Sam Clarke

Monitoring and Testing web Applications in the Wild

Introduction

- Something's not right, but what?
- We can do better.

Introduction

The **On-board Diagnostic (ODB) II** vehicle interface.

“Gives the vehicle owner or repair technician access to the status of the various vehicle subsystems.”

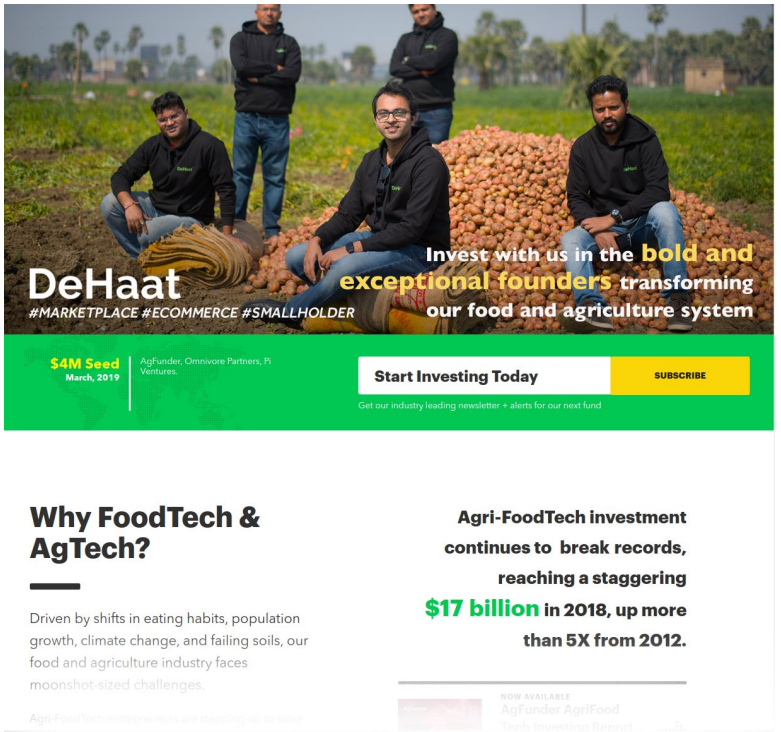
https://en.wikipedia.org/wiki/On-board_diagnostics

Who am I?

Sam Clarke

Senior Developer at

AgFUNDER



Summary

- Why monitoring matters
- What can be monitored
- What happens when an issue occurs
- Takeways

Definitions

- **Dev** - the construction site
- **Production** - user-facing software
- **Monitoring** - continuous analysis
- **Metric** - something measurable
- **Load** - quantity of requests

Why this talk?

Your app will fail.

Why this talk?

"No battle plan survives first contact with the enemy."

- Helmuth von Moltke the Elder, 1871

Dev != Production

“It worked in Dev.”

- Most developers at some point.

Dev != Production

- **Debug** mode is off (it is off right?)
- Multiple **users**, multiple concurrent **requests**, multiple **platforms**.
- **Network latency** (clients, dbs, APIs)
- Production architecture may be significantly more **complex** (containers, clusters, permissions etc.)

Dev != Production

Staging environments close the gap between **Dev** and **Production**.

