# {pins} + {shiny}

A love story

# What is {pins}?

{pins} is a package that helps you **read**, **write** and **publish** data, models and R objects in local or virtual machines

- It is actively developed & maintained by Rstudio
- Works well in combination with other packages like Shiny or Posit Connect

With {pins} it is easy **to share** the objects across multiple projects, users or even languages.

Think of it as an enhanced “read.csv” and “write.csv”

## What {pins} is not?

{pins} is not a database system, and it shouldn't substitute your data engineering pipeline

Don't use it as a permanent way of storing your files

And, you shouldn't use it for big size objects (higher than 500MB)

It is mostly meant for **sharing** and making your **workflow smoother**

## How does it work?

Just like you'd pin a note to a physical cork board, {pins} lets you pin an object, or a file to a **virtual (or local) board**

First register one of the supported boards with **`board_xxx()`**

- Posit Connect (formerly RStudio Connect)
- Amazon S3 / Microsoft 365 / Azure  / Google Cloud
- Local Folder (Dropbox)  / Temporal Folder

Then you can just read and write

- **`pin_read()`**
- **`pin_write()`**

User Group Finland

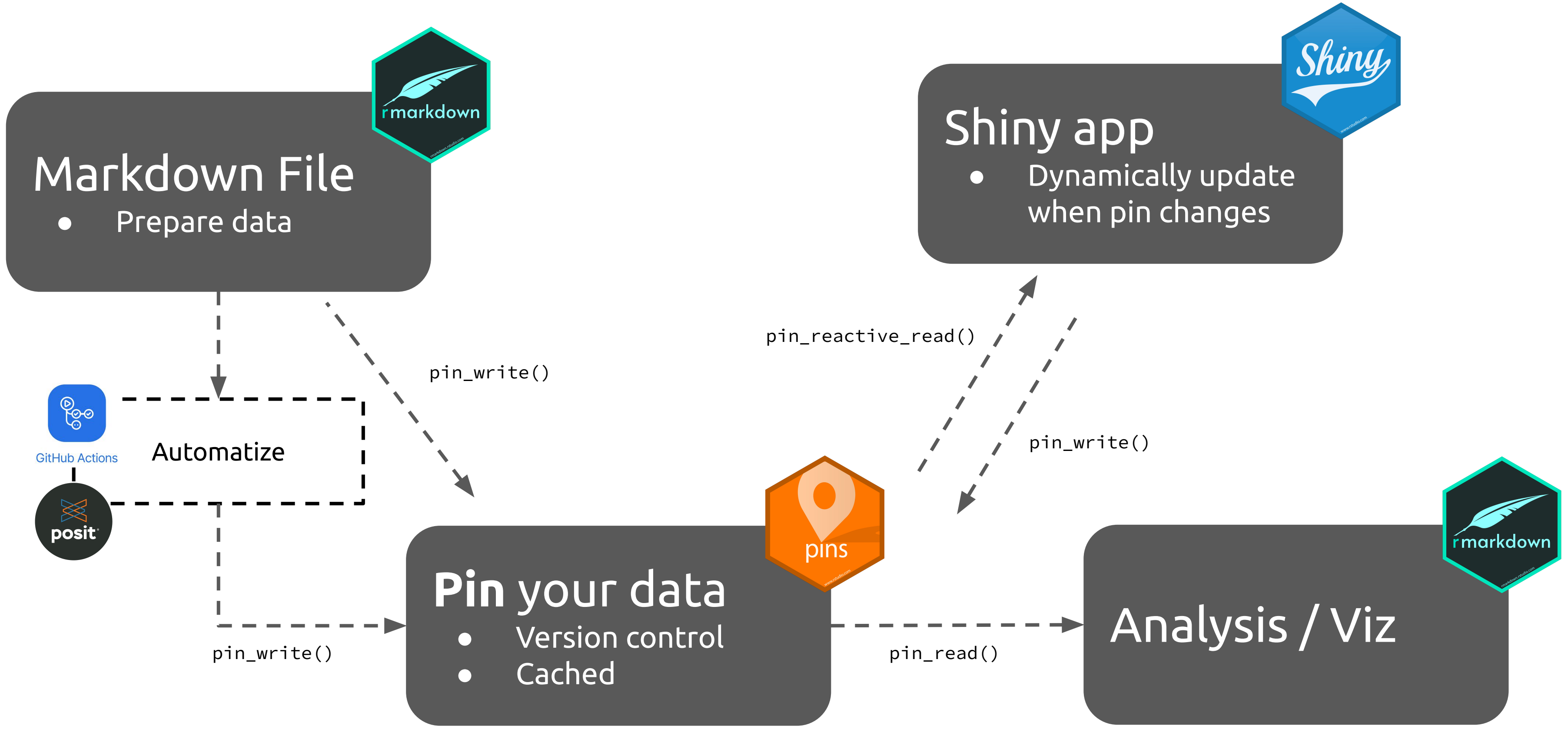
## What are some of the superpowers of {pins}?

