Scheduling

Scheduling in databases means setting up automated tasks (called *jobs*) that run at specific times or intervals — instead of you having to run them manually.

Main Reasons for Scheduling Jobs

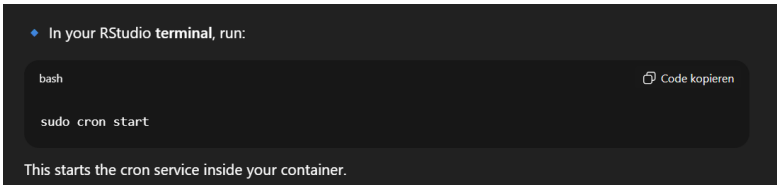
1. Resource Management
    - Goal: Prevent heavy operations from slowing down systems during busy hours.
    - Example: Run *intensive updates* or *data model training* at night or early morning when few users are online.
    - 🔹 *Analogy:* Like running big downloads at night so your internet isn't slow during the day.
2. Up-to-Date Reports and Visualizations
    - Goal: Keep dashboards, BI tools, and analytics current.
    - Example: Automatically refresh data every morning so managers always see the latest figures.
    - 🔹 *Think of it as:* Scheduling daily refreshes for Power BI or Tableau dashboards.
3. Auditing or Compliance Requirements
    - Goal: Meet mandatory reporting schedules required by regulations.
    - Example: Pharmaceutical firms may need *monthly audit reports* generated automatically.
    - 🔹 *Key point:* Automation ensures reports are always produced on time, with a traceable schedule.
4. Automated Backups
    - Goal: Protect data without human intervention.
    - Example: Schedule nightly database backups to prevent data loss.
    - 🔹 *Why it matters:* Manual backups are easy to forget — automation makes it reliable.
5. Avoid Manual Updating
    - Goal: Save time and reduce human error.
    - Example: Automate fetching new data, training models, and updating reports daily.
    - 🔹 *Benefit:* More consistent and less error-prone than manual routines.

Using the cronR Package

Purpose:

cronR lets you automate the execution of R scripts — for example, to refresh data, run analyses, or generate reports at regular intervals (daily, hourly, weekly, etc.).

1. What is cronR?
   - It's an R package that provides an easy interface to the cron scheduler (a standard job scheduler on Linux/Unix systems).
   - Instead of writing raw cron commands (which can be tricky), cronR allows you to set up, edit, and manage cron jobs directly from R or RStudio.
2. Relation to Cron
   - Cron = a system process that runs commands or scripts at scheduled times.
   - cronR = a user-friendly R wrapper that lets you communicate with that system scheduler.
3. Before Using cronR: Start the Cron Service
   - If you're working in a container (like a cloud RStudio or a Linux droplet), you need to make sure cron is running.



4. How to Use cronR
   - You have two ways:
   - Option A — *Visual Interface (Easiest)*
     - Go to the "Addins" menu in RStudio.
     - Choose "Schedule R scripts..." (from cronR).
     - Select your R script and set the time/frequency (e.g., every morning at 6:00 a.m.).
     - Save it — your job is now scheduled!
   - Option B — *Command Line (Programmatic)*
     - You can use commands in R like:

5. Resource Management Tips
   - If you're using a small droplet (1GB RAM), watch your resource icon in RStudio (top-right corner).
   - When resources get tight:
     - Clear objects from your workspace.
     - Restart RStudio to free memory.
   - This ensures your scheduled jobs won't crash due to low memory.

Create first Cron job

Part 1 – The R logic (the work you want to automate)

- To show how the cronR package works, we will use the following R script, which we will put on different schedules:

```
# Get the current datetime
date_time <- format(Sys.time(), digits = 0)
# Check if "increment_one.rds" exists
if(file.exists("/home/rstudio/scheduling/increment_one.rds")){
  # If "increment_one.rds exists then we read it into memory
  increment_one <- readRDS(file = "/home/rstudio/scheduling/increment_one.rds")
  # We add one to the R object
  increment_one <- increment_one + 1
  # The R object is saved to the disk
  saveRDS(increment_one, file = "/home/rstudio/scheduling/increment_one.rds")
  # We print the datetime and the value of increment_one.
  # This will be captured by the cronR logger and written to the .log file
  print(paste0(date_time, ": Value of increment_one.rds is ", increment_one))
}else{
  # If "increment_one.rds" does not exist we begin by 1
  increment_one <- 1
  # The R object is saved to the disk
  saveRDS(increment_one, file = "/home/rstudio/scheduling/increment_one.rds")
  # We print the datetime and the value of increment_one.
  # This will be captured by the cronR logger and written to the .log file
  print(paste0(date_time, ": Value of increment_one.rds is ", increment_one))
}
```

Part 2 – The scheduling setup (the actual cron job command)

Now, the actual cron job is defined by these lines:

cmd <- cron_rscript(rscript="/home/rstudio/scheduling/Scheduling/increment_one.rds")

```
30  cmd <- cron_rscript(rscript="/home/rstudio/scheduling/Scheduling/increment_one.rds")
31
```

- This tells cronR *what script* to run (the .R file that contains the code above).
- It creates a command that cron understands (a wrapper to run your R script non-interactively).
- We give the name of the script we wish to execute on a schedule (we include the script path if our script is not located directly in our R working directory). The “cron_rscript” function, constructs a command we can use in the “cron_add” function. The “cron_rscript” includes information of, e.g., where a log file will be stored and whether output from the R script should be stored in the log file. By default, the log file will have the same name as the R script but ending in “.log”

cron_add(cmd, frequency = 'minutely', id='job1', description='Our first cronR job')

```
31
32  cron_add(cmd, frequency = 'minutely', id='job1', description='Our first cronR job')
33
```

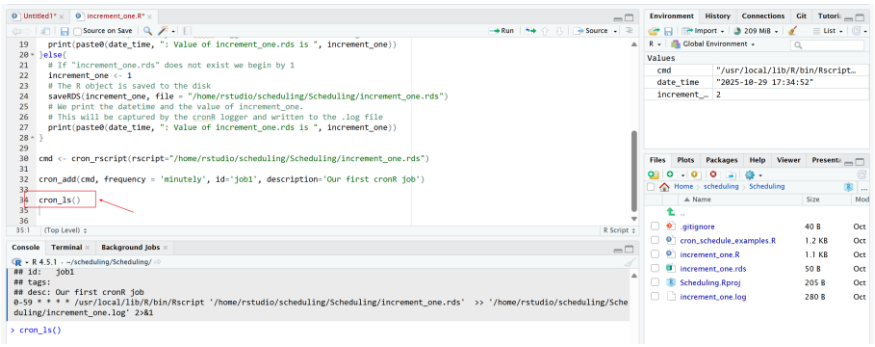
Select y

```
> cron_add(cmd, frequency = 'minutely', id='job1', description='Our first cronR job')
Are you sure you want to add the specified cron job: '/usr/local/lib/R/bin/Rscript '/home/rstudio/scheduling/Scheduling/increment_on
e.rds'  >> '/home/rstudio/scheduling/Scheduling/increment_one.log' 2>&1'? [y/n]:
1: y

Adding cronjob:
---------------

## cronR job
## id:   job1
## tags:
## desc: Our first cronR job
0-59 * * * * /usr/local/lib/R/bin/Rscript '/home/rstudio/scheduling/Scheduling/increment_one.rds'  >> '/home/rstudio/scheduling/Sche
duling/increment_one.log' 2>&1
```

