

How I Learned to Stop Worrying and Love the Firewall


Ian Lyttle, Schneider Electric
@ijlyttle

Other works in this series



Jennifer Thompson
RLadies Introduction to Purrr

PREAMBLE:
STOP WORRYING AND
**LOVE
LISTS**



- Lists in R are collections of elements - that's it
- Each element can be any length and any type... even another list (it's lists all the way down...)
- Totally valid example:

```
list("a" = 1:10,      ## numeric vector of length 10  
     "b" = list(1:10), ## list of length 1; element 1 = vector of length 10  
     "c" = LETTERS[1:10]) ## character vector of length 10
```
- With such flexibility comes both great power & great complexity
- purrr works really well with lists by providing ways to:
 - iterate quickly over lists comprising elements of the same type
 - quickly extract elements of complicated lists

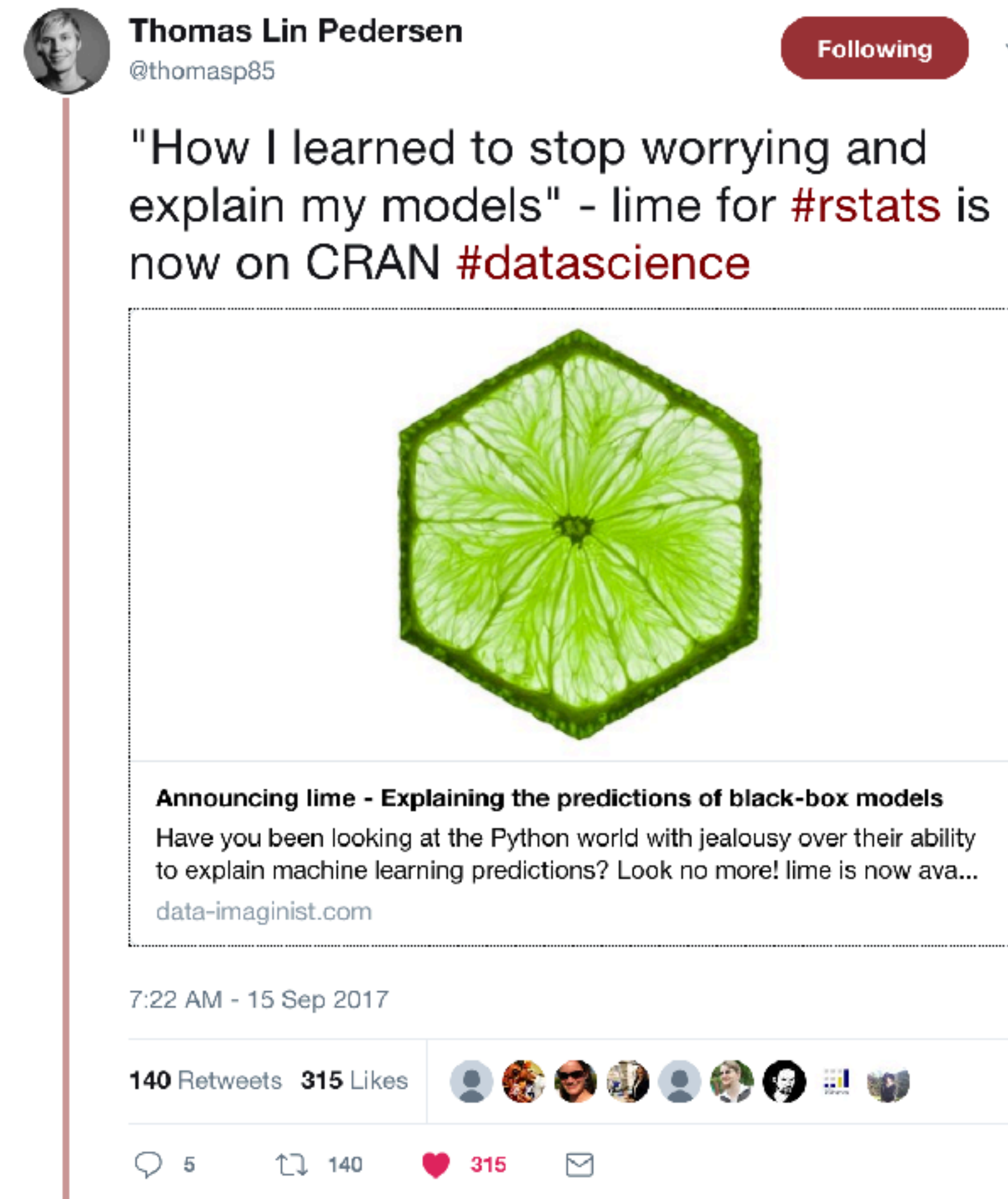
<https://github.com/jenniferthompson/RLadiesIntroToPurrr>

