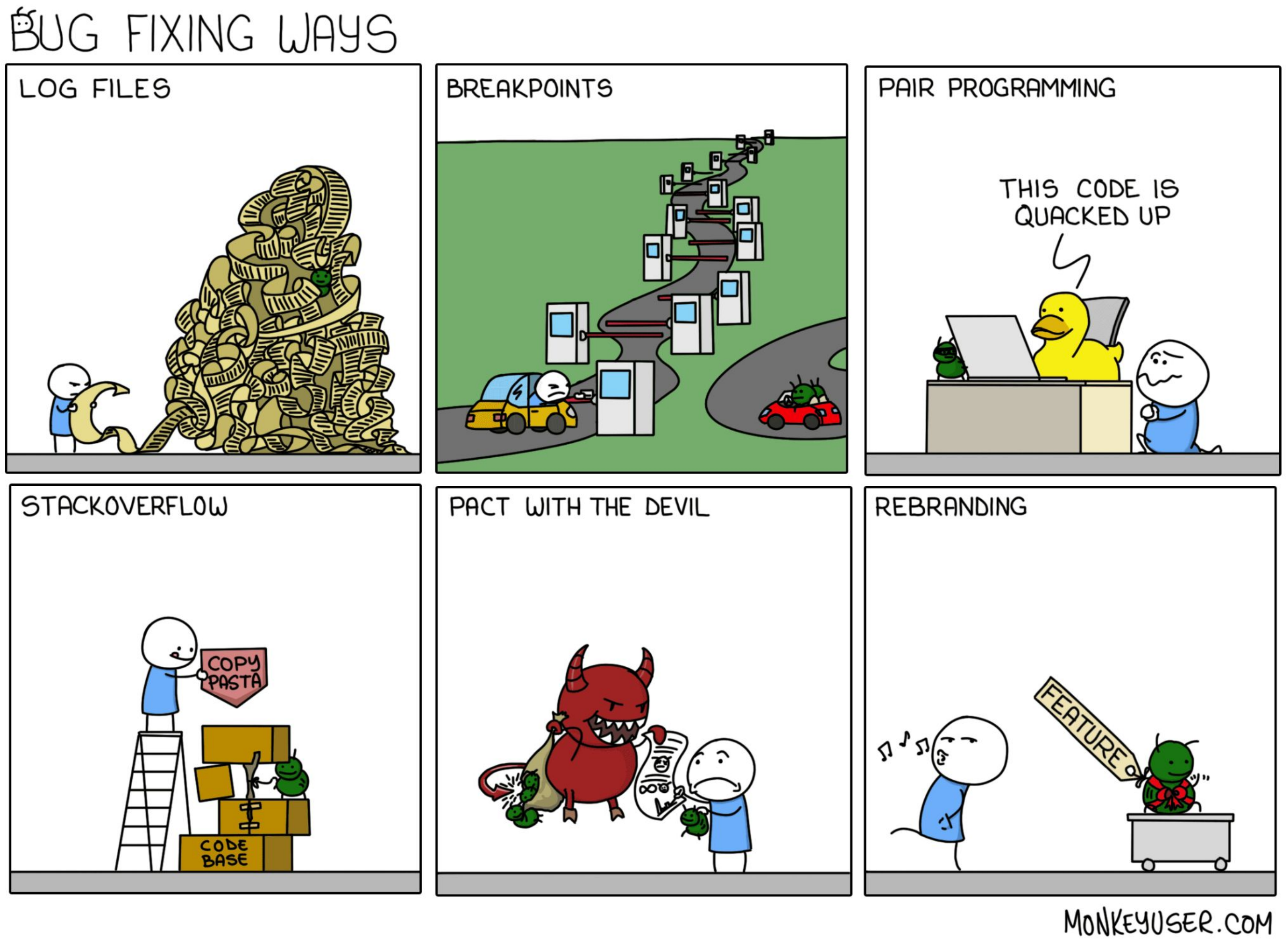
Effective Logging for Shiny

Tan Ho
ML Engineer, Zelus Analytics ⚽
Shiny in Production 2023

tanho.ca/logging-shiny

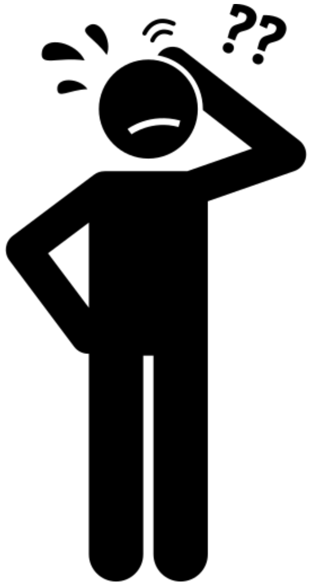
Motivations



https://www.monkeyuser.com/2019/bug-fixing-ways/

Me, trying to find the log files in our production Shiny apps

Me, finding out the log files weren’t useful



Me, trying to find best practices for logging in R & Shiny

WRITE THE BOOK YOU WANT TO READ.

