Creating, deploying and consuming RESTful APIs in R

Diana PHOLO STONE

Predictive Insights

PREDICTIVE INSIGHTS

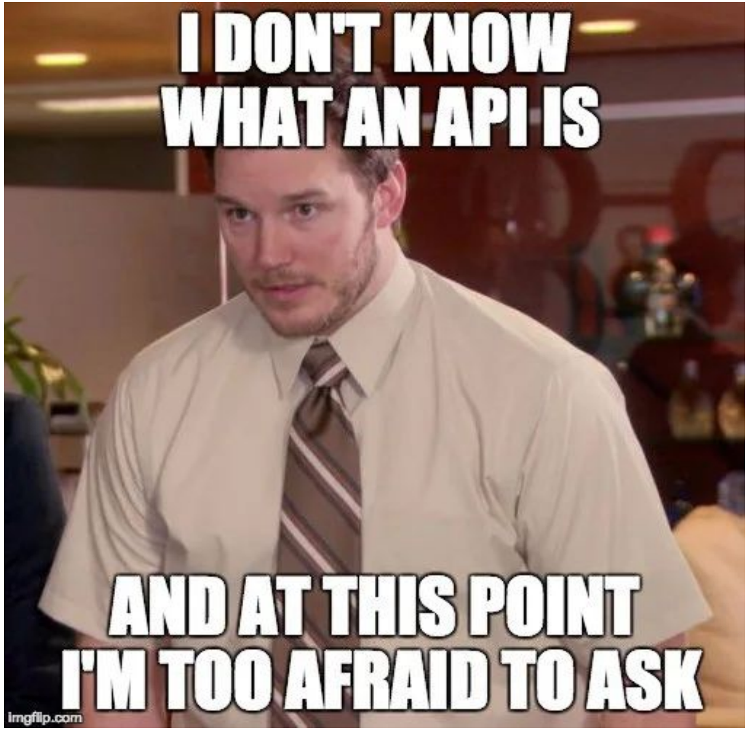
Introduction

Follow along

- https://github.com/anaidpm/plumber

What is an API?

- Acceptable answer


I DON'T KNOW WHAT AN API IS
AND AT THIS POINT I'M TOO AFRAID TO ASK
imgflip.com

What is an API?

- API = application programming interfaces
- = Messenger systems that allow communication between applications
- How it works:
  - API receives request from application
  - Sends request to server
  - Transmits response back

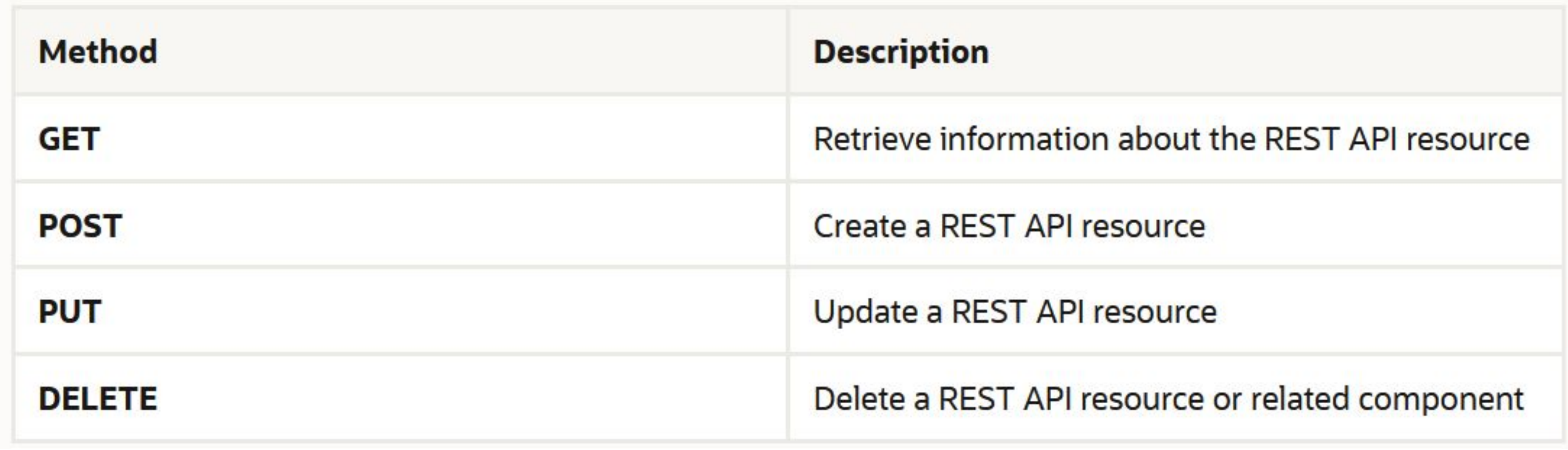
API requests

API requests have four components:

- Endpoint – part of URL
    - Example: endpoint of URL https://example.com/predict is /predict
- Headers – used for providing information (e.g. authentication credentials
- Body – info that is sent to server.
    - Used when not making GET requests.