Other works in this series



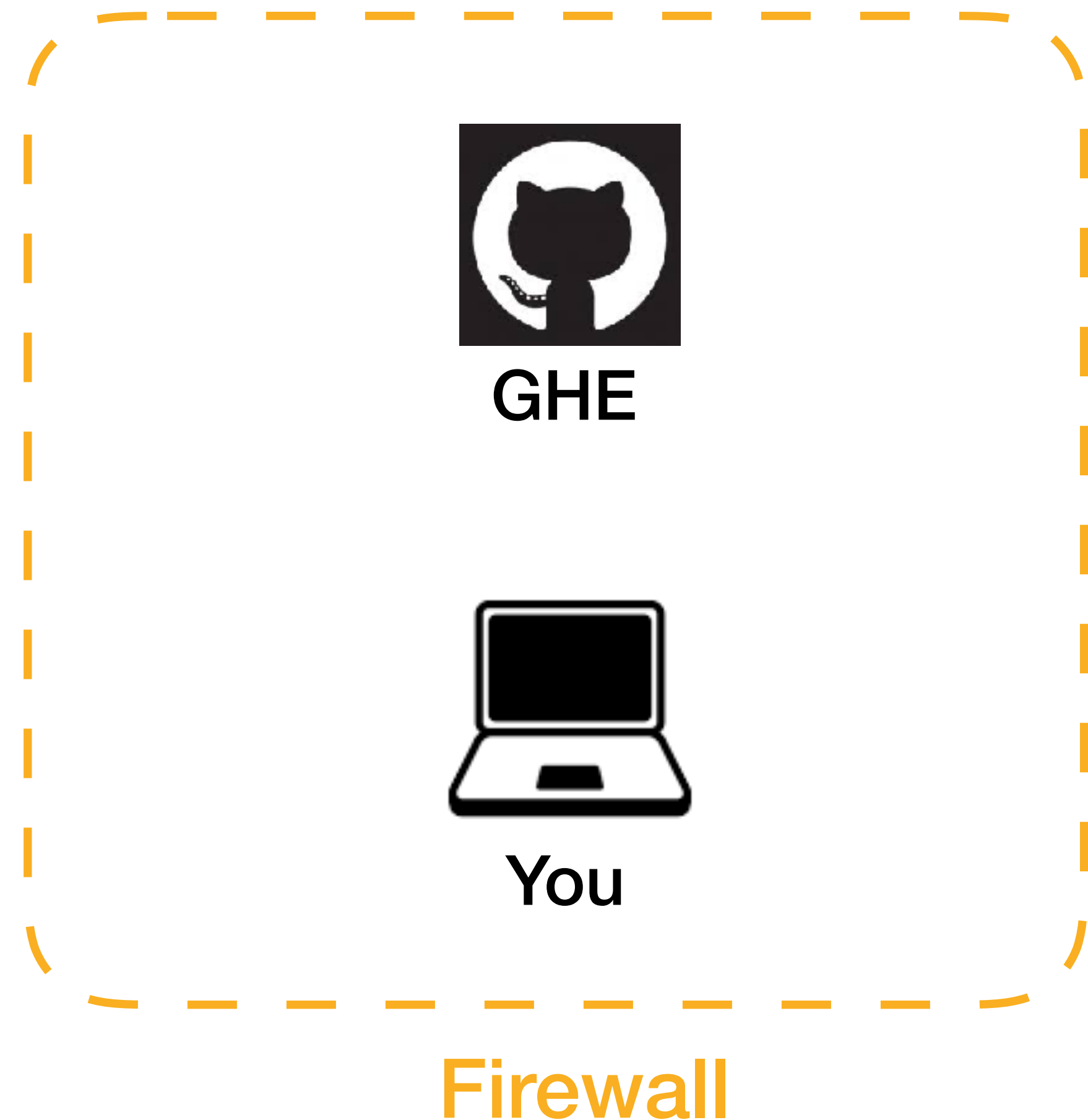
Thomas Lin Pedersen
Lime

<https://github.com/thomasp85/lime>



Who are you?

