# SWEN438 - DevOps

## Lecture 12 - Logging

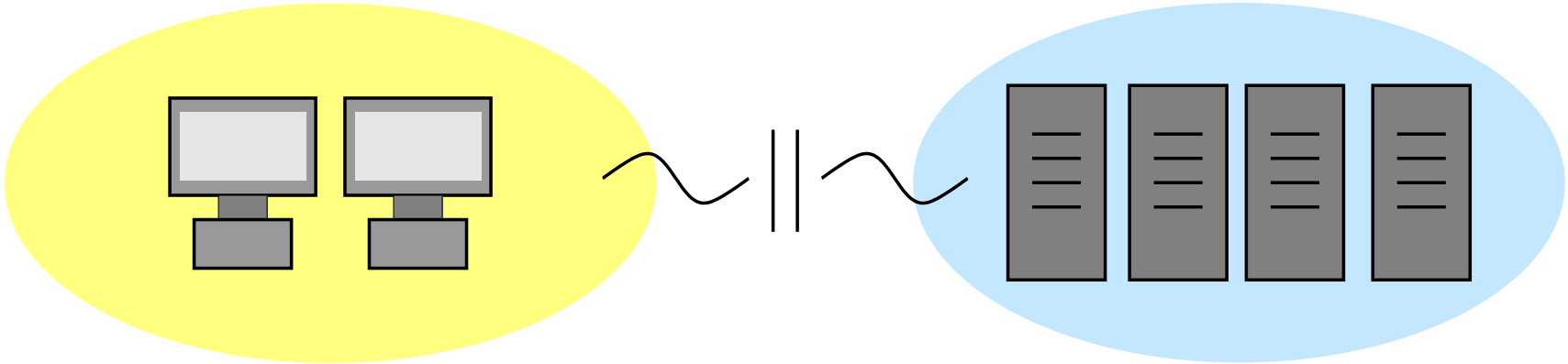
Dr David J. Pearce

*School of Engineering and Computer Science*
*Victoria University of Wellington*

*“Yeah, well, all the users know is that they can’t connect to Data Hub,” he says ... “We need the production logs for Data Hub and its connectors to see if they’re handling traffic or if they’ve crashed,”*

*– The Unicorn Project*

# Why Logging?

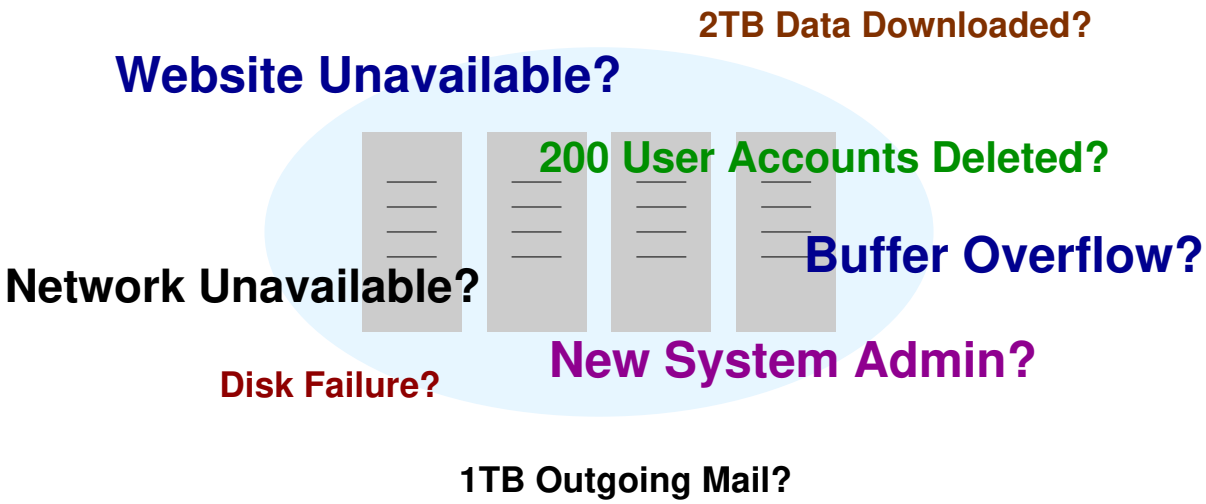


- Perhaps cannot get **physical access** to server.
- Perhaps **event** happened in the past.
- Perhaps need **detailed** information in moments before event.

