saml.key
auth-saml-sp-encryption-cert-path=/path/to/saml.cert
```

#### Warning

These key pair files are similar to the ones used for SSL/TLS. However, for security reasons you must never use your server's own SSL/TLS key and certificate for SAML encryption.

This example allows the creation of a simple self-signed public certificate and private key pair that can be used for encryption for the server "localhost" (you should use your server public facing hostname instead):

openssl req -x509 -newkey rsa:2048 -keyout saml.key -out saml.cert -days 365 -nodes -subj "/

#### 17.0.7. SAML Request Signing

#### Note

In most situations, SAML authentication request signing is not required or even supported. Be sure your IdP requires signing before using this functionality.

To enable support for signed SAML authentication requests, you need to set a signing method in your server configuration with the option auth-saml-sp-request-signing-method. The algorithms sha1, sha256, or sha512 are supported. When in doubt, try sha256 first which offers a good balance between security and compatibility.

```
# /etc/rstudio/rserver.conf
auth-saml-sp-request-signing-method=sha256
```

#### i Note

By using one of the signing methods listed above the RStudio metadata will contain a "signing" certificate and will have the attribute AuthnRequestsSigned with the true value.

You will also need a key pair in the form of a public certificate file and a private RSA key, both in PEM format. If you are using SAML Encryption, the already configured encryption key pair will also be used for request signing.

#### ! Important

RStudio only accepts a single key pair. You can use a key pair for **both** encryption and signing requests or one just for signing. Different key pairs for signing and encryption are not currently supported.

If you are not currently using SAML encryption, the following options should be added to your server:

- auth-saml-sp-signing-key-path The path to a PEM file containing the private RSA key for decrypting the assertion.
- auth-saml-sp-signing-cert-path The path to a PEM file containing the public certificate for encrypting the assertion. The contents of this file will be present in your server metadata after configured. You may also be asked to upload this certificate to the IdP instead.

```
# /etc/rstudio/rserver.conf
auth-saml-sp-signing-key-path=/path/to/signing.key
auth-saml-sp-signing-cert-path=/path/to/signing.cert
```

## **A** Warning

These key pair files are similar to the ones used for SSL/TLS. However, for security reasons you must never use your server's own SSL/TLS key and certificate for SAML encryption.

This example allows the creation of a simple self-signed public certificate and private key pair that can be used for encryption for the server "localhost" (you should use your server public facing hostname instead):

openssl req -x509 -newkey rsa:2048 -keyout saml.key -out saml.cert -days 365 -nodes -subj "/

#### 17.0.8. Unsupported SAML Options

RStudio Workbench supports at least a subset of SAML called Interoperable SAML. Notably, certain functionalities are currently absent:

- Single Logout
- Certificate chain validation
- RelayState URL handling (not part of the SAML standard)

#### 17.0.9. Proxy Considerations

If you are running RStudio behind a proxy, you will need to configure your proxy in a way that RStudio can tell the SAML IdP to redirect back to the correct location. There are number of options to choose from as described in Running with a Proxy.

The use of the X-RStudio-Request header in your proxy is recommended and the only method which works behind a path-rewriting proxy. In this case, the proxy must set the X-RStudio-Request header to the exact complete URL as requested by the browser. For example if your proxy was set up to serve RStudio requests at https://testdomain.com/rstudio/and an incoming request for /home came in, your proxy should set X-RStudio-Request: https://testdomain.com/rstudio/home which would allow RStudio to know about the added path prefix /rstudio.

If your proxy does not add path prefixes, RStudio is also compatible with two options using commonly available HTTP proxy headers:

- The headers X-Forwarded-Host, X-Forwarded-Proto, and X-Forwarded-Port.
- Or the header Forwarded with host, and proto values.

When using path-rewriting proxies, it's also recommended to use either the header X-RStudio-Root-Path or the option www-root-path to indicate the path defined for RStudio by the proxy. For example, if your URL to RStudio is www.example.com/rstudio your proxy should send the header X-RStudio-Root-Path: /rstudio or you should use:

```
# /etc/rstudio/rserver.conf
www-root-path=/rstudio
```

If none of the headers above are set by the proxy, RStudio will attempt to redirect back to the address present in the Host header and it will determine the protocol (http or https) based on its current configuration.

If you are running behind a proxy but cannot set headers for whatever reason, and www-root-path is not the right choice, you can use the auth-sp-saml-base-uri option in the RStudio configuration file to accomplish the same purpose:

```
# /etc/rstudio/rserver.conf
auth-saml-sp-base-uri=http://testdomain.com/rstudio/
```

#### 17.0.10. Load Balancing Considerations

Because RStudio stores SAML authentication context in server memory during the authentication flow, the entire authentication flow must be completed on a single server. If you're using an external load balancer in front of RStudio, you will experience authentication errors if the HTTP requests associated with the authentication flow are not all routed to the same server.

For this reason, you **must enable sticky sessions** in your external load balancer when using SAML authentication. This feature is sometimes called "sticky cookies" or "session affinity". Consult the documentation for your load balancing software for details; for example if you're using the Amazon Web Services Application Load Balancer (AWS ALB), more information can be found in Sticky Sessions for your Application Load Balancer.

#### 17.0.11. Outgoing Proxies

Some SAML authentication features require RStudio to make a call to an external service over HTTP or HTTPS; for example, to perform provider metadata discovery. If your environment requires an HTTP or HTTPS proxy for outbound requests, you must set the appropriate proxy environment variables for RStudio's server process so that it uses the proxy when making the request.

One way to do this is to add the variables to the env-vars file as follows:

```
# /etc/rstudio/env-vars
HTTP_PROXY=http://192.168.1.1:8080
HTTPS_PROXY=http://192.168.1.1:8080
NO_PROXY=localhost,192.168.1.10
```

#### 17.0.12. Troubleshooting

Additional information about the SAML flow and the received assertion may be written to the logs. Be sure to configure rserver logs to output info level messages in /etc/rstudio/logging.conf to see these entries.

## 18. OpenID Connect Authentication PRO

RStudio Workbench can be configured to authenticate users via OpenID Connect. This enables users to log in with their existing Single Sign-On (SSO) credentials and to be automatically authenticated to RStudio whenever they are already logged into their OpenID Provider (OP) account.

#### 18.0.1. Enabling OpenID Connect

To enable authentication with OpenID you add the auth-openid option to the RStudio configuration file:

# /etc/rstudio/rserver.conf
auth-openid=1

#### Important

Once you enable authentication with OpenID, that becomes the exclusive means of authentication (you can't concurrently use both PAM and OpenID authentication).

#### ! Important

OpenID authentication still requires PAM Sessions and sssd to automatically create local system accounts. Without them, local system accounts have to be provisioned manually one-by-one.

## 18.0.2. Configuring your OpenID Provider

In order to use OpenID with RStudio Workbench you need to register your server on your OP first.

#### Note

RStudio Workbench supports the OAuth2 "Authorization Code" flow with OpenID.

#### **OpenID URLs**

RStudio Workbench expects an OAuth2 callback in order to complete the OpenID authentication. The endpoint responsible for handling this callback from the OP is located at /openid/callback. For example, if your RStudio Workbench server is hosted at https://rstudio.example.com/the callback URL will be https://rstudio.example.com/openid/callback. While registering RStudio Workbench on your OpenID Provider, this URL will be requested. Your server URL may also be requested as part of that setup.

#### **Client Credentials**

As part of the RStudio Workbench registration on your OP, a "client ID" and a "client secret" may be presented to you. You need to add a configuration file (/etc/rstudio/openid-client-secret) containing the client-id and client-secret. For example, the configuration file might look like this:

The /etc/rstudio/openid-client-secret file should have user read/write file permissions (i.e., 0600) to protect its contents from other users. You can ensure this as follows:

\$ sudo chmod 0600 /etc/rstudio/openid-client-secret

#### ! Important

The above client-id and client-secret aren't the actual values you'll use. Rather, you should substitute the values that you obtained from your OP when registering your site for OAuth 2.0 authentication.

Client Secret Encryption An unencrypted value in the client-secret option of the /etc/rstudio/openid-client-secret file must only be used temporarily for testing purposes. A warning will be present in RStudio log output when an unencrypted secret is being used.

We strongly recommend encrypting the client secret using the command rstudio-server encrypt-password. This way, if you have to backup your configuration, save it to a repository or share it with RStudio Support, your OpenID client secret will be protected.

Use the following steps to encrypt the OpenID client secret:

- Run the command sudo rstudio-server encrypt-password and enter the client secret.
- Copy the resulting encrypted secret printed in the terminal.
- Add or replace the client-secret option in the openid-client-secret file using the encrypted secret copied above.
- Restart RStudio. Confirm it operates normally. You should no longer see a warning about unencrypted secret in RStudio logs.

#### OpenID Issuer and Well-Known configuration

RStudio Workbench also needs to be configured to be able to authenticate the authorization codes returned by the OP. In order to do that, RStudio needs to know the location of the OP with its "issuer" URL. This can be done by adding the option auth-openid-issuer. This must be an HTTPS URL and the location of the /.well-known/openid-configuration metadata discovery. For example, if your OpenID issuer is https://op.example.com/ the discovery endpoint should be https://op.example.com/.well-known/openid-configuration.

```
# /etc/rstudio/rserver.conf
auth-openid-issuer=https://op.example.com
```

#### **OpenID Claims**

Your OP must return at least one claim matching the user's Linux account username (lower-case). By default, RStudio Workbench will look for a claim called preferred\_username (case-sensitive). If you wish to use a different claim, add the option auth-openid-username-claim with the appropriate value.

```
# /etc/rstudio/rserver.conf
auth-openid-username-claim=other
```

#### 18.0.3. Advanced Options

Depending on your choice for the username claim your OpenID Provider may require the initial request to contain a certain "scope". RStudio Workbench automatically includes the scopes openid, email and profiles as part of the authentication. If you wish to include additional scopes, use the option auth-openid-scopes with a space-separated list of values.

```
# /etc/rstudio/rserver.conf
auth-openid-scopes=scope1 scope2
```

#### Important

Quotation marks are not accepted around the scope values.

#### 18.0.4. Proxy Considerations

If you are running RStudio behind a proxy, you will need to configure your proxy in a way that RStudio can tell the OP to redirect back to the correct location. There are number of options to choose from as described in Running with a Proxy.

The use of the X-RStudio-Request header in your proxy is recommended and the only method which works behind a path-rewriting proxy. In this case, the proxy must set the X-RStudio-Request header to the exact complete URL as requested by the browser. For example if your proxy was set up to serve RStudio requests at https://testdomain.com/rstudio/and an incoming request for /home came in, your proxy should set X-RStudio-Request: https://testdomain.com/rstudio/home which would allow RStudio to know about the added path prefix /rstudio.

If your proxy does not add path prefixes, RStudio is also compatible with two options using commonly available HTTP proxy headers:

- The headers X-Forwarded-Host, X-Forwarded-Proto, and X-Forwarded-Port.
- Or the header Forwarded with host, and proto values.

When using path-rewriting proxies, it's also recommended to use either the header X-RStudio-Root-Path or the option www-root-path to indicate the path defined for RStudio by the proxy. For example, if your URL to RStudio is www.example.com/rstudio your proxy should send the header X-RStudio-Root-Path: /rstudio or you should use:

```
# /etc/rstudio/rserver.conf
www-root-path=/rstudio
```

If none of the headers above are set by the proxy, RStudio will attempt to redirect back to the address present in the Host header and it will determine the protocol (http or https) based on its current configuration.

If you are running behind a proxy but cannot set headers for whatever reason, and www-root-path is not the right choice, you can use the auth-openid-base-uri option in the RStudio configuration file to accomplish the same purpose:

```
# /etc/rstudio/rserver.conf
auth-openid-base-uri=http://testdomain.com/rstudio/
```

#### 18.0.5. Outgoing Proxies

Some OpenID authentication features require RStudio to make a call to an external service over HTTP or HTTPS; for example, to retrieve the provider's OpenID configuration from a known endpoint. If your environment requires an HTTP or HTTPS proxy for outbound requests, you must set the appropriate proxy environment variables for RStudio's server process so that it uses the proxy when making the request.

One way to do this is to add the variables to env-vars; for example:

```
# /etc/rstudio/env-vars
HTTP_PROXY=http://192.168.1.1:8080
HTTPS_PROXY=http://192.168.1.1:8080
NO_PROXY=localhost,192.168.1.10
```

#### 18.0.6. Load Balancing Considerations

Because RStudio stores OpenID authentication context in server memory during the OAuth flow, the entire authentication flow must be completed on a single server. If you're using an external load balancer in front of RStudio, you will experience authentication errors if the HTTP requests associated with the authentication flow are not all routed to the same server.

For this reason, you **must enable sticky sessions** in your external load balancer when using OpenID authentication. This feature is sometimes called "sticky cookies" or "session affinity". Consult the documentation for your load balancing software for details; for example if you're using the Amazon Web Services Application Load Balancer (AWS ALB), more information can be found in Sticky Sessions for your Application Load Balancer.

#### 18.0.7. Troubleshooting

Additional information about the OpenID flow and the received claims may be written to the logs. Be sure to configure rserver logs to output info level messages in /etc/rstudio/logging.conf to see these entries.

## 19. Proxied Authentication PRO

You can configure RStudio Workbench to participate in an existing web-based single-sign-on authentication scheme using proxied authentication. In this configuration all traffic to RStudio Workbench is handled by a proxy server which also handles user authentication.

In this configuration the proxy server adds a special HTTP header to requests to RStudio Workbench letting it know which authenticated user is making the request. RStudio Workbench trusts this header, launching and directing traffic to an R session owned by the specified user.

The specified user must have a local system account on the server. You should set up local system accounts manually and then map authenticating users to these accounts.

#### 19.0.1. Enabling Proxied Authentication

To enable proxied authentication you need to specify both the auth-proxy and auth-proxy-sign-in-url settings (the sign-in URL is the absolute URL to the page that users should be redirected to for sign-in). For example:

```
# /etc/rstudio/rserver.conf
auth-proxy=1
auth-proxy-sign-in-url=http://example.com/sign-in
```

#### ! Important

Once you enable authentication with a proxy, that becomes the exclusive means of authentication - you can't concurrently use both PAM and proxied authentication.

#### Important

Proxied authentication still requires PAM Sessions and sssd to automatically create local system accounts. Without them, local system accounts have to be provisioned manually one-by-one.

#### 19.0.2. Implementing the Proxy

#### Sign In and Sign Out URLs

The sign in URL should host a page where the user specifies their credentials (this might be for example the main page for an existing web-based authentication system). After collecting and authorizing the credentials the sign in URL should then redirect back to the URL hosting RStudio Workbench.

RStudio will redirect to the sign in URL under the following conditions:

- Whenever an HTTP request that lacks the username header is received by the server; and
- When the user clicks the "Sign out" button in the RStudio IDE user interface and there is no Sign Out URL available.

The sign out URL should host a page responsible for finishing the user session in the authentication proxy. If such a URL is URL available in your proxy, the absolute URL should be configured in RStudio using the setting auth-proxy-sign-out-url. When the user clicks the "Sign out" button in the RStudio IDE user interface the browser will be taken to the configured sign out URL.

```
# /etc/rstudio/rserver.conf
auth-proxy-sign-out-url=http://example.com/sign-out
```

You should be sure in setting up the proxy server that traffic bound for the sign-in and sign-out URLs is excluded from forwarding to RStudio Workbench (otherwise it will end up in an infinite redirect loop).

#### Sign-In Delay

During proxied authentication in RStudio, there is a brief transition page that shows the username and some other information. By default this transition happens almost immediately. If you wish to present this page for a longer period of time, you can use the option auth-proxy-sign-in-delay to delay the transition for some seconds.

```
# /etc/rstudio/rserver.conf
auth-proxy-sign-in-delay=4
```

#### Forwarding the Username

When proxying pre-authenticated traffic to RStudio Workbench you need to include a special HTTP header (by default X-RStudio-Username) with each request indicating which user the request is associated with. For example:

#### X-RStudio-Username: jsmith

It's also possible to specify both a system username and a display username (in the case where system accounts are dynamically provisioned and don't convey actual user identity). For example, if the system user is ruser24 but the displayed username is jsmith, you could use:

X-RStudio-Username: rsuser24/jsmith

#### Note

It is highly recommended that you do not use the default X-RStudio-Username header name. The reasons for this are described in the section on security considerations below.

#### **Rewriting Usernames**

It may be that the proxy system you are using sends the username in a format that doesn't match that of users on the system, however can be easily transformed to one that does (e.g. it has a standard prefix before the username). If this is the case you can specify the auth-proxy-user-header-rewrite option to provide a re-write rule for the inbound header. For example, the following rule strips the prefix "UID-" from a username header:

```
auth-proxy-user-header-rewrite=^UID-([a-z]+)$ $1
```

The format of a re-write rule is a regular expression followed by a space and then a replacement string. The replacement string can reference captured parts of the regular expression using \$1, \$2, etc. Consult the Boost Perl Regular Expression Syntax reference for more syntax documentation.

#### 19.0.3. Proxy Security Considerations

When using proxied authentication, RStudio trusts that the proxy is the only element in the network capable of sending the special header with the username. Be sure to follow the recommendations below to decrease security risks in your implementation.

#### **Keeping the Header Name Secret**

Using the default header name X-RStudio-Username creates a security problem: code running behind the proxy (e.g., code within R sessions) could form requests back to the server which impersonate other users (by simply inserting the header in their request).

To prevent this issue you can specify a custom header name which is kept secret from end users. This is done by creating a special configuration file (/etc/rstudio/secure-proxy-user-header) that contains the name of the header, and then setting it's file permissions so that it's not readable by normal users. For example:

```
sudo sh -c "echo 'X-Secret-User-Header' > /etc/rstudio/secure-proxy-user-header"
sudo chmod 0600 /etc/rstudio/secure-proxy-user-header
```

#### Preventing Remote Use of the Header

When implementing the proxy it's important to remember that RStudio Workbench will always trust the username header to authenticate users. It's therefore critical from the standpoint of security that all requests originating from the proxy have this header set explicitly by the proxy (as opposed to allowing the header to be specified by a remote client). RStudio will reject requests containing multiple occurrences of the username header.

#### **Preventing Internal Access**

#### Note

In previous versions, RStudio offered the option auth-proxy-require-hmac to require trust signatures from the proxy, though most proxies have no straightforward means for providing this signature. Therefore, this option has been retired. Following the recommendation below mitigates the same security risks previously covered by the option. If your installation used this option, it should removed from /etc/rstudio/rserver.conf. RStudio will refuse to start if this option is still present and enabled.

Your RStudio and proxy configuration should be done in a way where it is impossible for anything other than the proxy to make requests to RStudio. Be sure that:

- RStudio is configured to listen on a network interface not accessible internally by other processes by adjusting the option www-address.
- The network interface where RStudio is running must have firewall settings to prevent any connection to RStudio other than from the proxy.

#### ! Important

This should not be considered an exhaustive list. Please consult with your security personnel or IT administrators to determine the exact measures to protect RStudio authentication via a proxy.

#### 19.0.4. Troubleshooting with Access Logs

If you want to see exactly which requests RStudio is receiving and whether they include the expected username information, you can temporarily enable server access logs using the server-access-log setting as follows:

```
# /etc/rstudio/rserver.conf
server-access-log=1
```

After restarting RStudio the following file will contain a record of each HTTP request made to the server along with it's HTTP response code:

```
/var/log/rstudio-server/rserver-http-access.log
```

The log file will contain entries that look like this:

```
127.0.0.1 - - [29/Jun/2015:06:30:41 -0400] "GET /s/f01ddf8222bea98a/ HTTP/1.1" 200 91 "http://localhost:8787/s/f01ddf8222bea98a/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.125 Safari/537.36" "jsmith"
```

Note that the very last item in the log file entry is "jsmith". This is the username that RStudio read from the header passed by the proxy server. If this shows up as blank ("-") then your proxy server isn't forwarding the header or using the correct header name in forwarding.

#### Important

Once you've concluded troubleshooting it's important that you remove the server-access-log=1 option from the /etc/rstudio/rserver.conf file (since this log file is not rotated it will eventually consume a large amount of disk space if you don't remove the option).

## 20. Google Accounts (deprecated) PRO

#### i Note

While it is still possible to configure this authentication method, note that in future releases of RStudio, this authentication method will migrate to use OpenID Connect instead; Google itself uses OpenID as an underlying authentication mechanism. Additionally, RStudio Desktop Pro clients will be unable to connect to RStudio Workbench when it is configured to use Google Authentication due to new restrictions from Google - see Google's Notice for more information.

RStudio Workbench can be configured to authenticate users via Google Accounts. This enables users to log in with their existing Gmail or Google Apps credentials and to be automatically authenticated to RStudio whenever they are already logged into their Google account.

#### 20.0.1. Registering with Google

In order to use Google Accounts with RStudio Workbench you need to register your server with Google for OAuth 2.0 Authentication. You do this by creating a new "Project" for your server in the *Google Developer Console*:

https://console.developers.google.com/

Once you've created a project you go to the *Credentials* area of *APIs and auth* and choose to **Create New Client ID**:

You'll then be presented with a dialog used to create a new client ID:

You should select "Web application" as the application type and provide two URLs that correspond to the server you are deploying on. The screenshot above uses https://www.example.com as the host, you should substitute your own domain and port (if not using a standard one like 80 or 443) in your configuration.

This will result in two values which you'll need to provide as part of the RStudio Workbench configuration: client-id and client-secret (they'll be displayed in the *Google Developer Console* after you complete the dialog).

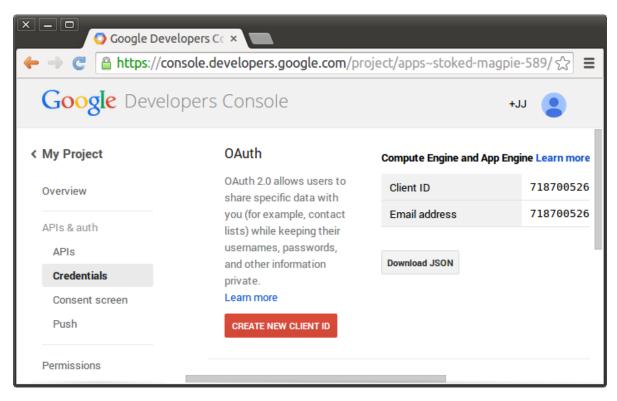


Figure 20.1.: Create Client Id

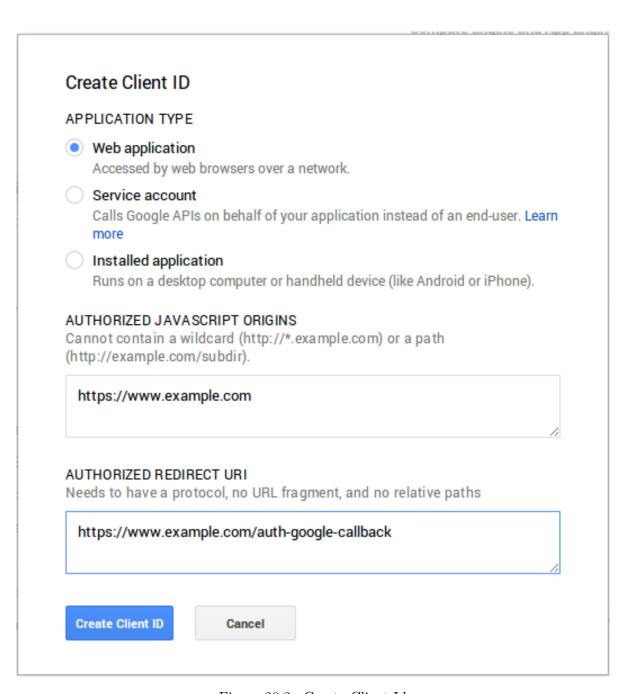


Figure 20.2.: Create Client Id

#### 20.0.2. Enabling Google Accounts

To enable authentication with Google Accounts you add the auth-google-accounts option to the RStudio configuration file:

```
# /etc/rstudio/rserver.conf
auth-google-accounts=1
```

Once you enable authentication with Google Accounts that becomes the exclusive means of authentication (you can't concurrently use both PAM and Google Account authentication).

In addition, you need to add a configuration file (/etc/rstudio/google-client-secret) containing the client-id and client-secret that you received when registering your site with Google. For example, the configuration file might look like this:

The /etc/rstudio/google-client-secret file should have user read/write file permissions (i.e., 0600) to protect it's contents from other users. You can ensure this as follows:

```
$ sudo chmod 0600 /etc/rstudio/google-client-secret
```

#### ! Important

The above client-id and client-secret aren't the actual values you'll use. Rather, you should substitute the values that you obtained from Google when registering your site for OAuth authentication.

#### 20.0.3. Translating to Local Accounts

#### **Creating Matching Accounts**

Once a user is authenticated via Google Accounts it's necessary to map their Google Accounts identity to a local system account. The default and most straightforward way to do this is to create a local account with a username identical to their Google email address.

If you choose to create local accounts that match Google email addresses, be sure to use only lowercase characters in the account name since Google email addresses are transformed to lower-case prior to matching them to local account names.

One problem with creating local accounts that match Google email addresses is that they often contain characters that are invalid by default within Linux usernames (e.g. @ or .). On Debian/Ubuntu systems it's possible to force the system to create a user with these characters. Here's an example of creating a user with a username that contains typically invalid characters:

```
$ sudo adduser --force-badname <username>
```

#### ! Important

The --force-badname option is only available on Debian/Ubuntu systems and is not available on RedHat/CentOS or SLES systems.

If the users you are creating will only be accessing the server via RStudio, you may also want to disable their ability to log in as a normal interactive user and to specify that they have no password. For example:

```
$ sudo adduser --force-badname --disabled-login --disabled-password <username>
```

#### **Using an Account Mappings File**

Alternatively, you map create local accounts that do not match Google email addresses and then specify a mapping of Google accounts to local accounts via the /etc/rstudio/google-accounts configuration file. For example:

```
# /etc/rstudio/google-accounts
john.smith@gmail.com=jsmith
sally.jones@gmail.com=sjones
```

Note that changes to the google-accounts configuration file take effect immediately and do not require a server restart.

#### 20.0.4. Proxy Considerations

If you are running RStudio behind a proxy, you will need to configure your proxy in a way that RStudio can tell the Google Web Services to redirect back to the correct location. There are number of options to choose from as described in Running with a Proxy.

The use of the X-RStudio-Request header in your proxy is recommended and the only method which works behind a path-rewriting proxy. In this case, the proxy must set the

X-RStudio-Request header to the exact complete URL as requested by the browser. For example if your proxy was set up to serve RStudio requests at https://testdomain.com/rstudio/and an incoming request for /home came in, your proxy should set X-RStudio-Request: https://testdomain.com/rstudio/home which would allow RStudio to know about the added path prefix /rstudio.

If your proxy does not add path prefixes, RStudio is also compatible with two options using commonly available HTTP proxy headers:

- The headers X-Forwarded-Host, X-Forwarded-Proto, and X-Forwarded-Port.
- Or the header Forwarded with host, and proto values.

When using path-rewriting proxies, it's also recommended to use either the header X-RStudio-Root-Path or the option www-root-path to indicate the path defined for RStudio by the proxy. For example, if your URL to RStudio is www.example.com/rstudio your proxy should send the header X-RStudio-Root-Path: /rstudio or you should use:

```
# /etc/rstudio/rserver.conf
www-root-path=/rstudio
```

If none of the headers above are set by the proxy, RStudio will attempt to redirect back to the address present in the Host header and it will determine the protocol (http or https) based on its current configuration.

If you are running behind a proxy but cannot set headers for whatever reason, and www-root-path is not the right choice, you can use the auth-google-accounts-redirect-base-uri option in the RStudio configuration file to accomplish the same purpose:

```
# /etc/rstudio/rserver.conf
auth-google-accounts-redirect-base-uri=http://testdomain.com/rstudio/
```

## 21. Authentication Migration

It is possible to migrate between the supported authentication mechanisms.

### 21.0.1. Migrating from PAM

Your local system accounts currently used with PAM can be used for Single Sing-On (SSO) authentication with SAML or OpenID. Ensure that:

- The existing PAM configuration is configured for PAM & Provisioning, as suggested above.
- RStudio is configured with PAM Sessions if your local system accounts are maintained by sssd.
- RStudio is configured with the appropriate SSO authentication mechanism.
- The configured SAML attribute or OpenID claim for username from your Identity Management system matches the names of your **existing** local system accounts.

#### ! Important

If PAM was used with Kerberos, please note that the credential forwarding functionality offered by Kerberos is only possible with PAM and it cannot be leveraged directly by RStudio when using SSO.

#### 21.0.2. Migrating to PAM

Since all other authentication methods already leverage PAM in some degree, there's no actual migration to PAM. You should only make sure PAM is configured for authenticating the existing users and configure RStudio to use PAM.

#### 21.0.3. Migrating from Proxied authentication

Your local system accounts currently used with Proxied authentication can be used for Single Sing-On (SSO) authentication with SAML or OpenID. Ensure that:

- If RStudio is placed under a different path by the proxy (e.g., example.com/rstudio), be sure to check the "Proxy Considerations" sections under SAML Single Sign-On Authentication or OpenID Connect Authentication for additional options your proxy or RStudio configuration may need.
- RStudio is configured with the appropriate SSO authentication mechanism.
- The configured SAML attribute or OpenID claim for username match the names of your existing local system accounts as they were sent by the proxy in the HTTP header for username.

#### 21.0.4. Migrating to Proxied authentication

#### Note

This migration is not recommended unless none of the other existing authentication mechanisms are sufficient for your organization's needs.

If Migrating from PAM, you can follow the same recommendations listed above for SSO, noting that the HTTP header for username must match existing accounts. If migrating from SAML or OpenID, the same observation on the HTTP header for username applies.

#### 21.0.5. Migrating from Google accounts

Migrating from Google accounts is similar to Migrating from PAM to SSO, or Migrating to PAM.

#### ! Important

Google accounts have been deprecated and we strongly recommend against migrating to this authentication.

#### Note

If you are currently using Google accounts for authentication, the migration from Google accounts to OpenID using Google itself as the OpenID provider is not yet supported. We recommend to keep using Google accounts or migrating to some other **non-Google** SSO authentication.

# Part IV. Access and Security

# 22. Overview

## 22.1. Access and Security

Application security is complex; there are many aspects to consider when deploying RStudio into your environment, and you should evaluate each of these areas based on your deployment scenario. These include:

- Network port, address, and IP access rules
- Browser considerations (Frame origin, cookies, and other browser compatibility)
- SSL
- Server account and permissions
- Proxy configurations

## 23. Network Port and Address

After initial installation RStudio accepts connections on port 8787. If you wish to listen on a different another port you can modify the www-port option. For example:

```
# /etc/rstudio/rserver.conf
www-port=80
```

By default RStudio binds to address 0.0.0.0 (accepting connections from any remote IP). You can modify this behavior using the www-address option. For example:

```
# /etc/rstudio/rserver.conf
www-address=127.0.0.1
```

Note that changes to the configuration will not take effect until the server is restarted.

# 24. Security Considerations

#### 24.0.1. IP Access Rules

RStudio Workbench can be configured to deny access to specific IP addresses or ranges of addresses. Access rules are defined in the configuration file /etc/rstudio/ip-rules

Access rules are established using the allow and deny directives and are processed in order, with the first matching rule governing whether a given address is allowed or denied. For example, to allow only clients within the 192.168.1.0/24 subnet but also deny access to 192.168.1.10 you would use these rules:

```
# /etc/rstudio/ip-rules
deny 192.168.1.10
allow 192.168.1.0/24
deny all
```

All clients outside of the specified subset are denied access because of the deny all rule at the end of the configuration.

Note that changes to the configuration will not take effect until the server is restarted.

#### 24.0.2. Frame Origin

For security reasons, RStudio will not load inside a browser frame (such as a frameset or IFrame) by default. You can modify this behavior by using the www-frame-origin option. For example, if you would like to host RStudio inside a browser frame at example.com, you can tell RStudio to allow this as follows:

```
# /etc/rstudio/rserver.conf
www-frame-origin=example.com
```

There are several special values available for the www-frame-origin option:

Value	Meaning
none	The default; do not allow RStudio to load in any frame.
same	Allow RStudio to load in a frame if it has the same origin (host
	and port) as RStudio.
any	Allow RStudio to load in a frame from any origin (not
	recommended)
$my ext{-}domain.com$	Allow RStudio to load in a frame at my-domain.com

To interact programmatically with RStudio in an IFrame, see the Tutorial API.

#### 24.0.3. CSRF Attack Mitigation

To help mitigate against CSRF attacks, RStudio can automatically reject any request originating from an Origin or Referer that does not match the Host, X-Forwarded-Host, Forwarded: host or X-RStudio-Request headers. To enable this check, add the following configuration:

```
# /etc/rstudio/rserver.conf
www-enable-origin-check=1
```

In some cases, such as if running behind a proxy that you cannot modify, this check may be too strict, and can prevent access to RStudio, causing requests to return a 400 status. In such cases, it is recommended that you disable the check. Origin checking is currently disabled by default, but will likely be enabled by default in a future version of RStudio.

You may wish to consider some origins to be safe in all cases, causing RStudio to consider such Origin or Referer values to be allowed instead of being rejected, even if they do not match a Host header. To specify such origins, add each of them as a www-allow-origin setting in rserver.conf. For example:

```
# /etc/rstudio/rserver.conf
www-allow-origin=mysubdomain.mydomain.com
www-allow-origin=*.mydomain2.com
```

Note that wildcards (\*) are accepted and will match any character, including hostname separators. For instance, the previous example of \*.mydomain2.com will match both foo.bar.mydomain2.com and foo.mydomain2.com.

#### 24.0.4. Additional Headers

In some cases, you may need to have RStudio set additional headers on client responses. To do this, simply specify server-add-header for each header that you need to add, in the form Header Name: Header Value. For example, to have the server set a few extra custom headers:

```
# /etc/rstudio/rserver.conf
server-add-header=X-Header-1:Value 1
server-add-header=X-Header-2:Value 2
```

# 25. Secure Sockets (SSL)

#### 25.0.1. SSL Configuration

If your RStudio Workbench is running on a public network then configuring it to use SSL (Secure Sockets Layer) encryption is strongly recommended. You can do this via the ssl-enabled setting along with related settings that specify the location of your SSL certificate and key. For example:

```
# /etc/rstudio/rserver.conf
ssl-enabled=1
ssl-certificate=/var/certs/your_domain_name.crt
ssl-certificate-key=/var/certs/your_domain_name.key
```

The .crt file should be encoded in the PEM format; that is, the first line should read ----BEGIN CERTIFICATE----, and the contents should be base64-encoded data. If your certificate is in another format, such as DER or PKCS, use the openss1 command-line tool to convert it to PEM. For example:

```
openssl x509 -inform DER -outform PEM -text -in your_domain_name.der -out your_domain_name.c
```

It's important when installing the certificate .crt file that you concatenate together any intermediate certificates (i.e. the generic one from your certificate authority) with the certificate associated with your domain name. For example you could use a shell command of this form to concatenate the CA intermediate certificate to your domain name's certificate:

```
$ cat certificate-authority.crt >> your_domain_name.crt
```

The resulting file should then be specified in the ssl-certificate option.

It's also important to ensure that the file permissions on your SSL certificate key are as restrictive as possible so it can't be read by ordinary users. The file should typically be owned by the **root** user and be set as owner readable and writable. For example:

#### 25.0.2. SSL Protocols

By default RStudio Workbench supports the TLSv1, TLSv1.1, TLSv1.2, and TLSv1.3 protocols for SSL. The list of supported protocols can configured via the ssl-protocols option. For example, to use only the TLSv1.1 and TLSv1.2 protocols you would use:

```
# /etc/rstudio/rserver.conf
ssl-protocols=TLSv1.1 TLSv1.2
```

The list of supported protocols is space delimited (as illustrated above). Valid protocol values are: SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2, and TLSv1.3.

Note that not all protocols may be available on your system; TLS 1.1 and 1.2 require OpenSSL 1.0.1, and TLS 1.3 requires OpenSSL 1.1.1 built with TLS 1.3 support.

#### 25.0.3. SSL Ports

When RStudio Workbench is configured to use SSL the default behavior with respect to ports is:

- 1) SSL is bound to port 443 (enabling access using the standard https protocol within the browser)
- 2) The server also listens on port 80 and redirects all requests to port 443 (allowing users to specify the domain without the https protocol and be automatically redirected to the secure port)

However, if SSL is bound to another port (using the www-port option) then the automatic redirect behavior is not enabled. It's also possible to disable automatic SSL redirects entirely using the ssl-redirect-http option as follows:

```
# /etc/rstudio/rserver.conf
ssl-redirect-http=0
```

Note that changes to the configuration will not take effect until the server is restarted.

#### 25.0.4. Strict Transport Security

When SSL is enabled, RStudio Workbench sends an HTTP Strict Transport Security (HSTS) header, Strict-Transport-Security, by default on outbound responses. This header tells the browser to forbid all HTTP connections to the domain for a period of time.

RStudio Workbench sets this period of time to 1 day (84600 seconds) by default, because if HTTPS issues arise it can be difficult to address them when the browser is locked to HTTPS because of HSTS. Once you are confident that your HTTPS setup is correct, you can increase the period by specifying the desired number of seconds in the ssl-hsts-max-age option. For example, to lock browsers to HTTPS for one year:

```
# /etc/rstudio/rserver.conf
ssl-hsts-max-age=31536000
```

If all subdomains of the server on which RStudio Workbench is hosted support HSTS, you can extend HSTS protection to them as well with the ssl-hsts-include-subdomains option. This doesn't happen by default since RStudio Workbench does not know what other services it's sharing a domain with, but it's a recommended security best practice, so you should turn it on if you can.

```
# /etc/rstudio/rserver.conf
ssl-hsts-include-subdomains=1
```

Finally, we do not recommend disabling HSTS, but if you need to, you can do so by setting ssl-hsts=0.

## 26. Cookies

#### 26.0.1. Secure Cookies

By default, when RStudio Workbench is configured with SSL, all authorization cookies are marked with the secure cookie attribute. If you wish to use your own SSL termination but force RStudio to mark cookies as secure, you can add the following configuration option:

```
# /etc/rstudio/rserver.conf
auth-cookies-force-secure=1
```

#### 26.0.2. Same Site Cookies

Most browsers have adopted recently a new attribute for cookies called SameSite which enforces additional security requirements around the cookie. RStudio does not define this attribute by default but you can use the option www-same-site for that purpose.

For example, the value SameSite=None is required by certain browsers (i.e. Chrome 84+) for embedding sites into an IFrame or frameset. To ensuring the delivery of RStudio cookies in this scenario, use the value www-same-site=none.

```
# /etc/rstudio/rserver.conf
www-same-site=none
```

#### ! Important

SameSite=None is insufficient without SSL enabled and may still prevent RStudio from working over non-SSL connections.

Some browsers will not behave as expected in the presence of the SameSite=None; Safari 12 on MacOS 10.14 and iOS 12.x or older notably will not work. To workaround these limitations, RStudio will emit a pair of cookies, one with SameSite=None for standard-conforming browsers and a "legacy" cookie without SameSite for non-conforming browsers.

You also can use the value www-same-site=lax to mark cookies as SameSite=Lax what effectively prohibits the use of RStudio embedded into an IFrame or frameset on compliant browsers by blocking all RStudio cookies.

#### 26.0.3. Third-Party Cookies (Safari 13+ Compatibility)

Starting in Safari 13, if you are loading RStudio inside a browser frame on a completely different domain all cookies will be blocked because those will be considered third-party cookies.

Embedding RStudio in a subdomain of the main site causes its cookies to be considered first-party, allowing RStudio to work.

Alternatively, you can place RStudio in the same domain as the main wrapping site but under a different URL path. This requires using a proxy in front of both the main site and RStudio. See Running with a Proxy below for more information.

#### i Note

In the near future, most browsers vendors plan to apply the same restrictions to cookies in some degree.

## 27. Server Permissions

#### 27.0.1. Server Account

RStudio runs as the system root user during startup and then drops this privilege and runs as a more restricted user. RStudio then re-assumes root privilege for a brief instant when creating R sessions on behalf of users (the server needs to call **setresuid** when creating the R session, and this call requires root privilege).

The user account that RStudio runs under in the normal course of operations is rstudio-server. This account is automatically added to the system during installation and is created as a system rather than end user account (i.e. the --system flag is passed to useradd).

#### **Alternate Server Account**

You can configure RStudio so that it will run from an alternate account with the following steps:

- 1. Create a new system user (if the one you want to use doesn't already exist)
- 2. Assign this user to the server-user option in the /etc/rstudio/rserver.conf configuration file (see example below)
- 3. Restart RStudio

For example, to shutdown the server and create a new system user named rs-user you'd use the following commands:

```
sudo rstudio-server stop
sudo useradd --system rs-user
```

Then edit the /etc/rstudio/rserver.conf configuration file as follows:

```
# /etc/rstudio/rserver.conf
server-user=rs-user
```

Finally, restart RStudio to begin running under the new user:

#### 27.0.2. umask

By default, RStudio sets its umask to 022 on startup. If you don't want this behavior, for instance because you'd prefer the server process to use the default umask set in init, it can be disabled as follows:

```
# /etc/rstudio/rserver.conf
server-set-umask=0
```

### 27.0.3. Running Without Permissions

RStudio Workbench can run in single user mode. This is primarily useful when it is a requirement to run entirely without root privilege; privilege is required to authenticate and run code on behalf of arbitrary users as noted above, but if only a single user will access the server and they have already been authenticated via other means, root privilege is not necessary.

Use a configuration like the following to run in single user mode; in this example, rstudiouser will be the single user:

```
# /etc/rstudio/rserver.conf
# Run the server under the single user account
server-user=rstudiouser

# Disable authentication altogether, since only one user can authenticate
auth-none=1

# Disable project sharing, since adjusting file permissions requires privilege
server-project-sharing=0

# Disable PAM sessions, since privilege is required to open and close PAM sessions
auth-pam-sessions-enabled=0
```

Finally, if you are using a service manager to invoke RStudio Workbench, you will need to configure your service manager to start RStudio Workbench under the desired account. For example, if you are using the systemd init system, run sudo systemctl edit rstudio-server and add the following directive:

[Service]

User=rstudiouser

## 28. Running with a Proxy

#### 28.0.1. Overview

If you are running RStudio behind a proxy server you need be sure to configure the proxy server so that it correctly handles all traffic to and from RStudio.

Beyond the normal reverse proxy configuration you'd apply for any HTTP server application, you also need to ensure that websockets are forwarded correctly between the proxy server and RStudio to ensure that all RStudio functions work correctly. In particular, they're needed to ensure that Shiny applications run from within the IDE work properly - if not, you may find that Shiny applications "gray out" and close without you being able to interact with them.

It's also important to ensure that your reverse proxy uses a relatively lenient connection timeout; we recommend 60 seconds. Several components of RStudio use HTTP Long Polling to push information to the browser; a connection timeout of 30 seconds or fewer will result in HTTP 504 (gateway timeout) errors from the reverse proxy.

This section describes how to correctly configure a reverse proxy with Nginx and Apache.

#### 28.0.2. Nginx Configuration

On Debian or Ubuntu a version of Nginx that supports reverse-proxying can be installed using the following command:

```
sudo apt-get install nginx
```

On CentOS or Red Hat you can install Nginx using the following command:

```
sudo yum install nginx
```

To enable an instance of Nginx running on the same server to act as a front-end proxy to RStudio you would add commands like the following to your nginx.conf file. Note that you must add code to proxy websockets in order to correctly display Shiny apps and R Markdown Shiny documents in RStudio. Also note that if you are proxying to a server on a different machine you need to replace references to localhost with the correct address of the server where you are hosting RStudio.

```
http {
  map $http_upgrade $connection_upgrade {
    default upgrade;
          close;
  }
  server {
    listen 80;
    location / {
      proxy_pass http://localhost:8787;
      proxy_redirect http://localhost:8787/ $scheme://$host/;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection $connection_upgrade;
      proxy_read_timeout 20d;
      # Use preferably
      proxy_set_header X-RStudio-Request $scheme://\$host:\$server_port\$request_uri;
      # OR existing X-Forwarded headers
      # proxy_set_header X-Forwarded-Host $host;
      # proxy_set_header X-Forwarded-Proto $scheme;
      # OR alternatively the Forwarded header (just an example)
      # proxy_set_header Forwarded "host=$host:$server_port;proto=$scheme;";
    }
  }
```

If you want to serve RStudio from a custom path (e.g. /rstudio) you would edit your nginx.conf file as shown below:

```
http {
  map $http_upgrade $connection_upgrade {
    default upgrade;
    '' close;
}
```

```
server {
  listen 80;
  location /rstudio/ {
    rewrite ^/rstudio/(.*)$ /$1 break;
    proxy_pass http://localhost:8787;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;
    proxy_read_timeout 20d;
    # Use preferably
    proxy_set_header X-RStudio-Request $scheme://$host:$server_port$request_uri;
    proxy_set_header X-RStudio-Root-Path /rstudio;
    # OR let the proxy rewrite the root path
    # proxy_redirect http://localhost:8787/ $scheme://$host/rstudio/;
    # proxy_cookie_path / /rstudio/;
    # OR existing X-Forwarded headers
    # proxy_set_header X-Forwarded-Host $host;
    # proxy_set_header X-Forwarded-Proto $scheme;
    # OR alternatively the Forwarded header (just an example)
    # proxy_set header Forwarded "host=$host:$server_port;proto=$scheme;";
}
```

#### Note

The header X-RStudio-Root-Path and the configuration option www-root-path server the same purpose. If either is set RStudio will always return cookies and redirects for the correct path, without requiring rewrite assistance from the proxy. The header value has precedence over the configuration value.

