

Centro Universitário  
Christus- UNICHRISTUS

Especialização em  
Ciência de Dados e  
Inteligência de Negócios  
(Big Data e BI)

Prof. Dr. Manoel Ribeiro

# Processamento de Dados em Tempo Real (Streaming)

## Aula 4

# Conteúdo da disciplina

- Dia 1 (sexta 18:00 às 22:00h)
  - Apresentação
  - Aula motivacional - Qual a importância do processamento de dados em tempo real?
- Dia 2 (sábado 8:00 às 18:00)
  - Fundamentos de sistema de processamento de tempo real
  - Fundamentos da arquitetura Apache Kafka, Apache ZooKeeper
  - Prática com Apache Kafka (tarde)

# Conteúdo da disciplina

- Dia 3 (sexta 18:00 às 22:00h)
  - Fundamentos Apache Flume
  - Fundamentos Spark Streaming
  - Configuração Hadoop e Spark
- Dia 4 (sábado 8:00 às 18:00)
  - Fundamentos Introdução ao Apache Storm
  - Implementação de projeto prático/aplicado envolvendo Real Time Analytics
  - Avaliação

# Repositório

<https://github.com/antoniomralmeida/streaming>

# Apache Storm

# Apache Storm

## ● Introdução

- Engine de processamento de streaming de dados distribuída, tolerante a falhas e altamente escalável.
  - 1.000.000 de mensagens por segundo em um cluster de 10 nós.
- Modelo de programação simples: **Topology - Spouts – Bolts**
- Primeira release: 19/set/2011
- Principais linguagens: Closure e Java, mas é independente
- Quem usa?



