CS130: Software Engineering

Lecture 8 Logging
Midterm Review

https://forms.gle/tH7W2JWsp5zPcxqj6

A word: Which is healthier, you or your server?

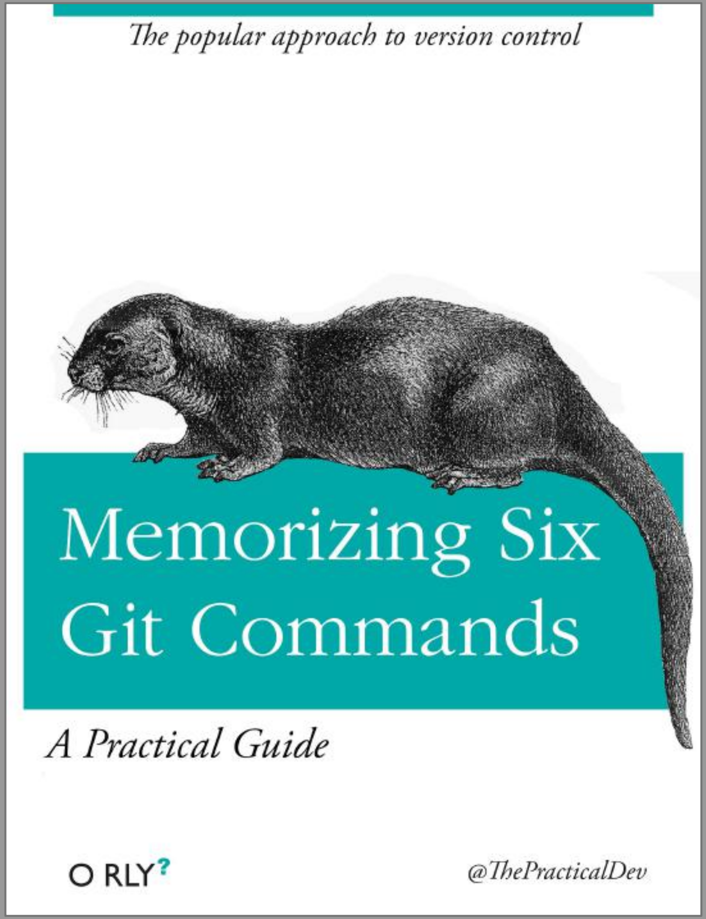
A tweet: What stats would you collect to improve your server's health?

A name: Best boba shop in SoCal?

Assignment 4: In progress

Notes: Source repos

- One repository going forward
- Only add source code / configs / data files to your repository
  - No built binaries
  - No generated files
  - No `.DS_Store` files
  - Use `.gitignore`!


The popular approach to version control
Memorizing Six Git Commands
A Practical Guide
O RLY?
@ThePracticalDev