- Austin Kleon, Steal Like An Artist

Agenda

App-level logging

Production-level logging

App-level logging

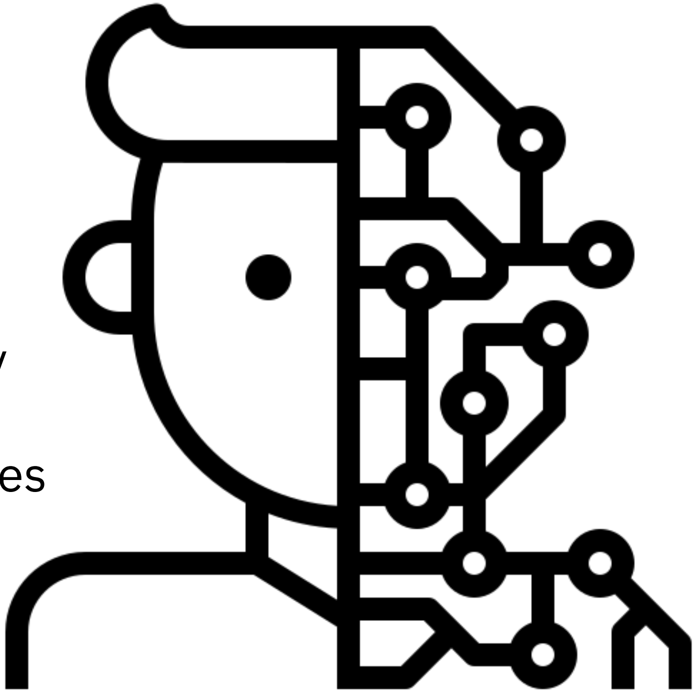
Who are you logging for?

Who are you logging for?

Humans

- Easy to scan visually
- Easy to understand
- Likes good summaries

Printing (to console)



Machines

- Easy to ingest
- Accuracy
- Completeness

Structured Data (JSON)

What should I be logging in my app?

What questions should my logs help me answer?

Errors

- Did we handle the error gracefully?
- Do we know the cause of the error?
- Can we get the context of this error?
- Can we reproduce this error?

Versioning and Environment Issues

Is this app running with...

- the latest versions of internal packages?
- the latest (or correct) versions of dependencies?
- the correct environment variables and package options?

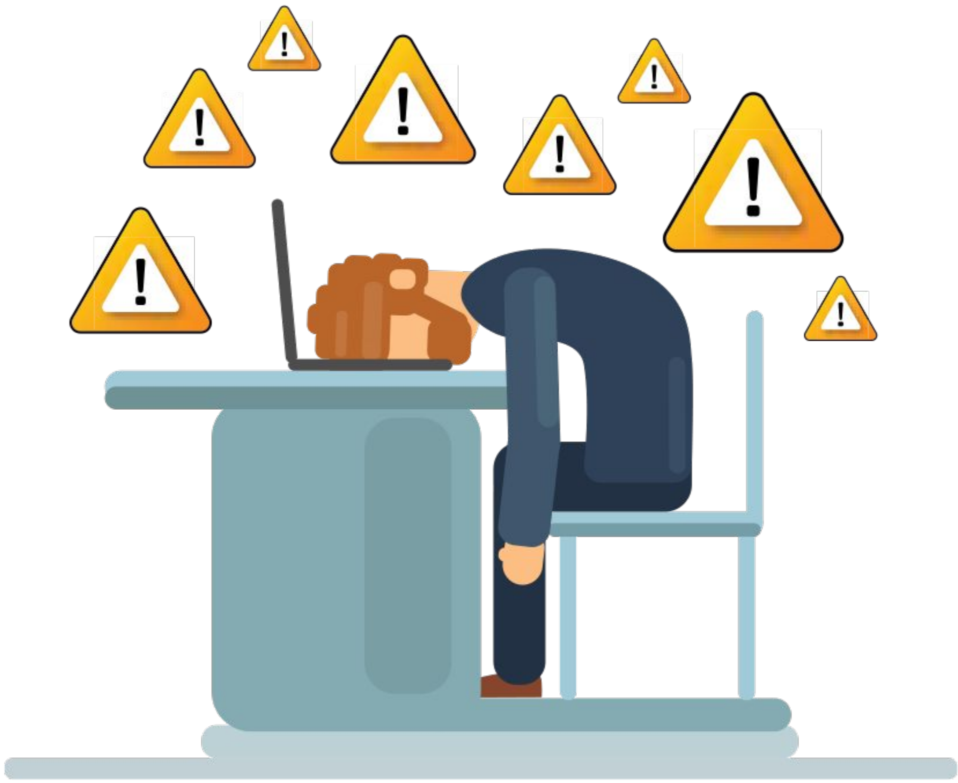
Users

- Who is using the app?
- What pages/sections/processes are they using?
- What actions did they take?
- Are they doing anything unexpected or suspicious?