- Boards and pins can have **version control on / off**
  - By default you always access the latest one, but you can access specific versions
- Pins have **metadata**
  - Basic things like title, description, or others
- Online pins are **cached**
  - It is only reloaded / downloaded if it has changed
- Pins can be **shared between R and Python**

## What are some of the superpowers of {pins}?

- Pins can be reactively read inside a **shiny app** in a really smooth and convenient way

# How might your workflow look like with {pins}?



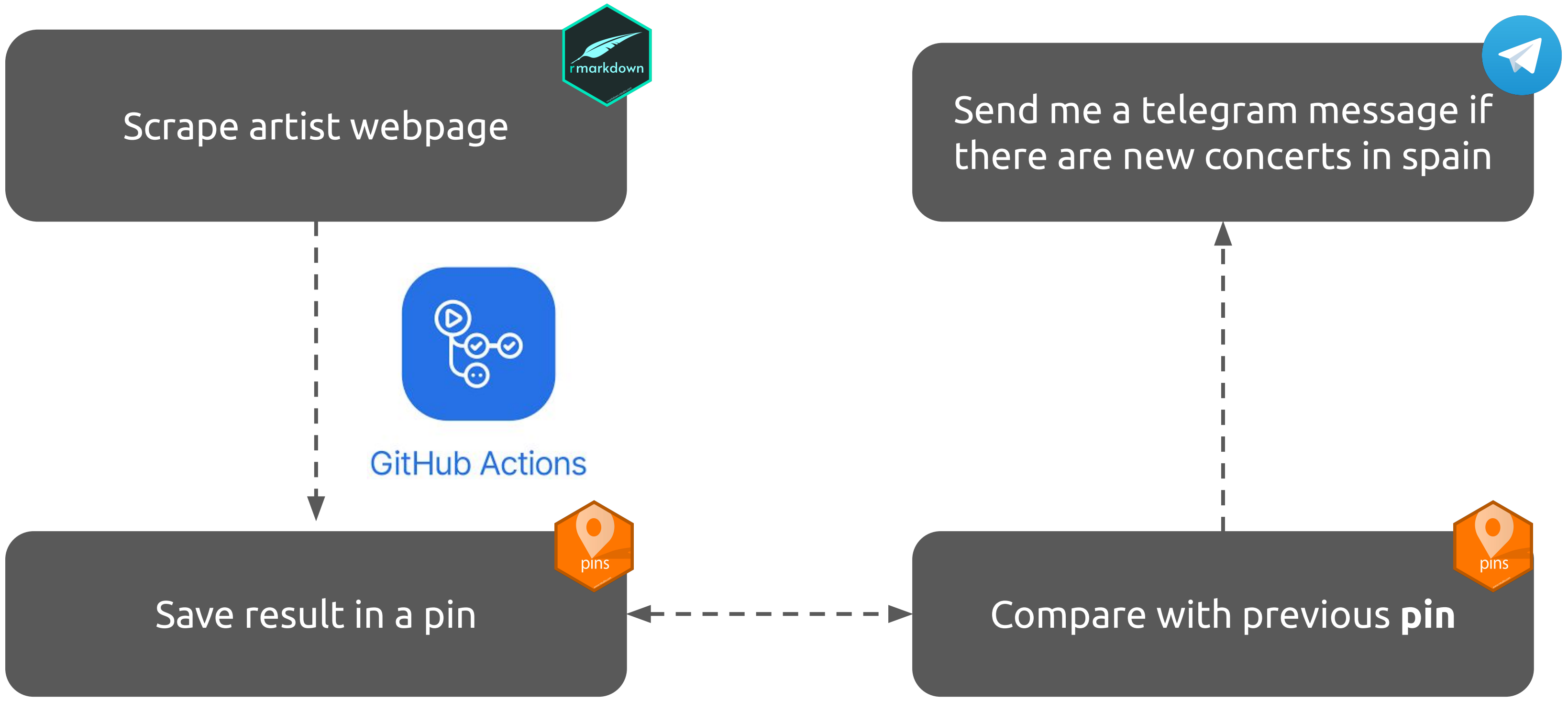
# Why do you need this step?

