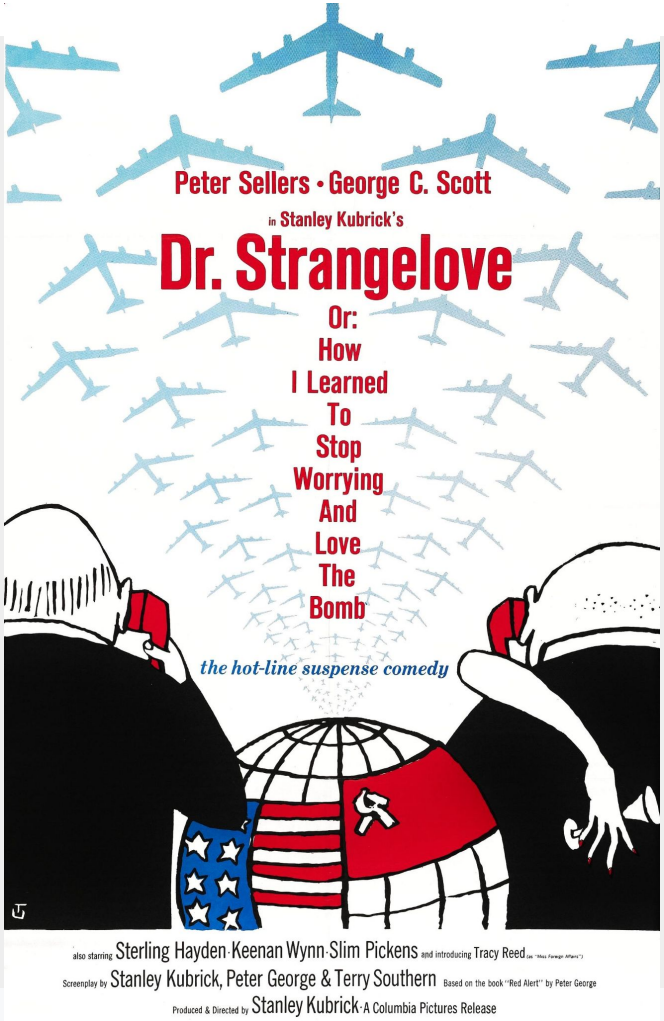

LOMA
Why I stopped worrying and learned to love the API

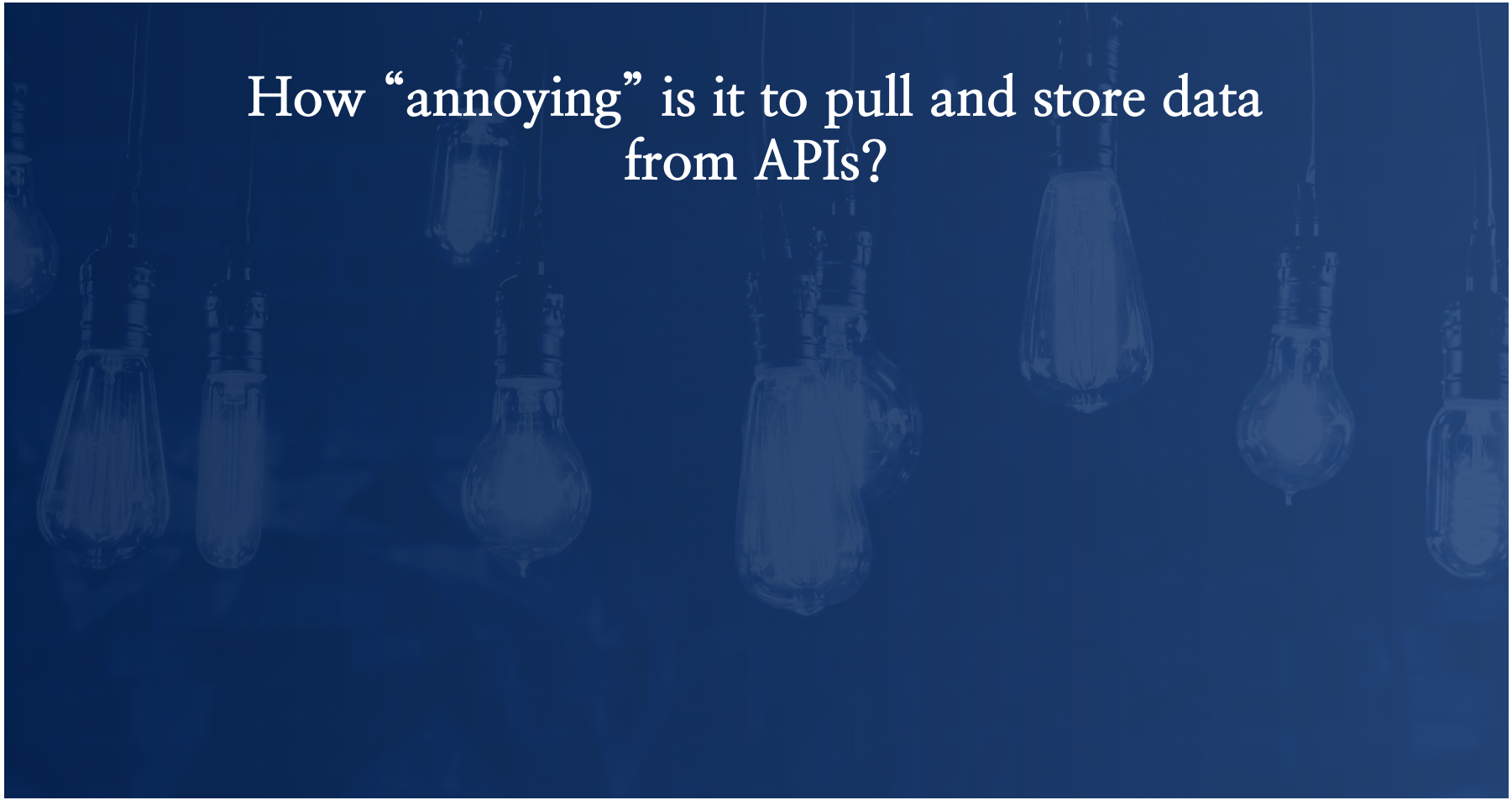

Peter Sellers • George C. Scott
in Stanley Kubrick's
Dr. Strangelove
Or:
How
I Learned
To
Stop
Worrying
And
Love
The
Bomb
the hot-line suspense comedy
also starring Sterling Hayden · Keenan Wynn · Slim Pickens and introducing Tracy Reed
Screenplay by Stanley Kubrick, Peter George & Terry Southern Based on the book "Red Alert" by Peter George
Produced & Directed by Stanley Kubrick · A Columbia Pictures Release

APIs = “the bomb”

Brief Background on Looma

Human-Centric, Point of Decision Content and Distribution

How “annoying” is it to pull and store data from APIs?