```
# /etc/rstudio/rserver.conf
www-root-path=/rstudio
```

After adding these entries you'll then need to restart Nginx so that the proxy settings take effect:

```
sudo /etc/init.d/nginx restart
```

Note that RStudio needs the X-RStudio-Request, X-Forwarded-[Host|Port|Proto], or Forwarded headers set for various security reasons, and nginx does not supply this header by default. It must contain the original Host value of the request, which is usually set to \$host in the nginx configuration file.

In some cases, such as when streaming job statuses from the launcher, the default response buffering in nginx can be too slow for delivering real-time updates, especially when configured to use SSL. If job output streams are not working properly from the home page, we recommend disabling response buffering by adding the following line under the **server** directive:

```
server {
    # ... follows previous configuration
    proxy_buffering off;
}
```

#### 28.0.3. Apache Configuration

To enable an instance of Apache running on the same server to act as a front-end proxy to RStudio you need to use the mod\_proxy and mod\_proxy\_wstunnel modules. The steps for enabling this module vary across operating systems so you should consult your distribution's Apache documentation for details. Apache as reverse proxy already includes X-Forwarded-Host (with port) and X-Forwarded-Proto by default.

On Debian and Ubuntu systems Apache can be installed with mod\_proxy using the following commands:

```
sudo apt-get install apache2
sudo apt-get install libapache2-mod-proxy-html
sudo apt-get install libxml2-dev
```

Then, to update the Apache configuration files to activate mod\_proxy you execute the following commands:

```
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod proxy_wstunnel
```

On CentOS and RedHat systems Apache can be installed with mod\_proxy and mod\_proxy\_wstunnel by following the instructions here:

#### http://httpd.apache.org/docs/2.4/platform/rpm.html

By default with Apache 2.4, mod\_proxy and mod\_proxy\_wstunnel should be enabled. You can check this by opening the file /etc/httpd/conf.modules.d/00-proxy.conf and making sure the following lines are included and not commented out:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
```

Once you have enabled mod\_proxy and mod\_proxy\_wstunnel in your Apache installation you need to add the required proxy commands to your VirtualHost definition. Note that you will also need to include code to correctly proxy websockets in order to correctly proxy Shiny apps and R Markdown documents within RStudio. Also note that if you are proxying to a server on a different machine you need to replace references to localhost with the correct address of the server where you are hosting RStudio.

```
<VirtualHost *:80>
 <Proxy *>
   Allow from localhost
 </Proxy>
 RewriteEngine on
 RewriteCond %{HTTP:Upgrade} =websocket
 RewriteRule /(.*) ws://localhost:8787/$1 [P,L]
 RewriteCond %{HTTP:Upgrade} !=websocket
 RewriteRule /(.*) http://localhost:8787/$1 [P,L]
 ProxyPass / http://localhost:8787/
 ProxyPassReverse / http://localhost:8787/
 # Use preferably this (store variable values with dummy rewrite rules)
 RewriteRule . - [E=req_scheme:%{REQUEST_SCHEME}]
 RewriteRule . - [E=http_host:%{HTTP_HOST}]
 RewriteRule . - [E=req_uri:%{REQUEST_URI}]
 RequestHeader set X-RStudio-Request "%{req_scheme}e://%{http_host}e%{req_uri}e"
 ProxyRequests Off
</VirtualHost>
```

Note that if you want to serve RStudio from a custom path (e.g. /rstudio) you would replace the directives described above to:

```
RewriteEngine on
RewriteCond %{HTTP:Upgrade} =websocket
RewriteRule /rstudio/(.*)
                              ws://localhost:8787/$1 [P,L]
RewriteCond %{HTTP:Upgrade} !=websocket
RewriteRule /rstudio/(.*)
                             http://localhost:8787/$1 [P,L]
ProxyPass /rstudio/ http://localhost:8787/
# Use preferably this (store variable values with dummy rewrite rules)
RewriteRule . - [E=req scheme:%{REQUEST SCHEME}]
RewriteRule . - [E=http_host:%{HTTP_HOST}]
RewriteRule . - [E=req uri:%{REQUEST URI}]
RequestHeader set X-RStudio-Request "%{req_scheme}e://%{http_host}e%{req_uri}e"
RequestHeader set X-RStudio-Root-Path "/rstudio"
# Or alternatively
ProxyPassReverse /rstudio/ http://localhost:8787/
ProxyPassReverseCookiePath "/" "/rstudio"
ProxyRequests Off
```

#### i Note

The header X-RStudio-Root-Path and the configuration option www-root-path server the same purpose. If either is set RStudio will always return cookies and redirects for the correct path, without requiring rewrite assistance from the proxy. The header value has precedence over the configuration value.

```
# /etc/rstudio/rserver.conf
www-root-path=/rstudio
```

Finally, after you've completed all of the above steps you'll then need to restart Apache so that the proxy settings take effect:

```
sudo /etc/init.d/apache2 restart
```

#### 28.0.4. RStudio Configuration

If your RStudio Workbench and proxy server are running on the same machine you can also change the port RStudio Workbench listens on from 0.0.0.0 (all remote clients) to 127.0.0.1 (only the localhost). This ensures that the only way to connect to RStudio is through the proxy server. You can do this by adding the www-address entry to the /etc/rstudio/rserver.conf file as follows:

Note that you may need to create this config file if it doesn't already exist.

#### 28.0.5. Custom Paths and Path-Rewriting Proxies

In the examples above we have configurations where RStudio is served by the proxy under a custom /rstudio path. This is called a path-rewriting proxy setup.

RStudio can use different combinations of HTTP headers and/or options to determine its location when path-rewriting in is use:

- Use X-RStudio-Request if possible. This way RStudio knows exacly the address presented in the user's browser.
  - Otherwise, you must use X-Forwarded-\* family of headers or the Forwarded header.
- Use X-RStudio-Root-Path if possible. This way RStudio knows which portion of the path was added by the proxy.
  - Alternatively, use the option www-root-path for the same effect.
  - Otherwise, you must use additional options in your proxy configuration to adjust redirects and cookies for the right path.

The most reliable configuration is using X-RStudio-Request and X-RStudio-Root-Path defined as in the examples above. There's little involvement of the proxy when using these headers.

If you have little experience with proxies but still want to use a custom path, we recommend using the www-root-path option in RStudio and at least the headers X-Forwarded-Host and X-Forwarded-Proto.

Finally, if you want the proxy to have total control of custom path then define the rewrite rules yourself directly in the proxy configuration based on the alternatives suggested in the example above. In this case, do not use the option www-root-path or the header X-RStudio-Root-Path.

#### 28.0.6. Customizing Default Proxy

RStudio Workbench exposes itself over TCP by means of an nginx proxy instance that runs as the rserver-http process on the local machine. In some cases, this proxy instance may need to be customized.

In order to customize it, you can create any of the following three files. Each file modifies the nginx configuration at /var/lib/rstudio-server/conf/rserver-http.conf in the following way:

- /etc/rstudio/nginx.http.conf allows you to add additional nginx directives under the root http node, and should be used for altering basic HTTP settings
- /etc/rstudio/nginx.server.conf allows you to add additional nginx directives under the server node, and should be used for altering basic server settings
- /etc/rstudio/nginx.site.conf allows you to add additional nginx directives under the location / node, and should be used for altering responses sent from RStudio

Simply add the desired nginx configuration in the files above to modify the desired section - the contents of each file is copied into the rserver-http.conf template verbatim. Then, restart rstudio-server for the changes to take effect.

In most cases, you should not need to create these files and modify the nginx template that is provided.

# Part V. R Sessions

# 29. Overview

## 29.1. R Sessions

RStudio Workbench's primary role is to launch and manage sessions for users. An R Session is a session configured with R and various shared libraries and R packages which the user accesses through the browser as the RStudio IDE. If using the job launcher, sessions can be launched with environments for R, Jupyter, Jupyter Lab, and VS Code. Without the job launcher, Workbench is limited to launching R Sessions. This section explains how R Sessions are configured.

## 30. R Executable and Libraries

#### 30.0.1. Locating R

RStudio uses the version of R pointed to by the output of the following command:

#### \$ which R

The which command performs a search for the R executable using the system PATH. RStudio will therefore by default bind to the same version that is run when R is executed from a terminal.

For versions of R installed by system package managers this will be /usr/lib/R. For versions of R installed from source this will typically (but not always) be /usr/local/lib/R.

If you want to override which version of R is used then you can use the rsession-which-r setting. For example:

```
# /etc/rstudio/rserver.conf
rsession-which-r=/usr/local/bin/R
```

Note that this change will not take effect until the server is restarted.

#### Using Multiple Versions of R

The section above describes how RStudio locates the global default version of R. It's also possible to specify alternate versions of R either by user or by group. The R Versions section describes this in more detail.

#### 30.0.2. Locating Shared Libraries

You can add elements to the default LD\_LIBRARY\_PATH for R sessions (as determined by the R ldpaths script) by adding an rsession-ld-library-path entry to the server config file. This might be useful for ensuring that packages can locate external library dependencies that aren't installed in the system standard library paths. For example:

```
# /etc/rstudio/rserver.conf
rsession-ld-library-path=/opt/someapp/lib:/opt/anotherapp/lib
```

Note that this change will not take effect until the server is restarted.

#### 30.0.3. Customizing Session Launches

#### **Profile Script Execution**

RStudio launches R sessions under a bash login shell. This means that prior to the execution of the R session the bash shell will read and execute commands from this file if it exists:

```
/etc/profile
```

After reading that file, it looks for the following files and reads and executes commands from the *first* one that exists and is readable (it's important to note that only one of these files will be read and executed):

```
~/.bash_profile
~/.bash_login
~/.profile
```

If you have further RStudio specific initialization logic (exporting environment variables, etc.) you can optionally create an R session specific profile script at:

```
/etc/rstudio/rsession-profile
```

If it exists this script will be executed prior to the bash shell that launches the R session. This script must be executable by all RStudio users, so it is recommended that you set its file permissions to 755 via the following command:

```
chmod 755 /etc/rstudio/rsession-profile
```

In some situations, you will not want to run user shell profile scripts. This is also a good way to troubleshoot the inability for sessions to launch, as it could indicate a conflict is occurring due to environment variables being set in the shell profiles. To disable execution of the shell profiles, set the rsession-no-profile option to 1 in /etc/rstudio/rserver.conf. For example:

```
# /etc/rstudio/rserver.conf
rsession-no-profile=1
```

#### **Environment Variables**

R sessions inherit environment variables that are explicitly exported from the profile scripts described above. It's also possible to append paths to the LD\_LIBRARY\_PATH environment variable using the rsession-ld-library-path option (see previous section for details).

Another source of environment variables are PAM sessions. On Debian/Ubuntu systems, the default PAM profile run by RStudio includes the environment variables defined in /etc/security/pam\_env.conf and /etc/environment. To learn more about setting environment variables with PAM you should consult the PAM Sessions section as well as the documentation on the pam\_env module here: http://linux.die.net/man/8/pam\_env.

#### **Program Supervisors**

You may also wish to run R sessions under a program supervisor that modifies their environment or available resources. You can specify a supervisor (and the arguments which control it's behavior) using the rsession-exec-command setting. For example:

```
# /etc/rstudio/rserver.conf
rsession-exec-command=nice -n 10
```

This example uses the nice command to run all R sessions with a lower scheduling priority. See <a href="http://linux.die.net/man/1/nice">http://linux.die.net/man/1/nice</a> for more details on nice. Note that for nice in particular it's possible to accomplish the same thing using user and group profiles (and even specify a custom priority level per user or group). See the User and Group Profiles section for more details.

#### 30.0.4. Diagnosing Session Launch Failures

RStudio Workbench allows you to launch sessions in a diagnostics mode to capture extended session launch information in the event that you run into problems when starting sessions. This mode captures the environment variables that are set by all of the profile scripts, loading of any modules, arguments passed to the session, and the exit code of the session and any stack trace information if the session crashes. To enable collection of this diagnostic data, set the rsession-diagnostics-enabled flag in /etc/rstudio/rserver.conf like below:

```
# /etc/rstudio/rserver.conf
rsession-diagnostics-enabled=1
```

When this setting is enabled, all sessions that are started will create a diagnostics file that contains extended launch diagnostics under the /tmp directory by default. To change the location, use the rsession-diagnostics-dir to point to the desired directory, like so:

```
# /etc/rstudio/rserver.conf
rsession-diagnostics-dir=/tmp/diagnostics
```

Be aware that this directory should be accessible by all users of the system. If it does not exist, RStudio will attempt to create it and set its permissions to Read/Write/Execute for everyone (file permissions of 777).

To diagnose containerized sessions (e.g. Kubernetes) launched via the Job Launcher, set rsession-diagnostics-dir to a shared folder mounted in the container via /etc/rstudio/launcher-mounts. Otherwise sessions may fail to launch due to the default location under /tmp not being available in the containers.

Strace information can be included in the diagnostics file by adding the rsession-diagnostics-strace-enabled flag as shown below. Before using this setting, ensure that strace is installed and is available on your users' path.

```
# /etc/rstudio/rserver.conf
rsession-diagnostics-enabled=1
rsession-diagnostics-strace-enabled=1
```

We recommend that you send these files to RStudio support to aid in troubleshooting any session launch issues should they arise.

#### Safe Mode

Two of the most common session startup problems are (1) oversized global R environments, which take a long time to load into memory when the session is resumed, and (2) problematic code in .Rprofile which takes too long to run or throws errors during startup.

In order to help eliminate these variables as the cause of session startup issues, RStudio can start sessions in Safe Mode. This mode starts the session without running .Rprofile or restoring the global environment. If a session takes too long to start, the user will be given the option to restart a session in Safe Mode.

It is also possible to control these features independently, which can be helpful when troubleshooting issues.

**Skipping Workspace Restoration** To skip workspace restoration (i.e. reloading the contents of the global environment), append <code>?restore\_workspace=0</code> to the R session's full URL. For example:

https://yourcorp/rstudio/s/4cc57da229b59e81c306b/?restore\_workspace=0

Note that this will work *only if the session is not already running*. If you are currently waiting for the session to start, it is too late to try to skip workspace restoration, since it is already in progress. Kill or force-kill the session before restarting the attempt with the restore\_workspace flag.

**Omitting .Rprofile** To skip executing .Rprofile, append run\_rprofile=0 to the R session's full URL. For example:

https://yourcorp/rstudio/s/4cc57da229b59e81c306b/?run\_rprofile=0

Just like restore workspace, this can only be used prior to the session starting.

Note that Safe Mode is only available when RStudio itself is starting sessions, so it is not a useful troubleshooting technique when sessions are being run using the Job Launcher.

# 31. Sessions Startup Scripts

When an R session starts up, the following scripts are run:

- 1. Any PAM session modules Pro
- 2. The /etc/rstudio/rsession-profile script Pro
- 3. The environment module for the R version in use, if any; see Extended R Version Definitions Pro
- 4. The prelaunch script for the R version in use, if any; see Extended R Version Definitions
  Pro
- 5. System and user shell profile scripts, such as .bash\_profile Pro
- 6. The R session itself, inside any supervisor specified by rsession-exec-command Pro
- 7. .Rprofile and other R initialization scripts; see Initialization at the Start of a Session for details
- 8. RStudio-internal session startup scripts
- 9. The rstudio.sessionInit hook

The very last of these provides a way to run code after the R session is fully booted and ready to use; since it only runs in RStudio it can be used to finalize R sessions in RStudio-specific ways, using for example methods in the rstudioapi package. Here is an example that prints the RStudio version in new R sessions:

```
# /opt/R/version/lib64/r/etc/Rprofile.site
setHook("rstudio.sessionInit", function(newSession) {
   if (newSession)
      message("Welcome to RStudio ", rstudioapi::getVersion())
}, action = "append")
```

# 32. User and Group Profiles PRO

User and Group Profiles enable you to tailor the behavior of sessions on a per-user or per-group basis. The following attributes of a session can be configured within a profile:

- 1) Version of R used
- 2) CPU affinity (i.e. which set of cores the session should be bound to)
- 3) Scheduling priority (i.e. nice value)
- 4) Resource limits (maximum memory, processes, open files, etc.)
- 5) R session timeouts (amount of idle time which triggers session suspend)
- 6) R session kill timeouts (amount of idle time which triggers a session to be destroyed and cleaned up)

#### 32.0.1. Creating Profiles

Profiles are defined within the file /etc/rstudio/profiles. Note that this file is not created by default so you'll need to create it if doesn't already exist. Profiles are divided into sections of three different type:

- 1) Global ([\*])
- 2) Per-group ([@groupname])
- 3) Per-user ([username])

Here's an example profiles file that illustrates each of these types:

```
# /etc/rstudio/profiles
[*]
cpu-affinity = 1-4
max-processes = 800
max-memory-mb = 2048
session-limit=5
session-timeout-minutes=60
session-timeout-kill-hours=24

[@powerusers]
cpu-affinity = 5-16
```

```
nice = -10
max-memory-mb = 4096
session-limit=10

[jsmith]
r-version = /opt/R/3.1.0
session-timeout-minutes=360
```

This configuration specifies that by default users will run on cores 1 to 4 with a limit of 800 processes and 2GB of virtual memory. It also specifies that members of the powerusers group will run on cores 5 to 16 with an elevated nice priority and a limit of 4GB of memory. Finally, the user jsmith is configured to use a different version of R from the system default.

Note that the /etc/rstudio/profiles file is processed from top to bottom (i.e. settings matching the current user that occur later in the file always override ones that appeared prior). Additionally, some settings do not apply to certain editor types (like Jupyter or VS Code). The settings available within /etc/rstudio/profiles are described in more depth below.

#### 32.0.2. Session Limit

To configure the maximum allowed number of sessions you can use the session-limit option. This is a more flexible choice than disabling multiple sessions entirely with server-multiple-sessions.

For example to limit user to 5 sessions:

```
session-limit=5
```

By default there is no limit to the number of sessions other than the one specified by the license. This limit has no effect if greater than the number of sessions allowed by your license.

This setting applies to all session types (RStudio, Jupyter, and VS Code).

#### 32.0.3. Session Timeout

To configure the amount of idle time to wait before suspending sessions you can use the session-timeout-minutes option. Idle time is the amount of time since a session has run any R commands, including commands typed by the user in the console and the execution of any R scripts.

For example:

#### session-timeout-minutes=360

The default value if none is explicitly specified is 120 minutes.

There are some conditions where an R session will not be suspended, these include:

- 1) When a top-level R computation is running
- 2) When the R prompt is not in it's default state (e.g. during a debugging session)

You can also specify that R sessions should never be suspended by setting the session-timeout-minutes to zero. For example:

#### session-timeout-minutes=0

This setting only applies to the RStudio IDE, and does not apply to Jupyter or VS Code. Session timeout can be configured for Jupyter sessions via the session-cull setting - see Jupyter configuration for more details. There is currently no mechanism for timing out VS Code sessions.

#### 32.0.4. Session Timeout Kill

To configure the amount of time to wait before forcefully killing and destroying sessions you can use the session-timeout-kill-hours option in the /etc/rstudio/rsession.conf file. This allows you to specify how much time should pass before a session is automatically cleaned up. This is the length of time during which a session exists without user input (regardless of computation status) allowing you to automatically reclaim temporary disk space used by the session, and to stop its processes and children.

This setting should only be used to ensure that any sessions that users have forgotten about are destroyed, reclaiming valuable disk space. Note that this setting does not work if the server-shared-storage-path is located on a root squash mount.

Sessions can be destroyed while important computations are executing. The user whose session is cleaned will also lose all unsaved code and data.

This setting can also be used in conjunction with session-timeout-minutes, allowing already suspended sessions to be cleaned up. For example:

#### session-timeout-kill-hours=48

The default value if none is explicitly specified is 0 hours, meaning sessions will never be killed and destroyed automatically. The supplied value should be an integer representing the amount of hours a session can go without user interaction before being killed.

This setting only applies to the RStudio IDE, and does not apply to Jupyter or VS Code. Session timeout can be configured for Jupyter sessions via the session-cull setting - see Jupyter configuration for more details. There is currently no mechanism for timing out VS Code sessions.

#### Interaction with session-timeout-minutes

It is recommended that the session-timeout-kill-hours be set to a much higher span of time than that specified by session-timeout-minutes. This will allow sessions to be suspended (and not destroyed) before they are forcefully killed and cleaned up, allowing for a better user experience. Note however, that if a session is running a long-running computation, it may not be suspended but it will still be killed.

If session-timeout-kill-hours is set to an amount of time less than session-timeout-minutes (which has a default of 2 hours), then sessions will *never* be suspended and they will be forcefully killed and cleaned up.

The two settings may be set to the same amount of time, but this is not recommended. In this case, it is undefined whether or not the session will be suspended, but the session will be killed and cleaned up regardless.

#### 32.0.5. CPU Affinity and Scheduling Priority

If you have users or groups that consistently require more compute resources than others you can use profile settings to reserve CPUs (cpu-affinity) as well as raise overall scheduling priority (nice).

#### **CPU Affinity**

The cpu-affinity setting specifies which cores on a multi-core system should be used to schedule work for a session. This is specified as a comma-separated list of core numbers (1-based) where both individual cores and ranges of cores can be specified. For example:

```
cpu-affinity = 1,2,3,4
cpu-affinity = 1-4
cpu-affinity = 1,2,15-16
```

To determine the number of addressable cores on your system you can use the nproc command:

```
$ nproc
```

This setting applies to all session types (RStudio, Jupyter, and VS Code).

#### **Scheduling Priority**

The nice setting specifies a relative priority for scheduling session CPU time. Negative 20 is the highest nice priority and positive 20 is the lowest priority. The system default niceness for processes is typically 0. The following are all valid nice values:

```
nice = -10
nice = 0
nice = 15
```

Scheduler behavior around nice priorities varies by system. For more details see nice use and effect.

This setting applies to all session types (RStudio, Jupyter, and VS Code).

#### 32.0.6. Resource Limits

Profiles can also be used to specify limits on available memory as well as the maximum number of processes and open files.

#### **Available Memory**

The max-memory-mb setting controls the maximum amount of addressable memory for R sessions (by default memory is unlimited). This example specifies a limit of 2GB:

```
max-memory-mb = 2048
```

Note that this value sets the amount of virtual memory that can be used by a process. Virtual memory includes code (i.e. shared libraries) loaded by the process as well as things like memory mapped files, so can often consume several hundred megabytes even for a vanilla R session. Therefore, you want to be sure not to set this threshold too low (in no case should you set it below 1024).

This setting applies to all session types (RStudio, Jupyter, and VS Code).

#### **Number of Processes**

The max-processes setting controls the maximum number of system processes createable by a user. This setting is useful to prevent either inadvertent or malicious fork bombs. The following example sets a limit of 800 processes:

```
max-processes = 800
```

Note that max-processes refers to system processes, not R processes. Users may need to create many system processes in order to use RStudio, so we recommend setting this value high and testing the outcome before deploying in production.

This setting applies to all session types (RStudio, Jupyter, and VS Code).

#### **Number of Open Files**

In most Linux environments there is a maximum of 1024 open files per process. This is typically more than enough, but if you have a particular applications that requires more open files the max-open-files setting can be used to increase the limit. For example:

```
max-open-files = 2048
```

This setting applies to all session types (RStudio, Jupyter, and VS Code).

#### 32.0.7. Using Multiple Versions of R

As illustrated above, you can bind users or groups to distinct versions of R installed on your server. This is controlled by the **r-version** option. Here are several examples of it's use:

```
r-version = /usr/lib/R
r-version = /usr/local/lib/R
r-version = /opt/R/3.1.0
r-version = /opt/R/3.2.0
```

Note that r-version specifies the full path to the directory where R is installed.

See the R Versions chapter for additional details on running multiple versions of R on a single server.

This setting only applies to the RStudio IDE, and does not apply to Jupyter or VS Code.

#### 32.0.8. Usage in Distributed Environments

User profiles are still applicable when R sessions are not all co-located on the same server, but some considerations apply.

#### Job Launcher

If launching sessions via the Job Launcher, scheduling priority and resource limits are not ignored, but these should generally be configured via the Job Launcher itself where applicable. For example, when running Kubernetes sessions, memory limits can be specified each time a session is launched, so memory limits should be configured by the Job Launcher in that case.

#### **Load Balancer**

When using load balancing, resource limits are applied to a session *after* it has been assigned to a server in a load-balanced cluster. So, for example, if you have cpu-affinity set to 1-4 for a user, then the user's session can run on cores 1 through 4 of any server in the cluster. Likewise, a session's max-memory-mb value will cap the memory it can use on the server on which it runs, but doesn't otherwise restrict memory usage across servers.

## 33. Multiple R Sessions PRO

RStudio Workbench enables users to have multiple concurrent R sessions on a single server or load balanced cluster of servers (the open-source version of RStudio Server supports only a single session at a time).

#### 33.0.1. Creating New Sessions

You can start a new R Session using the **New Session** command from the **Session** menu (or the corresponding toolbar button near the top-right of the IDE).

You can also open an existing RStudio project in a new R session by using the **Open Project in New Session** command. When switching projects there is also a button on the right side of the projects menu that lets you specify that the project should be opened in a new session rather than within the current one.

You can review all currently running sessions and switch between them using the **Sessions** toolbar near the top-right of the IDE.

#### 33.0.2. Session Lifetimes

R Sessions are long-running tasks that continue to be available until you explicitly quit them (you can think of them as you'd think of multiple top-level RStudio windows running on the desktop). This means that you can kickoff a long running job in one session and then switch to another session, revisiting the original session later to check on it's progress. As is also possible on the desktop, you can navigate between different projects and working directories within a session.

Sessions will suspend automatically when they are idle and then be automatically resumed next time they are accessed. To permanently quit a session you can use the **Quit Session** command located on the **File** menu or the corresponding toolbar button at the top right of the IDE.

## 33.0.3. Disabling Multiple Sessions

If you wish disable support for multiple sessions you can use the <code>server-multiple-sessions</code> option. For example:

# /etc/rstudio/rserver.conf
server-multiple-sessions=0

## 34. PAM Sessions PRO

RStudio Workbench uses PAM (Pluggable Authentication Modules) for both user authentication as well to establish the environment and resources available for R sessions. This is accomplished using the PAM session API. PAM sessions are used for a variety of purposes:

- To initialize environment variables
- To automatically create local users after authentication or authorization against a directory server.
- To mount remote drives
- To initialize and destroy Kerberos tickets

This section explains how to configure and customize PAM sessions with RStudio Workbench.

#### 34.0.1. Session PAM Profile

For PAM authentication RStudio Workbench uses the either the /etc/pam.d/other profile (Debian/Ubuntu) or /etc/pam.d/rstudio profile (RedHat/CentOS). However, for launching R sessions a different PAM profile is used. This is because the launching of R sessions may not coincide with authentication (e.g. returning to the site with login credentials cached in a cookie or resuming a suspended session). Therefore, the PAM directive that enables authentication with root privilege only (auth sufficient pam\_rootok.so) needs to be present in the PAM profile.

#### Note

Some sssd configuration additionally require PAM account verification as root to present on both the auth and account directives in the PAM profile (auth sufficient pam\_rootok.so and account sufficient pam\_rootok.so). Be sure to include this if you see errors when starting new R sessions.

The behavior that RStudio Workbench requires is essentially the same as that of the su command (impersonation of a user without a password). Therefore by default RStudio Workbench uses the /etc/pam.d/su profile for running R sessions.

The session PAM profile itself is also run whenever the user accesses the home page. Regardless of where the session actually runs (such as on another machine if using Job Launcher sessions),

the PAM profile is executed on the RStudio Workbench instance itself when opened via the home page.

#### Creating a Custom Profile

The /etc/pam.d/su profile has different default behavior depending upon your version of Linux and local configuration. Depending upon what type of behavior you want associated with R sessions (e.g. mounting of disks, setting of environment variables, enforcing of resource limits, etc.) you'll likely want to create a custom profile for R sessions. For example, if you wanted to use a profile named rstudio-session you would add this to the configuration file:

```
# /etc/rstudio/rserver.conf
auth-pam-sessions-profile=rstudio-session
```

Here is in turn what the custom profile might contain in order to enable a few common features of PAM sessions (this is based on a modified version of the default **su** profile on Ubuntu):

```
# /etc/pam.d/rstudio-session
# This allows root to su without passwords (this is required)
           sufficient pam_rootok.so
auth
# This module parses environment configuration file(s)
# and also allows you to use an extended config
# file /etc/security/pam_env.conf.
# parsing /etc/environment needs "readenv=1"
session
          required
                      pam_env.so readenv=1
# Locale variables are also kept into /etc/default/locale in etch
# reading this file *in addition to /etc/environment* does not hurt
session
          required pam_env.so readenv=1 envfile=/etc/default/locale
# Enforces user limits defined in /etc/security/limits.conf
          required
                      pam_limits.so
session
# The standard Unix authentication modules
@include common-auth
@include common-account
@include common-session
```

#### **Custom Profile with Passwords**

Note that in the above configuration we rely on pam\_rootok.so to enable authentication without a password. This is necessary because RStudio doesn't retain the passwords used during the authentication phase.

In some situations however passwords are important for more than just authentication. PAM profiles support a use\_first\_pass directive to forward passwords used during authentication into other modules (for example, to request a Kerberos ticket with pam\_krb5.so or to mount an encrypted or remote drive with pam\_mount.so). For these scenarios RStudio Workbench supports an optional mode to retain passwords after login and then forward them into the PAM session profile. This is enabled via the auth-pam-sessions-use-password setting:

```
# /etc/rstudio/rserver.conf
auth-pam-sessions-use-password=1
```

```
i Note
This setting is only available when using PAM authentication.
```

In this scenario you should remove the auth sufficient pam\_rootok.so directive and replace it with whatever authentication directives apply in your environment. You can then employ the use\_first\_pass directive to forward the password as necessary to other modules.

For example, here's a very simple RedHat/CentOS PAM configuration file that uses system default authentication and forwards the password into the pam\_mount.so module. Note that we are no longer using pam\_rootok.so because the password is now available when the session is created.

```
# /etc/pam.d/rstudio-session
# Auth/account (use system auth and forward password to pam mount)
auth
         include
                    system-auth
         optional
auth
                    pam_mount.so use_first_pass
account required
                    pam_unix.so
# Session (read environment variables and enforce limits)
session required
                    pam env.so readenv=1
                    pam env.so readenv=1 envfile=/etc/default/locale
session required
session required
                    pam_limits.so
```

Note that this configuration requires that RStudio Workbench retain user passwords in memory. This retention is done using industry best-practices for securing sensitive in-memory data including disabling ptrace and core dumps, using mlock to prevent paging into the swap area,

and overwriting the contents of memory prior to freeing it. When using Job Launcher, the passwords are securely transmitted in encrypted form to different nodes or containers running the R sessions.

## i Note

Launcher will require TLS/SSL to be configured in order to allow PAM profiles with passwords to be used.

#### More Resources

If you want to learn more about PAM profile configuration the following are good resources:

- http://www.linux-pam.org/Linux-PAM-html/Linux-PAM\_SAG.html
- http://linux.die.net/man/8/pam.d
- http://www.linuxjournal.com/article/2120
- http://www.informit.com/articles/article.aspx?p=20968

## 34.0.2. PAM Session Cleanup

By default, RStudio Workbench does not close PAM sessions when their associated R process exits. This is because PAM sessions often initialize and maintain resources that require more persistence that the lifetime of a single R session (e.g. mounted drives, Kerberos tickets, etc.). If a user has multiple active R sessions then closing the PAM session associated with one of them might unmount a drive or revoke a ticket that is still required by another R session.

It is however possible to manually close the PAM session associated with an R session by force suspending it. This can be accomplished in one of two ways:

- By pressing the **Suspend** button on the Sessions page of the Administrative Dashboard.
- By executing a force-suspend or force-suspend-all command as described in Suspending Sessions.

If you prefer that PAM sessions be closed whenever their associated R session exits you can use the auth-pam-sessions-close setting. For example:

```
# /etc/rstudio/rserver.conf
auth-pam-sessions-close=1
```

Note that if you specify this setting be aware that depending upon what resources are managed by your PAM sessions it may be incompatible with users running multiple concurrent R sessions (because for example a drive might be unmounted from underneath a running session). In this case you may wish to disable support for multiple R sessions (see the section on Multiple R Sessions for details on how to do this).

#### 34.0.3. PAM Sessions with the Job Launcher

When launching sessions via the Job Launcher, you must also configure PAM on any Launcher nodes (for Local or Slurm plugins) and within any containers used (if using Kubernetes). That's because PAM sessions are needed at the point where the R session is run, which for Launcher means some machine/container other than the one where RStudio Workbench is running. The PAM configuration is still required at the RStudio Workbench machine for authentication purposes.

Launcher session PAM profiles run with different permissions at different stages of the session launch pipeline. When a user visits the RStudio home page for the first time, their PAM session is initiated with root privilege, allowing the session to rely on pam\_rootok.so, as discussed in the previous sections, and it is opened on the RStudio Workbench instance itself. However, Job Launcher sessions themselves never have true root privilege (unlike traditional session launches which have root privileges during early initialization), and thus cannot rely on pam\_rootok.so. Their PAM sessions are opened on the actual instance where the session is running (e.g. a Kubernetes or Slurm node).

It is generally sufficient to simply do all your necessary setup, such as user directory creation, whenever the user accesses the home page for the first time. This means that you can rely on pam\_rootok.so and do not need password forwarding. However, if you need to also ensure that PAM sessions are created when actual sessions are started (e.g., for Kerberos or pam\_mount), you will need to enable auth-pam-sessions-use-password to forward the user's PAM credentials to the session.

## Note

When Kerberos or LDAP/Active Directory are required for PAM sessions, individual Launcher nodes or Kubernetes containers will have to either join the Kerberos domain or be configured to use sssd for obtaining LDAP/AD users and groups.

When forwarding credentials over a TCP connection, we require that Job Launcher connections be secured with SSL to prevent leaking of the credentials. Therefore, if auth-pam-sessions-use-password is set for Launcher sessions, you will also need to set launcher-use-ssl and configure the Job Launcher to use SSL (see the Job Launcher SSL Considerations section for more info).

The following sample configuration demonstrates forwarding of PAM user credentials to Job Launcher sessions. This is only recommended if it is absolutely necessary to start PAM sessions whenever Launcher sessions are started, and it is not sufficient to simply start a PAM session whenever the user accesses the homepage for the first time:

```
# /etc/rstudio/rserver.conf
launcher-sessions-enabled=1
launcher-use-ssl=1
auth-pam-sessions-enabled=1
auth-pam-sessions-profile=rstudio
auth-pam-sessions-use-password=1
```

The following sample configuration demonstrates a basic PAM setup where it is sufficient to only start PAM sessions when users visit the homepage, and thus no PAM sessions are started when Launcher sessions are started:

```
# /etc/rstudio/rserver.conf
launcher-sessions-enabled=1
auth-pam-sessions-profile=rstudio
```

## 34.0.4. Disabling PAM Sessions

If you don't want RStudio Workbench to utilize PAM sessions you can disable this feature using the auth-pam-sessions-enabled setting. For example:

```
# /etc/rstudio/rserver.conf
auth-pam-sessions-enabled=0
```

## △ Warning

When using <code>sssd</code> to automatically provision local system accounts using LDAP or Active Directory, RStudio relies on PAM sessions configured with <code>pam\_mkhomedir</code> (or equivalent) to create the home directories of a user that has never logged into the server. Disabling PAM sessions in this scenario may cause permission errors when starting session unless other methods for creating users' home directories are used.

## ! Important

PAM sessions are initially disabled by default when using RStudio Launcher to simplify its initial setup in a multi-node environment. However, many environments still require PAM sessions and in those cases that needs to be explicitly enabled with Launcher.

## 35. Kerberos PRO

You can use PAM sessions to arrange for Kerberos tickets to be made available for use by R sessions. This is accomplished using the pam\_sss PAM module. Note that you may need to install this module separately depending on which Linux distribution/version you are running.

## 35.0.1. Configuration

#### i Note

You should be sure to understand the previous section on PAM Sessions before attempting to modify your configuration to support Kerberos.

The following are simple examples of the pam\_sss and sssd.conf configuration directives you would need to add to use Kerberos with RStudio. Note that the sssd Kerberos backend supports a large number of options, some of which may be required to get Kerberos working correctly in your environment. You should consult the documentation before proceeding to ensure you've specified all options correctly.

## ! Important

If you are migrating your Kerberos settings from the now deprecated pam\_krb5 to pam\_sss, consult the pam\_krb5 migration documentation for additional information.

The main PAM profile for RStudio should be modified to include the following pam\_sss directives:

```
# /etc/pam.d/rstudio
auth sufficient pam_sss.so
account required pam_sss.so
session requisite pam_sss.so
```

In addition to modifying the main PAM profile, you will also need to create a custom PAM session profile for RStudio (as described in Creating a Custom Profile). This needs to include the appropriate pam\_sss directives. For example:

```
# /etc/pam.d/rstudio-session
auth    required    pam_sss.so
account    [default=bad success=ok user_unknown=ignore] pam_sss.so
password    sufficient    pam_sss.so use_authtok
session    requisite    pam_sss.so
```

Note that typically when you create a custom PAM session profile you include the auth sufficient pam\_rootok.so directive. However, in the case of configuring for Kerberos authentication you do not want this directive, rather you need to specify that authentication is done by Kerberos using an explicit password as illustrated in the above example.

To ensure that the custom PAM session profile is used by RStudio Workbench and that PAM passwords are correctly forwarded to pam\_sss you'll also need to add the following entries to the rserver.conf config file:

```
# /etc/rstudio/rserver.conf
auth-pam-sessions-profile=rstudio-session
auth-pam-sessions-use-password=1
```

Finally, you will need to specify Kerberos settings in sssd.conf (usually located at /etc/sssd/sssd.conf). For more information on SSSD configuration, see the sssd.conf documentation and the sssd-krb5 documentation.

```
# /etc/sssd/sssd.conf
[sssd]
services = nss, pam

# replace this with a comma-separated list of your configured SSSD domains
domains = TEST.EXAMPLE.COM

[domain/TEST.EXAMPLE.COM]
# can also be set to ad or local depending on your authentication setup
id_provider = ldap

auth_provider = krb5
# replace with the name of your Kerberos realm
krb5_realm = TEST.EXAMPLE.COM

# we recommend setting the debug level high to make troubleshooting easier
debug_level = 5
```

```
krb5_validate = true

# note that RHEL-7 default to KERNEL ccaches, which are preferred in most cases to FILE
krb5_ccachedir = /var/tmp

krb5_keytab = /etc/krb5.keytab
```

Some additional notes regarding configuration:

- The debug setting in sssd.conf is not required however we recommend adding it as it makes troubleshooting much more straightforward.
- The examples above are not *complete* examples but rather illustrations of the pam\_sss and sssd.conf entries that need to be present. Your local environment may have many additional entries which you should ensure are also included as necessary.

You should be sure to suspend active R sessions and to restart RStudio after making configuration changes to ensure that the new settings are being used. You can do this as follows:

```
sudo rstudio-server force-suspend-all
sudo rstudio-server restart
```

## 35.0.2. Testing and Troubleshooting

After making the required configuration changes you should test your updated PAM configuration in isolation from RStudio using the pamtester utility as described in Diagnosing PAM Authentication Problems. The following command will test both authentication as well as issuing of Kerberos tickets:

```
sudo /usr/lib/rstudio-server/bin/pamtester --verbose \
  rstudio-session <user> authenticate setcred open_session
```

Note that you should substitute an actual local username for the **<user>** part of the command line.

The specifics of both PAM configuration and Kerberos configuration can vary substantially by environment. As a result correct configuration likely requires additional entries and options which this guide isn't able to cover. Please refer to the documentation linked to in More Resources as well as the pam\_krb5 for additional details.

# 36. Directory Management

## 36.0.1. Working Directories

The default working directory for both new R sessions and new R projects is the user's home directory (~). You can change this behavior via the session-default-working-dir and session-default-new-project-dir configuration parameters within the rsession.conf config file.

For example, the set the default values to "~/working" and "~/projects" you'd use the following configuration:

```
# /etc/rstudio/rsession.conf
session-default-working-dir=~/working
session-default-new-project-dir=~/projects
```

You should ensure that users have the permissions required to write to the specified default directories. The specified directories will be automatically created if they don't already exist.

Note that these settings control only the *default* working and new project directories (users can still override these settings locally if they choose to).

### 36.0.2. Restricted Directories

Like most IDEs, RStudio allows users to open and edit any file to which they have read or write permission (respectively) at the filesystem level. However, in some environments it's important to restrict access to system files from web front ends. RStudio can optionally enforce a restrictive mode for most directories, which will prevent users from using the IDE to open files in the directories. This is enabled by the restrict-directory-view option as follows:

```
# /etc/rstudio/rsession.conf
restrict-directory-view=1
```

When enabled, RStudio users will only have access to open files from the following in the IDE:

• Home directories

- R and R library directories
- RStudio-specific directories
- User/session specific temporary directories

If you wish to allow users to open and view files in a directory that RStudio would ordinarily forbid access to, you can change the directory-view-whitelist setting. This setting accepts a list of directories to enable access to, separated by :. For example, if you wish to allow users to open files from /var/run and /usr/share/examples:

```
# /etc/rstudio/rsession.conf
directory-view-whitelist=/var/run:/usr/share/examples
```

Note that this setting only applies to RStudio's IDE web interface. Users will still have access to other files on the system using R itself and/or the Terminal interface. Follow security best practices by relying on operating system-level file permissions, not front end restrictions, to guard access to sensitive content and files.

# 37. Workspace Management

#### 37.0.1. Default Save Action

When a user exits an R session they need to choose whether to save their R workspace (i.e. .RData file). RStudio has global and per-project settings that control what happens when a workspace has unsaved changes. Possible values are:

- ask Ask whether to save the workspace file
- yes Always save the workspace file
- no Never save the workspace file

The default global setting is ask and the default project-level setting is derived from the current global setting (these options can be modified by end users via the *Global Options* and *Project Options* dialogs respectively).

The default global setting can also be changed via the session-save-action-default configuration parameter in the rsession.conf config file. For example, to change the default value to no you would use this:

```
# /etc/rstudio/rsession.conf
session-save-action-default=no
```

Note that this setting is specified in the rsession.conf config file and takes effect the next time a user launches an R session (rather than requiring a full restart of the server).

## 37.0.2. Suspend and Resume

When R sessions have been idle (no processing or user interaction) for a specified period of time (2 hours by default) RStudio suspends them to disk to free up server resources. When the user next interacts with their session it is restored from disk and the user resumes right back where they left off. This is all done seamlessly such that users aren't typically aware that a suspend and resume has occurred.

#### Session Timeout

To configure the amount of idle time to wait before suspending sessions you can use the session-timeout-minutes setting in the /etc/rstudio/rsession.conf file. For example:

```
# /etc/rstudio/rsession.conf
session-timeout-minutes=360
```

The default value if none is explicitly specified is 120 minutes.

## ! Important

This setting and a few others discussed in this section are specified in the /etc/rstudio/rsession.conf file (rather than the rserver.conf file previously referenced).

There are some conditions where an R session will not be suspended, these include:

- 1) When a top-level R computation is running
- 2) When the R prompt is not in it's default state (e.g. during a debugging session)

You can also specify that R sessions should never be suspended by setting the session-timeout-minutes to zero. For example:

```
# /etc/rstudio/rsession.conf
session-timeout-minutes=0
```

You can also set session timeouts on a per-user or per-group basis, see the User and Group Profiles section for details.

If you simply want the session process to quit (and lose all unsaved work in the process) instead of suspending to disk, you can turn the session-timeout-suspend option off, like so:

```
session-timeout-minutes=90
session-timeout-suspend=0
```

The above example will quit idle sessions after 90 minutes, discarding any unsaved data.

#### Forcing Suspends

You can force the suspend of individual sessions or even all sessions on the server. You can do this directly from the main page of the Administrative Dashboard or from the system shell as follows:

```
$ sudo rstudio-server force-suspend-session <pid>
$ sudo rstudio-server force-suspend-all
```

#### Resume and .Rprofile

By default the Rprofile.site and .Rprofile files are not re-run when a session is resumed (it's presumed that all of their side-effects are accounted for by simply restoring loaded packages, options, environment variables, etc.).

In some configurations it might be desirable to force the re-execution of profile files. There is an end user option that controls this on the *General* options pane which defaults to false. However, server administrators may wish to ensure that this option defaults to true. To do this you use the session-rprofile-on-resume-default option. For example:

```
# /etc/rstudio/rsession.conf
session-rprofile-on-resume-default=1
```

Note that this setting is specified in the rsession.conf config file and takes effect the next time a user launches an R session (rather than requiring a full restart of the server).

#### **Child Processes**

By default, when sessions are quit or suspended, child processes created in the session will continue to run. You can specify whether or not that should occur by specifying the session-quit-child-processes-on-exit setting in /etc/rstudio/rsession.conf. The allowed values are 1 or 0 to quit child processes or leave them running, respectively.

For example, to quit child processes when the session exits:

```
# /etc/rstudio/rsession.conf
session-quit-child-processes-on-exit=1
```

Note that this setting is specified in the rsession.conf config file and takes effect the next time a user launches an R session (rather than requiring a full restart of the server). Also, users can specifically override this setting in their project settings.

#### Session Timeout Kill

To configure the amount of time to wait before forcefully killing and destroying sessions you can use the session-timeout-kill-hours option in the /etc/rstudio/rsession.conf file. This allows you to specify how much time should pass before a session is automatically cleaned up. This is the length of time during which a session exists without user input (regardless of computation status) allowing you to automatically reclaim temporary disk space used by the session, and to stop its processes and children.

This setting should only be used to ensure that any sessions that users have forgotten about are destroyed, reclaiming valuable disk space.

## ! Important

Sessions can be destroyed while important computations are executing. The user whose session is cleaned will also lose all unsaved code and data.

This setting can also be used in conjunction with session-timeout-minutes, allowing already suspended sessions to be cleaned up.

For example:

```
# /etc/rstudio/rsession.conf
session-timeout-kill-hours=48
```

The default value if none is explicitly specified is 0 hours, meaning sessions will never be killed and destroyed automatically. The supplied value should be an integer representing the amount of hours a session can be idle before being killed.