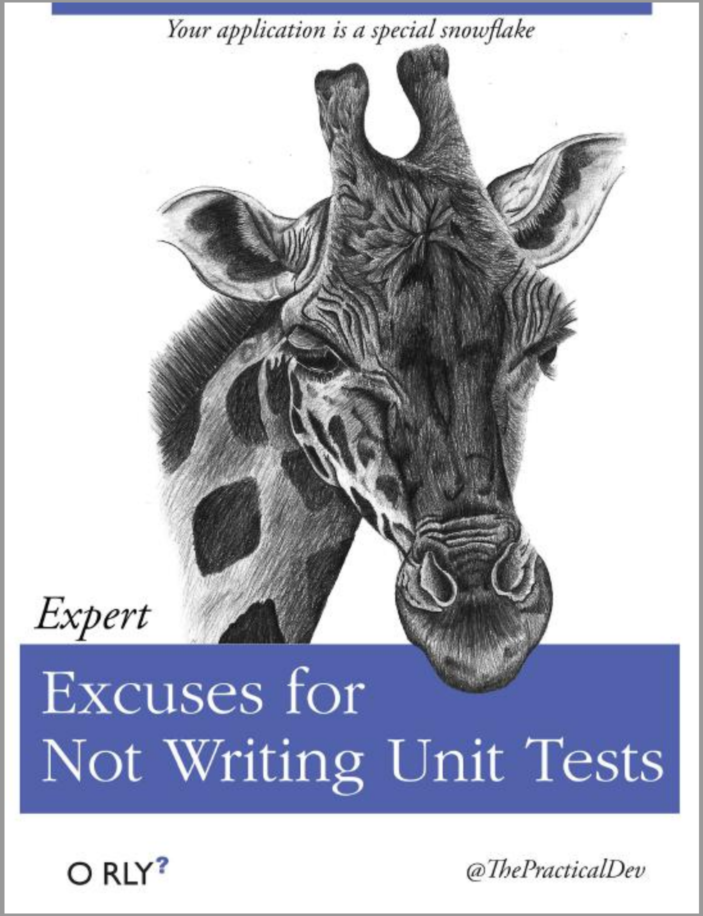
Notes: Testing

Unit tests

- Should be small, fast
- Should they bind to a network interface + port?
- Use dependency injection

Integration tests

- Should start server, kill server
- How to not bind to a port in use?


Your application is a special snowflake
Expert
Excuses for Not Writing Unit Tests
O RLY?
@ThePracticalDev

Notes: Development

- Build in development environment, run in development environment
  - Expose ports with `start.sh`
- Docker useful for testing deployment
- Google Cloud Build == Docker build
- Debugging
  - Bash: `#!/bin/bash -x`
  - C++: LLDB remote

Notes: Deployment

Google Cloud Platform

- One project per team
- DNS subdomains coming
  - Placeholder: `www` -> `127.0.0.1`
- Use tags to mark images to be used for grading
  - `:assignX`
- Update `:latest`
- No need to tag/deploy coverage image

Logging

Important questions

- Why to log?
- What to log?
- When to log?
- Where to log?
- How to log?

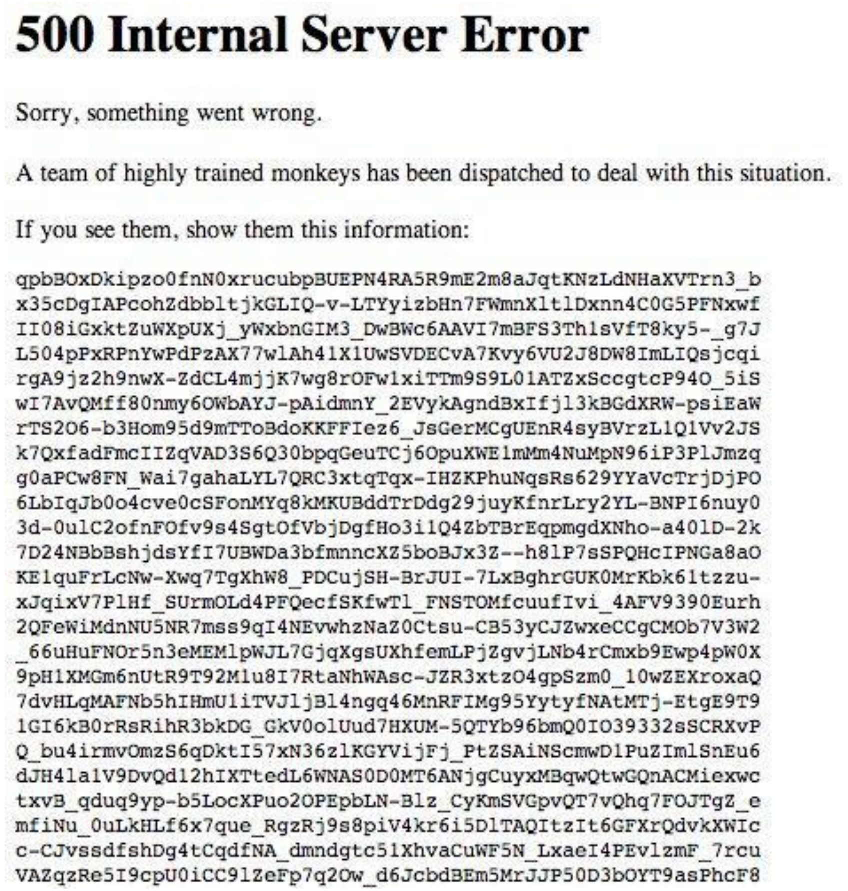
Less important

- Who to log?