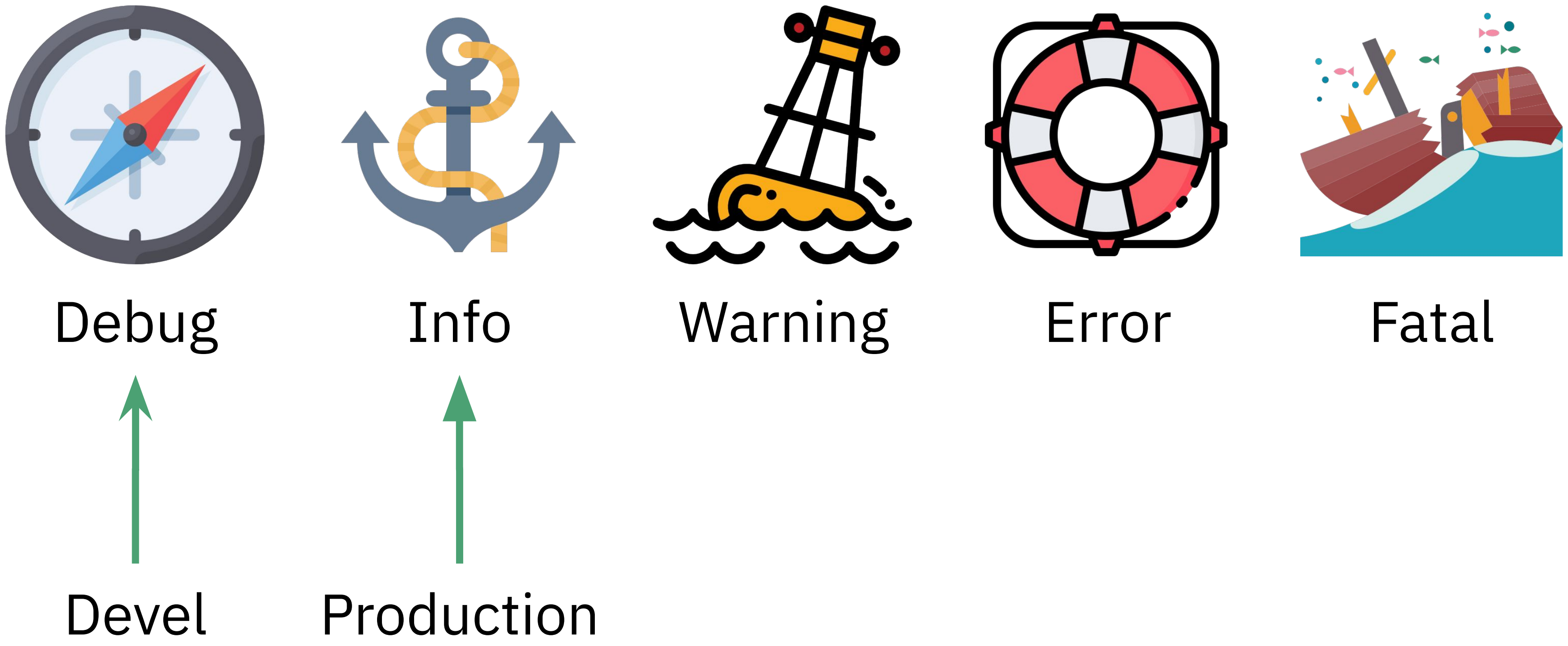
Expensive Functions & External Requests

- Which API/database/background job are we accessing?
- What (arguments, query) did we request from it?
- How long did it take? When did it start? When did it end?
- Did each step of the process complete successfully?

Problem: alert fatigue



Solution: alert levels!



What goes in each alert level?

Debug Info Warn Error Fatal

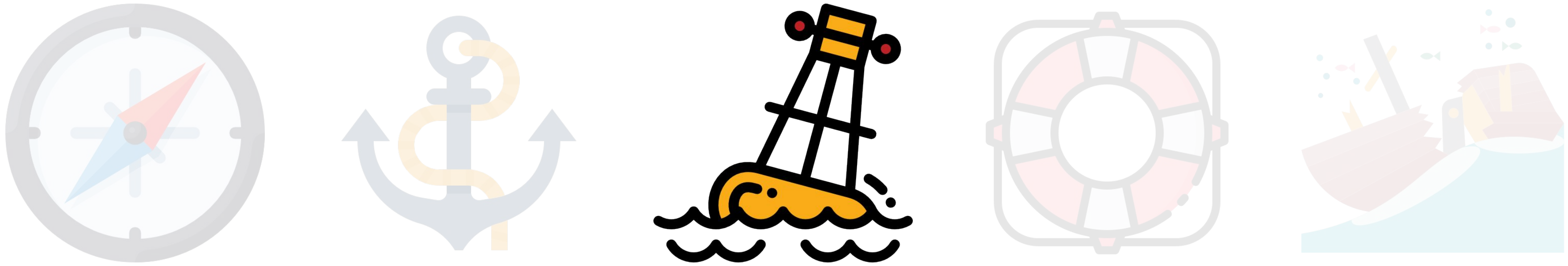
- Print debugging
  - input values, reactive states
  - helpful context for diagnostics
- Progress tracking within larger functions

What goes in each alert level?

Debug **Info** Warn Error Fatal

- Session-level information (eg user auth, system info)
- Performance timings of major/expensive calculations
- Significant user actions

What goes in each alert level?



Debug Info **Warn** Error Fatal

- Kind of like R’s *warning()*
- Edge cases, suspicious behaviours, and unusual inputs/states that still produce output

What goes in each alert level?

Debug Info Warn **Error** Fatal

- Errors that are handled gracefully by the app, e.g.
  - Input/argument validation
  - Function errors wrapped in *tryCatch()* handlers

What goes in each alert level?

Debug Info Warn Error **Fatal**

- Errors causing app crashes (ie Shiny grey screens)
- Corresponds to R’s *stop()* calls (unless properly handled)

How should I log things in my app?

Choosing a logging package

- futile.logger
- log4r
- logger
- logging
- loggit
- lumberjack
- rlog
- rsyslog

Choosing a logging package

logger

★ familiar syntax (glue)
★ performant & lightweight
★ actively maintained
★ flexible
  ○ alert levels
  ○ custom layouts & formatting
  ○ multiple destinations
  ○ can log as an async process

Some useful tricks

sitrep(): a customized session_info()