*Choosing the* **right logging level** *is important. Dan North, a former ThoughtWorks consultant who was involved in several projects in which the core continuous delivery concepts took shape, observes, "When deciding whether a message should be ERROR or WARN, imagine being woken up at 4 a.m. Low printer toner is not an ERROR."*

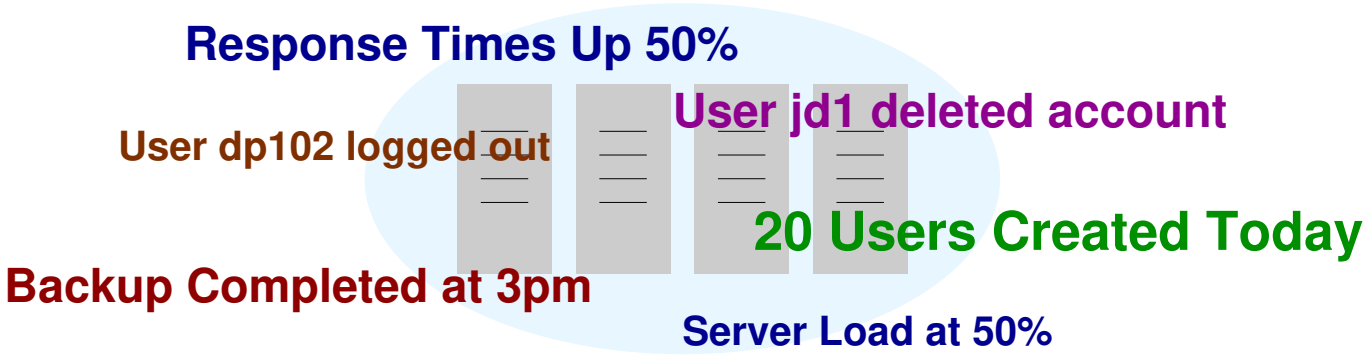
*– The DevOps Handbook*

# Log Levels (Concerning)




- **FATAL**. Captures unrecoverable events arising in production. *Should be brought to immediate attention (e.g. alert)*
- **ERROR**. Captures unexpected errors arising in production, but not fatal. *Should be brought to attention (e.g. email alert).*
- **WARN**. Captures something unexpected which is not an immediate concern. *Persistent warnings should be investigated.*

# Log Levels (Informative)




- **INFO**. Captures normal operation, and no investigation required. *Provides important background in timeline leading up to an event.*
- **DEBUG**. Captures just about anything that happens. *Use for development / debugging, but not production.*
- **TRACE**. Fine-grained information that captures almost everything. *Use for development / debugging, not production.*

# **Case Study**: LinkedIn

*"Is the hacker still in the network? What did they steal? How did they get in? Can we block them from getting in again? All these questions needed immediate answers. LinkedIn engineers and security team took over a conference room and called it a War Room. Something like forty to sixty people from LinkedIn were all working on this incident. They were flying in from foreign countries to help. They had the security team involved hunting through events and logs, looking for evidence."*

*– DarkNet Diaries (Episode 86)*

- **March, 2012**. Hacker steals 6.5million passwords.
- **May, 2012**. Hacker logs in as LinkedIn user(s) after cracking password hashes.
- **June, 2012**. LinkedIn finally notices (see above).

# Log Events

*Scott Prugh, Chief Architect and Vice President of Development at CSG, said, "Every time NASA launches a rocket, it has millions of automated sensors reporting the status of every component of this valuable asset. And yet, we often don't take the same care with software"*

*– The DevOps Handbook*

- **Authentication Decisions** (e.g. logon, failed logon, logoff, etc)
- **Privileged changes** (e.g. admin password change, data deleted)
- **System Health** (e.g. high CPU utilisation, low disk space)
- **Threats** (e.g. suspicious activity, unauthorized access, etc)

