



# Docker for R users

# My dream as a young data scientist:

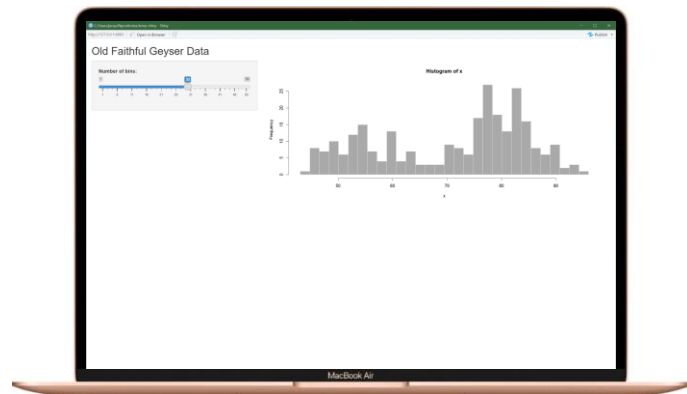
That I could write code and deploy it  
for other people to use

**THIS HAS BEEN HARDER THAN I WOULD HAVE EXPECTED**

(Spoiler: Docker makes this easier)

Maybe you want to host:

- **A Shiny dashboard** other people can see in their browser without installing R
- **An Plumber API** the engineers can pass data to without editing the code themselves
- **An arbitrary script** that runs on a fixed schedule



Could be publicly for everyone, internally within your company, as part of your companies product, or somewhere else

(Different from “I want coworkers to run my R scripts”)

**Making your code run somewhere else**

# Option 0: Use a managed service

Pay for a product like **Posit Connect** (or Azure ML!) that can host these things for you



## Pros:

- Easy UI for the developer and the end user!
- Manages permissions and other complexities

## Cons:

- \$\$\$
- May have # of user limitations
- May have concurrent user issues
- Requires engineers to set up and maintain
- Cannot give access to public

# Option 1: Run it on a computer you own

This is still a server! Your R code can run here!

You can host a shiny/plumber app on it by opening port 80 on your firewall and setting up port forwarding on your router! 🤖

```
shiny::runApp('app.R', port=80L, host='0.0.0.0')
```



TECHNICALLY A SERVER

## Pros:

- Feels like regular coding
- It's like, right there

## Cons:

- Power outages
- Your IT department HATES this

# Option 2: Rent a computer from a cloud provider

## GOOGLE CLOUD PLATFORM VMs

### Compute Engine

Secure and customizable compute service that lets you create and run virtual machines on Google's infrastructure.

New customers get \$300 in free credits to spend on Google Cloud. All customers get a general purpose machine (e2-micro instance) per month for free, not charged against your credits.

[Try Compute Engine free](#)

[Contact sales](#)

- ✓ [Predefined machine types](#): Start running quickly with pre-built and ready-to-go configurations
- ✓ [Custom machine types](#): Create VMs with optimal amounts of vCPU and memory, while balancing cost
- ✓ [Spot machines](#): Reduce computing costs by up to 91%.
- ✓ [Confidential computing](#): Encrypt your most sensitive data while it's being processed
- ✓ [Rightsizing recommendations](#): Optimize resource utilization with automatic recommendations

## AMAZON WEB SERVICES EC2

### Amazon EC2

Secure and resizable compute capacity for virtually any workload

[Get Started with Amazon EC2](#)

[Connect with an Amazon EC2 specialist](#)

**750 hours per month**  
for 12 months with the [AWS Free Tier](#)

Access reliable, scalable infrastructure on demand. Scale capacity within minutes with SLA commitment of 99.99% availability.

Provide secure compute for your applications. Security is built into the foundation of Amazon EC2 with the AWS Nitro System.

## AZURE VMs

Explore Azure Virtual Machines and the cloud with your Azure free account

Deploy and monitor virtual machines (VMs) using 12 months of free services

[Start free](#)

[Pay as you go >](#)

Popular services free for 12 months

40+ other services free always

+

Start with \$200 Azure credit

You'll have 30 days to use it—in addition to free services.

