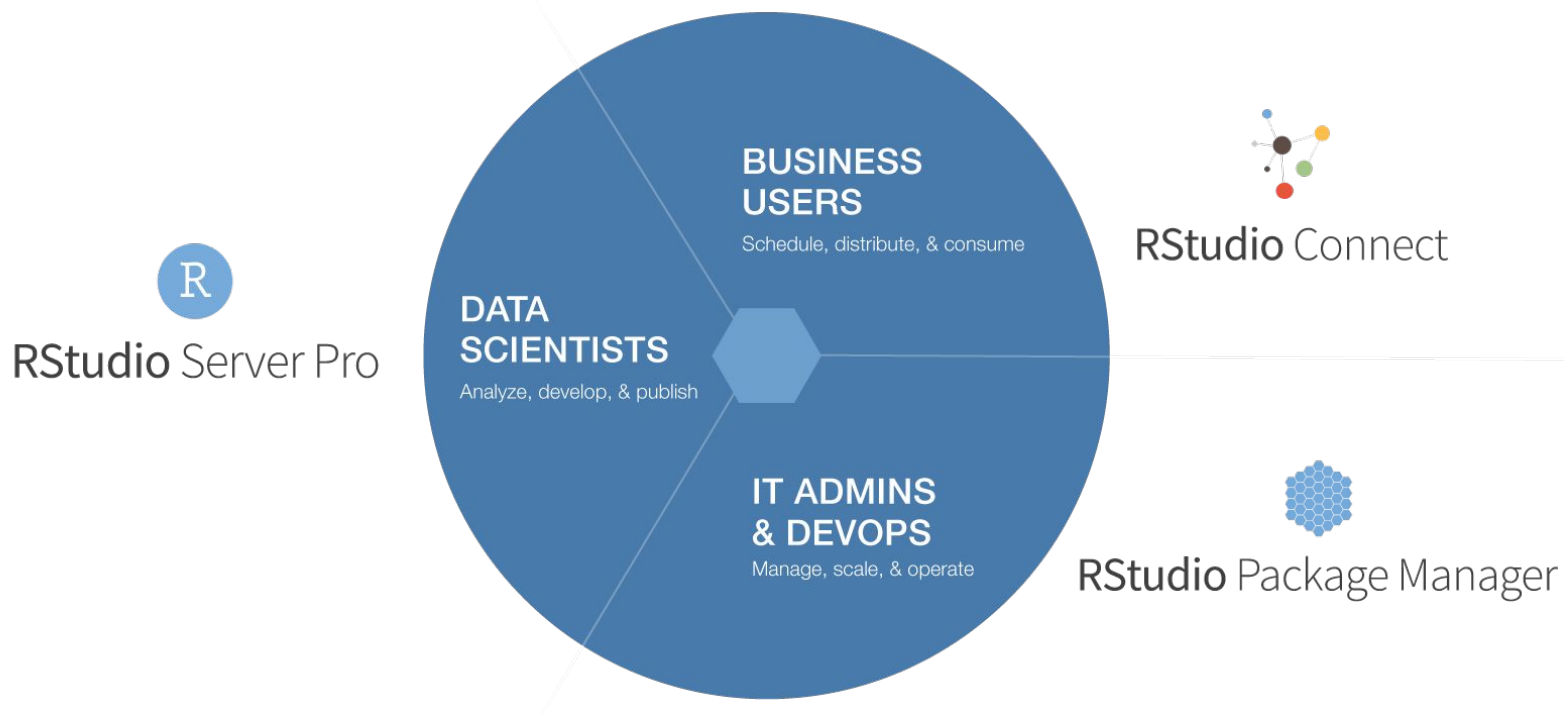


Automating the Delivery of Reproducible Data Products in R

Environment-based release patterns from DevOps



Solutions Engineering at RStudio



“R Admin” - Analytic Administrator Role

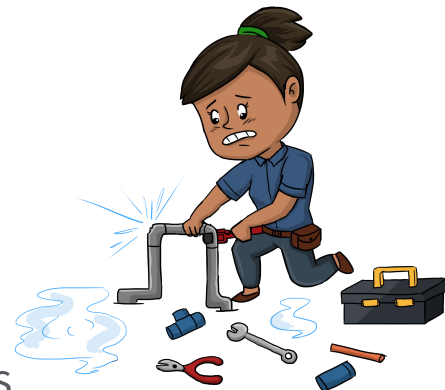
A data scientist who:

Onboards new tools, deploys solutions, supports existing standards

Works closely with IT to maintain, upgrade and scale analytic environments

Influences others in the organization to be more effective

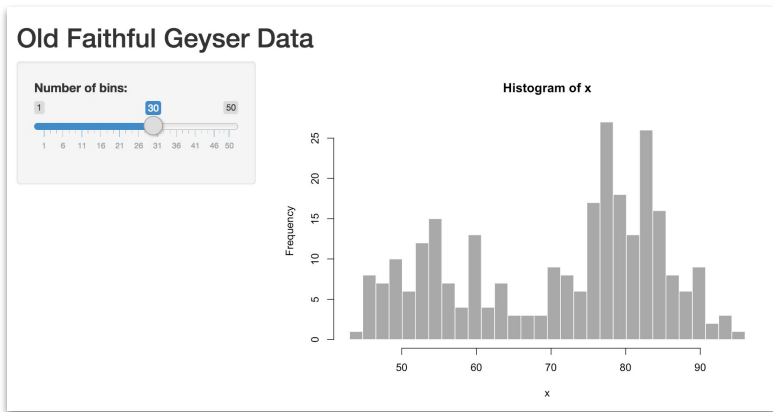
Passionate about making R a legitimate analytic standard within the organization



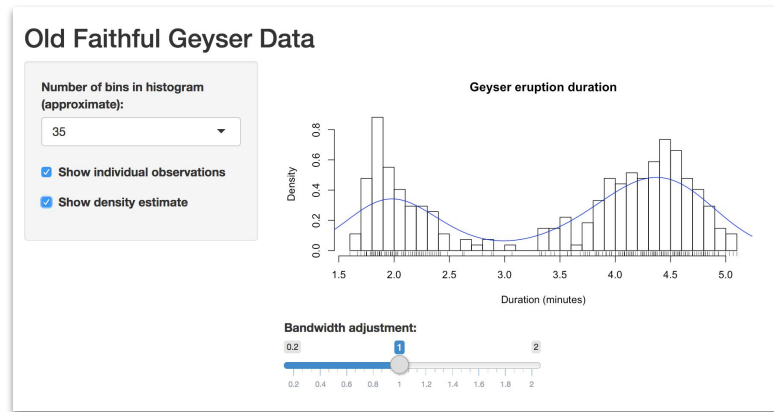
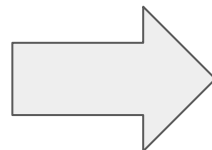
Check out Nathan Stephens on the RViews Blog -
[Analytics Administration for R](#)



Automating the Delivery of Data Products



Version 1



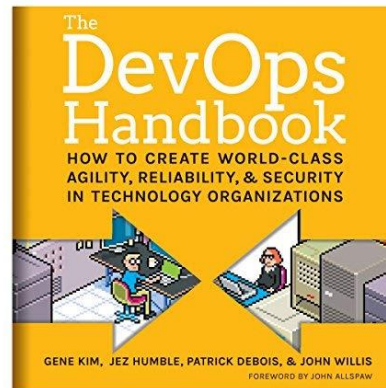
Version 2

DevOps Learning: Decouple **deployment** from **release**

- **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.

- Application-based release patterns ([yesterday](#))
- **Environment-based release patterns** (today!)