- You work at an institution where you need to keep some work private.
- You would like to share within your institution, by creating an internal version of the open R ecosystem.
- You have an instance of GitHub Enterprise
- For the purposes of this talk, you work at Acme Corporation.



Prelim

- You almost certainly know this already; included here for completeness
- To use a proxy, set environment variables in `.Renvi`ron

`.Renvi`ron

```
no_proxy="localhost,127.0.0.1,github.acme.com,rstudio-connect.acme.com"
```

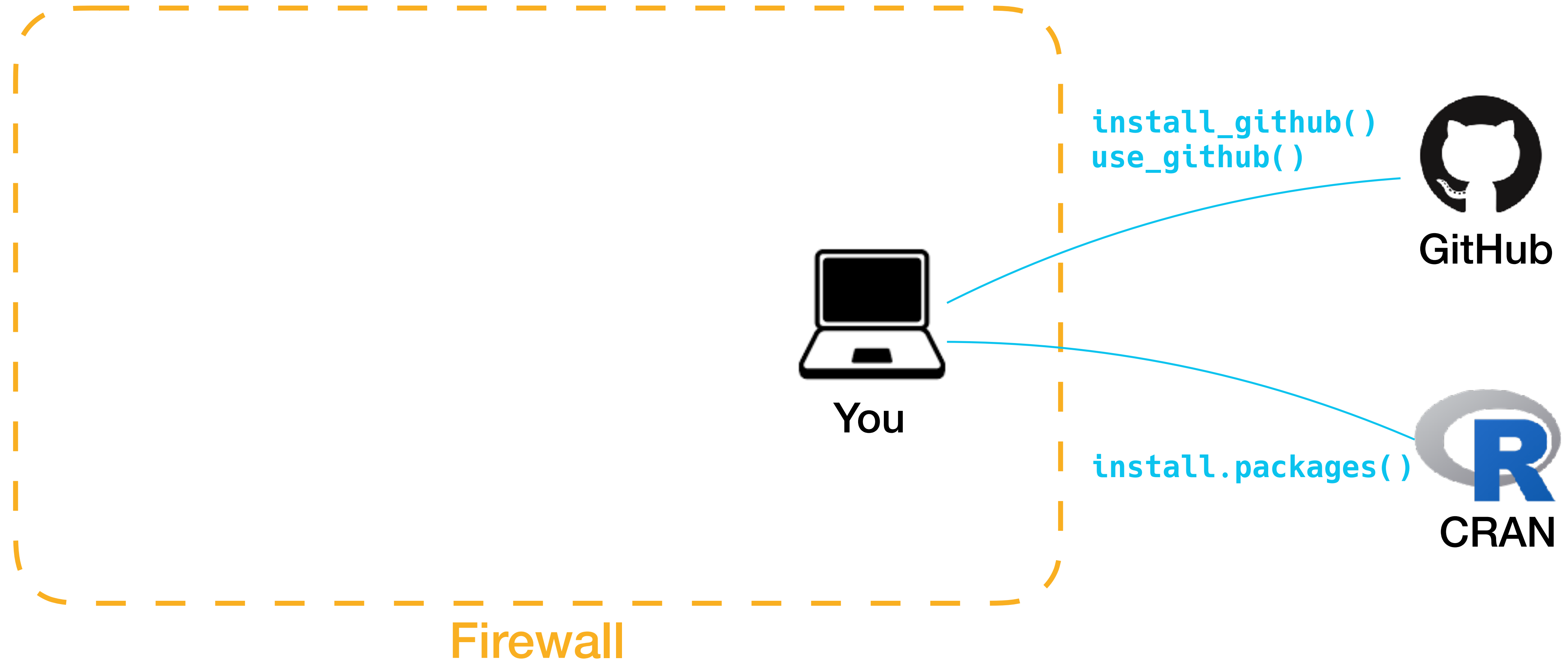
```
https_proxy="http://xxx.xxx.xxx.xxx:yyyyy" # you have to find this
```

```
http_proxy="http://xxx.xxx.xxx.xxx:yyyyy"
```

