**Application Logging** – the logs that form the business logic i.e. exceptions, warnings, validations and calculation errors, front end errors etc. These don’t include other logs e.g. request responses etc.

**Flavours of Logging** –

* Known / Coded => exceptions, ‘happy paths’, Direct validation errors, fatal errors
* Unknown => edge case algorithm fails, missing database defaults/configurations
* Live => event stream flags, threshold violations, broken service-to-service connections

**Why are we writing logs?**

* To **amend** code logic, add/update/remove elements from database/config files
* To **inform**/advise users what to do
* To **foresee** errors ahead of time – with PSC custom threshold violations, Top n slowest requests etc
* Better **firefighting** – get alerted by email, trace/visualise bugs, detect DDOS attacks and ban IP addresses
* **Analytics** – view where users are spending their time, frequently used services/queries, common errors by user/service
* **Auditing** – what users of the system are doing (e.g. who logged in when, edited, deleted, terminated, reverted, extracted, exported – how often)

**Research Questions:**

1. A comparison between Seq and Azure on application logging. Ease of setup, Docker/Azure containers, VM hosting, scripted IAAS?
2. The case for custom application logging: No-Sql sink with custom queries, threshold flagging, email capabilities, custom web end dedicated to application logging
3. Custom application logging Front end: Dashboard with ‘Application health’, Tabs with microservices, threshold bars (red, green, yellow, counts, percentages), Warning flags. notification, message queues
4. Demo project. Create two or more independent projects (API, Web App, Process) that interact and have their logs send to a central Seq/Logging instance.
5. Demonstrate tracing via correlation id
6. Demonstrate event threshold flagging
7. Demonstrate threshold notification via email
8. Demonstrate ‘Application Health’ via custom dashboard
9. Thread offs
10. Long term log storage vs level of detail when debugging problem
11. Custom vs. existing framework for query, dashboard, threshold, emailing
12. Advanced/Custom logging level setting from an external end point (Seq dashboard like)

**Notes:**

Trace logs

|  |  |  |
| --- | --- | --- |
| **Log field** | **Description** | **Example** |
| Timestamp | Date and time of the event | 04/04/2017 12:32:52 |
| Process | Process that generated the log entry | eLink |
| TID | Thread ID | 123356 |
| Area | Area of the event | eLink.Service |
| Category | The category of the Area of the event | Validation |
| Message | Actual error message | Invalid exit interview start Date |
| Correlation | The correlation ID | 5ea4-aec435-cdes5d |

**Logging Levels:**

1. Descriptions here <https://docs.getseq.net/v3/docs/logging-levels>
2. Seri log’s different levels explained

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Level** | **Use case** | **Example** | **Is it on normally?** | **Expected volume** |
| **VERBOSE** | High level detail aka Trace | Local variable in algorithm | NO | Very high  \*When enabled |
|  | Rarely enabled for production |  |  |  |
|  | When investigating difficult bug |  |  |  |
|  | Use for complicated business logic |  |  |  |
| **DEBUG** | Find out HOW things happened | Request/response to/from external end points | NO | Moderate  ~25% |
|  | Log Connections between services | Api/inter-service call latency timings? |  |  |
|  |  |  |  |  |
| **INFORMATION** | “The happy path” – “successful completions” | Saved/updated/deleted | YES | High  ~75% |
|  | Observable actions | Exported/email sent |  |  |
|  | Use sparingly and concisely to save space | User logged in |  |  |
|  |  | Extract queued |  |  |
|  |  |  |  |  |
| **WARNING** | When behaving outside expected parameters | User input validation errors | YES | Moderate  \*depends on user instance |
|  | When degraded, endangered | Failed login |  |  |
|  | Use restraint not overwhelm log Database | No picklists, defaults found on GET |  |  |
|  |  | Memory cache is near capacity |  |  |
| **ERROR** | Functionality/expectation broken | Exception committing inset/update/delete | YES | LOW |
|  | Exceptions | 500 errors on Front ends |  |  |
|  |  | Api 500s, 400s |  |  |
| **FATAL** | Critical, immediate attention | Database/API connection down | YES | Very LOW |
|  | Application failing on start up | Service unresponsive to health check pings |  |  |
|  | Failed API calls past a threshold | Logged just before exiting |  |  |

1. Some reading on logging levels here:

<https://blog.matthewskelton.net/2012/12/05/tune-logging-levels-in-production-without-recompiling-code>