API requests (cont.)

- Method – type of request you’re sending



Method	Description
GET	Retrieve information about the REST API resource
POST	Create a REST API resource
PUT	Update a REST API resource
DELETE	Delete a REST API resource or related component

To GET or to POST?

- **GET** requests info from specified resource.
    - should not be used for operations that cause side-effects
    - One reason: used arbitrarily by robots or crawlers
- **POST** submits data to be processed
    - e.g., from an HTML form
    - The data is included in body of request.
    - May result in creation of new resource or updates of existing resources

RESTful APIs

- REST = “Representational State Transfer”.
- Set of rules that developers follow when creating APIs.
- Most common rule: you should get a piece of data (response) whenever you make API request
- Most of the time, response returned by API is in JSON format.
    - Alternative formats: XML, images

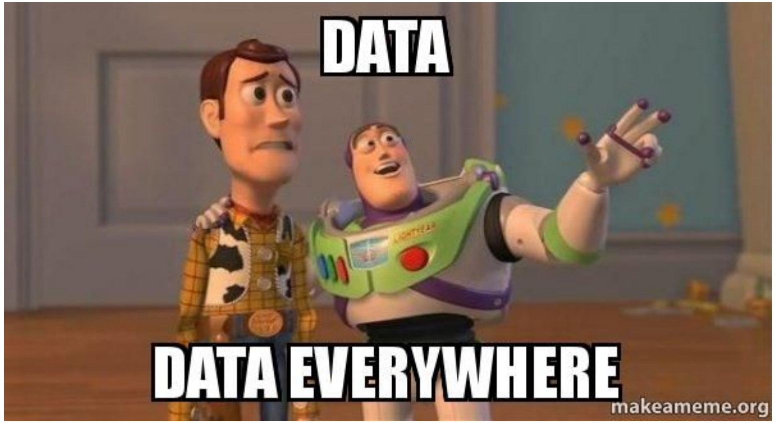
Why would I need an API?

- Share functionality with non-R people
- Use your own code in another application
- Protect your IP

Our Example: A demand forecasting API

Data-driven decision making

- Data helps business make **decisions**
  - Predicting sales trends
  - Reach new customers
  - Keep existing customers
  - Improve customer service
  - Direct marketing efforts
  - Understand social media impact



Demand Forecasting & Planning

- For products and services
- = Knowing who is going to visit your store, when, and for what.
- Allows for better customer experience + business efficiency

Let’s get our hands dirty!

PREDICTIVE
INSIGHTS

Steps

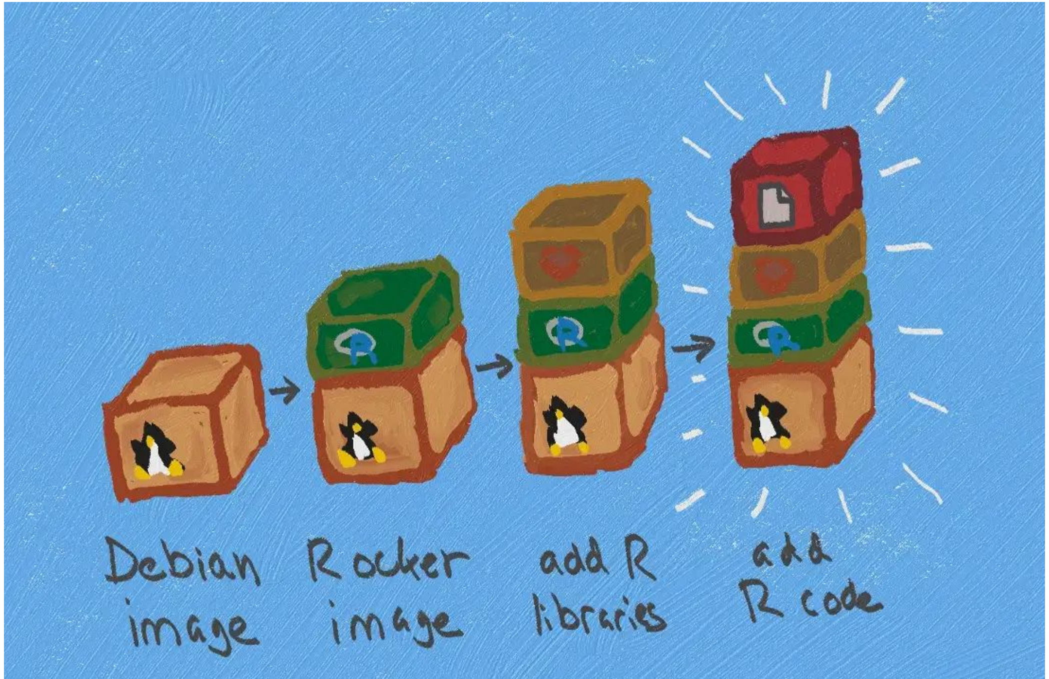
- Collect data
- Pre-process data
- Create and save model
- Create an API that exposes the model for use