Why log?

So a chemist, a physicist, and a software engineer are in a car going down a hill…

Why log?



500 Internal Server Error

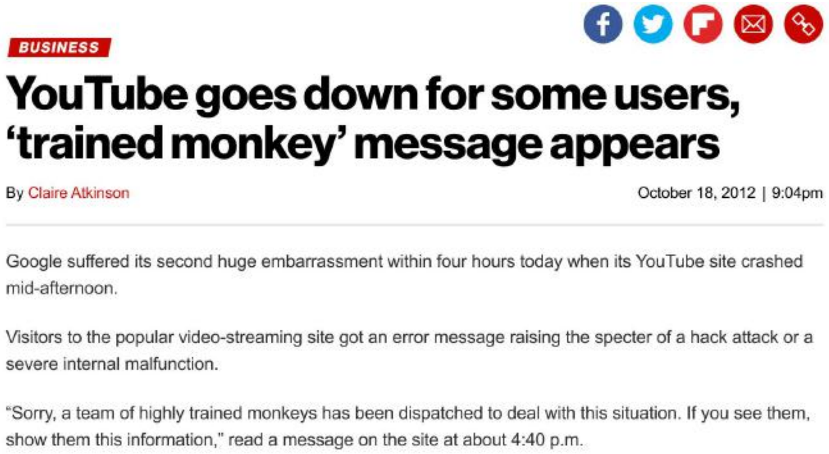
Sorry, something went wrong.

A team of highly trained monkeys has been dispatched to deal with this situation.

If you see them, show them this information:

```
qpbBOxDkipzo0fnN0xrucubpBUEPN4RA5R9mE2m8aJqtKNzLdNHaXVTrn3_b
x35cDgIAPcohZdbbltjkGLIQ-v-LTYyizbHn7FWmnXltlDxnn4C0G5PFNxwf
II08iGxktZuWXpUXj_yWxbnGIM3_DwBWc6AAVI7mBFS3ThlsVfT8ky5-_g7J
L504pPxRPnYwPdPzAX77wlAh41X1UwSVDECvA7Kvy6VU2J8DW8ImLIQsjcqi
rgA9jz2h9nwX-ZdCL4mjjK7wg8rOFw1xiTTm9S9L01ATZxSccgtcP94O_5iS
wI7AvQMff80nmy6OWbAYJ-pAidmnY_2EVykAgndBxIfjl3kBGdXRW-psiEaW
rTS2O6-b3Hom95d9mTToBdoKKFFIez6_JsGerMCgUEnR4syBVrzL1Q1Vv2JS
k7QxfadFmcIIZqVAD3S6Q30bpqGeuTCj6OpuXWE1mMm4NuMpN96iP3PlJmzq
g0aPCw8FN_Wai7gahaLYL7QRC3xtqTqx-IHZKPhuNqsRs629YYaVcTrjDjPO
6LbIqJb0o4cve0cSFonMYq8kMKUBddTrDdg29juyKfnrLry2YL-BNPI6nuy0
3d-0ulC2ofnFOfv9s4SgtOfVbjDgfHo3i1Q4ZbTBrEqpmgdXNho-a40lD-2k
7D24NBbBshjdsYfI7UBWDa3bfmnncXZ5boBJx3Z--h8lP7sSPQHcIPNGa8aO
KE1quFrLcNw-Xwq7TgXhW8_PDCujSH-BrJUI-7LxBghrGUK0MrKbk61tzzu-
xJqixV7PlHf_SUrmOLd4PFQecfSKfwTl_FNSTOMfcuufIvi_4AFV9390Eurh
2QFeWiMdnNU5NR7mss9qI4NEvwhzNaZ0Ctsu-CB53yCJZwxeCCgCMOb7V3W2
_66uHuFNOr5n3eMEMlpWJL7GjqXgsUXhfemLPjZgvjLNb4rCmxb9Ewp4pW0X
9pH1XMGm6nUtR9T92M1u8I7RtaNhWAsc-JZR3xtzO4gpSzm0_10wZEXroxaQ
7dvHLqMAFNb5hIHmU1iTVJljBl4ngq46MnRFIMg95YytyfNAtMTj-EtgE9T9
1GI6kB0rRsRihR3bkDG_GkV0olUud7HXUM-5QTYb96bmQ0IO39332sSCRXvP
Q_bu4irmvOmzS6qDktI57xN36zlKGYVijFj_PtZSAiNScmwD1PuZImlSnEu6
dJH4la1V9DvQd12hIXTtedL6WNAS0D0MT6ANjgCuyxMBqwQtwGQnACMiexwc
txvB_qduq9yp-b5LocXPuo2OPEpbLN-Blz_CyKmSVGpvQT7vQhq7FOJTgZ_e
mfiNu_0uLkHLf6x7que_RgzRj9s8piV4kr6i5DlTAQItzIt6GFXrQdvkXWIc
c-CJvssdfshDg4tCqdfNA_dmndgtc51XhvaCuWF5N_LxaeI4PEvlzmF_7rcu
VAZqzRe5I9cpU0iCC9lZeFp7q2Ow_d6JcbdBEm5MrJJP50D3bOYT9asPhcF8
```

