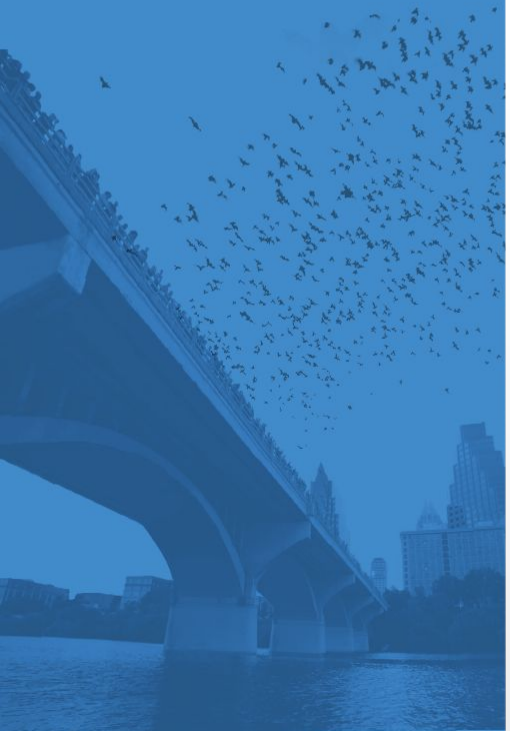

Shiny in Production
rstudio::conf
AUSTIN

Welcome



Sean Lopp

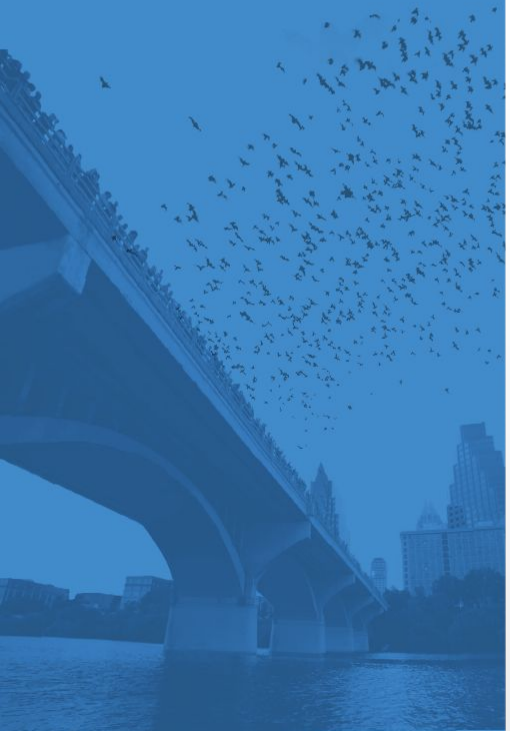
Kelly O'Briant

Tonya Filz

Mark Sellors

Devin Pastoor

Logistics



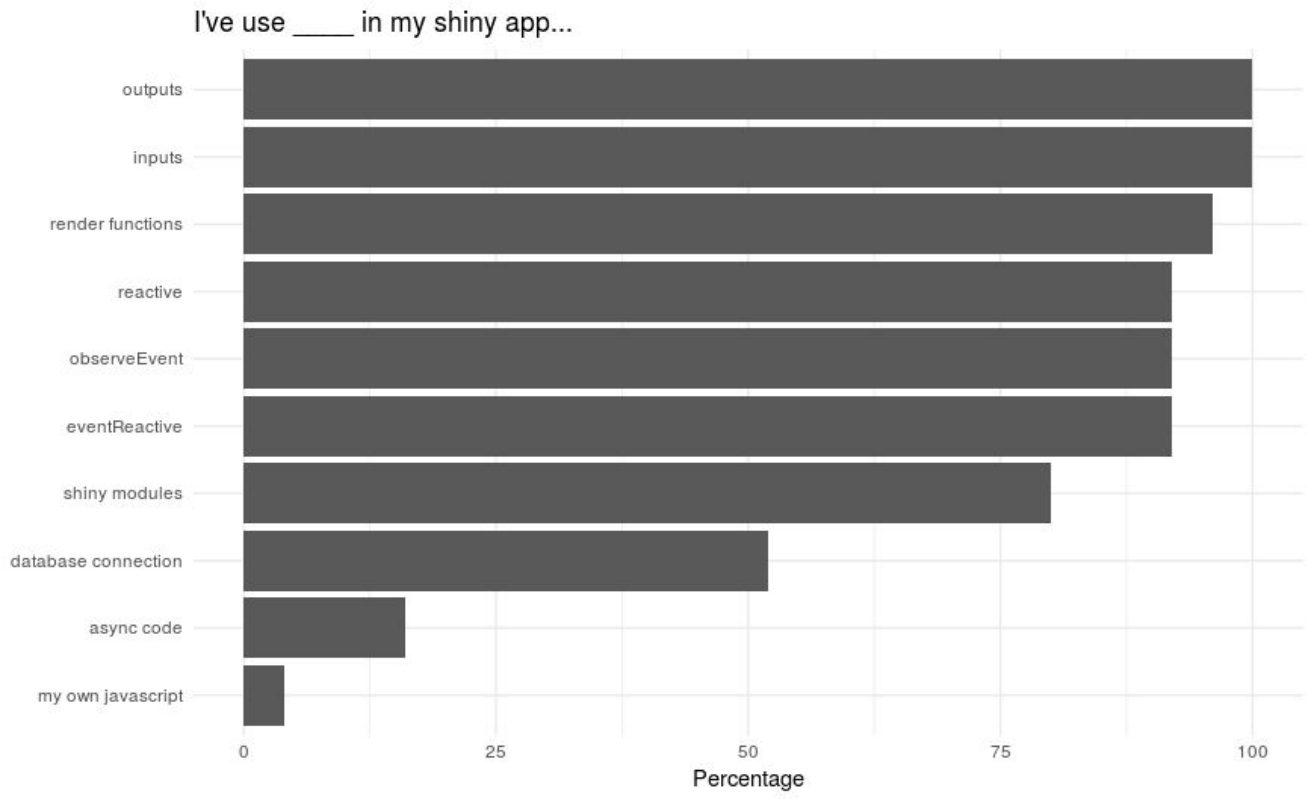
Bathrooms

Schedule

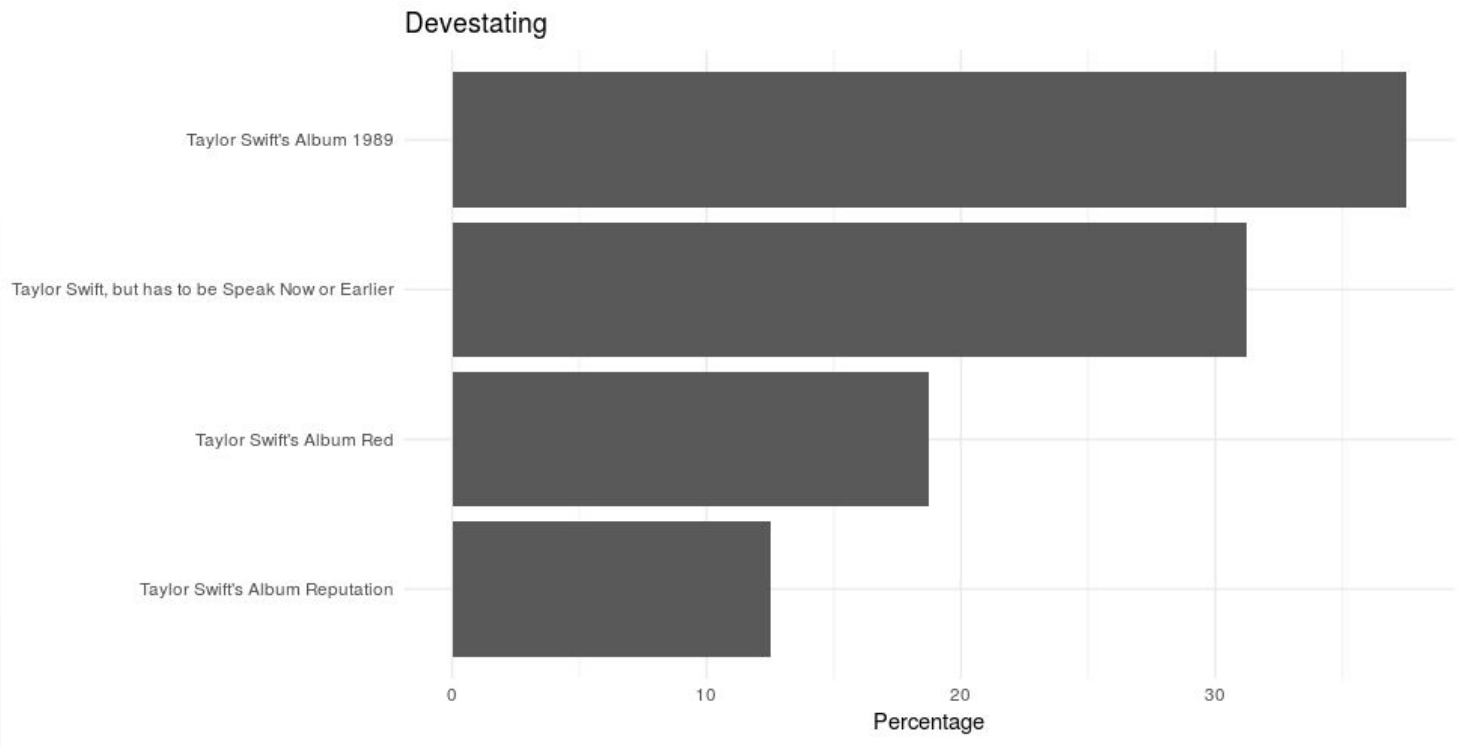
0800–0900 Registration/breakfast
0900–1030 Workshop
1030–1100 Break
1100–1230 Workshop
1230–1400 Lunch- 1.5 hours
1400–1530 Workshop
1530–1600 Break
1600–1730 Workshop

Resources

Survey Results



Survey Results



Meet & Greet 15 Minutes

First: Create small groups of 4-6 people

Discussion:

Introduce yourself to the group

- What is your name?
- How do you use R?
- What do you hope to achieve in this workshop?

Deliverable: *Verify Classroom Access*

Everyone in the group should visit:
https://...

Once you have a server address; visit
http://<your-server>/rstudio
UID: rstudio
PWD: rstudio

Objectives

Running Shiny in Production

- Is Shiny ready for production?
- What does it take to get there?
- What do you need to know?
- What tools are available?
- How do you develop a workflow?

Meet our App

- Awesome App Origin Story

TL;DR:

- We’ve developed a nice app
- We want to put it in production
- We want confidence that it will perform well in production, both now and in the future

App Exploration 15 Minutes

First: Open app.R and Run the Application

Discussion: *Explore the Application*

- Are there any parts of the app code that don't make sense?
- Is this application ready for production?
- How would you define "production"?
- What insights would be useful to have about the application before we try to deploy it?

Deliverable: *Start a Plan*

Create a checklist

Outline the steps you might take to put this application into production

Checklist

- [ ] Tests
- [ ] Performance Optimization
- [ ] Environment (Packages) Management
- [ ] Data Access
- [ ] Deployment Hand Off
- [ ] Scaling
- [ ] Monitoring

shinytest

Testing

Expectation Based Testing (Units Tests)

```
trim_whitespace <- function(characters) {
     for (c in characters) {
         if (c in whitespaces) {remove(c)}
     }
}

expect_that(trim_whitespace("the test"), "thetest")
```

Testing

Expectation Based Testing (Units Tests)

```
trim_whitespace <- function(characters) {
     for (c in characters) {
         if (c in whitespaces) {remove(c)}
     }
}

expect_that(trim_whitespace("the test"), "thetest")
```

+

Why is this type of testing hard for Shiny apps?