Deploying

Docker

- Docker is a platform that allows you to run processes in isolated environment
- environment emulates Linux environment

Docker (cont.)



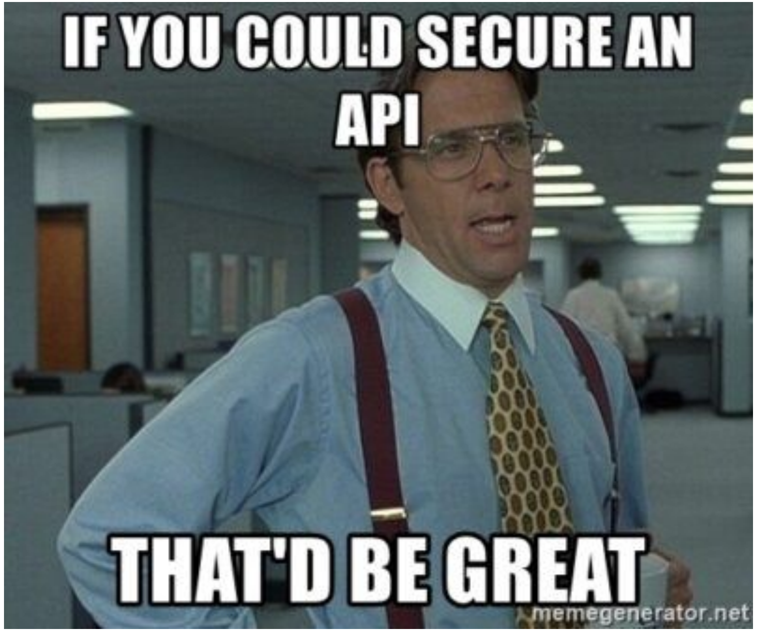
Docker on the cloud

- Deploying a prediction service with Plumber
- https://cran.r-project.org/web/packages/AzureContainers/vignettes/vig01_plumber_deploy.html

Security

Security

- Majority of R programmers are not trained make secure apps
- But APIs exposed on network require security