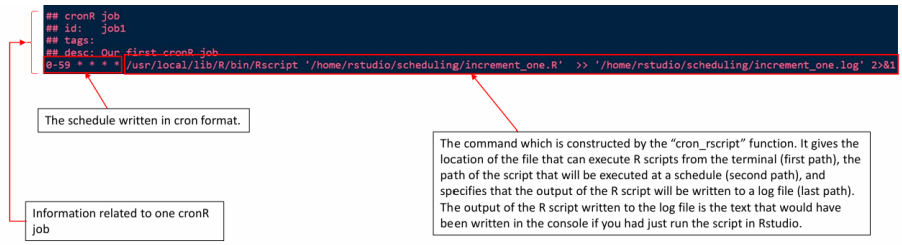
- This is the line that creates the cron job.
- frequency = 'minutely' → tells cron to run it every minute.
- id = 'job1' → gives your job a name for tracking.
- description → adds a short explanation.

Inspecting cronR jobs

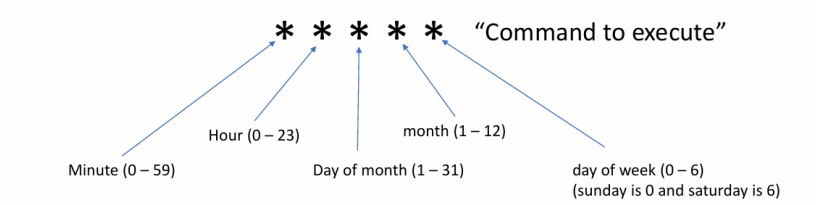
Cron ls() Calling this function, lists all cronR jobs in the console. We could have added further arguments as “id” or “tag” to only show a subset of the conR jobs.



Outcome

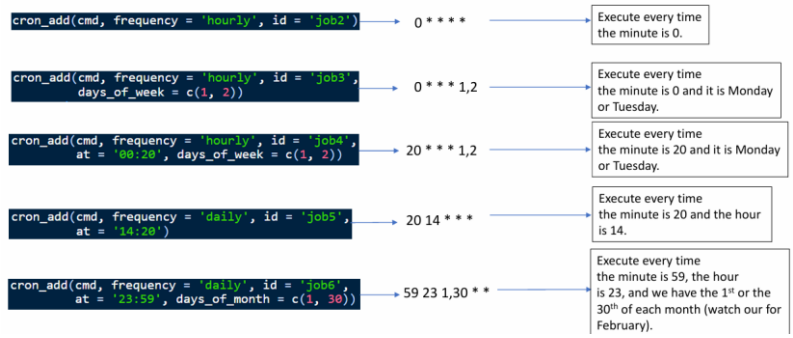


CronR schedule

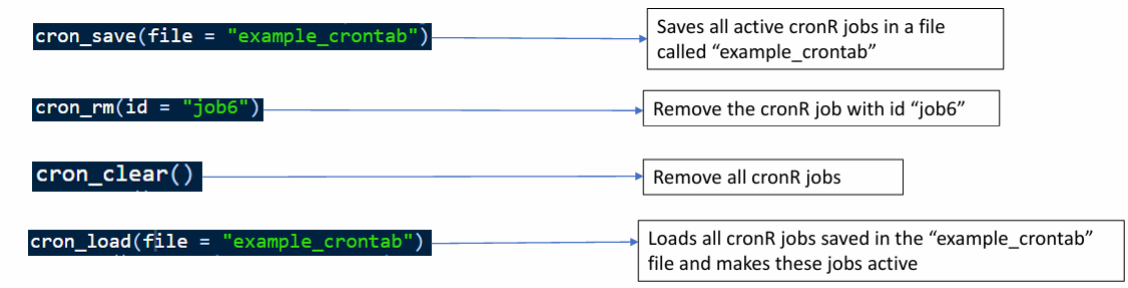


More details [Understanding Cron Syntax in the Job Scheduler - Cloud Manager Administrator Reference]