### 37.0.3. Workspace Storage

Storage of workspaces (.RData files) in RStudio does not use compression by default. This differs from the behavior of base R. Compression is disabled because we've observed that for larger workspaces (> 50MB) compression can result in much lower performance for session startup and suspend/resume (on the order of 3 or 4 times slower).

The default workspace save options under RStudio are as follows:

```
options(save.defaults=list(ascii=FALSE, compress=FALSE))
options(save.image.defaults=list(ascii=FALSE, safe=TRUE, compress=FALSE))
```

If you wish to use different defaults you can define the save.defaults and/or save.image.defaults options in your Rprofile.site or per-user .Rprofile and RStudio will respect the settings you specify rather than using it's own defaults.

See https://stat.ethz.ch/R-manual/R-devel/library/base/html/save.html for additional details on how R saves objects and the storage and performance implications of using compression.

## 37.0.4. User State Storage

By default, RStudio stores each user's state in their home directory, in the following folder:

#### ~/.local/share/rstudio

This folder contains information on all of a user's active RStudio sessions, including all of the session data when the session is suspended. It also includes various internal RStudio state. Deleting it will result in a factory-fresh RStudio experience for the user, with the exception of their per-user settings (which are described in Customizing Session Settings).

Because this folder can include arbitrary amounts of suspended session data, it can become very large. We recommend ensuring that each user has sufficient disk space quota to store suspended sessions along with the data for their R projects, and using other controls to clean up old data (such as session-timeout-kill-hours)

## ! Important

RStudio 1.3 and earlier stored user state in the folder ~/.rstudio (non-configurable). If you are upgrading to RStudio 1.4 from an earlier release, RStudio will automatically move user state to the new location when the user starts their first IDE session after the upgrade. If there's a possibility you may downgrade to a prior RStudio release, we recommend configuring RStudio to use the old location temporarily; see below for instructions.

#### **Storage Location Customization**

If it is necessary to store user state in another folder to relieve disk quota pressure or comply with other requirements, you can customize it using the XDG\_DATA\_HOME environment variable. XDG\_DATA\_HOME must be set for the entire RStudio process tree, since RStudio needs to write and read user state both inside and outside R sessions (so attempting to set it in session startup scripts like Rprofile.site or rsession-profile is inadequate and may result in inconsistent behavior).

If your Linux distribution uses the systemd init system, run sudo systemctl edit rstudio-server. In the editor, add the following section to the file (replacing /mnt/storage

with your choice of root, of course). Note that the **rstudio** folder is not included in this path; this is a configuration root directory that will be respected by other applications that use the XDG standard.

```
[Service]
Environment="XDG_DATA_HOME=/mnt/storage/$USER"
```

If your Linux distribution does not use the systemd init system, consult the documentation for your Linux distribution to learn which init system it uses and the appropriate method for setting environment variables for the rstudio-server service.

In the example above, the state for the user bob would be stored in /mnt/storage/bob/rstudio. The following special variables are expanded in the value:

Variable	Expands To
\$USER \$HOME \$HOSTNAME	User's Unix username User's home directory Name of current host

The \$HOSTNAME value can be useful to avoid conflicts between hosts running distinct RStudio installations that share a file system, but note that \$HOSTNAME should only be used in installations wherein the RStudio services and R session are on the same host. It's important that all the machines in an installation agree on the path to the user state.

#### **Permissions Considerations**

You must ensure that the folder that hosts user state data is writable by all RStudio users, since the R session (running as the user) will create the state folder if it does not exist. In the example above, bob would need write access to /mnt/storage to create /mnt/storage/bob/rstudio.

Note that the user's default *umask* will be used to set permissions on this folder (i.e. RStudio does not attempt to set them). We recommend ensuring that user umasks are configured such that the folder, when created, will not be readable by other users. If you want the folder to be created with permissions other than those it would receive via umask, you can create it via some other means prior to a user's first interaction with RStudio; RStudio will not attempt to re-create the folder or change permissions if it already exists.

## Compatibility, Sharing, and Exact Locations

XDG\_DATA\_HOME affects many different XDG-compliant applications, and it sets a base (root) directory, to which /rstudio is appended to form the final path. If you wish to set the

exact path, or want to avoid side effects in other applications, use the environment variable RSTUDIO\_DATA\_HOME instead of XDG\_DATA\_HOME.

For example, RStudio 1.3 and prior stored user state in the folder ~/.rstudio. If you wish to continue using this folder in RStudio 1.4 and later, you could use the following setting (again, in sysctl edit rstudio-server):

```
[Service]
Environment="RSTUDIO_DATA_HOME=$HOME/.rstudio"
```

Note however that having two different RStudio installations with different versions of RStudio sharing a user state folder is unsupported and can lead to runtime errors or data corruption, as RStudio's internal state structure changes between versions. If you have multiple RStudio installations with different versions that are run by concurrently by users, we recommend using RSTUDIO\_DATA\_HOME to create distinct user state folders, ensuring that these installations don't corrupt each others' state.

If RSTUDIO\_DATA\_HOME and XDG\_DATA\_HOME are both set, RSTUDIO\_DATA\_HOME takes precedence.

#### **Job Launcher Mounts**

Finally, if you are using containerized sessions with the Job Launcher, and you mount RStudio's user state data to a folder outside their home directory, you must ensure that directory is mounted into the containers in addition to the home directory itself. See Launcher Mounts for more information about mounting folders into containerized sessions.

# 38. First Project Template

#### **38.0.1.** Overview

RStudio allows you to specify a first project to automatically open for first time users of the system. To do this, set the session-first-project-template-path parameter in rsession.conf to an RStudio project directory. This directory will be copied into the user's home directory upon first running the IDE, and will automatically open the project contained within. For example:

```
# /etc/rstudio/rsession.conf
session-first-project-template-path=/etc/rstudio/welcome-project
```

In the example above, the project located within /etc/rstudio/welcome-project would be copied to users' home directories when first starting RStudio, and the project file welcome-project.Rproj would be run.

The project file must be named the same as the directory it is contained in. For the above example, the project file must be named welcome-project.Rproj. In addition, you must ensure that the project directory is fully readable and executable by your users, as they will be copying the contents of the directory into their home directory.

If you copy an existing project to be used as a project template, ensure that you delete the RStudio metadata folders and files contained within the project directory. You will want to ensure you delete the following:

- .Rproj.user
- .Rhistory
- .RData

If you are creating the project template for the first time, the project (.Rproj) file must contain the version specifier at a minimum. For example:

```
# welcome-project.Rproj
Version: 1.0
```

## 38.0.2. Project DefaultOpenDocs

Project files allow you to specify default documents that should be opened when a project is opened for the very first time. For example, you could have your welcome project bring up explanatory documents to help guide your users. To do this, add the <code>DefaultOpenDocs</code> line to the .Rproj file. For example:

```
# welcome-project.Rproj
Version: 1.0
DefaultOpenDocs: welcome.txt:firstMarkdown.Rmd
```

The DefaultOpenDocs parameters specifies a list of files to automatically be opened when the project is opened for the first time, separated by a : character. These files are relative paths to the project directory. Only files contained within the project directory can be opened.

# 39. Project Sharing PRO

#### 39.0.1. Overview

Project Sharing is a feature of RStudio Workbench that enables users to work together on RStudio projects. When enabled, a project owner may select any number of other RStudio Workbench users as project collaborators. RStudio Workbench manages the permissions of files in the project to ensure that all collaborators can access them, enables multiple collaborators to edit a file simultaneously, and lets collaborators see who else is working inside a project with them.

### 39.0.2. Prerequisites

Due to high latencies, use of EFS (Elastic File System) for project sharing directories within AWS is strongly discouraged. If EFS is used, RStudio will experience highly degraded performance. We recommend using a traditional NFSv3 or NFSv4 mount instead.

The following sections outline additional prerequisites that your file system must meet.

#### **Access Control Lists**

To use Project Sharing, the directories hosting the projects to be shared must be on a volume that supports Access Control Lists (ACLs). RStudio uses ACLs to grant collaborators access to shared projects; ordinary file permissions are not modified.

Instructions for enabling ACLs vary by Linux distribution and filesystem type (see the Guide to enabling ACLs on Ubuntu or RedHat, for example). RStudio supports both NFSv3 (POSIX compatible) and NFSv4 ACLs.

#### **Shared Projects Root Directory**

RStudio cannot grant access to shared projects in arbritary locations. This would lead to unintended changes to the file system ACLs.

By default only projects stored within the users' home directories can be shared. If you want share projects in a different location you can configure RStudio with an additional root directory for all users:

```
# /etc/rstudio/rserver.conf
server-project-sharing-root-dir=/projects
```

## **Project Sharing and NFS**

If you plan to use Project Sharing with NFS-mounted volumes, there are several caveats you should be aware of.

- We recommend mounting NFS with the noac mount option. Without this option, NFS
  caches file attributes, so it may not be possible for users working simultaneously in a
  project to know whether they're seeing each others' latest changes. The noac option does
  reduce performance, however, so we recommend testing to choose the right trade off for
  your environment.
- Some features which automatically update when directory contents change will not update
  on NFS. For instance, users may need to manually refresh the Files pane to see new files
  added by collaborators.
- ACL support on NFS is more complicated than ACL support on local file systems. Read ACLs on Linux NFS for a discussion of the issues; guidance for configuring specific NFS versions is provided below.

**Determining NFS Version** Project Sharing works with versions 3 and 4 of the NFS protocol. However, some additional configuration may be necessary depending on the NFS version and underlying filesystem. To determine your NFS client version, run the following command on your RStudio Workbench instance:

```
$ nfsstat -m
```

You should see vers=3.0 in the output if you're using NFSv3, and vers=4.0 if you're using NFSv4. Extra RStudio configuration is required for NFSv4 clients (see below).

**NFSv3** To use NFSv3 access control lists, you will need to ensure that the filesystem is mounted with the acl attribute, and modify /etc/fstab if necessary to persist the attributes.

Note that many Linux distributions now have ACLs enabled by default in which case no special configuration is required. You can use the tune2fs command to inspect the attributes with which your filesystem is mounted (user\_xattr and acl are required for project sharing).

On most systems this is the default, so you need only ensure that noacl is not present. It's also necessary for the file system on the NFS server to itself be appropriately configured for ACL support; see the section above on Access Control Lists for guidance. Not all file servers that support the NFSv3 protocol also support POSIX compatible ACLs.

To test for POSIX compatible ACLs, you can use the setfacl command as follows:

```
setfacl -m u:user2:x /home/user1/project
```

where /home/user1/project is the full path to a directory owned by user1, and user2 is another valid user on the system.

Because many Linux systems support POSIX compatible ACLs on the native filesystem, RStudio uses them by default. You can also specify them explicitly as follows:

```
# /etc/rstudio/rsession.conf
nfs-acl-version=nfsv3
```

**NFSv4** Version 4 of the NFS protocol uses a very different permissions and ACL model from Version 3. Not all NFSv4 servers support or expose the NFSv4 ACL model, so check with the server administrator to determine whether the capability exists and/or can be enabled.

#### Testing for support

To test NFSv4 ACL support, you can use the nfs4\_setfacl command as follows:

```
nfs4_setfacl -a A::user2@domain:rax /home/user1/project
```

where /home/user1/project is the full path to a directory owned by user1, user2 is another valid user on the system, and domain is your system's user domain. Verify that this command succeeds, and that the new ACL entry for user2@domain is visible when you retrieve it:

```
nfs4_getfacl /home/user1/project
```

This presumes that your environment supports using user/domain pairs as security principals. While most environments do, there are two different ways to specify security principals in NFSv4 ACLs: by user name and domain (preferred) or user ID. RStudio can use either, but you'll need to choose one appropriate for your network environment. If unsure, examine the output of the nfs4\_getfacl operation described above. The security principal (the part after e.g., A:: on each line) will be displayed as either a large integer (UID) or in username@domain format.

#### Using user name and domain security principals

NFSv4 ACLs differ from NFSv3 ACLs in that they can associate a *domain* with each user named in the access control list. This is typically the same as the domain part of the machine's host name, but can be any string that the NFSv4 client and server agree on. On Linux systems, the domain can be set in /etc/idmapd.conf.

Tell RStudio the NFSv4 domain you want to use as follows:

```
# /etc/rstudio/rsession.conf
nfs-acl-version=nfsv4
nfs4-principal-type=username
nfs4-domain=mydomain.com
```

### Using user ID security principals

In some environments, it may not be possible to resolve fully qualified user@domain security principals on all of the nodes running RStudio Workbench. This is common on e.g., Kubernetes nodes which are not domain-joined. In this case, it's possible to use raw User ID (UID) values as security principals in the access control list. You can tell RStudio to write raw UIDs as security principals in NFSv4 access control lists as follows:

```
nfs-acl-version=nfsv4
nfs4-principal-type=uid
```

#### User Visibility

Projects can only be shared with users with provisioned local system accounts unless LDAP is used for local account provisioning.

The RStudio user database is scanned to list Project Sharing users. This method of user scanning avoids making calls to the system to determine the available users, and prevents making queries to the system password database (such as LDAP). However, it only lists users who have previously logged in to RStudio. To instead show all users on the server, set the following configuration to scan the system password database:

```
# /etc/rstudio/rsession.conf
project-sharing-enumerate-server-users=1
```

When scanning via the system password database, if you're using sssd with LDAP, you may need to enable user enumeration so that RStudio Workbench can search the directory to create a list of users you can share a project with. To do this, set the following in your sssd.conf file:

```
[domain/<domainname>]
enumerate = true
```

Alternatively, the auth-required-user-group setting can be used. This setting allows RStudio Workbench to enumerate only the members of the named groups rather than the entire user directory. Therefore, if you cannot enable user enumeration on your LDAP provider, you can instead create a group containing all RStudio users and supply it as the auth-required-user-group.

You can read more about user enumeration in the sssd FAQ.

#### **Shared Storage**

To use Project Sharing, a directory must be specified to which all users on the server can read and write. It must also have the sticky bit set, so that users cannot remove each others' content. In a single-server installation, RStudio uses this location by default:

```
/var/lib/rstudio-server/shared-storage
```

In a load-balanced configuration, however, RStudio does not provide a default, so it is necessary to provision a path both visible to and writable by all users on the system (typically on the same filesystem on which home directories are mounted). This path can be supplied to RStudio Workbench via the server-shared-storage-path option, for example:

```
# /etc/rstudio/rserver.conf
server-shared-storage-path=/shared/rstudio-server/shared-storage
```

The server-shared-storage-path option (described above) configures the path used for shared project storage. Note that this storage contains only *links* to shared projects, not the projects themselves, so requires a very small amount of physical storage.

Launcher Considerations When configuring RStudio Workbench to work with the RStudio Job Launcher, it is recommended to configure the Shared Storage path in a location that will be reachable both by the RStudio Workbench instance and each Launcher Session. See the Launcher Sessions section for more details.

## **Proxy Settings**

If you are running RStudio Workbench with a proxy, you'll need to make sure that your proxy is correctly configured to pass websocket connections through in order for all Project Sharing features to work. See the Running with a Proxy section for more on this.

## 39.0.3. Disabling Project Sharing

Project Sharing is enabled by default however you can disable it using the server-project-sharing option, for example:

```
# /etc/rstudio/rserver.conf
server-project-sharing=0
```

# 40. Package Installation

You can customize the location of user packages installed from CRAN as well as the default CRAN repository. You can also configure the user-interface of the RStudio IDE to discourage end-user package installation in the case where packages are deployed centrally to a site library.

## ! Important

The settings discussed in this section are specified in the /etc/rstudio/rsession.conf file (rather than the rserver.conf file previously referenced).

## 40.0.1. User Library

By default R packages are installed into a user-specific library based on the contents of the R\_LIBS\_USER environment variable (more details on this mechanism are here: http://stat.ethz.ch/R-manual/R-devel/library/base/html/libPaths.html).

It's also possible to configure an alternative default for user package installation using the r-libs-user setting. For example:

```
# /etc/rstudio/rsession.conf
r-libs-user=~/R/library/%v
```

R interprets %v to be the major.minor version of R; for example the above would cause R 3.6 to use a package library located at ~/R/library/3.6.

## **Unversioned R Library**

R doesn't guarantee ABI compatibility between minor versions, so when you use packages that include compiled code, you will want to ensure that each version of R has its own library. Otherwise, compiled code may cause crashes or other unexpected behavior when R is upgraded.

However, if you do not use any compiled packages, you can share a library between R versions (e.g. just ~/R/library). This makes it possible to upgrade the major version of R on the server and have user's packages continue to work.

#### Upgrading R

You can read more about package library considerations when upgrading R in the following article: Upgrading to a New Version of R

## 40.0.2. Discouraging User Installations

It may be that you've configured RStudio Workbench with a site package library that is shared by all users. In this case you might wish to discourage users from installing their own packages by removing the package installation UI from the RStudio IDE. To do this you use the allow-package-installation setting. For example:

```
# /etc/rstudio/rsession.conf
allow-package-installation=0
```

Note that this setting merely discourages package installation by removing user-interface elements. It's still possible for users to install packages directly using the utils::install.packages function.

### 40.0.3. CRAN Repositories

RStudio uses the RStudio CRAN mirror (https://cran.rstudio.com) by default. This mirror is globally distributed using Amazon S3 storage so should provide good performance for all locales. You may however wish to override the default CRAN mirror. This can be done with the r-cran-repos settings. For example:

```
# /etc/rstudio/rsession.conf
r-cran-repos=http://cran.at.r-project.org/
```

Whatever the default CRAN mirror is, individual users are still able to set their own default. To discourage this, you can set the allow-r-cran-repos-edit settings. For example:

```
# /etc/rstudio/rsession.conf
allow-r-cran-repos-edit=0
```

Note that even with user editing turned off it's still possible for users to install packages from alternative repositories by directly specifying the repos parameter in a call to install.packages.

To specify a list of CRAN repos, define a /etc/rstudio/repos.conf file containing the primary CRAN repo and named secondary repos. For example:

```
# /etc/rstudio/repos.conf
CRAN=https://cran.rstudio.com
Australia=https://cran.ms.unimelb.edu.au/
Austria=https://lib.ugent.be/CRAN/
```

To change the location of repos.conf, use the r-cran-repos-file setting. For example, by adding to rsession.conf:

```
r-cran-repos-file=/etc/rstudio/mirrors.conf
```

Optional CRAN repos can be made available for users in RStudio with the r-cran-repos-url setting, this setting expects a URL to retrieve a configuration file containing named secondary repos. These secondary repos won't be set by default, but rather, RStudio will list them to users to be manually added to their repo selection. For example:

http://custom-domain/repos.conf

```
Australia=https://cran.ms.unimelb.edu.au/
Austria=https://lib.ugent.be/CRAN/
```

Notice that the allow-r-cran-repos-edit option can also be used to disallow secondary repos and that repos.conf overrides the r-cran-repos setting, if specified.

# 41. Feature Limits PRO

RStudio Workbench has a number of other limits that can be configured. This section describes these limits. Note that these settings are specified in the /etc/rstudio/rsession.conf file (rather than the rserver.conf file previously referenced).

### 41.0.1. Disabling Access to Features

Besides the limits on package installation and CRAN repository editing described in the previous section there are a number of other limits that can be specified in /etc/rstudio/rsession.conf. The following describes all of the options that can be used in rsession.conf to limit features.

allow-vcs Allow access to Git and SVN version control features.

- allow-vcs-executable-edit Allow editing of the underlying Git or SVN executable.
- allow-package-installation Allow installation of packages using the Packages Pane (note that even if this is set to 0 it's still possible to install packages using utils::install.packages from the command line).
- allow-r-cran-repos-edit Allow editing of the CRAN repository used for package downloads (note that it's still possible to specify an alternate repository using the repos parameter of utils::install.packages).
- allow-shell Enable integrated terminal feature (note that it's still possible to execute shell commands using the system function).
- **allow-terminal-websockets** Allow integrated terminal feature to use WebSockets for better responsiveness.
- allow-file-downloads Allow downloading files using the Export command in the Files Pane.
- allow-file-uploads Allow uploading files using the Upload command in the Files pane.
- allow-external-publish Allow content to be published to external (cloud) services. This includes publishing HTML documents created with R Markdown or R Presentations to RPubs (http://rpubs.com), and publishing Shiny applications and documents to ShinyApps.io (http://shinyapps.io). Note that this just removes the relevant user interface elements in the IDE, and that it may still be possible for users to publish content using the R console.

allow-publish Allow content to be published. If specified, this option removes all user interface elements related to publishing content from the IDE, and overrides allow-external-publish.

allow-launcher-jobs Allow running standalone adhoc Job Launcher jobs from the Jobs pane.

All of these features are enabled by default. Specify 0 to disable access to the feature.

Note that these options should be specified in the /etc/rstudio/rsession.conf configuration file (rather than the main rserver.conf configuration file).

## 41.0.2. Maximum File Upload Size

You can limit the maximum size of a file upload by using the limit-file-upload-size-mb setting. For example, the following limits file uploads to 100MB:

```
# /etc/rstudio/rsession.conf
limit-file-upload-size-mb=100
```

The default behavior is no limit on the size of file uploads.

## 41.0.3. CPU Time per Computation

If you want to prevent runaway computations that consume 100% of the CPU you can set the maximum number of minutes to allow top-level R computations to run for using the limit-cpu-time-minutes setting. For example:

```
# /etc/rstudio/rsession.conf
limit-cpu-time-minutes=30
```

This specifies that no top level computation entered at the R console should run for more than 30 minutes. This constraint is implemented by calling the R setTimeLimit function immediately prior to handing off console commands to R. As a result it is possible for a particular script to override this behavior if it knows that it may exceed the threshold. This would be done as follows:

```
setTimeLimit(cpu = Inf)
# Long running R code here...
```

## 41.0.4. XFS Disk Quotas

If your system uses the XFS file system (http://en.wikipedia.org/wiki/XFS) then RStudio Workbench can be configured to notify users when they come close to or exceed their disk quota. You can enable this using the limit-xfs-disk-quota setting. For example:

```
# /etc/rstudio/rsession.conf
limit-xfs-disk-quota=1
```

The user's XFS disk quota will be checked when the RStudio IDE loads and a warning message will be displayed if they are near to or over their quota.

## 42. Notifications PRO

Administrators can broadcast notifications to user sessions in real-time using the notifications.conf file located at /etc/rstudio/notifications.conf. This file comes by default with commented out entries that you can uncomment and use, and helps show you the available time and message formats.

Each session monitors for changes in the notifications.conf file, and if a new notification is detected, it will be shown to the user at the appropriate time (as defined in the next section). All open sessions for a user will receive the notification, and they will continue to see the notification in any new sessions they open until the notification is acknowledged.

Modifying a notification will cause it to count as a new notification, so make sure to only save changes to the file when you've confirmed what you want the message to be and what time it should be displayed. Otherwise, the same message could be shown multiple times.

#### 42.0.1. notifications.conf format

The notifications.conf file is a file consisting of multiple notification entries separated by a blank line. The following table lists the fields that are available for each notification entry in the file.

StartTime	The start time at which the notification can start to be delivered.
	This must be a time-formatted field. This field is not required.
EndTime	The end time at which the notification will no longer be delivered.
	This must be a time-formatted field. This field is required.
Message	The message content to show to the users. The message cannot
	have empty lines in it. This field is required.

An example notifications.conf file is shown below. For more information on the formatting of each field, see the subsequent sections.

```
# /etc/rstudio/notifications.conf
StartTime: 2017-08-30 09:00:00 -5:00
EndTime: 2017-08-30 20:00:00 -05:00
Message: Please remember to shut down your computers at the end of the day.
```

StartTime: 2017-05-30 05:55

EndTime: 2017-06-13

Message: This is a test notification. Notifications can span multiple lines by indenting the next line's message text.

Empty lines are not supported!

It is important that each entry consists of 2-3 fields as specified above (StartTime, EndTime, and Message). Each field must go on its own line. There should be no empty lines between field definitions.

For example, this is okay:

# /etc/rstudio/notifications.conf
StartTime: 2017-08-30 12:00:00 -5:00
EndTime: 2017-08-30 20:00:00 -05:00
Message: Please remember to shut down your computers at the end of the day.

But this is not:

# /etc/rstudio/notifications.conf
StartTime: 2017-08-30 12:00:00 -5:00
EndTime: 2017-08-30 20:00:00 -05:00
Message: Please remember to shut down your computers at the end of the day.

There must be **one** empty line (2 new line characters) in between separate notification entries.

For example, this is okay:

# /etc/rstudio/notifications.conf
StartTime: 2017-08-30 08:00:00 -5:00
EndTime: 2017-08-30 20:00:00 -05:00
Message: Please remember to shut down your computers at the end of the day.

StartTime: 2017-08-30 08:00:00 -5:00 EndTime: 2017-08-30 20:00:00 -05:00

Message: Remember to drop off any borrowed equipment at Grace's office today only.

But this is not:

```
# /etc/rstudio/notifications.conf
StartTime: 2017-08-30 12:00:00 -5:00
EndTime: 2017-08-30 20:00:00 -05:00
```

Message: Please remember to shut down your computers at the end of the day.

StartTime: 2017-08-30 12:00:00 -5:00 EndTime: 2017-08-30 20:00:00 -05:00

Message: Remember to drop off any borrowed equipment at Grace's office today only.

#### Time format

The time format fields, StartTime and EndTime, must be in one of the following formats:

YYYY-MM-DD

YYYY-MM-DD hh:mm

YYYY-MM-DD hh:mm:ss zh:zm

The following table shows the meaning of the format sections.

4 digit year (example: 2017)
2 digit month (example: 06)
2 digit day (example: 28)
2 digit hours (24 hour clock. example: 19)
2 digit minutes (example: 15)
2 digit seconds (example: 59)
Time zone hours offset (example: -06 for CST or
-08 for PST)
Time zone minutes offset (usually just 00,
different for only a few timezones)

If no time is specified, the time will be set to 00:00:00 in the current server time zone for start times and 23:59:59 in the current server time zone for end times.

If no seconds are specified, they will be set to 00 and the time is interpreted in the current server time zone.

Note that if you have sessions spanning multiple servers in different time zones and you want your notifications to display at a uniform time, you MUST manually set the timezone to what is appropriate. Otherwise, sessions in different time zones will see notifications in their local time.

The following table shows some example dates and how they would be formatted.

January 1st, 2020 at 6:00 PM in the server's time zone	2020-01-01 18:00
July 31st, 2018 at midnight in the server's time zone (for a	2018-07-31
start time)	
September 23rd, 2019 at 23:59:59 in the server's time zone	2019-09-23
(for an end time)	
November 30th, 2020 at 9:14:12 in Pacific Standard Time	2020-11-30 09:14:12 -08:00

## Message format

The message to deliver must be plain text and cannot have any empty lines. To start text on another line, simply indent the line as in the multiline example in the previous section.

# 43. RStudio Connect Server

Users of RStudio can publish content to RStudio Connect. To do so, they must first specify the RStudio Connect server they wish to use. You can set the default RStudio Connect server URL to use when users are connecting to an account. To do so, use the default-rsconnect-server option:

```
# /etc/rstudio/rsession.conf
default-rsconnect-server=http://connectserver/
```

# 44. Terminal WebSockets

The RStudio terminal pane uses WebSockets to communicate between the web browser and the session. If the attempt to connect with WebSockets fails, the terminal switches to a less responsive but generally more forgiving HTTP-based protocol.

Slow typing response in the terminal is a symptom of this slower protocol. You can check by starting a terminal, then going to Tools / Terminal / Terminal Diagnostics. Scroll down to *Connection Information* and look for "WebSocket connect error, switching to RPC". This indicates that the terminal was unable to use WebSockets and has fallen back on the slower protocol.

Several settings are available for tuning the terminal's use of WebSockets.

# 44.0.1. IDE Settings

In the IDE under Tools / Global Options / Terminal, there is a checkbox "Connect with WebSockets". Unchecking this will cause terminals to always connect with the HTTP-based protocol.

#### 44.0.2. Feature Limit

As mentioned earlier, the feature limit allow-terminal-websockets completely disables the use of WebSockets.

## 44.0.3. Advanced Settings

The following settings may be modified by adding them to the /etc/rstudio/rsession.conf file.

Setting	Default	Purpose
websocket-ping-seconds	10	How often a keep-alive is sent over the WebSocket, in seconds. Set to 0 to disable keep-alives. Many proxies will close inactive WebSockets so keeping enabled is recommended.
websocket-connect-timeout	3	How long terminal waits (in seconds) for WebSocket to connect before switching to the HTTP protocol. If set to 0 then the timeout of the web browser's WebSocket implementation will be used; this is often quite lengthy (minutes).
websocket-log-level	0	Controls logging of WebSocket diagnostics. These are for troubleshooting only, and will appear in the RStudio R Console pane while enabled.
		<ul> <li>0 = no WebSocket logging</li> <li>1 = log WebSocket errors</li> <li>2 = log WebSocket activity</li> <li>3 = log WebSocket errors and activity</li> </ul>
websocket-handshake- timeout	5000	How long the server waits (in milliseconds) for WebSocket handshake to complete when connecting and disconnecting. Set to 0 to disable.

For example, this would double the number of keep-alive packets sent, wait longer before switching to HTTP, log information on WebSocket errors, and increase the handshake timeout to 15 seconds.

```
websocket-ping-seconds=5
websocket-connect-timeout=10
websocket-log-level=1
websocket-handshake-timeout=15000
```

For more background on the terminal feature, see this support article.

# 45. Customizing Sessions Settings

All of the session settings file locations described in this section conform to the XDG Base Directory Specification. They are configurable using environment variables:

Scope	Default	Environment
User System	~/.config/rstudio /etc/rstudio	XDG_CONFIG_HOME XDG_CONFIG_DIRS

In accordance with the *Base Directory Specification*, the environment variables specify the location of the rstudio folder. For example, to store system-wide preference configuration in /var/config/rstudio/rstudio-prefs.json, you would set the XDG\_CONFIG\_DIRS variable to the value /var/config.

If specified, the RStudio variables take precedence over the XDG variables. These variables specify a specific directory (not a base directory). For example, to store system-wide preferences in /var/config/settings/rstudio-prefs.json, you would set the RSTUDIO\_CONFIG\_DIR variable to the value /var/config/settings.

The examples in this section presume you're setting system-wide settings in /etc/rstudio; in each case it's also possible to use a different folder by changing environment variables as described above, or to apply the settings to individual user accounts by changing files in ~/.config/rstudio.

#### 45.0.1. User Preferences

User preferences set in the RStudio IDE's *Global Options* dialog can also be set in the JSON file rstudio-prefs.json, located in the settings directory described above.

#### **Schema**

The schema for the JSON file can be found at:

/usr/lib/rstudio-server/resources/schema/user-prefs-schema.json

It documents all of the preferences, shows data types and allowable values, and briefly explains the usage of each. You can see a summary of this information in the Session User Settings appendix.

#### Example

By default, RStudio Workbench only shows the home page (session overview) to users who have multiple sessions running. If you'd like it to be shown to all users regardless of the number of running sessions, set it in the global user preferences file as follows:

/etc/rstudio/rstudio-prefs.json

```
{
    "show_user_home_page": "always"
}
```

# 45.0.2. Snippets

You can install global snippets files for all users in the /etc/rstudio/snippets folder. For example, if you'd like to create a snippet lib for an R library call:

```
# /etc/rstudio/snippets/r.snippets
snippet lib
library(${1:package})
```

You can also define snippets for CSS files in the file css.snippets, and so on. You can find documentation on the snippet file format in the Cloud 9 IDE snippet documentation.

Note that RStudio will not merge snippet files, which implies the following:

- If you define your own snippets (for a given file type), they will replace those that ship with RStudio (for that same file type).
- If users define their own snippets (for a given file type), changes to the system snippet file (for that same file type) won't have any effect on those users.

## 45.0.3. Default Document Templates

RStudio typically opens new documents with completely blank contents. You can, however, define the contents of the blank document by creating a file named default.X in /etc/rstudio/templates, where X is the file extension you wish to customize. For example, to start all R scripts with a standard comment header users can fill out, you could use the following:

There are also some special template files which ship with RStudio; these, too, are customizable. In /etc/rstudio/templates, you can customize the following:

File	Description
document.Rmd	The default R Markdown document file content (without YAML header)
notebook.Rmd presentation.Rmd	The default R Notebook file content (without YAML header) The default R Markdown presentation file content (without YAML header)
shiny.Rmd query.sql	The default Shiny R Markdown file content (without YAML header) The default SQL query

#### 45.0.4. Color Themes

You can define additional custom themes for RStudio by placing .rstheme files in the following directory:

#### /etc/rstudio/themes

The .rstheme file contains plain-text CSS with some special metadata. You can create one by importing an existing TextMate theme file, or by starting from scratch (using an existing theme file as a template). Run the R command ?rstudioapi::addTheme for more help.

#### 45.0.5. Fonts

RStudio's code editor and R console use a fixed-width font. By default, only fonts that end users have installed locally can be selected. If you wish to make additional fixed-width fonts available to your users, you can place them here:

#### /etc/rstudio/fonts

Fonts placed here will be automatically made available for selection in RStudio's Appearances settings (Tools -> Global Options -> Appearance) for all users. It's helpful to place the fonts preferred by your users here because it allows the font to be used in RStudio regardless of what fonts they have installed locally.

The following font formats are supported:

- Web Open Font Format (.woff, .woff2)
- OpenType (.otf)
- Embedded OpenType (.eot)
- TrueType (.ttf)

Only fixed-width fonts are supported by the RStudio IDE. Proportional fonts will still be installed, but if users select a proportional font, they will experience cursor positioning problems.

#### Naming and Directory Structure

The name of the file is presumed to be the name of the font. If you wish to give the font a custom name, you can place it in a directory with your name of choice. For example:

This directory structure would make two fonts available, Coding-Font and Coding Font Two.

Some fonts come in many different weights and styles. If you want these weights and styles to be treated as single font, you can place them underneath a single folder. This is useful when a theme uses bold or italic variants of a font to decorate code (e.g., to set comments in italics).

To do this, create subfolders with the font's weight or style as the folder's name. For example, this creates a single font, "Coding Font 3", which has two weights (400 and 700 for regular and bold, respectively) and an italic style for each weight.

#### **Autodetection**

In addition to displaying a list of fonts installed on the server, RStudio attempts to automatically detect available fixed-width fonts that are installed on a user's browser. For security reasons, it is not possible for RStudio to enumerate all the fonts on the user's machine, so a known list of popular programming fixed-width fonts are checked for compatibility. This list is stored in the option browser\_fixed\_width\_fonts.

If your users have a font they prefer but it can't be installed on the server, you can cause RStudio to start scanning for it locally by including it in the set of browser\_fixed\_width\_fonts in the global RStudio preferences file, /etc/rstudio/rstudio-prefs.json. See User Preferences for more information on setting global options.

## 45.0.6. Keybindings

RStudio keybindings can be globally defined using the following two files:

```
/etc/rstudio/keybindings/editor_commands.json
/etc/rstudio/keybindings/rstudio_commands.json
```

It isn't necessary to hand-author these files; RStudio can generate them for you:

- 1. Remove the ~/.config/rstudio/keybindings/ folder
- 2. Start a new R session and customize the keyboard shortcuts as desired
- 3. Copy the new .json files from ~/.config/rstudio/keybindings to /etc/rstudio/keybindings to make them active for all users on the server

# **45.0.7.** Spelling

You can define additional spelling dictionaries for RStudio by placing dictionary files in the following folders:

## Languages

Define additional system languages by placing Hunspell .aff files in:

/etc/rstudio/dictionaries/languages-system

#### **Dictionaries**

Define additional custom dictionaries by placing Hunspell.dic files in:

/etc/rstudio/dictionaries/custom

# Part VI. R Versions

# 46. Overview

# 46.1. R Versions

RStudio Workbench enables users and administrators to have very fine-grained control over which versions of R are used in various contexts. Capabilities include:

- Administrators can install several versions of R and specify a global default version as well as per-user or per-group default versions.
- Users can switch between any of the available versions of R as they like.
- Users can specify that individual R projects remember their last version of R and always use that version until explicitly migrated to a new version.

Flexible control over R versions makes it much easier to provide upgraded versions of R for users (or individual projects) that require them; while at the same time not disrupting work that requires continued use of older versions.

# 47. Installing R

We recommend installing multiple versions of R because an environment with multiple versions of R helps provide a stable, reproducible environment for your R developers.

Install R using the directions at https://docs.rstudio.com/resources/install-r/.

#### 47.0.1. Recommended Installation Directories

RStudio Workbench automatically scans for versions of R at the following locations:

```
/usr/lib/R
/usr/local/lib/R
/usr/local/lib64/R
/opt/local/lib/R
/opt/local/lib64/R
```

In addition, RStudio Workbench scans all subdirectories of the following directories within /opt:

```
/opt/R
/opt/local/R
```

For example, any of the following installed versions of R will be automatically detected by RStudio Workbench:

```
/opt/R/3.1.0
/opt/R/3.2.0
/opt/local/R/3.1.0
/opt/local/R/3.2.0
```

If you have versions of R located at other places in the file system, RStudio Workbench can still utilize them. However, you'll need to explicitly specify their location in a configuration file (this is covered in more detail in the Using Multiple Versions section).

# 48. Configuring the Default Version of R

When multiple versions of R are installed you will need to specify which version is the default one for new R sessions. This can be done automatically via the system PATH. However, several other mechanisms are provided when more flexibility is required.

## 48.0.1. Single Default Version of R

RStudio uses the version of R pointed to by the output of the following command:

#### \$ which R

The which command performs a search for the R executable using the system PATH. RStudio will therefore by default bind to the same version that is run when R is executed from a terminal.

For versions of R installed by system package managers this will be /usr/lib/R. For versions of R installed from source this will typically (but not always) be /usr/local/lib/R.

If you want to override which version of R is used, then you can use the rsession-which-r setting. For example:

```
# /etc/rstudio/rserver.conf
rsession-which-r=/usr/local/lib/R
```

Note: This change will not take effect until the server is restarted.

#### 48.0.2. Default Version Per User or Group

You can use the User and Group Profiles feature to specify distinct default versions of R for various users and groups. For example, the following profile configuration uses R 3.1.0 as the system default, R 3.2.0 for the powerusers group, and R 3.0.2 for the user jsmith:

```
[*]
r-version = /opt/R/3.1.0

[@powerusers]
r-version = /opt/R/3.2.0

[jsmith]
r-version = /opt/R/3.0.2
```

Note that r-version specifies the full path to the directory where R is installed.

# 48.0.3. User Configurable Default Version

Users can also configure their own default version of R. This is done using the **General** pane of the **Global Options** dialog:

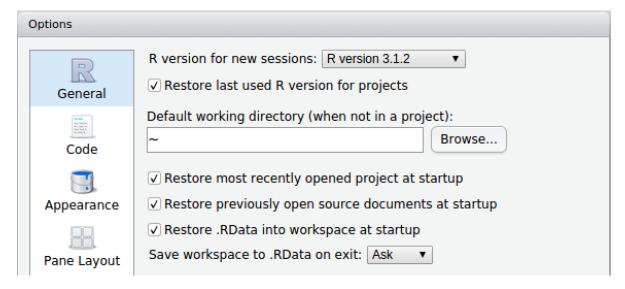


Figure 48.1.: Set Default R Version

See the Disabling Use of Multiple Versions section for details on how to disable version switching entirely either system-wide or on a per-user or per-group basis.

# 49. Using Multiple Versions of R Concurrently

# 49.0.1. Determining Available Versions

RStudio Workbench scans for and automatically discovers versions of R in the following locations:

```
/usr/lib/R
/usr/local/lib/R
/usr/local/lib64/R
/opt/local/lib/R
/opt/local/lib64/R
/opt/local/lib64/R
/opt/R/*
```

This is described in more detail in the Recommended Installation Directories section. If you have installed versions of R in alternate locations, you can list them within the /etc/rstudio/r-versions configuration file. For example:

```
# /etc/rstudio/r-versions
/opt/R-3.2.1
/opt/R-devel-3.3
```

In addition, any version of R referenced in an r-version directive within User and Group Profiles is also recognized.

In order to be usable, the R home path must be readable by the RStudio server account (usually rstudio-server; see Access and Security for details).

## Version Scan Report

At startup, RStudio Workbench collects information about each available R version as described above, and writes a report to the following file in JSON format:

/var/lib/rstudio-server/r-versions

This file is later read by the various RStudio processes responsible for displaying and switching between R versions. If you aren't seeing the R versions look and work as you expect, the content of this file can give you some insight into RStudio's understanding of your system's configuration. If you're unable to resolve the issue, include the contents of the file when submitting a support ticket to support@rstudio.com.

Note that the JSON format of the **r-versions** file is subject to change between RStudio versions, so avoid reading or using it in automated tooling.

The r-versions file must be available on all nodes that run R sessions. We don't generally recommend changing its location, but if you need to do so in order to mount it on R session nodes, you can do via the r-versions-path option as in the following example:

```
# /etc/rstudio/rserver.conf
r-versions-path=/mnt/config/rstudio-server/r-versions
```

#### **Extended R Version Definitions**

The /etc/rstudio/r-versions file allows you to specify extended information for a particular R Version, providing you:

- The ability to specify additional environment variables to set
- An optional preload script to run
- An optional environment module to load (more info here)
- A user-friendly label name for the version that is displayed in the UI

To specify extended format information, modify the /etc/rstudio/r-versions file to consist of multiple R entries separated by a blank line. The following table lists the fields that are available for each R entry in the file.

(Required if Module not specified, see Modules) The root
directory of the location of the R installation.
(Optional) The user-friendly name for the R version that will be
displayed to users in the UI.
(Optional) The name of an environment module to load for the R
version. This is loaded by running the command module load
[module] after sourcing the user's .bashrc file.
(Optional) A script to run once the environment has been loaded
but before the session process has been launched.
(Optional) A string representing a CRAN Repository URL, or
the path to a repos.conf file which lists multiple package
repositories. See CRAN Repositories for more information.

Library	(Optional) A: separated list of directories which house the
	desired R packages for the particular R version. Overrides the
	R_LIBS_SITE environment variable. This will be combined with
	R_LIBS_USER when forming the R library paths. Most R
	installations use a default site library located at
	\$R_HOME/site-library, so you may need to include the default
	site library path directories when setting this field.

An example /etc/rstudio/r-versions file is shown below.

```
# /etc/rstudio/r-versions
Path: /opt/R/R-2.15.3
Label: My special R Version
Module: testmodule
Script: ~/rload.sh
Repo: https://cran.ms.unimelb.edu.au/
Library: /share/packages/R-2.15.3

Path: /opt/R/R-2.15.3-alternate
Label: My special R Version 2

Module: r/latest
Label: Latest version of R

/opt/misc/R/SpecialR1
/opt/misc/R/SpecialR2
/opt/mic/R/AltnerateR
```

It is important that each entry consists of the fields as specified above. Each field must go on its own line. There should be no empty lines between field definitions.

Each R entry must be separated by one full blank line (two new-line \n characters). If only the path is being specified, with no label, script, or module, you may simply list the path to the installation (as in previous versions). Paths are not separated by a blank line, but they must be separate from extended definitions by a blank line (as in the above example).

**Modules** By setting the name of a module in an environment definition, that version of R will be loaded entirely by module. When a module is defined and the Path is not specified, the default R binary on the path will be used once the module is loaded. Otherwise, if Path is specified, that specific binary will be used.

If you do not specify a Path to the R installation, you must ensure that RStudio Workbench can load the module by specifying the location of the module shell initialization script for sh. For example:

```
# /etc/rstudio/rserver.conf
modules-bin-path=/usr/local/Modules/3.2.9/init/sh
```

Failure to do so will result in RStudio Workbench being unable to verify the version, which will cause it to be unavailable for use.

## **Reloading Configuration**

In order for the changes to the /etc/rstudio/r-versions file to be detected, you must either restart RStudio (via sudo rstudio-server restart) or send the SIGHUP message to the rserver process. This can be done using kill -s SIGHUP to the server process, or via the reload command:

```
sudo rstudio-server reload
```

#### **Excluding Versions**

If you have versions of R on your system that would normally be picked up by automatic scanning but which you'd like to exclude, the most straightforward thing to do is to disable R version scanning altogether and explicitly specify all versions you'd like to use in /etc/rstudio/r-versions. For example:

```
# /etc/rstudio/rserver.conf
r-versions-scan=0
```

## 49.0.2. Switching Between Versions

To switch between versions of R you use the version menu near the top right of the IDE:

After switching, the specified version will be used for the duration of the current session (see the section on Multiple R Sessions for more details on the lifetime of sessions). Newly created R sessions will continue to use whatever default R version has been configured for the user.



Figure 49.1.: Switching Versions

#### **Preserving Versions for Projects**

It's often useful to preserve the version used within an R project irrespective of whatever the current default R version is for a user. This is in fact the behavior by default for RStudio projects however can be changed from the **General** pane of the **Global Options** dialog.

This configuration enables users to easily migrate projects one-by-one to a new version of R after it's been confirmed that all the code continues to work as expected under the new version.

#### 49.0.3. Disabling Use of Multiple Versions

If you want to prevent users from being able to change R versions entirely you can use the r-versions-multiple option:

```
# /etc/rstudio/rserver.conf
r-versions-multiple=0
```

You can also configure this on a per-user or per-group basis by specifying the r-versions-multiple option within User and Group Profiles.

# 50. Managing Upgrades of R

There are various ways to handle upgrades to new versions of R ranging from allowing each user to control exactly when they upgrade all the way to forcing everyone to upgrade all at once.

By combining the various options described above you can create a highly customized upgrade policy that reflects both your internal policies and the preferences of your users.

# 50.0.1. User Controlled Migration

The most conservative approach is to start with a default version of R and preserve that default for the lifetime of the server. In this configuration you can continue to install new versions of R as they are released however users won't ever run those new versions unless they make an explicit gesture to do so. See the User Configurable Default Version and Switching Between Versions sections for details on how users can explicitly switch versions.