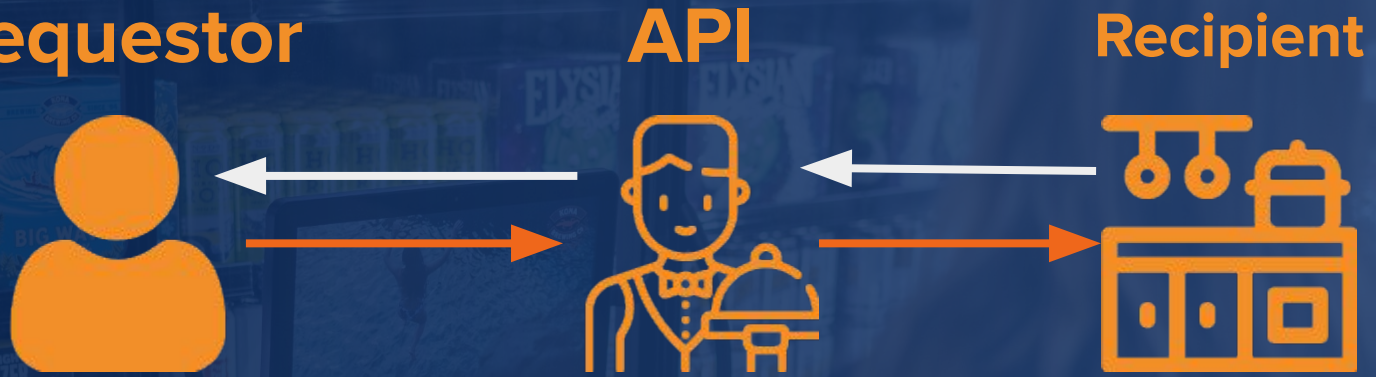


How easy is it to create and deploy an API?