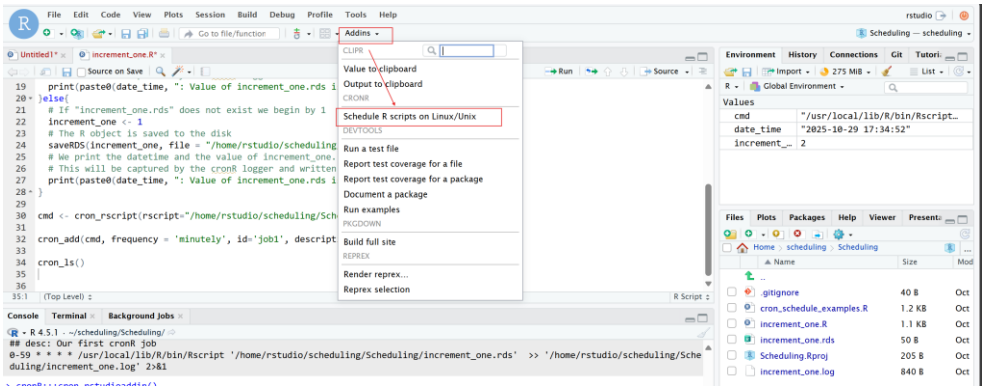
Examples

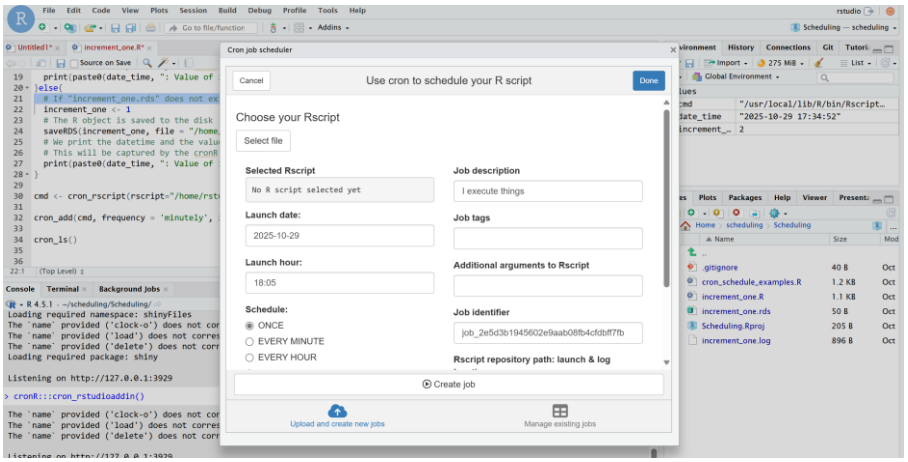


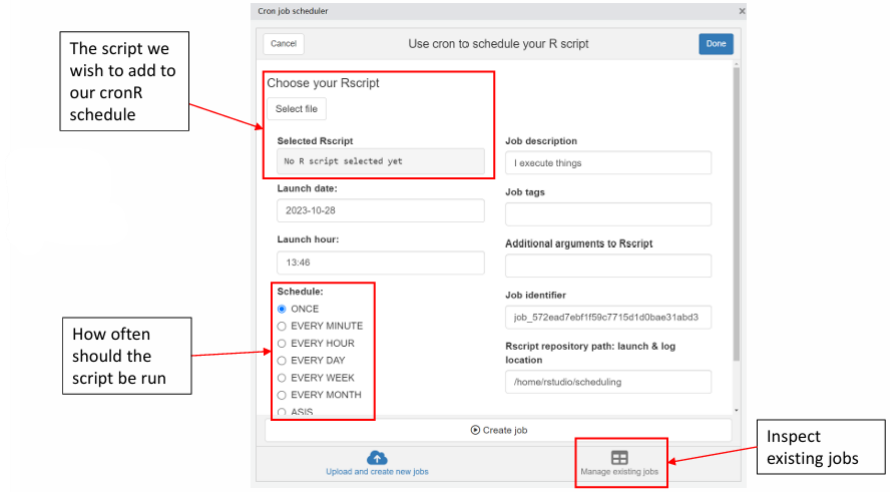
Saving/loading Crontab and removing cronR jobs



Using the visual cronR interface







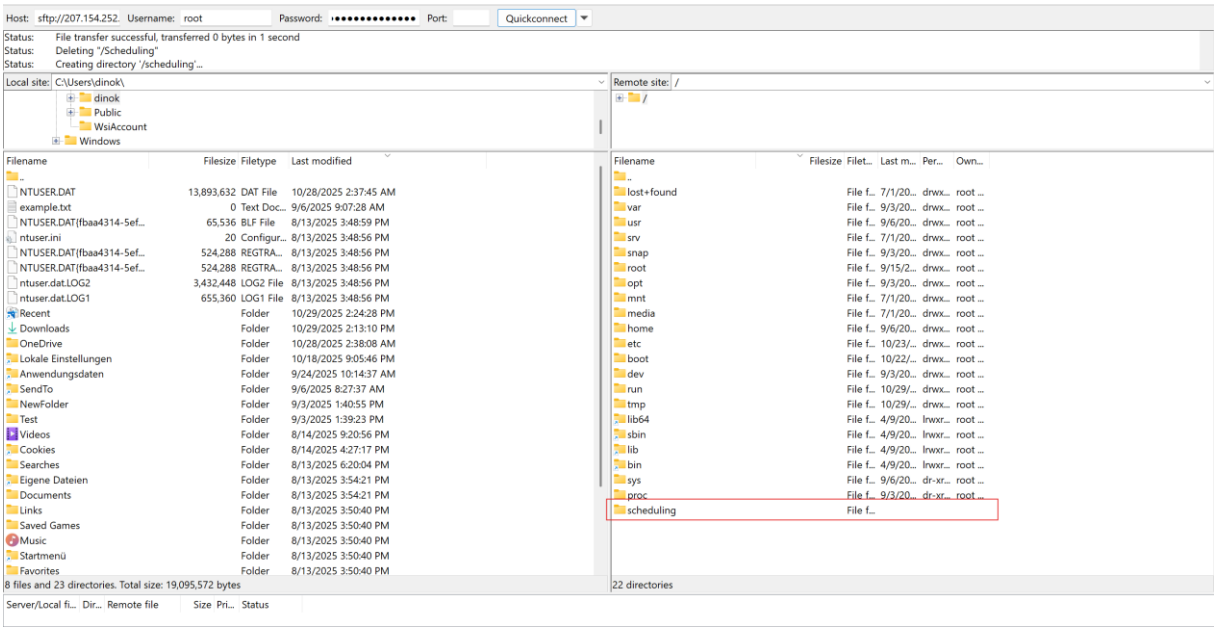
Full example to run CronR

To run CronR

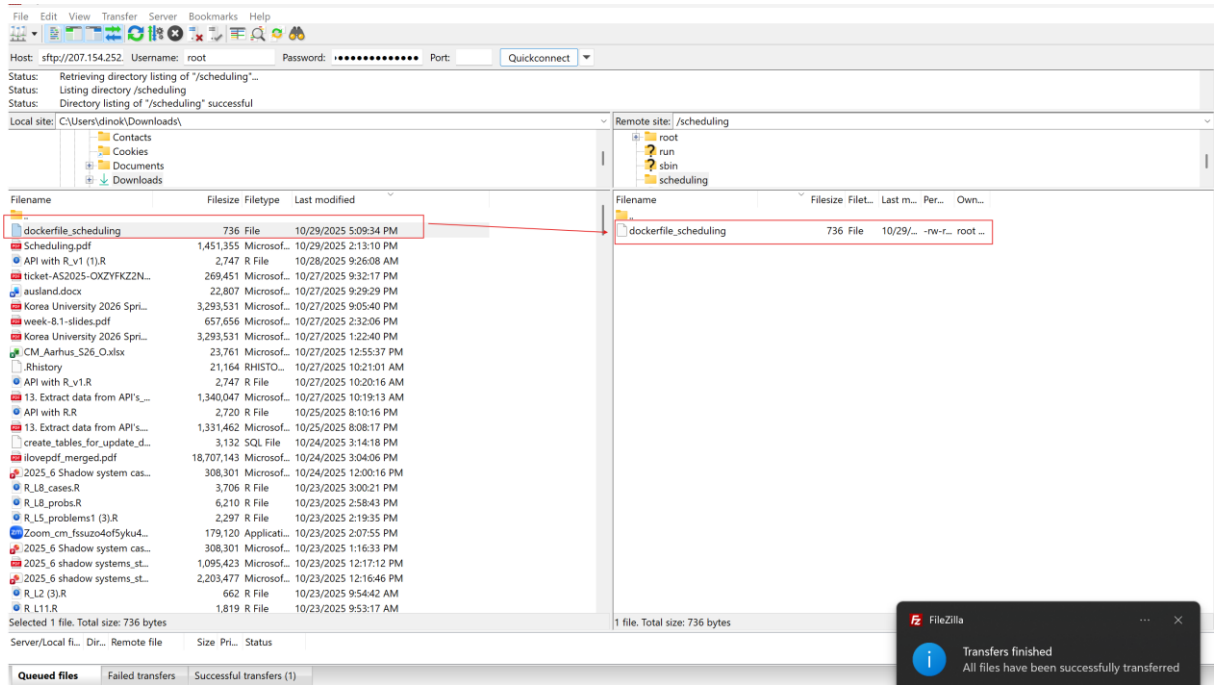
First create a new image with the CronR image