# 50.0.2. Partial Migration

If your posture towards new R versions is that you'd like users to migrate to the new version(s) as quickly as is convenient you can be more aggressive in how you introduce them. In this scenario you might use the Default Version Per User or Group feature to migrate a portion of new users immediately but preserve older versions for those who request it.

Note that in this scenario R projects will still preserve their previous R version so long as users have enabled the option described in Preserving Versions for Projects.

#### 50.0.3. Full Migration

The most aggressive approach is to force all users to upgrade to the new R version immediately (this is essentially what happens in the open-source version of RStudio Server). To implement this you'd set a Single Default Version of R as well as disabling the use multiple versions as described in Disabling Use of Multiple Versions.

Note that via User and Group Profiles you could also have a subset of R users that are always fully migrated to new versions while preserving user controlled migration or partial migration for others.

# 51. Session Initialization

In the preceding sections, we described how to allow the server to discover multiple versions of R to be used by the user. The R environment is loaded before the session begins execution, and several files are executed along the way that can cause problems with your R Environment if any variables are incorrectly set.

When a session is launched, the following steps occur:

- 1. If the file /etc/rstudio/rsession-profile exists, it is sourced.
- 2. If the R Version being loaded specifies a module to load, the user's ~/.bashrc file is sourced and the module is loaded using the module load command.
- 3. If the R version being loaded specifies a prelauch script to run, it is sourced.
- 4. A bash login shell is created, which sources the ~/.bash\_profile, ~/.bash\_login or the ~/.profile script, whichever is found first to exist (in that order).
- 5. The session is launched with the environment constructed in steps 1-4.

As you can see, there are many different scripts which can affect the R environment which gets initialized when starting a new session, so play close attention to how these scripts could be affecting your particular environment.

# Part VII. Load Balancing

# 52. Overview

# 52.1. Load Balancing

RStudio Workbench can be configured to load balance R sessions across two or more nodes within a cluster. This provides both increased capacity as well as higher availability.

Load balancing with RStudio Workbench always operates in an active-active fashion where all nodes are equally equipped to serve users. All nodes have a primary role.

Note that load balancing for RStudio Workbench has some particular "stickiness" requirements stemming from the fact that users must always return to the same R session where their work resides (i.e. their traffic can't be handled by more than one node). As a result, it's not enough to simply place multiple RStudio Workbench servers behind a conventional hardware or software load balancer—additional intelligence and routing is required.

Key characteristics of the RStudio Workbench load balancer include:

- 1. Multiple primary nodes for high availability all nodes can balance traffic to all other nodes.
- 2. Support for several load balancing strategies including least busy server (by active sessions or system load), even distribution by user, or a custom strategy based on an external script.
- 3. The ability to add and remove nodes while the cluster is running.
- 4. Works standalone or can be integrated with other front-end load balancing environments.

#### i Note

The standalone load balance and higher availability characteristics of RStudio Workbench are an exception among RStudio products. RStudio Connect and Shiny Server will require a front-end load balancing under the same scenarios. The use of an external load balancer can still be beneficial in a failover setup. See External Load Balancers for details.

# 53. Load Balancing vs. Job Launcher

The Job Launcher is another method supported by RStudio Workbench to achieve **increased capacity** by allowing sessions to run using a compatible computing infrastructure (i.e. Kubernetes). However, the Job Launcher does not aim to provide **higher availability**. At least two RStudio Workbench nodes in a load balancing configuration are still required to provide service continuity in failover scenarios.

# i Note

The Job Launcher itself can have its own load balancing strategy in place. See the loading balancing section in the Job Launcher documentation for more details.

# 54. Configuration

There are several requirements for nodes within RStudio clusters:

- 1. All nodes must run the same version of RStudio Workbench.
- 2. Server configurations (i.e. contents of the /etc/rstudio directory) must be identical, with the exception of options related to the address of each node (www-host-name in load-balancer, www-address and www-port in rserver.conf).
- 3. User accounts must be accessible from each node and usernames and user ids must be identical on all nodes. The same applies for any groups used by RStudio users, and also to the rstudio service user account.
- 4. The clocks on all nodes must be synchronized.
- 5. User home directories must be accessible via **shared storage** (e.g. all nodes mounting the same NFS volume).

#### i Note

Due to high latencies, use of EFS (Elastic File System) for home directories within AWS is strongly discouraged. If EFS is used, RStudio will experience highly degraded performance. We recommend using a traditional NFSv3 or NFSv4 mount instead.

- 6. An explicit server-wide shared storage path also must be defined. See the Shared Storage section for additional details.
- 7. RStudio must be configured to use a PostgreSQL database, and an empty database must be present for RStudio to write important cross-node state. If you have previously run RStudio with a SQLite database, it is strongly advised that you execute the database Migration to the PostgreSQL database first. For more information, see Database.

## 54.0.1. Defining The Cluster

Only one load balancing cluster can exist per database, and this cluster is defined by the first node that comes online within the cluster. The cluster data contains the hash of the secure cookie key and the communication protocol (http, https, or https no verify). When each node comes online, it verifies its own secure cookie key and protocol against the cluster's data and will only come online if this data matches. There are two ways to reset the data stored in the cluster:

- 1. Bring all nodes offline; then reconfigure each node. The first node that comes online will be able to update the cluster data.
- 2. Manually reset the cluster by running rstudio-server reset-cluster from the command line. The next node that is started, restarted, or reloaded will update the cluster data.

To view the nodes and their current statuses in the load balancer cluster, run the command rstudio-server list-nodes. The output contains a column 'Last Seen', which can be used to indentify nodes that have unexpectedly gone offline. When a node is online, it reguarly updates this database column with the current time (UTC) to indicate that it is Online and working as expected. When the time is displayed from the list-nodes command, it is displayed in the server's local time. If a node is not properly shutdown, it may have an Online Status but the 'Last Seen' column will not be recent.

#### 54.0.2. Defining Nodes

To define a cluster node, two configuration files need to be provided:

```
/etc/rstudio/load-balancer
/etc/rstudio/secure-cookie-key
```

The first of these defines the load balancing strategy and the node's public-facing address. The second defines a shared key used for signing cookies (in single-node configurations this key is generated automatically, however with multiple nodes explicit coordination is required. The same secure-cookie-key value **must** be used on each node).

Each setting in the load balancing configuration file has a default value, so the file may be empty, but its presence is required to activate load balancing.

When load balancing is configured, during startup each node will query the internal database for information about the active cluster and nodes. If the relevant data doesn't exist for a particular node, that node will insert it. It will then alert existing nodes of its presence and configuration. The address that it can be reached at is determined by attempting the following strategies, and using the first that is successful:

- 1. Use the value www-host-name provided in the configuration file.
- 2. Use the www-address defined in rserver.conf in combination with www-port or the default port.
- 3. Retrieve address from the hostname system call.

4. Use a system call to determine the machine's IP addresses and use the last v4, non-loopback address provided.

Most users will want to configure RStudio Workbench to use one of the first two approaches.

For example, to use the www-host-name option to define a cluster with two nodes that load balances based on the number of actively running R sessions you could use the following configuration:

On the first node, which can be reached at server1.example.com:

```
# /etc/rstudio/load-balancer
balancer=sessions
www-host-name=server1.example.com
```

On the second node, which can be reached at server2.example.com:

```
# /etc/rstudio/load-balancer
balancer=sessions
www-host-name=server2.example.com
```

```
# /etc/rstudio/secure-cookie-key
a55e5dc0-d6ae-11e3-9334-000c29635f71
```

The secure cookie key file above is only an example; you need to generate your own unique key to share among the nodes in your cluster.

#### i Note

Previous versions of RStudio Workbench required the host name of each node be included on every active node under a [nodes] title in lieu of the www-host-name field, and a [config] title prior to the balancing options. This configuration will continue to work, but it is no longer the preferred configuration method. It is highly recommended that you update your config files accordingly.

## 54.0.3. Key File Requirements

The following are the requirements for the secure cookie key file:

• The key value must have a minimum length of 128 bits (16 bytes/characters). RStudio won't start if the key is too weak.

- The key file must have restrictive permissions (i.e. 0600) to protect its contents from other users.
- The key file must be identical on all nodes in a load-balanced cluster, so that the nodes can communicate with each other.
- The key must have a secret value that cannot be guessed. Randomly generating the value is recommended; see below for one mechanism for doing so.

# 54.0.4. Generating a Key

You can create a secure cookie key using the uuid utility as follows:

```
$ sudo sh -c "echo `uuid` > /etc/rstudio/secure-cookie-key"
$ sudo chmod 0600 /etc/rstudio/secure-cookie-key
```

This is the recommended method, but any mechanism that generates a unique, random value will work.

You do not need to generate a secure-cookie-key file on each server; generate it once, and copy it to each node along with the rest of the /etc/rstudio directory.

This secure cookie key will also be used for encrypting and decrypting the PostgreSQL database password, if applicable. See PostgreSQL Password Encryption for more details.

#### **Key File Location**

You may optionally change the path of the secure-cookie-key by changing the secure-cookie-key-file setting in rserver.conf, though it is not necessary. Changing the path in this manner is only recommended in very specific circumstances when running the launcher with both RStudio Workbench and Package Manager simultaneously. For example:

```
# /etc/rstudio/rserver.conf
secure-cookie-key-file=/mnt/rstudio/secure-cookie-key
```

In addition, an explicit server-wide shared storage path must be defined (this is used for inter-node synchronization). This path is defined in the /etc/rstudio/rserver.conf file. For example:

```
# /etc/rstudio/rserver.conf
server-shared-storage-path=/shared/rstudio-server/shared-storage
```

For convenience, this path will often be located on the same volume used for shared home directory storage (e.g. at path /home/rstudio-server/shared-storage).

#### 54.0.5. Launcher Considerations

If you are running RStudio Workbench load balancing in addition to using Launcher sessions, you will need to ensure that the /etc/rstudio/launcher.pub and /etc/rstudio/launcher.pem files match on all Workbench nodes in the cluster. Failure to do so will prevent users from being able to connect to their sessions from Workbench nodes other than where their sessions were initiated.

For more information, see RStudio Workbench Integration.

#### 54.0.6. File Locking

In order to synchronize the creation of sessions across multiple nodes RStudio Workbench uses a cross-node locking scheme. This scheme relies on the clocks on all nodes being synchronized. RStudio Workbench includes a locktester utility which you can use to verify that file locking is working correctly. To use the locktester you should login (e.g. via SSH or telnet) to at least two nodes using the same user account and then invoke the utility from both sessions as follows:

#### \$ /usr/lib/rstudio-server/bin/locktester

The first node you execute the utility from should indicate the types of locks it was able to acquire, for example:

- \* Acquired advisory lock
- \* Acquired link-based lock

After the message is printed the process will pause so that it can retain the lock (you can cause it to release the lock by interrupting it e.g. via Ctrl+C).

The second and subsequent nodes you execute the utility will attempt to acquire the lock. A message will be printed to the console indicating which type of locks are supported, for example:

- \* Acquired advisory lock
- \* Unable to acquire link-based lock

Your filesystem appears to support link-based locks.

In this example, advisory locks are not supported (because both nodes were able to acquire an advisory lock), but link-based locks are. See Lock Configuration for more information on configuring lock types.

If you interrupt the first node (e.g. via Ctrl+C) the lock will be released and you can then acquire it from the other nodes.

If either of the following occurs then there is an issue with file locking capabilities (or configuration) that should be addressed prior to using load balancing:

- 1) All nodes successfully acquire the file lock (i.e. more than one node can hold it concurrently).
- 2) No nodes are able to acquire the file lock.

If either of the above conditions hold then RStudio won't be able to correctly synchronize the creation of R sessions throughout the cluster (potentially resulting in duplicate sessions and lost data due to sessions overwriting each others state).

#### **Lock Configuration**

RStudio's file locking scheme can be configured using a file at /etc/rstudio/file-locks. Valid entries are:

- lock-type=[linkbased|advisory]
- refresh-rate=[seconds]
- timeout-interval=[seconds]
- enable-logging=[0|1]
- log-file=[path]

The default locking scheme, linkbased, uses a file locking scheme whereby locks are considered acquired when the process successfully hardlinks a dummy file to a location within the folder RStudio uses for client state (typically ~/.local/share/rstudio). This scheme is generally more robust with older network file systems, and the locks should survive temporary filesystem mounts / unmounts.

The timeout-interval and refresh-rate options can be used to configure how often the locks generated in the linkbased locking scheme are refreshed and reaped. By default, a process refreshes any locks it owns every 20 seconds, and scans for stale locks every 30 seconds. If an rsession process crashes, it can leave behind stale lock files; those lock files will be cleaned up after they expire by any newly-launched rsession processes.

advisory can be selected to use advisory file locks (using e.g. fcntl() or flock()). These locks are robust, but are not supported by all network file systems.

If you are having issues with file locking, you can set enable-logging=1, and set the log-file option to a path where output should be written. When logging is enabled, RStudio will report

its attempts to acquire and release locks to the log file specified by log-file. When log-file is unset, log entries will be emitted to the system logfile, typically located at /var/log/messages or /var/lib/syslog.

#### 54.0.7. Managing Nodes

## Starting Up

After creating your configuration files you should ensure that these files (along with all other configuration defined in /etc/rstudio) are copied to all nodes in the cluster. Assuming that the server is already installed and running on each node, you can then apply the load balancing configuration by restarting the server:

```
$ sudo rstudio-server restart
```

#### **Current Status**

Once the cluster is running you can inspect its state (which sessions are running where) using the load balancing status HTTP endpoint. For example, when running the server on the default port (8787):

```
$ curl http://localhost:8787/load-balancer/status
```

Note that the status endpoint is accessed using localhost rather than an external IP address. This is because this endpoint is IP restricted to only be accessible within the cluster, so needs to be accessed directly from one of the nodes.

The status endpoint will return output similar to the following:

This output will show all of the nodes in the cluster. Each node is indicated by its address and an optional status indicating whether the node is unreachable or offline. If the node does not indicate a status, then it is healthy and servicing requests. Following the node address is its CPU Load, indicated by three decimal values indicating the last known 1-minute, 5-minute, and 15-minute load averages, represented as a fraction of total CPU load. On subsequent output lines, each RStudio IDE session that is running on that particular node is listed along with its process ID and running user.

An unreachable node indicates an issue connecting to it via the network. In most cases, this indicates that the rstudio-server service is not running on the node and should be troubleshooted by viewing any startup issues in the system logs for that particular node (see Diagnostics if the service is running and healthy). An offline node is one that was specifically put into offline mode via the command sudo rstudio-server offline, which causes it to stop servicing new sessions.

#### **Adding and Removing Nodes**

To temporarily remove a node from the cluster you can simply stop it:

#### \$ sudo rstudio-server stop

R sessions running on that node will be automatically moved to another active node. Note that only the session state is moved, not the running processes. The node will now appear in the list-nodes command with an offline status. To restore the node you can simply start it back up again:

#### \$ sudo rstudio-server start

To add a new node, create the file /etc/rstudio/load-balancer. Leave it empty for default settings. When the rstudio-server is restarted, it will broadcast its arrival to the other online nodes in the cluster. They do not have to be restarted or reloaded. All nodes sharing a database will be part of the same cluster.

You can suspend any actively running sessions by running sudo rstudio-server suspend-all on the node to be removed.

Reloading the load balancer configuration will also cause the rserver-http proxy configuration to be updated as well, which affects the RStudio's running HTTP server. It is recommended that you do not make any other HTTP-related changes when updating the load balancer configuration unless you are aware of the potential side-effects!

To permanently remove a node from the database, first stop restudio server on that node. From an active node, retrieve the to-be-deleted node's ID, then pass it to the delete-node command. For example, your commands may look like the following:

```
$ sudo rstudio-server list-nodes
Cluster
-----
Protocol
Http
Nodes
ID Host
                                   IPv4
                                                   Port
                                                           Status
                                                   8787
                                                           Failed to resolve
1
    rsw-primaryyy
                                   123.456.78.100 8787
2
    rsw-secondary
                                                           Online
3
                                   123.456.78.101 8787
                                                           Online
    rsw-primary
$ sudo rstudio-server delete-node 1
Node 1 deleted.
```

#### i Note

The output from the rstudio-server list-nodes command above was shortened to improve readability.

When the command is run, the node's database status will shortly change to 'Deleting' and then the node will be removed from the database. All other nodes in the cluster will be notified that this node has been removed and stop routing messages to it.

## 54.0.8. Troubleshooting

If users are having difficulty accessing RStudio in a load balanced configuration it's likely due to one of the load balancing requirements not being satisfied. This section describes several scenarios where a failure due to unsatisfied requirements might occur.

#### Node network instability

Some scenarios may causes RStudio to wait a long time for a node to respond due to network instability. You can limit how long is this waiting period with the timeout option, which is set to 10 seconds by default. This disable this timeout and use the system defaults, set it to zero.

```
# /etc/rstudio/load-balancer
[config]
balancer=sessions
timeout=5
...
```

#### SSL

If one of the nodes is temporarily using a self-signed or otherwise functional but invalid certificate the load balancer may fail to use that node. You can skip SSL certificate verification by disabling the option verify-ssl-certs, which is only applicable if connecting over HTTPS. For production use, you should always leave the default or have this set to true, but it can be disabled for testing purposes.

```
# /etc/rstudio/load-balancer
[config]
balancer=sessions
verify-ssl-certs=0
...
```

#### **User Accounts Not Synchronized**

One of the load balancing requirements is that user accounts must be accessible from each node and usernames and user ids must be identical on all nodes. If a user has the same username but *different* user ids on different nodes then permissions problems will result when the same user attempts to access shared storage using different user-ids.

You can determine the ID for a given username via the id command. For example:

```
$ id -u jsmith
```

## **NFS Volume Mounting Problems**

If NFS volumes containing shared storage are unmounted during an RStudio session that session will become unreachable. Furthermore, unmounting can cause loss or corruption of file locks (see section below). If you are having problems related to accessing user directories then fully resetting the connections between RStudio nodes and NFS will often resolve them. To perform a full reset:

- 1) Stop RStudio on all nodes (sudo rstudio-server stop).
- 2) Fully unmount the NFS volume from all nodes.
- 3) Remount the NFS volume on all nodes.
- 4) Restart RStudio on all nodes (sudo rstudio-server start).

#### File Locking Problems

Shared user storage (e.g. NFS) must support file locking so that RStudio can synchronize access to sessions across the various nodes in the cluster. File locking will not work correctly if the clocks on all nodes in the cluster are not synchronized. This condition may be surfaced as 502 HTTP errors. You can verify that file locking is working correctly by following the instructions in the File Locking section above.

#### **Diagnostics**

To troubleshoot more complicated load balancing issues, RStudio can output detailed diagnostic information about internal load balancing traffic and state. You can enable this by using the diagnostics setting as follows:

```
[config]
diagnostics=tmp
```

Set this on *every* server in the cluster, and restart the servers to apply the change. This will write a file /tmp/rstudio-load-balancer-diagnostics on each server containing the diagnostic information.

The value stderr can be used in place of tmp to send diagnostics from the rserver process to standard error instead of a file on disk; this is useful if your RStudio Workbench instance runs non-daemonized.

# 55. Access and Availability

Once you've defined a cluster and brought it online you'll need to decide how the cluster should be addressed by end users. There are two distinct approaches to this:

- 1. **Single Node Routing**. Provide users with the address of one of the nodes. This node will automatically route traffic and sessions as required to the other nodes. This has the benefit of simplicity (no additional software or hardware required) but also results in a single point of failure.
- 2. **Multiple Node Routing**. Put the nodes behind some type of system that routes traffic to them (e.g. dynamic DNS or a software or hardware load balancer). While this requires additional configuration it also enables all of nodes to serve as points of failover for each other.

Both of these options are described in detail below.

## 55.0.1. Single Node Routing

In a Single Node Routing configuration, you designate one of the nodes in the cluster as the main one and provide end users with the address of this node as their point of access. For example:

```
[nodes]
rstudio.example.com
rstudio2.example.com
rstudio3.example.com
```

Users would access the cluster using http://rstudio.example.com. This node would in turn route traffic and sessions both to itself and the other nodes in the cluster in accordance with the active load balancing strategy.

Note that in this configuration the **rstudio2.example.com** and **rstudio3.example.com** nodes can either fail or be removed from the cluster at any time and service will continue to users. However, if the main node fails or is removed then the cluster is effectively down.

# 55.0.2. Multiple Node Routing

In a Multiple Node Routing configuration all of the nodes in the cluster are peers and provide failover for each other. This requires that some external system (dynamic DNS or a load balancer) route traffic to the nodes; see below for examples and caveats. In this scenario any of the nodes can fail and service will continue, so long as the external router can respond intelligently to a node being unreachable.

For example, here's an Nginx reverse-proxy configuration that you could use with the cluster defined above:

```
http {
   upstream rstudio-server {
     server rstudio1.example.com;
     server rstudio2.example.com backup;
     server rstudio3.example.com backup;
}
server {
   listen 80;
   location / {
       proxy_pass http://rstudio-server;
       proxy_redirect http://rstudio-server/ $scheme://$host/;
   }
}
```

In this scenario the Nginx software load balancer would be running on **rstudio.example.com** and reverse proxy traffic to **rstudio1.example.com**, **rstudio2.example.com**, etc. Note that one node is designated by convention as the main one so traffic is routed there by default. However, if that node fails then Nginx automatically makes use of the backup nodes.

This is merely one example as there are many ways to route traffic to multiple servers—RStudio Workbench load balancing is designed to be compatible with all of them.

#### 55.0.3. External Load Balancers

When using an external load balancer with a Multiple Node Routing configuration, the external load balancer may be configured as active/active or active/passive.

RStudio Workbench load balances all requests internally in an active/active way, deciding where new sessions will be started, and routing requests to existing sessions, regardless which RStudio node received the initial request from the external load balancer. The RStudio node that receives the request will re-route the request appropriately. Therefore, the external load balancer does not determine which RStudio node will respond to the request.

- External load balancer configured as active/passive: All requests are routed by the external load balancer to a single RStudio node. If that node becomes unavailable or unresponsive, the external load balancer will select a different RStudio node. The RStudio node may route the request to another node to handle the request. The external load balancer provides failover / high availability, while RStudio Workbench's load balancer provides scalability across nodes.
- External load balancer configured as active/active: Per above, RStudio Workbench's internal load balancer may re-route the request to another node. Consequently, having the external load balancer select different nodes per request will not actually help balance the session load. Again, the external load balancer provides high availability, while scalability is still provided by the internal load balancer.

## 55.0.4. Using SSL

If you are running RStudio Workbench on a public facing network then using SSL encryption is strongly recommended. Without this all user session data is sent in the clear and can be intercepted by malicious parties.

The recommended SSL configuration depends on which access topology you've deployed.

### Single Node Routing

For a Single Node Routing deployment, you would configure each node of the cluster to use SSL as described in the Secure Sockets (SSL) section. The nodes will then use SSL for both external and intra-machine communication.

# i Note

In this configuration, you must ensure that your load-balancer file lists the hostname in the same format listed on host's SSL certificate in the Common Name (CN) or Subject Alternative Name (SAN) field, so that the nodes are able to validate each others' certificates when connecting.

#### Multiple Node Routing

For a Multiple Node Routing deployment, you would configure SSL within the external routing layer (e.g. the Nginx server in the example above) and use standard unencrypted HTTP for the individual nodes. You can optionally configure the RStudio nodes to use SSL as well, but this is not strictly required if all communication with outside networks is done via the external routing layer.

# 56. Balancing Methods

There are four methods available for balancing R sessions across a cluster. The most appropriate method is installation specific and depends on the number of users and type of workloads they create.

#### **56.0.1.** Sessions

The default balancing method is sessions, which attempts to evenly distribute R sessions across the nodes of the cluster:

```
[config]
balancer = sessions
```

This method allocates new R sessions to the node with the least number of active R sessions. This is a good choice if you expect that users will for the most part have similar resource requirements.

## 56.0.2. System Load

The system-load balancing method distributes sessions based on the active workload of available nodes:

```
[config]
balancer = system-load
```

The metric used to establish active workload is the 5-minute load average, divided by the number of cores on the machine. This is a good choice if you expect widely disparate CPU workloads and want to ensure that machines with high CPU utilization don't receive new sessions.

#### 56.0.3. User Hash

The user-hash balancing method attempts to distribute load evenly and consistently across nodes by hashing the username of clients:

```
[config]
balancer = user-hash
```

The hashing algorithm used is CityHash, which will produce a relatively even distribution of users to nodes. This is a good choice if you want the assignment of users/sessions to nodes to be stable.

#### 56.0.4. Custom

The custom balancing method calls out to external script to make load balancing decisions:

```
[config]
balancer = custom
```

When **custom** is specified, RStudio Workbench will execute the following script when it needs to make a choice about which node to start a new session on:

```
/usr/lib/rstudio-server/bin/rserver-balancer
```

This script will be passed two environment variables:

RSTUDIO\_USERNAME — The user on behalf or which the new R session is being created.

RSTUDIO\_NODES — Comma separated list of the host and port of available nodes.

The script should return the node to start the new session on using its standard output. Note that the format of the returned node should be identical to its format as passed to the script (i.e. include the host and port).

#### **Node Host Format**

In earlier versions of RStudio, the custom load balancing script would always be passed a list of raw IP addresses in RSTUDIO\_NODES; now, RSTUDIO\_NODES will contain the hosts as specified in the load-balancer file. If you want to specify host names in your load-balancer file but work with raw IPs in your custom load balancing script, you can set the following option:

# /etc/rstudio/rserver.conf
resolve-load-balancer-nodes=1

Note that this option is incompatible with SSL unless your servers' SSL certificates contain IP addresses in their  $\rm CN/SAN$ .

# Part VIII. Auditing and Monitoring

# 57. Overview

# 57.1. Auditing and Monitoring

RStudio Workbench can be configured to audit R console input and output for per user and session. It can monitor runtime resource use via the Administrative Dashboard or through integration with a system like Graphite.

# 58. Auditing Configuration

# 58.0.1. R Console Auditing

RStudio Workbench can be optionally configured to audit all R console activity by writing console input and output to a central location (the /var/lib/rstudio-server/audit/r-console directory by default). This feature can be enabled using the audit-r-console setting. For example:

```
# /etc/rstudio/rserver.conf
audit-r-console=input
```

This will audit all R console *input*. If you wish to record both console input and output then you can use the all setting. For example:

```
# /etc/rstudio/rserver.conf
audit-r-console=all
```

Note that if you choose to record both input and output you'll need considerably more storage available than if you record input only. See the Storage Options section below for additional discussion of storage requirements and configuration.

#### **Data Format**

The R console activity for each user is written into individual files within the r-console data directory (by default /var/lib/rstudio-server/audit/r-console). The following fields are included:

session_id	Unique identifier for R session where this action occurred.
project	Path to RStudio project directory if the action occurred within a
	project.
pid	Unix process ID where this console action occurred.
username	Unix user which executed this console action.
timestamp	Timestamp of action in milliseconds since the epoch.
type	Console action type (prompt, input, output, or error).
data	Console data associated with this action (e.g. output text).

The session\_id field refers to a concurrent R session as described in the section on Multiple R Sessions (i.e. it can span multiple projects and/or pids).

The default format for the log file is CSV (Comma Separated Values). It's also possible to write the data to Newline Delimited JSON by using the audit-r-console-format option. For example:

```
audit-r-console-format=json
```

Note that when using the JSON format the entire file is not a valid JSON object but rather each individual line is one. This follows the Newline Delimited JSON specification supported by several libraries including the R **jsonlite** package.

#### **Storage Options**

You can customize both the location where audit data is written as well as the maximum amount of data to log per-user (by default this is 50 MB). To specify the root directory for audit data you use the audit-data-path setting. For example:

```
# /etc/rstudio/rserver.conf
audit-data-path=/audit-data
```

Note that this path affects the location of both R console auditing and R session auditing data.

To specify the maximum amount of data to write to an individual user's R console log file you use the audit-r-console-user-limit-mb setting. For example:

```
# /etc/rstudio/rserver.conf
audit-r-console-user-limit-mb=100
```

The default maximum R console log file size is 50 megabytes per-user. To configure no limit to the size of files which can be written you set the value to 0, for example:

```
# /etc/rstudio/rserver.conf
audit-r-console-user-limit-mb=0
```

If you wish for RStudio to automatically roll the log files once the maximum size is reached, set the audit-r-console-user-limit-months setting. For example:

```
# /etc/rstudio/rserver.conf
audit-r-console-user-limit-months=2
```

This will cause log files to be rolled over once the maximum size is reached, and only two months of data will be kept. Note that this setting is not set by default.

Note that if the month limit is not set, then log files will not be rolled automatically. Depending on the number of users and their activity level this means that you should either create a scheduled (e.g. cron) job to periodically move the files off the server onto auxiliary storage and/or ensure that the volume they are stored on has sufficient capacity.

### 58.0.2. R Session Auditing

RStudio Workbench can be optionally configured to write an audit log of session related events (e.g. login/logout, session start/suspend/exit) to a central location (the /var/lib/rstudio-server/audit/r-sessions directory by default). This feature can be enabled using the audit-r-sessions setting. For example:

```
# /etc/rstudio/rserver.conf
audit-r-sessions=1
```

Note that this is enabled by default if you are using named user licenses.

## i Note

Session auditing is only supported for RStudio IDE R Sessions and is not currently supported for Jupyter or VS Code sessions.

#### **Data Format**

The R session event log is written by default to the file at /var/lib/rstudio-server/audit/r-sessions/r-sess. The following fields are included:

pid	Unix process ID the event is associated with (for auth events this
	will be the main rserver process, for session events the
	rsession process).
username	Unix user that the event is associated with.
timestamp	Timestamp of event in milliseconds since the epoch.
type	Event type (see documentation on event types below).
data	Event data (see documentation on event types below).

The following values are valid for the event type field:

auth_login	User logged in to RStudio Workbench; data contains name of
	admin user if they are impersonating the user
auth_throttled	User temporarily blocked due to multiple login attempts (as
	defined by the option auth-sign-in-throttle-seconds)
auth_unlicensed	User is locked or there is no license available
auth_license_failed	User blocked due to a failure in obtaining a license
auth_logout	User logged out of RStudio Workbench
auth_login_failed	User login attempt failed because a local account may not exist
session_start	R session started
session_suicide	R session exiting due to suicide (internal error)
session_suspend	R session exiting due to suspend
$session\_file\_upload$	File uploaded from browser to R session; data field contains file
	name
$session\_file\_download$	File downloaded from R session to browser; data field contains
	file name
session_quit	R session exiting due to user quit
session_exit	R session exited
session_admin_suspend	Administrator attempt to suspend R session; data field contains
	administrative user who initiated the event
$session\_admin\_terminate$	Administrator attempt to terminate R session; data field
	contains administrative user who initiated the event

The default format for the log file is CSV (Comma Separated Values). It's also possible to write the data to Newline Delimited JSON by using the audit-r-sessions-format option. For example:

#### audit-r-sessions-format=json

Note that when using the JSON format the entire file is not a valid JSON object but rather each individual line is one. This follows the Newline Delimited JSON specification supported by several libraries including the R **jsonlite** package.

# Storage Options

You can customize both the location where audit data is written as well as the maximum amount of R session event data to log (by default this is 1 GB). To specify the root directory for audit data you use the audit-data-path setting. For example:

```
# /etc/rstudio/rserver.conf
audit-data-path=/audit-data
```

Note that this path affects the location of both R console auditing and R session auditing data.

To specify the maximum amount of R session event data to log you use the audit-r-sessions-limit-mb setting. For example:

```
# /etc/rstudio/rserver.conf
audit-r-sessions-limit-mb=2048
```

The default maximum R session event log file size is 1 GB (1024 MB). To configure no limit to the size of files which can be written you set the value to 0, for example:

```
# /etc/rstudio/rserver.conf
audit-r-sessions-limit-mb=0
```

If you wish for RStudio to automatically roll the log files once the maximum size is reached, set the audit-r-sessions-limit-months setting. The default is set to 13 months. To set it manually, for example:

```
# /etc/rstudio/rserver.conf
audit-r-sessions-limit-months=13
```

This will cause log files to be rolled over once the maximum size is reached, and only thirteen months of data will be kept. We do not recommend you change this setting if using named user licenses.

Note that if the month limit is not set, then log files will not be rolled automatically. This means that you should either create a scheduled (e.g. cron) job to periodically move the file off the server onto auxiliary storage and/or ensure that the volume that it is stored on has sufficient capacity.

In any case, the amount of data written to the R session event log file is not large (less than 1 KB per session) so a large number of session events can be stored within the default 1 GB maximum log file size.

# 59. Monitoring Configuration

### 59.0.1. System and Per-User Resources

RStudio Workbench monitors the use of resources (CPU, memory, etc.) on both a per-user and system wide basis. By default, monitoring data is written to a set of RRD files and can be viewed using the Administrative Dashboard.

The storage of system monitoring data requires about 20MB of disk space and the storage of user monitoring data requires about 3.5MB per user. This data is stored by default at /var/lib/rstudio-server/monitor. If you have a large number of users you may wish to specify an alternate volume for monitoring data. You can do this using the monitor-data-path setting. For example:

```
# /etc/rstudio/rserver.conf
monitor-data-path=/monitor-data
```

You also might wish to disable monitoring with RRD entirely. You can do this using the monitor-rrd-enabled setting. For example:

```
# /etc/rstudio/rserver.conf
monitor-rrd-enabled=0
```

Note that changes to the configuration will not take effect until the server is restarted.

#### **Analyzing RRD files**

The RRD files powering RStudio's Administrative Dashboard are available for your own analysis, too. You can find them in /var/lib/rstudio-server/monitor/rrd (unless you've changed monitor-data-path as described above); they store all the metrics you can see on the dashboard, so you can use the information for your own reports and insights.

More information on how to read and visualize RRD data from R is available in the following blog post:

Reading and analysing log files in the RRD database format

#### 59.0.2. Using Graphite

If you are managing several servers it might be convenient to send server monitoring data to a centralized database and graphing facility as opposed to local RRD files. You can do this by configuring the server to send monitoring data to Graphite (or any other engine compatible with the Carbon protocol). This can be done in addition to or entirely in place of RRD.

There are four settings that control interaction with Graphite:

```
monitor-graphite-enabledWrite monitoring data to Graphite (defaults to 0)
monitor-graphite-host Host running Graphite (defaults to 127.0.0.1)
monitor-graphite-port Port Graphite is listening on (defaults to 2003)
monitor-graphite-client-Optional client ID for sender
```

For example, to enable Graphite monitoring on a remote host with the default Graphite port you would use these settings:

```
# /etc/rstudio/rserver.conf
monitor-graphite-enabled=1
monitor-graphite-host=134.47.22.6
```

If you are using a service like hosted graphite.com that requires that you provide an API key as part of reporting metrics you can use the monitor-graphite-client-id setting. For example:

```
# /etc/rstudio/rserver.conf
monitor-graphite-enabled=1
monitor-graphite-host=carbon.hostedgraphite.com
monitor-graphite-client-id=490662a4-1d8c-11e5-b06d-000c298f3d04
```

Note that changes to the configuration will not take effect until the server is restarted.

# 60. Server Health Checks

# 60.0.1. Enabling Health Checks

You may wish to periodically poll RStudio Workbench to ensure that it's still responding to requests as well as to examine various indicators of server load. You can enable a health check endpoint using the server-health-check-enabled setting. For example:

```
# /etc/rstudio/rserver.conf
server-health-check-enabled=1
```

After restarting the server, the following health-check endpoint will be available:

```
http://<server-address-and-port>/health-check
```

By default, the output of the health check will appear as follows:

```
active-sessions: 1
idle-seconds: 0
cpu-percent: 0.0
memory-percent: 64.2
swap-percent: 0.0
load-average: 4.1
license-status: Activated
license-days-left: 153
license-allow-product-usage: 1
```

## 60.0.2. Customizing Responses

The response to the health check is determined by processing a template that includes several variables. The default template is:

```
active-sessions: #active-sessions#
idle-seconds: #idle-seconds#
cpu-percent: #cpu-percent#
memory-percent: #memory-percent#
swap-percent: #swap-percent#
load-average: #load-average#
license-status: #license-status#
license-days-left: #license-days-left#
license-allow-product-usage: #license-allow-product-usage#
```

You can customize this template to return an alternate format (e.g. XML or JSON) that is parse-able by an external monitoring system. To do this you simply create a template and copy it to /etc/rstudio/health-check For example, an XML format:

Or a Prometheus endpoint. Prometheus is an open-source systems monitoring and alerting toolkit with a custom input format:

```
# /etc/rstudio/health-check
# HELP active_sessions health_check metric Active RStudio sessions
# TYPE active_sessions gauge
active_sessions #active-sessions#
# HELP idle_seconds health_check metric Time since active RStudio sessions
# TYPE idle_seconds gauge
idle_seconds #idle-seconds#
# HELP cpu_percent health_check metric cpu (percentage)
# TYPE cpu_percent gauge
cpu_percent #cpu-percent#
```

```
# HELP memory_percent health_check metric memory used (percentage)
# TYPE memory_percent gauge
memory_percent #memory-percent#
# HELP swap_percent health_check metric swap used (percentage)
# TYPE swap_percent gauge
swap_percent #swap-percent#
# HELP load_average health_check metric cpu load average
# TYPE load_average gauge
load_average #load-average#
```

# 60.0.3. Changing the URL

It's also possible to customize the URL used for health checks. RStudio Workbench will use the first file whose name begins with health-check in the /etc/rstudio directory as the template, and require that the full file name be specified in the URL. For example, a health check template located at the following path:

```
/etc/rstudio/health-check-B64C900E
```

Would be accessed using this URL:

```
http://<server-address-and-port>/health-check-B64C900E
```

Note that changes to the health check template will not take effect until the server is restarted.

# Part IX. License Management

# 61. Overview

# 61.1. License Management

RStudio Workbench uses a built-in license management system. A license controls the number of users that can connect to the server, as well as activation of the job launcher feature and the number of concurrent sessions that can be run. The license server supports named users as well as floating licenses.

#### 61.1.1. Activation Basics

When RStudio Workbench is first installed on a system it operates in evaluation mode for a period of time and then subsequently requires activation for continued use.

To determine the current license status of your system you can use the following command:

```
$ sudo rstudio-server license-manager status
```

After purchasing a license to RStudio Workbench you'll receive a license key that can be used to activate the license on a given system.

You can activate your license key with the command:

Note that you need to restart the server in order for licensing changes to take effect.

If you want to move your license of RStudio Workbench to another system you should first deactivate it on the old system with the command:

```
$ sudo rstudio-server license-manager deactivate
```

## 61.1.2. License Details

RStudio Workbench licenses include some details that you'll see when you run the status command. These are part of the license and can't be changed without purchasing a different license. They are as follows:

Value	Meaning
enable-launcher sessions	Whether the Job Launcher is enabled.  The maximum number of concurrent R sessions that can be run;
users	0 if unlimited. The maximum number of users that can connect to the server; $0$ if unlimited.

Typically, RStudio Workbench licenses either disable the Launcher but allow for unlimited local users and sessions, or enable the Launcher and provide limits on users and/or sessions. If you have questions about the best kind of license for your use case, contact sales@rstudio.com for help.

Note that when named user licenses are in effect (users is greater than 0), R Session Auditing is enabled by default to help you keep track of which users are using the product.

# 62. Connectivity Requirements

In order to activate or deactivate RStudio Workbench, internet connectivity is required for communication with the licensing server. If your server is behind an internet proxy or not connected to the Internet at all this section describes what's required to successfully activate.

Additionally, your server should have a synchronized system clock, using ntp or some other clock syncing service. If the server's clock is sufficiently incorrect, licensing verification will fail.

# 62.0.1. Proxy Servers

If your server is behind an internet proxy, you may need to add an additional command line flag indicating the address and credentials required to communicate through the proxy. This may not be necessary if either the http\_proxy or all\_proxy environment variable is defined (these are read and used by the license manager when available).

If you do need to specify a proxy server explicitly you can do so using the --proxy command line parameter. For example:

```
$ sudo rstudio-server license-manager --proxy=http://127.0.0.1/ activate cproduct-key>
```

Proxy settings can include a host-name, port, and username/password if necessary. The following are all valid proxy configurations:

```
http://127.0.0.1/
http://127.0.0.1:8080/
http://user:pass@127.0.0.1:8080/
```

If the port is not specified, the license manager will default to using port 1080.

#### 62.0.2. Offline Activation

If your system has no connection to the Internet it's also possible to perform an offline activation. To do this, we recommend using our offline activation application which will walk you through the process: RStudio Offline Activation

To activate your license offline, you first generate an offline activation request as follows:

\$ sudo rstudio-server license-manager activate-offline-request product-key>

Executing this command will print an offline activation request to the terminal which you should copy and paste and enter into our offline activation application or send to RStudio customer support (support@rstudio.com). You will receive a reply with a file attachment that can be used to activate offline as follows:

\$ sudo rstudio-server license-manager activate-offline <activation-file>

If you are renewing your license or want to move your license of RStudio Workbench to another system you can also perform license deactivation offline. You can do this as follows:

\$ sudo rstudio-server license-manager deactivate-offline

Executing this command will print an offline deactivation request to the terminal which you should copy and paste and enter into the offline activation application then send to RStudio customer support (support@rstudio.com).

You can also perform an offline check of your current license status using the following command:

\$ sudo rstudio-server license-manager status-offline

# 63. Evaluations

## 63.0.1. Extending Evaluations

If you are unable to complete your evaluation of RStudio Connect during the initial evaluation period, contact your Customer Success Representative or RStudio Sales (sales@rstudio.com) to speak about your evaluation process and how we may be able to help with the remaining items you need to test.

Once you have the key, supply it to RStudio Workbench using the extend-evaluation command.

#### \$ sudo rstudio-server license-manager extend-evaluation <key>

If you are performing the evaluation on a physical machine (not on virtualized hardware or containers) without a network connection, you may also request an offline evaluation extension key, which does not require an internet connection. This key may be supplied to RStudio Workbench as follows:

#### \$ sudo rstudio-server license-manager extend-evaluation-offline <key>

Note that offline evaluation extension keys are valid *only* on machines which do not have Internet access and are not virtualized. For most offline evaluation extensions, you will need to generate an offline evaluation request (see below for details).

## 63.0.2. Connectivity Requirements

### **Beginning Evaluations**

Generally speaking, there are no network requirements during the evaluation period. Inside virtual machines or sandboxes (such as Docker), however, Internet access is required to begin the evaluation period.

If you have a proxy, you can supply it using the --proxy argument as described above. If however you have no means of connecting to the Internet from inside the virtual environment, you can begin the evaluation as follows:

#### \$ sudo rstudio-server license-manager begin-evaluation-request

Executing this command will print an offline activation request to the terminal which you should copy and paste and then send to RStudio customer support (support@rstudio.com). You will receive a reply with a file attachment that can be used to begin the evaluation offline as follows:

\$ sudo rstudio-server license-manager begin-evaluation-offline <evaluation-file>

#### **Extending Evaluations**

You may extend evaluations offline using the same pattern described above (just use extend-evaluation-request and extend-evaluation-offline):

\$ sudo rstudio-server license-manager extend-evaluation-request

Then, when you've received the evaluation file:

\$ sudo rstudio-server license-manager extend-evaluation-offline <evaluation-file>

# 64. Floating Licensing

If you stop and start RStudio Workbench instances frequently, for instance because you're running them inside virtual machines or containers, you may wish to use floating licensing instead of traditional licensing.

To use floating licensing, you run a small, lightweight server, which holds a license that grants you the right to run a certain number of concurrent RStudio Workbench instances.

When RStudio Workbench starts, it will connect to the license server and obtain a temporary lease, releasing it when RStudio Workbench is stopped. Using this method, you can have any number of RStudio Workbench instances, so long as you do not run more instances at once than specified in your license.

#### 64.0.1. Floating License Keys

A license key which distributes floating license leases is not the same as a traditional license key, and the two cannot be used interchangeably. If you have purchased traditional license keys and wish to exchange them for a floating license key, or vice versa, please get in touch with RStudio customer support (support@rstudio.com).

# 64.0.2. The RStudio Workbench License Server

The RStudio Workbench License server supplies licenses to RStudio Workbench. It is available for Linux, Mac, and Windows. A license server can supply licenses to any platform; for instance, you can run your license server on Windows and distribute licenses to RStudio Workbench instances running Linux.

#### Linux

The RStudio License Server site contains Linux license server downloads for all RStudio products. Download and install the license server for RStudio Workbench. You then activate your license key with the command:

```
$ sudo dpkg -i rsp-license-server-1.1.2-x86_64.deb
$ sudo rsp-license-server activate product-key>
$ sudo rsp-license-server start
```

The file /etc/rsp-license-server.conf contains configuration settings for the RStudio Workbench License server, including the network port to listen on and any proxy settings required for connecting to the Internet.

#### Mac and Windows

The RStudio License Server site contains downloads for the Mac and Windows license servers. These require a few additional configuration steps. They can be set up as follows (order is important):

- 1. Download the license server (TurboFloatServer) appropriate to your platform, and place it in the directory where you wish it to run.
- 2. Copy the file /usr/lib/rstudio-server/bin/license-manager.conf to the same directory as TurboFloatServer, and change its name to TurboActivate.dat.
- 3. If activating online, run the command TurboFloatServer.exe -a="ABCD-EFGH-IJKL-MNOP-QRST-UVWX".
- 4. If activating offline, follow the offline activation steps. Note that it is necessary to supply the *fully qualified path* to file arguments to TurboFloatServer, even if they are in the same directory as the executable.
- 5. After successful activation, TurboFloatServer.exe -i with administrator permissions. This will install the license server as a service that will start automatically and run in the background.

For additional help with Mac and Windows license server installation and configuration, refer to the official TurboFloat Server documentation.

## 64.0.3. License Server Hardware Requirements

While the license server is designed to distribute licenses to ephemeral virtual machines or containers, which may move freely between hosts, the license server *itself* must stay on the same physical host on which it was activated.