Why log?


BUSINESS

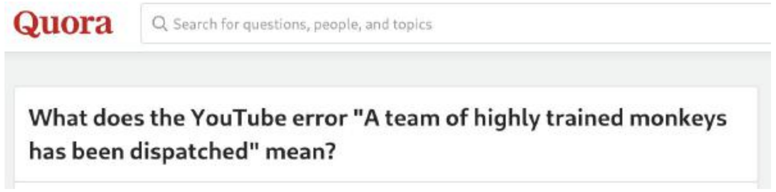
YouTube goes down for some users, 'trained monkey' message appears

By Claire Atkinson — October 18, 2012 | 9:04pm

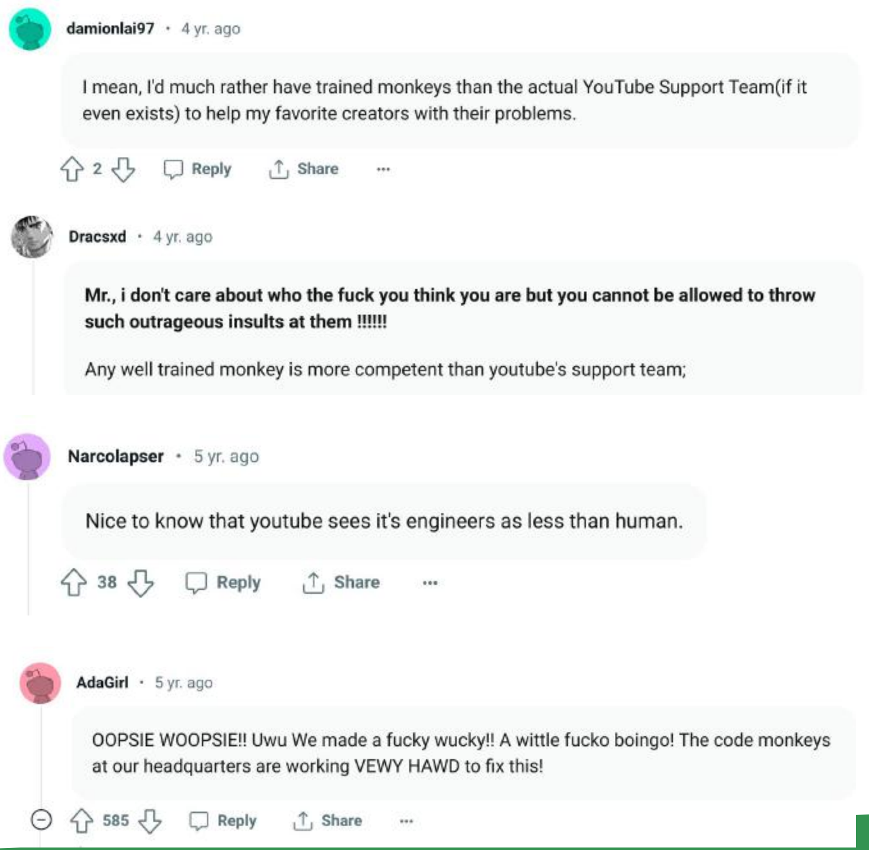
Google suffered its second huge embarrassment within four hours today when its YouTube site crashed mid-afternoon.