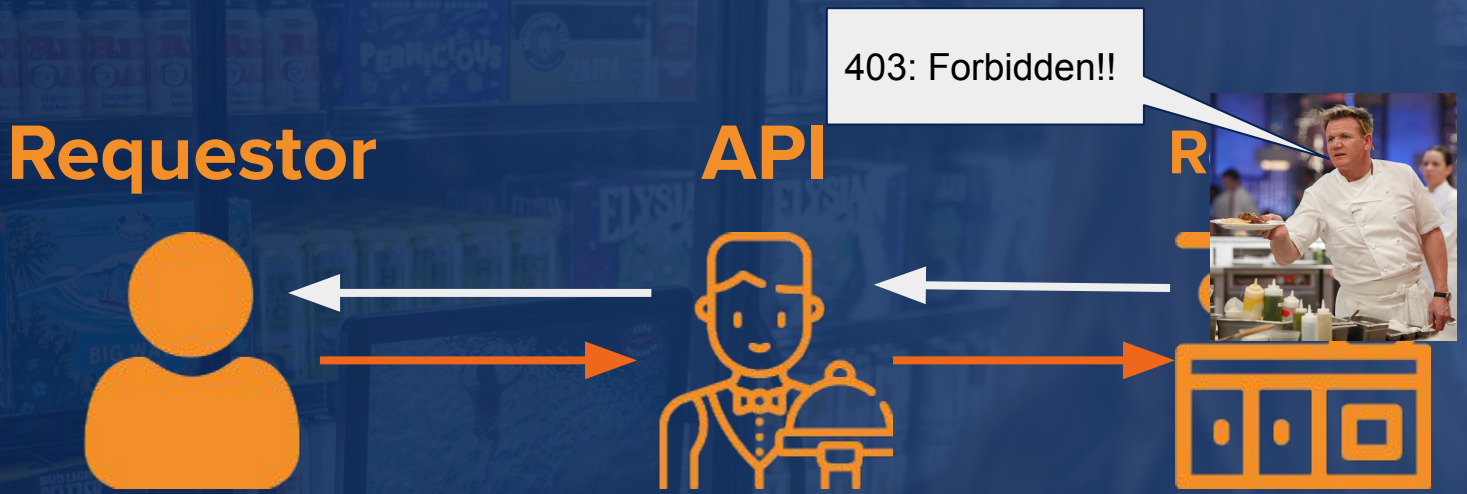
Agenda

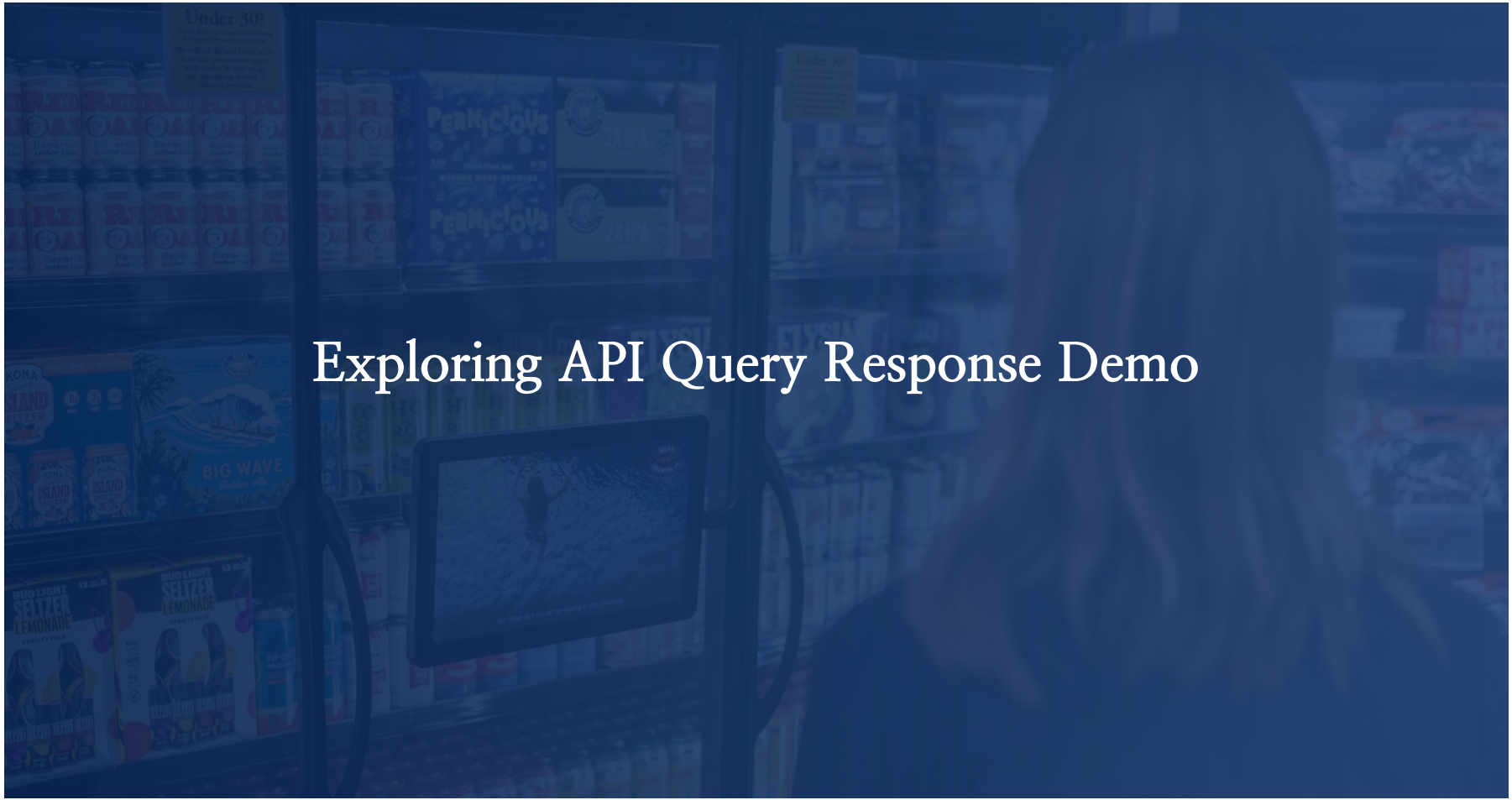
1. **Querying External APIs**
   - Exploring, Formatting and Organizing Queries
   - Productionalizing API Extraction w/ Posit Connect
2. **Functional API Request Generation w/ R**
   - Example: SCOR Typeform Surveys
3. **Creating and Deploying APIs with Posit Connect**
   - Sharing Data
   - Ingesting Data

What is an API?



What is an API?





Exploring API Query Response Demo

R Experience

Name	Type	Value
weather_content	list [1]	List of length 1
KRDU	list [15]	List of length 15
station_id	character [1]	'KRDU'
raw	character [1]	'KRDU 121451Z 08009KT 10
temp	character [1]	'8.3'
dewpoint	character [1]	'2.2'
wind	character [1]	'80'
wind_vel	character [1]	'9'
visibility	character [1]	'10.0'
alt_hg	character [1]	'30.20'
alt_mb	character [1]	'1023.0'
wx	NULL	Pairlist of length 0
auto_report	character [1]	'true'
sky_conditions	list [1]	List of length 1
category	character [1]	'VFR'
report_type	character [1]	'METAR'
time_of_obs	character [1]	'2022-12-12T14:51:00Z'

Terminal Experience

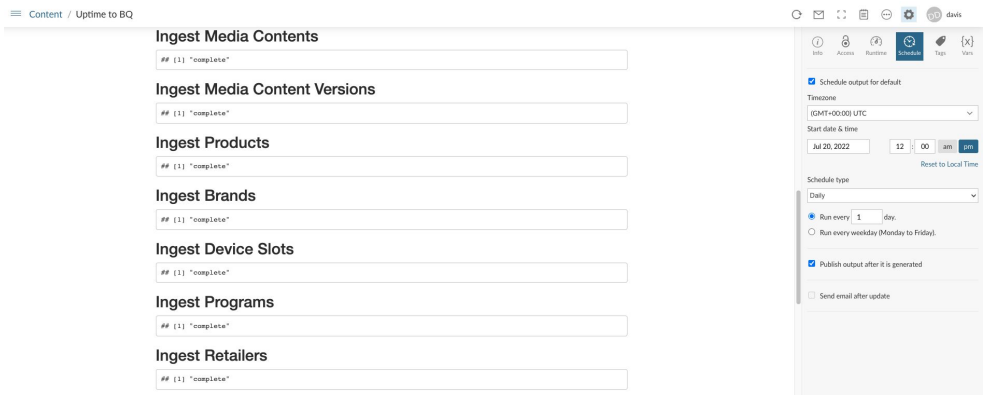
```
(base) ~ (0.545s)
curl -X GET "https://api.aviationapi.com/v1/weather/metar?apt=RDU"
{"KRDU":{"station_id":"KRDU","raw":"KRDU 121451Z 08009KT 10SM FEW035 08\/02 A3021 RMK AO2 SLP230 T00830022 51021","temp":"8.3","dewpoint":"2.2","wind":"80","wind_vel":"9","visibility":"10.0","alt_hg":"30.20","alt_mb":"1023.0","wx":null,"auto_report":"true","sky_conditions":[{"coverage":"FEW","base_agl":"3500"}],"category":"VFR","report_type":"METAR","time_of_obs":"2022-12-12T14:51:00Z"}}
```

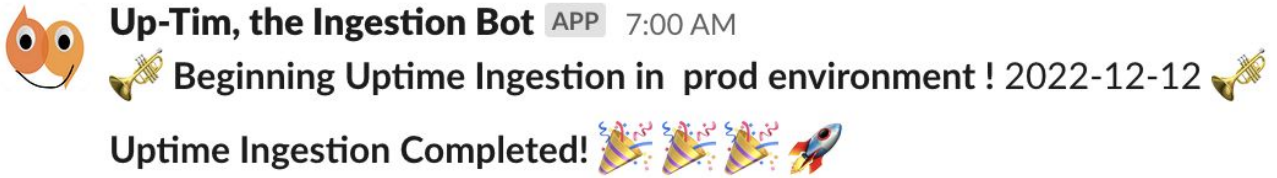
2 types of functions

- Query Generation
  - Get the actual text of the query
- Format for Output
  - Make the API Call
  - Format the response

```r
# Example Functions for Package ----------------------------------------

query_weather_response <- function(airport_code){

  glue::glue("https://api.aviationapi.com/v1/weather/metar?apt=", airport_code, sep = '')


}

get_weather_response <- function(airport_code){

  #The API call
  weather_response <- httr::GET(url = query_weather_response(airport_code))

  #Extract the content from the response
  weather_content <- httr::content(weather_response)

  #Format them using tidyverse functions
  weather_tbl <- tibble(weather_content) %>%
    unnest_wider(col = c(weather_content)) %>%
    unnest_longer(col = c(sky_conditions)) %>%
    unnest_wider(col = c(sky_conditions))

  return(weather_tbl)

}

atl_weather <- get_weather_response('ATL')
```