It is acceptable to run the license server inside a container or a virtual machine, as long as the container or the VM always runs on the same hardware. Stopping the license server, moving it to a different physical host, and starting it again will cause it to become deactivated. Because it's often impractical to try to ensure that the server only runs on a particular host, we recommend running the license server outside containers and virtualization software.

Note that the system requirements for the license server are very low, so it can be run on almost any server. If your environment will not allow for ensuring that the server stays on a single physical host, please contact support@rstudio.com to discuss alternate licensing options.

#### 64.0.4. License Server Offline Activation

The rsp-license-server activate command requires an internet connection. If your license server has no connection to the Internet it's also possible to perform an offline activation. The process for doing this on the license server is identical to the process used to activate RStudio Workbench offline. Generate an offline activation request as follows:

```
$ sudo rsp-license-server activate-offline-request product-key>
```

Executing this command will print an offline activation request to the terminal which you should copy and paste and then send to RStudio customer support (support@rstudio.com). You will receive a reply with a file attachment that can be used to activate offline as follows:

```
$ sudo rsp-license-server activate-offline <activation-file>
$ sudo rsp-license-server restart
```

#### 64.0.5. License Server Deactivation

If you are permanently decommissioning your license server, or need to transfer its license to a different license server, you should deactivate (remove) its license as follows:

```
sudo rsp-license-server deactivate
```

#### 64.0.6. Using Floating Licensing

Once your license server is up and running, you need to tell RStudio Workbench to use floating licensing instead of traditional licensing.

```
# /etc/rstudio/rserver.conf
server-license-type=remote
```

The value remote indicates that RStudio Workbench should connect to a remote licensing server to obtain a license; the value local can be used to explicitly specify traditional (local) activation.

Then, tell RStudio Workbench which licensing server to connect to:

```
$ sudo rstudio-server license-manager license-server <server-hostname-or-ip>
$ sudo rstudio-server restart
```

You only need to run the license-server command once; RStudio Workbench saves the server name and will use it on each subsequent startup.

Depending on your system configuration, it is possible that the RStudio Workbench service will be started before the service which allows hostname resolution (this is known to be the case for example on some Amazon EC2 systems). If this is the case, you'll want to specify the license server using a private IP address rather than a hostname, so that RStudio Workbench can acquire a license immediately when starting up.

**Note:** If you are renewing your floating license offline or if you want to move your floating license for RStudio Workbench to another system, then you should first perform license deactivation offline.

To renew your floating license offline or to move your floating license, run the following to perform license deactivation offline:

## 64.0.7. Specifying a License Server

RStudio supports three methods for connecting to a license server. You can select the method by specifying the license server as follows:

#### **Connecting Over TCP**

Example

```
$ sudo rstudio-server license-manager license-server mylicensehost:9403
```

By default, the RStudio Workbench License Server listens via raw TCP on port 8989. If you specify the only the hostname or IP address of the license server, this the kind of connection you'll get.

If you wish to use a different port, you will need to specify the port in /etc/rsp-license-server.conf, and specify license-server to RStudio Workbench as <server-hostname-or-ip:port>.

#### **Connecting Over HTTPS**

Example

\$ sudo rstudio-server license-manager license-server https://mylicensehost/

You can also connect to the license server via HTTPS rather than raw TCP. This can be useful when using proxies or load balancers in front of the license server. Note that the license server does not use HTTPS by default, so you can only use this connectivity method if you have configured your license server to use HTTPS.

Configuring the license server for HTTPS support is outside the scope of this guide. You can read instructions here:

Configuring TurboFloat Server for HTTPS Communication

#### Connecting to a Hosted Server

Example

\$ sudo rstudio-server license-manager license-server 871A2BFA-87C5-11E9-BD16-4749DB7B7927

Finally, if you are connecting to a hosted license server, specify the UUID of the server rather than its hostname. Hosted license servers are run by our licensing vendor, WyDay, and do not require you to run anything in your own network. Read more about setting up a hosted license server and getting a UUID here:

Creating a LicenseChest Server

## 64.0.8. Configuring License Leases

When using floating licenses, you can optionally determine how long the license leases last by setting the lease length value on the licensing server. This value is in seconds, so for instance to make license leases last 30 minutes you would use the following syntax:

```
<!--/etc/rsp-license-server.conf-->
<lease length="1800"/>
```

The lease length controls how frequently the RStudio Workbench instances need to contact the licensing server to renew their license leases in order for the lease to remain valid.

A **shorter** lease length will increase tolerance to failures on RStudio Workbench instances by making leases available for reuse more quickly. RStudio Workbench will release its lease immediately if shut down normally, but if abnormally terminated, the lease will not be released until it expires.

A longer lease length will increase tolerance to transient failures on the network and the RStudio Workbench License Server. Any such issues that can be resolved before the lease is due for renewal won't interrupt use of RStudio Workbench.

We generally recommend using a longer lease length. Use a short lease length only if your environment routinely encounters abnormal terminations of the server or the container/instance on which it runs.

#### 64.0.9. License Server Downtime Tolerance

RStudio Workbench does not wait until its lease has fully expired before it renews it. It renews its lease when it is *halfway to expiry*. For instance, if you use 30 minute leases, RStudio Workbench will actually renew its lease every 15 minutes.

This means that it is possible to take down the license server for a short period of time without affecting any running RStudio Workbench instances. Because of the aforementioned behavior, no existing lease will be more than halfway to expiry if the server goes down. You have a grace period of N/2 (where N is the length of the lease) during which the server can be offline without consequences. For instance, if you use 30 minute leases, your license server can be offline for 15 minutes.

#### 64.0.10. Lease Expiration and Renewal

Under normal conditions RStudio Workbench will automatically renew its license lease in a configurable interval as described above. However, there are situations in which it will be unable to do so, such as a network problem, or an issue on the host running the license server.

When RStudio Workbench cannot obtain a license lease, either because there are no leases currently available or because it can't reach the licensing server, it will begin automatically attempting to acquire a lease every 10 seconds. This interval is configurable; for instance, to retry every 30 seconds instead you would set the following value:

# /etc/rstudio/rserver.conf license-retry-seconds=30 If you don't want RStudio Workbench to attempt to reestablish a license lease automatically, set the value to 0 to disable retries. In this case you will need to manually restart RStudio Workbench in order to reestablish the lease. This can be useful if you often run more instances than you have keys for, and wish to have more control over which RStudio Workbench instances receive license leases from the limited pool on the license server.

## 64.0.11. Troubleshooting Floating Licensing

To validate that the license server has been successfully activated, run the activation-status command. This will report the version of the server as well as the license key and the number of available slots.

```
$ sudo rsp-license-server activation-status
```

If your server is activated but you're still having trouble with floating licensing, you can tell the RStudio Workbench License Server to emit more detailed logs. Change the log level to notification:

```
<!--/etc/rsp-license-server.conf-->
<log file="/var/log/rstudio-licensing.log" level="notification"/>
```

Then, restart the license server, tail the licensing log, and start your RStudio Workbench instances.

```
$ sudo rsp-license-server restart
$ tail -f /var/log/rstudio-licensing.log
```

At the notification level, the licensing log will tell you the total number of licenses associated with your key, and how many are currently in use. It will also notify you when RStudio Workbench instances acquire leases, and when those leases are released, renewed, or expired. No rotation is done for this log, so it's recommended to use the warning level in production.

# 65. Userspace Licensing

In a standard installation of RStudio Workbench, licensing configuration operations require elevated access (e.g. via sudo). This is appropriate given the product will normally be installed and activated by an administrator on behalf of multiple user accounts.

Alternatively, the licensing system may be switched to a "userspace" model where the licensing commands are available to a non-elevated user account. This is a system-wide setting; it is not possible to mix both models on a single system.

# 65.0.1. Switching to Userspace Licensing

Before switching, stop the server, and deactivate any existing license-key or license-server.

```
$ sudo rstudio-server stop
$ sudo rstudio-server license-manager deactivate
$ sudo rstudio-server license-manager clear-license-server
```

Next, disable the standard system-level licensing mode by deleting the "verify" file.

```
$ sudo rm /var/lib/rstudio-server/verify
```

Initialize the license system in userspace mode. Note this command must **not** be run elevated, but as the RStudio server account (normally rstudio-server) as described in Server Account.

```
$ rstudio-server license-manager initialize --userspace
```

Confirm that the mode was successfully changed.

```
$ rstudio-server license-manager verify
Trial-Type: Verified
Status: Evaluation
Days-Left: 5
License-Scope: User
```

The line License-Scope: User indicates successful switch to userspace licensing (versus the default setting of System).

At this point, the user may perform licensing commands without using "sudo", such as activating a license-key or configuring a license-server.

# Part X. Data Connectivity

# 66. Overview

# 66.1. Data Connectivity

You can set up connections to shared data sources which will be available to users in the Connections pane of the RStudio IDE.

# 67. Connectivity using ODBC

RStudio makes ODBC connections available in the Connections Pane. ODBC connections are obtained from the odbcinst.ini file and can be further customized using Snippet Files.

## 67.0.1. Professional Drivers

RStudio Workbench provides connectivity to data sources through RStudio Professional Drivers. See Getting Started with RStudio Professional Drivers and Databases using R for more information.

# 68. Connectivity Using R Packages

For R Packages that provide data connectivity through the Connections Contract, RStudio makes these connections also available in the Connections Pane and can be further customized using Snippet Files. Currently, the odbc and sparklyr packages provide this connectivity.

## 69. Snippet Files

A Connection Snippet File is an R code snippet with additional metadata which is intended to initialize a connection. This file can be as simple as:

```
library(readr)
data <- read_csv(readr_example("mtcars.csv"))</pre>
```

Once this file is saved under /etc/rstudio/connections/ as Motor Trend Cars.R, RStudio will make this connection as available under the Connection Pane.

The path is configurable through the connections-path environment variable and multiple connection files can be specified.

In order to parameterize this connection, one can create fields using using the \${Position:Label=Default} syntax:

- **Position:** The row position starting at zero.
- Label: The label assigned to this field.
- **Default:** An optional default value.

For example, we can filter out this dataframe to produce the following connection interface:

```
library(readr)
data <- read_csv(readr_example("mtcars.csv"))
data[data$mpg == ${0:Miles per Gallon=21.4} | data$cyl == ${1:Cylinders=6}, ]</pre>
```

In order to create a; separated list of values, one can use the syntax \${Position:Label=Default:Key}. Semicolon-separated list are common in database connections and therefore, natively supported in snippet files, for instance:

```
"${2:Letters=ABC:LettersKey}${3:Numbers=123:NumbersKey}"
```

There are a couple of escape characters supported: **\$colon\$** to escape : and **\$equal** to escape =.

Additional resources are available under RStudio Extensions - Connections.

# Part XI. Job Launcher

# 70. Overview

#### 70.1. Job Launcher

The RStudio Job Launcher provides the ability for RStudio Workbench to start processes within various batch processing systems (e.g., Slurm) and container orchestration platforms (e.g., Kubernetes). RStudio Workbench integrates with the Job Launcher to allow you to run your R Sessions within your compute cluster software of choice, and allows you to containerize your sessions for maximum process isolation and operations efficiency. Furthermore, users can submit standalone adhoc jobs to your compute cluster(s) to run computationally expensive R scripts.

#### i Note

Integration with the Job Launcher is not enabled in all editions of RStudio Workbench. You can run rstudio-server license-manager status to see if the Launcher is enabled. If it isn't, contact sales@rstudio.com to purchase a license with the Job Launcher enabled.

# 71. Job Launcher Configuration

Before the Job Launcher can be run, it must be properly configured via the config file /etc/rstudio/launcher.conf; see the Job Launcher documentation for supported configuration options. If the Launcher was installed with RStudio Workbench, a default working configuration that uses the Local plugin is installed for your convenience.

The Launcher configuration parameter admin-group should be configured to the group value of the RStudio Workbench server user, specified in the server-user configuration parameter in rserver.conf (which defaults to rstudio-server). This makes the server user a Job Launcher admin, which is necessary to properly launch sessions on behalf of other users.

#### 71.0.1. RStudio Workbench Integration

RStudio Workbench must be configured in order to integrate with the Job Launcher. There are several files which house the configuration, and they are described within subsequent sections.

#### **Server Configuration**

The RStudio Workbench process rserver must be configured to communicate with the Job Launcher in order to enable session launching. The following table lists the various configuration options that are available to be specified in the rserver.conf configuration file:

#### /etc/rstudio/rserver.conf

Config Option	Description	Requir@efault (Y/N) Value	
launcher- sessions- enabled	Enables launching of rsession processes via the Job Launcher.  This must be enabled to use the Job Launcher.	N	0
launcher- address	TCP host/IP of the launcher host, or unix domain socket path (must match /etc/rstudio/launcher.conf configuration value). If using the default launcher configuration that ships with RStudio, this should be localhost (assuming you run the launcher side-by-side with RStudio Workbench).	Y	

Config Option	Description	_	uir@efault N) Value
launcher- port	Port that the launcher is listening on. Only required if not using unix domain sockets. If using the default launcher configuration that ships with RStudio, this should be 5559.	Y	
launcher- default- cluster	Name of the cluster to use when launching sessions. Can be overridden by the launching user.	N	
launcher- sessions- callback- address	Address (HTTP or HTTPS) of RStudio Workbench that will be used by launcher sessions to communicate back for project sharing and launcher features. The address must be the reachable address of the rserver process from the host that will be running rsession, which in the case of launcher sessions can be on a different network segment entirely. If RStudio is configured to use SSL, you must also ensure that the callback address hostname matches the FQDN of the Common Name or one of the Subject Alternate Names on the HTTPS certificate. See the example configuration below for more details.	Y	
launcher- sessions- callback- verify-ssl- certs	Whether or not to verify SSL certificates when Launcher sessions are connecting to RStudio. Only applicable if connecting over HTTPS. For production use, you should always leave the default or have this set to true, but it can be disabled for testing purposes.	N	1
launcher- use-ssl	Whether or not to connect to the launcher over HTTPS. Only supported for connections that do not use unix domain sockets.	N	0
launcher- verify-ssl- certs	Whether or not to verify SSL certificates when connecting to the launcher. Only applicable if connecting over HTTPS. For production use, you should always leave the default or have this set to true, but it can be disabled for testing purposes.	N	1
launcher- sessions- clusters	Whitelist of clusters to allow for submitting interactive session jobs to. The default allows all job launcher clusters to run interactive sessions.	N	
launcher- adhoc- clusters	Whitelist of clusters to allow for submitting adhoc jobs from the Launcher pane. The default allows all job launcher clusters to run adhoc jobs.	N	
launcher- sessions- container- image	The default container image to use when creating sessions.  Only required if using a plugin that requires containerization.  If none is specified, the Job launcher-specified default will be used, if the plugin supports it.	N	

Config Option	Description	Requir <b>®</b> lefault (Y/N) Value	
launcher- sessions- container- images	Comma-separated list of images which may be used for launching sessions. Used to filter out incompatible entries from the UI when a user is selecting an image to use for running the session. Leave blank to allow all images to be used.	N	
launcher- adhoc- container- images	Comma-separated list of images which may be used for launching adhoc jobs. Used to filter out incompatible entries from the UI when a user is selecting an image to use for running an adhoc job. Leave blank to allow all images to be used.	N	
launcher- sessions- container- run-as- root	Whether or not to run as root within the session container. We recommend you do not use this in most cases.	N	0
launcher- sessions- create- container- user	Whether or not to create the session user within the container. Only applicable if using container sessions and not running containers as root. The created user will have the same UID, GID, home directory, and login shell as the user that launched the session. It is recommended that this option be used, unless your containers connect to an LDAP service to manage users and groups. The container starts as root so it can create the correct user and group ids, then drops privilege to use the created user account. If it cannot drop privilege the container will fail to start.	N	1
launcher- sessions- forward- container- environmen	Whether or not to forward any container environment variables to the session. This is useful for example, propogating Kubernetes secrets to the sesion. However, the variables USER, HOME, and LOGNAME are not forwarded, and are atloaded from the user's passwd entry.	N	1
launcher- sessions- connection timeout- seconds	Number of seconds to allow for making the initial connection to a launcher session. Connection failures are retried - automatically - this is simply to prevent unreachable hosts from hanging the retry process as the default connection timeout on most systems is very high. Only change this if you are having trouble connecting to sessions. A value of 0 indicates that there should be no timeout (system default).	N	3

Config Option	Description	Requir@efault (Y/N) Value	
launcher- sessions- container- forward- groups	Whether or not to forward the user's supplemental groups to the created containers. This will only be done when not creating the container user, and when running the container as a non-root user, such as if integrating with LDAP. This is enabled by default, but if group lookups are very expensive in your environment and supplemental groups are not necessary, this can be disabled.	N	1

For example, your rserver.conf file might look like the following:

```
# /etc/rstudio/rserver.conf
launcher-address=localhost
launcher-port=5559
launcher-sessions-enabled=1
launcher-default-cluster=Kubernetes

# the callback address that launcher sessions will reconnect to rserver on
# since our Kubernetes jobs run on a different network segment, this needs
# to be the routable IP address of the web server servicing RStudio traffic
# (routable from the point of view of any Kubernetes nodes)
launcher-sessions-callback-address=http://10.15.44.30:8787

launcher-use-ssl=1
launcher-sessions-container-image=rstudio:R-3.5
launcher-sessions-container-run-as-root=0
launcher-sessions-create-container-user=1
```

#### SSL Considerations

Both RStudio Workbench and the Job Launcher can be configured to use SSL. When the Launcher is configured to use SSL, the RStudio Workbench node(s) that are connecting to the Launcher must ensure that the hostname configured in the launcher-address field matches the FQDN of the Common Name or Subject Alternate Name of the certificate that is presented by the Launcher. If the hostnames do not match exactly, SSL verification will fail, and RStudio will be unable to connect to the Job Launcher.

Similarly, if RStudio Workbench is configured to use SSL, the hostname configured in the launcher-sessions-callback-address field must match the FQDN of the Common Name or Subject Alternate Name of the certificate that is presented by RStudio. Failure to do so will

cause certificate verification to fail when sessions attempt to connect to RStudio, preventing you from using Job Launcher functionality such as starting Launcher jobs.

Additionally, both the RStudio Workbench and Job Launcher root certificates need to be imported into the trusted root certificate store on the systems that are accessing those addresses. For example, the Workbench server nodes need to have the Job Launcher root certificate installed in their trusted certificate store to ensure that certificate verification works correctly. The exact steps for importing a certificate into the trusted root store are operating system specific and outside of the scope of this document.

#### Job Launcher and PAM Sessions

PAM Sessions work slightly differently when used with Launcher sessions. See PAM Sessions with the Job Launcher for more information.

#### **Authentication**

RStudio Workbench authenticates with the Job Launcher via the secure-cookie-key file, a secret key that is read on startup of both the launcher and RStudio which is only readable by the root account. The file is present at /etc/rstudio/secure-cookie-key. If the Job Launcher is running on a different machine than RStudio Workbench, you will need to make sure that the exact same secure-cookie-key file is present on both machines.

To do this, create a secure cookie key file on one of the nodes like so:

```
# generate secure-cookie-key as a simple UUID
sudo sh -c "echo `uuid` > /etc/rstudio/secure-cookie-key"

# ensure that the cookie is only readable by root
sudo chmod 0600 /etc/rstudio/secure-cookie-key
```

Once this file has been created, copy it to the other node to the same location so that both services use the same key. Alternatively, you could accomplish this via a symlink to a location on a file share.

The path to the secure-cookie-key file can be changed, but it is not recommended in most cases. If you need to change it, it can be done by adding the following line to the /etc/rstudio/rserver.conf and /etc/rstudio/launcher.conf configuration files:

```
# /etc/rstudio/rserver.conf
secure-cookie-key-file=/path/to/secure-cookie-key
```

and:

```
# /etc/rstudio/launcher.conf
secure-cookie-key-file=/path/to/secure-cookie-key
```

When running Launcher sessions in a load balanced RStudio deployment, sessions do additional authorization verification to ensure that they are only used by the user that created them. This is accomplished by an RSA key pair, located at /etc/rstudio/launcher.pem and /etc/rstudio/launcher.pub. These files must be the same on every RStudio node, or users will be unable to use their sessions on multiple nodes.

In order to create the RSA files, run the following commands:

```
sudo openssl genpkey -algorithm RSA -out /etc/rstudio/launcher.pem -pkeyopt rsa_keygen_bits:
sudo openssl rsa -in /etc/rstudio/launcher.pem -pubout > /etc/rstudio/launcher.pub
sudo chmod 0600 /etc/rstudio/launcher.pem"
```

You must ensure that the above private key (.pem) file is owned by root and has 600 permissions, as it *must* remain secret to your users.

Once the files are created, simply copy them to each RStudio node in your cluster.

#### **Launcher Sessions**

It is recommended that you configure the Shared Storage path (see Shared Storage for configuration) in a location that will be reachable both by the RStudio Workbench instance and each Launcher Session in order to support various RStudio features. Failure to do so could cause subtle, unintended issues.

See the Launcher Mounts section for more details about how to configure this correctly with Containerized Sessions.

#### **Containerized Sessions**

In order to run your R sessions in containers, you will need a Docker image that contains the necessary rsession binaries installed. RStudio provides an official image for this purpose, which you can get from Docker Hub.

For example, to get the RHEL6 image, you would run:

```
docker pull rstudio/r-session-complete:centos7
```

After pulling the desired image, you will need to create your own Dockerfile that extends from the r-session-complete image and adds whatever versions of R you want to be available to your users, as well as adding any R packages that they will need. For example, your Dockerfile should look similar to the following:

```
FROM rstudio/r-session-complete:centos7

# install desired versions of R
RUN yum install -y R

# install R packages
...
```

See Docker Hub for more information.

Launcher Mounts When creating containerized sessions via the Job Launcher, you will need to specify mount points as appropriate to mount the users' home drives and any other desired paths. In order for sessions to run properly within containers, it is **required** to mount the home directories into the containers, as well as any directories containing per-user state (e.g., a customized XDG\_DATA\_HOME). The home mount path within the container must be the same as the user's home path as seen by the RStudio Workbench instance itself (generally, /home/{USER}).

To specify mount points, modify the /etc/rstudio/launcher-mounts file to consist of multiple mount entries separated by a blank line. The following table lists the fields that are available for each mount entry in the file.

Field	Description	•	ui <b>īlēd</b> fault N)Value
MountType	The type of mount. Can be Host, NFS, CephFs, GlusterFs, AzureFile, KubernetesPersistentVolumeClaim, or Passthrough	Y	
MountPath	The path within the container that the directory will be mounted to.	Y	
ReadOnly	Whether or not the mount is read only. Can be true or false.	N	false
JobType	What type of jobs the mount is applied to. Can be session, adhoc, or any.	N	any
Workbench	What type of workbench the mount is applied to. Can be rstudio, jupyterlab, jupyter notebook, vs code, or any.	N	any

Field	Description	Requi <b>de</b> dfault (Y/N)Value
Cluster	The specific cluster that this mount applies to. Applies to all clusters if not specified.	N

Depending on the MountType specified above, different settings may be used to control the mount.

## MountType: Host

Field	Description	Required (Y/N)
Path	The source directory of the mount, i.e. where the mount data comes from.	$\mathbf{Y}$

## MountType: NFS

Field	Description	Required (Y/N)
Path Host	The source directory of the mount, i.e. where the mount data comes from. The NFS host name for the NFS mount.	Y N

## MountType: CephFs

Field	Description	Required $(Y/N)$
Monitors	A comma-separated list of Ceph monitor addresses. For example:	$\overline{\mathbf{Y}}$
	192.168.1.200:8765,192.168.1.200:8766	
Path	The path within the Ceph filesystem to mount	${f N}$
User	The Ceph username to use	${f N}$
SecretFile	The file which contains the Ceph keyring for authentication	${f N}$
SecretRef	Reference to Ceph authentication secrets, which overrides SecretFile if specified	N

## MountType: GlusterFs

Field	Description	$\frac{\text{Required}}{(Y/N)}$
Endpoints	The name of the endpoints object that represents a Gluster cluster	Y
Path	configuration The name of the GlusterFs volume	Y

#### MountType: AzureFile

Field	Description	$\frac{\text{Required}}{(Y/N)}$
SecretNam	e The name of the secret that contains both the Azure storage account	$\overline{\mathbf{Y}}$
	name and the key	
ShareName	The share name to be used	

#### $Mount Type: \ Kubernetes Persistent Volume Claim$

Field	Description	Required (Y/N)
ClaimName	The name of the Kubernetes Persistent Volume Claim to use	Y

#### MountType: Passthrough

Field	Description	Required (Y/N)
FilePath	Path to a file that contains the raw JSON object representing the mount, which is sent directly to the back-end without transformation	Y

Note that for many mount types, paths may contain the special variable {USER} to indicate that the user's name be substituted, enabling you to mount user-specific paths.

An example /etc/rstudio/launcher-mounts file is shown below.

# /etc/rstudio/launcher-mounts

# User home mount - This is REQUIRED for the session to run

MountType: NFS Host: nfs01

Path: /home/{USER} MountPath: /home/{USER}

ReadOnly: false

# Shared code mount
Cluster: Kubernetes
MountType: NFS
Host: nfs01
Path: /dev64
MountPath: /code
ReadOnly: false

# Only mount the following directory when the user is launching a JupyterLab session
Cluster: Kubernetes

Cluster: Kubernetes
Workbench: JupyterLab
MountType: CephFs

Monitors: 127.0.0.1:8080,127.0.0.1:8081

SecretFile: /etc/secrets/ceph

ReadOnly: true

It is important that each entry consists of the fields as specified above. Each field must go on its own line. There should be no empty lines between field definitions. Each entry must be separated by one full blank line (two new-line \n characters).

If you choose to run your containers as root, the user home drive **must** be mapped to /root. For example:

# /etc/rstudio/launcher-mounts

MountType: NFS Host: nfs01

Path: /home/{USER}
MountPath: /root
ReadOnly: false

As noted in the Launcher Sessions section, it is recommended that you also mount the Shared Storage path (see Shared Storage for configuration) into the session container to support various RStudio features. When mounting the shared storage path, ensure that the folder is mounted to the same path within the container to ensure that the rsession executable will correctly find it. For example:

# /etc/rstudio/launcher-mounts

MountType: NFS
Host: nfs01

Path: /rstudio/shared-storage MountPath: /rstudio/shared-storage

ReadOnly: false

**Launcher Environment** You may optionally specify environment variables to set when creating launcher sessions.

To specify environment overrides, modify the /etc/rstudio/launcher-env file to consist of multiple environment entries separated by a blank line. The following table lists the fields that are available for each environment entry in the file.

Field	Description		ui <b>:De</b> fault V)Value
JobType	What type of jobs the environment value(s) is applied to. Can be session, adhoc, or any.	N	any
Workbench	What type of workbench the mount is applied to. Can be rstudio, jupyterlab, jupyter notebook, vs code, or any.	N	any
Cluster	The specific cluster that the environment applies to. Applies to all clusters if not specified.	N	
Environment	The environment variables to set, one per line (each subsequent line must be indented with an arbitrary amount of spaces or tabs), in the form of KEY=VALUE pairs.	N	

Additionally, you can use the special {USER} variable to specify the value of the launching user's username, similar to the mounts file above.

An example /etc/rstudio/launcher-env file is shown below.

```
# /etc/rstudio/launcher-env
```

JobType: session

Environment: IS\_LAUNCHER\_SESSION=1

IS\_ADHOC\_JOB=0

USER\_HOME=/home/{USER}

JobType: adhoc

Environment: IS\_LAUNCHER\_SESSION=0

IS\_ADHOC\_JOB=1

USER\_HOME=/home/{USER}

JobType: any

Cluster: Kubernetes

ENVIRONMENT: IS\_KUBERNETES=1

If you do not need to set different environment variables for different job types or different

clusters, you may simply specify KEY=VALUE pairs, one per line, which will be applied to all launcher ad-hoc jobs and sessions. For example:

```
IS_LAUNCHER_JOB=1
USER_HOME=/home/{USER}
```

Launcher Ports You may optionally specify ports that should be exposed when creating containerized jobs. This will allow the ports to be exposed within the host running the container, allowing the ports to be reachable from external services. For example, for Shiny applications to be usable, you must expose the desired Shiny port, otherwise the browser window will not be able to connect to the Shiny application running within the container.

To specify exposed ports, modify the /etc/rstudio/launcher-ports file to consist of multiple port entries separated by a blank line. The following table lists the fields that are available for each port entry in the file.

Field	Description	•	ui <b>:De</b> fault N)Value
JobType	What type of jobs the port(s) is applied to. Can be session, adhoc, or any.	N	any
Workbench	What type of workbench the mount is applied to. Can be rstudio, jupyterlab, jupyter notebook, vs code,	N	any
Cluster	or any.  The specific cluster that this set of ports applies to.  Applies to all clusters if not specified.	N	
Ports	The ports to expose, one per line (each subsequent line must be indented with an arbitrary amount of spaces or tabs).	N	

An example /etc/rstudio/launcher-ports file is shown below.

```
# /etc/rstudio/launcher-ports
JobType: adhoc
Ports: 6210
  6143
  6244
  6676

# additional Kubernetes ports to expose
JobType: adhoc
Cluster: Kubernetes
```

#### Ports: 4434

If you do not need to set different exposed ports for different job types or different clusters, you may simply specify port values, one per line, which will be applied to all launcher ad-hoc jobs and sessions. For example:

```
# /etc/rstudio/launcher-ports
5873
5874
64234
64235
```

#### **Containerized Adhoc Jobs**

To run adhoc jobs in containers from the Launcher pane, you need a Docker image containing the bash shell and the desired version of R on the path.

The adhoc job container will run using the same userId and groupId value as the RStudio user. In order for scripts under the home directory to be found in the container, the home directory must be mounted with the same absolute path inside the container.

Jobs started from the RStudio console via rstudioapi::launcherSubmitJob() have no specific container requirements.

# 72. Running the Launcher

Once it is configured, you can run the Job Launcher by invoking the command sudo rstudio-launcher start, and stop it with sudo rstudio-launcher stop. The Job Launcher must be run with root privileges, but similar to rstudio-server, privileges are immediately lowered. Root privileges are used only to impersonate users as necessary.

# 73. Load Balancing

Both RStudio Workbench and the Job Launcher services can be load balanced, providing maximum scalability and redundancy. When using the RStudio Workbench load balancer with the Launcher, it is generally sufficient to simply have each Workbench node point to its own node-local Launcher service via rserver.conf configuration - no external load balancer needs to control access to the Launcher itself.

#### Note

In this mode, when using the local Launcher, sessions will be balanced according to the setting you have defined under balancer in /etc/rstudio/load-balancer.

However, in some cases, you may want to scale the Job Launcher separately from RStudio Workbench. For example, if your Launcher cluster needs to exist in a different network for security reasons, such as to limit node connectivity to backend services (e.g., Kubernetes). In such cases, you will need to scale the Job Launcher separately via an external load balancer, and Workbench should be configured to point to this load balanced instance of the Job Launcher. In most cases, the external load balancer should be configured for sticky sessions, which will ensure that each instance of Workbench connects to just one Job Launcher node, providing the most consistent view of the current job state. For more information on configuring the Job Launcher for load balancing, see the Job Launcher documentation.

It should be noted that in most cases, load balancing is not needed for performance reasons, and is generally used for redundancy purposes.

# 74. Creating Plugins

Plugins allow communication with specific batch cluster / container or chestration systems like Slurm and Kubernetes. However, you may be using a system that R Studio does not natively support. The R Studio Launcher Plugin SDK can be used to quickly develop Plugins in C/C++.

## 75. Job Launcher Troubleshooting

If you experience issues related to running Launcher sessions, adhoc jobs, Jupyter sessions, or VS Code sessions, you can use the Launcher verification tool which will attempt to launch jobs and provide diagnostic output about what could be going wrong. To run the verification process, run the following command:

sudo rstudio-server verify-installation --verify-user=user

Replace the --verify-user value with a valid username of a user that is setup to run RStudio Workbench in your installation. This will cause the test jobs to be started under their account, allowing the verification tool to check additional aspects of launching jobs, including mounting the user's home directories into containers. You can also specify a specific test to run by using the --verify-test flag, like so:

sudo rstudio-server verify-installation --verify-user=user --verify-test=r-sessions

The above example will only test R Sessions, skipping adhoc jobs and Jupyter/VS Code sessions. The parameter can be one of r-sessions, adhoc-jobs, jupyter-sessions, or vscode-sessions. If the parameter is unspecified, all tests will be run.

# Part XII. Tutorial API

## 76. Overview

#### 76.1. Tutorial API

The Tutorial API provides an interface for driving automated interactions with the RStudio IDE. The Tutorial API assumes that RStudio is hosted within an <iframe> with the hosting page content surrounding it (e.g. in a sidebar).

The hosting <iframe> must be connected to the DOM and have non-zero dimensions (including not having the display: none style) when the IDE is loaded into it.

The API supports a variety of interactions with the IDE including typing console input, opening source files, opening projects, creating projects, showing help topics, and executing arbitrary R code.

This document describes the basic workings of the Tutorial API, and related settings. A simple example page is provided to demonstrate invoking the APIs.

The Tutorial API files are installed with RStudio Workbench in /usr/lib/rstudio-server/extras/tutorial.

- demo.htm is an example host page
- rstudio.js is used by a hosting page to interact with the Tutorial API; always use the version of rstudio.js that came with the installed version of RStudio Workbench and ensure it is cache-busted to prevent web browsers from using an older cached version (one option would be to rename it and reference it via that new name, e.g. rstudio001.js)

#### ! Important

These instructions, and the example page itself, assume the following regarding the domains utilized:

- The demo host page is served from domain http://localhost:8080
- The RStudio IDE is served from domain http://localhost:8787

If the IDE is being served from a different domain than http://localhost:8787 edit the **demo.htm** file and change all instances of http://localhost:8787 to the actual domain where the IDE is available.

# 77. Configuration

This section describes how to configure the example page, shown below. The upper-region has controls for experimenting with the Tutorial API calls, and the lower region is an <iframe> hosting RStudio Workbench.

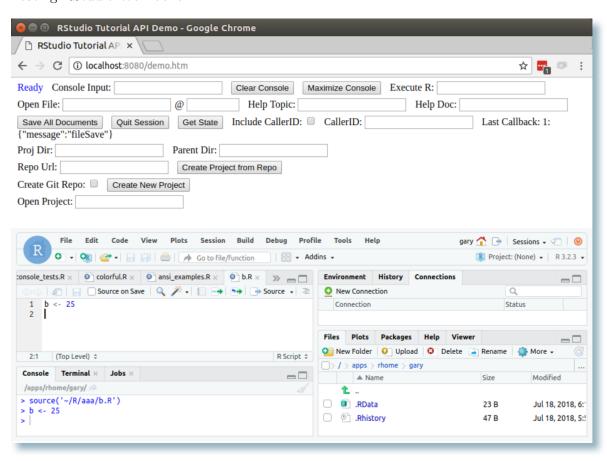


Figure 77.1.: Example Page

#### 77.0.1. Allow IFrame

For security reasons, RStudio will not load inside a browser frame (such as an IFrame) by default. You can modify this behavior by using the www-frame-origin option. See Frame Origin for full details.

To enable the Tutorial API example page to host RStudio Workbench on a **non-production** localhost server, do the following:

```
# /etc/rstudio/rserver.conf
www-frame-origin=http://localhost:8080
```

#### 77.0.2. Enable API Calls from Host Page

To enable calling the Tutorial APIs from the hosting page:

```
# /etc/rstudio/rsession.conf
tutorial-api-enabled=1
```

#### 77.0.3. Enable Callbacks from RStudio to Host Page

RStudio may optionally notify the hosting page of certain events using the JavaScript PostMessage mechanism.

The receiver is responsible for parsing, interpreting, and reacting to the value of the callbacks. Any JavaScript running in the context of the IDE IFrame could do an arbitrary SendMessage to the parent (in addition to those being generated by the RStudio IDE JavaScript). The host page should ensure the response string is valid JSON, and follows one of the patterns described in Tutorial API Callbacks.

To enable callbacks:

```
# /etc/rstudio/rsession.conf
tutorial-api-parent-notify-enabled=1
```

## 77.0.4. Disable RStudio Workbench Home Page

The Tutorial API cannot be used to manipulate the RStudio Workbench Home Page. To disable the home page:

```
# /etc/rstudio/rserver.conf
server-user-home-page=0
```

#### Note

The server-user-home-page setting is automatically forced on if Job Launcher sessions are enabled via launcher-sessions-enabled=true in /etc/rstudio/rserver.conf.

#### 77.0.5. Disable Multiple Session Support

Even with the Home Page disabled, a user can still open new sessions via Session / New Session which will open a new IDE instance outside of the host page. To prevent this:

```
# /etc/rstudio/rserver.conf
server-multiple-sessions=0
```

If multiple sessions are disabled, the Home Page is automatically turned off, so not necessary to include both settings.

#### Note

The server-multiple-sessions setting is automatically forced on when Job Launcher sessions are enabled via launcher-sessions-enabled=true in /etc/rstudio/rserver.conf.

#### 77.0.6. Serve the Example Page

Serve the folder /usr/lib/rstudio-server/extras/tutorial with a web server. For example, change to that directory in a terminal and run this command:

```
python -m SimpleHTTPServer 8080
```

Load http://localhost:8080/demo.htm in a web browser and you should see the page shown earlier.

## 78. Interface

You can access the Tutorial API by including rstudio.js within your host page. This will enable you to instantiate an **RStudio** object that has a connection to the RStudio IDE within an IFrame. For example, if the IDE were hosted in an IFrame as follows:

```
<iframe id="rstudio" src="http://localhost:8787"></iframe>
```

Then you would instantiate an RStudio object as follows:

The third argument is a callback that will be invoked once the RStudio API is ready to be called (i.e. once the IDE has loaded). Note that you should be sure to instantiate this object before the IFrame containing the IDE is fully loaded (if you don't then you might miss the onReady callback). The best way to do this is to either:

- 1. Create the object immediately after defining the IFrame within the host page; or
- 2. More conservatively, create the IFrame, then create the object, then provide the src to the IFrame (this is however almost certainly not necessary and #1 should suffice).

The fourth argument is a callback that will be invoked when certain events described below in Tutorial API Callbacks are emitted by the IDE. These are only emitted if tutorial-api-parent-notify-enabled=1 was set as described earlier. The response will be a text string in JSON format, as described below in Tutorial API Callbacks.

# 79. Tutorial API Methods

Methods are asynchronous and do not directly return a result.

If **Report Result** is **true** below then that API will attempt to invoke a **success** or **error** callback when it completes, but this is never guaranteed.

Methods which report a result take an optional final string parameter **callerID** which will be returned in both success and error callbacks. This can be used to correlate calls and responses.

The **RStudio** object supports the following methods:

		Reports
Method	Description	Re- sult
isReady()	Is the RStudio API available yet? (has the IDE loaded)	No
•	oda) by the result of the resu	No
consoleClear()	Clear all previous input and output from the console.	No
consoleMaximize	(Maximize the console so it occupies the full height of the IDE.	No
executeR(code)	Execute arbitrary R code within the global environment.	No
openFile(file,	·	No
location)	location. The location parameter can either be a line number	
	(e.g. 42) or a regular expression delimited by / (e.g. /foo/).	
helpTopic(topic) Navigate the help pane to a topic. The topic should be a namespace		No
	qualified reference to package documentation (e.g. graphics::plot).	
	Note that help topics defined within the base package do not need	
	qualification.	
helpDoc(doc)	Show a markdown (.md) or R Markdown (.Rmd) document within	No
	the help pane.	
saveAllSourceDo	csaye all unsaved source documents.	No
<pre>quitSession()</pre>	Quit session. User cannot cancel but is prompted to save files and	No
	workspace if needed.	
createProjFromG	i <b>€\[wep@Upl</b> gject from Git and open it.	Yes
projDir,		
parentDir,		
<pre>[callerID])</pre>		

Method	Description	Reports Re- sult
createNewProj parentDir, createRepo, [callerID])	(pr <b>6jpàr</b> e, and open a new project.	Yes
openProj(proj	Filepen an existing project, can specify existing .Rproj, or just the	Yes
<pre>[callerID]) getState([cal</pre>	folder. lerRepuest info on current RStudio IDE state.	Yes

## 79.1. Example Calls

Here are some sample JavaScript calls, followed by the response JSON.

## 79.1.1. Clone project from github and open it (with callerID = 'abcd')

```
createProjFromGit('https://github.com/rstudio/rstudioapi', 'rstudioapi', '~/R', 'abcd');

{
    "message":"success",
    "api":"createProjFromGit",
    "result":"",
    "callerID":"abcd"
}
```

## 79.1.2. Create an new project but don't create a git repo (no callerID)

```
createNewProj('myproj', '~/R', false);

{
   "message":"success",
   "api":"createNewProj",
   "result":""
}
```

## 79.1.3. Try to open a project that doesn't exist

```
openProj('~/myproj', 'abcd');

{
    "message":"error",
    "api":"openProj",
    "result":"Unable to find .Rproj file in [~/myproj]",
    "callerID":"abcd"
}
```

# 80. Tutorial API Callbacks

As seen in the examples above, the callback returns JSON. The following are examples of calls that trigger callbacks.

Method	Description	Example payload
getState	response from getState	See examples below
error	a prior API call failed	{"message": "error", "api":
		"createProjFromGit", "result": "access
		denied", "callerID": "abcd"}
success	a prior API call succeeded	{"message": "success", "api":
		"createProjFromGit", "callerID":
		abcd

There are also callbacks sent directly by RStudio (i.e. not in response to Tutorial API calls).

Method	Description	Example payload
fileSave	contents of a file loaded in RStudio IDE were saved (including auto-save of content and/or metadata)	{"message": "fileSave"}
sessionDisconnect	connection between RStudio IDE and its rsession process was closed	{"message": "sessionDisconnect"}
sessionSuspend	the rsession process has been suspended	{"message": "sessionSuspend"}

## 80.1. getState

The getState() method returns the path to the currently loaded .Rproj (if any) and any git remotes.

The results come via the callback. Here are some examples.

#### 80.1.1. Project with remotes

```
"message": "getState",
"callerID": "abcd",
"project": "~/aaa/aaa.Rproj",
"remotes": [
  {
    "active": true,
    "remote": "origin",
    "type": "fetch",
    "url": "https://github.com/rstudio/aaa"
  },
    "active": true,
    "remote": "origin",
    "type": "push",
    "url": "https://github.com/rstudio/aaa"
  }
]
```

## 80.1.2. Project with no remotes

```
"message": "getState",
  "callerID": "abcd",
  "project": "~/localproj/localproj.Rproj",
  "remotes": []
}
```

#### 80.1.3. No project and no remotes

```
{
   "message": "getState",
   "callerID": "abcd",
   "project": null,
```

```
"remotes": []
}
```

# Part XIII. Jupyter Sessions

# 81. Overview

## 81.1. Jupyter Sessions

RStudio Workbench allows you to launch Jupyter sessions from the home page via the Job Launcher, if configured. Users have the option of starting either JupyterLab or Jupyter Notebook sessions that allow them to work with Jupyter while still working within the administrative framework provided by RStudio, such as authentication, PAM session management, etc.

#### Note

Before Jupyter sessions can be launched, the Job Launcher must be setup correctly. For more information, see the Job Launcher section.

# 82. Jupyter Configuration

Configuration of the Jupyter feature is handled via the config file /etc/rstudio/jupyter.conf. Note that this file is not automatically created by RStudio, and must be created before being configured, and the Jupyter feature is disabled by default. The following table lists the various configuration options that are available to be specified in the jupyter.conf configuration file:

#### /etc/rstudio/jupyter.conf

Config		Default
Option	Description	Value
labs-	Enables launching of JupyterLab sessions.	0
enabled		(dis-
		abled)
notebooks-	Enabled launching of Jupyter Notebook sessions.	0
enabled		(dis-
		abled)
jupyter- exe	Path to the Jupyter executable.	/usr/local/bin/jupyter
lab- command	The Jupyter command to run when starting a Lab session.	lab
lab-	The version of Jupyter Lab being used. If set to auto, the	auto
version	version is automatically detected by running the Jupyter Lab	
	version command. If Jupyter is being launched via the	
	Launcher and is not installed locally, you are strongly	
	encouraged to set this to the version in use. Running against	
	multiple different Jupyter versions simultaneously in different	
	Launcher clusters is not supported - they must all be running	
	the same version, or proxying issues will occur.	

Config Option	Description	Default Value
lab-args	Arguments to be passed to the Jupyter Lab command. Note that this does not override the default value - if you wish to use some of the default arguments, you must contain the default arguments in the configuration value. It is strongly recommended that you do not change this unless you know what you're doing!	no- browser  allow- root
notebook- command	The Jupyter comman to run when starting a Notebook session.	ip=0.0.0.0 notebook
notebook- version	The version of Jupyter Notebook being used. If set to auto, the version is automatically detected by running the Jupyter Notebook version command. If Jupyter is being launched via the Launcher and is not installed locally, you are strongly encouraged to set this to the version in use. Running against multiple different Jupyter versions simultaneously in different Launcher clusters is not supported - they must all be running the same version, or proxying issues will occur.	auto
notebook- args	Arguments to be passed to the Jupyter Notebook command. Note that this does not override the default value - if you wish to use some of the default arguments, you must contain the default arguments in the configuration value. It is strongly recommended that you do not change this unless you know what you're doing!	no- browser  allow- root
session- clusters default- session- cluster	List of available Job Launcher clusters for launching Jupyter sessions. Leave blank to specify all clusters.  The default Job Launcher cluster to use when launching a Jupyter session.	ip=0.0.0.0
default- session- container-	The default container image to use when launching a containerized Jupyter session.	
image session- container- images	Comma-delimited list of images that may be used for running Jupyter sessions.	