Visitors to the popular video-streaming site got an error message raising the specter of a hack attack or a severe internal malfunction.

"Sorry, a team of highly trained monkeys has been dispatched to deal with this situation. If you see them, show them this information," read a message on the site at about 4:40 p.m.


Quora

What does the YouTube error "A team of highly trained monkeys has been dispatched" mean?


damionlai97 · 4 yr. ago

I mean, I'd much rather have trained monkeys than the actual YouTube Support Team(if it even exists) to help my favorite creators with their problems.

2 · Reply · Share

Dracsxd · 4 yr. ago

Mr., i don't care about who the fuck you think you are but you cannot be allowed to throw such outrageous insults at them !!!!!!

Any well trained monkey is more competent than youtube's support team;

Narcolapser · 5 yr. ago

Nice to know that youtube sees it's engineers as less than human.

38 · Reply · Share

AdaGirl · 5 yr. ago

OOPSIE WOOPSIE!! Uwu We made a fucky wucky!! A wittle fucko boingo! The code monkeys at our headquarters are working VEWY HAWD to fix this!

585 · Reply · Share

Why log?

Take a guess!

- Development
- Debugging
- Security
- Monitoring
- Usage insights

How to log?

No need to guess:

```
#include <boost/log/trivial.hpp>
#include <iostream>

int main(int, char *[]) {
  BOOST_LOG_TRIVIAL(info) << "This is some info";

  BOOST_LOG_TRIVIAL(warning) << "Not great...";

  BOOST_LOG_TRIVIAL(error) << "OH NOOOO";

  BOOST_LOG_TRIVIAL(fatal) << "eff this I'm out";

  return 0;
}
```

- Boost.Log
- C++ string streams
- Severity
- Message

When to log?

Every time interesting things happen!

What's happening in your webserver?

- Ready to serve
- Serving a request
- Errors
- Shutting down

What to log?

Basic purpose of web server logs:

- Verify correct usage
- Detect abuse

But what specific info? No answers are wrong!

Per request:

- Timestamp
- Thread ID
- IP of request
- Severity
- … and more!

Where to log?