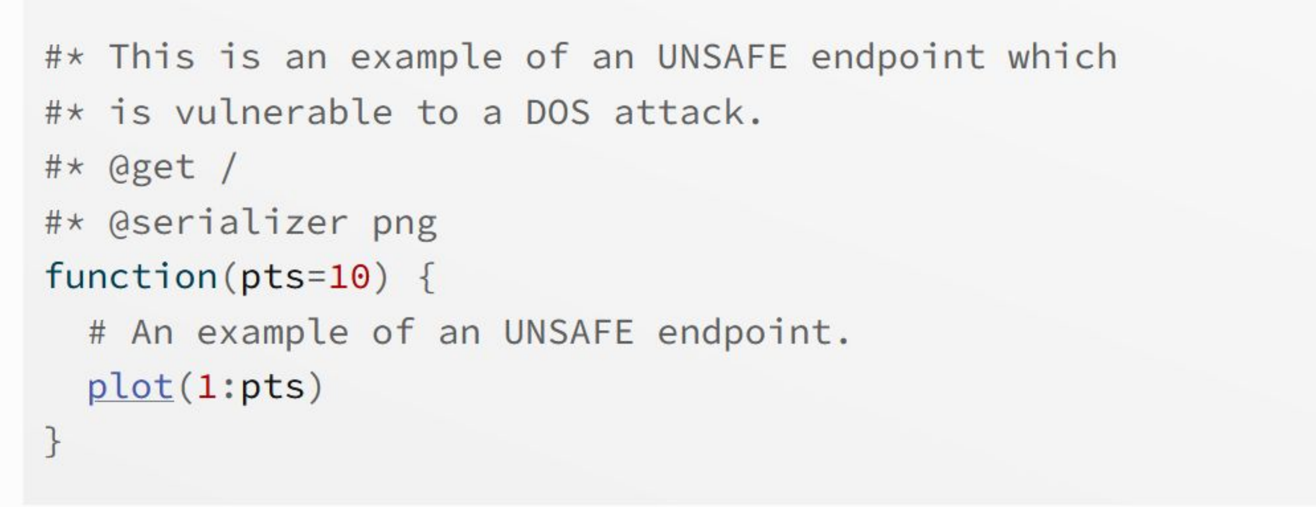


Denial Of Service (DoS)

- DoS attacks temporarily shut down server or service by overwhelming it with traffic
  - Can be unintentional

Denial Of Service (DoS)

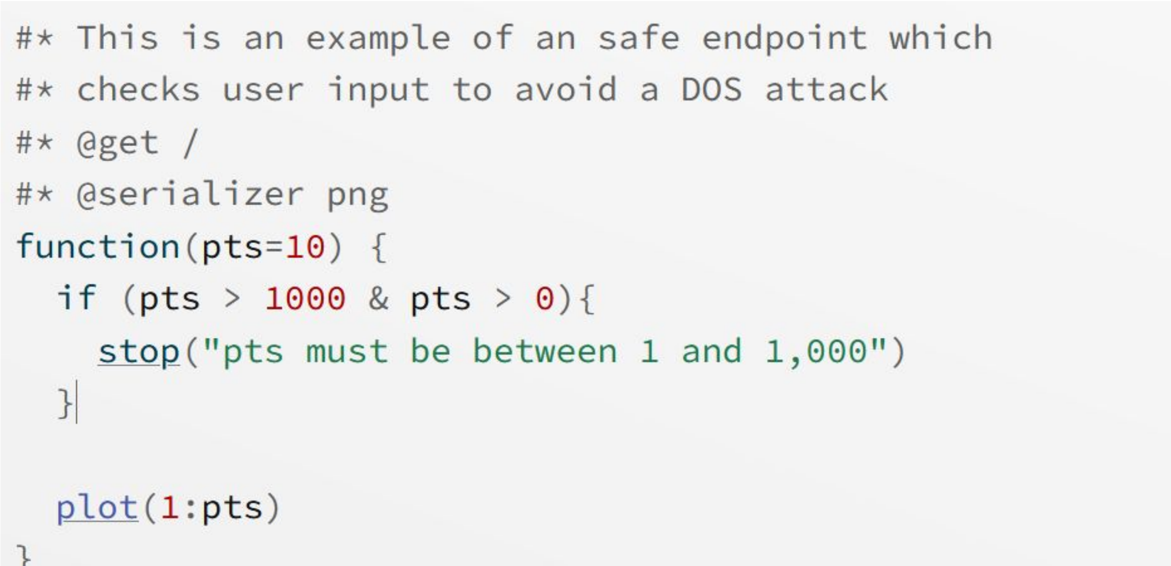
- Example unsafe API endpoint



```r
#* This is an example of an UNSAFE endpoint which
#* is vulnerable to a DOS attack.
#* @get /
#* @serializer png
function(pts=10) {
  # An example of an UNSAFE endpoint.
  plot(1:pts)
}
```

Denial Of Service (DoS)

- Example safe API endpoint

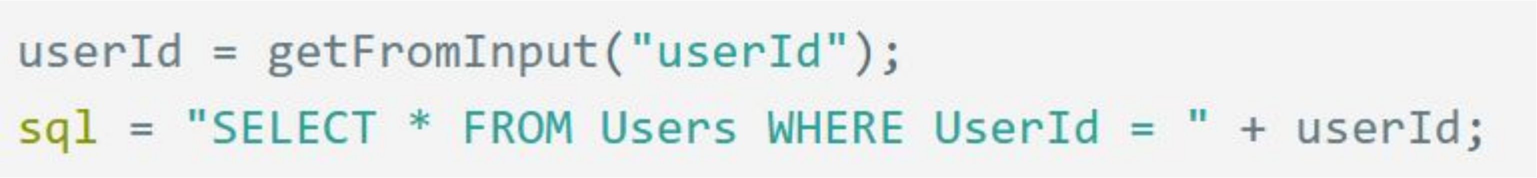


```
#* This is an example of an safe endpoint which
#* checks user input to avoid a DOS attack
#* @get /
#* @serializer png
function(pts=10) {
  if (pts > 1000 & pts > 0){
    stop("pts must be between 1 and 1,000")
  }

  plot(1:pts)
}
```