To do that:

Create new folder in Filezella where the file later will be saved



Move fille in the newly created folder (from Local to remote)



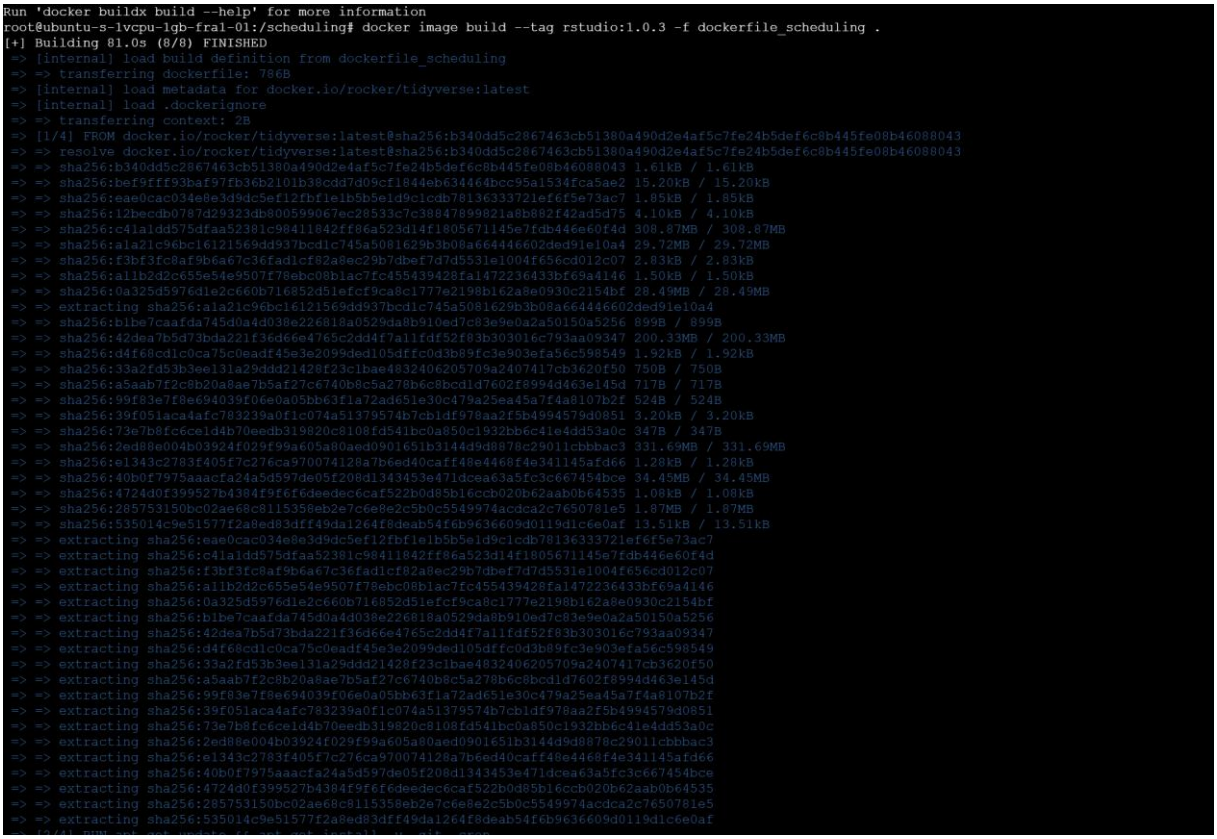
In Digital Ocean go into the folder we just created and saved the file

pwd	current working directory
cd ..	moves one level up
ls -l	information about files and directories in the current directory
cd scheduling^	in this case to get into the scheduling folder

```
root@ubuntu-s-1vcpu-1gb-fra1-01:/# cd scheduling
root@ubuntu-s-1vcpu-1gb-fra1-01:/scheduling#
```

Build the Image

docker image build --tag rstudio:1.0.3 -f dockerfile_scheduling .



build a new rstudio container using the image we just created:

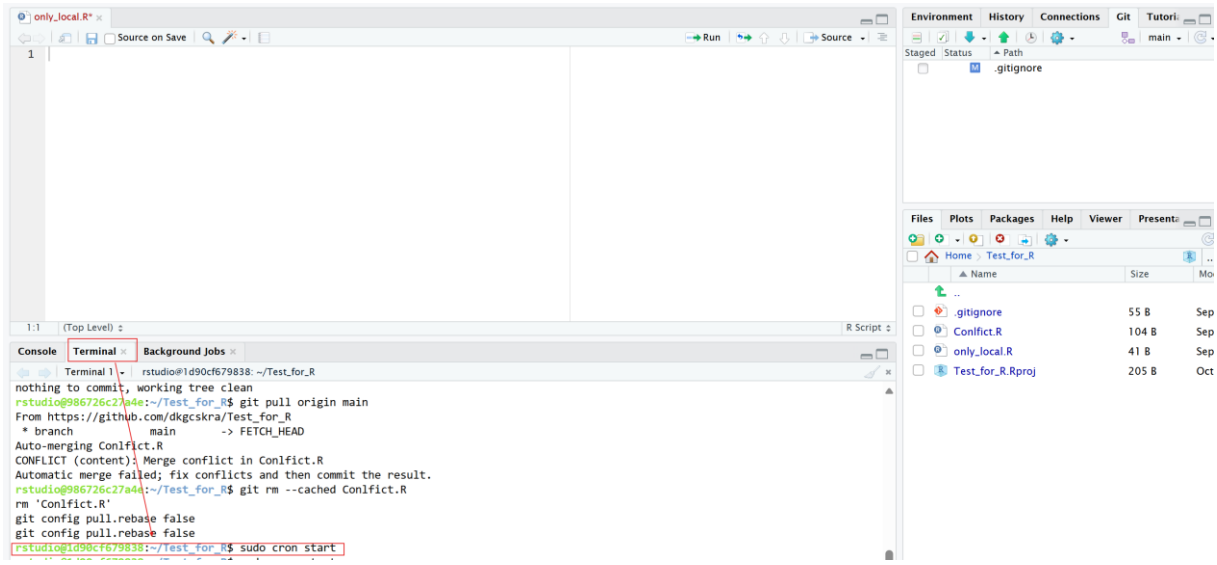
docker run -d --network db_r_shiny -p 8787:8787 -e PASSWORD=xxxxxxxxxxxx --name rstudio -v rstudio_data:/home/rstudio rstudio:1.0.3

