

# How to logging

# Agenda

- Why log?
- What is logging?
- What's your expectation about logging?
- Some good practices of logging
- Tools recommendation
- Tips
- Q&A

# Why log?

What's the importance of logging?

# Why log?

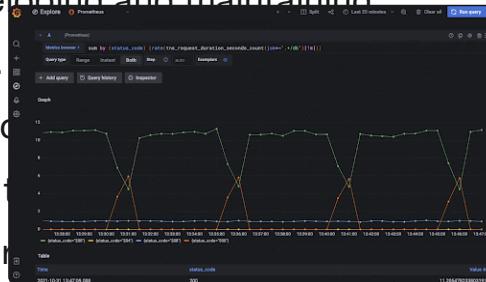
- Troubleshooting
- Alerting
- Business Data Visualization
- ...

# Why log?

## Troubleshooting

- Imagine that the system you are responsible for developing and maintaining is found to be faulty. What should you do? Check the logs for information to verify the errors printed on the server. As programmers, logs are the most familiar tool for solving problems.

- Restore the crime scene**



# Why log?

## Alerting

Logging can be used as an important data source for our business system monitoring; mature product systems have alarm systems. If there is a problem exceeding the defined indicator in the system, it will automatically send the alarm information to the notification platform. We can locate and solve the problem according to the alarm information.



# Why log?

## Business Data Visualization

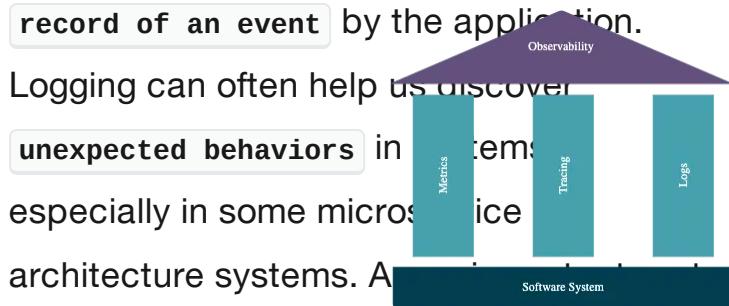
Many companies can use the logs of the production environment stored in their own databases to visualize business data in combination with the cloud monitoring tools such as Grafana and Splunk.



# What is logging?

# What is logging?

- Logging is **structured** or **unstructured** text information generated by the system during operation. Usually, it can be regarded as a **record of an event** by the application.



Logging can often help us discover **unexpected behaviors** in systems, especially in some microservice architecture systems. A

of **observability**, Logging plays an irreplaceable role in our system development and maintenance.

**What's your expectation about  
logging?**

# What's your expectation about logging?

#Troubleshooting #Sensitive Data #Timestamp #Log level #Log tool #Env #Formatting #Store #Alert  
#Visualization #Hostname #Method #Number of line #Message #Filter #PII #PCI #Law #Error #Error  
stack ...

# **Some good practices of logging**

# Log Format



# Some good practices of logging

## Basic template

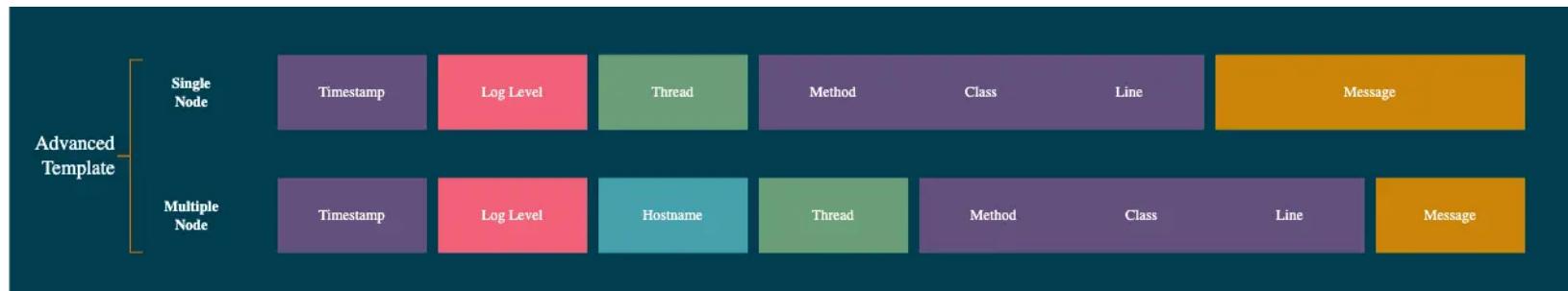
- At minimum, qualified logs should have the following information: `Timestamp`, `Log Level`, and `Message`



# Some good practices of logging

## Advanced template - Graph

- Advanced Template Adds `Thread`, `Hostname`, `Method` name, `Class` name, and the `Number of lines` corresponding to the method



# Some good practices of logging

## Advanced template - Explanation

- **Thread**: Most applications don't have a single user. For a single-instance service, many users accessing the same interface will execute the application in different threads. So using the thread name is best to differentiate the business process of corresponding users.
- **Hostname**: Most current applications are deployed in the cloud with multiple instances, so on the basis of a single node, the logging needs to be distinguished at the instance level on multiple instances, and the hostname is the best way to differentiate.
- **Method Name**: A convenient way to differentiate the source within the same log.
- **Class Name**: A convenient way to quickly locate the business process.
- **Number of line**: A quick way to find the specific location of the log.

