What I’ve learned about running Shiny in production

Luis de Sousa <luisd@syeop.co.za>

satRday Johannesburg 2019
Saturday, 6 April 2019
$ whoami
$ agenda
Where
Learn
Why
When not to use
Production

Where
Learn
Why
When not to use

Define
Goals

Challenges
Tools
Tips
Conclusion
Where to run Shiny

• R Studio Shiny Server Pro
• R Studio Connect
• ShinyApps.io
• Self Hosted
Learning about Shiny

• Documentation - https://shiny.rstudio.com
• Videos and written tutorials - https://shiny.rstudio.com/tutorial
• Blogs
  • R Studio - https://blog.rstudio.com/categories/shiny
  • Dean Attali - https://deanattali.com
• Twitter - #rstats
Why use Shiny

• Interactive R application
• Fast prototyping
• Build and deploy model in R
  • No code rewrite
• Open-source R package
Reasons not to use Shiny

• High throughout
  • R is an interpreted computer language

• Multi-page applications
Definition of Production

- Users relying on an environment with consequences when it’s down
Goals of Production

- Uptime
- Secure
- Bug free
- Performance
Challenges to Shiny in Production

• Organisation - Proving it can be production ready
• Cultural – Team’s willingness to adopt
• Technical – R developers are not traditional web or software developers
Tools: shinytest

- [https://rstudio.github.io/shinytest/](https://rstudio.github.io/shinytest/)
Tools: shinytest

- [https://rstudio.github.io/shinytest/](https://rstudio.github.io/shinytest/)
Tools: shinytest

• [https://rstudio.github.io/shinytest/](https://rstudio.github.io/shinytest/)
Tools: shinyloadtest

- [https://rstudio.github.io/shinyloadtest/](https://rstudio.github.io/shinyloadtest/)
- Record, Replay and Analyse

**Shiny:loadtest**

- **Sessions**
- **Session Duration**
- **Event Waterfall**
- **Latency**
- **Event Duration**
- **Event Concurrency**

*5 simulated users executing back to back sessions. Warmup or cooldown sessions (that are desaturated) start before or end after the vertical dotted line. Note that narrower event bars mean better performance.*
Tools: profvis

• https://rstudio.github.io/profvis/
Tools: Asynchronous programming

- [https://rstudio.github.io/promises/](https://rstudio.github.io/promises/)
- “DO use promises if you have a Shiny app with long-running operations, and want to serve multiple users simultaneously.”
- “DON’T use promises to improve the performance of Shiny apps for a single user.”
Tools: Feather

- [https://github.com/wesm/feather](https://github.com/wesm/feather)
- Binary columnar serialization for data frames
- Long running query that interrupts user experience
Tips: Setting up Production pipeline

- Linting
- Version control
- Code reviews and documentation
- Testing
- Deployment
  - CI / CD
    - Docker and Kubernetes
  - RStudio Connect
Tips: Filter as close to source as possible

- Move data wrangling as close to data source as possible
Tips: Plot caching

- [https://shiny.rstudio.com/articles/plot-caching.html](https://shiny.rstudio.com/articles/plot-caching.html)

```r
output$plot <- renderCachedPlot(
  {
    # Plotting code here...
  },
  cacheKeyExpr = { list(input$n, dataset()) }
)
```
Tips: Horizontal scaling

- Shiny applications can be scaled horizontally for high loads
Conclusion

• Use the tools and tips covered

• These things take time
  • Start early
  • Retrofitting is difficult

• Deploying in production is a team effort
  • Learn on the expertise of your friendly IT/Engineering resource
Resources

• https://github.com/luisdza/satRday_johannesburg_2019
• https://resources.rstudio.com/shiny/shiny-in-production-principles-practices-and-tools-joe-cheng
• https://twitter.com/_ColinFay/status/1085930604305235970