- For normal analysis → It is good practice!
  - Easier to reproduce
  - Easier to share
  - Reusable

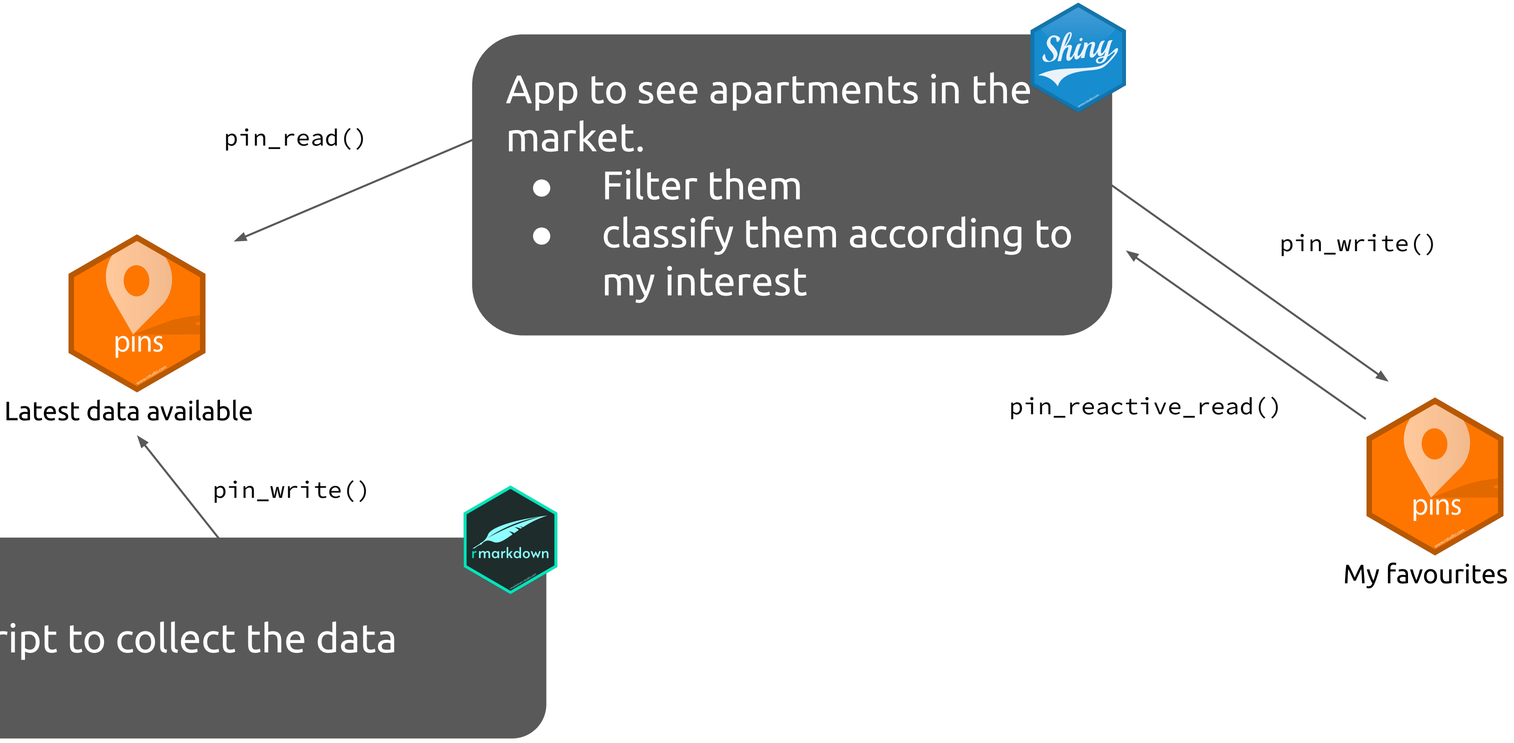
- For Shiny Apps → Avoid having to initialize the app every time your data changes
  - Lighter app
  - Continuously running

## Let’s see some examples

# Checking for new concerts


Scrape artist webpage
rmarkdown
GitHub Actions
Save result in a pin
pins
Compare with previous **pin**
pins
Send me a telegram message if there are new concerts in spain

# Dynamically update a plot in shiny



App to see apartments in the market.

- Filter them
- classify them according to my interest

Latest data available

My favourites

A script to collect the data