# Some good practices of logging

## Formatting - Graph



# Some good practices of logging

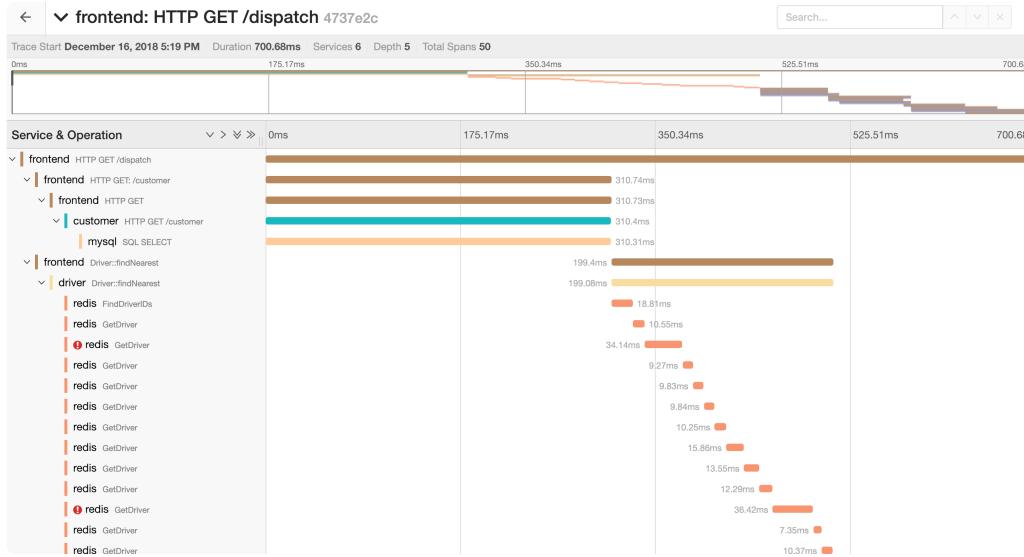
## Formatting - Explanation

- Put **square brackets** around the log level, host name, and thread name;
- Add **parentheses** to the class name and line number where the method name is located, and separate the class name and line number with a colon;
- Add a **horizontal line** between the line number and log information; Specific formatting is also possible for log information:
- For regular requests, response or other business logs, you can separate custom information and parameters with an **underscore**; multiple parameters are separated by **commas** ;
- For error message formatting, you can use **Key:Value** .

# Some good practices of logging

## Chain-Tracking

If the recorded log is just a simple text description line by line, it'll be hard to read. In a complex system or a system with frequent business operations, there will be multiple logs and we'll have to spend time filtering out the relevant logs. The best way to solve this problem is through chain tracking of logs: putting one or more unique IDs in the business system that are added to each log, so that when locating business problems, we can quickly filter out the relevant logs through these unique IDs and Other criteria (e.g. time).



# Some good practices of logging

## Logging on demand - Log level

The output of the log is divided into levels. Different scenarios need to print different levels of logs.

- Debug: Record technical details to help understand the system.
- Info: Record business information.
- Warn: Acceptable error that is manageable and not urgent.
- Error: Unexpected errors or abnormal behavior, usually caused by system bugs or environmental problems.



# Some good practices of logging

## Logging on demand - Log level

Environment

Log Leave

---

Dev

Debug

---

Test

Debug

---

UAT

Info

---

Prod

Info

---

# Some good practices of logging

## Logging on demand - Logging position

The location of the log print also needs to be made clear.

- When other systems call their own systems, they need to print the log once each when they receive a request and when the request is completed;
- The log is printed once before the self-hosted system calls the interface of the third-party system and after receiving the response;
- The log needs to be printed in any abnormal place in the system.

# Some good practices of logging

## Emotional statement

Give the user a clear state of application

### KEY WORDS

`Start to ....`, `Successfully finish ....`, `Failed to ...`

### EXAMPLE

```
2022-11-01T21:30:05.687 [INFO] [10-171-200-68] [71605] get(uid.ts:10)
  -Start to generate uid
```

```
2022-11-01T21:30:05.691 [INFO] [10-171-200-68] [71605] getUid(UidService.ts:13)
  -Start to get UUID from https://www.uuidtools.com/api/generate/v4
```

```
2022-11-01T21:30:10.700 [INFO] [10-171-200-68] [71605] getUid(UidService.ts:19)
  -Successfully get the result from https://www.uuidtools.com/api/generate/v4, the respo
```

```
2022-11-01T21:30:10.700 [INFO] [10-171-200-68] [71605] get(uid.ts:19)
  -Successfully get the result from https://www.uuidtools.com/api/generate/v4, the respo
```

```
2022-11-01T21:35:09.478 [ERROR] [10-171-200-68] [72228] getUid(UidService.ts:19)
  -Failed to request uuid_error: AxiosError: timeout of 1ms exceeded
```

# Tools recommendation

# Tools Recommendation

- Log4j
- Pino

# Tools recommendation

## Log4j

Different programming languages have different logging tools;

- The most famous is [Apache's Log4j](#), which is highly [configurable](#) and can be configured via external files at runtime. It is based on [logging priority](#) and provides [mechanisms](#) to  <http://logging.apache.org> information to many [des](#) database, file, console, [UNIX](#) system log, etc.;
- Log4j has been ported to other programming languages, such as [logging](#) in Python, [log4js](#) in NodeJS, [log4rs](#) in Rust.

# Tools recommendation

## Pino

Very low overhead Node.js logger.

- Integrate with many popular NodeJS web framework
- Many plugins
- Support `Asynchronous Logging`



# Tips

# Tips

## Should not print any PII, PCI

- PII - Personally identifiable information
- PCI - Payment Card Industry



# Tips

## Should obey local laws and regulations

- PILI - Personal Information Law
- GDPR - General Data Protection Regulation



# Q & A

# Refs

- [Some Good Practices of Logging](#)
- [Apache Log4j™ 2](#)
- [logging](#)
- [log4js](#)
- [log4rs](#)
- [Audit Logging Overview](#)

**Thank you!**