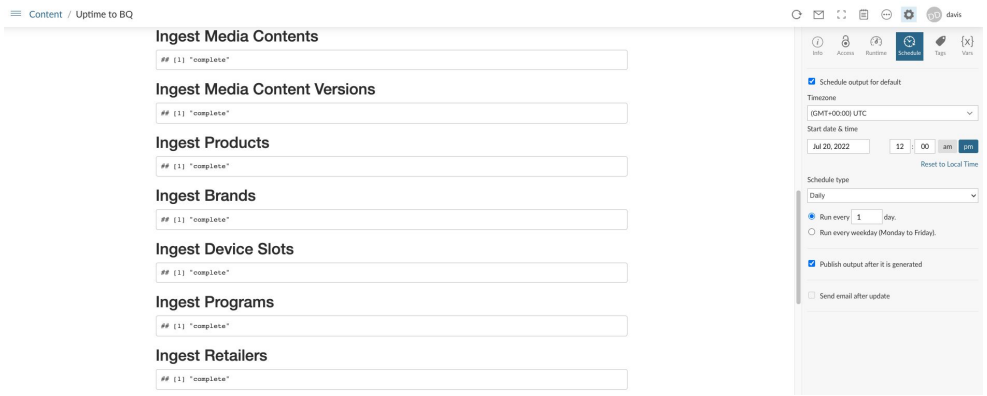
- Make a package including all of your functions!
- Make an .Rmd (or Quarto)
- Publish it and schedule it's generation on Posit Connect
- Even add a SlackBot using the slackr package

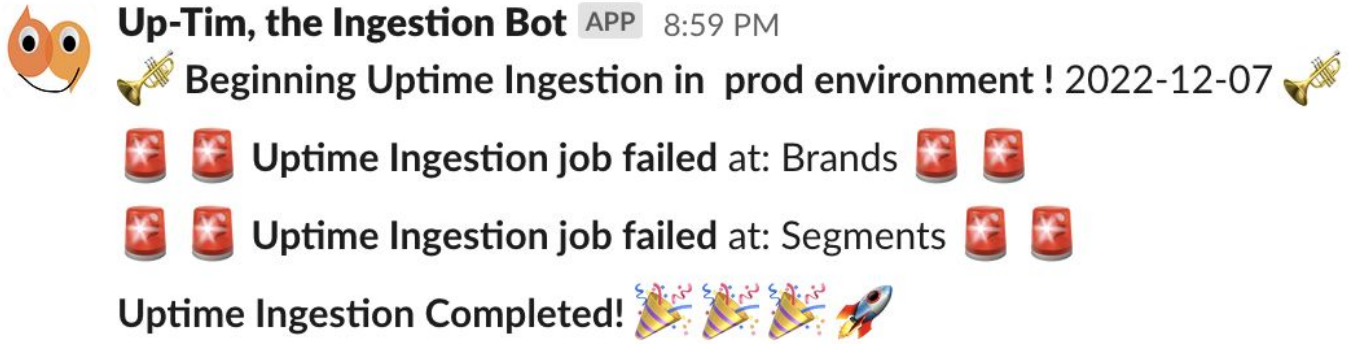




Productionalizing API Querying

- Make a package including all of your functions!
- Make an .Rmd (or Quarto)
- Publish it and schedule it's generation on Posit Connect
- Even add a SlackBot using the slackr package