Testing

Snapshot based testing

1. Take a snapshot

2. Make changes to your code

3. Compare new results to the approved snapshot

Shiny Test Demo

shinytest 20 Minutes

First: Open app.R and use “Record Test”

Discussion:

Understanding shinytest

- What does the recording file create?
- What challenges might our app pose to testing?
- Can you run the tests and get a success?
- How might you automate tests?

Deliverable: *Try shinytest*

A set of tests that can catch unintentional changes to our app.

shinytest: answers

Discuss the need for a random seed

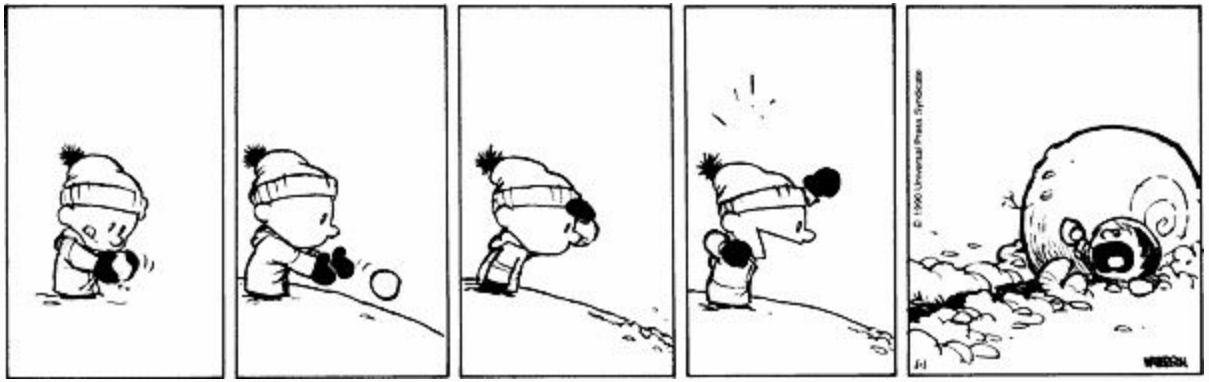
Checkout `answers/tests`

Break

Come back at 11

Profiling

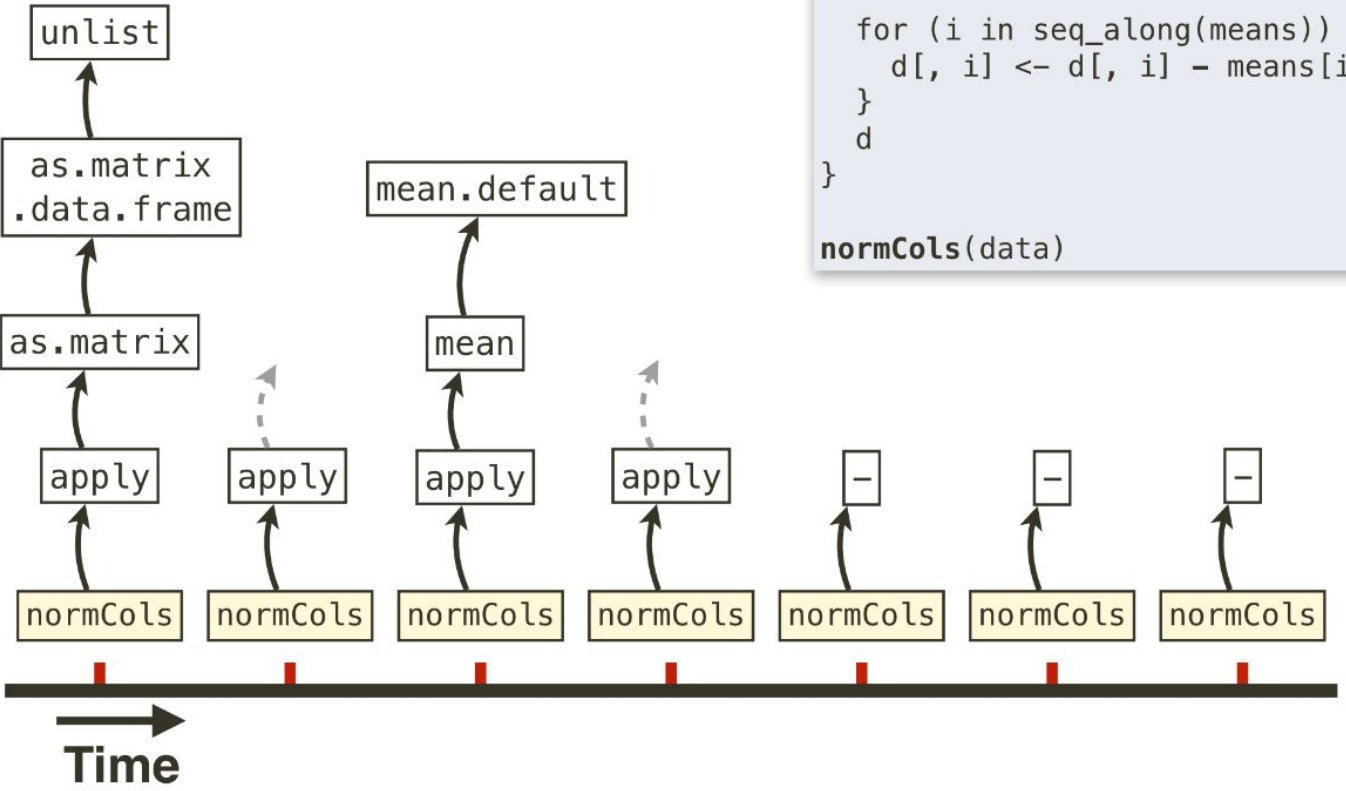
The Most Important Tool



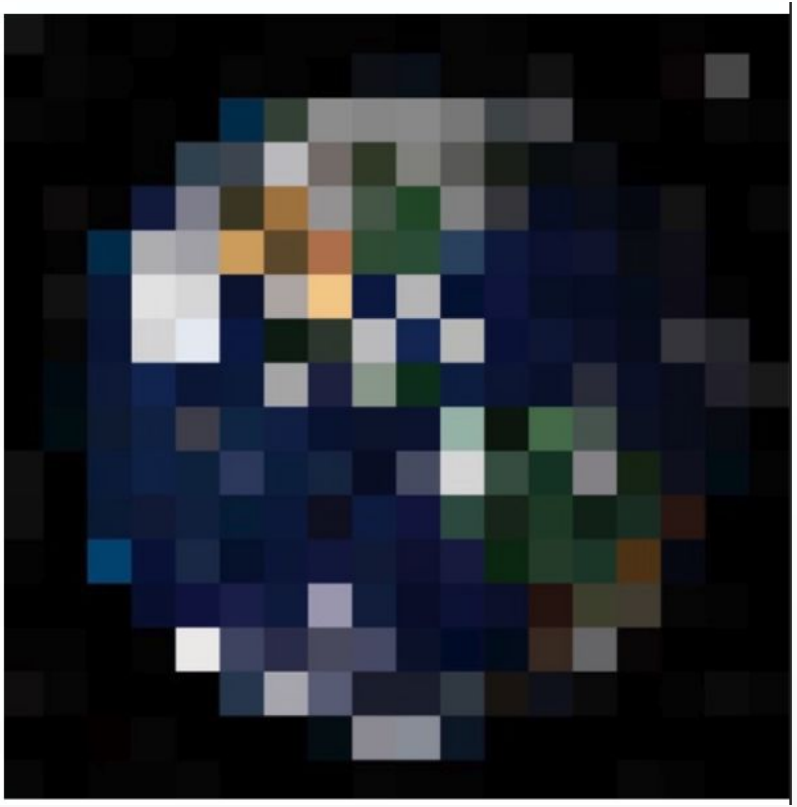
The Most Important **Tool**

Sampling profiler

```
normCols <- function(d) {
  means <- apply(d, 2, mean)

  for (i in seq_along(means)) {
    d[, i] <- d[, i] - means[i]
  }
  d
}

normCols(data)
```



Profiling



profvis + shiny

```
profvis::profvis({
  shiny::runApp()
  # interact with app
  # close app
})
```

profiling 20 Minutes

First: Create a profile of our app

Discussion:

Understanding app performance

- Where does our app spend most of its time?
- Do any parts of the profile surprise you?
- If you test the app again, do you get the same results?

Deliverable: *Optimization Recommendations*

One recommendation for how we could speed up our app.

profvis + shiny: answers

`answers/app-profile.Rprofvis`

Slowest thing is the lime explanation model & plotting

`renderCachePlot` will come to the rescue!

Other Q&A to discuss:

- Is the call to gt slow? (Answer: no, it is really fast!)
- Why does getStudentNum take so much longer than getStudentBin? How could we speed this up?

Lunch

Come back at 2:00

rstudio::conf AUSTIN

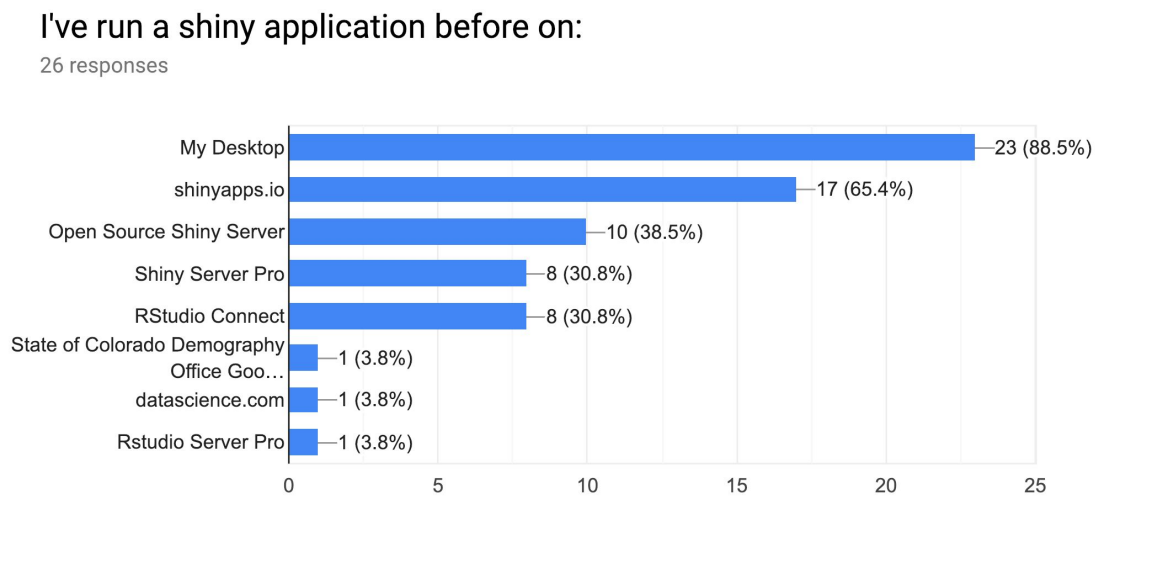
Deployment

Deployment

Game Plan

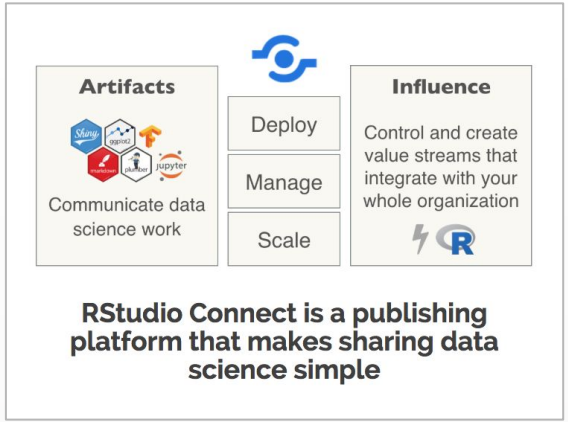
Do a little bit of deployment now.

Talk about scaling, deployment architectures, and other DevOps things tomorrow.



Briefly: RStudio Connect & Why We Built It

Helping data scientists communicate their work is of utmost importance.



It doesn't matter how great your analysis is unless you can explain it to others: you need to communicate your results.