# Apache Storm

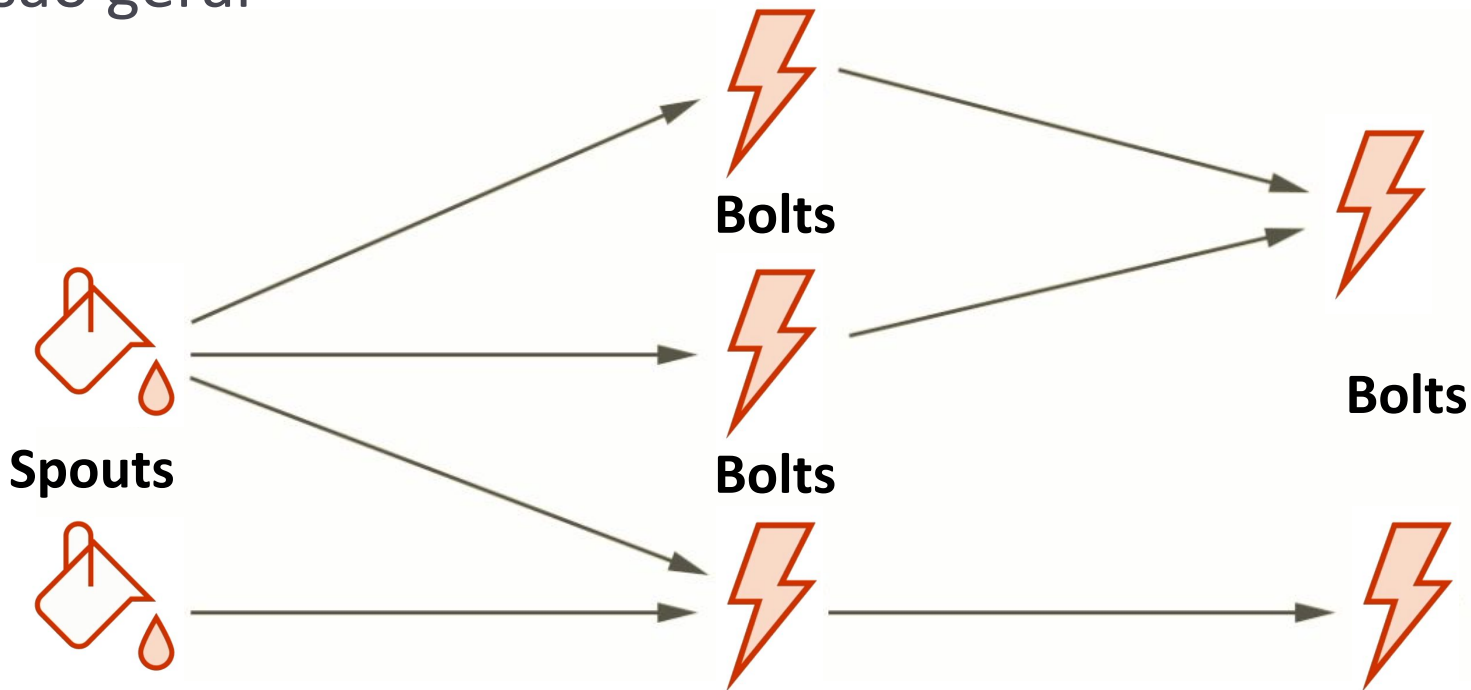
- Visão geral

- Comparar preços com uma constante
- Monitorar mensagens de erro (log)
- Encontrar trending topics (tweet)



# Apache Storm

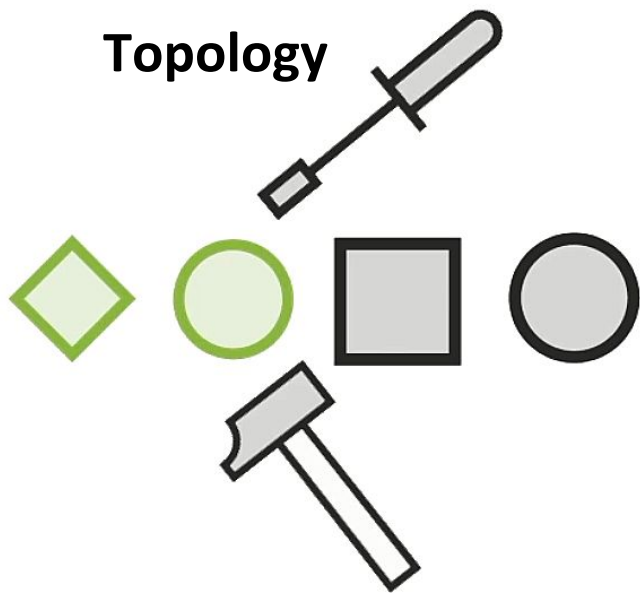
- Visão geral





# Apache Storm

- Componentes



## Spout



- Recebe o dado a partir de uma fonte;
- Emite o dado para o restante da topologia.