Config Option	Description	Default Value
jupyter- session- path	Path to the Jupyter Session launcher executable/script. It is recommended that you do not change this unless you know what you're doing, and you need to point to a different script.	/usr/lib/rstudio- server/bin/jupyter- session- run
session- no-profile	Enables/disables running of bash profile scripts when starting Jupyter sessions.	0 (run profile scripts)
session- cull- minutes	The amount of idle time to wait in minutes before a Jupyter session's kernels and terminals is automatically "culled" (shut down). To disable culling, set the value to 0. Enabling this sets the following Jupyter arguments: MappingKernelManager.cull_interval=60 MappingKernelManager.cull_idle_timeout=<60 * session-cull-minutes value>	120
session- shutdown- minutes	The amount of idle time to wait in minutes before a Jupyter session is shut down after its kernels and terminals have been culled. To disable automatic shutdown, set the value to 0.  This setting sets the the following Jupyter arguments: NotebookApp.shutdown_no_activity_timeout=<60 * session-shutdown_minutes value> LapApp.shutdown_minutes value>	5
session- cull- connected	Whether or not to cull sessions that have a browser connected. Regardless of this setting, only idle (unused) sessions are culled. Sets theMappingKernelManager.cull_connected Jupyter setting.	1 (true)

For example, your jupyter.conf file might look like the following:

```
# /etc/rstudio/jupyter.conf
jupyter-exe=/usr/bin/jupyter
labs-enabled=1
notebooks-enabled=1
session-cull-minutes=240
default-session-cluster=Kubernetes
default-session-container-image=rstudio:jupyter-session
```

### 82.0.1. Jupyter Versions

Currently, RStudio Workbench supports JupyterLab 2.x and 3.x, as well as Jupyter Notebooks 6.x. Due to differences between Jupyter versions, RStudio Workbench needs to know the versions of Jupyter Notebook and Jupyter Lab that are in use. They can be automatically detected by RStudio Workbench on start up by running the Jupyter command, or can be specified by using the lab-version and notebook-version settings in /etc/rstudio/jupyter.conf. Automatic version detection is done if the default value of auto is used.

If you are running Jupyter in a remote cluster like Kubernetes, you will need to manually specify the version of Jupyter if it is not locally installed or is not running the same version as the remote cluster. Due to differences in routing, running against multiple different versions of Jupyter in different Launcher clusters simultaneously is not supported - the versions must match in all clusters.

You should specify a full Jupyter 3-part version number like so:

```
# /etc/rstudio/jupyter.conf
lab-version=3.0.6
notebook-version=6.2.0
```

### 82.0.2. Launcher Configuration

When creating containerized Jupyter sessions via the Job Launcher, you will need to specify mount points as appropriate to mount the users' home drives and any other desired paths. In order for sessions to run properly within containers, it is **required** to mount the home directories into the containers.

For more information, see Launcher Mounts. Note that you can specify the Workbench with either JupyterLab or Jupyter Notebook to configure mount entries that should only be mounted for JupyterLab and Jupyter Notebook sessions, respectively.

#### Note

If you are using NFSv3 for the shared storage for user home directories, you will need to set the <code>local\_lock=all</code> NFS setting in /etc/fstab in order for Jupyter sessions to work properly. Failure to do this will cause Jupyter sessions to hang. For this reason, we strongly recommend the use of NFSv4 for Jupyter sessions. See the NFS documentation for information on NFS settings.

### i Note

Only a user's home folder is visible within Jupyter sessions. To access other folders, create symbolic links to them within the home folder.

### 82.0.3. Container Configuration

When running Jupyter sessions in containers, such as by using the Kubernetes Job Launcher plugin, you will need to ensure that the image(s) used to launch Jupyter sessions contain, at minimum, the following:

- 1) Python 2.7 or Python 3.x
- 2) JupyterLab and/or Jupyter Notebook installation
- 3) RStudio session binaries
- 4) If creating container users (see Server Configuration), you must have the libuser1-dev or libuser-devel packages, depending on your platform to install the libuser library and development tools.

For ease of use, it is recommended that you use the r-session-complete Docker image as a base for any Jupyter session images you intend to create. This will allow you to use that one base image to provide Jupyter itself and a default version of Python, which you can extend if necessary to add/modify Jupyter versions or add additional versions of Python if desired. See Docker Hub for more information.

### 82.0.4. Installing the Jupyter Notebook Plugin

RStudio Workbench can further integrate with Jupyter Notebooks by utilizing the rsp-jupyter Jupyter Notebook plugin. This plugin will automatically track and write recently opened notebooks to the Workbench homepage under the *Recent Projects* section, and also provides an easy way for users to leave their notebooks and return to the homepage.

If running Workbench in containers via the **r-session-complete** Docker image, you do not need to install the Notebook plugin, as it comes with the container. Otherwise, to install and enable the plugin, run the following commands:

```
pip install rsp_jupyter
jupyter-nbextension install --sys-prefix --py rsp_jupyter
jupyter-nbextension enable --sys-prefix --py rsp_jupyter
```

If you need to uninstall the plugin for whatever reason, run the following command:

```
jupyter-nbextension uninstall --sys-prefix --py rsp_jupyter
```

To upgrade the plugin, you must first uninstall and then reinstall it.

### 82.0.5. Adding Python Environments to Jupyter

You can add new Python virtual environments or conda environments to Jupyter as Python Kernels that users can select, allowing them to have flexibility in the Python versions they can use.

The following steps show how to add a new Python virtualenv as a Kernel to Jupyter:

- 1. Create the virtualenv or conda environment
- 2. Add the environment as an iPython kernel
- 3. Verify the kernel is available in Jupyter

```
# first, create the virtual env in a directory accessible by all users
cd /opt/python-kernels

# then make the virtualenv
virtualenv myenv

# alternatively, you can make a conda environment
# conda create -n myenv

# then, add the virtualenv to the Kernels list globally for all users
sudo ipython kernel install --name "myenv" --display-name "My Python Environment"

# finally, verify that the kernel is registered with Jupyter
sudo jupyter kernelspec list
```

Once the Kernel is listed in the output of the jupyter kernelspec list command, it will be available for use in both JupyterLab and Jupyter Notebooks.

To remove the kernel, simply run the following command:

```
sudo jupyter kernelspec remove "myenv"
```

For more information, see the IPython docs.

### 82.0.6. Installing JupyterLab Extensions

With the release of JupyterLab 3, it is now easier for system administrator to add extensions to Jupyter by using pip. Simply pip install the extension you want, and it will show up in JupyterLab sessions launched from RStudio Workbench.

For example, to install the jupyterlab-kernelspy extension using pip:

### pip install jupyterlab-kernelspy

### i Note

You will need to install the desired extensions on all Launcher nodes that are running JupyterLab sessions. For containerized sessions, the extension must be installed in the container image(s) that are used.

There is no need to explicitly enable or disable the extension. Once installed, active sessions can see the extension after refreshing the browser. Other extensions can be found on the Pypi.org site.

In previous versions of JupyterLab, you needed to install packages from source and build them with nodejs. This flow is still supported, but installing prebuilt extensions via pip is much simpler. For more information, see the JupyterLab extension docs.

# 83. Jupyter Troubleshooting

If you experience issues related to running Jupyter sessions, you can use the Launcher verification tool which will attempt to launch both RStudio and Jupyter launcher sessions and provide diagnostic output about what could be going wrong. For more information, see the Troubleshooting section for the Job Launcher integration documentation.

# Part XIV. VS Code Sessions

### 84. Overview

### 84.1. VS Code Sessions

RStudio Workbench allows you to launch VS Code sessions from the home page via the Job Launcher, if configured. Users can start VS Code sessions that allow them to work with VS Code while still working within the administrative framework provided by RStudio, such as authentication, PAM session management, etc.

### i Note

Before VS Code sessions can be launched, the Job Launcher must be setup correctly. For more information, see the Job Launcher section.

### Note

RStudio is not affiliated with Microsoft. Your usage of the open source code-server and VS Code are subject to their respective licenses: see here and here.

### 84.1.1. RStudio Workbench Extension

VS Code sessions are intended to be used with the RStudio Workbench VS Code extension installed. The extension provides users with a button to open the RStudio Workbench homepage in a new tab. Because VS Code sessions are running remotely, any web server a user may be developing, such as Shiny, Dash, or Streamlit, will also be running remotely. The RStudio Workbench VS Code extension allows users to access these remote web servers through their browser. If the extension is not installed, VS Code sessions will launch but without the additional features; users working on these types of apps will not be able to access the servers they are running on.

### 85. Installation

#### 85.0.1. General Installation

In order to add VS Code integration to RStudio Workbench, you must first install the open-source code-server wrapper. This wrapper allows access to VS Code via a web server in a browser, allowing Workbench to create VS Code sessions and proxy them through the browser.

To install, simply run the command rstudio-server install-vs-code <path to installation directory>. This will install the code-server binary, and automatically configure /etc/rstudio/vscode.conf.

VS Code utilizes extensions to provide support for different languages and other features, such as custom themes. Normally extensions are managed by a user who installs them through an extension marketplace onto their desktop. For RStudio Workbench, you need to decide if users will be able to install extensions individually themselves, or if you will provide a global installation of extensions. User installation of extensions provides the most flexibility and is consistent with a user's desktop experience. However, if RStudio Workbench will be in a restricted or airgapped network, admins may need to install extensions globally for users.

The default configuration will allow users to manage their own extensions; if use of a global extensions directory is desired, then it may be passed as an option to the install-vs-code command. Alternatively, vscode.conf can be manually updated to include --extensions-dir after it has been created. For example, to install everything at /opt/code-server with per-user extensions directories:

```
# likely need root privilege to install to /opt
sudo rstudio-server install-vs-code /opt/code-server
```

To install everything at /opt/code-server with a global extensions directory at /opt/code-server/extensions:

```
# likely need root privilege to install to /opt
sudo rstudio-server install-vs-code /opt/code-server --extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/code-server/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extensions-dir=/opt/extens
```

Any arguments after the installation directory will be added to the args entry in vscode.conf, which is supplied to the code-server binary when the session is started. Note that if a global extensions directory is provided this way, this command will also automatically install the RStudio Workbench VS Code extension as described in the Installing the RStudio Workbench Extension section.

Once the installation completes, you'll need to restart the RStudio service for it to detect availability of VS Code sessions.

### Note

You must also install code-server on all Launcher nodes (for Local or Slurm plugins) and within any containers used (if using Kubernetes).

### 85.0.2. Installing the RStudio Workbench Extension

### Requirements

The RStudio Workbench extension requires code-server 3.9.3. While later versions may work as expected, compatibility is only guaranteed with version 3.9.3. The install-vs-code-ext command can upgrade code-server automatically for you if you include the -d flag with the directory that code-server should be installed in. For example, your command may look like this:

rstudio-server install-vs-code-ext -d /opt/code-server

#### Installation

If using a global extensions directory, the install-vs-code-ext script can be used to install the latest version of the extension. This script is run while upgrading RStudio Workbench, but may fail with a warning if the version requirement is not met.

If allowing user extension directories, and the requirements above are met, the extension will be installed on a per user basis when the user launches their first VS Code session.

### 85.0.3. Manual Installation

code-server can be installed manually, if desired. Additionally, if you have opted for a global extensions directory you can also opt to manually install the RStudio Workbench VS Code extension, and it may be necessary to manually install any other extensions that you require, if users will not have write access to the extensions directory.

### Manually Installing code-server

If you'd like to install code-server manually, simply download the code-server Linux distributable (available at <a href="https://rstd.io/vs-code-server-3-9-3">https://rstd.io/vs-code-server-3-9-3</a>) and extract it to the desired location on the RStudio Workbench host(s).

For example, to manually install code-server under /opt:

```
# create directory to house code-server
mkdir /opt/code-server
cd /opt/code-server

# download the code server package
wget https://rstd.io/vs-code-server-3-9-3 -0 vs-code-server.tar.gz

# extract code-server binary
tar zxvf vs-code-server.tar.gz --strip 1

# remove the archive
rm vs-code-server.tar.gz
```

### Manually Installing the RStudio Workbench VS Code Extension

When VS Code sessions are configured with a global extensions directory, it is possible to manually install the VS Code extension rather than using the provided <code>install-vs-code-ext</code> script. If VS Code sessions are not configured with a global extensions directory, the RStudio Workbench VS Code Extension will be automatically installed the first time a user launches a VS Code session. It is still possible to manually install the extension in this circumstance, however the operation will have to be performed for every user.

The RStudio Workbench VS Code extension can be downloaded from https://rsw-vscode-extension.s3.amazonaws.com/rstudio-workbench-1.0.4.vsix. The current latest version is 1.0.4.

For example, to install the RStudio Workbench VS Code extension to the global extension directory /opt/code-server/extensions:

```
# Ensure the extension directory exists
mkdir -p /opt/code-server/extensions
cd /opt/code-server/extensions

# Download the extension
export RSW_EXT_VERSION=1.0.4
```

```
wget https://rsw-vscode-extension.s3.amazonaws.com/rstudio-workbench-${RSW_EXT_VERSION}.vsix
# Install the extension
/opt/code-server/code-server --extensions-dir /opt/code-server/extensions --install-extension
```

To install the extension manually on behalf of a particular user, when not using a global installation directory, run the following commands as the desired user:

```
# Ensure the extension directory exists
mkdir -p "${XDG_DATA_HOME:-~/.local/share}/rstudio"
cd "${XDG_DATA_HOME:-~/.local/share}/rstudio"

# Download the extension
export RSW_EXT_VERSION=1.0.4
wget https://rsw-vscode-extension.s3.amazonaws.com/rstudio-workbench-${RSW_EXT_VERSION}.vsix
# Install the extension
/opt/code-server/code-server --install-extension ./rstudio-workbench-${RSW_EXT_VERSION}.vsix
```

### Note

Installing the extension manually when a global installation directory is not in use will require you to repeat the operation for every user.

### Manually Installing Language Extensions

When VS Code sessions are configured with a global extensions directory, users without write access to this directory will not be able to install extensions and the extensions marketplace will be disabled. Otherwise, users are able to manage their own extensions, and installs must be performed on a per user basis by the user. There are three ways in which extensions can be installed:

1. Install the extension through code-server's hosted repository. This can be done from within a VS Code session via the UI. Alternatively, this can be done via the command line with the following command, with the --extensions-dir argument omitted when a global directory is not in use:

/opt/code-server/code-server --extensions-dir /opt/code-server/extensions --install-extension

2. Download an extension in VSIX format from an online marketplace such as Open VSX. Then, install it using the --install-extension command like the example above, passing the VSIX file path as the extension name. Note that if the latest version of the extension is incompatible with the version of code-server, it will fail to install. You can try using an older version of the extension in this case.

### Note

It is against VS Code's Terms of Service to use the official Marketplace extensions with third party tools like code-server. We strongly recommend using a free and open source alternative like Open VSX.

3. Build the extension from source. There are several extensions freely available on GitHub that can be built into a VSIX file yourself and then installed via the --install-extension command. Building third party extensions from source is outside of the scope of this document.

RStudio does not provide support for external extensions. If you have questions or issues we encourage you to check Stack Overflow, the extension's repository, or the code-server repository.

# 86. Configuration

### 86.0.1. VS Code Configuration

Configuration of VS Code sessions is handled via the config file /etc/rstudio/vscode.conf. Note that this file is not automatically created by RStudio and must be created manually or with the install-vs-code script, as VS Code sessions are disabled by default. The following table lists the various configuration options that are available to be specified in the vscode.conf configuration file:

### /etc/rstudio/vscode.conf

Config Option	Description	Default Value
enabled	Enables launching of VS Code sessions.	0 (disabled)
exe	Path to the code-server executable. VS Code sessions rely on the open source code-server project. For more information, see the project site	/opt/code- server/bin/code-server

Config Option	Description	Default Value
version	The version of VS Code code-server being used. If set to auto, the version is automatically detected by running the code-server version command. If VS Code is being launched via the Launcher and is not installed locally, you are strongly encouraged to set this to the version in use. Running against multiple different code-server versions simultaneously in different Launcher clusters is not supported - they must all be running the same version, or issues will occur.	cur.   auto
args	Arguments to be passed to the code-server launch command. You can supply an —extensions-dir= to point to previously installed extensions via this parameter. If no—host argument is supplied, a default of—host=0.0.0.0 will be assumed.	-host=0.0.0.0
session-clusters	Comma-delimited list of available Job Launcher clusters for launching VS Code sessions.  Leave blank to specify all clusters.	

Config Option	Description	Default Value
default-session-cluster	The default Job	
	Launcher cluster to use	
	when launching a VS	
	Code session.	
default-session-container-image	The default container	
	image to use when	
	launching a	
	containerized VS Code	
	session.	
session-container-images	Comma-delimited list of	
	images that may be	
	used for running VS	
	Code sessions.	/ /2-2 / 2-
vscode-session-path	Path to the VS Code	/usr/lib/rstudio-
	Session launcher	server/bin/vscode-
	executable/script. It is	session-run
	recommended that you	
	do not change this	
	unless you know what	
	you're doing, and you	
	need to point to a	
session-no-profile	different script. Enables/disables	0 (run profile scripts)
session-no-prome	running of bash profile	o (run prome scripts)
	scripts when starting	
	•	
	VS Code sessions.	

For example, your vscode.conf file might look like the following:

```
# /etc/rstudio/vscode.conf
exe=/usr/bin/code-server
enabled=1
default-session-cluster=Kubernetes
default-session-container-image=rstudio:vscode-session
```

### 86.0.2. VS Code code-server Versions

Currently, RStudio Workbench supports code-server 3.9.3. Due to differences between code-server versions, RStudio Workbench needs to know the versions of code-server

that is in use. It can be automatically detected by RStudio Workbench on start up by running the code-server command, or can be specified by using the version setting in /etc/rstudio/vscode.conf. Automatic version detection is done if the default value of auto is used.

If you are running VS Code in a remote cluster like Kubernetes, you will need to manually specify the version of code-server if it is not locally installed or is not running the same version as the remote cluster. Due to version differences, running against multiple different versions of code-server in different Launcher clusters simultaneously is not supported - the versions must match in all clusters.

You can manually specify the version number like so:

```
# /etc/rstudio/vscode.conf
version=3.9.3
```

### 86.0.3. VS Code User Settings

By default, code-server writes VS Code user settings under ~/.local/share/code-server. This can be changed by adding the user-data-dir path argument to the VS Code options, like so:

```
# /etc/rstudio/vscode.conf
exe=/usr/bin/code-server
args=--host=0.0.0.0 --verbose
user-data-dir=<desired user path>
```

Note that the specified in the configuration can contain the ~ to represent the user's home directory.

Whenever a VS Code session is launched, RStudio automatically merges the user settings template file at /etc/rstudio/vscode-user-settings.json with the user's settings file. This file should be a valid JSON file that contains desired VS Code user settings. Any matching settings that already exist in the user settings file are not overridden, allowing users the freedom to customize their environment. You should use the vscode-user-settings.json file to provide them with a simple baseline configuration.

If you installed VS Code via the sudo rstudio-server install-vs-code command, a simple template is created for you automatically which instructs VS Code to use the bash shell by default and to disable automatic extension updates. The generated file looks like the following:

/etc/rstudio/vscode-user-settings.json

```
{
    "terminal.integrated.shell.linux": "/bin/bash",
    "extensions.autoUpdate": false,
    "extensions.autoCheckUpdates": false
}
```

### 86.0.4. Launcher Configuration

When creating containerized VS Code sessions via the Job Launcher, you will need to specify mount points as appropriate to mount the users' home drives and any other desired paths. In order for sessions to run properly within containers, it is **required** to mount the home directories into the containers.

For more information, see Launcher Mounts. Note that you can specify the Workbench with VS Code to configure mount entries that should only be mounted for VS Code sessions.

### i Note

Only a user's home folder is visible within VS Code sessions. To access other folders, create symbolic links to them within the home folder.

### 86.0.5. Container Configuration

When running VS Code sessions in containers, such as by using the Kubernetes Job Launcher plugin, you will need to ensure that the image(s) used to launch VS Code sessions contain, at minimum, the following:

- 1) code-server 3.9.3 binary
- 2) RStudio Workbench session binaries
- 3) If creating container users (see Server Configuration), you must have the libuser1-dev or libuser-devel packages, depending on your platform to install the libuser library and development tools.

For ease of use, it is recommended that you use the **r-session-complete** Docker image as a base for any VS Code session images you intend to create.

# 87. Multiple Sessions

Multiple VS Code windows with distinct processes and projects can be opened within the same VS Code session in separate browser windows. Closing a VS Code session window causes the processes associated with that window to exit, while other processes associated with the same session will remain running.

After a user closes all windows associated with a session, restarting the session from the RStudio Workbench homepage will open the last project that the user opened in this session. The order in which the windows are closed does not impact this. Reopening the VS Code Workspace or folder the user was previously working in will return the window to the state that it was left in.

# 88. VS Code Troubleshooting

If you experience issues related to running VS Code sessions, you can use the Launcher verification tool which will attempt to launch VS Code sessions and provide diagnostic output about what could be going wrong. For more information, see the Troubleshooting section for the Job Launcher integration documentation.

Part XV.

**Database** 

# 89. Overview

### 89.1. Database

RStudio supports multiple database options. Currently, the supported databases are SQLite and PostgreSQL. When running RStudio Workbench in a load balanced configuration, you *must* use a PostgreSQL database, as SQLite is insufficient for managing state between multiple nodes.

The contents of the database are described in the Data Dictionary appendix.

### 90. Configuration

In order to set up a database connection, modify the file /etc/rstudio/database.conf. The file contains documentation about how to use it, and you can simply uncomment any lines that are relevant to your configuration. Note that because the file can contain password data, this file must be user read/write only (file mask 600).

### 90.0.1. SQLite

By default, RStudio creates a SQLite database for you automatically under the /var/lib/rstudio-server directory. For single-node installations, this is sufficient, but as stated before, will not be sufficient for load balanced deployments.

You should **never** specify a SQLite directory that is on shared storage, such as NFS. Per the SQLite documentation, this can cause data corruption.

Sample configuration:

```
# /etc/rstudio/database.conf
provider=sqlite

# Directory in which the sqlite database will be written
directory=/var/lib/rstudio-server
```

### 90.0.2. PostgreSQL

If you wish to use PostgreSQL, you must create an empty database for RStudio to connect to. You must not share this database with other products or services. If running an HA setup, it is strongly recommended to run the database on a separate server than the RStudio services to ensure maximum availability. The minimum supported PostgreSQL version is 9.5.

RStudio requires PostgreSQL to be configured for password-based authentication. For more details on this and other PostgreSQL fundamentals, see the support article Install and Configure PostgreSQL for RStudio Workbench / RStudio Server Pro.

Sample configuration:

```
# /etc/rstudio/database.conf
# Note: when connecting to a PostgreSQL database, a default empty rstudio database must firs
provider=postgresql
# Specifies the host (hostname or IP address) of the database host
host=localhost
# Specifies the database to connect to
database=rstudiodb
# Specifies the TCP port where the database is listening for connections
port=5432
# Specifies the database connection username
username=rstudio
# Specifies the database connection password. This may be encrypted with the secure-cookie-k-
# The encrypted password can be generated using the helper command rstudio-server encrypt-pa
# It is strongly recommended that you encrypt the password!
password=test
# Specifies the maximum amount of seconds allowed before a database connection attempt
# is considered failed. The default if not specified is 10 seconds. Corresponds to the
# PostgreSQL "connect_timeout=" connection string parameter.
connection-timeout-seconds=12
```

PostgreSQL connection URIs are also supported if preferred. If specifying additional options other than the ones provided above, such as sslmode, the use of a URI is required.

### For example:

```
# /etc/rstudio/database.conf
provider=postgresql

# Specifies the connection URL in the form of a postgresql:// connection URL. This can be use
# to set special database settings that are not available with the other parameters. If set,
# override any other postgresql parameters that have been set, with the exception of the pass
# the URI is supported as a convenience but we strongly recommend using the separate password
# always replace any password specified in the URI.
connection-uri=postgresql://rstudio@localhost:5432/rstudiodb?sslmode=allow&options=-csearch_]
```

Available PostgreSQL connection string parameters are documented in the official PostgreSQL documentation.

### i Note

The password in connection-uri may contain characters that may need to be URL-encoded to work properly. Avoid encoding the password by using the separate password field in the configuration.

### SSL

SSL for the PostgreSQL database can be used with RStudio by specifying sslmode=allow within the connection-uri parameter. The connection-uri mode of configuration must be used to specify additional database connection options such as these, beyond the simple name/value pairs that are supported.

For example:

```
# /etc/rstudio/database.conf
provider=postgresql
connection-uri=postgresql://postgres@localhost:5432/rstudio?sslmode=allow
```

#### **Schemas**

If you need to, you can tell RStudio to restrict itself to keeping its tables within a specific schema. You control this by giving PostgreSQL a search path as part of the URL by adding options=-csearch\_path=<schema-name> to the connection-uri. If it's the only item you're adding, separate it from the rest of the URL with? (just like the sslmode item above). Otherwise, separate it from other items with '&'.

For example:

```
provider=postgresql
connection-uri=postgresql://postgres@localhost:5432/rstudio?sslmode=allow&options=-csearch_page.
```

RStudio will refuse to start when given a schema that does not already exist. The schema must be owned by the connecting user or by a group that contains the connecting user.

### PostgreSQL Account

When setting up your PostgreSQL database for use with RStudio, ensure that you do not use the default postgres user account that comes with a standard installation. This is to ensure that your database is secure. Always ensure that whichever account that is used to access the database contains a strong password - do **not** use an account that has no password! You should also ensure that only one PostgreSQL user has access to the RStudio database for maximum security.

### PostgreSQL Password Encryption

A plain-text password in the password or connection-uri options of the /etc/rstudio/database.conf file must only be used temporarily for testing purposes. A warning will be present in RStudio log output when a plain-text password is being used.

We strongly recommend encrypting the password using the command rstudio-server encrypt-password. This way, if you have to backup your configuration, save it to a repository or share it with RStudio Support, your PostgreSQL password will be protected.

Use the following steps to encrypt the PostgreSQL password:

- Remove the password from the connection-uri option if defined in the database.conf
  file.
- Run the command sudo rstudio-server encrypt-password and enter the PostgreSQL password.
- Copy the resulting encrypted password printed in the terminal.
- Add or replace the password option in the database.conf file using the encrypted password copied above.
- Restart RStudio. Confirm it operates normally. You should no longer see a warning about plain-text password in RStudio logs.

The password encryption uses the secure-cookie-key value. By default RStudio generates this key during installation and stores it in /var/lib/rstudio-server/secure-cookie-key.

The same key value must be used for both encryption and decryption. If the key value used to encrypt the PostgreSQL password changes, the password must be re-encrypted with the new key and updated in /etc/rstudio/database.conf.

If preparing RStudio configuration files on one system for deployment on other system(s), you must manually generate a key and store it in /etc/rstudio/secure-cookie-key, then encrypt the password again, update /etc/rstudio/database.conf, and ensure this secure-cookie-key file is deployed along with other RStudio configuration files. The technique for creating this file is described in Generating a Key.

### PostgreSQL Connection Testing and Troubleshooting

Once the settings have been entered in /etc/rstudio/database.conf, use the sudo rstudio-server verify-installation command to test connectivity and quickly view errors and warnings.

### 90.0.3. Connection Pool

RStudio will create a pool of connections to the database at startup. The size of this pool defaults to the number of logical CPUs on the host running RStudio Workbench, up to 20. It is not generally recommended that you adjust the size of this pool manually unless you need to address a specific problem, such as exceeding a connection limit on the database or experiencing delays in RStudio Workbench caused by unavailable connections.

If you do need to adjust the size of the pool, you can do so by setting pool-size in database.conf as follows:

# /etc/rstudio/database.conf
pool-size=5

## 91. Migration

When changing database providers, you **must** migrate your existing database data from your current database provider to the new provider to prevent unexpected data loss.

The following steps should be taken to perform a successful migration from SQLite to Post-greSQL:

- 1. Create an empty database called rstudio in PostgreSQL (or any custom name according to your RStudio configuration below). Ensure the connection credentials work for the new database.
- 2. Stop RStudio.
- 3. Switch to PostgreSQL by modifying the /etc/rstudio/database.conf file. If you are storing the SQLite database in a different location be sure to keep the directory option in the file during the migration.
- 4. Run the command sudo rstudio-server migrate-db. Watch for the output and confirm that the migration was successful.
- 5. Once the data has been imported to the new database, restart RStudio.

### i Note

The migration from PostgreSQL to SQLite is not currently supported.

# Part XVI. Hardening

### 92. Overview

### 92.1. Hardening

RStudio uses secure defaults wherever possible, but for maximal security hardening it's necessary to use values that make stronger assumptions or require additional configuration. This section of the Administration Guide demonstrates the use of these more secure configuration values and describes other security considerations.

A summary of these recommendations in the form of a set of example configuration files is presented at the end of this section: Example Secure Configuration

## 93. Set Up SSL PRO

A secure installation of RStudio encrypts network traffic using SSL. SSL doesn't come preconfigured since it requires certificates signed by a Certificate Authority (CA) trusted by all parties.

### 93.0.1. Use SSL for Web Users

If your configuration of RStudio is accessed directly by end users, see the SSL Configuration section, which describes how you can ensure that HTTPS is used when RStudio is accessed via a web browser. Note that this does not apply if you are terminating SSL upstream, for example when you are using nginx or Apache in front of RStudio as described in Running with a Proxy and handling SSL there.

### 93.0.2. Use SSL for the Job Launcher

Ensure that communication with the Job Launcher is encrypted by setting launcher-use-ssl=1 as follows:

```
# /etc/rstudio/rserver.conf
launcher-use-ssl=1
```

Note that additional configuration for the Job Launcher is required to make it possible to connect to it over SSL. See Job Launcher Configuration for details. Example Launcher configuration:

```
# /etc/rstudio/launcher.conf
enable-ssl=1
certificate-file=/var/certs/your_domain_name.crt
certificate-key-file=/var/certs/your_domain_name.key
```

### 93.0.3. Restrict TLS Versions

RStudio Workbench supports many different SSL protocols for compatibility with older browsers, but several are no longer considered secure. We recommend disabling support for all SSL protocols except the most recent two, TLS 1.2 and 1.3. See the SSL Protocols section for more details.

```
# /etc/rstudio/rserver.conf
ssl-protocols=TLSv1.2 TLSv1.3
```

### 93.0.4. Use HTTP Strict Transport Security (HSTS)

When configured with SSL, RStudio Workbench uses HTTP Strict Transport Security automatically. This is a security setting that forces the browser to always use HTTPS when connecting to RStudio Workbench. We recommend including the maximum age to 1 year, and extending coverage to subdomains.

```
# /etc/rstudio/rserver.conf
ssl-hsts-max-age=31536000
ssl-hsts-include-subdomains=1
```

This ensures that the browser will not connect via HTTP to the domain running RStudio Server (and any of its subdomains) for one year.

### 93.0.5. Using SSL with RStudio Server Open Source

RStudio Workbench has built-in SSL and HTTPS controls as described in this section. However, much of the same advice applies if you are securing an installation of the Open Source edition of RStudio Server; you can run RStudio Server behind a reverse proxy such as Nginx and perform SSL termination upstream.

### 94. Browser Security

This section summarizes the recommendations in the Access and Security section.

### 94.0.1. Enable Origin Checks

To help mitigate against CSRF attacks, RStudio can automatically reject any request that originated from a domain it doesn't recognize. To enable this check, add the following configuration:

```
# /etc/rstudio/rserver.conf
www-enable-origin-check=1
www-allow-origin=mysubdomain.mydomain.com
```

The www-allow-origin setting is optional, but is helpful when RStudio is running behind a proxy. See Additional Security Considerations for details.

### 94.0.2. Disable Frame Embedding

By default, RStudio does not permit frame embedding (that is, it will not load inside another web page's <frameset> or <iframe>). No change is necessary to enforce this, but you can request it explicitly as follows:

```
# /etc/rstudio/rserver.conf
www-frame-origin=none
```

### 94.0.3. Set SameSite Cookie Header

RStudio does not set the SameSite cookie header by default (see Same Site cookies for details). We recommend setting it explicitly.

```
# /etc/rstudio/rserver.conf
www-same-site=lax
```

### 95. R Session Security

RStudio includes a number of options which can help harden the surface of the RStudio IDE itself. The settings in this section all apply to the IDE's user interface for R sessions.

Remember that RStudio is an interface to R itself, which has a variety of tools that can access the file system and shell as the user themselves. Follow security best practices by relying on operating system-level permissions, not front end restrictions, to guard access to sensitive content and files.

### 95.0.1. Limit Idle Time

By default, RStudio allows users to be idle for up to an hour before automatically signing them out. If your users work with sensitive data, you may wish to decrease this.

```
# /etc/rstudio/rserver.conf
auth-timeout-minutes=20
```

See Inactivity Timeout for details.

### 95.0.2. Restrict System Directory Access

RStudio can optionally prevent users from browsing to system directories; see Restricted Directories for details. Enable this feature as follows:

```
# /etc/rstudio/rsession.conf
restrict-directory-view=1
```

### 95.0.3. Disable External Publishing

RStudio includes support for publishing to several external services, including RPubs and Shinyapps.io. If your users work with sensitive information, you should disable publishing to these services as follows:

### 95.0.4. Disable Other Features

The are a few other features you should consider disabling. We have not included them in our Example Secure Configuration because they can impede productivity for end users.

- **Disable shell access** (allow-shell=0); disables the Terminal tab used to execute system commands
- Disable file downloads (allow-file-downloads=0); disables downloading files using the Files pane
- Disable file uploads (allow-file-uploads=0); disables uploading files using the Files pane
- Disable package installation (allow-package-installation=0); disables the user interface for installing R packages

Note that regardless of the values of these settings, users can execute system commands, install packages, and upload and download content using R itself.

### 96. Database

This section describes how to secure information described in RStudio's internal database.

#### 96.0.1. Set permissions for SQLite

If you are using SQLite, set the permissions such that users cannot access the database. For example, run the command: chmod 600 /var/lib/rstudio-server/rstudio.sqlite or wherever the the SQLite database is stored.

#### 96.0.2. Set permissions of the configuration file

The database configuration may contain sensitive data. Set the permissions such that users cannot access the database configuration file. For example, run the command: chmod 600 /etc/rstudio/database.conf

#### 96.0.3. Encrypt Database Password

When using PostgreSQL as a database provider, ensure that you're using an encrypted database password as described in PostgreSQL password encryption.

```
# /etc/rstudio/database.conf
# Generated by rstudio-server encrypt-password
password=ThX7skaB8VhMRk7jQr1J3lS0fk+GLmXDp3JIVcHwPiK1CMixSIEsNTt3cNBYj9Rx
```

### 97. Other

### 97.0.1. Enforce Group Requirement

By default, anyone who can successfully authenticate on the system can use the IDE. You can get more control over who's able to log into the system by creating a group such as rstudio-users and instructing RStudio to limit access to that group.

# /etc/rstudio/rserver.conf
auth-required-user-group=rstudio-users

### 98. Example Secure Configuration

This section aggregates all of the security recommendations from the above sections. Note, again, that some adjustment is likely to be necessary depending on your environment; for example, this set of configuration values presumes that SSL termination is happening in RStudio, that RStudio is the only application running on its domain, and that it is never embedded in another page.

Therefore, use these files as a starting point rather than copying and pasting them into your own system.

```
# /etc/rstudio/rsession.conf
# Disable publishing to RPubs and shinyapps.io
allow-external-publish=0

# Prevent exploration of system directories
restrict-directory-view=1
```

```
# /etc/rstudio/rserver.conf
# Limit access to those users to whom it's been explicitly granted via group membership
auth-required-user-group=rstudio-users

# Sign users out after 20 minutes of inactivity (default is 60)
auth-timeout-minutes=20

# Use HTTPS when connecting to web browsers
ssl-enabled=1
ssl-certificate=/var/certs/your_domain_name.crt
ssl-certificate-key=/var/certs/your_domain_name.key

# Limit SSL protocol versions to modern TLS
ssl-protocols=TLSv1.2 TLSv1.3

# Increase HTTP Strict Transport Security to 1 year and include subdomains
ssl-hsts-max-age=31536000
ssl-hsts-include-subdomains=1
```

```
# Enable origin checks on all HTTP requests (CSRF defense)
www-enable-origin-check=1

# Ensure that the domain on which RStudio is hosted is permitted as an origin
www-allow-origin=mysubdomain.mydomain.com

# Ensure the SameSite attribute is set on all cookies
www-same-site=lax

# Disallow embedding on other pages
www-frame-origin=none

# Use HTTPS when connecting to the Job Launcher
launcher-use-ssl=1
```

```
# /etc/rstudio/launcher.conf
enable-ssl=1
certificate-file=/var/certs/your_domain_name.crt
certificate-key-file=/var/certs/your_domain_name.key
```

```
# /etc/rstudio/database.conf
# Generated by rstudio-server encrypt-password
password=ThX7skaB8VhMRk7jQr1J3lS0fk+GLmXDp3JIVcHwPiK1CMixSIEsNTt3cNBYj9Rx
```

# A. Session User Settings

The following table enumerates the settings supported in the user (or system) rstudio-prefs.json file, along with their type, allowable values, and defaults.

PropertyDescription	Type	Default
allow_soWhetherlumnsable the ability to add source columns to display.	boolean	true
always_eWallethemto adorayordano enable the concordance for RNW files.	ceboolean	true
always_statethistonyalways save the R console history.	boolean	true
always_shistwnf_filatensions (beginning with ., not case sensitive) that are always shown in the Files Pane, regardless of whether hidden files are shown	array	.circleci, .gitattributes, .github, .gitignore, .httr-oauth, .r, .rbuildignore, .rdata, .renvignore, .renviron, .rhistory, .rprofile, .ruserdata
always_shiswnf_filesnames (case sensitive) that are always shown in the Files Pane, regardless of whether hidden files are shown	array	.build.yml, .gitlab-ci.yml, .travis.yml
ansi_con <b>Notw_twodde</b> at ANSI escape codes in the console.	string (off, on, strip)	on
auto_ap <b>Whd</b> theewtimensure that source files end with a newline character.	boolean	false

PropertyDescription	Type	Default	
auto_detAdh_etilmelentoation	boolean	false	
automatically detect			
indentation settings			
from file contents.			
auto_dis <b>Whethquatk</b> age_depend	enboiestean	true	
automatically discover			
and offer to install			
missing R package dependencies.			
auto_explaindtherror_tracebacks	boolean	false	
automatically expand	boolean	Tanse	
tracebacks when an			
error occurs.			
auto_runWhethpr_thunk	boolean	true	
automatically run an			
R Markdown			
document's Setup			
chunk before running			
other chunks.		1000	
auto_savEheddelepperiod, in	integer	1000	
milliseconds, after which documents			
should be auto-saved.			
auto save herita	boolean	false	
automatically save	boolean	Taise	
when the editor loses			
focus.			
auto_saveowntoideal with	string	backup	
changes to documents	(commit,	-	
on idle.	backup,		
	none)		
backgrou <b>Wd<u>he</u>tdhægntæstiius</b> n code	boolean	true	
diagnostics in the			
background, as you			
type.	. ,	2000	
backgroufide_diagnostics_delay_	_mateger	2000	
milliseconds to delay before running code			
diagnostics in the			
background.			
zaciono dana.			

PropertyDescription	Type	Default
biocondu <b>Etter_name of_the</b> me default Bioconductor mirror.	string	Seattle (USA)
biocondu <b>Etur <u>U</u>Rikrof <u>t</u>he</b> l default Bioconductor mirror.	string	http://www.bioconductor.org
blinking Whester to flash the cursor off and on.	boolean	true
browser <u>Ifistedf widthwfdths</u> fonts to check for browser support.	array	Andale Mono, Bitstream Vera Sans Mono, Cascadia Code, Consolas, Courier New, Courier, DejaVu Sans Mono, Droid Sans Mono, Fira Code, Hack, IBM Plex Mono, Inconsolata, JetBrains Mono, Lucida Console, Lucida Sans Typewriter, Menlo, Monaco, Monoid, Operator Mono, Pragmata, SF Mono, Source Code Pro, Vera Sans Mono, Victor Mono, Ubuntu Mono
busy_de <b>Hoti</b> oto detect busy status in the Terminal.	string (always, never, list)	always
busy_ex <b>@</b> ukintnoflistps that should not be considered busy in the Terminal.	array	tmux, screen
check_ar whether ttocheck unction arguments to R function calls.	n <u>b</u> oallsan	false
check_foWhatdateso check for new versions of RStudio when RStudio starts.	boolean	true
check_ntWhentenablechointers RStudio will detect R objects containing null external pointers when building the Environment pane, and avoid introspecting their contents further.	boolean	false

PropertyDescription	Type	Default
check_unwhoethedtcashigatmfeart_ unexpected variable assignments inside R	_irbo <b>bleat</b> ion_	_caffilse
function calls.  clang_verbosverbosity level to	integer	0
use with Clang (0 - 2) clean_te\notine \text{Valetheouttpulean} output after running	boolean	true
Texi2Dvi. cleanup_Wither_tondleachenk	boolean	true
temporary files after running R CMD CHECK.		
code_contention use auto-completion for R code in the RStudio code editor.	string (always, never, triggered, manual)	always
code_coffipletium_behavacters characters in a symbol that can be entered before completions are offered.	integer	3
code_coffipletium_berlay  milliseconds to wait  before offering code suggestions.	integer	250
code_contifletion_usther auto-completion for other languages (such as JavaScript and SQL) in the RStudio code editor.	string (always, triggered, manual)	always
command V pathetre tonkeep track of recently used commands in the Command Palette	boolean	true
console_Widetheontplationys use code completion in the R console.	boolean	true

PropertyDescription	Type	Default
console_Wonetherclick_select double-clicking should select a word in the Console pane.	boolean	false
console_The reasithubimit number of characters to display in a single line in the R console.	integer	1000
console_Thax mlaxismum number of console actions to store and display in the console scrollback buffer.	integer	1000
continue Whathemtsontinue ewline comments (by inserting the comment character) after adding a new line.	e boolean	false
cran_miff be CRAN mirror to use.	object	
custom_3hedlfudbynopmanhified path to the custom shell command to use in the Terminal tab.	string	
custom_\filed\continusd-line options to pass to the custom shell command.	string	
data_viewhe maximenhumns number of columns to show at once in the data viewer.	integer	50
default_ <b>directing</b> ult character encoding to use when saving files.	string	
default_Ishex_defaogtapmogram to use when processing LaTeX documents.	string	pdfLaTeX

PropertyDescription	Type	Default
default_ <b>pilojedt</b> relacetyiquath under which to place new projects by default.	string	
default_ <u>if</u> _heRiversion to use by default.	object	
default_swavefændingine to use when processing Sweave documents.	string	Sweave
diagnostive heher touration_calls diagnostics in R function calls.	boolean	true
diagnosti <b>Whethesave</b> check code for problems after saving it.	boolean	true
disabled <u>Listaofliuria-livro</u> uncement announcements to disable.	n <b>as</b> ray	Empty
doc_out \textit{Maicshobjects to show} in the document outline pane.	string (sections_only, sec-	sections_only
	tions_and_c all)	hunks,
documen <u>Thauthfault</u> name to use as the document author when creating new documents.	string	
document have milliseconds to wait before linting a document after it is loaded.	integer	5000
editor_kayhinkeyngindings to use in the RStudio code editor.	string (default, vim, emacs, sublime)	default
editor_tHemeename of the color theme to apply to the text editor in RStudio.	string	Textmate (default)

PropertyDescription	Type	Default
emoji_skimeocured emoji skintone	string ((None), (Default), Light, Medium- Light, Medium, Medium- Dark, Dark)	(None)
enable_stropporteaderssibility aids such as screen readers (RStudio Server).	boolean	false
enable_s\( \mathbb{M}\) pether to enable code snippets in the RStudio code editor.	boolean	true
enable_twhetheagto enable moving text on the editing surface by clicking and dragging it.	boolean	true
execution The havior R code to execute when the Execute command is invoked.	string (line, statement, paragraph)	statement
file_monlfist_oigpated_componentsrray components; file monitor will ignore paths containing one or more of these		Empty
components.  find_panel_slagacyeditbr_saquenceboolean     panel, tab key moves     focus directly from     find text to replace     text.		false

PropertyDescription	Type	Default
focus_coWshktheftto focus the R console after executing an R command from a script.	boolean	false
fold_style he style of folding to use.	string (begin- only, begin-and- end)	begin-and-end
$\begin{array}{c} \text{font\_size} \underline{\Gamma} \underline{\textbf{pointfa}} \text{ult editor} \\ \text{font size, in points.} \end{array}$	number	10
full project in desktop window title.	t <b>ibb</b> olean	false
git_diff_Whetherwholicspace whitespace when generating diffs of version controlled files.	boolean	false
git_exe_Bathpath to the Git executable to use.	string	
global_tHdmetheme to use for the main RStudio user interface.	string (classic, default, alternate)	default
graphics Typteabifasintg-aliasing to be used for generated R plots.	string (default, none, gray, subpixel)	default
graphics <u>R</u> b <b>grkph</b> ics backend.	string (default, cairo, cairo-png, quartz, windows, ragg)	default
handle <u>eWhesthem tuskeanddede</u> or errors only when user code is on the stack.	n <b>ly</b> oolean	true
help_fonThsizelppopiansel font size, in points.	number	10

PropertyDescription	Type	Default
hide_console_ther_tchhide_texeRut console when executing inline R Markdown chunks.	eboolean	true
hide_obj\chiteffles to hide object files in the Files pane.	boolean	true
highlight Woodherltonkighlight code chunks in R Markdown documents with a different background color.	boolean	true
highlight Wednetshebe tordisplay error, warning, and message output in a different color.	boolean	true
highlight Whethertton highlight R function calls in the code editor.	boolean	false
highlight Highlight the selected line in RStudio's code editor.	boolean	false
highlight <u>Highlighd_thwose</u> lected word in RStudio's code editor.	boolean	true
highlight Wheth dinke b links in comments are clickable.	boolean	true
ignore_u\mathbf{u}\mathbf{p}\mathbf{q}\mathbf{erthsre_twoigds}\text{re} words in uppercase when spell checking.	boolean	true
ignore_wWhathwitto_igunouters words with numbers in them when spell checking.	boolean	true
initial_wbheingitialrewookyng directory for new R sessions.	string	

PropertyDescription	Type	Default
insert_mWhhhther to insert matching pairs, such as () and [], when the first is typed.	boolean	true
insert_navikethoipehoipesettr Pipe Operator command should insert the native R	boolean	false
pipe operator,  > insert_n\text{Will tations erstections} numbered sections in LaTeX.	s boolean	false
insert_pWensthafteo_ifisaction_comparentheses after function completions.	o <b>bpoletáo</b> n	true
insert_sp\\destherotm\destherotm\destherotm\destherotm\destherotm\destherotm\destructure spaces around the equals sign in R code.	boolean	true
install_p\(\frac{\partinitistdlla}{\partinitistdlla}\) package dependencies one at a time.	boolean	true
jobs_tab <u>T</u> hesihishibility of the Jobs tab.	string (closed, shown, default)	default
knit_wof <b>Rhegwoli</b> king directory to use when knitting R Markdown documents.	string (default, current, project)	default
latex_pr <b>Wiewn_tonpmervisow</b> _idle  LaTeX mathematical  equations when cursor  has not moved  recently.	string (never, in- line_only, always)	always
latex_sh\\\M\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	boolean	false
launcher <u>Hjødsto</u> ssørt jobs in the Launcher tab in RStudio Pro and RStudio Workbench.	string (recorded, state)	recorded

Property Description	Type	Default
limit_vistbleethensteleonly show a limited window of the total console output	boolean	false
line_endinge_honevensling format to use when saving files.	string (default, windows, posix, native, passthrough)	native
load_wollder to load the workspace when the R session begins.	boolean	true
margin_ddhemumber of columns of text after which the margin is shown.	integer	80
memory Hovermainteseadness dorwait between automatic requeries of memory statistics (0 to disable)	_	10
navigate Whetheildoeravigate to build errors.	boolean	true
new_pro Weithenia git repo should be initialized inside new projects by default.	boolean	false
new_pro\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	boolean	false
num_spaces to insert when pressing the Tab key.	integer	2
packages Whatther nobled ble RStudio's Packages pane.	boolean	true
panes Layout of panes in the RStudio workbench.	e object	

PropertyDescription	Туре	Default
pdf_previeweprogram to use to preview PDF files after generation.	string (none, default, rstudio, desktop- synctex, system)	default
plumber Wiewertotylpaplay Shiny applications when they are run.	string (user, none, pane, window, browser)	window
posix_tellihintdrishindl shell to use on POSIX operating systems (MacOS and Linux).	string (default, bash, zsh, custom, none)	default
project Salie istarbep of seconds seconds after which a project is deemed to have successfully started.	integer	30
publish The pataleto the custom certificate authority (CA) bundle to use when publishing content.	string	
publish Whekheettifidates remote server SSL certificates when publishing content.	boolean	true
python_patch path to the default Python interpreter.	string	

PropertyDescription	Type	Default
python_ <b>Wojec</b> tenarblindnifiehte_a	u <b>toomktan</b> _ac	tivantae
active project contains		
a Python virtual		
environment, then		
RStudio will		
automatically activate		
this environment on		
startup.		
python_ftylpee Python type.	string	
python_TerrioPython version.	string	
rainbow_Vpanetalutelretseshighlight	boolean	false
parentheses in a		
variety of colors.		
real_tim&V_lsptdlkahtokengble	boolean	true
real-time spellchecking		
by default.		
reduced_Randtion use of	boolean	false
animations in the user		
interface.		
reindent Whetphæsteo	boolean	true
automatically		
re-indent code when		
it's pasted into		
RStudio.		
remove_ <b>NV3tet</b> kr <u>er</u> dtophicantese	boolean	false
duplicate entries from		
the R console history.		
restore_Mshetphærjetæt restore	boolean	true
the last project when		
starting RStudio.		
restore_ <b>W</b> ojether <u>to</u> versitore	boolean	true
the last version of R		
used by the project in		
RStudio Pro and		
RStudio Workbench.	1 1	
restore_Mountedheddccusaevettheurse	or <u>b</u> ookston	true
position of the cursor		
when a file is closed,		
restore it when the file		
is opened.		