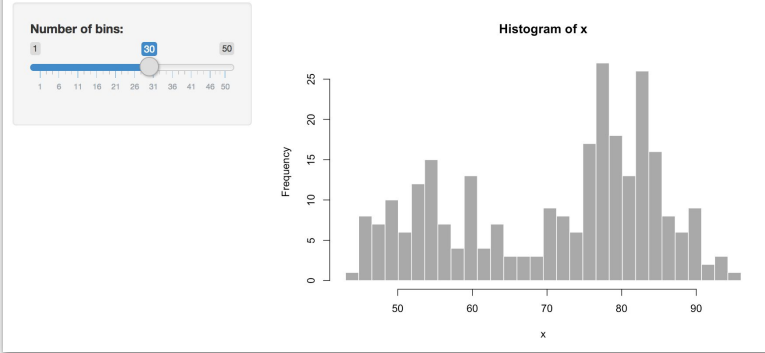


Environment-Based Release Patterns

Blue/Green Release Pattern

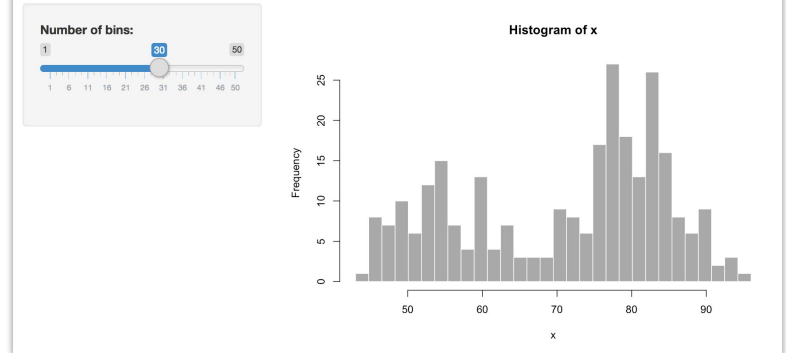


Old Faithful Geyser Data



Production Blue
Serving user traffic

Old Faithful Geyser Data

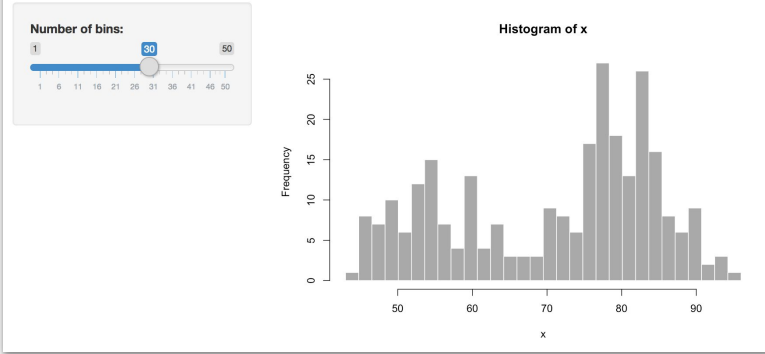


Production Green
Inactive

Blue/Green Release Pattern

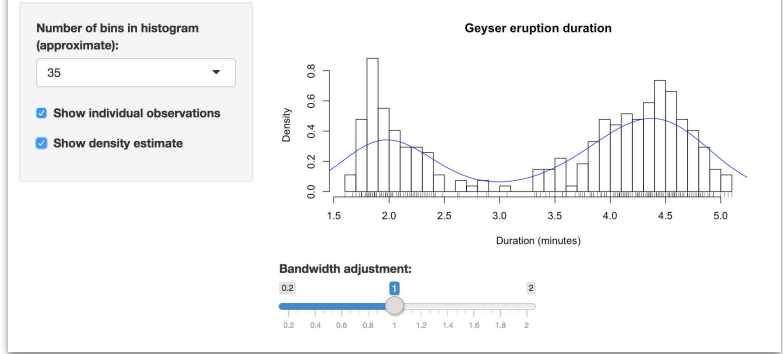


Old Faithful Geyser Data



Production Blue
Serving user traffic

Old Faithful Geyser Data

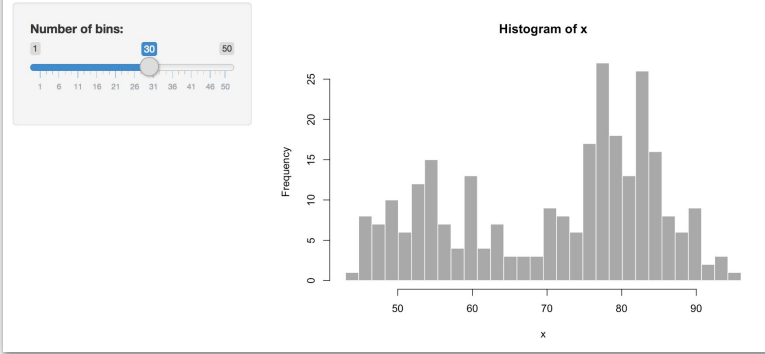


Production Green
Inactive - Staging

Blue/Green Release Pattern

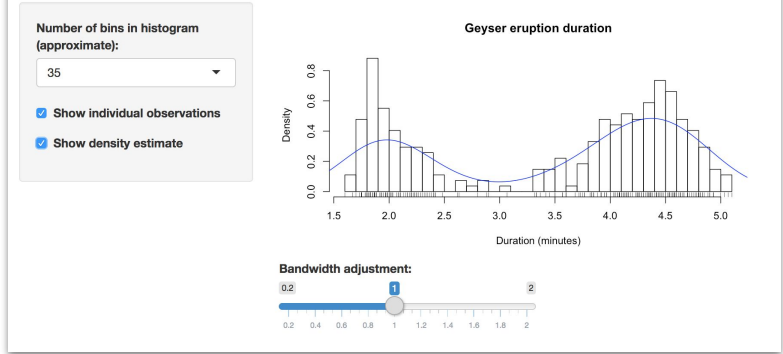


Old Faithful Geyser Data



Production Blue
Roll back if necessary

Old Faithful Geyser Data



Production Green
Start serving user traffic

Blue/Green Environment Release Pattern

Pros: Require no changes to application code

In this pattern, we have two production environments: blue and green. At any time, only one of these is serving customer traffic

To release a new version of our service, we deploy to the inactive environment where we can perform our testing without interrupting the user experience. When we are confident that