Inspect the network

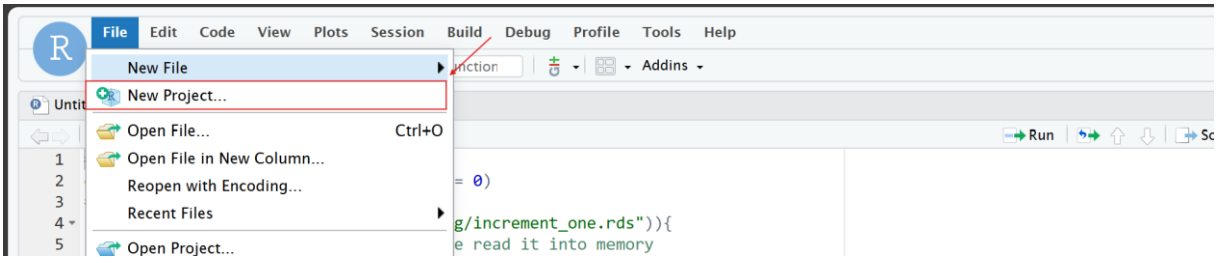
docker network inspect db_r_shiny

```
root@ubuntu-s-1vcpu-1gb-fra1-01:/scheduling# docker network inspect db_r_shiny
[
    {
        "Name": "db_r_shiny",
        "Id": "72bf3977433b7dad563e545a13ec866902d910799a362fdc3b68c12ba8192140",
        "Created": "2025-09-06T10:02:07.167438936Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "10.0.42.0/24",
                    "IPRange": "10.0.42.128/25"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "16da47902dcd8f0510d3bb5cf47c81c1ed4ce093e87581571a45071ca068559c": {
                "Name": "postgres",
                "EndpointID": "3a75a7ba7837e13ab4517f91b1baab421b0c6413ee00f87f1dfbccaf2398baa7",
                "MacAddress": "46:8e:78:a9:c3:44",
                "IPv4Address": "10.0.42.130/24",
                "IPv6Address": ""
            },
            "1d90cf679838c43fc03bccea4e240a6517af54493f3a7bd077b913ff400b0ea6": {
                "Name": "rstudio",
                "EndpointID": "3c6166f65df9a6cecd3b406b59a96d2069b36b53b432cc76e94883971ae517bc",
                "MacAddress": "fe:d5:24:49:a8:49",
                "IPv4Address": "10.0.42.129/24",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

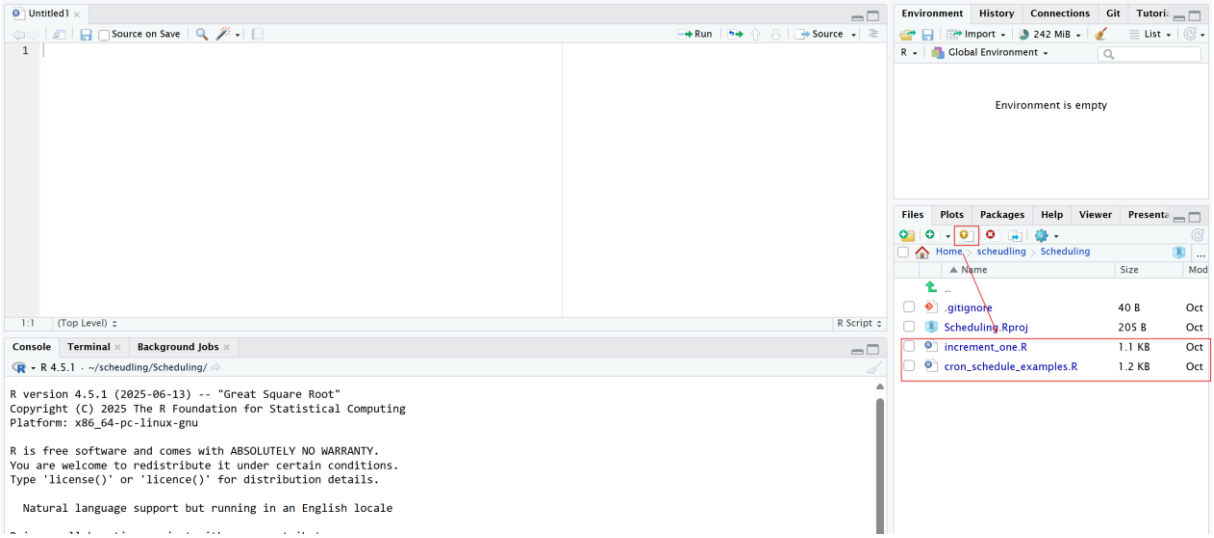
Log into R and start Crone from the terminal



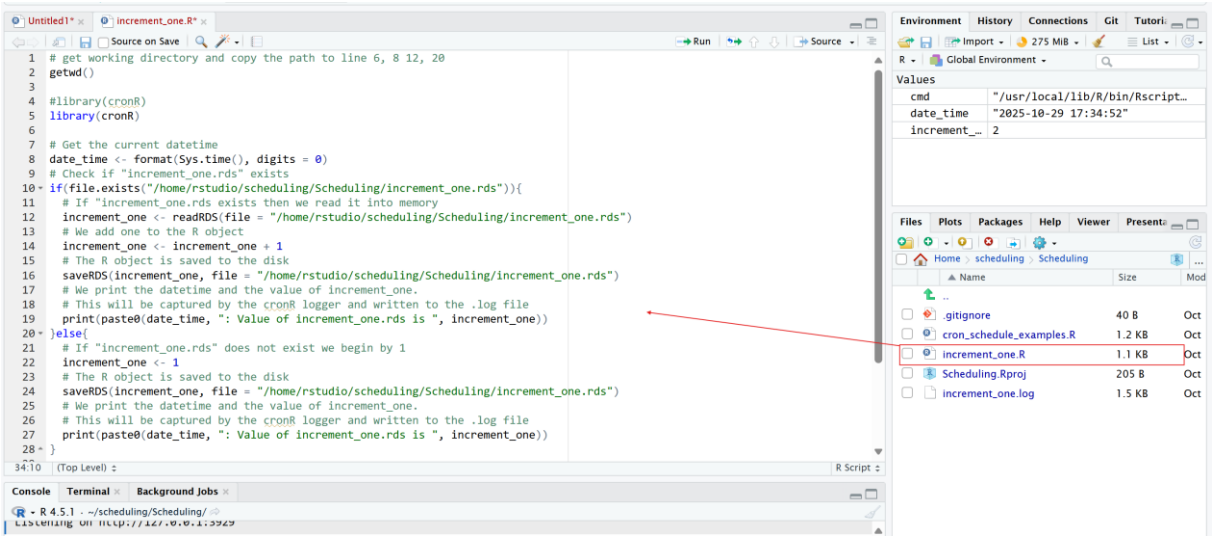
Create new R Project



Upload the scripts into R



Open the script



Define the crone job

cmd <- cron_rscript(rscript="/home/rstudio/scheduling/Scheduling/increment_one.rds")

