Introduction to logging

bit.ly/csse6400-logging

email: contact@larene.dev

Outline

- Start downloading dependencies for the practical.
- Introduction to logging
- Practical exercise - logging to Seq logging server

Download prerequisites

Download and install prerequisites

1. Docker
2. .NET 6.0 SDK Installer (not Runtime)
3. Postman App

email: contact@larene.dev

What is logging?

What is logging?

- Keeping a log of what happens in a program or OS
- In its simplest form, it's writing strings or messages to a file (and that file can be STDOUT)
- Diagnostic logging - low-level program-related logs
  - function calls
  - errors
  - health check
- Audit logging - business logic logs
  - when a user has logged in
  - password changed
  - purchase order made
- Transaction logs (database specific)

email: contact@larene.dev

Logging in practice

- Logging to the console (stdout) — you've probably done lots of this
  - print("some string") in Java or Python
  - Console.Writeline("some string") in .NET
  - console.log("some string") in JavaScript
- Log to a file
  - You might have seen logs on a server like "202205031120.log"
  - Use java.logging.util or Log4j2 or a bunch of other libraries...
- Log to a logging server
  - basically the same as logging to a file, but over HTTP (or some other transfer protocol)

email: contact@larene.dev

Why logging?

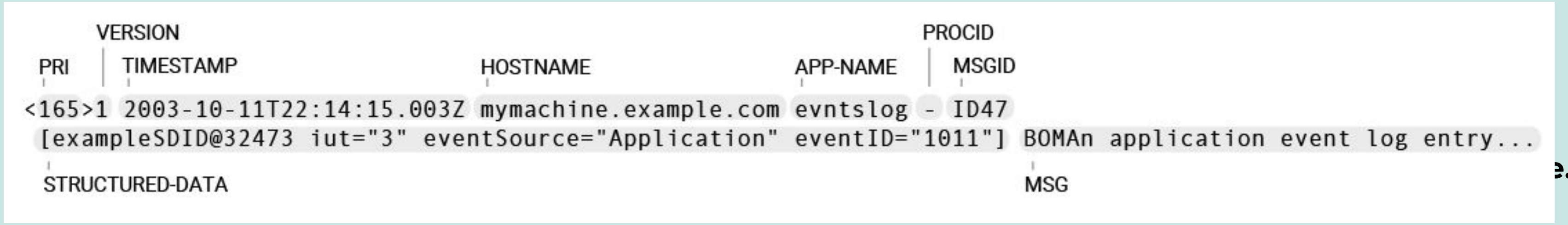
- Debug problems when programming
- Know if your program is running correctly in production
- Find out what your users are doing (e.g. for reporting, revenue, metrics)
- Troubleshooting for users (e.g. support desk)
- Find out related actions between services (log correlation)

Simple logging

- A log message, in its simplest form, is just a string
- Add a date and time at the start of that string
- Add a "log level" between the date and the message
  - e.g. debug, info, warning, error

```
2021-07-29 14:54:55.1623|INFO|New report created by user 4253
2021-07-29 14:56:19.2814|INFO|Report updated by user 257
2021-07-29 14:59:20.4211|INFO|User 478 deleted report
```

- There are standards that have been developed, such as "Syslog"



```
<165>1 2003-10-11T22:14:15.003Z mymachine.example.com evntslog - ID47
 [exampleSDID@32473 iut="3" eventSource="Application" eventID="1011"] BOMAn application event log entry...
```

Problems with simple logging

- For large scale systems, how do we efficiently search and filter?
  - grep, or find
  - regex
- How do we create reports?
  - with a lot of manual work
- No consistent pattern, very fragile

Structured logging

- From this (strings)