System info

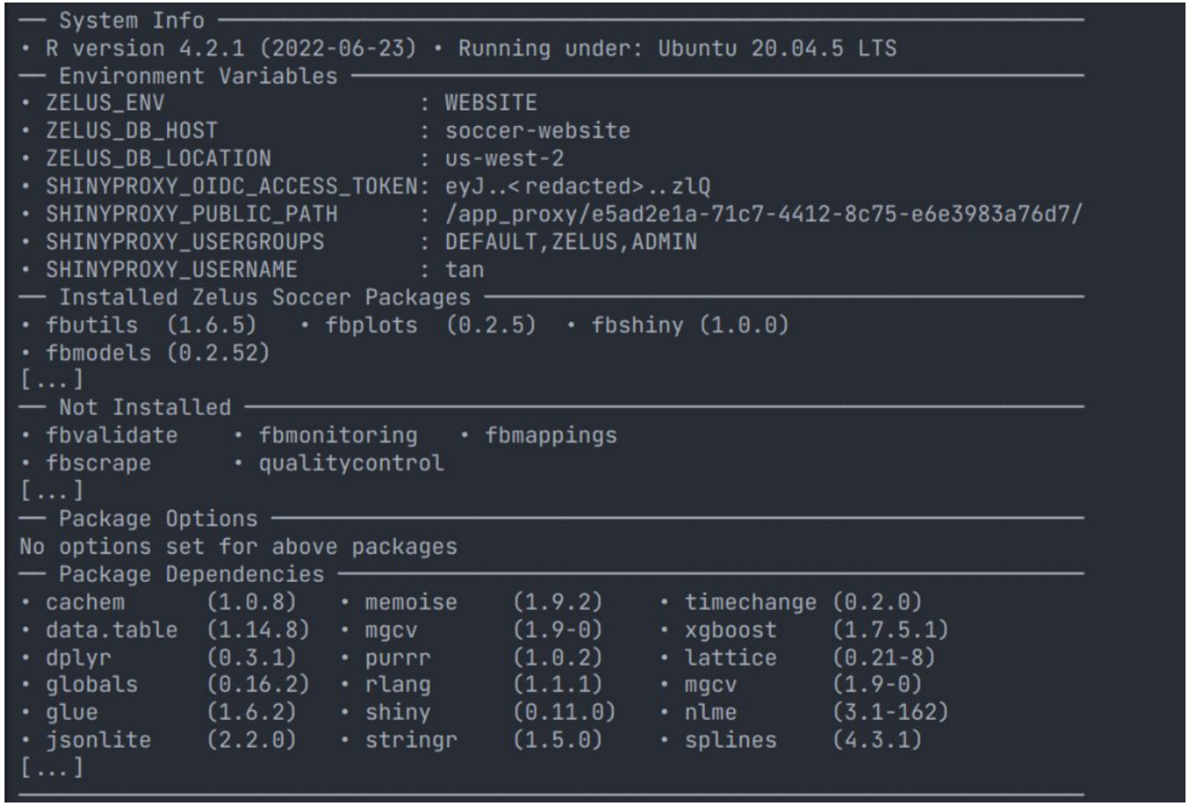
Env variables

Internal packages

Package options

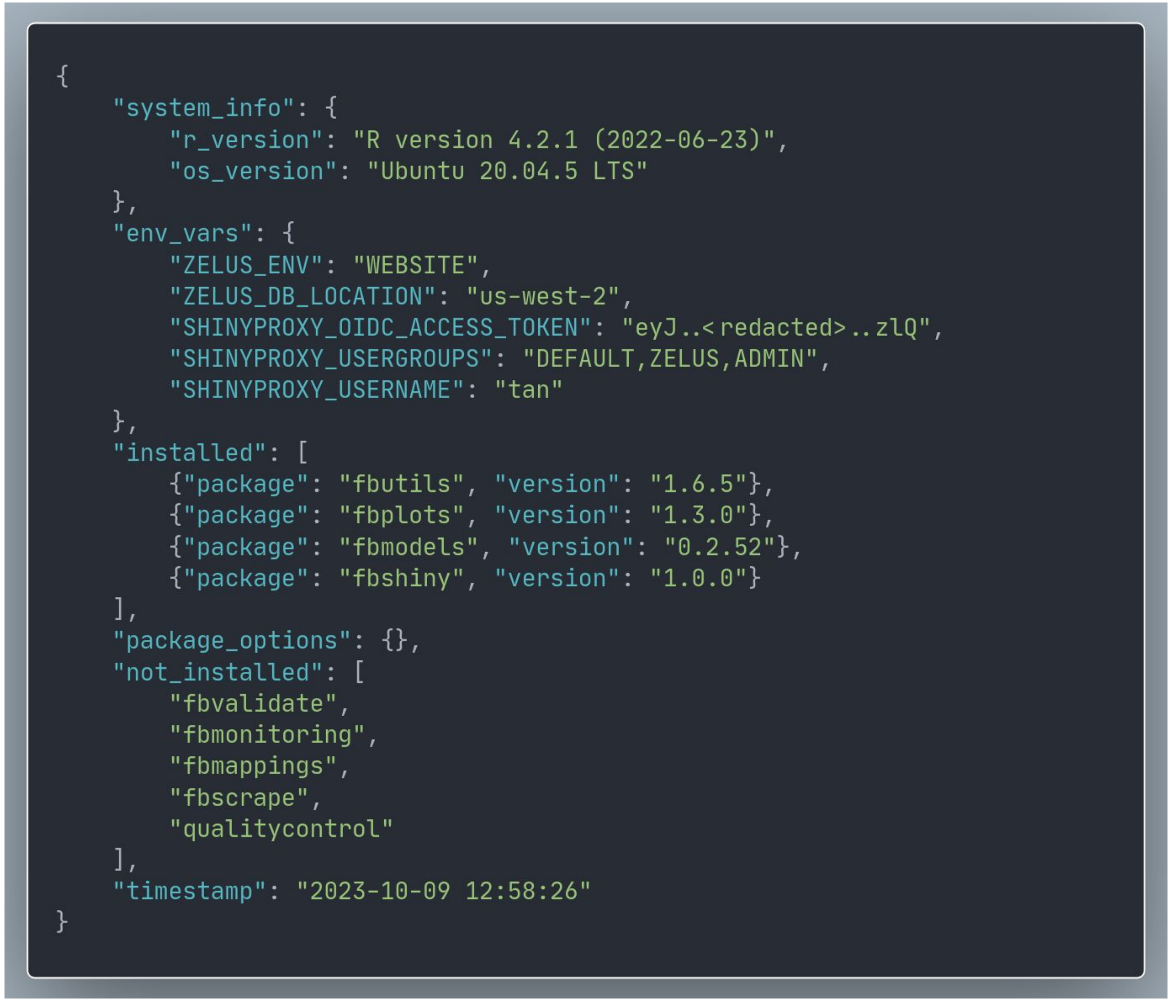
Dependency versions

Code



```
── System Info ──────────────────────────────────────────────────────────
• R version 4.2.1 (2022-06-23) • Running under: Ubuntu 20.04.5 LTS
── Environment Variables ────────────────────────────────────────────────
• ZELUS_ENV                   : WEBSITE
• ZELUS_DB_HOST               : soccer-website
• ZELUS_DB_LOCATION           : us-west-2
• SHINYPROXY_OIDC_ACCESS_TOKEN: eyJ..<redacted>..zlQ
• SHINYPROXY_PUBLIC_PATH      : /app_proxy/e5ad2e1a-71c7-4412-8c75-e6e3983a76d7/
• SHINYPROXY_USERGROUPS       : DEFAULT,ZELUS,ADMIN
• SHINYPROXY_USERNAME         : tan
── Installed Zelus Soccer Packages ──────────────────────────────────────
• fbutils  (1.6.5)   • fbplots  (0.2.5)   • fbshiny (1.0.0)
• fbmodels (0.2.52)
[...]
── Not Installed ────────────────────────────────────────────────────────
• fbvalidate    • fbmonitoring    • fbmappings
• fbscrape      • qualitycontrol
[...]
── Package Options ──────────────────────────────────────────────────────
No options set for above packages
── Package Dependencies ─────────────────────────────────────────────────
• cachem      (1.0.8)   • memoise    (1.9.2)    • timechange (0.2.0)