```
30  cmd <- cron_rscript(rscript="/home/rstudio/scheduling/Scheduling/increment_one.rds")
```

Add the crone job

cron_add(cmd, frequency = 'minutely', id='job1', description='Our first cronR job', days_of_week = 3)

```
31  cron_add(cmd, frequency = 'minutely', id='job1', description='Our first cronR job', days_of_week = 3)
```

Schedule the script “increment_one.R” to execute every minute for this current weekday (Wednesday)

Add another crone job

cron_add(cmd, frequency = 'daily', id='job2', description='Our second cronR job', at= '04:00', days_of_month = 15)

```
32  cron_add(cmd, frequency = 'daily', id='job2', description='Our second cronR job', at= '04:00', days_of_month = 15)
```

In this case Schedule the script “increment_one.R” to execute at 04:00 but only on the 15th of each month.

List the crone Jobs

cron_ls()

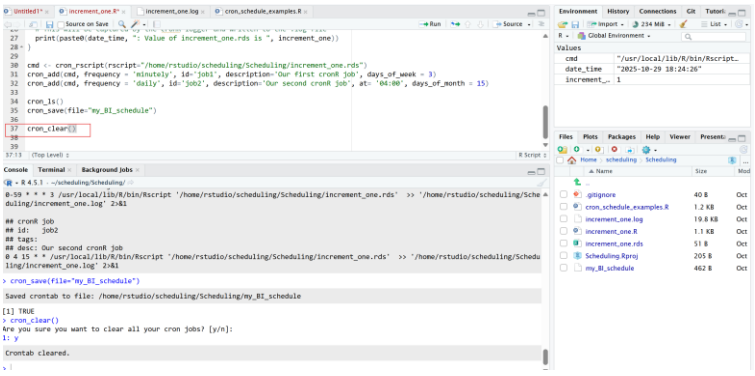
```
> cron_ls()


## cronR job
## id:   job1
## tags:
## desc: Our first cronR job
0-59 * * * 3 /usr/local/lib/R/bin/Rscript '/home/rstudio/scheduling/Scheduling/increment_one.rds'  >> '/home/rstudio/scheduling/Sche
duling/increment_one.log' 2>&1

