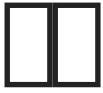


(Obserability)



![qownnotes-media-drHnRi](media/qownnotes-media-drHnRi.png)



(observability)



?

1. 在 系统 中 添加 新的 组件 时 , 需要 考虑 该 组件 是否 具有 可观测性 , 即 是否 能够 收集 到 该 组件 的 运行 数据 , 并 能够 对这些 数据 进行 分析 和 诊断 。
2. 在 系统 中 添加 新的 组件 时 , 需要 考虑 该 组件 是否 具有 可观测性 , 即 是否 能够 收集 到 该 组件 的 运行 数据 , 并 能够 对这些 数据 进行 分析 和 诊断 。
- (ef. 在 系统 中 添加 新的 组件 时 , 需要 考虑 该 组件 是否 具有 可观测性 , 即 是否 能够 收集 到 该 组件 的 运行 数据 , 并 能够 对这些 数据 进行 分析 和 诊断 。
3. 在 系统 中 添加 新的 组件 时 , 需要 考虑 该 组件 是否 具有 可观测性 , 即 是否 能够 收集 到 该 组件 的 运行 数据 , 并 能够 对这些 数据 进行 分析 和 诊断 。
4. 在 系统 中 添加 新的 组件 时 , 需要 考虑 该 组件 是否 具有 可观测性 , 即 是否 能够 收集 到 该 组件 的 运行 数据 , 并 能够 对这些 数据 进行 分析 和 诊断 。



1. 如何實現 監控

“ 如何實現 監控 ”

2. 如何實現 監控 的 方法

- 如何實現 監控 的 方法
- 如何實現 監控 的 方法
- 如何實現 監控 的 方法
- 如何實現 監控 的 方法

3. MELT(Metric / Event / Log / Trace)如何實現 監控 的 方法

4. 如何實現 監控 的 Tool 如何實現 監控 的 方法



1. 如何實現 (cardinality) - 如何實現 監控 的 方法

2. 如何實現 監控 的 方法

- 如何實現 監控 的 方法

3. 如何實現 監控 的 方法

- 如何實現 監控 的 方法
- 如何實現 監控 的 方法

4. 如何實現 監控 的 方法

5. 如何實現 監控 的 方法

6. 如何實現 監控 的 方法

指标 维度 粒度 范围 单位, 精度 备注.



1. 指标 (Metric)

- 指标名称 维度 粒度 范围 单位
- 指标 维度 粒度 范围 单位

2. 日志 (Log)

- 指标名称 维度 粒度 范围 单位 精度 备注
- 指标名称 维度 粒度 范围 单位 精度 备注
- 指标名称 维度 粒度 范围 单位 精度 备注
- 指标名称 维度 粒度 范围 单位 精度 备注

3. 事件 (Event), Newrelic / Redhat 指标 ** - 指标 指标 **

- 指标名称 维度 粒度 范围 单位 精度 备注

4. 追踪 (Trace)

- 指标名称 维度 粒度 范围 单位 精度 备注
- 指标名称, 维度, 粒度 范围 单位 精度 备注



alt text



1. 指标名称 维度 粒度 范围 单位 精度 备注

- 2. 可用性指標 (MTTR)は、システムが故障した後に復旧にかかる平均時間を示す。
- 3. 可用性指標は、システムの信頼性を評価するための重要な指標である。
- 4. 可用性指標は、システムの信頼性を評価するための重要な指標である。
- 5. 可用性指標は、システムの信頼性を評価するための重要な指標である。



alt text

- 1. 可用性指標
- 可用性指標は、システムの信頼性を評価するための重要な指標である。
- 可用性指標は、システムの信頼性を評価するための重要な指標である。
- 可用性指標は、システムの信頼性を評価するための重要な指標である。
- 2. 可用性指標
- 可用性指標は、システムの信頼性を評価するための重要な指標である。
- 可用性指標は、システムの信頼性を評価するための重要な指標である。
- 可用性指標は、システムの信頼性を評価するための重要な指標である。

APM/DPMは、システムの信頼性を評価するための重要な指標である。

- 1. APM/DPMは、システムの信頼性を評価するための重要な指標である。
- 2. APM/DPMは、システムの信頼性を評価するための重要な指標である。
- 3. APM/DPMは、システムの信頼性を評価するための重要な指標である。

Tool






















Opentelemetry

- observability란 OS, 인프라, 애플리케이션
- 지표 : 메트릭 (prometheus)
- 로그 : 이벤트 기반 (로그 / 트레일링 로그)
- 트레이스 : 사용자 요청을 추적 (애플리케이션 , E/S)
- 2022년 GA(일반적으로 사용 가능)인 (클라우드 , 오픈소스)의 API는 클라우드 API와 유사한 구조를 가진다

Appendix#1. ☐☐☐☐ ☐☐



alt textalt text

1. MTBF(Mean Time Between Failures) : 20000 jam , 24 jam MTBF

$$= \frac{20000}{24} = 833.33 \text{ jam}$$
2. MTTR(Mean Time To Repair) : 24 jam MTTR , 10 jam MTTR

$$= \frac{240}{10} = 24 \text{ jam}$$
3. MTTA(Mean Time To Acknowledge) : 10 jam MTTA , 10 jam MTTA

$$= \frac{10}{10} = 1 \text{ jam}$$

那么 40 个 10 秒的窗口，40/10=4，即 4 个窗口

4. MTTD(Mean Time to Detect) : 从异常发生到被检测到的平均时间

5. MTTF(Mean Time To Failure) : 从系统部署到发生故障的平均时间

- MTTA 与 MTTD 的区别：MTTD 是指从异常发生到被检测到的平均时间，MTTA 是指从异常发生到被修复的平均时间

Appendix #2. 数据仓库

数据仓库

1. 数据仓库

- 数据仓库的存储引擎：TSDB(Time Series DataBase) 时间序列数据库，LRU 最近最少使用
- 数据仓库的索引：B+ 树索引，B 树索引，哈希索引，倒排索引，全文索引，位图索引
- 数据仓库的分区：分区表 (Chunk) 将数据分成多个小块，每个小块是一个数据块，每个数据块是一个数据块
- 数据仓库的压缩：数据压缩技术，如 LZ4、Snappy、GZIP 等
- 数据仓库的备份：定期备份数据，防止数据丢失

2. 数据仓库的架构：hierarchical

```
./data
-abc
-def # 数据块
-chunks # 数据块
-1
-tombstones # 数据块
-index # 数据块
-meta.json # 数据块
-checks_head # 数据块
-1
-wal
```

- 001

- checkpoint.001 # wal

- 000 # wal

- WAL(Write Ahead Logging) 在写入数据之前，先将数据写入到 WAL 中，然后再写入到主存储中。flush 操作会将 WAL 中的数据写入到主存储中。WAL 的作用是防止数据丢失，确保数据的持久性。WAL 的写入顺序与主存储的写入顺序一致，保证了数据的一致性。

1. 配置

- storage.tsdb.min-block-duration : 设置块的最小持续时间，2h 表示 2 小时。如果块的数据持续时间小于 2 小时，则不会写入到主存储中。
- storage.tsdb.max-block-duration : 设置块的最大持续时间，10% 表示 10% 的时间。如果块的数据持续时间大于 10% 的时间，则会将块写入到主存储中。

2. 配置

- 配置项

Appendix#3. 附录

1. 配置

- 配置项
- 配置项

2. 配置

opentelemetry 配置项

efk stack 配置项



1. 配置 Tool

Reference

- 1111_____
- Elastic
- New Relic
- 1111111_____
- IBM
- ServiceNow
- Redhat
- 1111_____