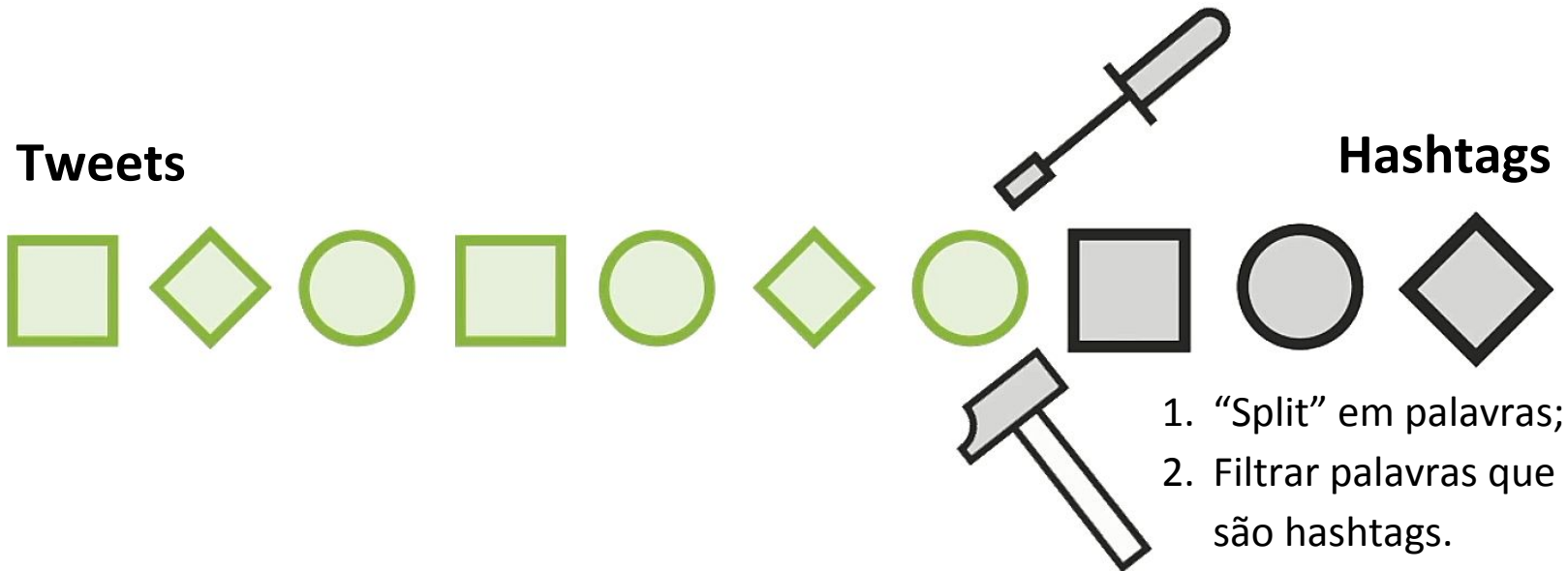
## Bolt



- Processa o dado.