- Grolemund & Wickham in R For Data Science

Deploy 15 Minutes

First: Login to RStudio Connect

http://<your-server/rsconnect
Username: rstudio Password: rstudio

Discussion:

Pre-deployment Brainstorm

- What is our goal in deploying this code?
- What does our code depend on locally?
- What needs to deploy with our code?

Deliverable: *Deployed App*

Press the publish button in RStudio: 

- Link to your Connect account
- Select the files to publish (do **NOT** use the default to publish *all* the files)
- Poke around at the results

Note: We are expecting an error! That is ok

```
password authentication failed for user
"rstudioadmin"
```

Deploy: answers

- [ ] Supporting Files (`use_models.R, *.RDS`)
- [ ] Database Connection 
    - [ ] Database Driver
    - [ ] Configuration (Credentials, DSN, `config.yml`, etc)
- [ ] R Version
- [ ] R Packages
- [ ] ???

Databases & config

```
library(odbc)
library(DBI)
library(config)
db <- get('database')

pool <- dbConnect(
  odbc::odbc(),
  Driver   =  db$Driver,
  Server   =  db$Server,
  Database =  db$Database,
  UID      =  db$UID,
  PWD      =  db$PWD,
  Port     =  db$Port
)
```

```
default:
  database:
       Driver: PostgreSQL
       Server:
'sol-eng-shiny-prod-class.cih
ykudhzbgw.us-west-2.rds.amazo
naws.com'
       Database:  'students'
       UID:  'rstudioadmin'
       PWD:  'rstudioadmin'
       Port:  5432
```

Databases 25 Minutes

First: Read https://db.rstudio.com/best-practices/portable-code/

Discussion:

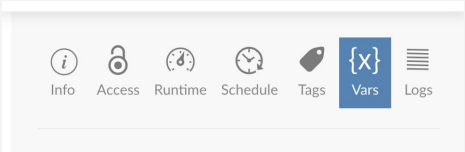
Data Management

- How does your organization connect to data?
- What data are we exposing to viewers? Is it appropriate for all viewers?
- What else could we manage with the config package?

Deliverable: *Running, Deployed App*

Update `config.yml` and Redeploy

Edit the Environment Variables in RStudio Connect



Refresh and 🎉

Deploy Checklist

- ☒ Supporting Files (`use_models.R, *.RDS`)
- ☒ Database Connection
  - ☒ Database Driver
  - ☒ Configuration (Credentials, DSN, `config.yml`, etc)
  - ❓ Think about data sensitivity & viewership

- ☐ R Version
- ☐ R Packages

packrat

```
rsconnect::writeManifest(
     appDir = ".",
          appFiles = c("app.R", "config.yml", "model.RDS", "use_models.R",
                       "model_explainer.RDS", "data_preprocessor.RDS",
                       "www/rstudio.png")
)
```

Break

Come back at 4:00

Joe Cheng War Stories & QA

rstudio::conf AUSTIN

Wrap Up

First: Brainstorm how you could apply what we've learned today

Discussion:

- Share your brainstorming ideas

Deliverable A:

- Write down 3 concerns or questions you have for tomorrow.

Deliverable B:

- Do you want to present your own 5 minute "war story" tomorrow morning? Come talk to us.


Day 2
rstudio::conf
AUSTIN

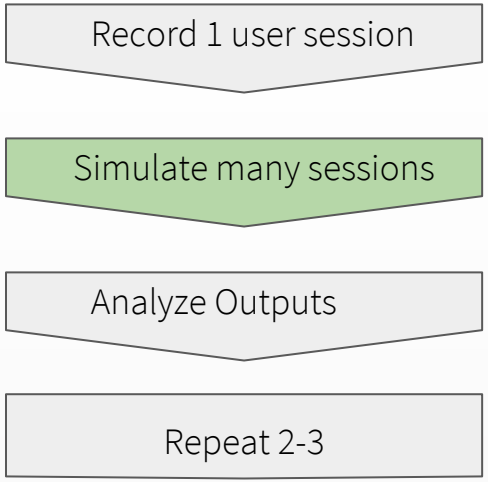
War Stories

Load Testing

shinyloadtest

shinyloadtest

shinyloadtest



This part is done outside of R

shinyloadtest 20 Minutes

First: Open `runloadtest.R`, do the pre-run checklist

Discussion:

Preparing for the load test

- First, what is up with the pre-run checklist? Any idea why these steps are (currently) necessary?
- The `runloadtest.R` file is going to start with a baseline test of 1 and then a test of 25. Why is the baseline important?

Deliverable: *Run the load test*

Follow the first set of commands in `runloadtest.R`

How did the experience for 1 user compare to the experience for 25 users?

shinyloadtest: answers

What is up with our pre-run checklist?

```
# [] In use_models.R, un-comment the set.seed call on line 52
# [] Redeploy your app to RStudio Connect
# [] Make sure your app's access is set to "Anyone - no
login"
# [] Log out of RStudio Connect
```

Recall: set.seed

Bug being fixed

Compare the two runs. How did the experience for the 25 users compare to the 1 user?

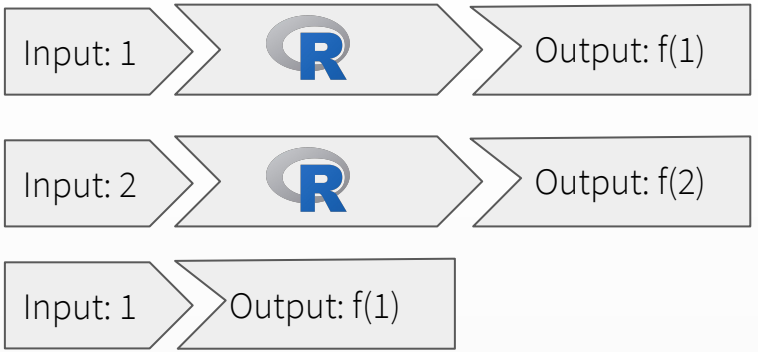
```
loadtest <- load_runs(
   baseline = “answers/baseline”,
   twentyfive = “answers/twentyfive”
)

shinyloadtest_report(loadtest)
```

Caching

renderCachedPlot - what?

Without:

With:



renderCachedPlot - how?

```
shinyOptions(cache = "./cache-dir")
output$plot <- renderCachedPlot({
      plot(...)

}, cacheKeyExpr = list(input$student))
```

*specific to our example, normally use the object that represents a set of inputs that should generate a new output

renderCachedPlot 30 Minutes

First: Update your app code to use `renderCachedPlot`

Discussion:

caching + shinyloadtest

- How will introducing `renderCachedPlot` affect our load test experience?

Deliverable: *Re-run Load Test*

Redeploy the version of the app with `renderCachedPlot`

Re-run the load test and compare the outputs (continue to follow along with `runloadtest.R`)

renderCachePlot: answers

answers/cached-app.R

```
loadtest <- load_runs(
   baseline = “answers/baseline”,
   twentyfive = “answers/twentyfive”,
   cached = “answers/cached”
)

shinyloadtest_report(loadtest)d
```

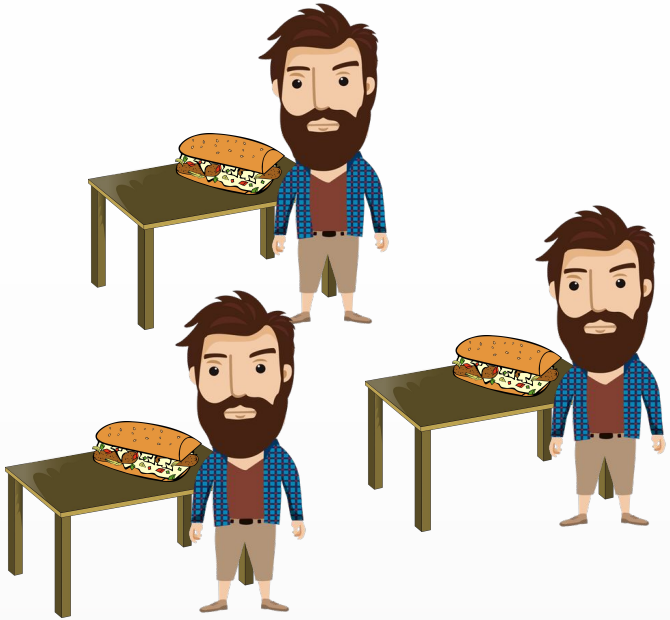
Break

Come back at 11

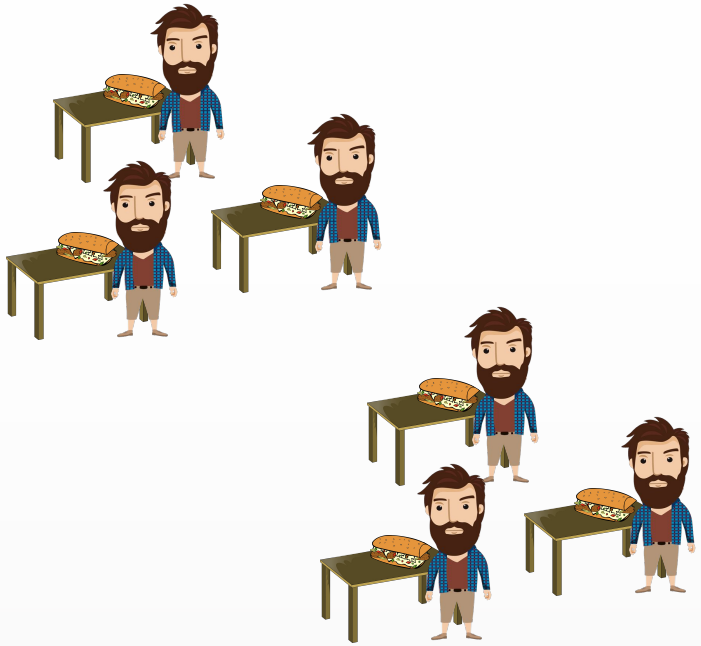
Scaling

Scaling 101

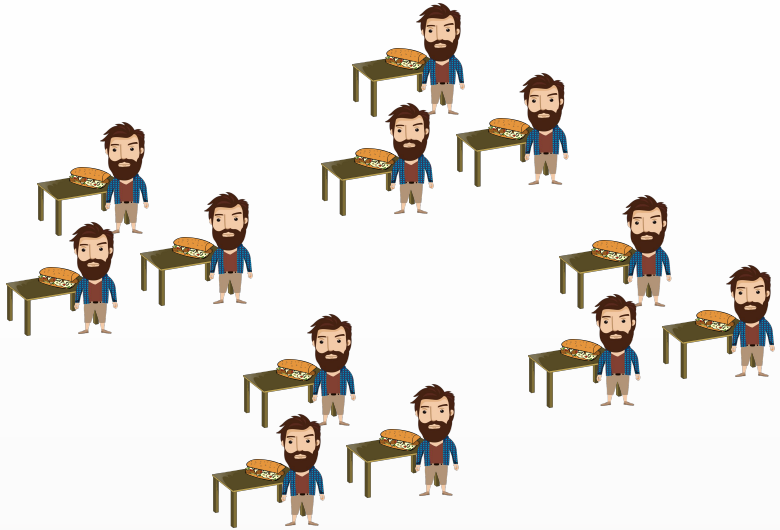
Scaling 101



Scaling 101



Scaling 101



Scaling 15 Minutes

First: Open the runtime settings for your app

Discussion:

Performance Tuning

- What do each of the settings mean?
- What changes to the settings might affect our app?