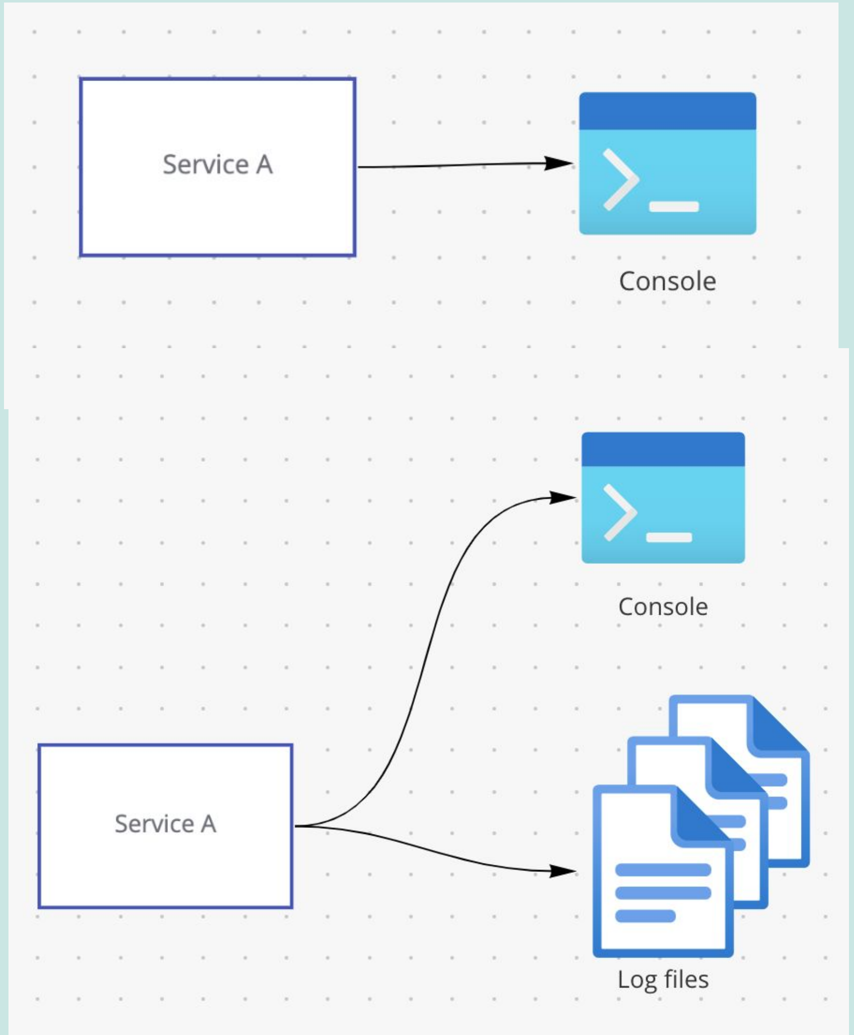
```
2021-07-29 14:54:55.1623|INFO|New report created by user 4253
2021-07-29 14:56:19.2814|INFO|Report updated by user 257
2021-07-29 14:59:20.4211|INFO|User 478 deleted report
```

- To this (objects)

```
{
    "TimeStamp": "2021-07-29 14:52:55.1623",
    "Level": "Info",
    "Message": "New report created",
    "UserId": "4253",
    "ReportId": "4567",
    "Action": "Report_Creation"
}
```

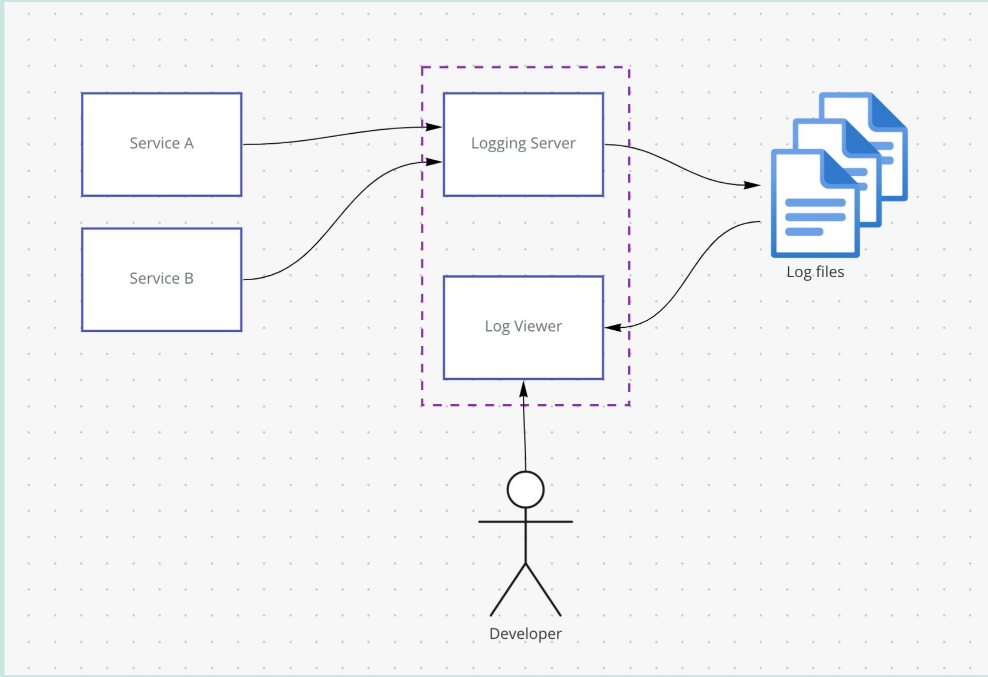
Logging architectures

- Service A has a **logging library** installed, logging libraries provide developers with special functions to make structured logging easier.
- Service A also has a **log "appender" or "sink"** installed, they are helpers are responsible for sending the logs where they're meant to go (e.g. console, or file, or server)



Logging architectures

- Service A and B both have the same **logging library** installed, as well as a **log sink/appender** that sends log messages to a **specific log server URL**.
- The **log server** receives logs from the applications, and stores the logs.
- The developer or internal staff view the logs on a **log viewer**. Log servers often have **search and analysis tools**, and often come packaged with the log server.



How to log effectively

- Don't log everything, the more noise, the more likely you'll miss something important
- Don't expect to store logs forever, most have retention policies of 7 days or a month.
- Do not log passwords
- Do not log personally identifiable information (PII) unless you absolutely have to
  - e.g. use User ID instead of Email Address

email: contact@larene.dev

Popular logging libraries

Logging libraries are dependencies that you add to your software project. There are many and it depends what you're used to, and the appenders or sinks they support.