everything is functioning as designed, we execute our release by directing traffic to the blue environment. Thus blue becomes live and green becomes staging. Roll back is performed by sending customer traffic back to the green environment.

Blue/Green Release Implementation

Does your environment support publishing to multiple destinations?

Vanity URLs can be useful for all published content types. The example below demonstrates how creating a custom vanity URL for a plumber API will update the base URL to a cleaner, more user-friendly address:

Custom URL (i)

/demo-api/

https://rstudio.connect.com/demo-api/

Copy To Clipboard

1. Default Plumber API Base URL:

[Base url: rstudio.connect/content/1638/

(Content ID based address)

2. Updated Vanity Base URL:

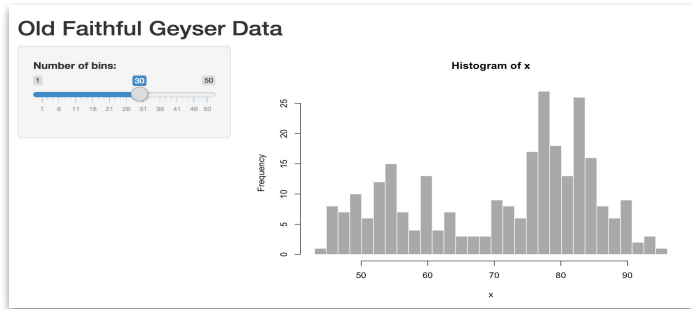
Plumber Example API 1.0.0

[Base url: rstudio.connect/demo-api/

(Custom user-friendly address)

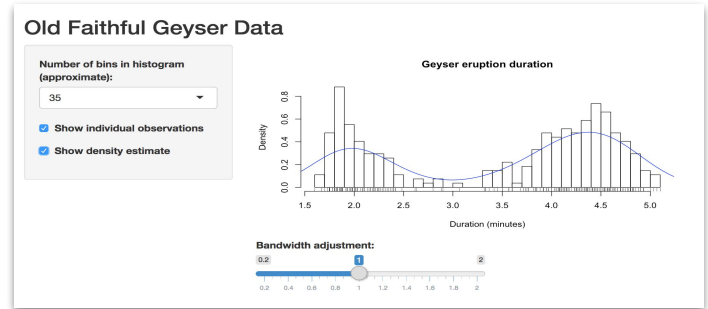
Blue/Green Release Implementation

1. Surface a single access point for your content
2. Assign a vanity URL to the original deployment location
3. Later assign it to a different piece of content on the same server