Deliverable: *Re-run Load Test*

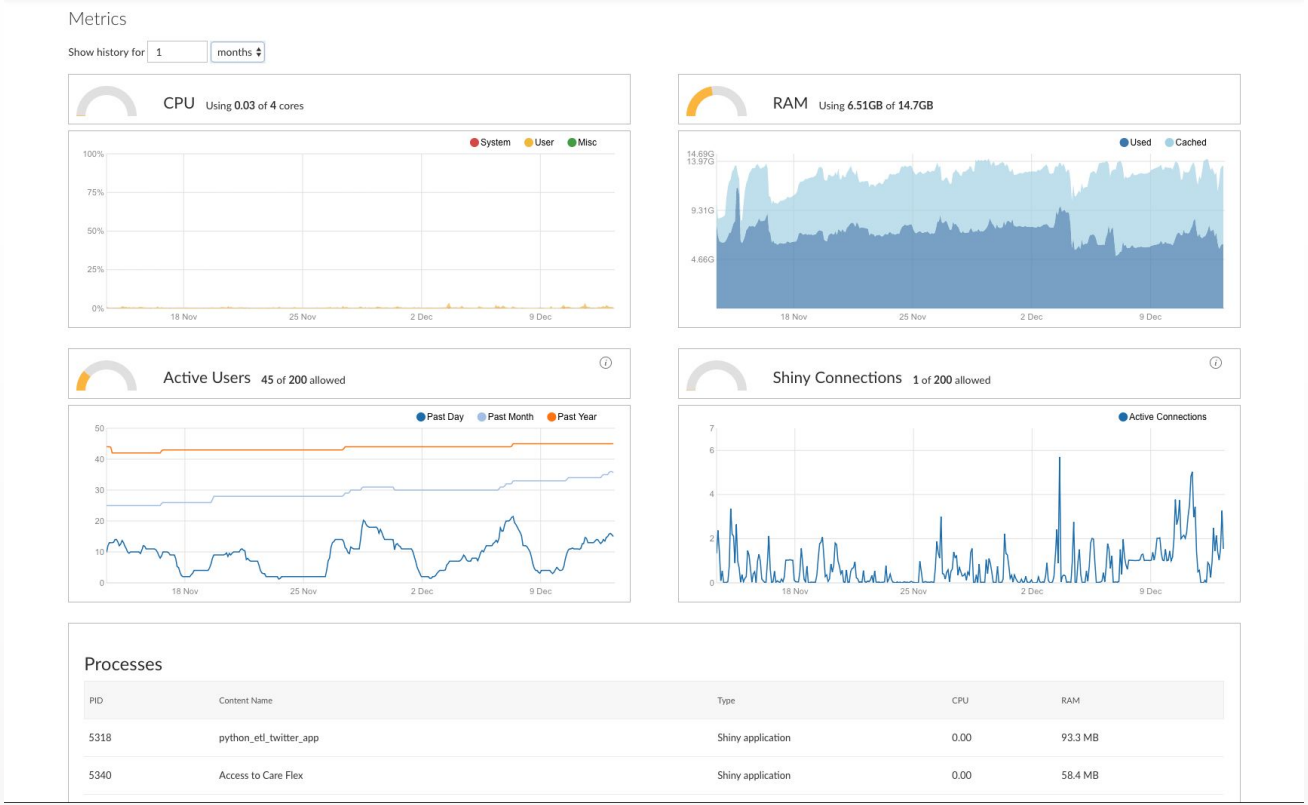
Test your hypothesis by changing the runtime settings and then re-run the load test.

Compare the results to the prior test(continue to follow along with `runloadtest.R`)

Scaling: answers

```
loadtest <- load_runs(
   baseline = “answers/baseline”,
   twentyfive = “answers/twentyfive”,
   cached = “answers/cached”,
   scaled = “answers/scaled”
)

shinyloadtest_report(loadtest)
```

Sharing is Caring



Sharing 10 Minutes

First: Open the admin dashboard in RStudio Connect

Discussion:

Multi-Tenant Environment

- What are the tradeoffs for modifying the scaling settings?
- What else might you need to consider when sharing infrastructure?
- How does these concerns change over time?

Deliverable: *Update checklist*

Create one pre-deployment check and one every-2-months check based on your discussion.

Sharing is Caring: answers

Pre-Deployment:

[ ] What other content is running on the server? Do I have enough resources for my app?

Every 2 Months:

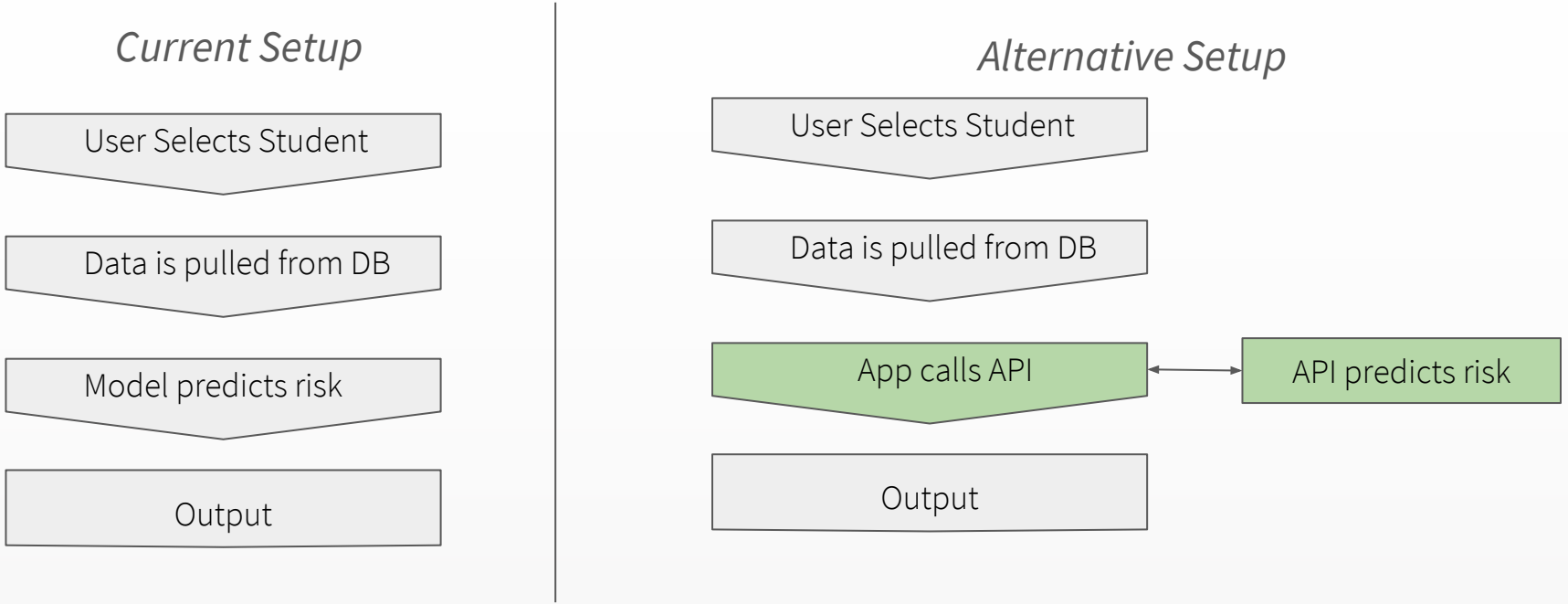
[ ] Check logs to ensure server resource consumption is reasonable. Can I automate this?

Alternatives to Shiny

Review survey results for plumber and R Markdown familiarity

Plumber

Could our student "scoring" be hosted outside of the Shiny app?



Plumber 15 Minutes

First: Try creating your own plumber API in the IDE (New File)

Discussion:

Consider our new architecture

- Why might a RESTful API be easier to scale than a Shiny app?
- What benefits come from pulling the modeling code out of our app?

Deliverable:

Add an item to our production checklist to consider plumber.

Add 2 decision criteria to the checklist to decide when a plumber API would be useful

Plumber: answers

Why might a RESTful API be easier to scale than a Shiny app?

Requests are stateless, so it is easier for R processes to come and go

What benefits come from pulling out our modelling code?

We can update the model independently of the app, e.g. if we wanted to retrain.

Decision Criteria:

Does my R functionality need to be accessed by other systems?

R Markdown

Scheduled data updates, email distribution, and client side interactivity.

Shiny App

Browser and R are connected!

HTML gets rendered. Time passes. Then the server gives the HTML to the browser.

R Markdown 15 Minutes

First: Create your own R markdown document (New File)

Discussion:

Consider R Markdown

- How does an R Markdown document scale?
- What types of interactivity can be added to R Markdown?

Deliverable: *Create a decision matrix comparing Shiny to R Markdown*

Alternatives *Criteria*	Shiny	R Markdown
Criteria A		
Criteria B		
Criteria C		
Criteria D		

R Markdown: answers

Alternatives *Criteria*	Shiny	R Markdown
# of Inputs	Users need multiple levels of dependent inputs to get an answer	All inputs can be specified up front
Update Frequency	Updates need to happen in real time	Updates occur on a regular basis
Output	Users come to the system to do exploratory Q&A, the outputs are the results	Users want a copy of the results or want results distributed to their email
Interactivity	Interactivity depends on lots of data	Interactivity can be based on data sent to the browser

Lunch

Come back at 2:00

rstudio::conf AUSTIN

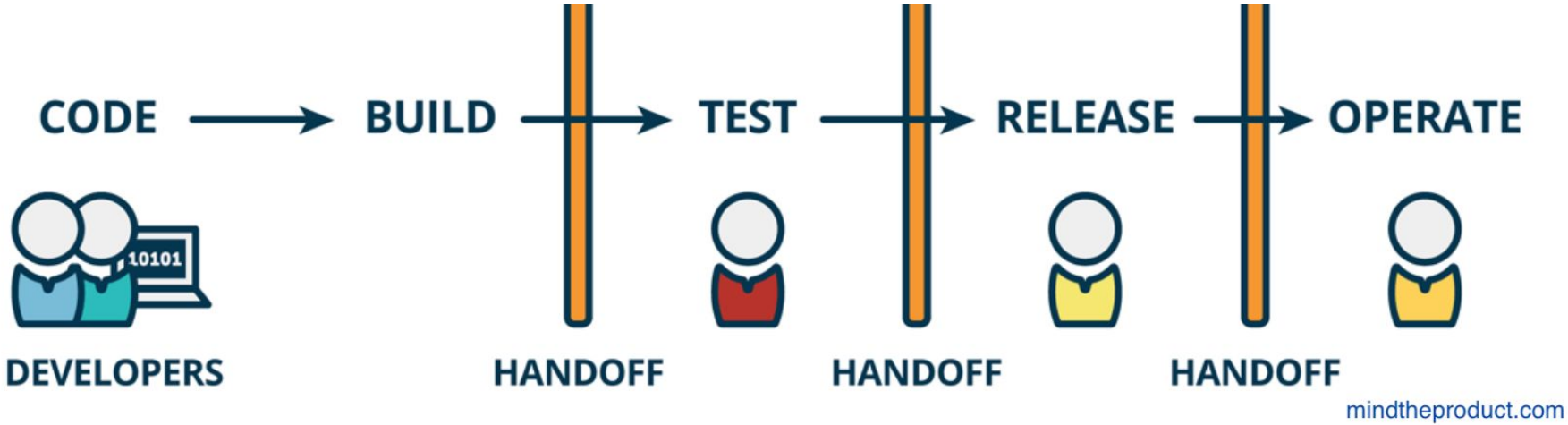
Adventures in Mixing Shiny Apps with IT Groups

DevOps 101

rstudio::conf AUSTIN

Software Developers Work with IT and so can YOU

Managing Code Deployment Handoffs



It's not "your job" to understand operations and systems administration.
It's not "their job" to understand R programming. SO HOW DO WE MAKE THIS LESS PAINFUL??