• data.table  (1.14.8)  • mgcv       (1.9-0)    • xgboost    (1.7.5.1)
• dplyr       (0.3.1)   • purrr      (1.0.2)    • lattice    (0.21-8)
• globals     (0.16.2)  • rlang      (1.1.1)    • mgcv       (1.9-0)
• glue        (1.6.2)   • shiny      (0.11.0)   • nlme       (3.1-162)
• jsonlite    (2.2.0)   • stringr    (1.5.0)    • splines    (4.3.1)
[...]
─────────────────────────────────────────────────────────────────────────
```

sitrep() |>
jsonlite::toJSON()



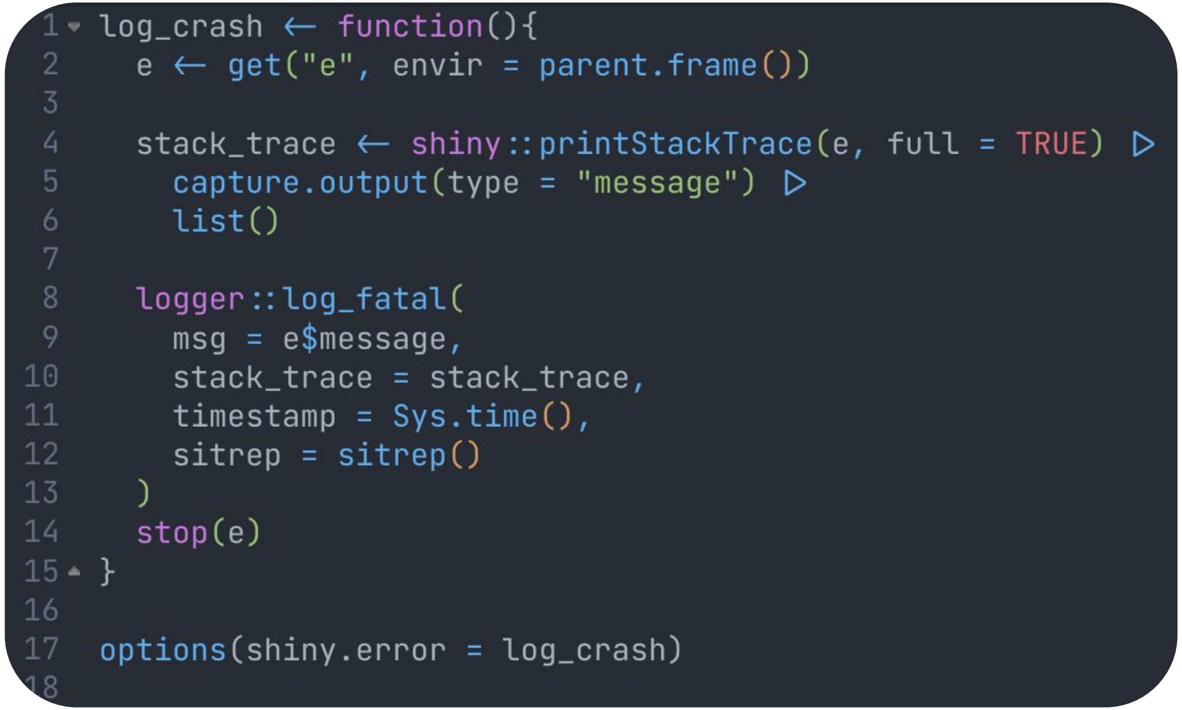
```
{
    "system_info": {
        "r_version": "R version 4.2.1 (2022-06-23)",
        "os_version": "Ubuntu 20.04.5 LTS"
    },
    "env_vars": {
        "ZELUS_ENV": "WEBSITE",
        "ZELUS_DB_LOCATION": "us-west-2",
        "SHINYPROXY_OIDC_ACCESS_TOKEN": "eyJ..<redacted>..zlQ",
        "SHINYPROXY_USERGROUPS": "DEFAULT,ZELUS,ADMIN",
        "SHINYPROXY_USERNAME": "tan"
    },
    "installed": [
        {"package": "fbutils", "version": "1.6.5"},
        {"package": "fbplots", "version": "1.3.0"},
        {"package": "fbmodels", "version": "0.2.52"},
        {"package": "fbshiny", "version": "1.0.0"}
    ],
    "package_options": {},
    "not_installed": [
        "fbvalidate",
        "fbmonitoring",
        "fbmappings",
        "fbscrape",
        "qualitycontrol"
    ],
    "timestamp": "2023-10-09 12:58:26"
}
```