# Log Messages

```
10:04:05.775 [main] INFO  j.s.u.TypicalUser - Christian attempting to save "Trie.java"
10:04:05.776 [http] TRACE j.s.Simulation - Christian:save(Trie.java,// Copyright ...
10:04:05.877 [http] TRACE j.s.Simulation - 1fe0:login(Wayne,dallas)
10:04:05.877 [http] TRACE j.s.Simulation - 1fe0:succeeded creating new account
10:04:05.877 [main] INFO  j.s.Simulation - Wayne successfully created account
10:04:05.877 [main] INFO  j.s.u.RandomUser - Samuel attempting to load "Main.java"
10:04:05.878 [http] TRACE j.s.Simulation - Samuel:load(Main.java) => public class Main ..
10:04:05.878 [main] INFO  j.s.Simulation - ??? opening connection
```

- **Assumption:**
  - » Assume writing for situation where log **only** source of information.
  - » For example, if production server running in cloud crashes.
- **Context:**
  - » Must include sufficient context to understand message.
  - » For example, just reporting “Exception Occurred” is not helpful!

# **Case Study**: LinkedIn (Cont'd)

*"... Lots of different teams were pulling logs and saving them for incident responders to comb through. It's best to have logging turned on so you can sort of go back in time and see what happened where. One problem though is that there's now a* **lot of logs** *to go through and if you think about the millions of users who are on the site every day, trying to find a needle in a haystack is tricky."*

*– DarkNet Diaries (Episode 86)*

# Log Size

```
20:22:10.140 [main] TRACE j.Main - Attempting to start JavaFiddle on port 8080
20:22:10.161 [main] WARN  j.Main - Port 8080 in use by another application.
20:22:10.161 [main] ERROR j.Main - Failed starting JavaFiddle (all ports in use)
20:22:19.344 [main] TRACE j.Main - Attempting to start JavaFiddle on port 8080
20:22:19.360 [main] TRACE j.Main - JavaFiddle running on port 8080.
10:03:23.770 [main] INFO  j.s.Simulation - Performing health check on http://localhost:8080
10:03:23.936 [main] WARN  j.s.Simulation - Unable to contact server (http://localhost:8080)
10:03:28.938 [main] WARN  j.s.Simulation - Unable to contact server (http://localhost:8080)
10:03:33.941 [main] WARN  j.s.Simulation - Unable to contact server (http://localhost:8080)
10:03:38.943 [main] WARN  j.s.Simulation - Unable to contact server (http://localhost:8080)
10:03:43.945 [main] WARN  j.s.Simulation - Unable to contact server (http://localhost:8080)
10:03:48.947 [main] WARN  j.s.Simulation - Unable to contact server (http://localhost:8080)
10:03:53.950 [main] WARN  j.s.Simulation - Unable to contact server (http://localhost:8080)
10:03:58.953 [main] WARN  j.s.Simulation - Unable to contact server (http://localhost:8080)
10:04:00.018 [j.Main.main()] TRACE j.Main - Attempting to start JavaFiddle on port 8080
10:04:00.034 [j.Main.main()] TRACE j.Main - JavaFiddle running on port 8080.
...
```

- **Too Much**. Harder to analyse **large** log files to find clues.
- **Too Little**. May be **unable to diagnose** problem when it arises!

# Log Analysis

```
10:04 ... Christian:save(Trie.java,// Copyright ...
10:04 ... 1fe0:login(Wayne,dallas)
10:04 ... 1fe0:succeeded creating new account
10:04 ... Wayne successfully created account
10:04 ... Samuel attempting to load "Main.java"
10:04 ... Samuel:load(Main.java) => public class Main {
```

- **Automated** log analysis can help but requires suitable logs.
- Structured log formats (e.g. JSON or XML) make life **easier**!