Production Blue
Serving user traffic

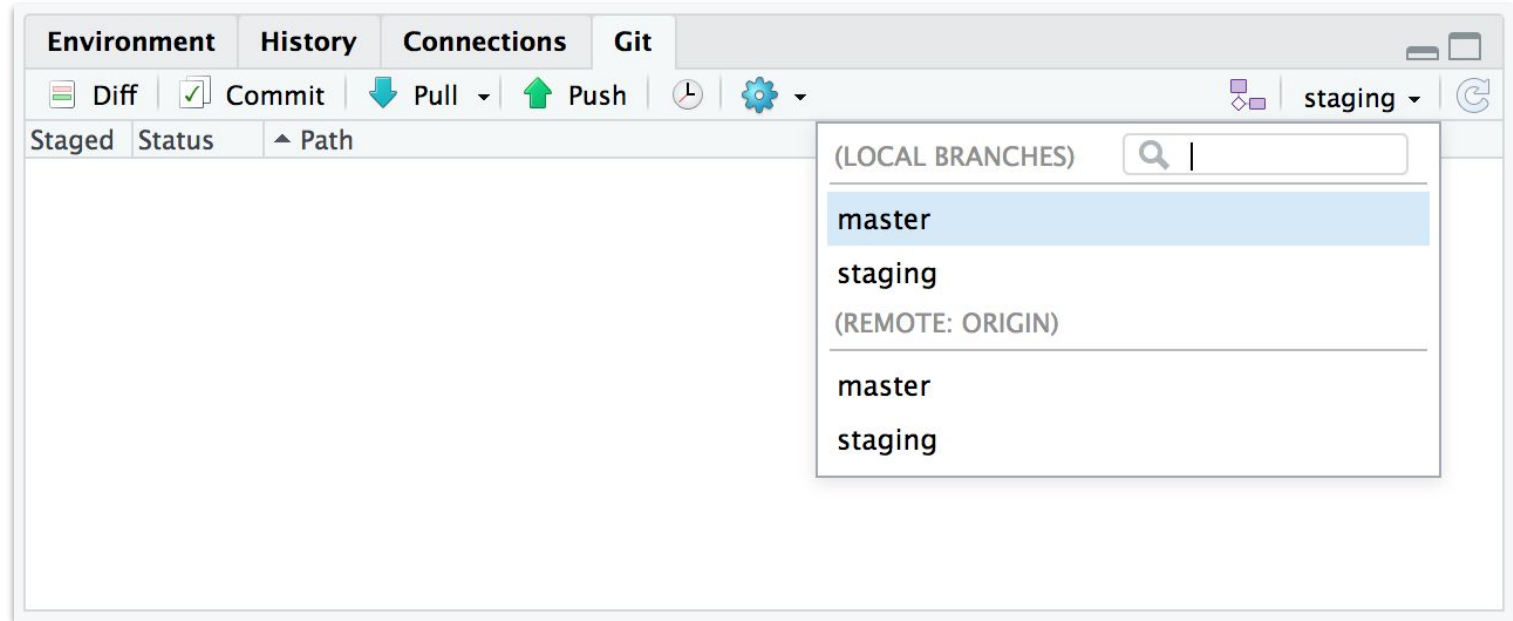
Vanity
URL



Production Green
Inactive - Staging

Implementation Recommendations

Git Branching Strategies: Beyond the master branch



Master (production)

Sprint-Branch (staging)

Manually Publish to Multiple Locations

http://127.0.0.1:5840 [Open in Browser](#) [Refresh](#) [Republish](#)

Old Faithful Geyser Data

Number of bins in histogram (approximate):

20

Show individual observations

Show density estimate

Geyser eruption duration

Density

Duration (minutes)

- ✓ histogram-BLUE
kelly@colorado.rstudio.com
- histogram-GREEN
kelly@colorado.rstudio.com
- upgrade-histogram-app
kelly@colorado.rstudio.com
- [Other Destination...](#)
- [Manage Accounts...](#)