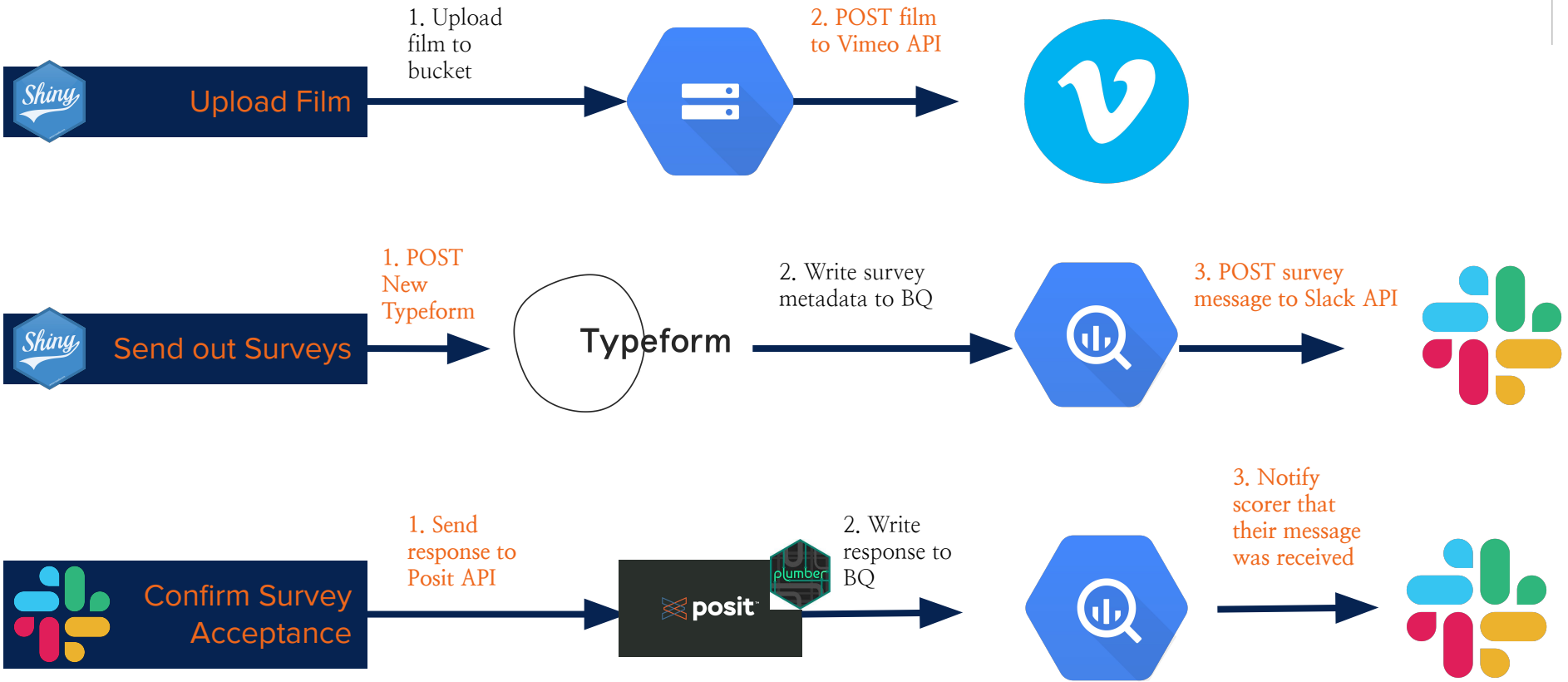
Agenda

1. **Querying External APIs**
   - Exploring, Formatting and Organizing Queries
   - Productionalizing API Extraction w/ Posit Connect
2. **Dynamic External API Posting w/ R**
   - Example: SCOR Typeform Surveys
3. **Creating and Deploying APIs with Posit Connect**
   - Sharing Data
   - Ingesting Data

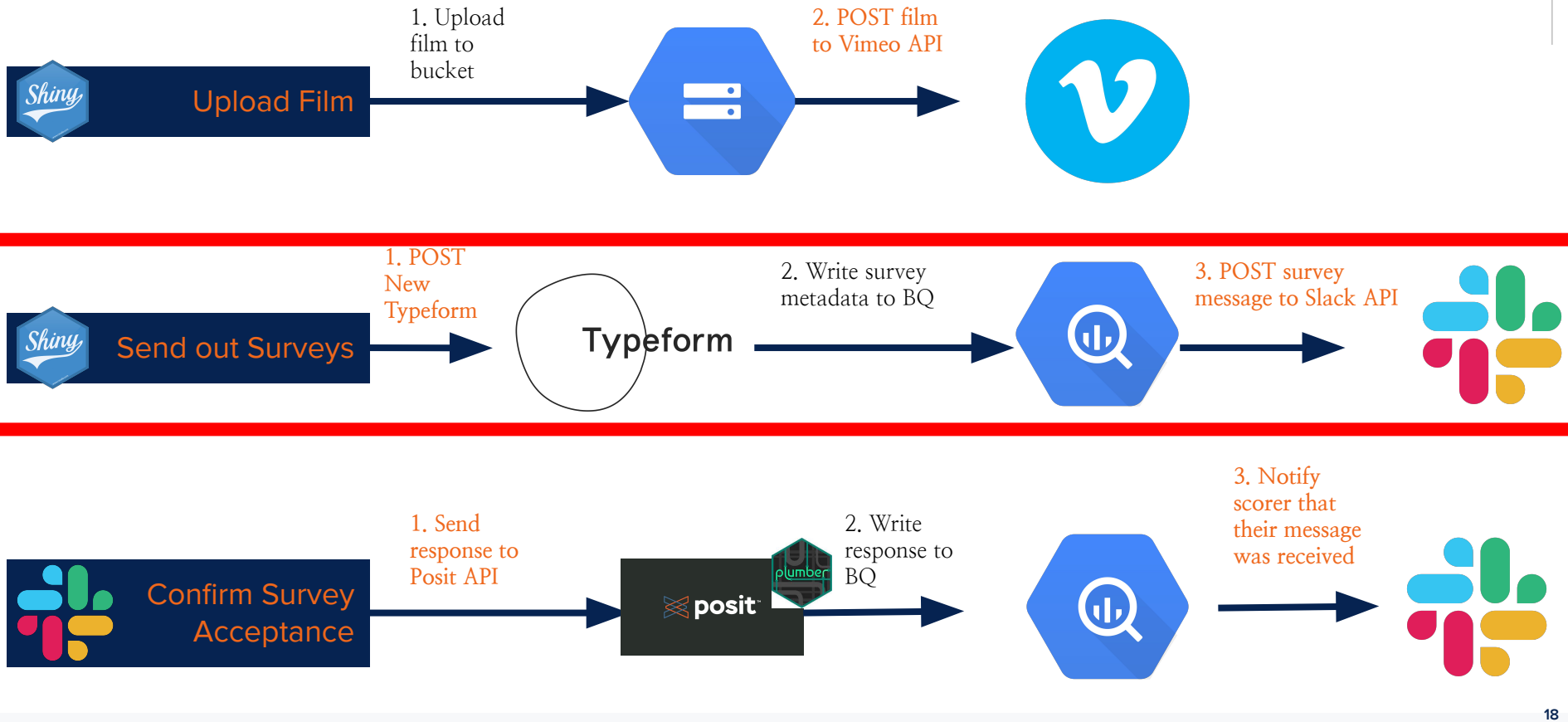
Background

- Send surveys internally asking for employees to watch and review films
- We needed:
  - UI to upload and manage films
  - Click-button generate Typeform survey
  - Slack all requested users the survey
  - Ability for users to reject scoring request