Steal Existing

Strategies for Managing Code Handoffs

& Define Shared Goals

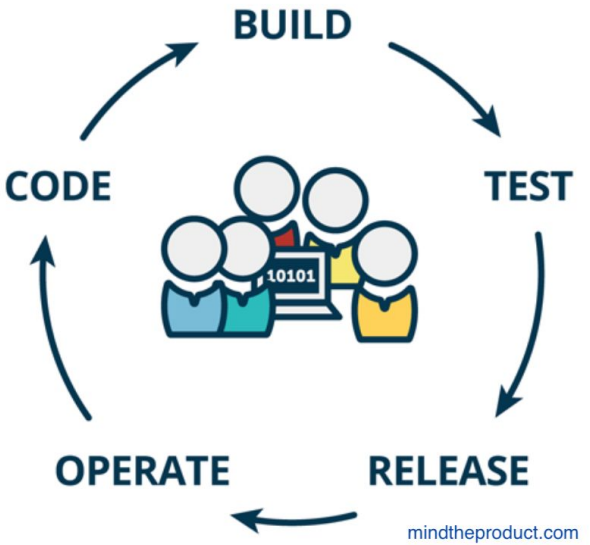
DevOps: Self-help philosophy for IT and software development teams

Solve a core problem:

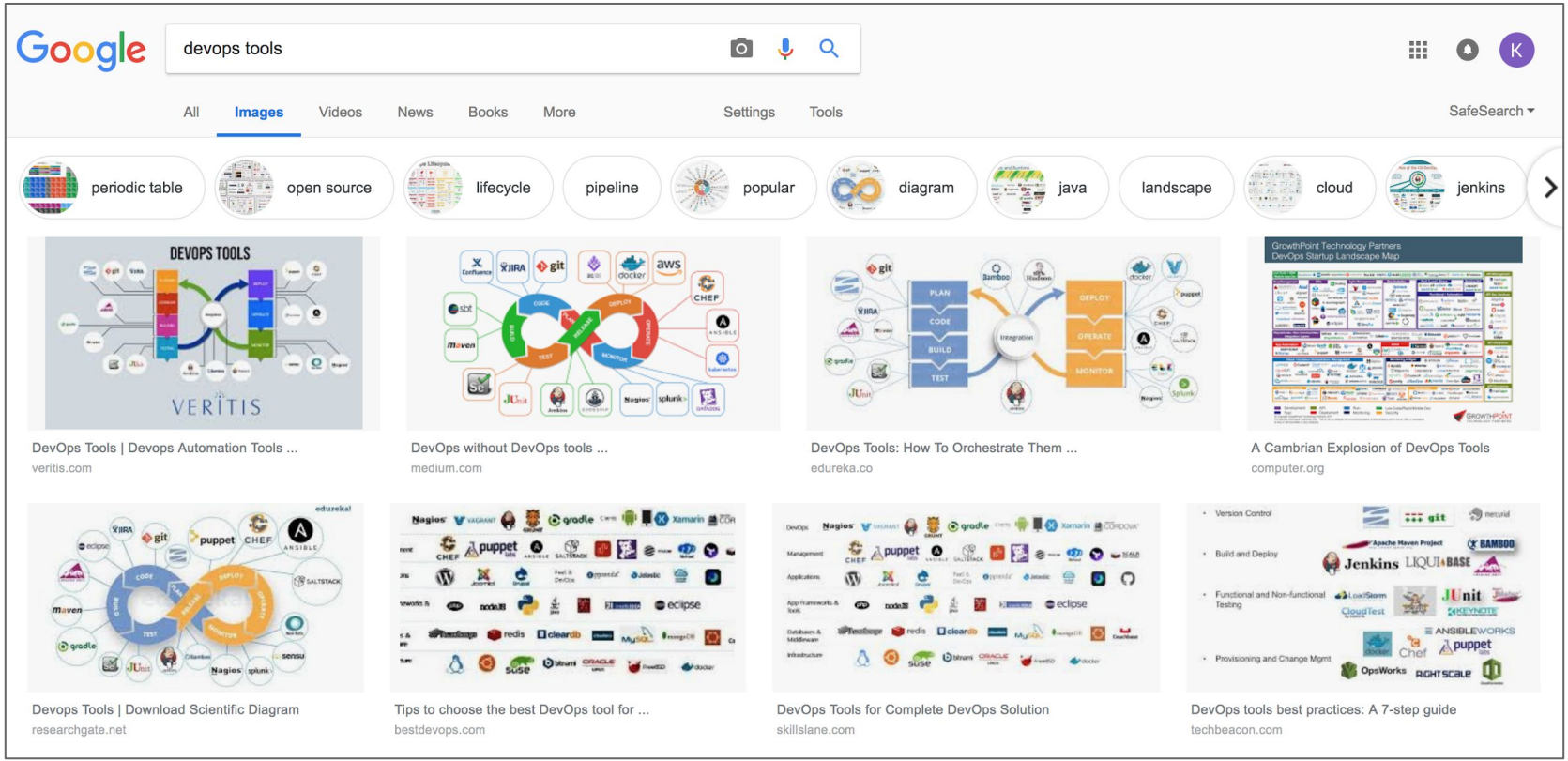
Everyone fears taking code into production

Code deployments are HIGH RISK EVENTS (performed rarely)

And they need to be TRIVIAL NON-EVENTS (daily work)



DevOps - Land of 1000 Intimidating New Tools



1. DevOps is a philosophy or set of best practices
2. It’s a new approach to software development and delivery
3. **There’s nothing new in DevOps**

A framework for making sense out of common sense

Are DevOps Principles just nerdy New Year's Resolutions?

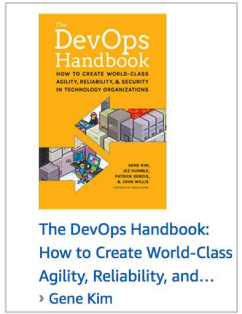
- **Flow:** Accelerate the delivery of work from development to operations to customers
- **Feedback:** Create ever safer systems of work
- **Continual Learning and Experimentation:** A high-trust culture and scientific approach to organizational improvement risk-taking as part of daily work

-- Support --

Technology Value Stream: The process required to convert a business hypothesis into a technology-enabled service that delivers value to the customer

The DevOps Handbook

Three principles form the underpinnings of DevOps:



1. Accelerate Flow

- Make work visible
- Limit Work in Progress (WIP)
- Reduce Batch Sizes
- Reduce the number of handoffs
- Continually identify and elevate constraints
- Eliminate hardships and waste

2. Utilize Feedback

- See problems as they occur
- Swarm to solve problems and build new knowledge
- Keep pushing quality closer to the source
- Enable optimizing for downstream work centers

3. Learn and Experiment

- Enable organizational learning and a safety culture
- Institutionalize the improvement of daily work
- Transform local discoveries into global improvements
- Inject resilience patterns into daily work

Principles are business-y and it takes some digging to get to the actionable self-help advice.

Everyone’s Production is Different

And Evolves at a Different Rate

Production is...

DOCUMENTATION - TESTING & MONITORING

- Creating apps that can reach a wider audience and are deployed/tested in a consistent manner
- Running in a way that is stable to use, documented and monitored

AT SCALE

- Scaled to a larger audience
- Bulletproof, scalable, fails predictably
- Live to 1000 of users with production vehicle data

CUSTOMER/USER FACING

- Ready to use
- Software that end users are using
- An app that is live and available to the end user
- Apps on our production server are available to our clients
- Client facing

Credibility

ENVIRONMENTAL REQUIREMENTS

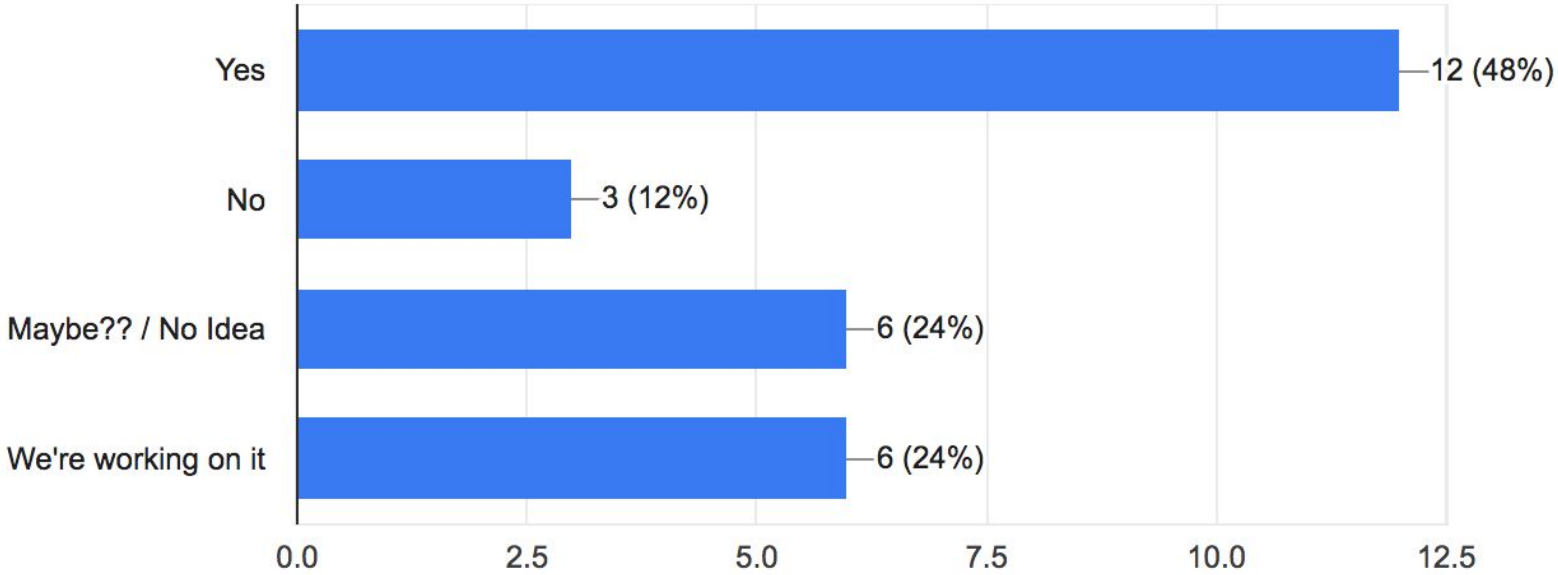
- An area where validated applications are deployed in a locked down environment
- The main part of a company that handles all process
- Application or system operates effectively without much maintaining effects
- A server or environment that runs the “final” applications that your ultimate end-users (often external customers) use to get stuff down

SERVICE LEVEL AGREEMENTS

- Required for mission-critical operations; downtime affects the ability to serve customers
- Deployed for end users to have continual access without performance issues

My organization has a team or process in place for handling code deployment

25 responses



Put a name on what you're working with

Pathological (power-oriented)	Bureaucratic (rule-oriented)	Generative (performance-oriented)
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to enquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

How to wade in … with Empathy *and* Strategery!

Does DevOps Exist in Your Org?

Yes

Is IT/Ops comfortable helping you bring Shiny to production?

🤩 *Best case scenario!* Get ready to help implement some novelty

😬 Make a checklist, answer questions, build a POC, be prepared to take it slow

Maybe?

¯_(ツ)_/¯

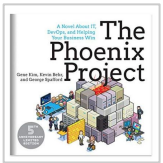
This is your chance to meet some people! Talk to a developer or IT Human!

Nope.

Are you comfortable bringing DevOps to your Org?

🤩 Cool Beans! Figure out who else needs convincing

Read Books!



😬 Noodle on it! Maybe it's worth it?

Make a communication plan and come with an open mind.