All cloud providers let you rent a virtual machine by the hour!

### Pros:

- Still works like a regular computer
- Always on

### Cons:

- Expensive to run all the time
- Install libraries, packages, and keep track of every machine

# Option 3?

What if we had a way to:

- Keep track of each step used to set up a server
- Take a snapshot after each step to neatly keep track
- Run any of these snapshots very efficiently
- In a way that any engineering team could understand



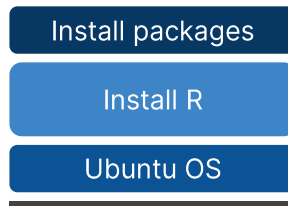
# Option 3: Containers.



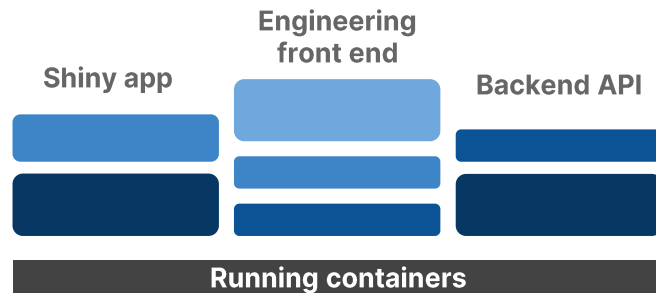
# Containers: basic idea

- A Docker **image** is a snapshot of a computer that you can run
  - Images are built on top of other images
- A **Dockerfile** specifies how to build the image
  - Start with a base image
  - Specify other libraries and packages to install
  - State the command that should run at start
- A **container** is a running image
  - You can have many containers at once

(These days people say “containers” instead of “Docker”)



Build a new image on top of other ones



A lot of different containers can be running at once

## Pros:

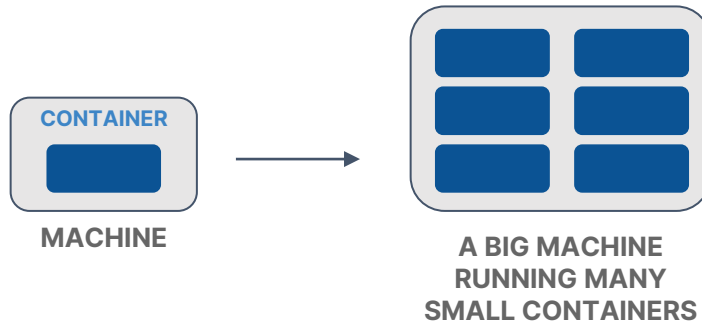
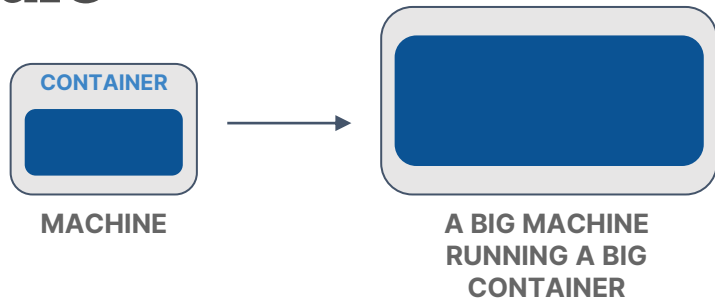
- A universal language for engineers
- Can be saved forever and will still run
- Don't have to worry about what's on the computer running them

## Cons:

- You have to get used to them
- Ephemeral –when they turn off you lose saved info

# Containers can easily scale

- Need more CPU/RAM?  
Run a container on a larger machine.
- Need to handle lots of requests or parallelize your code? Run many of the same containers at once.



# Example: make a container run a Shiny app

## Three files

```
app.R (shiny app)  
main.R (has the runApp command)  
setup.R (install.packages)
```

# Example Dockerfile

This is an example of a Dockerfile. Each Docker command adds new layer, meaning it will create a new image on top of the previous one.