## cronR job
## id:   job2
## tags:
## desc: Our second cronR job
0 4 15 * * /usr/local/lib/R/bin/Rscript '/home/rstudio/scheduling/Scheduling/increment_one.rds'  >> '/home/rstudio/scheduling/Schedu
ling/increment_one.log' 2>&1
```

Save cron jobs to a file

cron_save(file="my_BI_schedule")

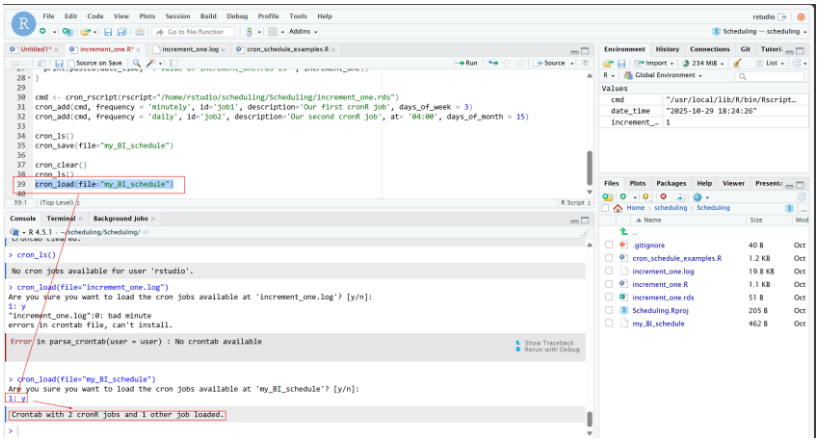
➔ my_BI_schedule is the name of the file

Clear cron jobs



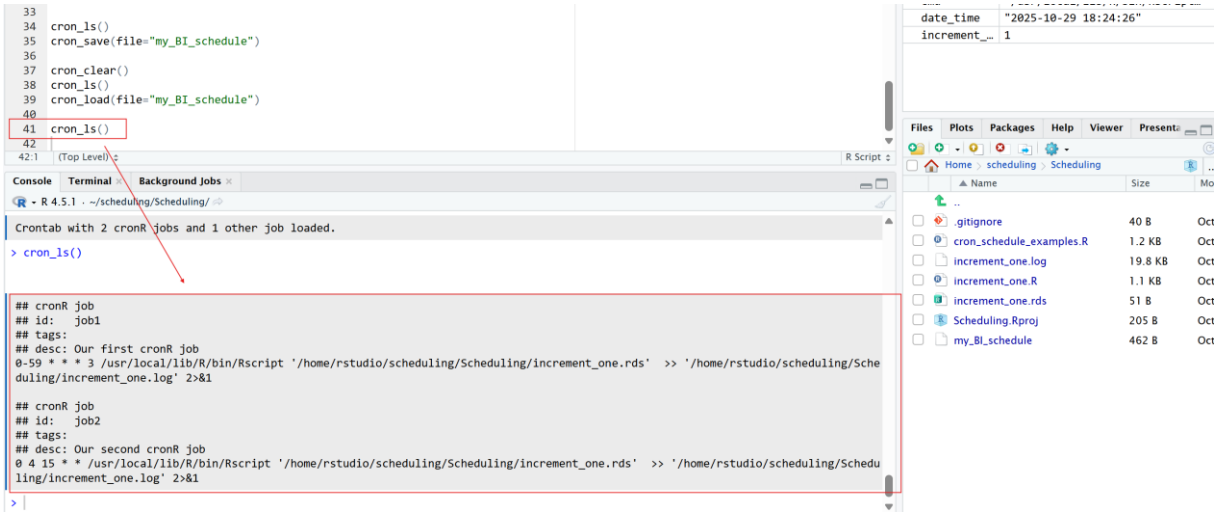
Load Cron jobs from a file

Croad_load (file= “my_BI_schedule”)

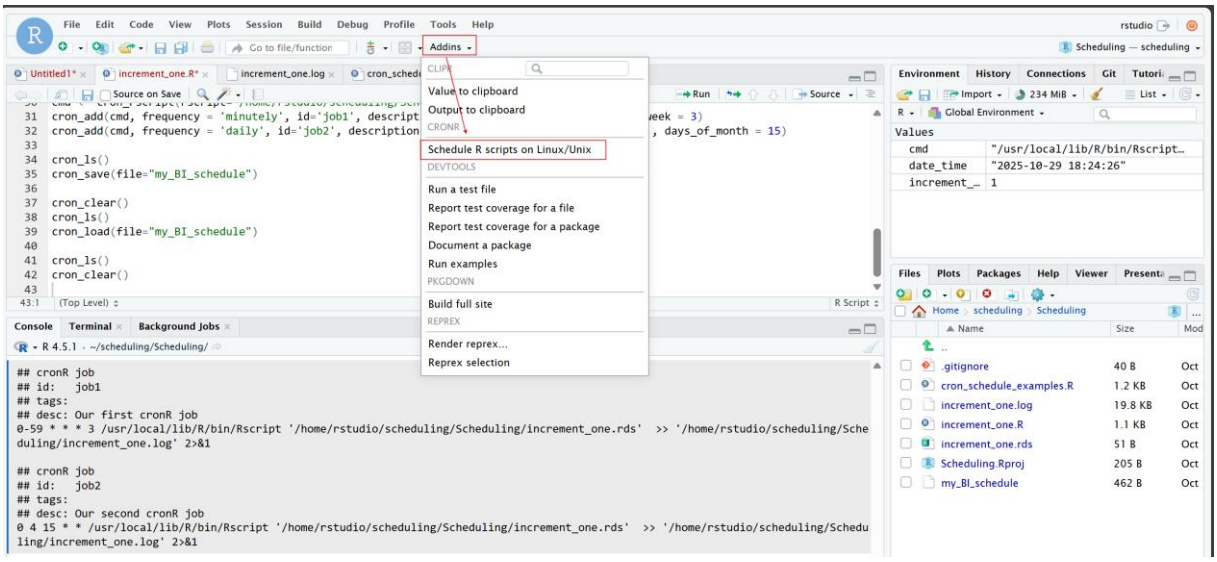


verify that they are active

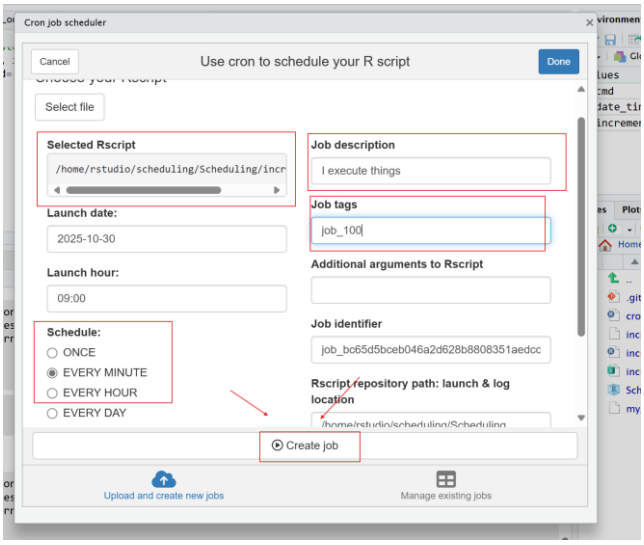
cron_ls()



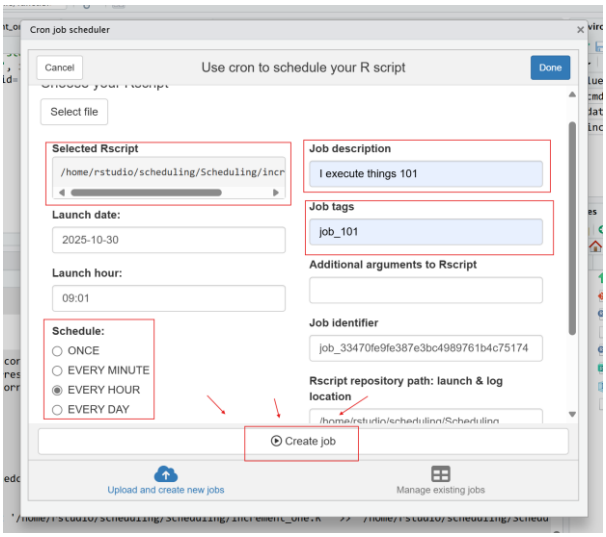
Cron Job from the cronR addin



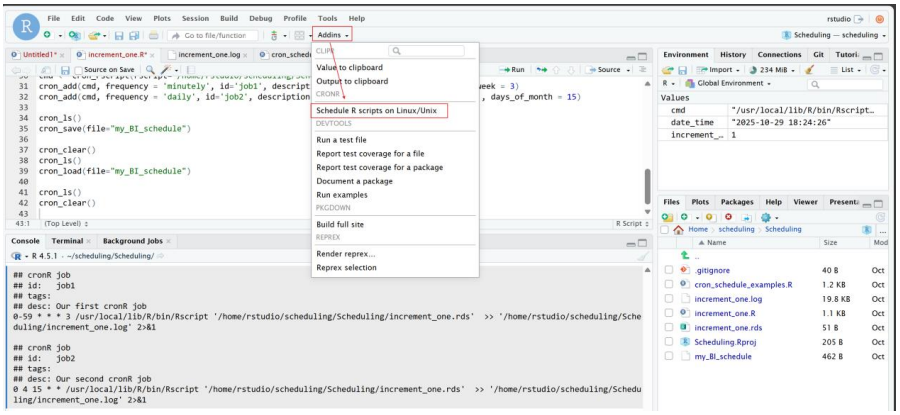
set the “increment_one.R” script to execute every minute

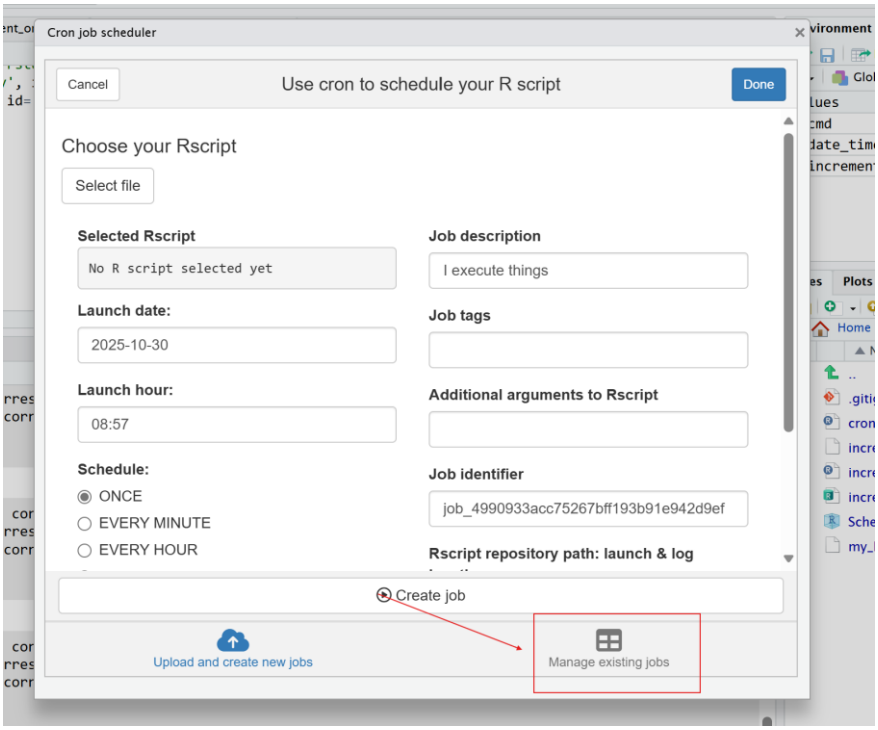


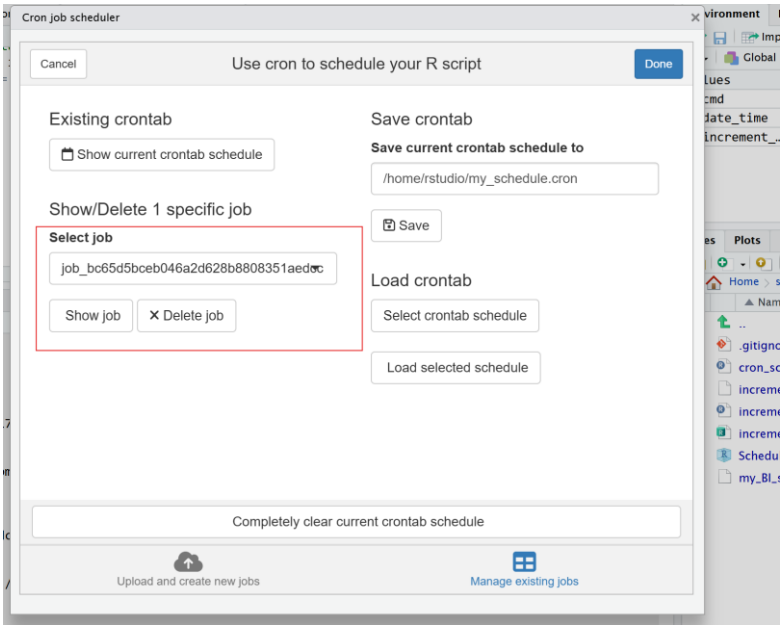
set the “increment_one.R” script to execute every hour



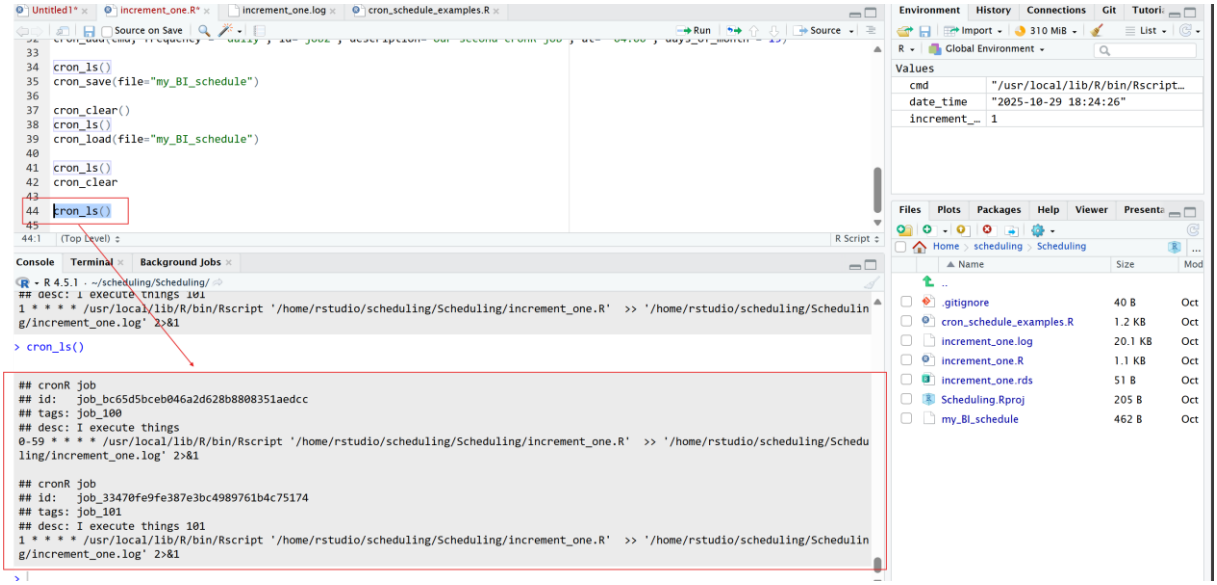
Manage existing jobs, and confirm that you can see both jobs







Or cron_ls()


Untitled1*
increment_one.R*
increment_one.log
cron_schedule_examples.R
Source on Save
Run
Source
cron_ls()
cron_save(file="my_BI_schedule")
cron_clear()
cron_ls()
cron_load(file="my_BI_schedule")