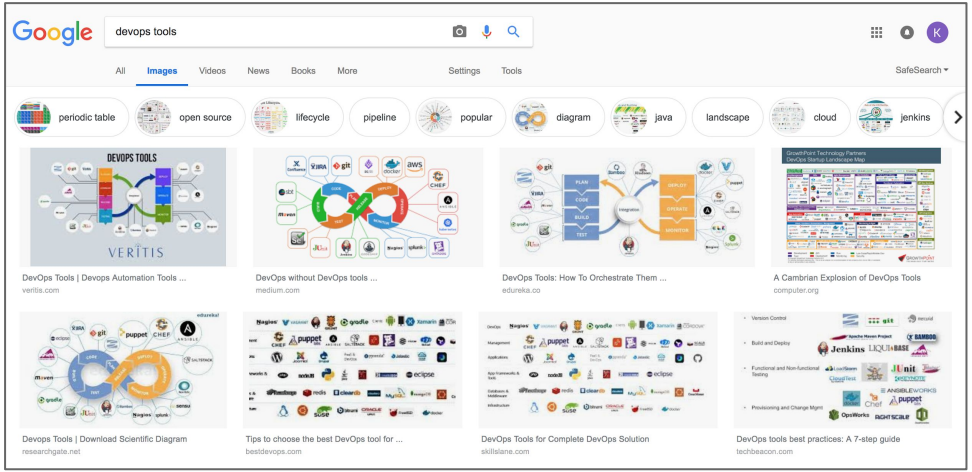
Empathetic Communication is Challenging

Start by answering some questions…

- What is a Shiny Application?
- Who is the audience?
- What is your service level agreement definition? (SLA)
- What does your analytic architecture look like today?
  - What are your goals for evolving this architecture?
- How will monitoring be handled?
- Who is responsible for maintenance?

Make work visible - Build a comprehensive checklist - Optimize

Don’t get caught up in the hype!



There are no easy answers

There is no “typical workflow diagram”

The *standard architecture* might not make sense for you

There is no perfect deployment pipeline - everything is *and should be* evolving

Even more important than daily work is the improvement of daily work

-- Mike Orzen, Lean IT

DevOps Foundations: Start Setting Goals

Goal: Develop in Production-like Environments

What do your dev, test and production environments look like?

- Do developers have access to production-like environments on their own workstations?

- Can these environments be assembled on-demand?

- Version Control 

- Infrastructure as Code 

Exercise: Relate it Back to R and Shiny

1. In what ways can analytic infrastructure for R change between development and production environments?
2. What processes or tools could help solve these issues?
3. *Should development inform how a production environment is built, or should production inform how an application is developed?*

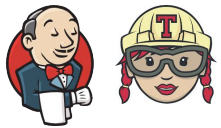
Goal: Make Code Deployment a Low-Risk Process

Why should data scientists design deployment processes?

You bring a uniquely informed perspective.

Having developers focus on automating and optimizing the deployment process can lead to significant improvements in deployment flow.

- Smoke testing
- Build validation tools
- Environment consistency



Exercise: Relate your Shiny deployment checklist to the pipeline an IT group might consider. Where is there risk? How can it be reduced?

- Packaging code in ways suitable for deployment
- Creating pre-configured virtual machine images or containers
- Automating the deployment and configuration of middleware
- Copying packages or files onto production servers
- Restarting server, applications or services
- Generating configuration files from templates
- Running automated smoke tests to make sure the system is working and correctly configured
- Running testing procedures
- Scripting and automating database migrations

Goal: Decouple Deployments from Releases

Deployment is any push of code to an environment (test, prod)

Release is when that code (feature) is made available to users or customers

Deployment on demand and thoughtful release strategies allow more control (and more success) over the delivery of features to end users.

Environment-based Release Patterns

- Blue-Green Deployment Pattern
- Canary and Cluster Immune System

Application-based Release Patterns

- Feature Toggles
- Dark Launches

Exercise: Which of these release patterns is most well-suited for Shiny Application development and deployment with the RStudio Connect publishing platform?

Case Studies 30 Minutes

A) Before publishing an update to our app, we want to run user acceptance tests.

B) All of our production code must be deployed from Git.

C) We scale applications through Docker - how does Shiny fit in?

First: Pick one case study

Discuss:

What tools does your admin team use?

How would you explain the goals of a shiny app to DevOps?

Deliverable

Whiteboard (in a medium of your choice) what an architecture might look like for your case study.

Delineate who (and what tool) is responsible for code, R, R packages.

Break

Come back at 4:00

Case Studies 10 Minutes

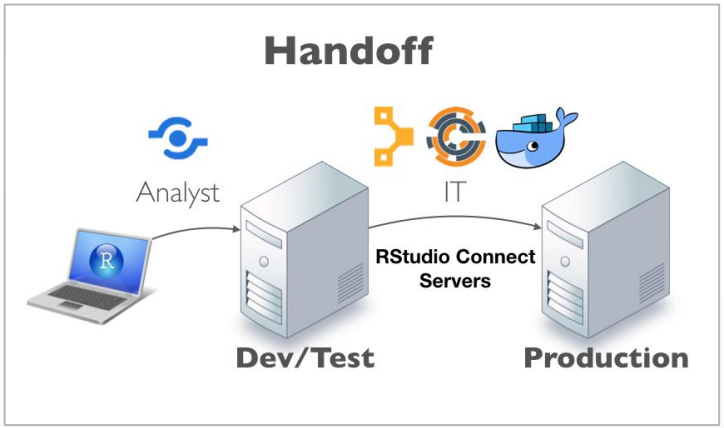
Re-acquaint yourself with the case study you picked.

Case Study A

Before publishing an update to our app, we want to run user acceptance tests.

Case Study A answers

Before publishing an update to our app, we want to run user acceptance tests.



Code: Commited to Git by users after being deployed to Dev.

R: Multiple versions installed on server by admin's Ansible scripts.

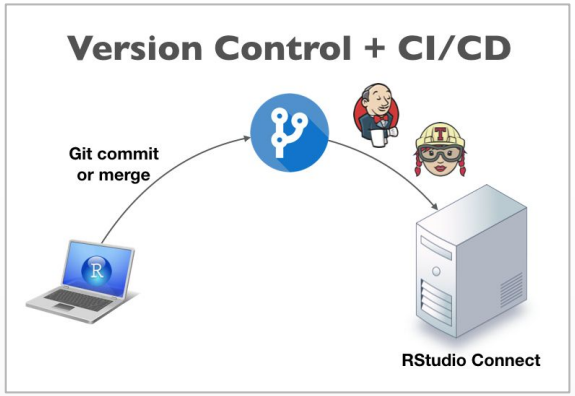
R packages: Restored by RStudio Connect

Case Study B

All of our production code must be deployed from Git.

Case Study B answers

All of our production code must be deployed from Git.



Code: Commited to Git by users. Deployed by Jenkins.

R: Multiple versions installed on server by admin's Ansible scripts.

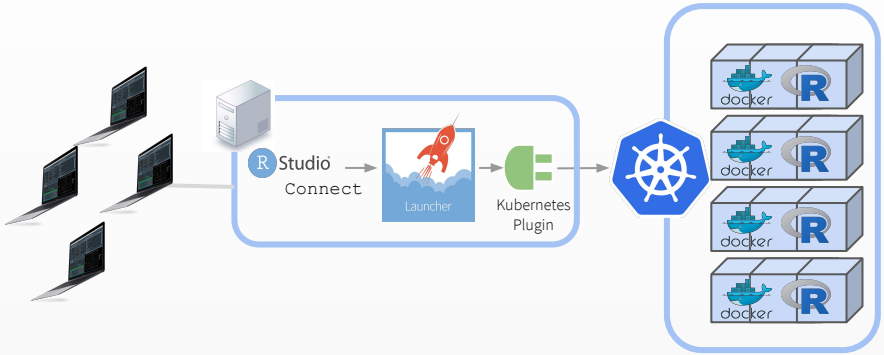
R packages: Restored from a manifest in Git during Jenkins runtime.

Case Study C

We scale applications through Docker - how does Shiny fit in?

Case Study C answers

We scale applications through Docker - how does Shiny fit in?

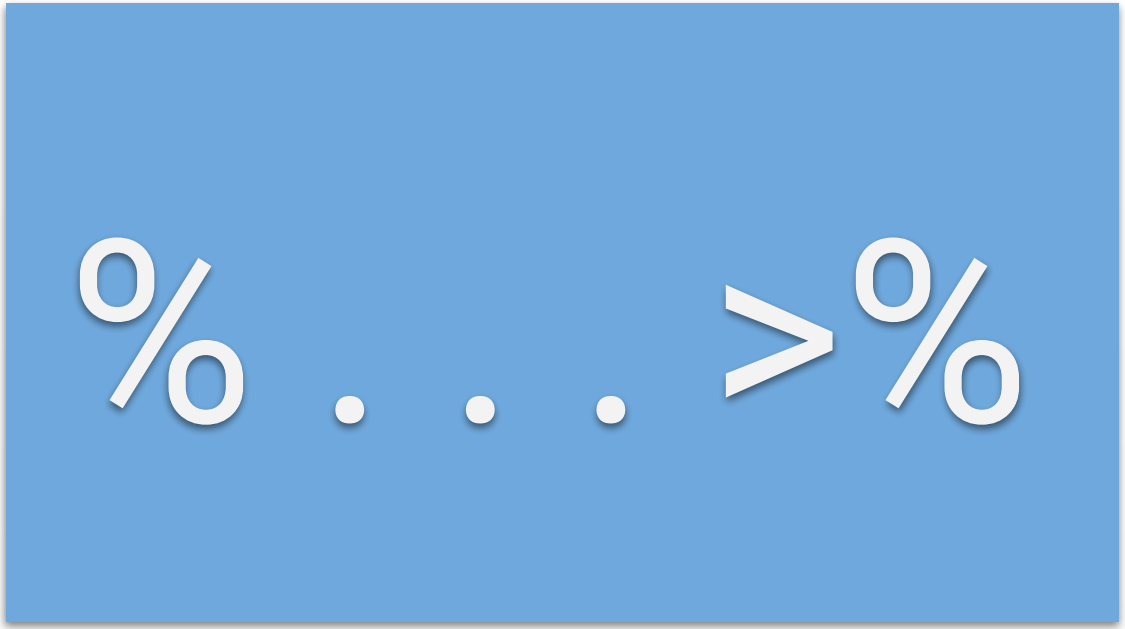


Code: Cloned into GitHub as a step in the Dockerfile.

R: Inherited from a base image.

R packages: Installed as a step in the Dockerfile.

Async



Checkout the Webinar

Wrap Up

What does our production-ready checklist look like?

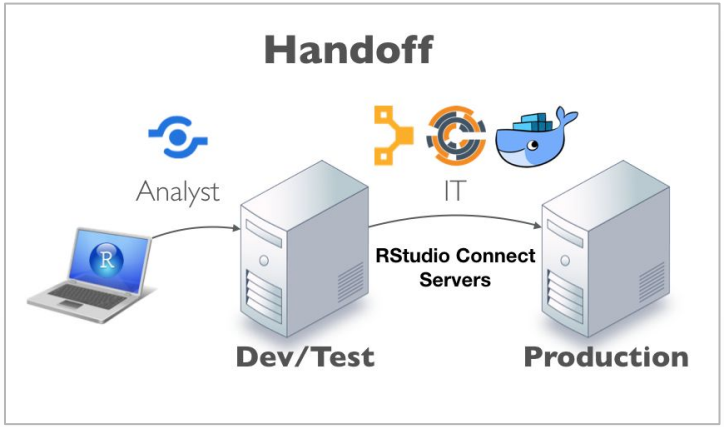
Thank you!