- Log4j2 (Java)
- java.util.logging (Java)
- Serilog (.NET)
- Microsoft.Extensions.Logging (.NET)
- Bunyan (Node JS)
- any many more...

Popular Log Server and Analysis Tools

You can either have self-managed logging server, or use a SaaS or PaaS product (check the prices first!)

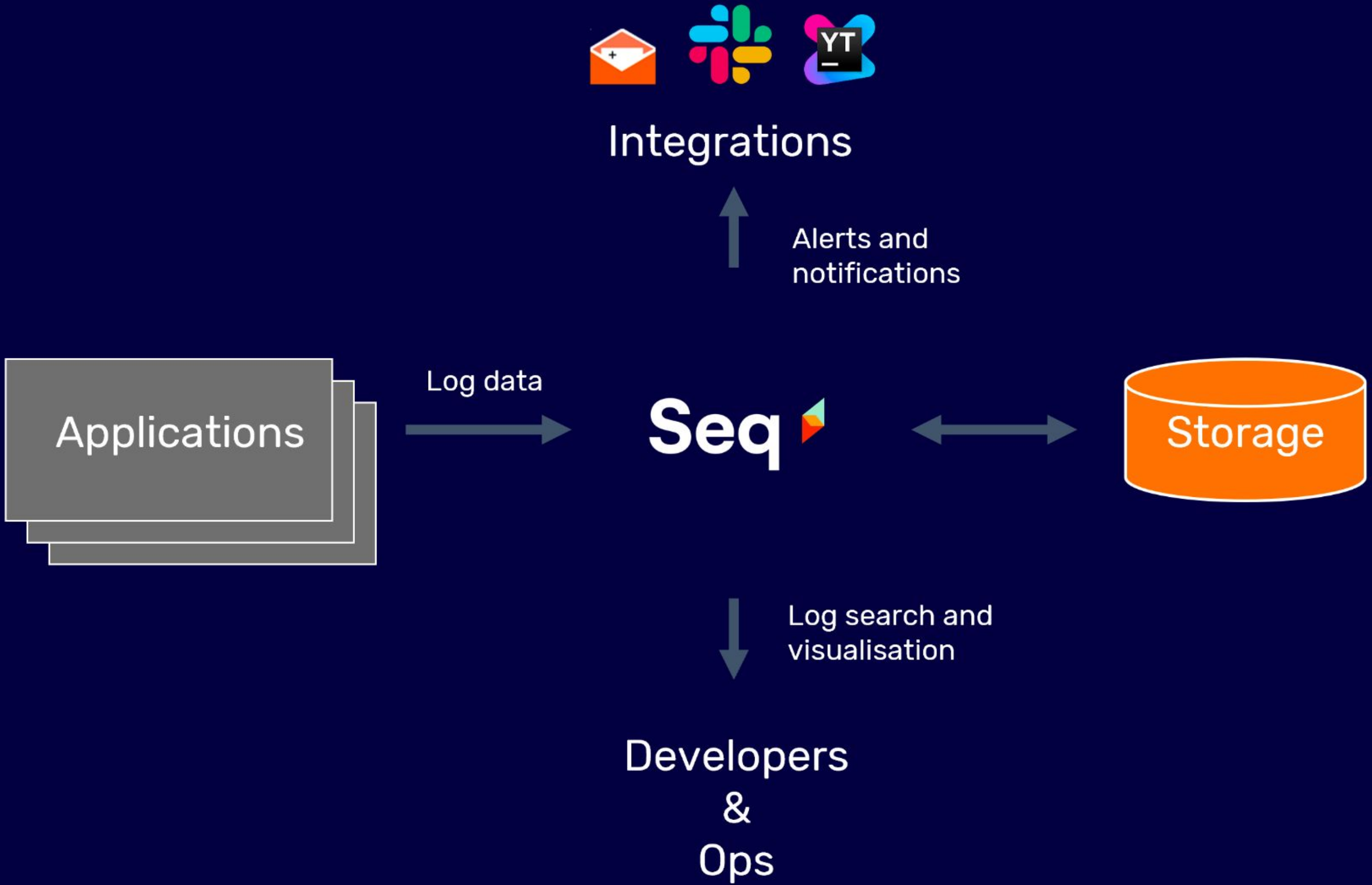
- Splunk
- Datadog
- Logstash+Elasticsearch+Kibana ("ELK" stack)
- Azure Monitoring
- AWS Cloudwatch
- Seq (self-managed only)
- many more...

Practical

About Seq Logging Tool

- Easy to run locally
- Developer friendly
- Created by the same people who built Serilog logging library
- Not just managing log storage, but also alerts, search, analytics, dashboards, etc.
- Read more in the Seq documentation

email: contact@larene.dev


Integrations
Alerts and notifications
Applications
Log data
Seq
Storage
Log search and visualisation
Developers & Ops

Set up a .NET 6 API, add Serilog and log to Seq

1. Go to the repo and clone it
2. Follow the instructions in README.md (we'll do this together)
3. Bonus - look at the Seq docs, and try to log from an existing app of yours to Seq

email: contact@larene.dev