Tipos de métricas

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Métricas de trabalho

Vamos entender um pouco sobre cada uma delas:

Rendimento

A quantidade de trabalho que o sistema realiza por unidade de tempo.

A apresentação geralmente é registrada como um número absoluto.

Success/Error

As métricas que representam a porcentagem de trabalho executado com sucesso e aqueles que falharam.

Performance

É a quantificação da eficiência com que um componente está fazendo seu trabalho. Geralmente é comparado com medidas pré-estabelecidas.

Métricas de recursos

Vamos entender um pouco sobre cada uma delas:

Utilização

É a porcentagem de tempo que um recurso está ocupado ou em uso.

Saturação

É a medida da quantidade de trabalho que o recurso ainda não pode manipular, geralmente em espera ou na fila.

Disponibilidade

Representa a porcentagem de tempo que o recurso tem para oferecer às solicitações. Ou seja, o que ele tem disponível para aquilo que é solicitado

Eventos

Vamos entender um pouco sobre cada um deles:

Mudança no código

Envolve todos os tipos de mudanças no código: modificações, compilações e / ou travamentos ou falhas.

Alertas

Eventos gerados com base em algum parâmetro pré-estabelecido em algum recurso.

Autoescalado

A adição e / ou subtração automática de recursos com base na carga ou outra configuração pré-estabelecida.

Métricas de Evento

Tipo	Descrição
FATAL	Descrevem um aplicativo irrecuperável ou falha do sistema, ou uma falha catastrófica que requer atenção imediata.
ERROR	Destacam quando o fluxo atual de execução é interrompido devido a uma falha. Isso deve indicar uma falha na atividade atual, não uma falha em todo o aplicativo.
WARN	Destacam um evento anormal ou inesperado no fluxo do aplicativo, mas não causam a interrupção da execução do aplicativo.
INFO	Rastreiam o fluxo geral do aplicativo. Esses logs devem ter valor de longo prazo.
DEBUG	São usados para investigação interativa durante o desenvolvimento. Esses logs devem conter principalmente informações úteis para depuração e não têm valor de longo prazo.
ALL	Contêm as mensagens mais detalhadas. Essas mensagens podem conter dados confidenciais do aplicativo. Essas mensagens são desabilitadas por padrão e nunca devem ser habilitadas em um ambiente de produção.

Material extra / Bibliografia

Material extra

Amazon Documentation. “Visualizar as métricas disponíveis”, https://docs.aws.amazon.com/pt_br/AmazonCloudWatch/latest/monitoring/viewing_metrics_with_cloudwatch.html

Amazon Documentation. “Criar grafos de métricas”, https://docs.aws.amazon.com/pt_br/AmazonCloudWatch/latest/monitoring/graph_metrics.html

Digital Ocean Documentation(inglês). “An Introduction to Metrics, Monitoring, and Alerting”, https://www.digitalocean.com/community/tutorials/an-introduction-to-metrics-monitoring-and-alerting

Software para monitoramentos

ZABBIX

AWS Cloud Watch

Grafana

DATADOG

kibana

splunk>

new relic®

O que iremos ver para Logs?

AWS Cloud Watch

Exemplos de Logs

```
2020-10-16 18:16:45,291 event=DEBUG class="main" line=15 message="Isso é um teste!" entity=""
httpCode=0 cause=""
2017-07-25 17:02:12,300 event=ERROR class="produto" message="conexão recusada"
entity="listener" userId="1234" ip="10.0.0.1"
```

Como faço um log na aplicação?

```
<!-- spring logback.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <springProperty scope="" name="LEVEL" source="logging.level.root"
defaultValue="info"/>
   <property name="dateTimePattern" value="yyyy-MM-dd'T'HH:mm:ss.SSSZ" />
   <property name = "logPattern" value="%d{${dateTimePattern}, GMT-3} event=%5p
class=%c{1} method=%M line=%L httpStatus=%s %m%n -traceId=%X{traceId}%n" />
   <appender name="Console" class="ch.qos.logback.core.ConsoleAppender">
       <layout class="ch.qos.logback.classic.PatternLayout">
           <pattern>${logPattern}</pattern>
       </layout>
   </appender>
   <root level="${LEVEL}">
       <appender-ref ref="Console"/>
   </root>
</configuration>
```

Como faço um log na aplicação?

```
import org.apache.log4j.Logger;
import org.apache.log4j.xml.DOMConfigurator;

public class main {
  static Logger logger = Logger.getLogger(main.class);
  static final String LOG_INFO_WITH_FULL_ENTITY = "message=\"%s\" entity=\"%s\"
httpCode=%d cause=\"%s\"";

  public static void main(String[] args) {
      //DOMConfigurator is used to configure logger from xml configuration file
      DOMConfigurator.configure("log4j.xml");
      //Log in console in and log file
      logger.debug(String.format(LOG_INFO_WITH_FULL_ENTITY, "Isso é um teste!",
"", 0, ""));
  }
}
```

DigitalHouse>