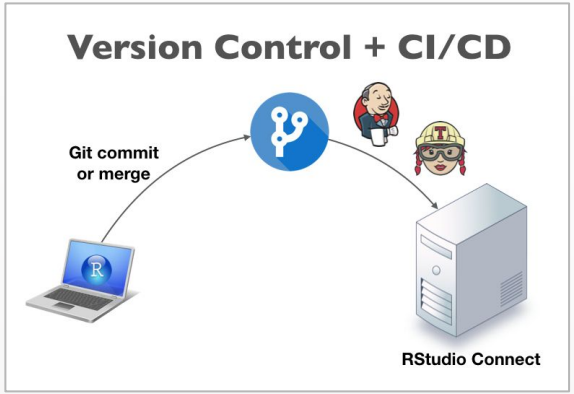
Course Feedback Survey: **https://bit.ly/2VFQRPo**

rstudio::conf AUSTIN

Appendix
(aka Garbage Slides)

DevOps

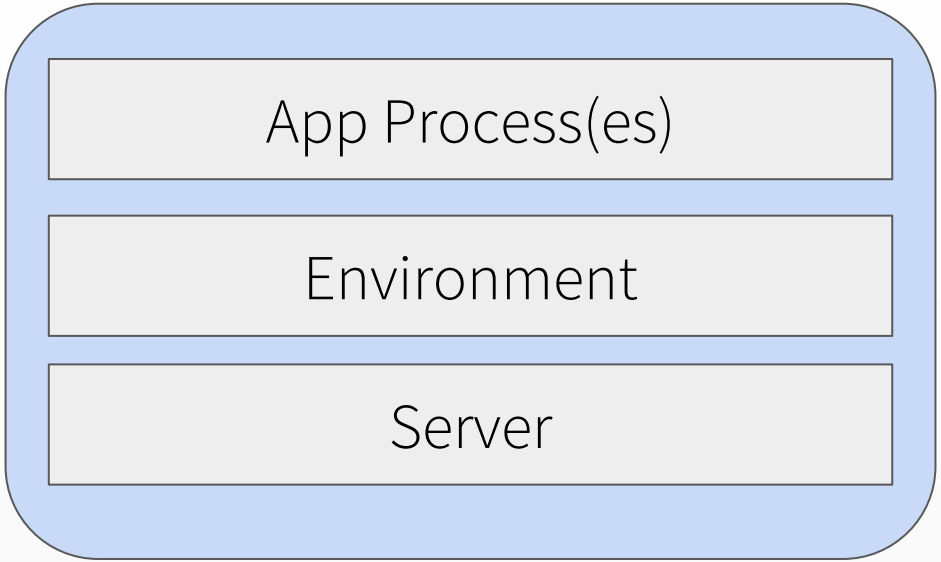

Handoff
Analyst
IT
RStudio Connect
Servers
Dev/Test
Production


Version Control + CI/CD
Git commit
or merge
RStudio Connect

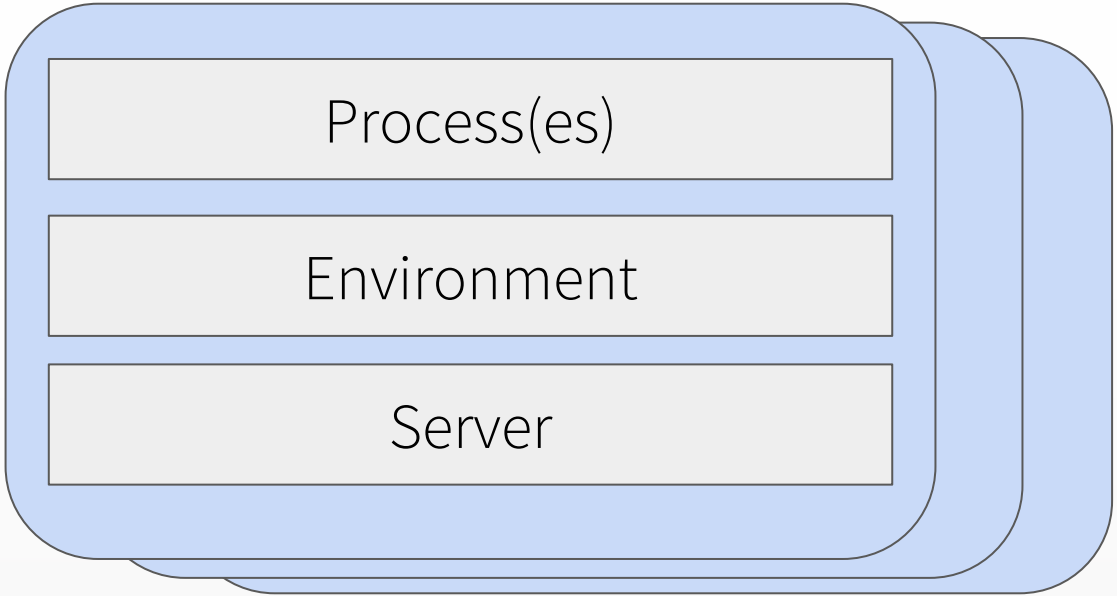
DevOps Vocab 101

Skip to 2:52

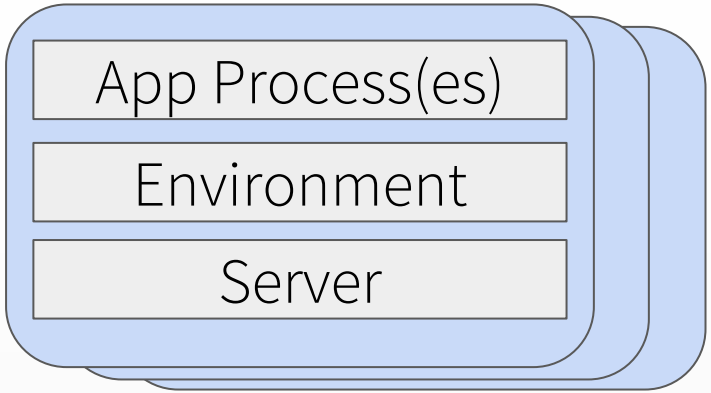
DevOps Vocab 101

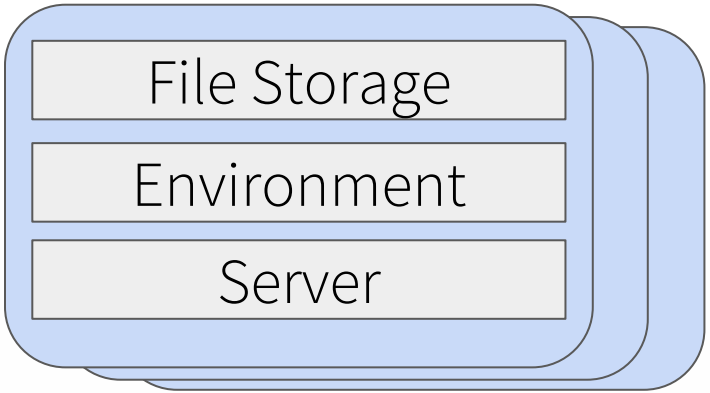


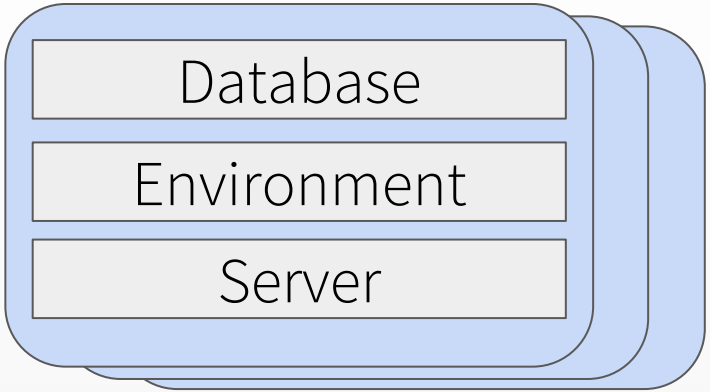
DevOps Vocab 101



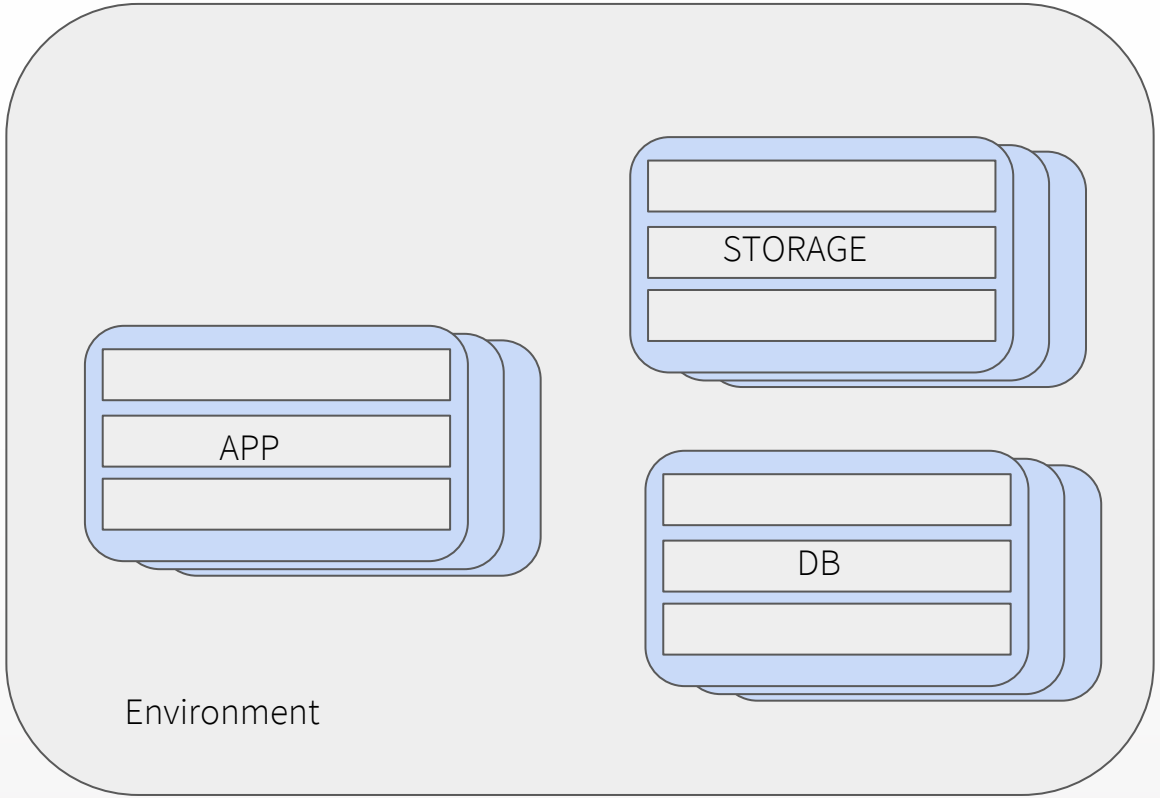
DevOps Vocab 101



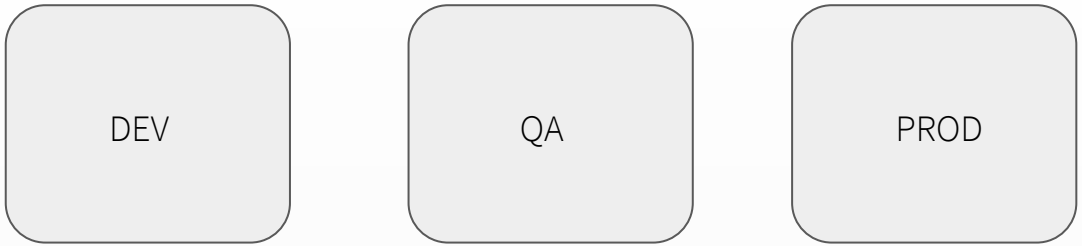




DevOps Vocab 101



DevOps Vocab 101



DevOps Vocab 101

App Process(es)