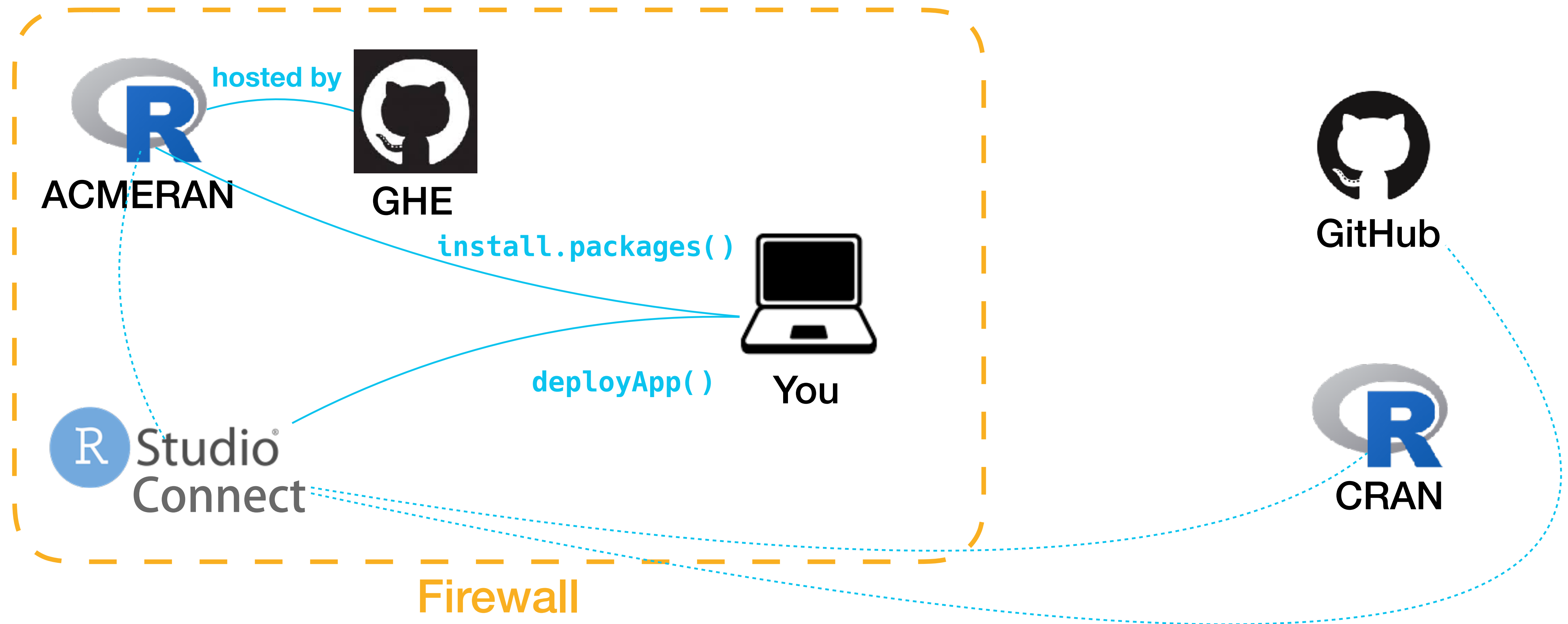
```
HTTPS_PROXY="${https_proxy}"
```

```
HTTP_PROXY="${http_proxy}"
```