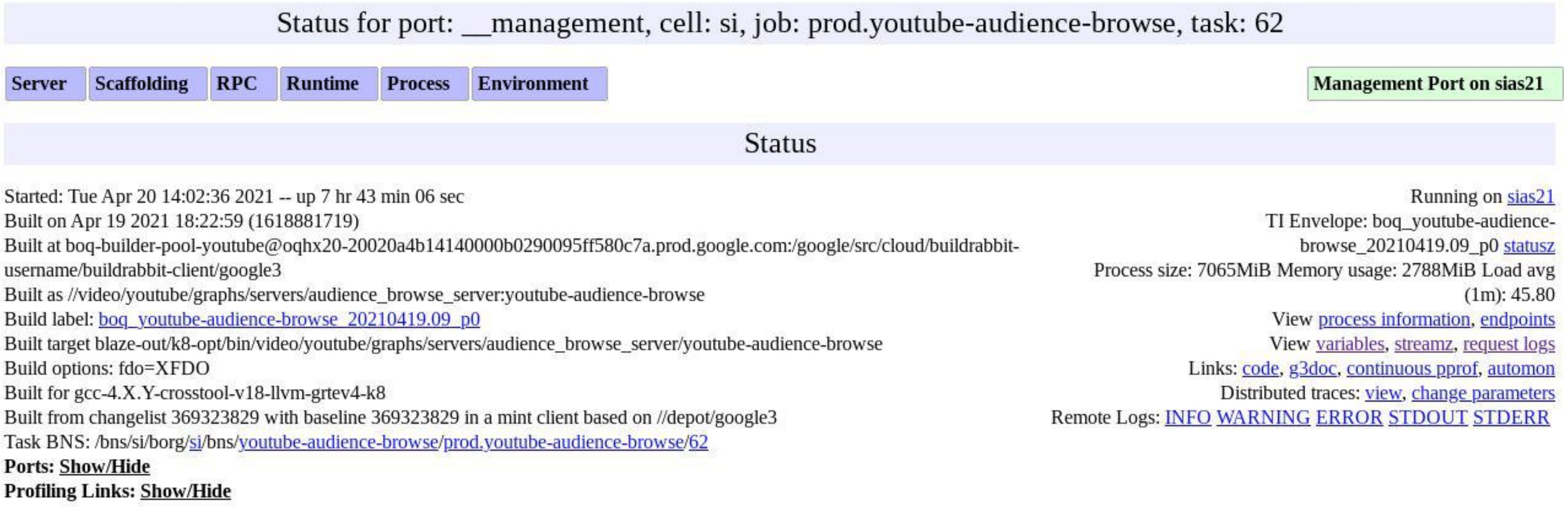
Goals for statement logs:

- Inspect manually
- Parse with tools
- Do not fill disk
- Persist after restart or crash
- Resilient to hardware failures

Log sinks:

- Console stdout / stderr
- File(s) on disk
- Remote servers

Status pages (human readable)



Status for port: __management, cell: si, job: prod.youtube-audience-browse, task: 62

Status

Started: Tue Apr 20 14:02:36 2021 -- up 7 hr 43 min 06 sec
Built on Apr 19 2021 18:22:59 (1618881719)
Built at boq-builder-pool-youtube@oqhx20-20020a4b14140000b0290095ff580c7a.prod.google.com:/google/src/cloud/buildrabbit-username/buildrabbit-client/google3
Built as //video/youtube/graphs/servers/audience_browse_server:youtube-audience-browse
Build label: boq_youtube-audience-browse_20210419.09_p0
Built target blaze-out/k8-opt/bin/video/youtube/graphs/servers/audience_browse_server/youtube-audience-browse
Build options: fdo=XFDO
Built for gcc-4.X.Y-crosstool-v18-llvm-grtev4-k8
Built from changelist 369323829 with baseline 369323829 in a mint client based on //depot/google3
Task BNS: /bns/si/borg/si/bns/youtube-audience-browse/prod.youtube-audience-browse/62
Ports: Show/Hide
Profiling Links: Show/Hide

Running on sias21
TI Envelope: boq_youtube-audience-browse_20210419.09_p0 statusz
Process size: 7065MiB Memory usage: 2788MiB Load avg (1m): 45.80
View process information, endpoints
View variables, streamz, request logs
Links: code, g3doc, continuous pprof, automon
Distributed traces: view, change parameters
Remote Logs: INFO WARNING ERROR STDOUT STDERR

Status pages (machine readable)

```
build-timestamp "Built on Apr 19 2021
18:22:59 (1618881719)"
build-timestamp-as-int 1618881719
build-changelist 369323829
build-target
blaze-out/k8-opt/bin/video/youtube/...
argv "./server/youtube-audience-browse ..."
gplatform
gcc-4.X.Y-crosstool-v18-llvm-grtev4-k8
debug-binary 0
cpu-speed 1000000000
memory-address-size 64
hostname sias21
start-time "Tue Apr 20 14:02:36 2021"
uptime-in-ms 27787733
virtual-process-size 7409201152
memory-usage 2926772224
heap-size 3007819008
log-errors 270380
```