PropertyDescription	Type	Default
restore_\textit{Nthethedacuments} the last opened source documents when RStudio starts up.	boolean	true
reuse_se\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	xsboolean	false
rmd_chu <b>N</b> Nhe <b>chdeputo</b> simbiane chunk output inline for ordinary R Markdown documents.	boolean	true
rmd_prefeneepathentplate_path preferred R Markdown template.	string	
rmd_vie\ \text{weherty pe} \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	string (window, pane, none) string	window
rsa_key_Thathpath to the RSA key file to use.	string	
run_rprofflhetduer_tesume .Rprofile again after resuming a suspended R session.	boolean	false
save_befWhetshearding automatically save scripts before executing them.	boolean	true
save_file\( \frac{\text{N}}{\text{het}}\) bether thus does all open, unsaved files before building the project.	boolean	false
save_retilyhetinaxintum amount of seconds of retry for save operations.	integer	15

PropertyDescription	Type	Default
save_wolkspatcher to save the workspace to an .Rdata file after the R session ends.	string (always, never, ask)	ask
screenrealdaxinousolounabaousce lines of console output announced after a command.	_linteger	25
scroll_paththetro6_adbownent scrolling past the end of a file.	boolean	false
server_edihernatement of the fixed-width editor font to use with RStudio Server.	string	
server_e <b>Withortforntoensebl</b> ed custom editor font in RStudio Server.	boolean	false
session_pinatohelsessiong protocol debug logging showing all session requests and events	boolean	false
shiny_bandlerthundtojoths Shiny applications as background jobs.	boolean	false
shiny_vi <b>Whe</b> ret <b>tp</b> edisplay Shiny applications when they are run.	string (user, none, pane, window, browser)	window
show_dia\( \) showties_topphow diagnostic messages for C++ code as you type.	boolean	true
show_dia@hosties_totsleow diagnostic messages for other types of code (not R or C++).	boolean	false

PropertyDescription	Type	Default
show_diagnostic messages diagnostic messages (such as syntax and usage errors) for R code as you type.	boolean	true
show_doWhetthienetorshow the document outline by default when opening R Markdown documents.	boolean	false
show_fo@sntredtamidlekeyboard focus displays a visual focus indicator.	boolean	true
show_fuNVtiethesignashure_tooltic function signature tooltips during autocompletion.	p <b>s</b> oolean	true
show_helphethetipo_showdhelp tooltips for functions when the cursor has not been recently moved.	boolean	false
show_hiddentifeles show hidden files in the Files pane.	boolean	false
show_indvinet_genides show indentation guides in the RStudio code editor.	boolean	false
show_in \( \text{in the ethocal boar slfow_ar_cool} \) toolbar on code chunks in R Markdown documents.	le <u>b</u> addenks	true
show_interhed herritotishusw functions without source references in the Traceback pane while debugging.	boolean	false

PropertyDescription	Type	Default
show_in Wishelds er to show invisible characters, such as spaces and tabs, in the RStudio code editor.	boolean	false
show_lashodotthealesult of the last expression (.Last.value) in the Environment pane.	boolean	false
show_law\data begin slate the Launcher jobs tab in RStudio Pro and RStudio Workbench.	boolean	true
show_linghownhibernumbers in RStudio's code editor.	boolean	true
show_margin guide in the RStudio code editor.	boolean	true
show_methortheustageompute and show memory usage in the Environment Pane	boolean	true
show_pa <b>Sheb</b> wfowhichreataelgle contains keyboard focus.	boolean	false
show_pu\\disthdiatgns\tivs verbose diagnostic information when publishing content.	boolean	false
show_rm\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	boolean	false
show_te\texts\texts\texts show the Terminal tab.	boolean	true
show_us\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	string (always, never, sessions)	sessions

PropertyDescription	Type	Default
soft_wra\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	boolean	false
soft_wra\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	boolean	true
sort_file Whether that sortllfile names naturally, so that e.g., file 10.R comes after file 9.R	boolean	true
source_w\therefore the sourcing it.	boolean	false
spelling_ <b>That dist_of ictistuani</b> es dictionaries to use when spell checking.	array	Empty
spelling That ilanguaghang the spelling dictionary to use for spell checking.	string	en_US
strip_trawhitestripe trailing whitespace from each line when saving.	boolean	false
style_dia\( \) diagnostics to show style diagnostics (suggestions for improving R code style)	boolean	false
submit_Whishtheeptorts automatically submit crash reports to RStudio.	boolean	true
surround Wsclehtkinds of delimiters can be used to surround the	string (never, quotes,	quotes_and_brackets
current selection.	quotes_and_	_brackets)

PropertyDescription	Type	Default
svn_exe_T hathpath to the Subversion executable to use.	string	
sync_file Whather tookinggelir the directory in the Files pane automatically when the working directory in R changes.	boolean	false
syntax_Whetheensoluse syntax highlighting in the R console.	boolean	false
tab_com/Mentithmer to attempt completion of statements when pressing Tab.	boolean	true
tab_key_Tabokeyforouses focus out of text editing controls instead of inserting tabs.	boolean	false
tab_mul\( \forall \for	boolean	false
terminal Tershlinstly loell style	string (none, sound)	sound
terminal Webschebetkoavlose the terminal pane after the shell exits.	string (always, clean, never)	always
terminal <u>Initial dirictory</u> rfor new terminals.	string (project, current, home)	project
terminal Worsthercho use local echo in the Terminal. terminal That hath to the terminal executable to use.	boolean	true

PropertyDescription	Type	Default
terminal Epathler Pythtegration terminal hooks. When enabled, the RStudio-configured version of Python will be placed on the PATH.	boolean	true
terminal Termieuer rendering engine: canvas is faster, dom may be needed for some browsers or graphics cards	string (canvas, dom)	canvas
terminal Whatheentvitonckentd save changes to system environment variables in the Terminal.	boolean	true
terminal Wwettlineksweb links displayed in the Terminal tab are made clickable.	boolean	true
terminal Whetshockets use websockets to communicate with the shell in the Terminal tab.	boolean	true
toolbar_Wishther to show the toolbar at the top of the RStudio workbench.	boolean	true
typing_sNaturbedebfy_ms milliseconds to wait after last keystroke before updating live region.	integer	2000
use_data\vinpohter to use RStudio's data import feature.	boolean	true
use_devt\d\d\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	boolean	true

PropertyDescription	Type	Default
use_inte <b>Wet2</b> ther to use Internet2 for networking on R for Windows.	boolean	true
use_new White ther_transfiles newlines when saving Makefiles.	boolean	true
use_publishedaer brondse a custom certificate authority (CA) bundle when publishing content.	boolean	false
use_rcpp <u>W</u> henthplate use RCPP templates.	boolean	true
use_roxy@dmether to use Roxygen for documentation.	boolean	false
use_secu <b>W</b> ehedownloadse secure downloads when fetching R packages.	boolean	true
use_spaces when pressing the Tab key.	boolean	true
use_tinytese tinytex to compile .tex files.	boolean	false
vcs_autoAcfreshatically refresh VCS status?	boolean	true
vcs_enabled determined to enable RStudio's version control system interface.	boolean	true
vertically Walkigher argument it all in a align arguments to R function calls during automatic indentation.	leibtoolean	true
view_dir Wantether torniewchleek directory after running R CMD CHECK.	boolean	false

PropertyDescription	Type	Default
visual_mankdowne_ofdeeditor editor to use to provide code editing in visual mode	string (ace, codemir-ror)	ace
visual_mWiketown_toditiitg_canor canonical visual mode markdown when saving from source mode.	n <b>iwa</b> dlean	false
visual_mankdoofmuleditsingl_font_ editing mode font size, in points	<b>sintegen</b> oints	0
visual_mWilkeldwen_t@ditinlyle_is_de visual editing by default for new markdown documents	f <b>aod</b> tean	false
visual_mackdowspacintingorlist_s lists created in the visual editor	splating (tight, spaced)	spaced
visual_m\deltarkdowmn_edittagt_max_ width for visual editing mode, in pixels	- /	E#00
visual_nParkdowent_editing_reference footnotes within markdown output.	entringlocation (block, section, document)	block
visual_mWilkeldnen_t@dslting_tlshow_document outline by default when opening R Markdown documents in visual mode.	<u>b</u> dod <u>e</u> antline	true
visual_mWilkelduen_t@dsitting_tlshow margin guide in the visual mode code blocks.	<u>bookgin</u>	false
visual_mWiketown_toditing_wrap automatically wrap text when writing markdown	string (none, column, sentence)	none

PropertyDescription	Type	Default
visual_makkowwmmeditingraprap_iategeorlumn text at when writing markdown		72
warn_if Wholethuchtovgrindratein_ warning if a variable is used without being defined in the current scope.	_s <b>copol</b> ean	false
warn_valldhdehedefingdndrate_not_busddan warning if a variable is defined without being used in the current scope		false
windows <u>Tterntairalin</u> shell to use on Windows.	string (default, win-git- bash, win-wsl- bash, win-cmd, win-ps, ps-core, custom, none)	default
wrap_tally hat high at ion wrap around when going to the previous or next editor tab.	boolean	true
zotero_lizoteries libraries to insert citations from.	array	My Library

## **B. RStudio IDE Commands**

The following table enumerates all of the commands currently supported by RStudio. These command IDs can be used in RStudio API calls such as rstudioapi::executeCommand.

Id	Description
activateBuild	Show Build
activateCompilePDF	Show Compile PDF
activateConnections	Show Connections
activateConsole	Move Focus to Console
activateConsolePane	Move Focus to Console Panel
activateData	Show Data
activateDeployContent	Show Deploy Content
activateEnvironment	Show Environment
activateFiles	Show Files
activateFindInFiles	Show Find in Files
activateHelp	Move Focus to Help
activateHistory	Show History
activateJobs	Show Jobs
activateLauncherJobs	Show Launcher
activateMarkers	Show Markers
activatePackages	Show Packages
activatePlots	Show Plots
activatePresentation	Show Presentation
activateRMarkdown	Show R Markdown
activateSource	Move Focus to Source
activateSourceCpp	Show Source Cpp
activate SQLR esults	Show SQL Results
activateTerminal	Move Focus to Terminal
activateTutorial	Show Tutorial
activateVcs	Show Vcs
activateViewer	Show Viewer
addCursorAbove	Add Cursor Above Current Cursor
addCursorBelow	Add Cursor Below Current Cursor
authoring RP resentations Help	Guide to using R Markdown
browseAddins	Browse addins

Id	Description
browseCheatSheets	Browse available cheat sheets in your web browser
buildAll	Install the package and restart R
buildBinaryPackage	Build a binary package
buildSourcePackage	Build a source package
buildToolsProjectSetup	Configure build tools
checkForUpdates	Check for Updates
checkPackage	R CMD check
checkSpelling	Check spelling in document
cleanAll	Clean all
clearBuild	Clear build
clearCommandPaletteMru	Clear Recently Executed Command List
clearHelpHistory	Clear history
clearHistory	Clear all history entries
clearJobs	Clean up all completed local jobs
clearKnitrCache	Clear the knitr cache for the current document
clearPlots	Clear all Plots
clearPrerenderedOutput	Clear the prerendered output for the current document
clearPresentationCache	Clear knitr cache for this presentation
clearRecentFiles	Clear List
clearRecentProjects	Clear Project List
clearTerminalScrollbackBuffer	Clear terminal
clearUserPrefs	Clear User Prefs
clearWorkspace	Clear objects from the workspace
closeAllSourceDocs	Close All
closeAllTerminals	Close All Terminals
closeOtherSourceDocs	Close All Except Current
closeProject	Close the currently open project
closeSourceDoc	Close
closeTerminal	Close current terminal session
codeCompletion	Show code completions at the current cursor location
commentUncomment	Comment or uncomment the current line/selection
compileNotebook	Compile a report from the current R script
compilePDF	Compile a PDF from the current LaTeX or Sweave document
consoleClear	Clear console
copyDummy	Сору
copyFile	Copy selected file or folder
copyFileTo	Copy selected file or folder to another folder
copyLinesDown	Copy Lines Down
copyPlotToClipboard	Copy the current plot to the clipboard
copySourceDocPath	Copy current document path
cutDummy	Cut

Id	Description
debugBreakpoint	Set or remove a breakpoint on the current line of code
${\it debugClearBreakpoints}$	Remove all the breakpoints in the current project
debugContinue	Continue execution until the next breakpoint is encountered
debugDumpContents	Dump Editor Contents
debugFinish	Execute the remainder of the current function or loop
debugHelp	Guide to debugging features
debugImportDump	Import Editor Contents
debugStep	Execute the next line of code
debugStepInto	Step into the current function call
debugStop	Exit debug mode
deleteFiles	Delete selected files or folders
devtoolsLoadAll	Execute devtools::load_all
diagnosticsReport	Write Diagnostics Report
disconnectConnection	Disconnect from a connection
editCodeSnippets	Edit code snippets
editLinesFromStart	Create a new cursor at start of each line in selection
editRmdFormatOptions	Edit the R Markdown format options for the current file
editUserPrefs	Edit User Prefs File
enableProsemirrorDevTools	Enable Prosemirror DevTools
errorsBreak	Break when any unhandled error occurs
errorsMessage	Print the error message when an unhandled error occurs
errorsTraceback	Show the error inspector when an unhandled error occurs
executeAllCode	Run all of the code in the source file
executeCode	Run the current line or selection
executeCodeWithoutMovingC	ur <b>For</b> n the current line or selection without moving the cursor
executeCurrentChunk	Run the current code chunk
executeCurrentFunction	Run the top-level function definition, if any, that contains the cursor
executeCurrentLine	Execute the line which contains the cursor
execute Current Paragraph	Execute the current paragraph of code, delimited by blank lines.
executeCurrentSection	Run the code section that contains the cursor
executeCurrentStatement	Execute the entire R statement which contains the cursor.
executeFromCurrentLine	Run from the current line through the end of the source file
executeLastCode	Re-run the previous code region
executeNextChunk	Run the next code chunk
executePreviousChunks	Run all chunks above the current one
executeSetupChunk	Run the initial setup chunk
executeSubsequentChunks	Run all chunks below the current one
executeToCurrentLine	Run from the beginning of the source file up through the current line

Id	Description
expandSelection	Expand selection
expandToLine	Expand Selection to Line
expandToMatching	Expand selection to matching bracket
exportFiles	Export selected files or folders
extractFunction	Turn the current selection into a function
extractLocalVariable	Extract a variable out of the current selection
findFromSelection	Use Selection for Find
findInFiles	Find in Files
findNext	Find next occurrence
findPrevious	Find previous occurrence
findReplace	Find
findSelectAll	Find and select all matches
findUsages	Find source locations where this symbol is used
firstTab	First Tab
focusCenterSeparator	Adjust Center Splitter
focusConsoleOutputEnd	Focus Console Output
focusLeftSeparator	Adjust Left Splitter
focusMainToolbar	Focus Main Toolbar
focusNextPane	Focus Next Pane
focusPreviousPane	Focus Previous Pane
focusRightSeparator	Adjust Right Splitter
focusSourceColumnSeparator	Adjust Source Column Splitter
fold	Collapse
foldAll	Collapse All
forceQuitSession	Quit the current R session even if busy
freeUnusedMemory	Free Unused R Memory
goToDefinition	Go to to the definition of the currently selected function
goToFileFunction	Go To File/Function
goToHelp	Go to help for the currently selected function
goToLine	Go to Line
goToNextChunk	Go to next chunk
goToNextSection	Go to next section/chunk
goToPrevChunk	Go to previous chunk
goToPrevSection	Go to previous section/chunk
gotoProfileSource	Open sources associated with the selection
goToWorkingDir	View the current working directory
helpBack	Previous topic
helpForward	Next topic
helpHome	Show R Help
helpKeyboardShortcuts	Keyboard Shortcuts Help
helpPopout	Show in new window
neipi opout	DHOW HI HEW WILLIAM

Id	Description
helpSearch	Search R Help
helpUsingRStudio	RStudio Docs
hideToolbar	Hide Toolbar
historyDismissContext	« Back
historyDismissResults	Done
historyRemoveEntries	Remove the selected history entries
historySendToConsole	Send the selected commands to the R console (Enter)
historySendToSource	Insert the selected commands into the current document
J	(Shift+Enter)
importDatasetFromCsv	From CSV
importDatasetFromCsvUsingB	astrom Text (base)
importDatasetFromCsvUsingR	· · · · · · · · · · · · · · · · · · ·
importDatasetFromFile	From Local File
importDatasetFromSAS	From SAS
importDatasetFromSAV	From SPSS
importDatasetFromStata	From Stata
importDatasetFromURL	From Web URL
importDatasetFromXLS	From Excel
insertChunk	Insert a new code chunk
insertChunkBash	Insert a new Bash chunk
insertChunkD3	Insert a new D3 chunk
insertChunkPython	Insert a new Python chunk
insertChunkR	Insert a new R chunk
insertChunkRCPP	Insert a new Rcpp chunk
insertChunkSQL	Insert a new SQL chunk
insertChunkStan	Insert a new Stan chunk
insert Roxygen Skeleton	Insert a roxygen comment for the current function
insertSection	Insert a new code section
insertSnippet	Expand snippet at cursor
installPackage	Install R packages
interruptR	Interrupt R
interrupt Terminal	Send Ctrl+C to Current Terminal
joinLines	Join Lines
jumpTo	Jump To
jumpToMatching	Jump to matching bracket
knitDocument	Knit the current document
knitWithParameters	Knit the document with a set of custom parameters
lastTab	Last Tab
layout Console On Left	Console on Left
layout Console On Right	Console on Right
layout End Zoom	Show All Panes

Id	Description
layoutZoomBuild	Zoom Build
layoutZoomConnections	Zoom Connections
layoutZoomConsole	Zoom Console
layoutZoomConsolePane	Zoom Console Pane
layoutZoomEnvironment	Zoom Environment
layoutZoomFiles	Zoom Files
layoutZoomHelp	Zoom Help
layoutZoomHistory	Zoom History
layoutZoomLeftColumn	Zoom Left / Center Column
layoutZoomPackages	Zoom Packages
layoutZoomPlots	Zoom Plots
layoutZoomRightColumn	Zoom Right Column
layoutZoomSource	Zoom Source
layoutZoomTutorial	Zoom Tutorial
layoutZoomVcs	Zoom VCS
layoutZoomViewer	Zoom Viewer
loadHistory	Load history from an existing file
loadServerHome	RStudio Server Home
loadWorkspace	Load workspace
logFocusedElement	Log focused element
macPreferences	Preferences
markdownHelp	Markdown quick reference
maximizeConsole	Maximize Console
modifyKeyboardShortcuts	Modify keyboard shortcuts
moveFiles	Move selected files or folders
moveLinesDown	Move Lines Down
moveLinesUp	Move Lines Up
moveTabLeft	Move Tab Left
moveTabRight	Move Tab Right
moveTabToFirst	Move Tab to First
moveTabToLast	Move Tab to Last
newCDoc	Create a new C file
newConnection	Create a new connection
newCppDoc	Create a new C++ file
newCssDoc	Create a new CSS file
newD3Doc	Create a new D3 Script
newFolder	Create a new folder
${\it newHeaderDoc}$	Create a new header file
${\it newHtmlDoc}$	Create a new HTML file
${\it new Java Script Doc}$	Create a new JavaScript file
${\it newMarkdownDoc}$	Create a new Markdown document

Id	Description
newProject	Create a project
newPythonDoc	Create a new Python script
newQuartoDoc	Create a new Quarto document
newRDocumentationDoc	Create a new Rd documentation file
newRHTMLDoc	Create a new R HTML document
newRMarkdownDoc	Create a new R Markdown document
newRNotebook	Create a new R Markdown notebook
newRPlumberDoc	Create a new Plumber API
newRPresentationDoc	Create a new R presentation
newRShinyApp	Create a new Shiny web application
newSession	Open a new R session
newShellDoc	Create a new shell script
newSourceColumn	Add Source Column
newSourceDoc	Create a new R script
newSqlDoc	Create a new SQL script
newStanDoc	Create a new Stan program
newSweaveDoc	Create a new R Sweave document
newTerminal	Create a new terminal
newTextDoc	Create a new text file
nextPlot	Next plot
nextTab	Next Tab
nextTerminal	Show next terminal
${\bf notebook Clear All Output}$	Remove all code chunk output in the current file
notebook Clear Output	Clear the output of the current notebook chunk
${\bf notebook Collapse All Output}$	Collapse all code chunk output in the current file
${\bf notebook Expand All Output}$	Expand all code chunk output in the current file
${\bf notebook Toggle Expansion}$	Expand or collapse the output of the current notebook chunk
open Data Import Cheat Sheet	Import data with readr
openDataTransformationCheatSDatta transformation with dplyr	
openDataVisualizationCheatSheData visualization with ggplot2	
open Data Wrangling Cheat Shee	t Data manipulation with dplyr and tidyr
${\it open Developer Console}$	Open Developer Console
openHtmlExternal	View the page with the system web browser
open New Terminal At Editor Local Control of the	at@ppen New Terminal at File Location
openNewTerminalAtFilePaneLo@picm New Terminal Here	
openPackageDevelopmentChea	at Shadtage development with devtools
openProfile	Opens a profile from a file
open Profile In Browser	Opens current profile in a web browser
openProject	Open a project
open Project In New Window	Open project in a new R session
open Purrr Cheat Sheet	List manipulation with purrr

Id Description

openRMarkdownCheatSheet R Markdown cheat sheet openRMarkdownReferenceGuideR Markdown reference guide openRoxygenQuickReference Roxygen quick reference openRStudioIDECheatSheet RStudio IDE cheat sheet

openSharedProject Open a project shared with you openShinyCheatSheet Build web applications with Shiny

openSourceDoc Open an existing file

openSourceDocNewColumn Open an existing file in a new column openSparklyrCheatSheet Interfacing Apache Spark with sparklyr

packratBootstrap Use packrat with this project packratBundle Bundle a Packrat Project

packratCheckStatus Check the status of the Packrat library

packratClean Remove unused packages from your packrat library

packratHelp Help on using packrat with R projects packratOptions Configure packrat options for this project

paneLayout Pane Layout...

pasteDummy Paste

pasteWithIndentDummy Paste with Indent

plumberRunInBrowser Run the Plumber API in the system's default Web browser

plumberRunInPane Run the Plumber API in an RStudio pane

plumberRunInViewer Run the Plumber API in an RStudio viewer window

popoutDoc Show in new window

presentationEdit Edit this slide of the presentation presentationFullscreen Show presentation in full screen mode

presentationHome Go to the first slide
presentationNext Go to the next slide
presentationPrev Go to the previous slide

presentationSaveAsStandalone Save the presentation as a standalone web page presentationViewInBrowser View the presentation in an external web browser previewHTML Show a preview of the current document as HTML

previewJS Preview the active JavaScript document previewSql Preview the active SQL document

previousPlot Previous plot previousTab Previous Tab

 $\begin{array}{ll} previous Terminal & Show previous terminal \\ print Cpp Completions & Print C++ Completions \end{array}$ 

printHelp Print topic

printSourceDoc Print the current file

profileCode Profile the current line or selection

profileHelp Guide to profiling features

projectOptions Edit options for the current project

publishHTML         Publish the current document           quartARenderDocument         Render the current document           quitAResision         Quit the current R session           raiseException         Raise Exception           raiseException2         Raise Exception JS           repHelp         Help on using Rcpp           rebuildAll         Clean previous output and rebuild all           redoOummy         Redo           reflowComment         Reflow selected comment lines so they wrap evenly           reformat Code         Reformat the current line/selection           refreshEnvironment         Refresh data           refreshEnvironment         Refresh the list of objects in the environment           refreshHelp         Refresh the preview           refreshHullPreview         Refresh the preview           refreshPackages         Refresh package listing           refreshPresentation         Refresh Workspace           refreshWorkspace         Refresh Workspace           reindent         Reindent the current line/selection           reloadPlumberAPI         Reload the Plumber API           reload Ui         Reload the Shiny application           renoweConnection         Remove tine           removeDine         Remove Line           <	Id	Description
quitckAddNext quitch current R session raiseException Raise Exception raiseException2 Raise Exception JS reppHelp Help on using Rcpp rebuildAll Clean previous output and rebuild all redoDummy Redo refreshComment Reflow selected comment lines so they wrap evenly reformatCode Reformat the current line/selection refreshConnection Refresh data refreshEnvironment Refresh the list of objects in the environment refreshEnles Refresh the list of objects in the environment refreshHullPreview Refresh topic refreshHullPreview Refresh the preview refreshPackages Refresh Package listing refreshPot Refresh urrent plot refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application reloadUi Reload UI removeConnection Remove connection from the connection history removeLine Remove the current plot renameFile Rename selected file or folder renameFile Rename selected file or folder renameTerminal Change terminal session name remVHelp Learn how to use renv renvRestore Restore your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR Restart R restartRRunAllChunks Restart R session and run all chunks	publishHTML	Publish the current document
quitSession Quit the current R session raiseException Raise Exception raiseException2 Raise Exception JS rcppHelp Help on using Rcpp rebuildAll Clean previous output and rebuild all redoDummy Redo reflowComment Reflow selected comment lines so they wrap evenly reformatCode Reformat the current line/selection refreshConnection Refresh data refreshEnvironment Refresh the list of objects in the environment refreshFiles Refresh file listing refreshHelp Refresh topic refreshHumlPreview Refresh topic refreshPackages Refresh Package listing refreshPot Refresh current plot refreshPresentation Refresh the presentation refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application reloadUi Reload UI removeConnection Remove connection from the connection history removeLine Remove the current plot renameFile Rename selected file or folder renameFile Rename selected file or folder renameSourceDoc Rename current document renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvSanpshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	quartoRenderDocument	Render the current document
raiseException	quickAddNext	Find and add next occurrence
raiseException	quitSession	Quit the current R session
raiseException2 Raise Exception JS rcppHelp rebuildAll Clean previous output and rebuild all redoDummy Redo reflowComment Reflow selected comment lines so they wrap evenly reformatCode Reformat the current line/selection refreshConnection Refresh data refreshEnvironment Refresh the list of objects in the environment refreshFiles Refresh file listing refreshHelp Refresh topic refreshHumlPreview Refresh the preview refreshPackages Refresh Package listing refreshPlot Refresh current plot refreshPresentation Refresh workspace reindent Reload the Plumber API reloadShinyApp Reload the Shiny application removeConnection Remove Line removeLine Remove Line removePlot Renowe Line removePlot Renowe Line removePlot Renome selected file or folder renamefile Rename selected file or folder renameforminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding repstartRRunAllChunks Restart R session and clear chunk output restartRrestartRunAllChunks Restart R session and run all chunks		Raise Exception
rebuildAll Redo reflowComment Reflow selected comment lines so they wrap evenly reformatCode Reformat the current line/selection refreshConnection Refresh data refreshEnvironment Refresh the list of objects in the environment refreshFiles Refresh file listing refreshHelp Refresh topic refreshHelp Refresh topic refreshPackages Refresh Package listing refreshPackages Refresh workspace reindent Reindent the current line/selection refreshWorkspace Refresh Workspace reindent Reload the Plumber API reloadShinyApp Reload the Shiny application reloadPlumberAPI Reload UI removeConnection Remove connection from the connection history removeLine Remove the current plot renameFile Rename selected file or folder renameFore renameSourceDoc Rename surrent document renawlelp renvRestore Restore your project library from renv.lock renvSnapshot Restart R restartRRunAllChunks Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	raiseException2	Raise Exception JS
redoDummy reflowComment reflowComment reflowComment Reflow selected comment lines so they wrap evenly reformatCode Reformat the current line/selection refreshConnection Refresh data refreshEnvironment Refresh the list of objects in the environment refreshFiles Refresh the listing refreshHelp Refresh topic refreshHelp Refresh the preview refreshPackages Refresh Package listing refreshPlot Refresh current plot refreshPresentation refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload UI Reload UI Remove Connection Remove connection from the connection history removePlot Remove Line removePlot Rename selected file or folder renameFile Rename selected file or folder renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Replace and find next occurrence restartR Restart R Restart R Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	rcppHelp	
reflowComment reformatCode Reformat the current line/selection refreshConnection Refresh data refreshEnvironment Refresh the list of objects in the environment refreshFiles Refresh the list of objects in the environment refreshFiles Refresh the listing refreshHelp Refresh topic refreshPackages Refresh Package listing refreshPackages Refresh Package listing refreshPresentation refreshWorkspace Refresh Workspace reindent Refresh Workspace reindent Reload the Plumber API reloadShinyApp Reload the Shiny application removeConnection Remove connection from the connection history removeLine Remove the current plot renameFile Rename selected file or folder renameSourceDoc Rename current document renaweTerminal Change terminal session name renvHelp Learn how to use renv repolaceAndFind Replace and find next occurrence restartRR Restart R Restart R session and run all chunks	rebuildAll	Clean previous output and rebuild all
reformatCode Reformat the current line/selection refreshConnection Refresh data refreshEnvironment Refresh the list of objects in the environment refreshFiles Refresh file listing refreshHelp Refresh topic refreshHumlPreview Refresh the preview refreshPackages Refresh Package listing refreshPackages Refresh current plot refreshPresentation Refresh the presentation refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRRunAllChunks Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	redoDummy	Redo
reformatCode Reformat the current line/selection refreshConnection Refresh data refreshEnvironment Refresh the list of objects in the environment refreshFiles Refresh the list of objects in the environment refreshFiles Refresh the listing refreshHelp Refresh topic refreshHelp Refresh topic refreshHulPreview Refresh the preview refreshPackages Refresh Package listing refreshPosentation Refresh current plot refreshPresentation Refresh the presentation refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR RenvAllChunks Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks		Reflow selected comment lines so they wrap evenly
refreshEnvironment Refresh the list of objects in the environment refreshFiles Refresh file listing refreshHelp Refresh topic refreshHtmlPreview Refresh the preview refreshPackages Refresh Package listing refreshPot Refresh Package listing refreshPresentation Refresh workspace refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renveRestore Restore your project library from renv.lock renvSnapshot Replace and find next occurrence restartR Restart R restartRClearOutput Restart R session and run all chunks	$\operatorname{reformatCode}$	
refreshFiles Refresh file listing refreshHelp Refresh topic refreshHtmlPreview Refresh the preview refreshPackages Refresh Package listing refreshPlot Refresh current plot refreshPresentation Refresh the presentation refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application removeConnection Remove connection from the connection history removeLine Remove Line removePlot Rename selected file or folder renameFile Rename symbol in current scope renameSourceDoc Rename current document renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library restartR RenallChunks Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	refreshConnection	,
refreshFiles Refresh file listing refreshHelp Refresh topic refreshHtmlPreview Refresh the preview refreshPackages Refresh Package listing refreshPresentation Refresh current plot refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remame selected file or folder renameFile Rename surrent document renameSourceDoc Rename current document renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	refreshEnvironment	Refresh the list of objects in the environment
refreshHtmlPreview Refresh the preview refreshPackages Refresh Package listing refreshPlot Refresh current plot refreshPresentation Refresh the presentation refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application reloadUi Remove Connection from the connection history removeConnection Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and run all chunks	refreshFiles	
refreshPackages Refresh Package listing refreshPlot Refresh current plot refreshPresentation Refresh the presentation refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application reloadUi Remove Connection from the connection history removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renawTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	refreshHelp	Refresh topic
refreshPlot Refresh current plot refreshPresentation Refresh the presentation refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application reloadUi Remove Connection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and run all chunks	refreshHtmlPreview	Refresh the preview
refreshPresentation Refresh the presentation refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application reloadUi Remove Connection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	refreshPackages	Refresh Package listing
refreshWorkspace Refresh Workspace reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application reloadUi Reload UI removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	refreshPlot	Refresh current plot
reindent Reindent the current line/selection reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application reloadUi Reload UI removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	refreshPresentation	Refresh the presentation
reloadPlumberAPI Reload the Plumber API reloadShinyApp Reload the Shiny application reloadUi Reload UI removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	refreshWorkspace	Refresh Workspace
reloadShinyApp Reload the Shiny application reloadUi Reload UI removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	reindent	Reindent the current line/selection
reloadUi Reload UI removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	${\it reloadPlumberAPI}$	Reload the Plumber API
removeConnection Remove connection from the connection history removeLine Remove Line removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	reloadShinyApp	Reload the Shiny application
removeLine removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	reloadUi	Reload UI
removePlot Remove the current plot renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	remove Connection	Remove connection from the connection history
renameFile Rename selected file or folder renameInScope Rename symbol in current scope renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	removeLine	Remove Line
renameInScope Rename symbol in current scope renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	removePlot	Remove the current plot
renameSourceDoc Rename current document renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR Restart R restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	renameFile	Rename selected file or folder
renameTerminal Change terminal session name renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR Restart R restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	rename In Scope	Rename symbol in current scope
renvHelp Learn how to use renv renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR Restart R restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	rename Source Doc	Rename current document
renvRestore Restore your project library from renv.lock renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR Restart R restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	rename Terminal	Change terminal session name
renvSnapshot Snapshot the state of your project library reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR Restart R restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	$\operatorname{renvHelp}$	Learn how to use renv
reopenSourceDocWithEncoding Reopen the current file with a different encoding replaceAndFind Replace and find next occurrence restartR Restart R Restart R Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	renvRestore	Restore your project library from renv.lock
replaceAndFind Replace and find next occurrence restartR Restart R restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	renvSnapshot	Snapshot the state of your project library
restartR Restart R restartRClearOutput Restart R session and clear chunk output restartRRunAllChunks Restart R session and run all chunks	reopenSourceDocWithEncoding	Reopen the current file with a different encoding
restartRClearOutput Restart R session and clear chunk output restartRunAllChunks Restart R session and run all chunks	replace And Find	Replace and find next occurrence
restartRRunAllChunks Restart R session and run all chunks	restartR	Restart R
		Restart R session and clear chunk output
returnDocToMain Return to main window	restart RRun All Chunks	Restart R session and run all chunks
	${\it return} {\it Doc} {\it ToMain}$	Return to main window
roxygenizePackage Build package documentation	roxygenizePackage	Build package documentation

Id	Description
	*
rsconnectConfigure	Configure the application
rsconnectDeploy	Publish the application or document
rsconnectManageAccounts	Connect or disconnect accounts
rstudio Community Forum	RStudio Community Forum
rstudioLicense	RStudio License
rstudioSupport	RStudio Support
runDocumentFromServerDotR	Run the interactive document
runSelectionAsJob	Run the selected code as a local job
${\rm run Selection As Launcher Job}$	Run the selected code as a launcher job
save All Source Docs	Save all open documents
saveHistory	Save history into a file
save Html Preview As	Save the page to another location
save Html Preview As Local File	Download the page to a local file
savePlotAsImage	Save the current plot as an image file
savePlotAsPdf	Save the current plot as a PDF file
saveProfileAs	Saves current profile into a file
save Source Doc	Save current document
save Source Doc As	Save current file to a specific path
save Source Doc With Encoding	Save the current file with a different encoding
saveWorkspace	Save workspace as
send Filename To Terminal	Send Filename to Terminal
sendTerminalToEditor	Copy current terminal's buffer to a new editor buffer
sendToTerminal	Send the current line or selection to terminal
serveQuartoSite	Run development server for Quarto site
setTerminalToCurrentDirectory	
setWorkingDir	Select and change to a new working directory
${\it setWorkingDirToActiveDoc}$	Change working directory to path of active document
setWorkingDirToFilesPane	Change working directory to location of Files pane
setWorkingDirToProjectDir	Change working directory to project root directory
shareProject	Share this project with others
shinyCompareTest	Compare test results for Shiny application
shinyRecordTest	Record test for Shiny application
shinyRunAllTests	Run tests for Shiny application
shinyRunInBrowser	Run the Shiny application in the system's default Web browser
shinyRunInPane	Run the Shiny application in an RStudio pane
shinyRunInViewer	Run the Shiny application in an RStudio viewer window
showAboutDialog	About RStudio
showAccessibilityHelp	Accessibility Help
showAccessibilityOptions	Accessibility Options
showBuildMenu	Show Build Menu
showCodeMenu	Show Code Menu

 $\operatorname{Id}$ Description showCommandPalette Show Command Palette showDebugMenu Show Debug Menu showDiagnosticsActiveDocumenShow diagnostics for the active document showDiagnosticsProject Show diagnostics for all source files in the current project showDomElements Show DOM Elements showEditMenu Show Edit Menu showFileMenu Show File Menu showGpuDiagnostics Show GPU Diagnostics showHelpMenu Show Help Menu showHtmlPreviewLog Show the compilation log for this document showLicenseDialog Manage License... Show Log Files showLogFiles showManipulator Show the manipulator for this plot showMemoryUsageReport Memory Usage Report... showOptions Global Options... showPdfExternal Show in an external PDF viewer window showPlotsMenu Show Plots Menu showProfileMenu Show Profile Menu showRequestLog Show internal request log Show Session Menu showSessionMenu showSessionServerOptionsDialogConfigure available session servers Execute shell commands showShellDialog showShortcutCommand Show Keyboard Shortcut Commands showTerminalInfo Show info on current terminal showTerminalOptions Terminal Options... Show Toolbar showToolbar showToolsMenu Show Tools Menu showViewMenu Show View Menu showWarningBar Show warning bar shrinkSelection Shrink selection signOut Sign out from RStudio sortLauncherJobsRecorded Sort jobs by time submitted sortLauncherJobsState Sort jobs by current state sourceActiveDocument Source the contents of the active document sourceActiveDocumentWithEch&ource the contents of the active document (with echo) sourceAsJob Run the current R script as a local job sourceAsLauncherJob Run the current R script on a cluster sourceFile Source the contents of an R file sourceNavigateBackGo back to the previous source location sourceNavigateForward Go forward to the next source location sparkHelp Help on using Spark with RStudio

Id	Description	
sparkLog	View the log for the Spark connection	
sparkUI	View the browser UI for the Spark connection	
speakEditorLocation	Speak Text Editor Location	
splitIntoLines	Create a new cursor on each line in current selection	
startJob	Run a background local job	
startLauncherJob	Run a background job on a cluster	
startProfiler	Start profiling R code	
stopBuild	Stop the current build	
stopProfiler	Stop profiling R code	
suspendSession	Suspend R Session	
switchToChunkBash	Switch chunk to Bash	
switchToChunkPython	Switch chunk to Python	
switchToChunkR	Switch chunk to R	
switchToChunkRCPP	Switch chunk to Rcpp	
switchToChunkSQL	Switch chunk to SQL	
switchToChunkStan	Switch chunk to Stan	
switchToTab	Switch to Tab	
synctexSearch	Sync PDF view to editor location (Ctrl+Click)	
terminateR	Forcibly terminate R session	
testPackage	Run tests for package	
testShinytestFile	Run test using the shinytest package	
testTestthatFile	Run tests using the testthat package	
toggleDocumentOutline	Show document outline	
toggle Editor Token Info	Toggle Editor Token Information	
toggleFullScreen	Toggle Full Screen	
toggle Rainbow Parens	Rainbow Parentheses	
toggleRmdVisual $Mode$	Toggle visual markdown editor	
toggle Screen Reader Support	Screen Reader Support	
toggle Show Memory Usage	Show Current Memory Usage	
toggle Soft Wrap Mode	Soft Wrap Long Lines	
toggle Tab Key Moves Focus	Tab Key Always Moves Focus	
toggle Toolbar	Toggle Toolbar	
tutorialBack	Go back	
tutorial Forward	Go forward	
tutorialHome	Return to home	
tutorial Popout	Show in new window	
tutorialRefresh	Refresh tutorial	
tutorialStop	Stop tutorial	
tutorial Zoom	View a larger version in a new window	
undoDummy	Undo	
unfold	Expand	

 $\operatorname{Id}$ Description unfoldAll Expand All updateCredentials Update Credentials updatePackages Check for package updates uploadFile Upload files to server usingRMarkdownHelp Guide to using R Markdown vcsAddFiles Add the selected files or folders vcsBlameOnGitHub Blame view for this file on Github vcsCleanup Recursively clean up the working copy (removing locks, etc) vcsCommit Commit pending changes vcsDiff Diff selected file(s) vcsFileDiff Show differences for the file vcsFileLog Show log of changes to the file vcsFileRevert Revert changes to the file vcsIgnore Ignore the selected files or folders vcsOpen Open selected file(s) vcsPull Pull Branches vcsPullRebase Pull with Rebase vcsPush Push Branch vcsRefresh Refresh listing vcsRemoveFiles Delete the selected files or folders vcsResolve Resolve conflicts in the selected files or folders vcsRevert Revert selected changes vcsShowHistory View history of previous commits vcsViewOnGitHub View this file on Github versionControlHelp Help on using version control with RStudio versionControlOptions Configure version control options versionControlProjectSetup Setup version control for the current project versionControlShowRsaKey Show RSA public key viewAllPrefs View All Prefs viewerBack Go back viewerClear Remove current viewer item viewerClearAll Clear all viewer items viewerCopyToClipboard Copy to the system clipboard viewerForward Go forward viewerPopout Show in new window viewerRefresh Refresh viewer viewerSaveAllAndRefresh Save source files and refresh viewer viewerSaveAsImage Save as an image file viewerSaveAsWebPage Save as a standalone web page viewerStop Stop application

View a larger version in a new window

viewerZoom

Id	Description
wordCount	Count words in selection or document
zoomActualSize	Actual Size
zoomIn	Zoom In
zoomOut	Zoom Out
zoomPlot	View a larger version of the plot in a new window

# C. R Package Dependencies

The following is a list of all of the R packages RStudio depends on in some way. None of these packages are necessary for the basic operation of RStudio; they all enable additional IDE functionality as noted in in *Features*.