Logging app crashes as (FATAL) errors

Experimental…

hopefully not (too) off-label? 😅

[Code]



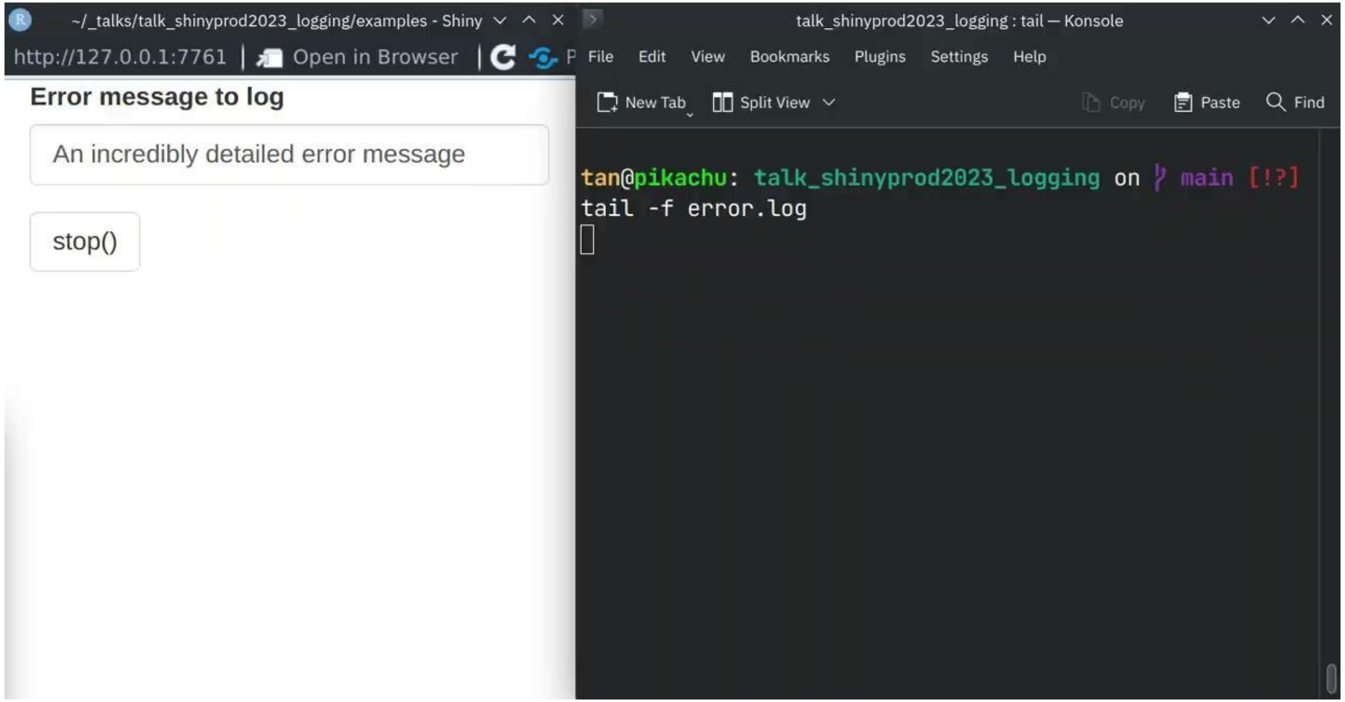
```r
log_crash ← function(){
  e ← get("e", envir = parent.frame())

  stack_trace ← shiny::printStackTrace(e, full = TRUE) ▷
    capture.output(type = "message") ▷
    list()

  logger::log_fatal(
    msg = e$message,
    stack_trace = stack_trace,
    timestamp = Sys.time(),
    sitrep = sitrep()
  )
  stop(e)
}

options(shiny.error = log_crash)
```

Logging app crashes as (FATAL) errors


~/_talks/talk_shinyprod2023_logging/examples - Shiny
http://127.0.0.1:7761
Open in Browser
talk_shinyprod2023_logging : tail — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab
Split View
Copy
Paste
Find
stop()

Error message to log

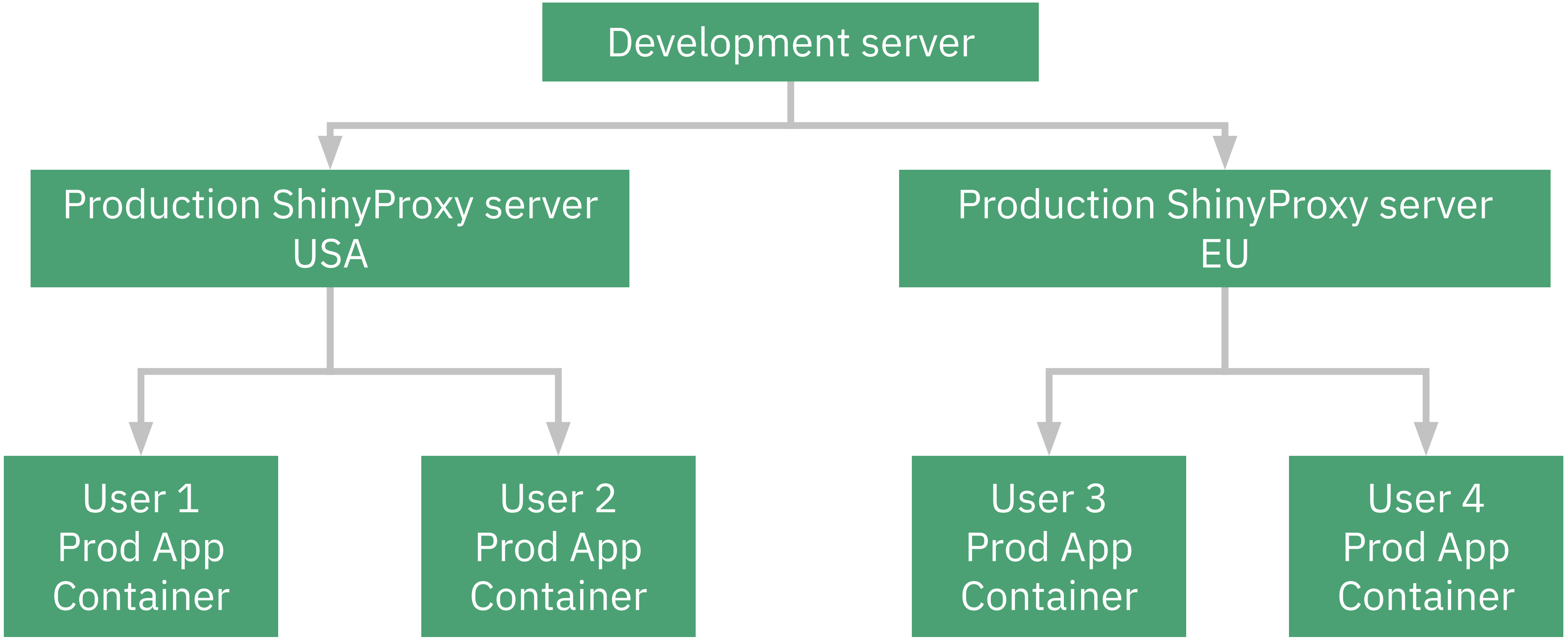
An incredibly detailed error message

```
tan@pikachu: talk_shinyprod2023_logging on  main [!?]
tail -f error.log
```