Useful for monitoring (assignment 8)

Notes on severity

- Usually multi-level:
  - `trace/verbose/debug`
  - `info`
  - `warning`
  - `error`
- Specify minimum severity level for log destinations
  - `logLevel=info` includes `info`, `warning`, and `error`

Consider:

- Only log `trace` during debugging
- Logging `info` and higher to files
- Logging only `error` to console

Logging to disk

How do you avoid filling up the disk?

Rotate log files!

To cap usage at 100MB:

- When log file > 10MB:
  - Move logname.log to logname_YYYYMMDDhhmm.log
  - Delete oldest log if 10 or more exist

Logs on Docker

- Console logging works out of box
- Captures stdout, stderr
- `docker logs [-f] ${CONTAINER}`

Logs on Google Cloud

- Console logging works out of box
- Captures stdout, stderr
- Find “Logs Explorer” in Cloud UX
- Supports monitoring and alerting based on logs (assignment 8)

Log Joining

- In a large system, one user request might branch out to many microservices
- How could we trace activity related to that request across the system?
  Use a common key!
    - Request Id: UUID or Hash(request)

- How could we sort all the events?
  Gotta make a timestamp, but each server has their own clock!
    - <machine_ip, process_id or thread_id, system time>
        - helpful, but not perfect
    - Lamport clock (add +1 to the incoming request)
        - is that +1 globally? or +1 only for events about a specific request_id?
        - doesn’t provide wallclock latency

Logging vs Privacy

Be careful what you log!

The cost of ignoring privacy:

- Google Street View cars "accidentally" collected 600GB of unencrypted WiFi data from 2006 to 2010
- Google settled consolidated lawsuit in 2019 for $13 million
- Real damage to Google's reputation

```
# TODO: Clear privacy considerations
# with product counsel
```

https://www.cnn.com/2019/07/22/tech/google-street-view-privacy-lawsuit-settlement

There are laws now...

Since 2018 companies must:

- Obtain consent
- Report data breaches
- Allow takeout
- Right to be forgotten
- Design with privacy

Privacy challenges

- Must delete all data N days after a user deletes their account
- Many logging systems designed to be unmodifiable
  - Appending data is *fast*
  - Finding and removing arbitrary data in the middle is **slow**
- Systems with PII must now pay attention to account deletions
- What about tape backups?

Hard analytics questions:

- How many **visits** did we see last year?
- How many **unique visitors** did we see last year?

How can we answer these?

Solutions: Aggregation

- Aggregated data is generally not personally identifiable
    - Some exceptions to this
- Derived counts and stats are OK to keep indefinitely

So how many **visits** did we see last year?

- Aggregate logs at the end of the day to record the number of visits that day
- Sum up daily visits over desired time period

Solutions: Pseudononymization

- Hash the PII and store the hash to count uniques
- HMAC-SHA2 hashes are hard to reverse
    - For now!
- Real anonymization is hard!

So how many **unique visitors** did we see last year?

- Hash the visitor ID and log just the hashed ID to permanent logs
- Count the number of unique hashed IDs over the given time period

Disclaimer: this may not be a legally compliant solution but you get the idea

Privacy in practice

- Privacy needs to be a first-class concern
- Privacy needs to be considered early
- Privacy and security go hand in hand
- Data access controls are critical
    - **We can't read your gmail!**

New-ish processes:

- Privacy design documents
- Privacy sections in technical design documents
- Privacy review councils
- Privacy approvals for launch

Error Handling and Recording Information

You thought you were finished?

- Things working is just the first step.
- Now for lots of thinking about every possible error condition.
- The good news: Your code is working, much easier to add code now.