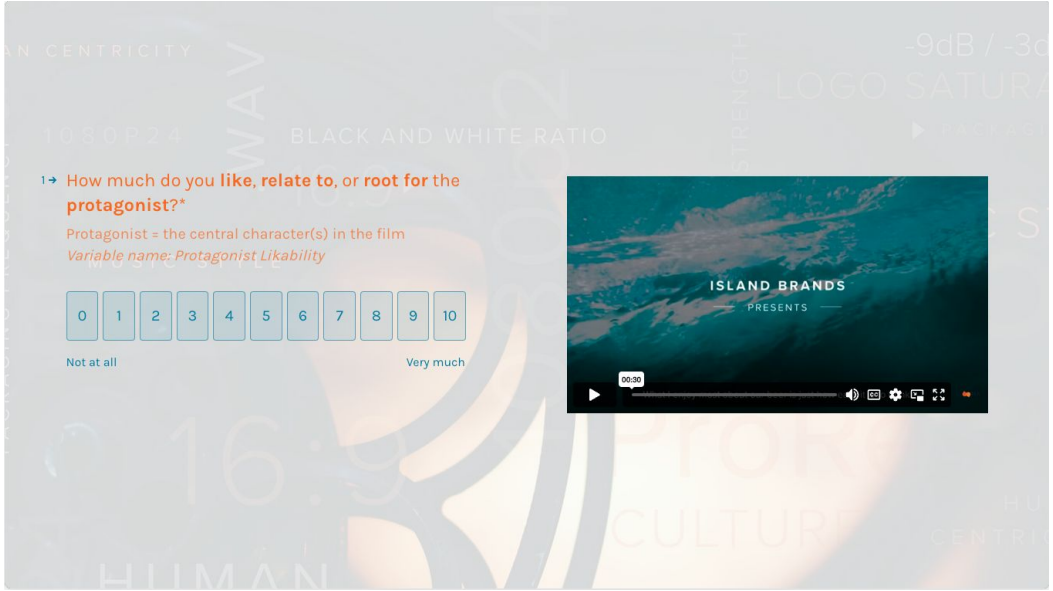
SCOR Infrastructure



SCOR Infrastructure



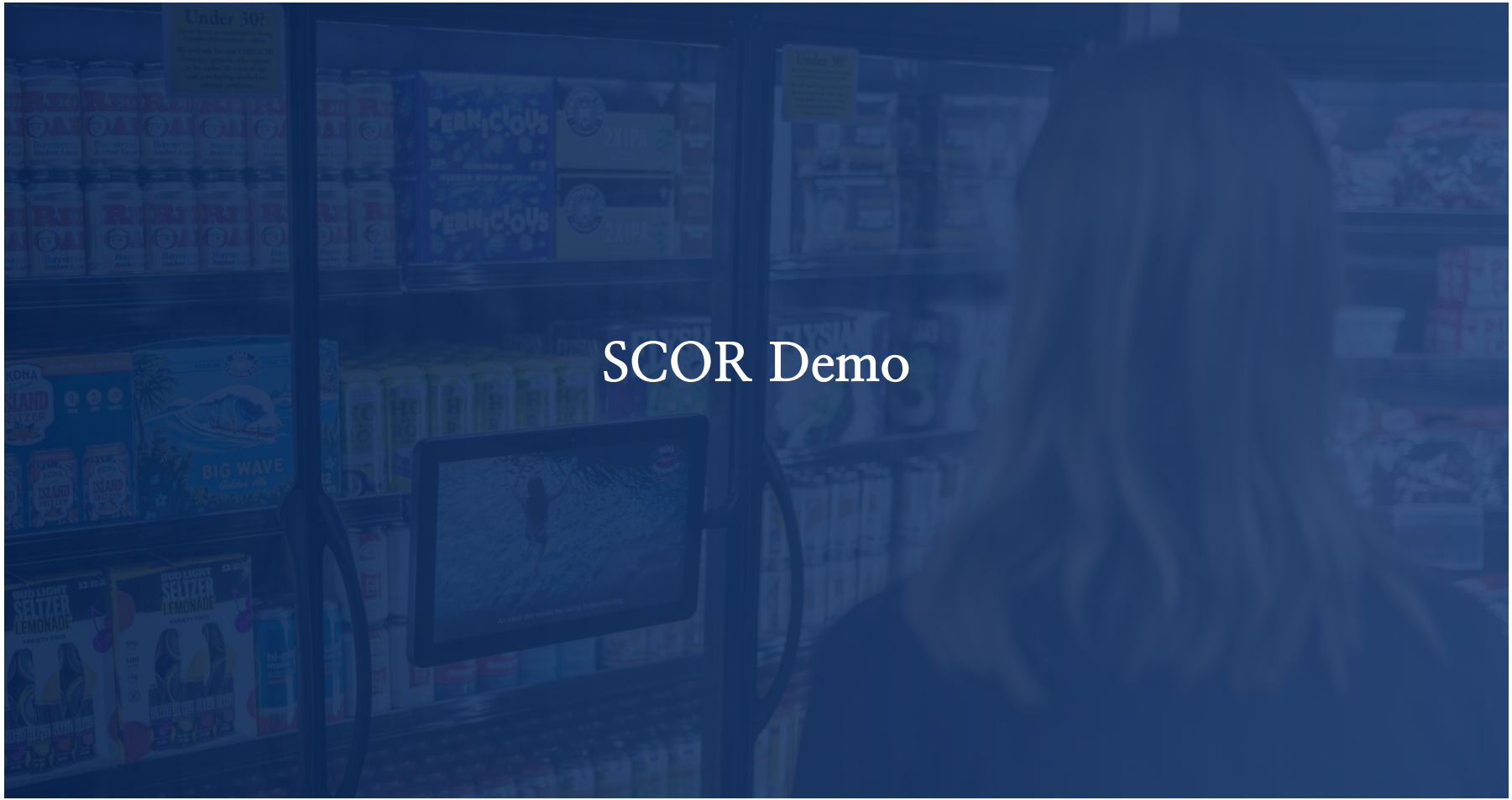
- Manual survey creation would be challenging because have to manually insert Vimeo link on every question
- R can be used to generate queries for the exact survey we need



paste +

- Base on the parameters from the options selected in Shiny
- The “Format the Output and Push” function sends this to the Typeform API and creates the survey

```
{
  "title": "How much do you *like*, *relate to*, or *root for* the *protagonist*?",
  "ref": "protagonist_likability",
  "properties": {
    "description": "Protagonist = the central character(s) in the film\\n_Variable name: Protagonist Likability_",
    "start_at_one": false,
    "steps": 11,
    "labels": {
      "left": "Not at all",
      "right": "Very much"
    }
  },
  "validations": {
    "required": true
  },
  "type": "opinion_scale",
  "attachment": {
    "type": "video",
    "href": "', vimeo_film_link, '"
  },
  "layout": {
    "type": "float",
    "attachment": {
      "type": "video",
      "href": "', vimeo_film_link, '"
    },
    "placement": "right"
  }
},
```