Logging as a Production System

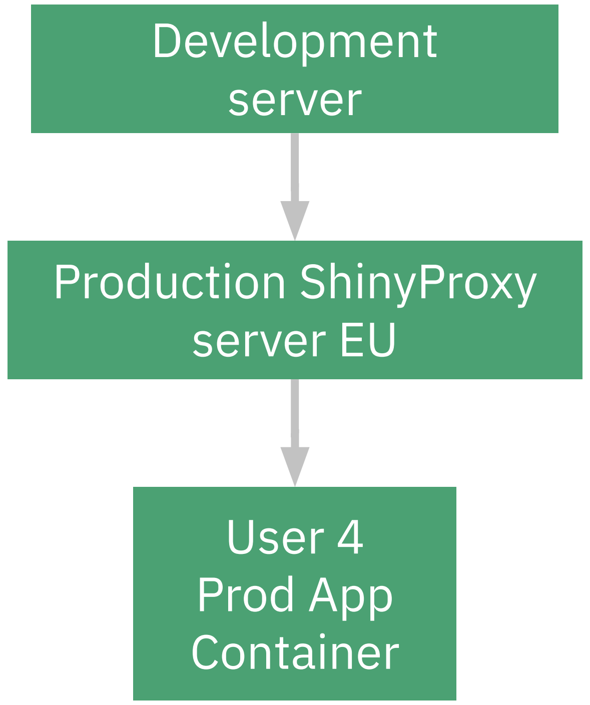
Me, trying to find the log files in our production Shiny apps

Zelus Soccer’s Shiny Architecture


Development server
Production ShinyProxy server USA
Production ShinyProxy server EU
User 1 Prod App Container
User 2 Prod App Container
User 3 Prod App Container
User 4 Prod App Container

Digging through the logs

1. ssh into development server
2. ssh from dev into correct prod server
3. Find container's log files
4. Try to find specific error in log files
5. Pray:
    a. that my vim-fu isn't too rusty
    b. that I'm looking at the right set of log files
    c. that the logs are useful



It would be really nice if we had a centralized system…

Choosing a log management system

- Grafana + Loki + Kubernetes
  - aka ShinyProxy Monitoring Stack
- AWS Cloudwatch
- Splunk
- Datadog, Sentry, Amplitude
- Other proprietary tools
- Creating our own tooling

How to choose?

~~One size fits all~~

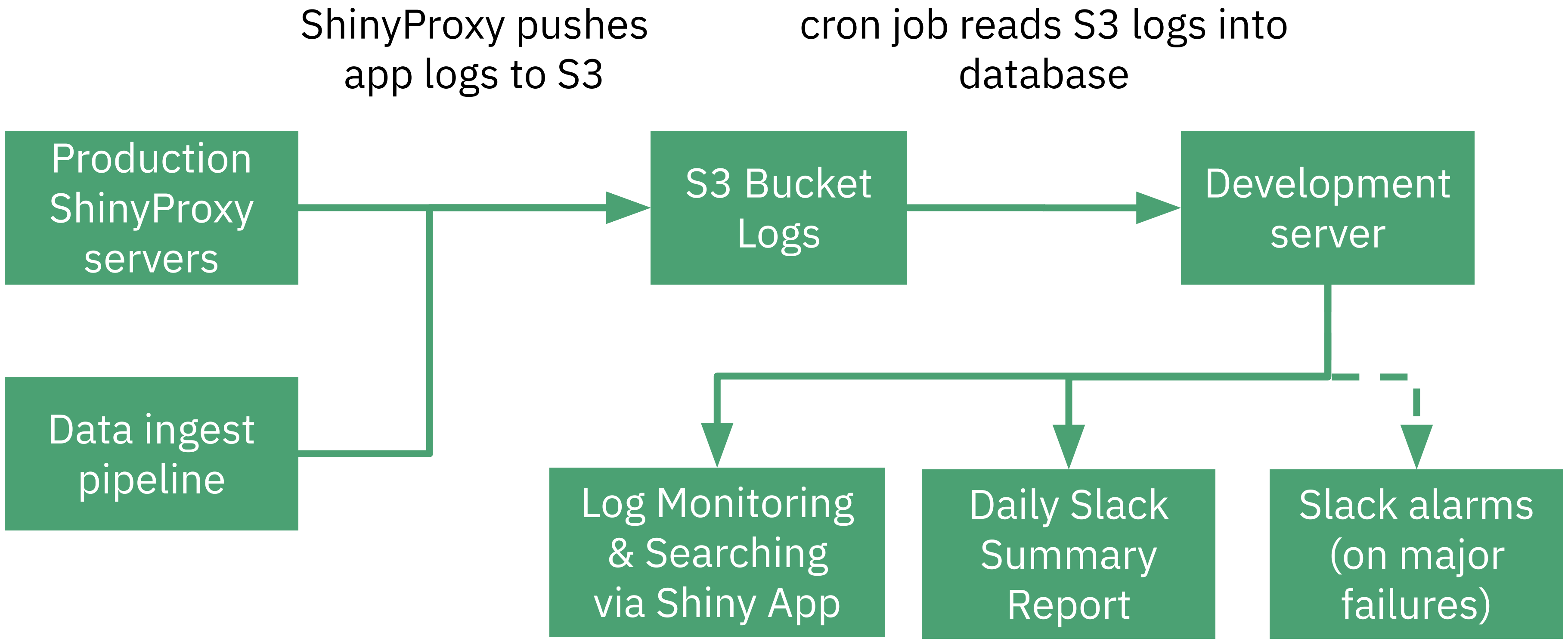
The right solution is the one that:

- fits within the existing architecture
- has the correct tradeoff of simplicity / features
- is easy to implement
- ...actually gets used!

"It is also possible to store the container log files in an S3 back-end using the following settings..."

shinyproxy.io/documentation/configuration/#s3-back-end

Implementing ShinyProxy → S3 logging


ShinyProxy pushes app logs to S3
cron job reads S3 logs into database
Production ShinyProxy servers
Data ingest pipeline
S3 Bucket Logs
Development server
Log Monitoring & Searching via Shiny App
Daily Slack Summary Report
Slack alarms (on major failures)

How might we outgrow this system?

- Ingesting logs from other (non-R / non-JSON) systems
- Inability to use one main S3 bucket for all logs
- Log volume increases beyond capacity
- Monitoring latency/speed requirements increase
- Monitoring metrics/analysis demands increase
- ...?

Until we outgrow it, this system works for us!

Takeaways

Logging...

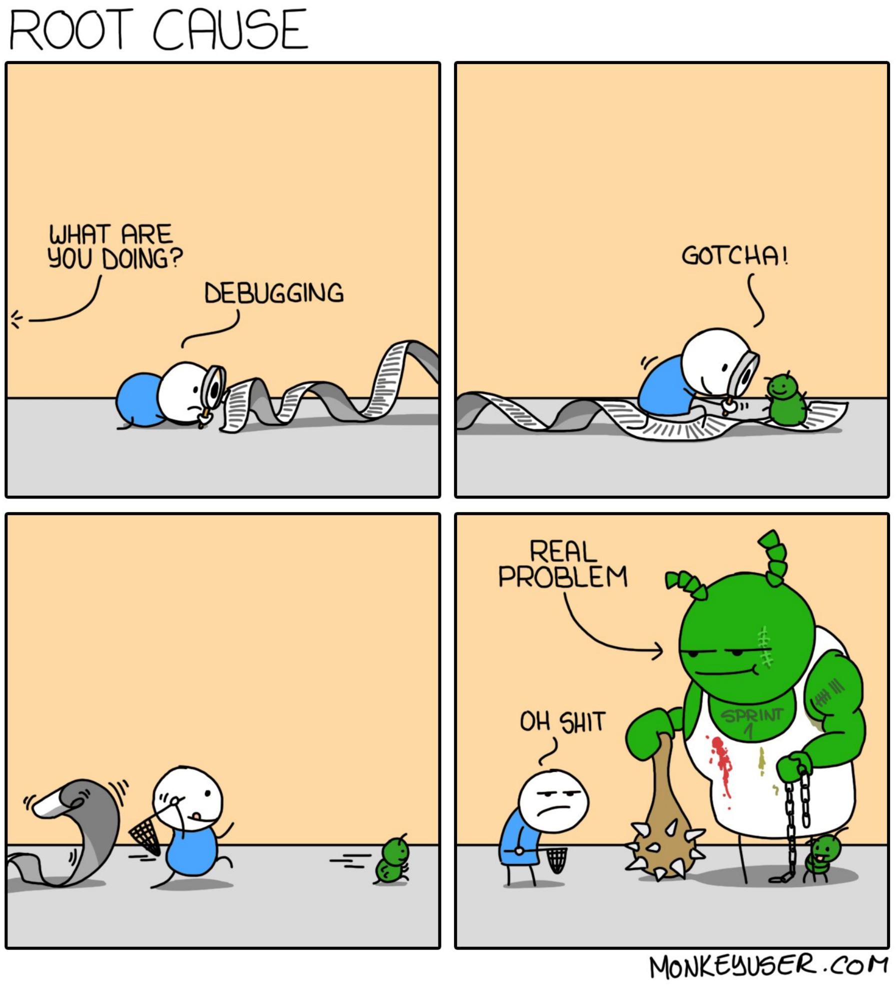
is about being proactive for future debugging

Logging…

generates data that can guide improvements

Logging…

is a production system that you can improve on iteratively



https://www.monkeyuser.com/2018/root-cause/

Logging...

might not always be enough to save you from a painful debugging experience

Thank you!

tanho.ca/logging-shiny
github.com/tanho63
tan@tanho.ca

Resources and Notes

R-specific tooling:

- {logger} pkg by Gergely Daróczi
- Shiny's official docs for options(shiny.error)
- shinymetrics by John Coene

General principles:

- A Guide To Application Logging - Andre Rabold
- Structured logging - Reflectoring
- What Should I Log In My Application - Loupe