<http://www.masterzen.fr/2013/01/13/the-10-commandments-of-logging/>

1. Important reading on decoupling framework (SeriLog) logging levels from Run time logging levels by adding abstraction layers. This involved assigning known events unique Ids and creating a (JSON) map of event Ids to Logging levels that is configurable at run time from file/dashboard. Required; JSON file, Event-Enums, ‘DecoupledLogger’ custom C# Utility/Middleware. More here: <https://blog.matthewskelton.net/2012/12/05/tune-logging-levels-in-production-without-recompiling-code>
2. Alternative logging Framework : loggly – cloud based ( free 200Mb/day, 7 day retention) [intro here - <https://www.loggly.com/docs/setup/>, Capability demo 25 min video - <https://www.youtube.com/channel/UCG53aDP09BH-s7fsb1k9Ntw>]

**Seq vs Azure**

**Comparison metrics**

1. **Cost in £**
2. **Data size**
3. **Can it be a script? IaaS**
4. **Ease of Set up – service discovery, configuration etc.**
5. **Scalability – Auto / Manual**
6. **Data Retention policy**
7. **Thresholds, Email alerts, Charts, Logging level changes**
8. **Dashboards per service – Api key, SSL**
9. **Availability Testing**
10. **Disaster recovery & backup**

**Limitations of Seq on Azure:**

1. **Seq requires a virtual machine** for Azure hosting. Due to limitations in the available local storage it's not currently feasible to run Seq as an Azure *Web App* or *Cloud Service*
2. To configure **SSL** you'll need a **Seq license**
3. Seq is inherently a single-node system. For disaster recovery purposes, a combination of **nightly backups and event forwarding** can be used to keep a second instance on stand-by
4. Storage requires **scaling configuration** on Azure

(<https://docs.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-attached-disks>)

1. Seq makes heavy use of RAM caching to efficiently compute queries on log data
2. Email, Azure Event Hub, Threshold Detection have to be added as plugin ‘Apps’ after Seq is installed. (?) complicated set up, especially when generating infrastructure by script (?)

**Benefits of Seq on Azure:**

1. Six step process described here: <https://docs.getseq.net/v3/docs/azure-installation>
2. SeriLog has a nuget package that writes directly to application Insights [Serilog.Sinks.ApplicationInsights]
3. Intuitive Search query and ability to Pin hashed search messages to Dash/Custom query list.

**Limitation of Azure Stream Analytics /Application Insights**

1. Costs £ to run [ 1GB monthly allowance (free) of telemetry data] [https://azure.microsoft.com/en-gb/pricing/details/application-insights/]
2. Need to learn the (analytics) query language
3. Application Insights - **5 million** data points limit for Telemetry, **7 days** retention. Will stop recording any telemetry until calendar moth rolls over.
4. If we have x microservices (api, WebApps, service, etc.), we would have x application Insight instances i.e. dash boards. We’d also need to tab through x of them to check(\*)

**Benefits of Azure stream analytics /Application Insights**

1. Both Instrumentation and application monitoring in one place with as many **customisable dashboards** as needed.
2. Application Insights **Telemetry** Collection [Diagnostic traces, Exceptions, Custom Telemetry, JavaScript Logging]
3. Application Insights **Analytics** [ Dashboards, Interactive Drill down on events, Power BI]
4. Application Insights Proactive **alerts** [Metric rules, Proactive diagnostics, Availability Testing]
5. Can add **web hooks to Alerts** to feed into our own custom web end point.
6. \* If we have x microservices, we can route some alerts via web hooks to an aggregating web endpoint, custom built by ourselves, to view a dash board of Alerts/health checks.

Resources on Azure stream Analytics

1. Explore Microsoft Azure monitoring and diagnostics, <https://azure.microsoft.com/en-us/resources/videos/microsoft-ignite-2016-explore-microsoft-azure-monitoring-and-diagnostics/>
2. <https://app.pluralsight.com/library/courses/azure-solutions-web-apps-70-532/table-of-contents>
3. Good demo of Application Insight capabilities , <https://channel9.msdn.com/Events/Ignite/New-Zealand-2016/M386>

**OTHER ALTERNATIVES**

**New Relic**

This is Instrumentation monitoring tool

**Pros**

1. Best for aggregating multiple applications on multiple servers
2. Has platform packages with azure [https://newrelic.com/partner/azure]
3. Better (vs. Azure) interface on Instrumentation (Disk, Memory, Network etc.) Monitoring
4. Can set thresholds, alarms, see charts and custom views
5. Can also monitor applications

**Cons**

1. Seems out of date
2. Somewhat lengthy to set up (?)
3. Transaction tracing and code level diagnostics are not free.
4. Has to be created as a New cloud service for existing projects