This information is also available in the IDE itself using the **rstudioapi** package as follows:

#### rstudioapi::getRStudioPackageDependencies()

Note that some of these packages may require a higher version of R than RStudio itself requires; consult the individual package's documentation for more details.

Name	Version	n Features
base64en		R Markdown
crayon	1.3.4	Plumber R APIs, Shiny
$\operatorname{curl}$	4.2	Publishing
DBI	0.8	Database Interface, RSQLite
devtools	1.11.1	R Unit Testing
digest	0.6	R Markdown, Shiny
evaluate	0.13	R Markdown
glue	1.3.0	R Markdown
haven	0.2.0	SPSS/SAS/Stata Import
highr	0.3	R Markdown
htmltools	0.3.6	R2D3, R Markdown, Shiny, Visual Profiling
htmlwidg	ets2	R2D3, Visual Profiling
httpuv	1.3.3	Plumber R APIs, Shiny
jsonlite	0.9.19	R2D3, Plumber R APIs, Publishing, R Markdown, Shiny, Python/Reticulate,
		JSON Import, Mongo DB Import, Visual Profiling
keyring	1.1.0	Secret Management
knitr	1.22	R Markdown
later	0.7.2	Shiny
learnr	0.10.1	RStudio Tutorials
magrittr	1.5	R Markdown
markdow	n0.7	R Markdown
mime	0.5	R Markdown, Shiny

miniUI 0.1.1 Shiny Add-Ins mongolite 0.8 odbc 1.1.6 ODBC Connections openssl 1.0.2 Publishing packrat 0.4.8- Packrat, Publishing 1 plumber 1.0.0 Plumber R APIs png 0.1-7 Python/Reticulate profvis 0.3.2 Visual Profiling
odbc 1.1.6 ODBC Connections openssl 1.0.2 Publishing packrat 0.4.8- Packrat, Publishing 1 plumber 1.0.0 Plumber R APIs png 0.1-7 Python/Reticulate
odbc 1.1.6 ODBC Connections openssl 1.0.2 Publishing packrat 0.4.8- Packrat, Publishing 1 plumber 1.0.0 Plumber R APIs png 0.1-7 Python/Reticulate
packrat 0.4.8- Packrat, Publishing 1 plumber 1.0.0 Plumber R APIs png 0.1-7 Python/Reticulate
plumber 1.0.0 Plumber R APIs png 0.1-7 Python/Reticulate
plumber 1.0.0 Plumber R APIs png 0.1-7 Python/Reticulate
png 0.1-7 Python/Reticulate
profvis 0.3.2 Visual Profiling
•
promises 1.0.1 Shiny
r2d3 0.2.2 $R2D3$
R6 2.0 Plumber R APIs, Shiny
ragg 0.1.5 R Interface to AGG
Rcpp 0.11.5 Shiny, CSV Import, SPSS/SAS/Stata Import, Excel Import
readr 1.1.0 CSV Import
readxl 0.1.0 Excel Import
renv 0.9.3 renv
reticulate 1.20 Python/Reticulate
rJava 0.4- JDBC Import
DIDDG 0.05 IDDG I
RJDBC 0.2-5 JDBC Import
rlang 0.2.2 Shiny rmarkdown2.10 R Markdown
RODBC 1.3- ODBC Import
12
roxygen2 6.0.1 R Package Documentation
rsconnect 0.8.24 Publishing
RSQLite 2.1.0 RSQLite
rstan 2.15.1 Stan
rstudioapi 0.11 Publishing, Shiny Add-Ins, ODBC Connections, RStudio Tutorials
shiny 1.2.0 Shiny
shinytest 1.3.1 Shiny Tests
sourcetools 9.1.5 Shiny
stringi 0.3.0 Plumber R APIs, R Markdown
stringr 1.2.0 R Markdown, Visual Profiling
testthat 2.0.0 R Unit Testing
tinytex 0.16 R Markdown, TinyTeX
xfun 0.15 R Markdown
xml2 1.2.0 XML Import
xtable 1.7 Shiny

Name	Version Features
yaml	2.1.19 Publishing, R Markdown, Visual Profiling

# **D.** Command Line Interface

# E. NAME

rstudio-server - a command-line interface for administering an RStudio server.

# F. SYNOPSIS

rstudio-server COMMAND [ARGS...]

# **G. DESCRIPTION**

rstudio-server is a utility script containing a collection of subcommands for performing administrative actions on an RStudio Workbench instance.

# H. COMMANDS

# H.1. SERVER MANAGEMENT

offline Takes the server offline. Users who attempt to access the server while it is offline will see a friendly message indicating that the server is currently offline, rather than the generic HTTP error they would see if the server is fully stopped.

**online** Brings the server back online after the **offline** command was used.

**reload** Reloads the server's configuration without starting or stopping it. Only some configuration values are affected; others cannot be reloaded in place and require a full restart. Consult the administration guide for information on specific values.

**restart** Restarts the server, similar to a stop followed by a start. Useful for applying configuration changes.

**start** Start the server, if it is currently stopped.

**status** Report the status of the server according to the service manager, for instance whether it is stopped, starting, or running

**stop** Stop the server. Also stops all R sessions running locally on the server.

# H.2. USER MANAGEMENT

add-user NAME [0|1] Adds the user NAME to the list of licensed users. This is useful for allocating license seats; note however that it does not create Linux users or other system resources. The second argument, which is optional, indicates whether to create an ordinary user (0) or administrator (1).

**list-users** Lists all the users, their license status (active or locked), and whether or not they are administrators.

**lock-user** NAME Locks a user. Locked users cannot log in and don't count against the licensing seat limit.

**set-admin NAME 0|1** Sets the user NAME to either be an administrator (as in **set-admin** johndoe 1) or an ordinary user (0).

unlock-user NAME Unlocks a previously locked user.

# H.3. LOAD BALANCING

**list-nodes** Lists all active nodes in the load balancing cluster, along with the status of each. **reset-cluster** Clears the internal load balancing state. This is used to reset the secure cookie key hash and transfer protocol used by the load balancing cluster.

**delete-node ID** Permanently deletes the node with ID from the database and stops all nodes from messaging that node.

# H.4. SESSION MANAGEMENT

- active-sessions Lists all currently active sessions. Only lists local R sessions; sessions running on other nodes or via the Job Launcher are not included.
- **force-suspend-all** Forcefully suspends all R sessions at once; doesn't give users an opportunity to save their work. For a gentler alternative, use **suspend-all**.
- force-suspend-session ID|PID Forcefully suspends one specific session. If using a session controlled by the Job Launcher, its ID should be supplied (the eight characters at the end of its URL). Otherwise, supply the ID of session's process ID (PID), which you can obtain via Sys.getpid() inside the session or by inspecting the process ID of the rsession process with ps.
- kill-all ID|PID Kills all sessions. This uses SIGKILL or the equivalent, so the session will not shut down gracefully and work can be lost. When the user next attempts to use the session, R will report it as having crashed.
- kill-session IDIPID Kills one specific session. If using a session controlled by the Job Launcher, its ID should be supplied (the eight characters at the end of its URL). Otherwise, supply the ID of session's process ID (PID), which you can obtain via Sys.getpid() inside the session or by inspecting the process ID of the rsession process with ps.

suspend-all Suspends all R sessions at once.

suspend-session ID|PID Suspends one specific session. If using a session controlled by the Job Launcher, its ID should be supplied (the eight characters at the end of its URL). Otherwise, supply the ID of session's process ID (PID), which you can obtain via Sys.getpid() inside the session or by inspecting the process ID of the rsession process with ps.

# H.5. LICENSE MANAGEMENT

The server's license can be managed through the **license-manager** command, which has an extensive set of subcommands. Most of these subcommands accept the **--output** option, which can be used to set the output format to either **text** (the default when connected to a terminal) or **json**, which is useful for scripting or machine readability.

- **license-manager activate** [--proxy=PROXY] PRODUCT-KEY Attempts to activate the server with the given product key. If successful, the server uses one "activation" of the product key and becomes fully licensed. It is not necessary to restart the server after activating it; the license will take effect immediately. Requires an Internet connection; use the --proxy option to specify a proxy URL.
- **license-manager activate-file LICENSE-FILE** Attempts to activate the server with the given license file. License files take precedence over other types of licensing.
- **license-manager activate-offline ACTIVATION-FILE** Applies an activation file from RStudio Support to the server.
- **license-manager activate-offline-request** Starts the process of requesting a offline activation and generates a file you can send to RStudio to complete the activation.
- license-manager acquire-lease-verbose Attempts to acquire a lease on a floating license, printing diagnostic information while doing so. Not useful except as a troubleshooting tool; RStudio automatically acquires and releases leases during the normal course of operation, and also acquires a temporary lease while executing the status command.
- **license-manager begin-evaluation-offline EVALUATION-FILE** Begins the evaluation period, using an evaluation file supplied by RStudio Support.
- license-manager begin-evaluation-request Starts the process of requesting an evaluation period and generates a file you can send to RStudio. This typically isn't necessary except on air-gapped machines which can't automatically connect to the Internet to start an evaluation.
- **license-manager deactivate** [--proxy=PROXY] Deactivates the current license, making it available so that it can be used on another server. If successful, removes one "activation" of the product key. Requires an Internet connection; use the --proxy option to specify a proxy URL.
- **license-manager clear-license-server** Clears a license server previously set with the **license-server** command. Used to switch from floating licensing to traditional key-based licensing.
- **license-manager deactivate-offline** Deactivates the current license and prints an offline deactivation request; the license won't be fully deactivated until you send the request to RStudio for processing.
- **license-manager extend-evaluation EVALUATION-KEY** Extends the evaluation period with an evaluation key supplied by RStudio Support. Requires an Internet connection.
- **license-manager extend-evaluation-offline EVALUATION-FILE** Extends the evaluation period with an evaluation file supplied by RStudio Support.
- **license-manager initialize** [--userspace] Initializes the licensing system. This is normally done during installation, so only needs to be performed manually when using alternative installation methods or switching to userspace licensing with the --userspace flag (see administration guide).
- license-manager license-server HOSTNAME | URL Sets the host (e.g. license.corp.com:8989) or URL (e.g. https://license.corp.com/) to use for floating licensing. Note that this has no effect unless RStudio is configured to use floating licensing; see documentation for server-license-type.
- license-manager verify Verifies that the current license is valid; prints basic information about

the license and exits. Typically **status** will give you richer information.

**license-manager status** Prints detailed information about the current status of the licensing system: whether the license is currently valid, what licensing mode is in use (trial, key, floating), etc.

# H.6. DIAGNOSTICS

run-diagnostics [--output-file=PATH] Prepares an extensive diagnostics report containing configuration files, logs, system information, and other output useful to help troubleshoot errors and problems. The report is placed in /tmp/rstudio-diagnostics by default, but you can specify any other path using the --output-file option.

**test-config** Tests the server's configuration for validity.

upload-minidump MINIDUMP Uploads a previously collected crash dump file (.dmp) to RStudio's crash reporting service. Requires an active Internet connection. Crash dump files can be found in the crash-db-path path set in your crash-handler.conf file.

**verify-installation** Performs a variety of tests to ensure that the server is installed and working correctly, including starting an R session, launching a trial job (if the Job Launcher is configured), etc.

Note that the server must be stopped in order for these tests to run, so use the **stop** command first if your server is running.

**version** Reports the current version of the server and exits.

# H.7. VS CODE

install-vs-code DIRECTORY Installs the open source VS Code server (code-server) into the given DIRECTORY. The VS Code server must be installed if you plan to use VS Code with RStudio Workbench.

install-vs-code-ext [-d DIRECTORY] [-1] [-v VERSION] Installs RStudio's VS Code extension, which enables integration between VS Code and RStudio Workbench. Pass -d with the VS Code Server's installation directory to optionally upgrade the VS Code Server if required; you may also specify -1 to install locally (only for the current user) and -v to install a specific version of the extension (advanced usage only).

# H.8. DATABASE

**encrypt-password** Encrypts a password. Doesn't take any arguments; will prompt you for the password to encrypt and then emit the encrypted password. This utility is useful to encrypt the password used to access the database.

**migrate-db** Migrates the internal database from a SQLite database to a PostgreSQL database. Useful when transitioning from a single-server to a multiple-server configuration.

# Part XVII. RStudio Configuration

# I. Overview

# I.1. RStudio Server Configuration

The following sections detail user-configurable options for RStudio Server and RStudio Workbench.

# J. rserver.conf

The following is a list of available options that can be specified in the rserver.conf configuration file, which controls behavior of the rserver process, allowing you to tune HTTP, authorization options, and other settings that broadly affect RStudio Server.

# J.0.1. verify Settings

# verify-installation

Runs verification mode to verify the current installation.

Type: bool Default: 0

# verify-user

Specifies the run-as user for additional Job Launcher verification.

Type: string

Default: <empty string>

# verify-test

Specifies the verify-installation test to run. Leave empty to run all tests.

Type: string

Default: <empty string>

# J.0.2. server Settings

# server-working-dir

The default working directory of the rserver process.

Type: string Default: /

#### server-user

The user account of the rserver process.

Type: string

Default: rstudio-server

#### server-daemonize

Indicates whether or not the rserver process should run as a daemon.

Type: bool

Default: 1 (true) if rserver was run with root privilege, otherwise 0 (false).

# server-pid-file

The path to a file where the rserver daemon's pid is written.

Type: string

Default: /var/run/rstudio-server.pid

# server-set-umask

If enabled, sets the rserver process umask to 022 on startup, which causes new files to have rw-r-r permissions.

# secure-cookie-key-file

If set, overrides the default path of the secure-cookie-key file used for encrypting cookies.

Type: string

Default: <empty string>

# server-data-dir

Path to the data directory where RStudio Server will write run-time state.

Type: string

Default: /var/run/rstudio-server

#### server-add-header

Adds a header to all responses from RStudio Server. This option can be specified multiple times to add multiple headers.

Type: string

Default: <empty string>

# server-nginx-path

The relative path from the RStudio installation directory, or absolute path where the nginx binary is located.

Type: string

Default: bin/rserver-http

# server-nginx-conf-template-path

The relative path from the RStudio installation directory, or absolute path where the nginx config file templates are located.

Type: string Default: conf

#### server-nginx-conf-path

Specifies the path to the nginx config files.

Type: string

Default: /var/lib/rstudio-server/conf

# server-nginx-ld-library-path

Specifies the LD\_LIBRARY\_PATH for the nginx executable.

Type: string

Default: <empty string>

# server-access-log

Indicates whether or not to write HTTP access logs to /var/log/rstudio-server.

Type: bool Default: 0

#### server-nginx-http-directives-path

Specifies the path to custom nginx http directives.

Type: string

Default: The first nginx.http.conf file that is found on the XDG\_CONFIG\_DIRS environment, or /etc/rstudio/nginx.http.conf if no XDG\_CONFIG\_DIRS are specified.

#### server-nginx-server-directives-path

Specifies the path to custom nginx server directives.

Type: string

 $\label{eq:default:Default:Default:Default:Default:Default:The first nginx.server.conf file that is found on the XDG_CONFIG_DIRS environment, or /etc/rstudio/nginx.server.conf if no XDG_CONFIG_DIRS are$ 

specified.

## server-nginx-site-directives-path

Specifies the path to custom nginx site directives.

Type: string

Default: The first nginx.site.conf file that is found on the XDG\_CONFIG\_DIRS environment, or /etc/rstudio/nginx.site.conf if no XDG\_CONFIG\_DIRS are specified.

#### server-health-check-enabled

Indicates whether or not to allow access to the server health check URL.

Type: bool Default: 0

#### server-license-type

Specifies whether to use remote (floating) or local (activation) licensing.

Type: string
Default: local

#### license-retry-seconds

Specifies the number of seconds to wait between floating license retries.

Type: int Default:

#### resolve-load-balancer-nodes

Indicates whether or not to resolve IP addresses associated with load balancer nodes; not compatible with SSL unless the IP address is in the CN/SAN.

# server-balancer-path

The relative path from the RStudio installation directory, or absolute path where the custom load balancing script is located.

Type: string

Default: bin/rserver-balancer

#### server-multiple-sessions

Indicates whether or not to allow multiple sessions per user.

Type: bool Default: 1

#### r-versions-multiple

Indicates whether or not to allow the use of multiple R versions.

Type: bool Default: 1

# server-project-sharing

Indicates whether or not to allow project sharing.

Type: bool Default: 1

#### server-project-sharing-root-dir

Specifies the root directory for shared projects in addition to users' own home directories.

Type: string

Default: <empty string>

# server-user-home-page

Indicates whether or not to show the user home page upon login.

#### r-versions-scan

Indicates whether or not to scan for available R versions on the system.

Type: bool Default: 1

# modules-bin-path

Specifies the path to modules sh init binary. This is necessary if you intend to load R versions via modules.

Type: string

Default: <empty string>

#### admin-enabled

Indicates whether or not to allow access to the administration dashboard.

Type: bool Default: 0

# admin-group

Limits admin dashboard access to users belonging to the specified group.

Type: string

Default: <empty string>

# admin-superuser-group

Limits admin superusers to those belonging to the specified group.

Type: string

Default: <empty string>

# admin-monitor-log-use-server-time-zone

Indicates whether or not to use the server time zone when displaying the monitor log. If disabled, uses UTC.

#### r-versions-path

Specifies the path to the file containing the list of available R Versions in JSON format. This file will be automatically generated by the reserver process after discovering the R versions available on the system. It is strongly recommended not to modify this setting in most cases.

Type: string

Default: /var/lib/rstudio-server/r-versions

#### launcher-address

Specifies the address of the Launcher service (local unix domain socket file or IP address).

Type: string

Default: <empty string>

#### launcher-port

Specifies the port of the Launcher to connect to (if not using a unix domain socket).

Type: string

Default: <empty string>

#### launcher-use-ssl

Indicates whether or not to use SSL connections when connecting to the Launcher (if not using a unix domain socket).

Type: bool Default: 0

#### launcher-verify-ssl-certs

Indicates whether or not to verify the Launcher certificate(s) when using an SSL connection.

#### launcher-sessions-enabled

Indicates whether or not to use the Launcher for creating sessions.

Type: bool Default: 0

# launcher-default-cluster

Specifies the default cluster to launch jobs on when using the Launcher.

Type: string

Default: <empty string>

#### launcher-sessions-callback-address

The callback address (hostname, IP address, or HTTP URL) of rserver for Launcher sessions to communicate back.

Type: string

Default: <empty string>

# launcher-sessions-callback-verify-ssl-certs

Indicates whether or not to enforce SSL certificate verification of the server when Launcher sessions communicate back via the callback address.

Type: bool Default: 1

#### launcher-sessions-callback-timeout

The number of seconds to wait before timing out a connection from a Launcher session to the callback address.

Type: int Default: 10 launcher-sessions-container-image

Specifies the default container image to use for Launcher sessions. Only applicable for container-

based job systems (e.g. Kubernetes).

Type: strng Default:

launcher-sessions-container-run-as-root

Indicates whether or not to run the Launcher session containers as root. If not, uses the

requesting user's UID. Only applicable for container-based job systems.

Type: bool Default: 0

launcher-sessions-create-container-user

Indicates whether or not to create a user for the container's owner when running Launcher

session containers. Only applicable for container-based job systems.

Type: bool Default: 1

launcher-sessions-connection-timeout-seconds

Specifies the connection timeout in seconds to use when establishing a connection to a Launcher

session.

Type: int Default: 3

launcher-sessions-clusters

Specifies a comma-separated list of available clusters for launching interactive sessions (or all

Launcher clusters if empty).

Type: string

Default: <empty string>

353

#### launcher-adhoc-clusters

Specifies a comma-separated list of available clusters for launching adhoc jobs (or all Launcher clusters if empty).

Type: string

Default: <empty string>

# launcher-sessions-container-images

Specifies a comma-separated list of available container images for launching interactive sessions (or all cluster images if empty). Only applicable for container-based job systems.

Type: string

Default: <empty string>

#### launcher-adhoc-container-images

Specifies a comma-separated list of available container images for launching ad-hoc jobs (or all cluster images if empty). Only applicable for container-based job systems.

Type: string

Default: <empty string>

#### launcher-sessions-forward-container-environment

Indicates whether or not to forward the existing container environment variables to the session. Only applicable for container-based job systems.

Type: bool Default: 1

# launcher-sessions-container-forward-groups

Indicates whether or not to forward the user's supplemental groups to the container. Only applicable for container-based job systems.

# J.0.3. www Settings

#### www-address

The network address that RStudio Server will listen on for incoming connections.

Type: string
Default: 0.0.0.0

## www-port

The port that RStudio Server will bind to while listening for incoming connections. If left empty, the port will be automatically determined based on your SSL settings (443 for SSL, 80 for no SSL).

Type: string

Default: <empty string>

#### www-root-path

The path prefix added by a proxy to the incoming RStudio URL. This setting is used so RStudio Server knows what path it is being served from. If running RStudio Server behind a path-modifying proxy, this should be changed to match the base RStudio Server URL.

Type: string

Default: Assume the root path '/' if not defined.

#### www-thread-pool-size

The size of the threadpool from which requests will be serviced. This may be increased to enable more concurrency, but should only be done if the underlying hardware has more than 2 cores. It is recommended to use a value that is <= to the number of hardware cores, or <= to two times the number of hardware cores if the hardware utilizes hyperthreading.

Type: int Default: 2

#### www-proxy-localhost

Indicates whether or not to proxy requests to localhost ports over the main server port. This should generally be enabled, and is used to proxy HTTP traffic within a session that belongs to code running within the session (e.g. Shiny or Plumber APIs)

Type: bool Default: 1

## www-verify-user-agent

Indicates whether or not to verify connecting browser user agents to ensure they are compatible with RStudio Server.

Type: bool Default: 1

#### www-same-site

The value of the 'SameSite' attribute on the cookies issued by RStudio Server. Accepted values are 'none' or 'lax'. The value 'none' should be used only when RStudio is hosted into an iFrame. For compatibility with some browsers (i.e. Safari 12), duplicate cookies will be issued by RStudio Server when 'none' is used.

Type: string

Default: <empty string>

# www-frame-origin

Specifies the allowed origin for the iFrame hosting RStudio if iFrame embedding is enabled.

Type: string
Default: none

#### www-enable-origin-check

If enabled, cause RStudio to enforce that incoming request origins are from the host domain. This can be added for additional security. See https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\_Request\_Forgery\_Prevention\_Cheat\_Sheet.html#verifying-origin-with-standard-headers

# www-allow-origin

Specifies an additional origin that requests are allowed from, even if it does not match the host domain. Used if origin checking is enabled. May be specified multiple times for multiple origins.

Type: string

Default: <empty string>

#### ssl-enabled

Enables or disables SSL.

Type: bool Default: 0

#### ssl-certificate

Specifies the path to the SSL certificate for RStudio Server to use.

Type: string

Default: <empty string>

# ssl-certificate-key

Specifies the path to the SSL certificate private key.

Type: string

Default: <empty string>

# ssl-protocols

Specifies the list of supported SSL protocols separated by a space.

Type: string

Default: TLSv1 TLSv1.1 TLSv1.2

# ssl-redirect-http

Indicates whether or not HTTP requests should automatically be redirected to HTTPS.

#### ssl-hsts

Indicates whether or not to enable Strict Transport Security when SSL is in use.

Type: bool Default: 1

# ssl-hsts-max-age

Specifies the maximum age for Strict Transport Security.

Type: int

Default: 86400

#### ssl-hsts-include-subdomains

Indicates whether or not to include subdomains in HSTS protection.

Type: bool Default: 0

# J.0.4. rsession Settings

#### rsession-which-r

The path to the main R program (e.g. /usr/bin/R). This should be set if no versions are specified in /etc/rstudio/r-versions and the default R installation is not available on the system path.

Type: string

Default: <empty string>

# rsession-path

The relative path from the RStudio installation directory, or absolute path to the rsession executable.

Type: string

Default: rsession

#### rldpath-path

The path to the r-ldpath script which specifies extra library paths for R versions.

Type: string

Default: r-ldpath

# rsession-Id-library-path

Specifies additional LD\_LIBRARY\_PATHs to use for R sessions.

Type: string

Default: <empty string>

# rsession-config-file

If set, overrides the path to the /etc/rstudio/rsession.conf configuration file. The specified path may be a relative path from the RStudio installation directory, or an absolute path.

Type: string

Default: <empty string>

# rsession-proxy-max-wait-secs

The maximum time to wait in seconds for a successful response when proxying requests to rsession.

Type: int Default: 10

# rsession-exec-command

Specifies the wrapper command used when executing the rsession binary.

Type: string

Default: <empty string>

#### rsession-no-profile

Indicates whether or not to disable user profiles from executing on session start.

## rsession-diagnostics-enabled

Indicates whether or not session diagnostic data should be collected. This can be used for troubleshooting issues with session starts.

Type: bool Default: 0

#### rsession-diagnostics-dir

Specifies the directory where session diagnostic data should be written.

Type: string
Default: /tmp

# rsession-diagnostics-strace-enabled

Indicates whether or not strace data should be included when collecting session diagnostic data.

Type: bool Default: 0

# rsession-diagnostics-libsegfault

Specifies the path to libSegFault.so library which is used for dumping stack trace information when session diagnostics are collected.

Type: string

Default: <empty string>

# J.0.5. database Settings

#### database-config-file

If set, overrides the path to the /etc/rstudio/database.conf configuration file.

Type: string

Default: <empty string>

## J.0.6. auth Settings

#### auth-none

If set, disables multi-user authentication. Workbench/Pro features may not work in this mode.

Type: bool

(false).

### auth-validate-users

Indicates whether or not to validate that authenticated users exist on the target system. Disabling this option may cause issues to start or to run a session.

Type: bool

Default: 1 (true) if rserver was run with root privilege, otherwise 0 (false).

## auth-stay-signed-in-days

The number of days to keep a user signed in when using the "Stay Signed In" option. Will only take affect when auth-timeout-minutes is 0 (disabled).

Type: int Default: 30

### auth-timeout-minutes

The number of minutes a user will stay logged in while idle before required to sign in again. Set this to 0 (disabled) to enable legacy timeout auth-stay-signed-in-days.

Type: int Default: 60

### auth-encrypt-password

Indicates whether or not to encrypt the password sent from the login form. For security purposes, we strongly recommend you leave this enabled.

### auth-login-page-html

The path to a file containing additional HTML customization for the login page.

Type: string

Default: /etc/rstudio/login.html

## auth-rdp-login-page-html

The path to a file containing additional HTML customization for the login page, as seen by RDP users.

Type: string

Default: /etc/rstudio/rdplogin.html

### auth-required-user-group

Specifies a group that users must be in to be able to use RStudio.

Type: string

Default: <empty string>

#### auth-minimum-user-id

Specifies a minimum user id value. Users with a uid lower than this value may not use RStudio.

Type: string Default: auto

## auth-pam-require-password-prompt

Indicates whether or not to require the "Password:" prompt before sending the password via PAM. In most cases, this should be enabled. If using a custom PAM password prompt, you may need to disable this setting if PAM logins do not work correctly.

### auth-sign-in-throttle-seconds

The minimum amount of time a user must wait before attempting to sign in again after signing out.

Type: int Default: 5

#### auth-revocation-list-dir

If set, overrides the path to the directory which contains the revocation list to be used for storing expired tokens. As of RStudio Server 1.4, this has been moved to database storage, and so this setting is deprecated, but will be used to port over any existing file-based expired tokens.

This option is deprecated and should not be used.

Type: string

Default: <empty string>

### auth-cookies-force-secure

Indicates whether or not auth cookies should be forcefully marked as secure. This should be enabled if running an SSL terminator infront of RStudio Server. Otherwise, cookies will be marked secure if SSL is configured.

Type: bool Default: 0

### auth-stay-signed-in

Indicates whether or not to allow users to stay signed in across browser sessions.

Type: bool Default: 1

### auth-google-accounts

Enables/disables authentication via Google accounts.

## auth-google-accounts-redirect-base-uri

Specifies an override URI to use instead of the redirect URI detected for Google accounts. This is needed if running behind a proxy without the X-RStudio-Request header.

Type: string

Default: <empty string>

## auth-openid

Enables/disables authentication via OpenID SSO.

Type: bool Default: 0

### auth-openid-base-uri

Overrides the detected base URI for the server. This is needed if running behind a proxy without the X-RStudio-Request header.

Type: string

Default: <empty string>

## auth-openid-issuer

Specifies the HTTPS URL of the OpenID issuer and the location of '/.well-known/open-configuration'

Type: string

Default: <empty string>

### auth-openid-scopes

Specifies any additional space-separated scopes required by the OpenID OP to return a username claim.

Type: string

## auth-openid-username-claim

Specifies the name of the OpenID claim used to define the username.

Type: string

Default: preferred\_username

### auth-saml

Enables/disables authentication via SAML SSO.

Type: bool Default: 0

## auth-saml-metadata-path

Specifies the path to the XML SAML metadata file. Overrides the metadata URL option if present.

Type: string

Default: <empty string>

#### auth-saml-metadata-url

Specifies the location of the XML SAML metadata on the Identity Provider. Requires backend connectivity.

Type: string

Default: <empty string>

## auth-saml-idp-entity-id

Specifies the entity identifier (name or URI) of the Identity Provider. Only used if no metadata is defined.

Type: string

### auth-saml-idp-sso-url

Specifies the endpoint that will receive SSO requests on the Identity Provider. Only used if no metadata is defined.

Type: string

Default: <empty string>

### auth-saml-idp-sign-cert-path

Specifies the path to the PEM certificate file for verifying SAML signatures. Only used if no metadata is defined.

Type: string

Default: <empty string>

### auth-saml-idp-post-binding

When enabled, uses HTTP POST for SSO. Otherwise, uses an HTTP redirect. This must match the metadata specification if metadata is defined.

Type: bool Default: 0

### auth-saml-sso-initiation

Indicates if only "idp" or "sp" can initiate a SAML SSO sequence. If not defined, both can initiate.

Type: string

Default: <empty string>

## auth-saml-sp-base-uri

Overrides the detected base URI for the server. This is needed if running behind a proxy without the X-RStudio-Request header.

Type: string

## auth-saml-sp-encryption-key-path

Specifies the path to the PEM file containing the private key for decrypting SAML responses. Also used for request signing if a signing method is defined.

Type: string

Default: <empty string>

### auth-saml-sp-encryption-cert-path

Specifies the path to the PEM certificate file for encrypting SAML responses. Also used for request signing if a signing method is defined.

Type: string

Default: <empty string>

### auth-saml-sp-signing-key-path

Specifies the path to the PEM file containing the private key for signing SAML requests. Not used if an encryption key is defined.

Type: string

Default: <empty string>

### auth-saml-sp-signing-cert-path

Specifies the path to the PEM certificate file for verifying SAML requests signature. Not used if an encryption certificate is defined.

Type: string

Default: <empty string>

## auth-saml-sp-request-signing-method

Indicates whether "sha1", "sha256", or "sha512" is used to sign SAML requests. If not defined, the SAML requests will not be signed.

Type: string

### auth-saml-sp-name-id-format

Requests that the NameID Format be one of "unspecified", "emailAddress", "persistent" or "transient". This must match the metadata specification if metadata is defined.

Type: string

Default: <empty string>

### auth-saml-sp-attribute-username

Specifies the name of the attribute in the SAML assertion used to define the username.

Type: string

Default: Username

### auth-proxy

Enables/disables authentication via proxy by using a special header field.

Type: bool Default: 0

## auth-proxy-sign-in-url

Specifies the URL of the sign in page for proxied authentication.

Type: string

Default: <empty string>

## auth-proxy-sign-out-url

Specifies the optional URL of the sign out page for proxied authentication.

Type: string

Default: <empty string>

## auth-proxy-sign-in-delay

Specifies the delay in seconds to show user sign in info when redirecting from the proxy sign in page.

Type: int Default: 0

### auth-proxy-user-header

Specifies the name of the HTTP header that RStudio should read the proxied user identity from.

Type: string

Default: X-RStudio-Username

### auth-proxy-user-header-rewrite

Specifies the re-write rule for the auth-proxy-user-header. The format of a re-write rule is a regular expression followed by a space and then a replacement string. The replacement string can reference captured parts of the regular expression using \$1, \$2, etc.

Type: string

Default: <empty string>

### auth-pam-sessions-enabled

Enables or disables PAM sessions when new sessions are started.

Type: bool

Default: Disabled if using Launcher sessions. Enabled otherwise.

### auth-pam-sessions-profile

Specifies the profile to use for PAM sessions.

Type: string Default: su

### auth-pam-sessions-use-password

Indicates whether or not to use passwords when creating PAM sessions. Requires storing of user passwords in memory, though we use industry best-practices for keeping the passwords secure.

## auth-pam-sessions-close

Indicates whether or not to close the PAM session when the R session exits.

Type: bool Default: 0

## J.0.7. monitor Settings

### monitor-interval-seconds

The interval in seconds at which the monitor is probed for new data.

Type: int Default: 60

### monitor-stderr-enabled

Indicates whether or not to log metrics to stderr.

Type: bool Default: 0

### monitor-rrd-enabled

Indicates whether or not to enable logging of metrics to RRD.

Type: bool Default: 1

## monitor-data-path

Specifies the path where monitor logs and RRD databases should be written.

Type: string

Default: /var/lib/rstudio-server/monitor

### monitor-rstudio-session-metrics

Indicates whether or not to collect metrics about session utilization per user.

Type: bool Default: 1

## monitor-rrd-rrdtool-binary

Specifies the path to the rrdtool binary.

Type: string

Default: /usr/bin/rrdtool

## monitor-graphite-enabled

Enables/disables logging of metrics to graphite.

Type: bool Default: 0

## monitor-graphite-host

Specifies the host to send graphite metrics to.

Type: string

 ${\bf Default:~127.0.0.1}$ 

## monitor-graphite-port

Specifies the port to send graphite metrics to.

Type: int Default: 2003

## monitor-graphite-client-id

Specifies the optional client id to include along with graphite metrics.

Type: string

## audit-data-path

Specifies the path to where audit data should be stored.

Type: string

Default: /var/lib/rstudio-server/audit

### audit-r-console

Specifies the level of console activity that should be audited (none, input, or all).

Type: string
Default: none

### audit-r-console-user-limit-months

Specifies the number of months of user console data to retain within the audit directory.

Type: int Default: 0

#### audit-r-console-user-limit-mb

Specifies the limit in megabytes on user console actions to retain in the audit log.

Type: int Default: 50

### audit-r-console-compress

Indicates whether or not to compress console audit logs using gzip compression.

Type: bool Default: 0

### audit-r-console-format

Specifies the format to use for the console audit log (csv or json).

Type: string Default: csv

### audit-r-sessions

Indicates whether or not to audit R session activity.

Type: bool

Default: Enabled if using named user licensing. Disabled otherwise.

## audit-r-sessions-limit-months

Specifies the number of months of session action data to retain within the audit directory.

Type: int Default: 13

### audit-r-sessions-limit-mb

Specifies the limit in megabytes on session actions to retain in the audit log.

Type: int Default: 1024

### audit-r-sessions-format

Specifies the format to use for the session audit log (csv or json).

Type: string Default: csv

## server-shared-storage-path

Specifies the path to the shared storage directory.

Type: string

Default: /var/lib/rstudio-server/shared-storage

# K. rsession.conf

The following is a list of available options that can be specified in the rsession.conf configuration file, which controls behavior of the rsession process, allowing you to tune various R session parameters.

## K.0.1. verify Settings

## verify-installation

Verifies that the session installation is working correctly and exits.

Type: bool Default: 0

## K.0.2. version Settings

### version

Prints the version number and exits.

Type: bool Default: 0

## K.0.3. docs Settings

### docs-url

If specified, overrides the URL to navigate to when a user clicks on the RStudio help link.

Type: string

## K.0.4. session Settings

### session-timeout-minutes

The amount of minutes before a session times out, at which point the session will either suspend or exit.

Type: int Default: 120

### session-timeout-suspend

Indicates whether or not to suspend the session after the timeout has elapsed. Setting this to false will cause the session to quit instead of suspending to disk when the session times out.

Type: bool Default: 1

### session-disconnected-timeout-minutes

If set, causes the session to time out after not receiving any new connections within the specified minutes. This behavior is generally not needed, and you should instead use session-timeout-minutes

Type: int Default: 0

## session-preflight-script

Sets a script to be run on startup before any R initialization has occurred.

Type: string

Default: <empty string>

### session-create-public-folder

Indicates whether or not to create a Public folder for the user whenever the session starts. This folder will have global read permissions, and can be used as a simple means for users to share files. It is recommended you do not use this feature, and instead use the more advanced Project Sharing features.

### session-create-profile

Indicates whether or not to create a default empty .Rprofile script within the user's home directory, which can be altered to perform any desired common startup tasks.

Type: bool Default: 0

### session-rprofile-on-resume-default

Specifies the default user setting for running the Rprofile when sessions are resumed.

Type: bool Default: 0

#### session-save-action-default

Specifies the default save action (yes, no, or ask).

Type: string

Default: <empty string>

## session-default-working-dir

Specifies the default working directory to use for new sessions.

Type: string

Default: <empty string>

## session-default-new-project-dir

Specifies the default directory to use for new projects.

This option is deprecated and should not be used.

Type: string
Default: ~

## show-help-home

Indicates whether or not to show the help home page on startup.

session-default-console-term

Specifies the default TERM setting for the R console.

Type: string

Default: xterm-256color

session-default-clicolor-force

Specifies the default CLICOLOR\_FORCE setting for the R console.

Type: bool

Default: 1

session-quit-child-processes-on-exit

Indicates whether or not to quit child processes of the session on exit. If unset, child processes created by forking or parallel processing may continue to run in the background after the

session is terminated.

Type: bool Default: 0

session-first-project-template-path

Specifies the path to a first project template which will be copied into new users' home directories and opened the first time they run a session. The template can optionally be configured with <code>DefaultOpenDocs</code> to cause documents to automatically be opened for the first

project.

Type: string

Default: <empty string>

default-rsconnect-server

Specifies the default RStudio Connect server URL.

Type: string

Default: <empty string>

377

## websocket-ping-seconds

Specifies the WebSocket keep-alive ping interval for session terminals.

Type: int Default: 10

## websocket-connect-timeout

Specifies the WebSocket initial connection timeout in seconds for session terminals.

Type: int Default: 3

## websocket-log-level

Specifies the WebSocket log level for session terminals ((0=none, 1=errors, 2=activity, 3=all).

Type: int Default: 0

#### websocket-handshake-timeout

Specifies the WebSocket protocol handshake timeout for session terminals in milliseconds.

Type: int Default: 5000

### package-output-to-package-folder

Specifies whether or not package builds output to the package project folder.

Type: bool Default: 0

### restrict-directory-view

Indicates whether or not to restrict the directories that can be viewed within the IDE.

## directory-view-allow-list

Specifies a list of directories exempt from directory view restrictions, separated by a colon character (:).

Type: string

Default: <empty string>

### session-ephemeral-env-vars

Specifies a list of environment variables that will not be saved when sessions suspend, separated by a colon character (:).

Type: string

Default: <empty string>

### session-suspend-on-incomplete-statement

Specifies whether the session should be allowed to suspend when a user has entered a partial R statement.

Type: bool Default: 0

### session-async-rpc-enabled

Enables async responses to rpc requests to prevent connection logiams in the browser, allowing interrupt of busy sessions

Type: bool Default: 1

## session-async-rpc-timeout-ms

Duration in millis before requests are converted to async - i.e. how fast will the server free up connections when it's busy

Type: int Default: 200

### session-handle-offline-enabled

Enables offline request handling. When the R session is busy, some requests are allowed to run

Type: bool Default: 1

#### session-handle-offline-timeout-ms

Duration in millis before requests that can be handled offline are processed by the offline handler thread.

Type: int Default: 200

## K.0.5. allow Settings

### allow-vcs-executable-edit

Indicates whether or not to allow editing of VCS (Version Control Systems) executables.

Type: bool Default: 1

### allow-r-cran-repos-edit

Indicates whether or not to allow editing of CRAN repositories.

Type: bool Default: 1

## allow-vcs

Indicates whether or not to allow the use of version control features.

## allow-package-installation

Indicates whether or not to allow installation of packages from the packages pane.

Type: bool Default: 1

## allow-shell

Indicates whether or not to allow access to the shell dialog.

Type: bool Default: 1

### allow-terminal-websockets

Indicates whether or not to allow connections to terminal sessions with websockets.

Type: bool Default: 1

### allow-file-downloads

Indicates whether or not to allow file downloads from the files pane.

Type: bool Default: 1

### allow-file-uploads

Indicates whether or not to allow file uploads from the files pane.

Type: bool Default: 1

### allow-remove-public-folder

Indicates whether or not to allow removal of the user public folder.

## allow-rpubs-publish

Indicates whether or not to allow publishing of content to external services.

Type: bool Default: 1

## allow-external-publish

Indicates whether or not to allow publishing of content to external services.

Type: bool Default: 1

## allow-publish

Indicates whether or not to allow publishing of content.

Type: bool Default: 1

### allow-presentation-commands

Indicates whether or not to allow presentation commands.

Type: bool Default: 0

## allow-full-ui

Indicates whether or not to allow full standalone UI mode.

Type: bool Default: 1

### allow-launcher-jobs

Indicates whether or not to allow running jobs via the Launcher.

## K.0.6. *r* Settings

### r-core-source

Specifies the Core R source path.

Type: string Default: R

### r-modules-source

Specifies the Modules R source path.

Type: string

Default: R/modules

## r-session-package-archives

Specifies the R package archives path.

Type: string

Default: R/packages

### r-libs-user

Specifies the R user library path.

Type: string

Default: <empty string>

## r-cran-repos

Specifies the default CRAN repository.

Type: string

### r-cran-repos-file

Specifies the path to a configuration file which contains default CRAN repositories.

Type: string

Default: The first repos.conf file that is found within the XDG\_CONFIG\_DIRS environment, or /etc/rstudio/repos.conf if not XDG\_CONFIG\_DIRS are specified.

### r-cran-repos-url

Specifies the URL to a configuration file which contains optional CRAN repositories.

Type: string

Default: <empty string>

#### r-auto-reload-source

Indicates whether or not to automatically reload R source if it changes during the session.

Type: bool Default: 0

## r-compatible-graphics-engine-version

Specifies the maximum graphics engine version that this version of RStudio is compatible with.

Type: int Default: 14

### r-resources-path

Specifies the directory containing external resources.

Type: string

Default: resources

### r-doc-dir-override

Specifies the override for R\_DOC\_DIR (used for debug configurations).

Type: string

### r-restore-workspace

If set, overrides the user/project restore workspace setting. Can be 0 (No), 1 (Yes), or 2 (Default).

Type: int

Default: 2 (Default).

### r-run-rprofile

If set, overrides the user/project .Rprofile run setting. Can be 0 (No), 1 (Yes), or 2 (Default).

Type: int

Default: 2 (Default).

## K.0.7. limits Settings

## limit-file-upload-size-mb

Sets a size limit in megabytes on files that are uploaded via the files pane.

Type: int Default: 0

### limit-cpu-time-minutes

Sets a limit in minutes for the amount of time top level R computations may run before being interrupted.

Type: int Default: 0

## limit-xfs-disk-quota

Indicates whether or not XFS quotas should be enforced when performing file operations via the files pane.

## K.0.8. external Settings

## external-consoleio-path

Specifies the path to the consoleio executable (Windows-only).

Type: string

Default: bin/consoleio.exe

## external-gnudiff-path

Specifies the path to gnudiff utilities (Windows-only).

Type: string

Default: bin/gnudiff

### external-gnugrep-path

Specifies the path to gnugrep utilities (Windows-only).

Type: string

Default: bin/gnugrep

## external-msysssh-path

Specifies the path to msys\_ssh utilities (Windows-only).

Type: string

Default: bin/msys-ssh-1000-18

## external-sumatra-path

Specifies the path to SumatraPDF (Windows-only).

Type: string

Default: bin/sumatra

## external-winutils-path

Specifies the path to Hadoop Winutils (Windows-only).

Type: string

Default: bin/winutils

## external-hunspell-dictionaries-path

Specifies the path to hunspell dictionaries.

Type: string

Default: resources/dictionaries

## external-mathjax-path

Specifies the path to the mathjax library.

Type: string

Default: resources/mathjax-27

### external-pandoc-path

Specifies the path to pandoc binaries.

Type: string

Default: bin/pandoc

### external-libclang-path

Specifies the path to the libclang shared library

Type: string

Default: bin/rsclang

## external-libclang-headers-path

Specifies the path to the libclang builtin headers.

Type: string

Default: resources/libclang/builtin-headers

## external-winpty-path

Specifies the path to winpty binaries.

Type: string Default: bin

## K.0.9. git Settings

## git-commit-large-file-size

Warns when attempting to commit files larger than this size (in bytes; set 0 to disable).

Type: int

Default: 5242880

## K.0.10. user Settings

## show-user-identity

Indicates whether or not to show the user identity in the session UI.

Type: bool Default: 1

## K.0.11. misc Settings

## tutorial-api-enabled

Enables/disables the tutorial API.

Type: bool Default: 0

## tutorial-api-parent-notify-enabled

Enables/disables tutorial API parent notification.

### tutorial-api-client-origin

Specifies the tutorial API client origin.

Type: string

Default: <empty string>

### nfs-acl-version

Specifies the protocol version for the NFS Access Control List to use with Project Sharing (nfsv3 or nfsv4).

Type: string

Default: <empty string>

### nfs4-principal-type

Specifies the type of security principal to use with NFSv4 Access Control Lists ('username' or 'uid').

Type: string

Default: username

### nfs4-domain

Specifies the domain for NFSv4 Access Control Lists. Needed when using Project Sharing on an NFSv4 share with username-style security principals.

Type: string

Default: <empty string>

### project-sharing-enumerate-server-users

Indicates whether or not to enumerate the server's user groups when determining the users available for Project Sharing. If disabled, enumerates the users that are present in the RStudio user database instead.

## session-timeout-kill-hours

Specifies the amount of hours to wait before forcefully killing a running session after it has been idle.

Type: int Default: 0