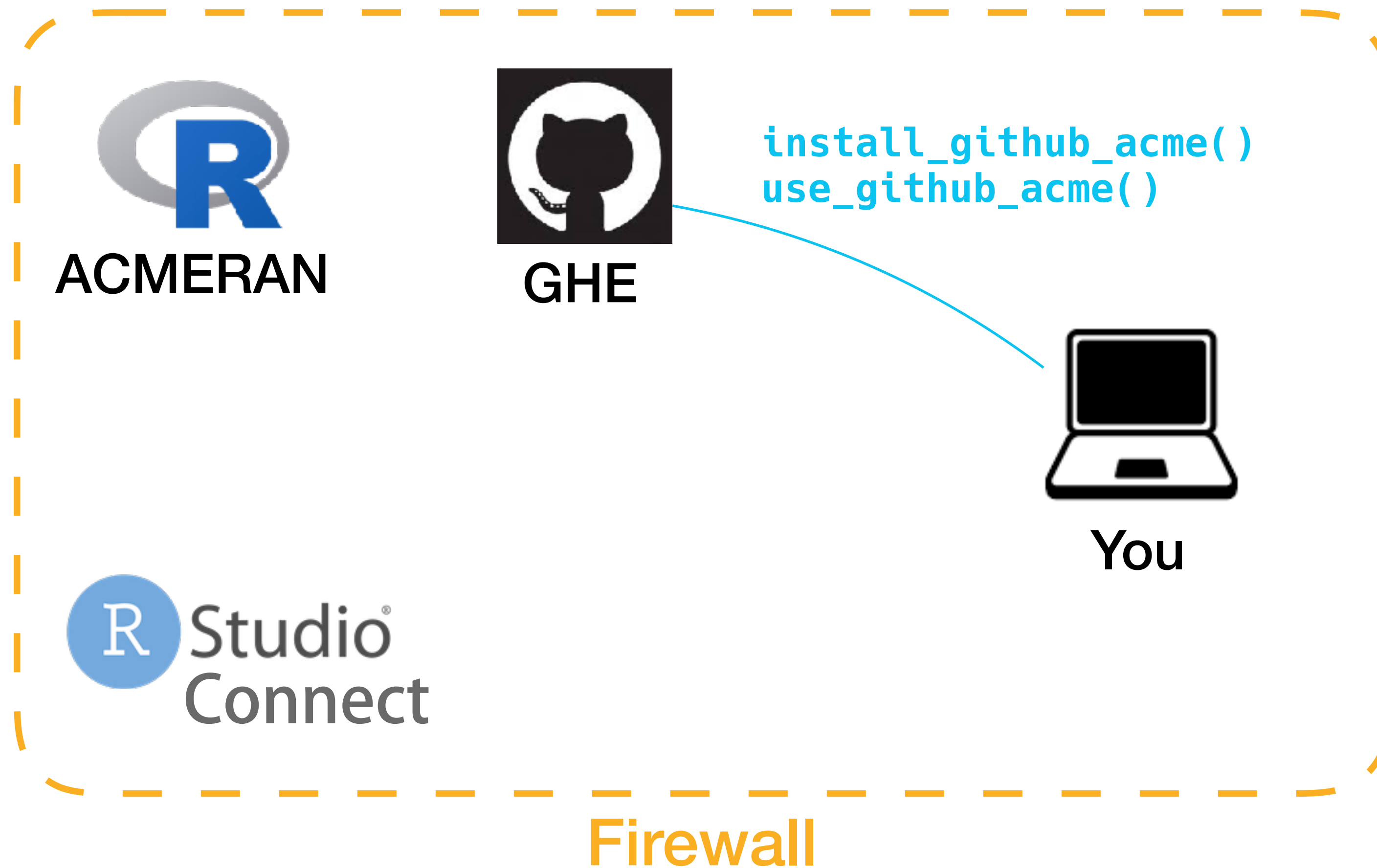
How does this all fit together?



How does this all fit together?



How does this all fit together?



Using your GitHub Enterprise

- We have nifty functions to establish repositories and install from GitHub.
- They have a `host` argument, so you can use them with GHE:

```
usethis::use_github(host = "https://github.acme.com/api/v3", auth_token = Sys.getenv(...))  
devtools::install_github("user/repo", host = "github.acme.com/api/v3")
```

- It's a hassle to type all these arguments.
- It would be nice to wrap these up in functions in your **acmetools** package.

Enter ghentr

- **ghentr** is a package with templating functions to access your GHE
- templating provided by **usethis** (Hadley Wickham and Jenny Bryan)

```
usethis::use_github(host = "https://github.acme.com/api/v3", auth_token = Sys.getenv(...))  
devtools::install_github("user/repo", host = "github.acme.com/api/v3")
```



```
acmetools::use_github_acme()  
acmetools::install_github_acme("user/repo")
```

