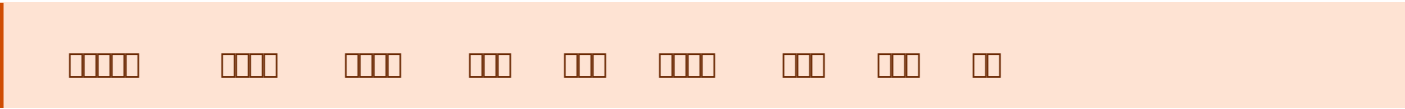


?? ??? (Obserability) ??



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

? ??????(observability)? ????? ?????

1. 我们 可以 通过 系统 日志 数据 , 分析 系统 运行 状态 和 性能 指标 , 来 发现 系统 故障 和 异常 行为 .
2. 我们 可以 通过 系统 日志 数据 来 发现 系统 故障 和 异常 行为 , 但 是 我们 需要 有 足够 的 数据 来 支持 分析 .
(ef. 我们 可以 通过 系统 日志 数据 来 发现 系统 故障 和 异常 行为 , 但 是 我们 需要 有 足够 的 数据 来 支持 分析 .)
3. 我们 可以 通过 系统 日志 数据 来 发现 系统 故障 和 异常 行为 , 但 是 我们 需要 有 足够 的 数据 来 支持 分析 .
4. 我们 可以 通过 系统 日志 数据 来 发现 系统 故障 和 异常 行为 , 但 是 我们 需要 有 足够 的 数据 来 支持 分析 .

?????? ??

1. 我们 可以 通过 系统 日志 数据 来 发现 系统 故障 和 异常 行为 , 但 是 我们 需要 有 足够 的 数据 来 支持 分析 .
“ 我们 可以 通过 系统 日志 数据 来 发现 系统 故障 和 异常 行为 , 但 是 我们 需要 有 足够 的 数据 来 支持 分析 . ”
2. 我们 可以 通过 系统 日志 数据 来 发现 系统 故障 和 异常 行为 , 但 是 我们 需要 有 足够 的 数据 来 支持 分析 .

1. 目的

- どの どの どの どの どの どの
- どの どの どの どの どの どの
- どの どの どの どの どの どの どの どの どの どの

2. 前提

- どの どの どの どの どの どの どの どの
- どの どの どの どの
- どの どの どの どの どの どの どの どの どの どの

APM/DPM? ???

1. どの APM/DPM どの , どの , どの , どの どの どの どの

2. APM/DPM どの どの どの (どの , どの , どの どの) どの , UX どの どの どの どの どの どの

3. APM/DPM どの どの (どの どの) どの どの どの どの , どの どの どの どの どの .

Tool

?????

どの どの どの どの どの どの , Whatap, New Relic, Elastic, IBM(Redhat), ServiceNow, どの

Opentelemetry

- observability どの どの OSS, どの どの
- どの : どの (prometheus)
- どの : どの (どの / どの)
- どの : どの どの (どの , E/S)

- 2022년 GA(기타) (기타, 기타) API(기타) API(기타)

Appendix#1. ????? ?? ??

alt textalt text

1. MTBF(Mean Time Between Failures) : 기 기 , 기 기 . 기
기 기 기
기 기 기 . ef. 24기 기 2기 기 MTBF
기 11기 (24-2)/2(기)
2. MTTR(Mean Time To Repair) : 기 기 기 , 기 기 , 기
기 기 기 , 기 기 기
기 기 기 / 기 10기 기 , 4기 기 240(기)/10(기) = 24기 , MTTR기 24기
3. MTTA(Mean Time To Acknowledge) : 기 기 , 기 기 기
기 기 기
기 기 기 기 기 . 10기 기 기 , 10기 기
기 기 기 기 40기 기 40/10=4, 기 4기 기
4. MTDD(Mean Time to Detect) : 기 기 기 , 기 기 기 기
5. MTTF(Mean Time To Failure) : 기 기 , 기 기 기 기 기
기 기 기 기 .

- MTTA기 MTDD기 기 MTDD기 기 기 기 기 , MTTA기 기
기 기 기 기 기 기 기 기

Appendix #2. ??????? ???

1. 기 기
 - 기 기 TSDB(Time Series DataBase)기 기 , LRU(기)
 - 기 기 기 , 기 기 기 기 기 , 기

- 数据块 (Chunk) 由 一个 或多个 数据 组成，每个 数据 块 包含 一个 或多个 数据 点
- 数据块 按照 时间 顺序 排列
- 每个 数据块 包含 一个 或多个 数据 点

2. 数据块 hierarchical

```
./data
-abc
-def # 数据块
- chunks # 数据块
- 1
- tombstones # 数据块 删除 标记
- index # 数据块 索引
- meta.json # 数据块 元数据
- checks_head # 数据块 检查
- 1
- wal
- 001
- checkpoint.001 # wal 数据块
- 000 # 数据块 删除 标记 wal
```

- WAL(Write Ahead Logging) 是一种 数据 写入 方式，它 确保 数据 在 写入 磁盘 之前 已经 写入 到 内存 中，从而 避免 数据 丢失

1. 配置

- storage.tsdb.min-block-duration : 数据块 的最小 持续时间，2h 表示 2 小时
- storage.tsdb.max-block-duration : 数据块 的最大 持续时间，1d 表示 1 天
- storage.tsdb.retention.time : 数据块 的 保留 时间，10% 表示 10% 的数据块 会被 删除

2. 数据块 管理

- 数据块 的 管理 包括 数据块 的 删除 和 重建

Appendix#3. ?? ??

1. 在本地部署 3 个节点
- 节点 1 : 部署 master 节点 , 并 部署 etcd
 - 节点 2 : 部署 master 节点 并 部署 etcd
2. 在本地部署 1 个节点 : 部署 master 节点 并 部署 etcd
- 在本地部署 opentelemetry 节点 并 部署 etcd
- 在本地部署 efk stack 节点 并 部署 etcd

?????? ???? ?? ??

1. 部署 Tool 节点 并
2. 部署 节点 并

Reference

- [部署 etcd](#)
- [Elastic](#)
- [New Relic](#)
- [部署 etcd](#)
- [IBM](#)
- [ServiceNow](#)
- [Redhat](#)
- [部署 etcd](#)