You can build the image (let's call it shiny-docker) from the Dockerfile using `docker build -t shiny-docker .`

```
FROM rocker/tidyverse:4.4.1
```

```
COPY /src /
```

```
RUN Rscript setup.R
```

```
ENTRYPOINT ["Rscript", "main.R"]
```

**FROM** indicates what the starting image is. All images start on top of another image. Here we are using a container with Tidyverse already installed. You can use other people's images as your starting point!

**COPY** will copy a file or folder from the location of the Dockerfile into the Docker image.

**RUN** indicates a Linux command to run. Here we are saying to run the setup R script which installs packages.

**ENTRYPOINT** specifies the program that should run when the container starts.

# Running a container

Once you have built an image, you can run it with a command like

```
docker run --rm -it -p 80:80 shiny-docker
```

We want to run a container

When the container is  
stopped, delete the attached  
temp volume (disk drive)

Let us directly interact with  
the container from the  
command line

Any traffic at port 80 to the  
machine running Docker  
should be directed to this  
container

The actual image to run

Then go to 127.0.0.1 in your browser to see the app

Live demo!

You just ran Shiny on your laptop.  
Why should I care?



**R isn't installed on my computer**



**Everything the Shiny app needs is in the image**

# The Shiny app not even running on Windows

## It's running in LINUX



# How can that possibly be?

- The container is totally isolated from the system running it
- It doesn't even have to be the same OS
- The image will stay the same forever



# Let's take this baby to the cloud



- If we can isolate a container on our laptop why not run that same container somewhere else
- **Google Cloud Run** is a great place to easily run containers
- Will turn off containers when not needed (spend pennies a month!)
- Will instantly spin up more containers if a lot of people are using it

# Containers on Google Cloud Run

- Upload an image to Google Artifact Registry via **docker push**
- Create a Google Cloud Run service that deploys the container
  - Choose the image to run
  - Select the max number concurrent containers, memory
  - Set up a schedule to start it on
- You can now have lots of concurrent containers running in the cloud



Shiny  
Container

Shiny  
Container

Shiny  
Container

Shiny  
Container

Live demo!

Try and break me  
(with every device you own at once)



<http://bit.ly/rdockerdemo>

# Is there anything a container can't do?

**It will not:** Be an object you can hold persistent data in  
(local data gets deleted when containers stopped)

**It will not:** Let you (reasonably) interact with the R code  
Very difficult to edit R code within a running container

**It will not:** let you easily pass code between humans  
Hassle to install Docker, files aren't small, can't easily see/edit the code

**It will not:** be a useful tool for running reproducible analyses  
This is a hill I will die on

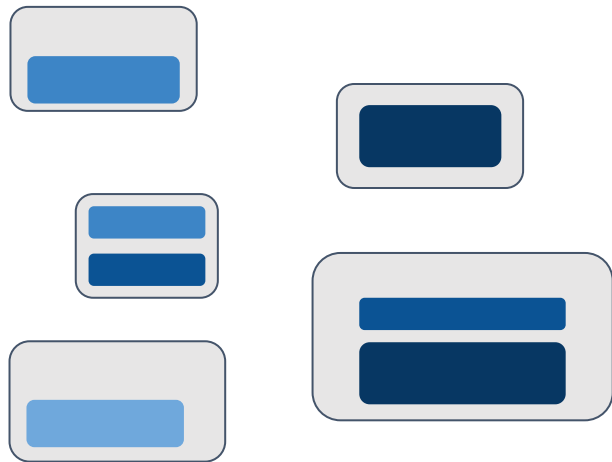


# Epilogue What the heck is Kubernetes?

- What if instead of Google Cloud Run we had a department of our company to handle this
- That team is DevOps and they use Kubernetes as a platform
- I don't think data scientists should have to deal with this



**kubernetes**



MANY MACHINES RUNNING MANY CONTAINERS

# Thank you!

Code available at: <https://github.com/jnolis/shiny-docker>