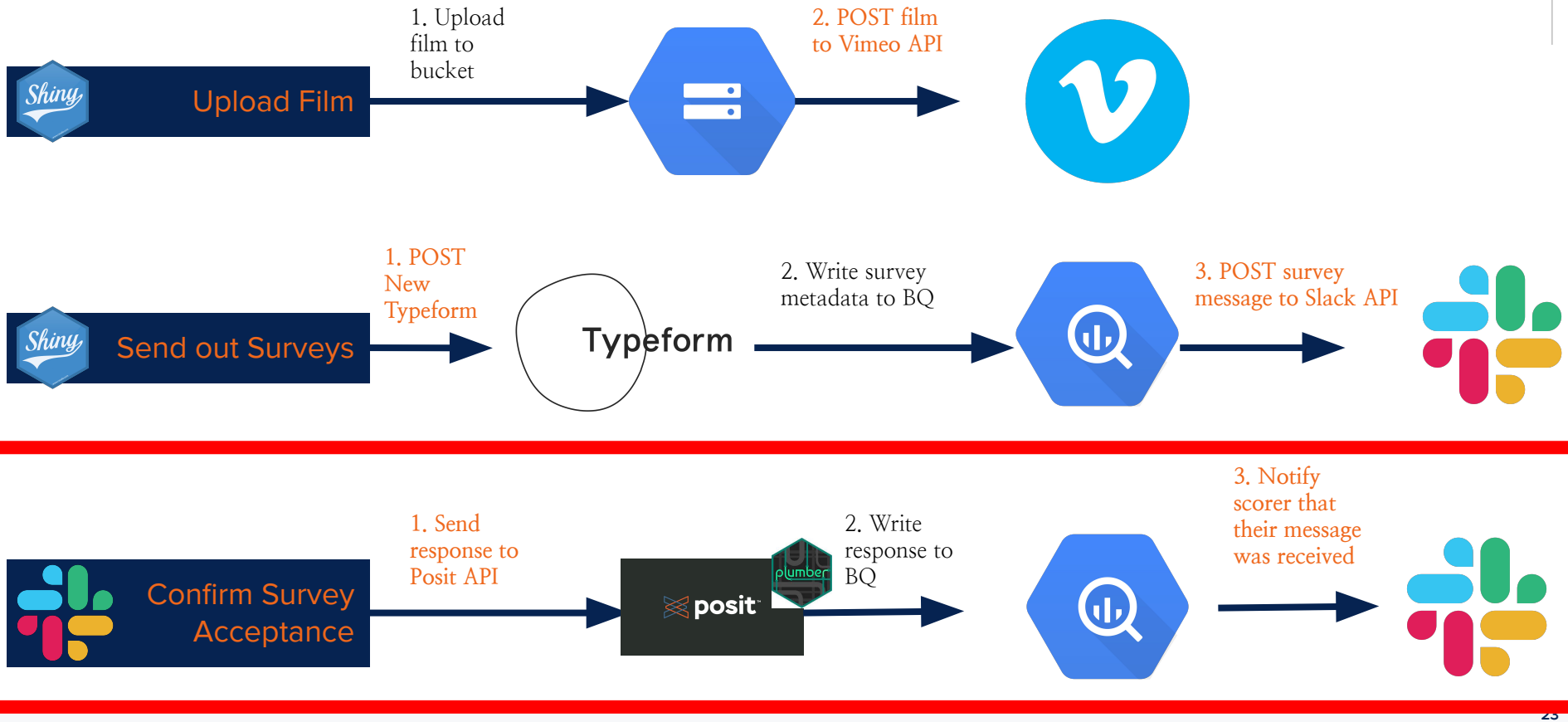

Under 30?
PERNICIOUS
PERNICIOUS
BIG WAVE
BUD LIGHT SELTZER LEMONADE
BUD LIGHT SELTZER LEMONADE

SCOR Demo

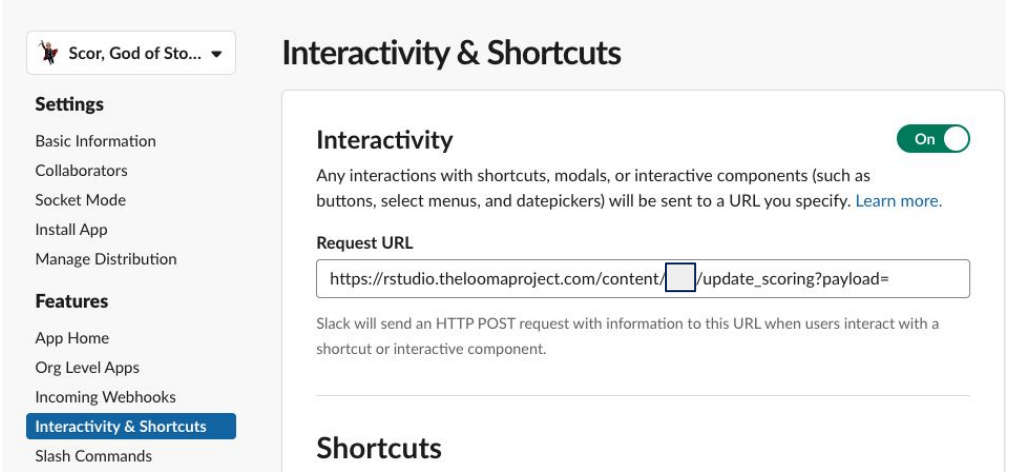
Agenda

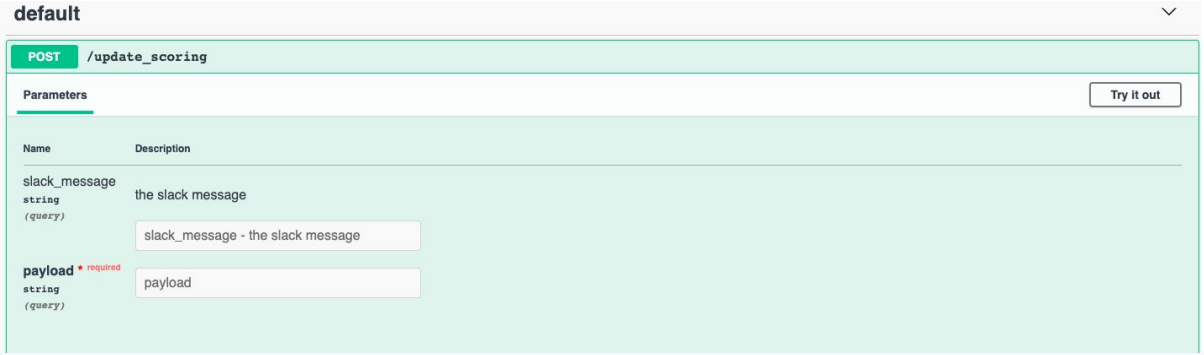
1. **Querying External APIs**
   - Exploring, Formatting and Organizing Queries
   - Productionalizing API Extraction w/ Posit Connect
2. **Dynamic External API Posting w/ R**
   - Example: HEB Steward Selection Typeform
3. **Creating and Deploying APIs with Posit Connect**
   - Sharing Data
   - Ingesting Data