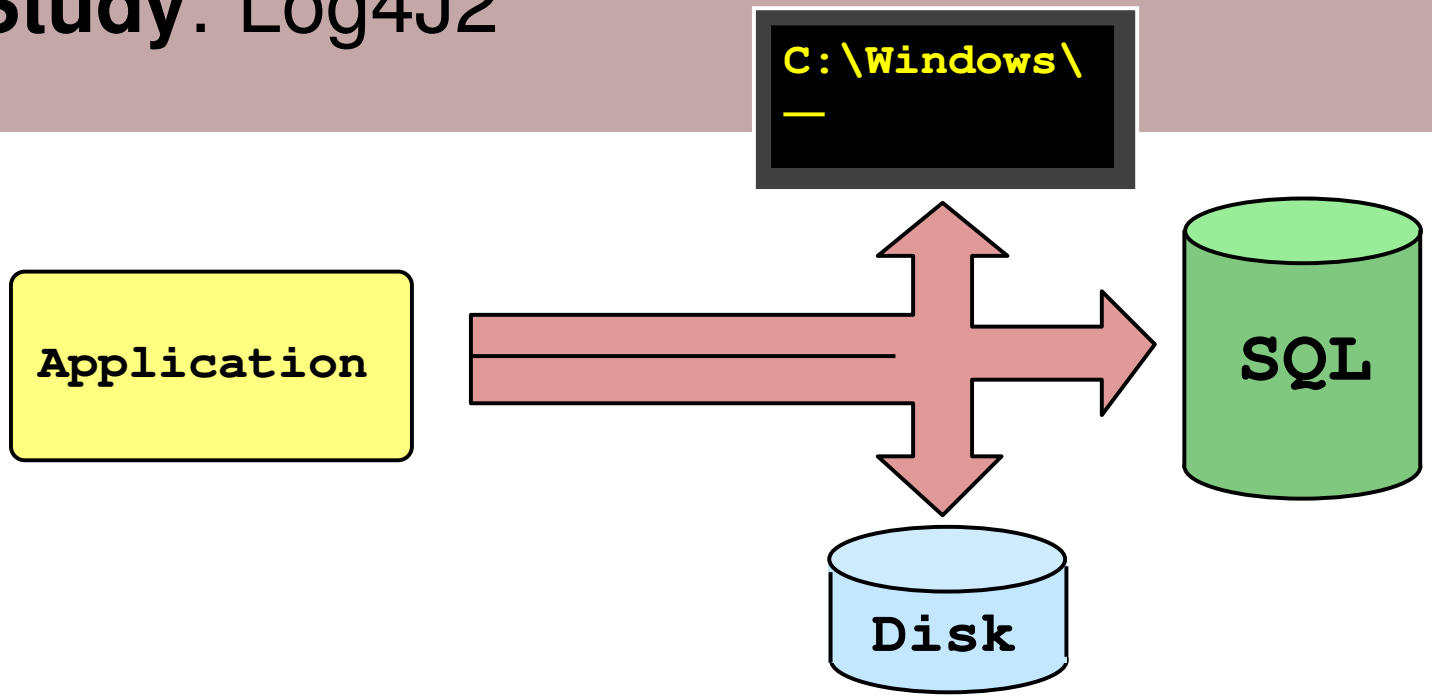
*What's wrong with the log format above?*

# **Case Study**: Log4J2


- **Configuration** specified in `log4j2.xml` file.
- Can log **simultaneously** to console, disk, SQL and network.
- Can generate logs in **different** formats (e.g. JSON).
- Can **customize** formats for entries in the log.

[1] Image Copyright © Apache Software Foundation, Apache 2.0 License

# **Case Study**: Log4J2




```
Logger logger = LogManager.getLogger("Main");
...
logger.warn("User reached storage quota");
...
logger.fatal("Insufficient disk space",e);
```

# **Case Study:** Log Rotation

*"... log rotation is an automated process used in system administration in which log files are compressed, moved (archived), renamed or deleted once they are too old or too big (...). New incoming log data is directed into a new fresh file (at the same location). The main purpose of log rotation is to restrict the volume of the log data to avoid overflowing the record store ..."*

*–Wikipedia*

- **Log4J2 `RollingFileAppender`**:
  - » Can automatically **rotate** log based on e.g. size or time.
  - » Can automatically **compress** archived log files.
  - » Can automatically **delete** files after amount of time.

# Measurement

*“By transforming logs into* **metrics***, we can now perform statistical operations on them, such as using anomaly detection to find outliers and variances even earlier in the problem cycle”*

*– The DevOps Handbook*