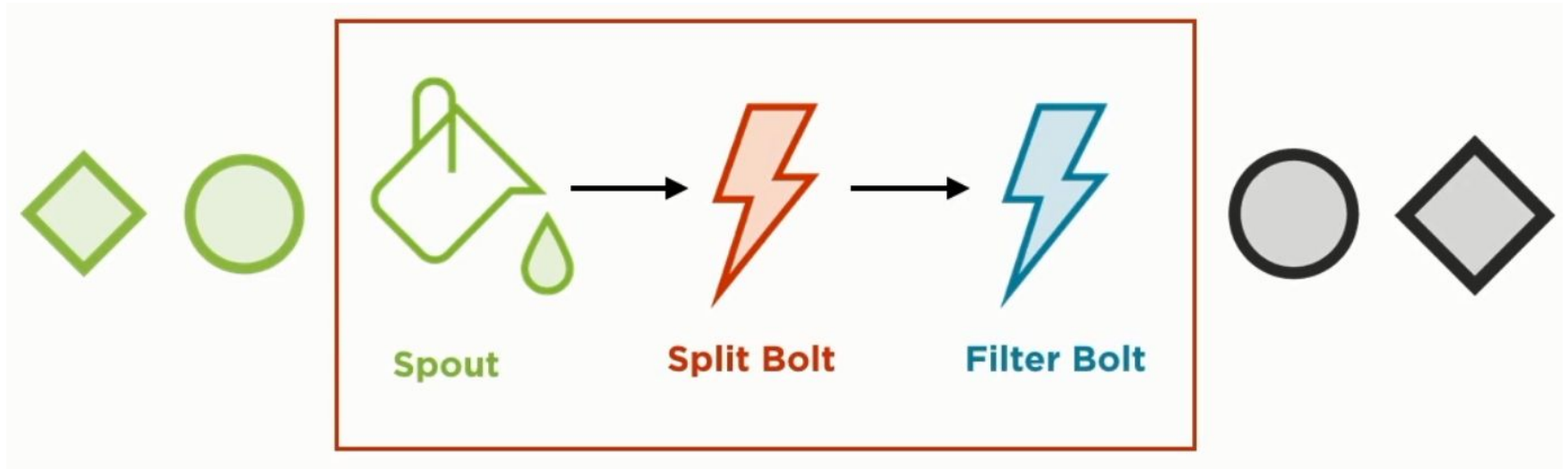
# Apache Storm

- Exemplo: Topologia para extrair hashtags de tweets



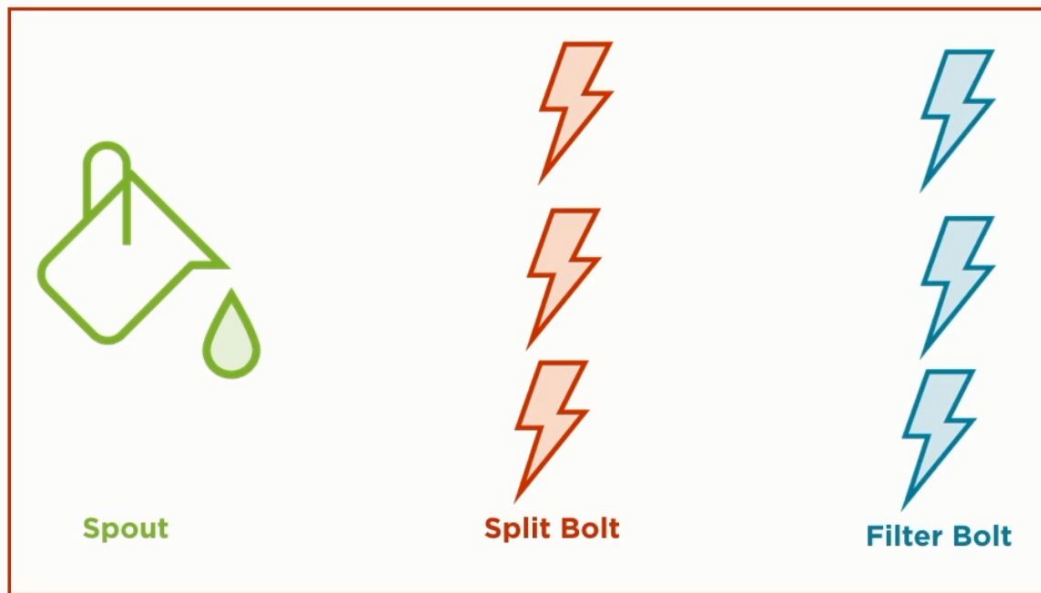
# Apache Storm

- Exemplo: Topologia para extrair hashtags de tweets



# Apache Storm

- Exemplo: Topologia para extrair hashtags de tweets



# Apache Storm

- Rodando o Storm.