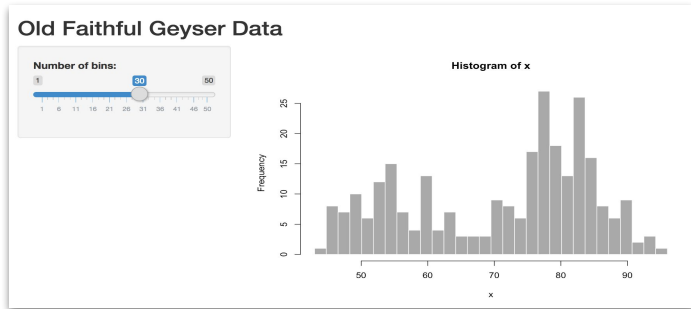
Automate Publishing to Multiple Locations

[Git-Backed Content Deployment Demo](#)

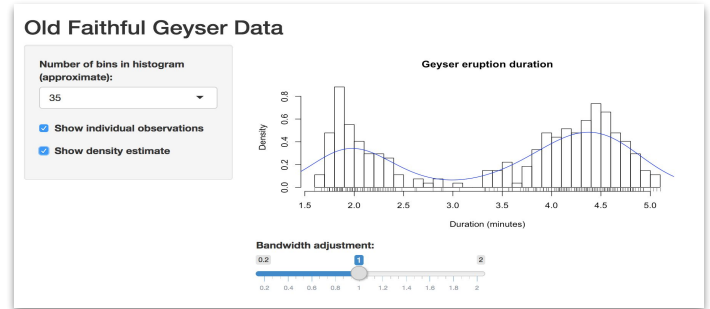
Manage the Vanity URL

Two applications cannot share a single vanity URL at any given time

1. Use the Access settings panel to manually swap custom vanity URL assignments
2. (Future) Automate with the [RStudio Connect Content API](#)

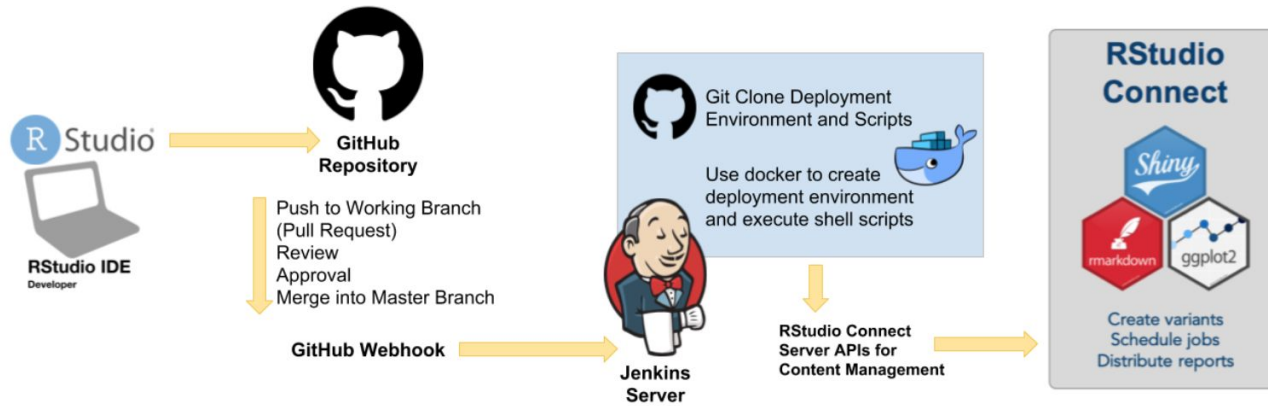


Production Blue
Rollback if needed



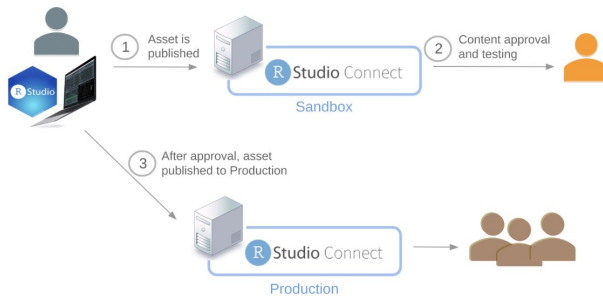
**Vanity
URL**

Production Green
Start serving user traffic



solutions.rstudio.com

Multiple environments



Continuous integration