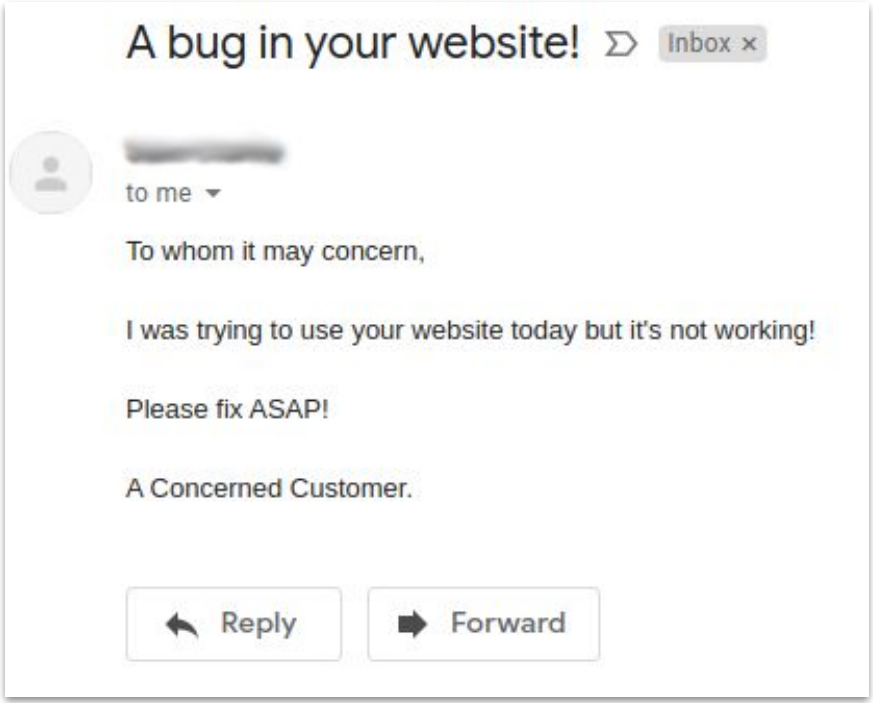
The value of Monitoring

The value of Monitoring

Insight requires Feedback

The value of Monitoring

- This is **not** considered high quality feedback.
- We want to be aware of and fix an issue **before** it gets to this stage.



The value of Monitoring

- Catch warning signs of problems before they **escalate**.
- Improve **response** times.
- Create a **record** of performance over time.

What can go wrong?

- **Web server** - timeouts, file handling
- **Application** - bugs and stuff
- **Database** - data corruption, hardware
- **Host** - overloading, crashing, network
- **Caching***

And the Best Monitoring Tool is ...

The value of Monitoring

Onsite or Offsite?

What to Monitor?

- **Logging** - application code
- **Load** - host performance at scale
- **Uptime** - availability
- **Metrics** - system resources, response times
- **Performance** - all of the above

Logging

- **What** should you log, what format?
- **Where** to store logs - local log files, systemd binaries, offsite.
- **Who** will read these?

The 5 Levels of Logging

1. **DEBUG** - Detailed information, typically of interest only when diagnosing problems in a debugging session.
2. **INFO** - Confirmation that things are working as expected.
3. **WARNING (default)** - An indication that something unexpected happened, or indicative of some problem in the near future (e.g. ‘disk space low’). The software is still working as expected.
4. **ERROR** - Due to a more serious problem, the software has not been able to perform some function.
5. **CRITICAL** - A serious error, indicating that the program itself may be unable to continue running.

Source: https://docs.python.org/3/howto/logging.html

Python Logging Good Practices

- Use **__name__** as the logger name

```
logger.getLogger(__name__)
```

- Dump **tracebacks** in log message

```
logger.error('S3 API call failed', exc_info=True)
```

Alerting

Getting notified when stuff breaks.

Alerting - Policies

- What is your alerting mechanism?
- **Who** should be alerted when a problem arises?
- Is there a **chain of escalation**?
- How is **urgency** determined?
- Is it clear what **action** should be taken?

Alerting - Fixing

- Is there sufficient **context** to debug the problem?
- Avoid **desentizing** your team by over-alerting (alerting fatigue).
- What happens **post-fix**?

Traffic and Load

- **Traffic Monitoring** can aide historical debugging.
- **Load Testing** can help identify bottlenecks and determine baseline host capacity.
- Are you load testing the most **resource-intensive** APIs?
- What **metrics** are being gathered during testing?

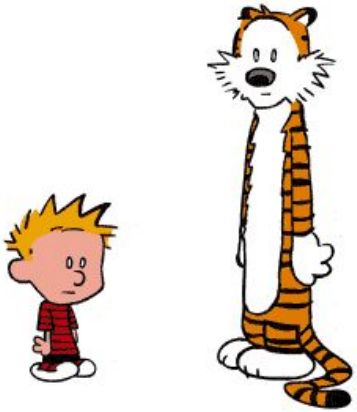
Monitoring is your Friend

Application Monitoring as Continuous Testing.

Takeaways

1. Have a clear idea of **what** you want to monitor.
2. Quality **logs** provide context.
3. Consider your response **policies**.
4. **Third party** monitoring tools can save time and increase insight.

We can do better.



@samclarkeg