Sanitization & Injection

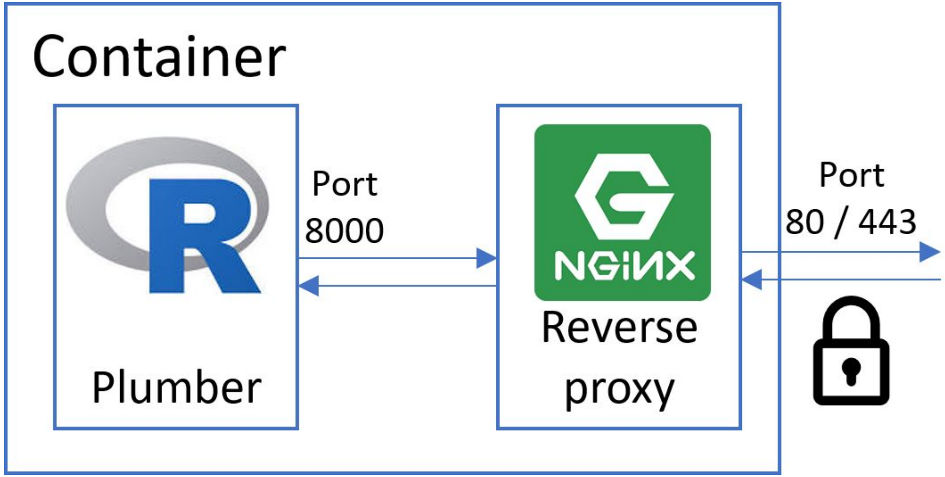
- SQL injection = most common form of data sanitization attack
- attacker is able to query or modify DB



```
userId = getFromInput("userId");
sql = "SELECT * FROM Users WHERE UserId = " + userId;
```

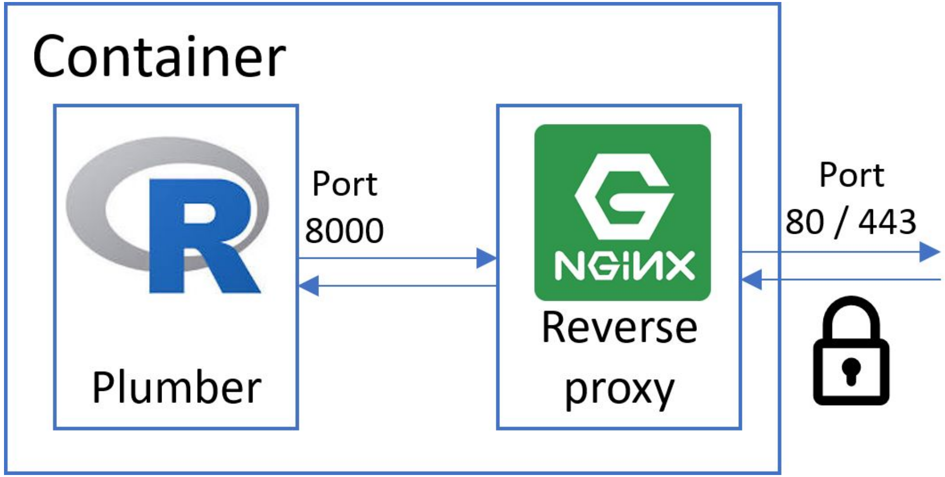
Using reverse proxy

- Simplest solution: running NGINX reverse proxy alongside R application.
- NGINX handles simple username/password authentication



Using reverse proxy (cont.)

- The plumber API listens on port 8000
    - not publicly available
- NGINX http server listens on port 80 and routes traffic to API



Resources

- Dockerized Plumber with NGINX
  - https://qunis.de/how-to-make-a-dockerized-plumber-api-secure-with-ssl-and-basic-authentication/
- Accessing REST API using R Programming
  - https://cran.r-project.org/web/packages/httr/vignettes/quickstart.html
  - https://www.geeksforgeeks.org/accessing-rest-api-using-r-programming/

Resources

- Plumber security
  - https://www.rplumber.io/articles/security.html
- Using docker to deploy an R plumber API
  - https://medium.com/tmobile-tech/using-docker-to-deploy-an-r-plumber-api-863ccf91516d

Questions?

E-mail: hello@predictiveinsights.net