But how should you think about errors?

Example 1: What’s wrong with this code?

```
void HandleLookup(const Request& request) {
  if (request.lookup_string().empty()) {
    LOG(FATAL) << “Received empty request lookup”;
  }
  ...
}
```

Example 2: How about this?

```
int main(int argc, char** argv) {
  Flags flags = ProcessFlags(argc, argv);
  unsigned short port = 0;
  // We require the port be specified in the flags.
  if (!flags.parseInt(“port”, &port)) {
    // If we fail to parse the port from flags, just use 8080.
    port = 8080;
  }
}
```

Example 3: This?

```
void BigFastCalculation() {
  for (int i = 0; i < BIG_NUMBER; ++i) {
    LOG(INFO) << “Processing iteration ” << i;
    // Calculation…
  }
}
```

Example 4: How about this?

```
void Server::Init() {
  ReadConfiguration();
  // Could take up to several seconds.
  ConnectToBackends();
  ListenOnConnections();
  // Good to go!
  while (ProcessRequest());
}
```

Example 4: How about this?

```
void Server::Init() {
  ReadConfiguration();
  // Could take up to several seconds.
  ConnectToBackends();
  ListenOnConnections();
  LOG(INFO) << "Listening for requests";
  while (ProcessRequest());
}
```

Error handling strategies

- Crashing
- Returning failure responses from requests
- Recovering from errors
- Recording information

- When is it good to crash?
- When is it good to surface errors to users?
- When is it good to hide errors from users?

When to crash/fail

- Crash? Seriously?
  - Fatal errors can be useful
  - Good for client apps or entire servers only if you can't recover
  - Worth taking **some** steps to avoid for high-availability servers. (But even there, redundancy may be preferred.)

- Can you fail just one request/operation?
  - "Light" version of crashing

Aside: How to handle errors

Many different strategies:

- Boolean success return value
    - Requires actual return value in parameter
    - Doesn't allow for easy detailed return values
- Error codes
    - Either returned directly or through out param a la boost
- Exceptions
- "Global" (thread local) errno variable (not recommended)
- "StatusOr" -- return a class that contains an error status, or success + return value

Aside: How to handle errors

- One recommendation: “Translate” the error on the way up.
    - For example, how many times have you seen a website tell you SQLException?

- What should the user see?
    - And what should you do with the SQLException?

The difficulties of recovering from errors

Recovery:

- Seems like a good idea, but is it worth the effort?
  - Bailing out is often a great idea in comparison!
- Requires careful design
  - Handling errors in complex ways greatly increases the state space your execution can inhabit.
- Will likely only work if it's tested well, or exercised often enough for you to notice how it works.
- Needs to be thought about system-wide

Midterm Preview

Impact

- 15% of your grade
- 1 assignment → <10% of your grade.

Scope

- Lectures 1-8
- Assignments 1-4

Format

- In class
- Expected time 1 hour
- Short answers
- Closed book / closed notes / closed internet

Topics we've covered

- Source control
- Testing
- Code reviews
- Tools for web server development
- Build systems

- Deployment
- Refactoring and debugging the web server
- Testability
- Static analysis
- Logging and exception handling

Source control

- We discussed and compared different revision control systems
    - How they work
    - What properties they have
- You've been using git

Testing

- Picking good test cases
- Unit testing, using fixtures and mocks
- Refactoring for testability
- Integration testing
- Other kinds of testing

Code reviews

- Why and how to do code reviews
- In-class code reviews

Webserver development

- You've started developing a web server

Build systems

- CMake
- Google's build system
- Similarities, differences, tradeoffs

Static and runtime analysis

- How they work/their applications
- What kinds of problems you can catch with each

Exception handling

- Crashing
- Logging (error logs, request logs, etc.)
- Communicating errors up the stack.

https://bit.ly/3xKBURe

We got a midterm coming...
A word: Are you prepared?
A word: When will you prepare?
A list: What do you need to review?

Watch your stress stats.
Stay healthy!