**Local**

**Os processos rodam localmente.  
Topologia roda a partir da IDE.**

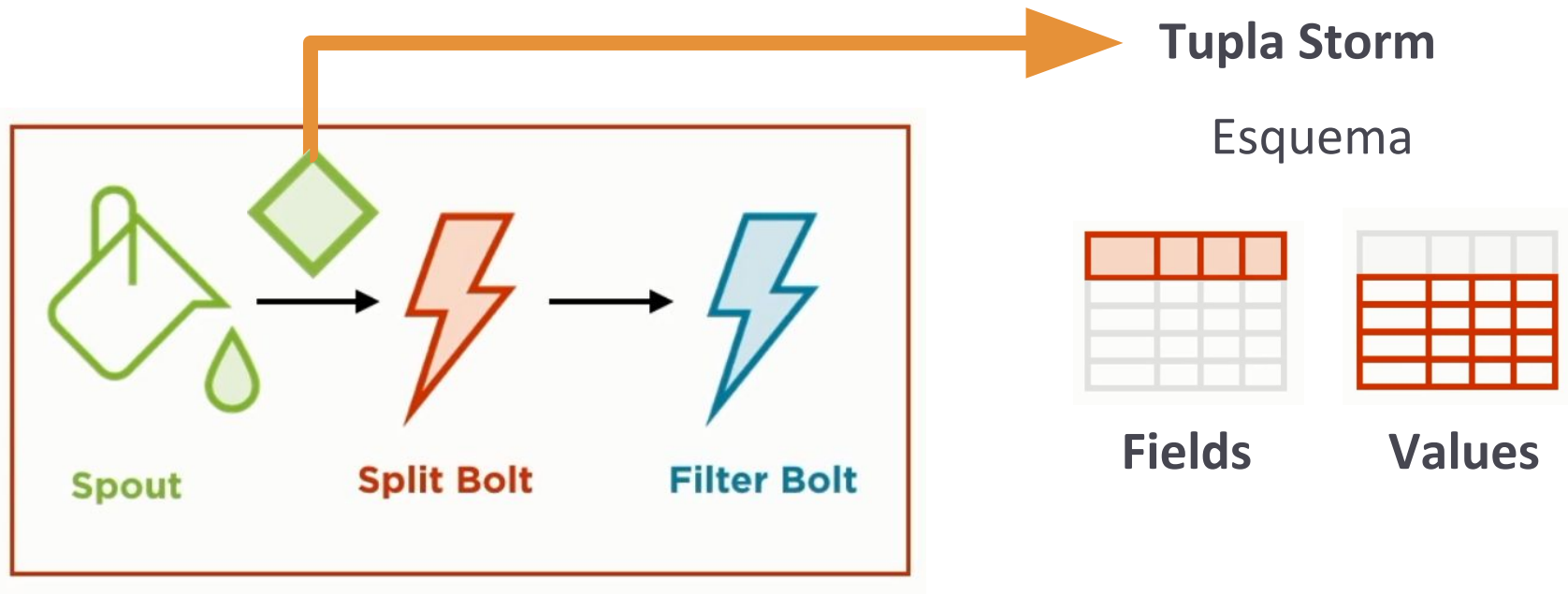


**Remoto**

**Os processos rodam em várias máquinas.  
Submeter um jar ao cluster.**

# Apache Storm

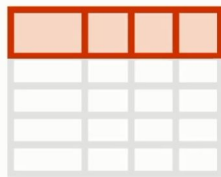
- Cada componente emite tuplas com **esquema fixo**.



# Apache Storm

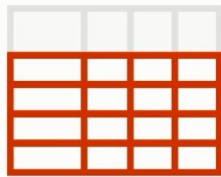
- Cada componente emite tuplas com **esquema fixo**.

Esquema




Fields

1




Values

2

## 1 Declarar campos

```
public void declareOutputFields(OutputFieldsDeclarer declarer){  
    declarer.declare(new Fields("symbol", "date", "price"));
```

## 2 Emitir valores

```
        "symbol", "date", "price"  
collector.emit(new Values("MSFT", "Mar 12 9:00 AM", 62.5))
```

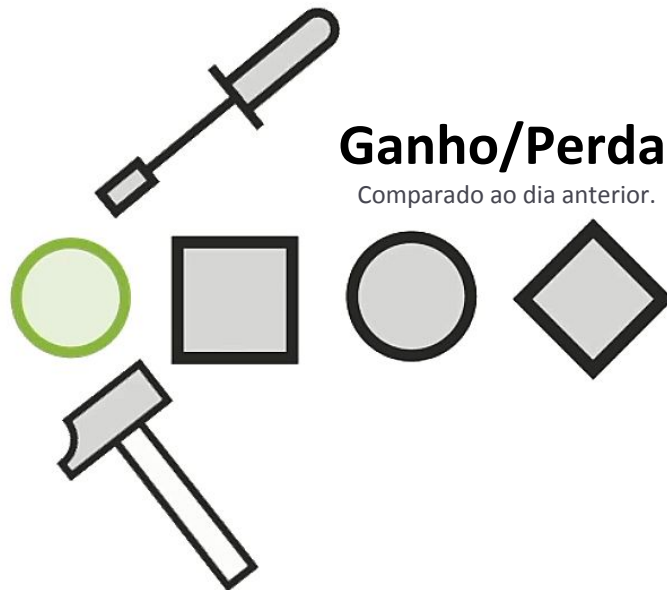
Storm - Yahoo Finance



# Exemplo (Yahoo Finance)

- Análise de preço de ações utilizando o Yahoo Finance.