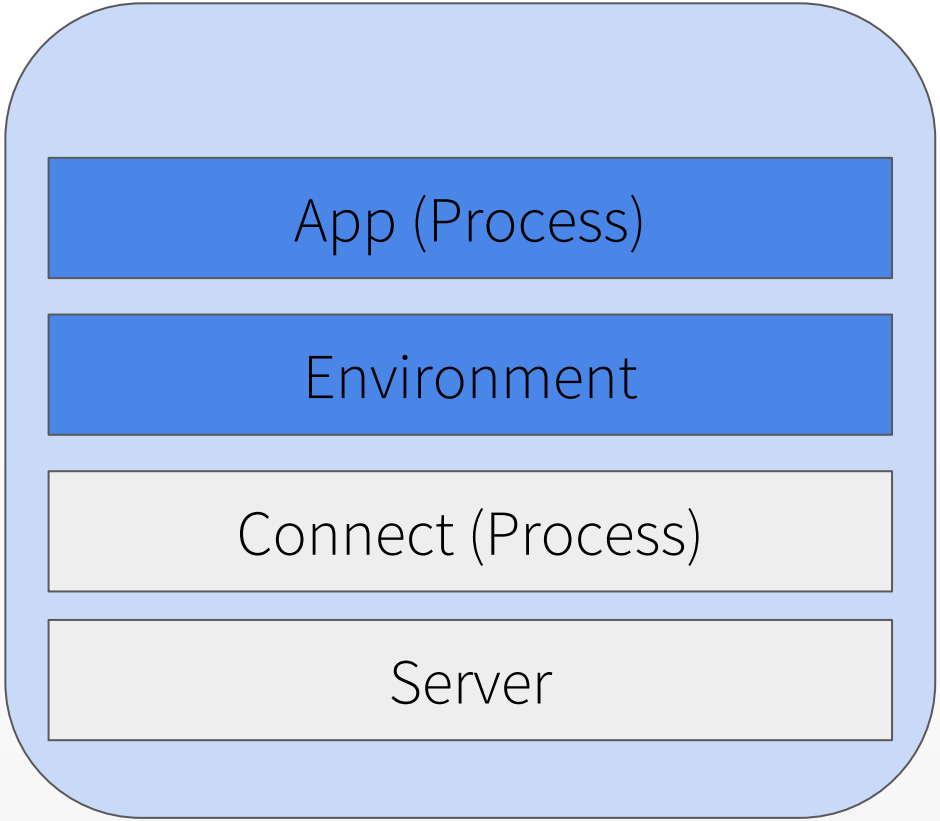
Environment

Server

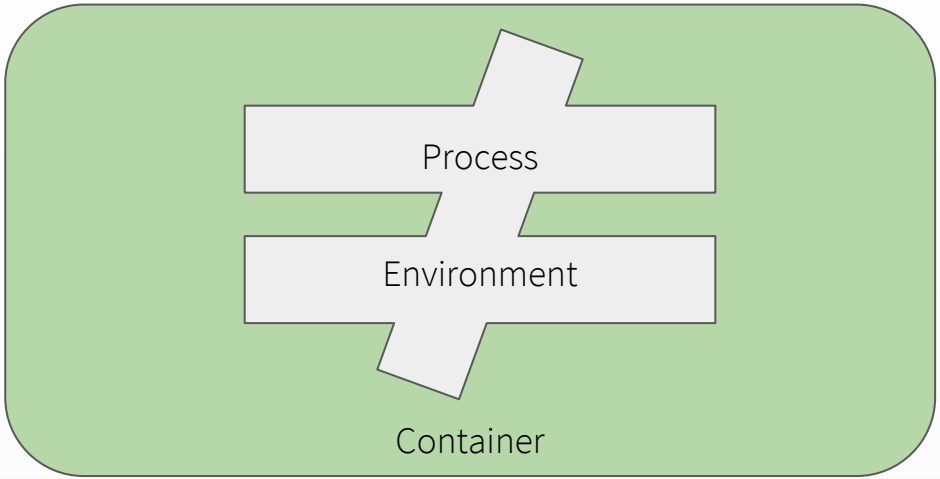
Things we care about!

Tools you might hear...

DevOps Vocab 101 - The RStudio Connect Version



DevOps Vocab 101 - The Docker Version



> “ Float like an environment, sting like a process.

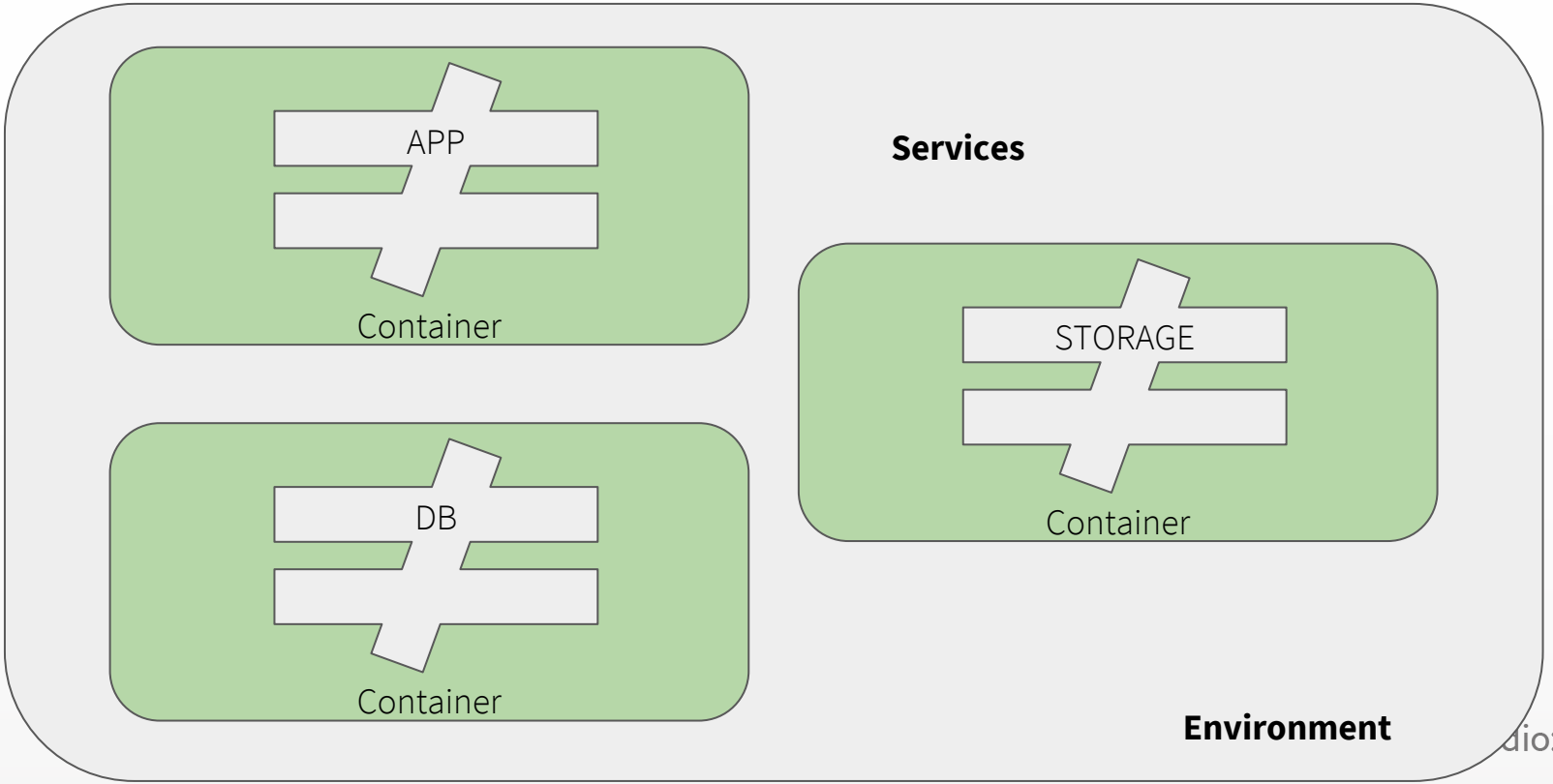
TBD

App Process(es)

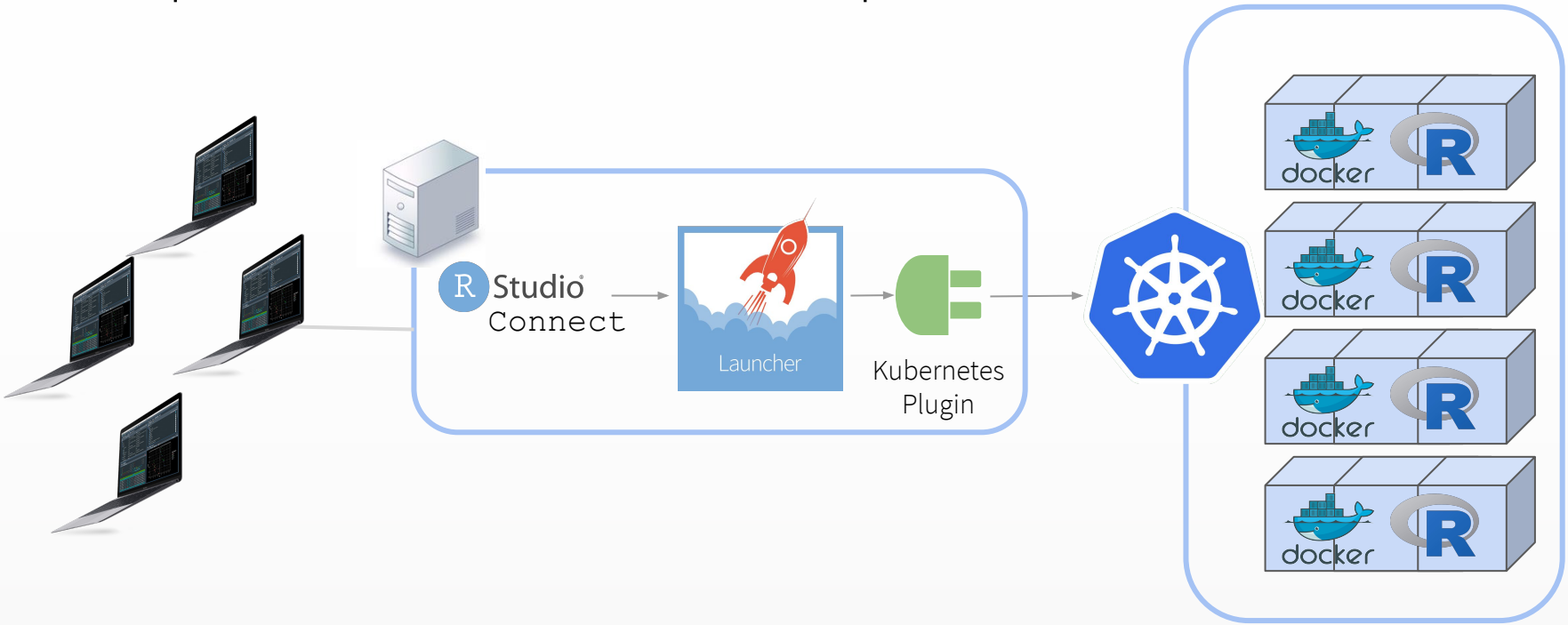
Environment

Server

DevOps Vocab 101 - Kubernetes?

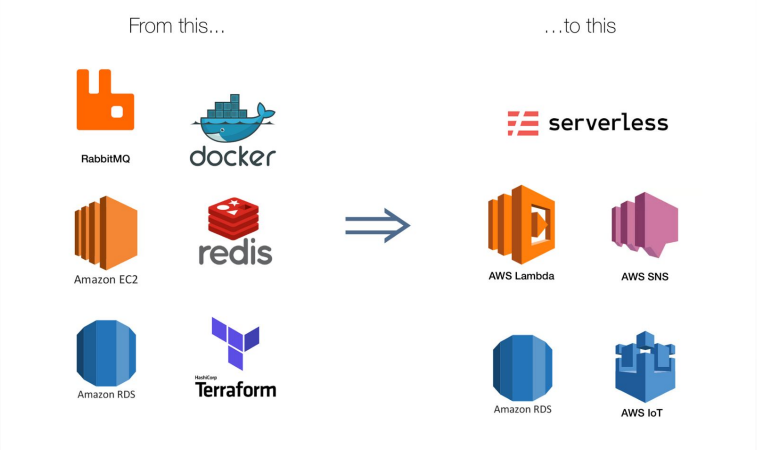


DevOps Vocab 101 - The Mashup



DevOps Vocab 102 - Other Hype Concepts

DevOps Vocab 102 - Names for Things



X

amazon

Azure