cron_ls()
cron_clear
cron_ls()
44:1 (Top Level)
R Script
Console
Terminal
Background Jobs
R 4.5.1 · ~/scheduling/Scheduling/
desc: I execute things 101
1 * * * * /usr/local/lib/R/bin/Rscript '/home/rstudio/scheduling/Scheduling/increment_one.R' >> '/home/rstudio/scheduling/Scheduling/increment_one.log' 2>&1
> cron_ls()
cronR job
id: job_bc65d5bceb046a2d628b8808351aedcc
tags: job_100
desc: I execute things
0-59 * * * * /usr/local/lib/R/bin/Rscript '/home/rstudio/scheduling/Scheduling/increment_one.R' >> '/home/rstudio/scheduling/Scheduling/increment_one.log' 2>&1
cronR job
id: job_33470fe9fe387e3bc4989761b4c75174
tags: job_101
desc: I execute things 101
1 * * * * /usr/local/lib/R/bin/Rscript '/home/rstudio/scheduling/Scheduling/increment_one.R' >> '/home/rstudio/scheduling/Scheduling/increment_one.log' 2>&1
Environment
History
Connections
Git
Import
310 MiB
List
Global Environment
Values
cmd "/usr/local/lib/R/bin/Rscript…
date_time "2025-10-29 18:24:26"
increment_… 1
Files
Plots
Packages
Help
Viewer
Home › scheduling › Scheduling
Name
Size
Mod
.gitignore 40 B Oct
cron_schedule_examples.R 1.2 KB Oct
increment_one.log 20.1 KB Oct
increment_one.R 1.1 KB Oct
increment_one.rds 51 B Oct
Scheduling.Rproj 205 B Oct
my_BI_schedule 462 B Oct