**Preço de Ações**



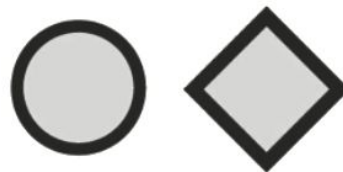
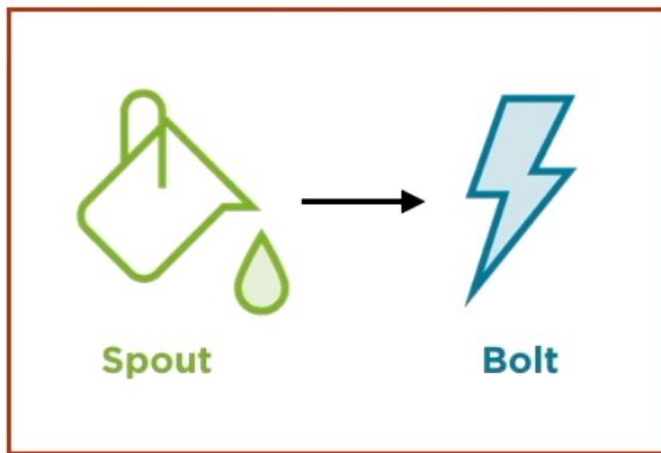
# Exemplo (Yahoo Finance)

- Análise de preço de ações utilizando o Yahoo Finance.

## Topology



**Yahoo Finance**



**Ganho/Perda**

Escrever em um arquivo.

# Exemplo (Yahoo Finance)

- Análise de preço de ações utilizando o Yahoo Finance.

## Set up do Projeto

Utilizar maven para add dependência do Storm.

## Set up do Bolt

Estender a classe abstrata BaseBasicBolt.

## Set up do Spout

Estender a classe abstrata BaseRichSpout.

## Rodar a Topologia

Construir uma topologia e rodar localmente.

# Etapa 1. Set up do Projeto

## 1. Download Eclipse

<http://www.eclipse.org/downloads/>

## 1. Extrair e executar Instalação Eclipse IDE for Java Developers



# Etapa 1. Set up do Projeto

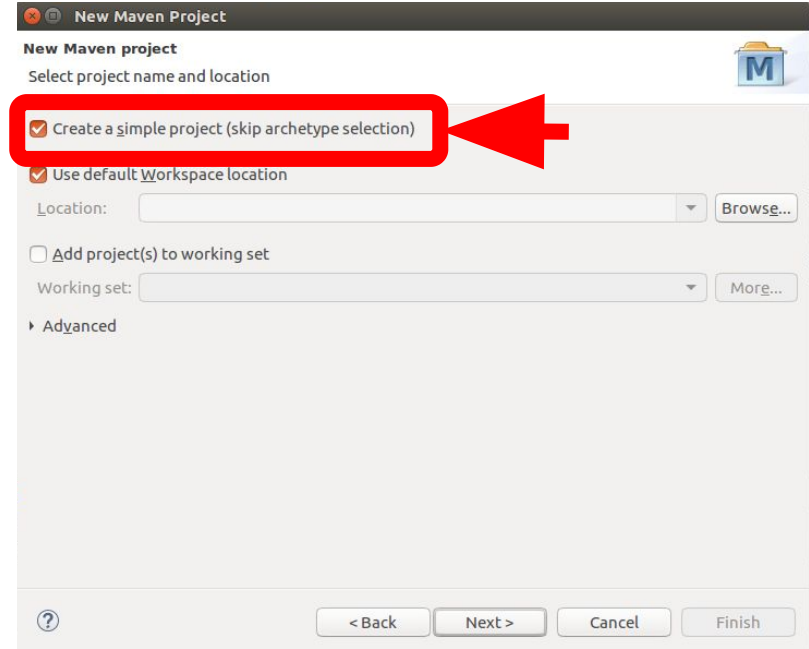
3. Iniciar o eclipse

4. File -> New -> Other ->  
Maven Project

3. Group id: streaming

4. Artifact id: streaming

5. Finish!



# Etapa 1. Set up do Projeto

Caso apresente *warning* de build path:

7.1. Botão direito no projeto -> Propriedades

7.2. Java Build Path

7.3. Aba Libraries -> Remove JRE System Library

7.4. Add Library -> JRE System Library -> next

7.5. Alternate JRE java-8-oracle

7.6. Finish -> Apply and close

# Etapa 1. Set up do Projeto

## 8. Configurar dependências no pom.xml

```
<dependencies>
  <dependency>
    <groupId>org.apache.storm</groupId>
    <artifactId>storm-core</artifactId>
    <version>1.0.2</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>com.yahoo.finance-api</groupId>
    <artifactId>YahooFinanceAPI</artifactId>
    <version>3.12.3</version>
  </dependency>
</dependencies>
```

# Etapa 1. Set up do Projeto

## 9. Botão direito no projeto:

### a. Run as -> Maven clean

[INFO] -----

[INFO] BUILD SUCCESS

[INFO] -----

### b. Run as -> Maven install

[INFO] -----

[INFO] BUILD SUCCESS

[INFO] -----



# Etapa 2. Set up do Spout

Nova classe java **YahooFinanceSpout** em main/src

```
public void open(Map conf, TopologyContext context, SpoutOutputCollector collector) {  
    this.collector = collector;  
}
```

```
public void declareOutputFields(OutputFieldsDeclarer declarer) {  
    declarer.declare(new Fields("company", "timestamp", "price",  
"prev_close"));  
}
```

## Etapa 2. Set up do Spout

```
public void nextTuple() {  
    try {  
        StockQuote quote = YahooFinance.get("MSFT").getQuote(); //  
Financas da Microsoft  
        BigDecimal price = quote.getPrice();  
        BigDecimal prevClose = quote.getPreviousClose();  
        Timestamp timestamp = new  
Timestamp(System.currentTimeMillis());  
        collector.emit(new Values("MSFT", sdf.format(timestamp),  
            price.doubleValue(), prevClose.doubleValue()));  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

# Etapa 3. Set up do Bolt

## Nova classe java **YahooFinanceBolt** em main/src

```
public void prepare(Map stormConf, TopologyContext context) {  
    String filename = stormConf.get("fileToWrite").toString();  
  
    try {  
        this.writer = new PrintWriter(filename, "UTF-8");  
    } catch (Exception e) {  
        throw new RuntimeException("Erro ao abrir o arquivo  
["+filename+"]");  
    }  
}
```

## Etapa 3. Set up do Bolt

```
public void declareOutputFields(OutputFieldsDeclarer declarer) {  
    declarer.declare(new Fields("company", "timestamp", "price",  
"gain"));  
}  
  
public void cleanup() {  
    writer.close();  
}
```

## Etapa 3. Set up do Bolt

```
public void execute(Tuple input, BasicOutputCollector collector) {  
    String symbol = input.getValue(0).toString();  
    String timestamp = input.getString(1);  
    Double price = (Double) input.getValueByField("price");  
    Double preClose = (Double) input.getValueByField("prev_close");  
    Boolean gain = true;  
    if(price <= preClose) {  
        gain = false;  
    }  
    collector.emit(new Values(symbol, timestamp, price, gain));  
    writer.println(symbol+", "+timestamp+", "+price+", "+gain);  
}
```

# Etapa 4. Rodar a topologia

Nova classe java **TopologyMain** em main/src

```
public class TopologyMain throws InterruptedException{

    public static void main(String[] args) {
        // Construir topology

        // Configurar

        // Submeter no cluster
    }
}
```

# Exercício

- Alterar a saída para emitir uma tupla no seguinte formato:

```
(symbol, timestamp, prev_close, price, gain)
```

- Alterar para analisar finanças do Google

```
YahooFinance.get("GOOGL")
```

- Alterar para analisar finanças da Apple

```
YahooFinance.get("AAPL")
```

- Alterar para parar a aplicação depois de 30 seg. de execução

```
cluster.shutdown()
```

# Exercício (valendo ponto)

- Alterar para que a saída emita tuplas do tipo:

`(symbol, timestamp, prev_close, price, gain)`

não mais para a mesma empresa, mas de forma simultânea para Microsoft, Google e Apple.



Fim