Using ghentr

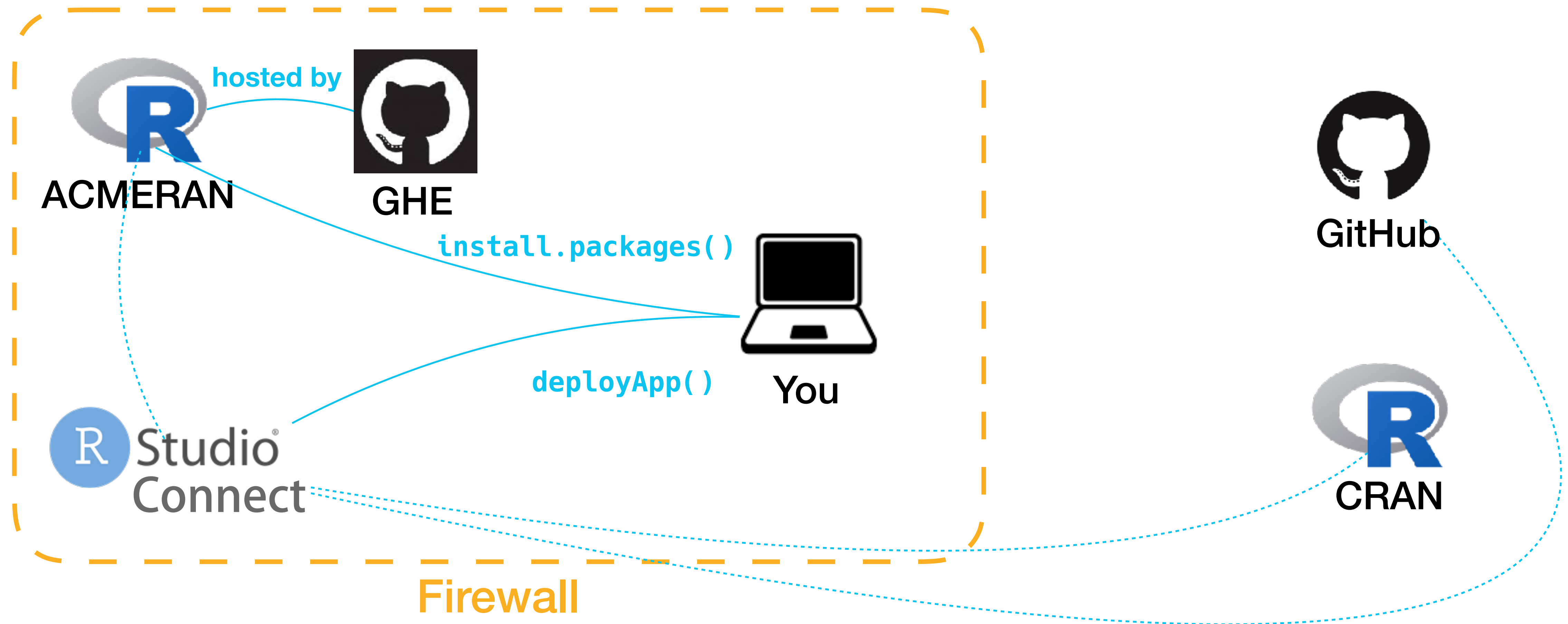
- **you** create a package, **acmetools**, then:

```
ghentr::use_github_enterprise(  
  host = "github.acme.com/api/v3",  
  suffix = "acme",  
  name = "Acme Corporation"  
)
```

- **ghentr** writes documented function-files to R directory of **acmetools** project:

```
use_github_acme <- function(organisation = NULL,  
                             private = FALSE,  
                             protocol = c("ssh", "https"),  
                             credentials = NULL,  
                             auth_token = github_acme_pat(),  
                             host = "https://github.acme.com/api/v3") {  
  usethis::use_github(...)  
}
```

How does this all fit together?



Using a repository

- Native way for R to install packages
- Lets you deploy work to RStudioConnect or use the **packrat** package
- As a user, all you have to do is add a single line to `.Rprofile`

`.Rprofile`

```
options(  
  repos = c(  
    CRAN = "https://cran.rstudio.com/",  
    ACMERAN = "https://pages.github.acme.com/ACME-R/ACMERAN/"  
  ),  
  ...  
)
```


Care and feeding of a repository

- Dirk Eddelbuettel and colleagues created **drat**: Drat R Archive Template
- A CRAN-like repository is simply a filesystem made available using a URL.
- The **drat** package helps you to build this filesystem.
- You can use GitHub Enterprise to host this filesystem, make it available using GitHub pages.

Initializing repository

- Create new project in RStudio

Empty project will do

- Create the filesystem

```
ghentr::init_drat_repo()  
ghentr::make_drat_bin_placeholders()
```

Creates directory for source packages
Creates placeholders for binary packages

- Edit your .Rprofile

```
options(dratRepo = "path/to/repo")
```

Used by drat to determine path when
inserting a package into repository

- Create git repository, establish GHE repository

```
usethis::use_git()  
acmetools::use_github_acme()
```

Creates remote at your GHE and pushes

- At GHE, activate pages on master branch
- Publicize URL for repository

Suggest to use master branch so that
filesystem will not be shared by default

Adding a package

On same computer as DRAT repository, open package-project:

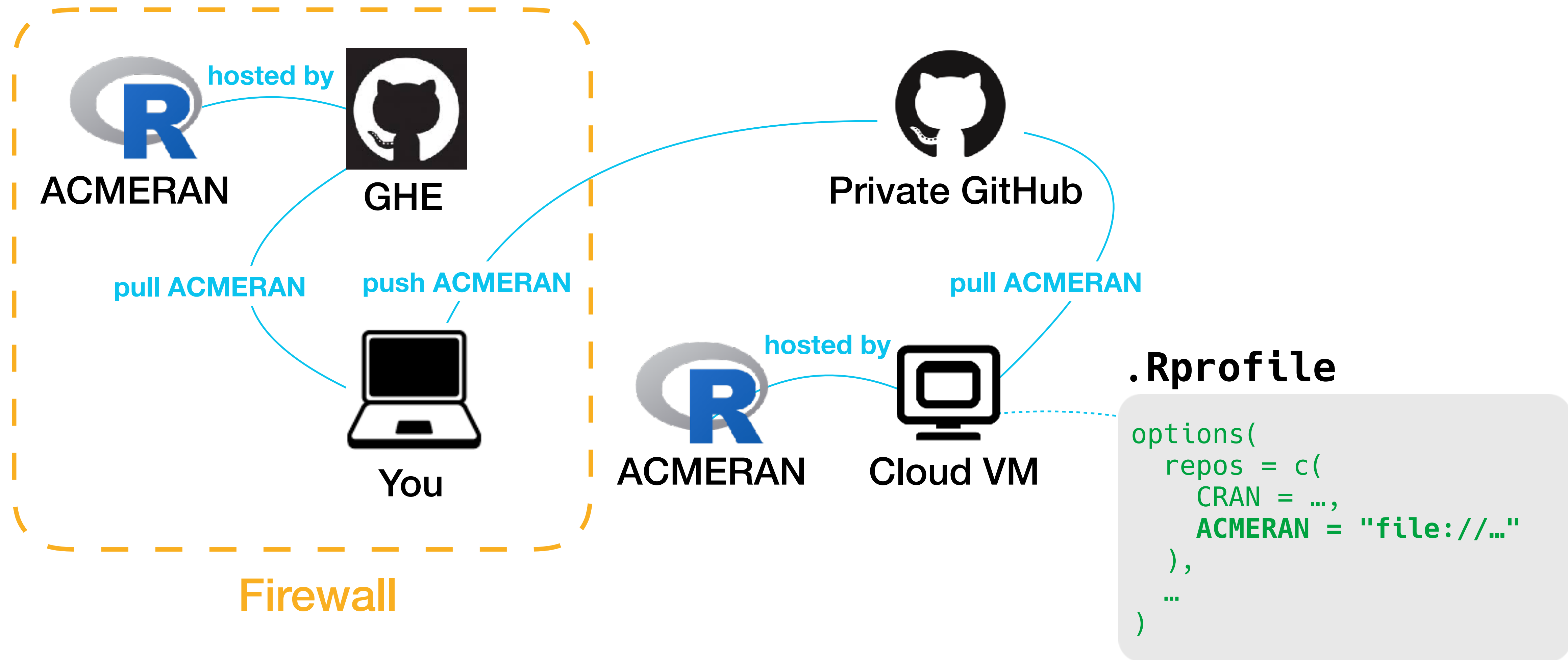
- Add "Repository" field to DESCRIPTION
`ghentr::use_drat_repository()`
- Prepare and build source package for release
`devtools::check()`
`devtools::build()`
- Insert package into DRAT repository
`drat::insertPackage("path/to/pkg.tar.gz")`
- Commit and push DRAT repository

Lets **packrat** and **RStudioConnect** figure out how to recreate your packages

`build()` returns string describing the path to the package tar.gz file

`insertPackage()` uses that string

One more scenario



Acknowledgements

usethis authors:

- Hadley Wickham
- Jenny Bryan

drat authors:

- Dirk Eddelbuettel et al.

ghentr checker:

- Emily Bosak



Acme's #1 Customer

Thank you

ghentr package

- helps you use *your* instance of GitHub Enterprise
- lets you build functions:
`use_github_acme()`
`install_github_acme()`
- helps you build an internal CRAN-like repository using **drat**
- not yet on CRAN:
`install_github("ijlyttle/ghentr")`

Ian Lyttle
Schneider Electric

 @ijlyttle

 <https://ijlyttle.github.io/ghentr/>