SCOR Infrastructure

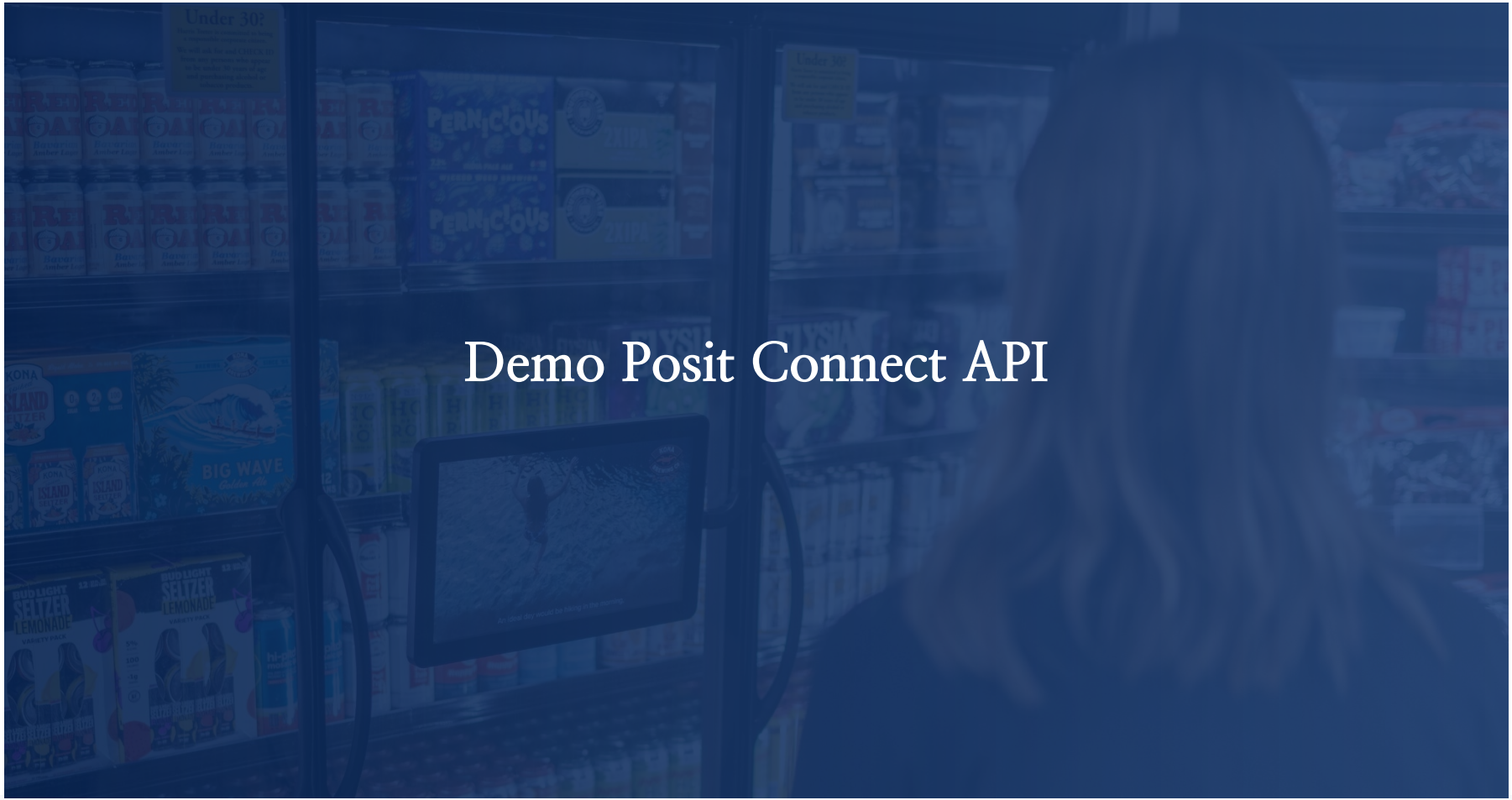


- Using Slack's Interactivity functionality and a Basic Plumber API, we can have Slack "talk back" to BigQuery.





SCOR Demo Pt 2



Demo Posit Connect API

Submit Your Location!

Terminal

curl -X POST "https://rstudio.theloomaproject.com/content/ff7097de-942f-44c0-ab29-c2a0cb9baa79/poll_submission?submission_name=**Davis**&submission_state=**NC**&submission_country=**US**" -H "accept: */*" -d ""

R

httr::POST(url = "https://rstudio.theloomaproject.com/content/ff7097de-942f-44c0-ab29-c2a0cb9baa79/poll_submission?submission_name=**Davis**&submission_state=**NC**&submission_country=**US**")

Terminal

```
curl -X GET "https://rstudio.theloomaproject.com/content/ff7097de-942f-44c0-ab29-c2a0cb9baa79/pizza_rankings" -H "accept: */*" -d ""
```

R

```
httr::GET(url = curl -X GET "https://rstudio.theloomaproject.com/content/ff7097de-942f-44c0-ab29-c2a0cb9baa79/pizza_rankings" -H "accept: */*" -d "")
```

Final Notes on Authentication

- **Never** hard code your API keys. Store them in external files or in your .Rprofile as global environment variables.
- For those that don't know, the httr packages has a default for you to put your authorization code in
    - Standard format is "Bearer <API KEY>"
- When creating and deploying APIs, be careful with public APIs.
- Posit Connect makes it really easy to have an authenticated API.
    - Change your API's privacy to "All Users - login required" and be careful about sharing API keys to external partners

Agenda

1. **Querying External APIs**
   - Exploring, Formatting and Organizing Queries
   - Productionalizing API Extraction w/ Posit Connect
2. **Functional API Request Generation w/ R**
   - Example: SCOR Typeform Surveys
3. **Creating and Deploying APIs with Posit Connect**
   - Sharing Data
   - Ingesting Data