RETCHAT

Flume, Kafka and Scalability Spark Streaming

Aiden HuangTable of Contents

[1 Introduction/Purpose 3](#_Toc451162519)

[1.1 Purpose 3](#_Toc451162520)

[1.2 Scope 3](#_Toc451162521)

[1.3 Overview 3](#_Toc451162522)

[2 Version 3](#_Toc451162523)

[3 Architecture and Setup 3](#_Toc451162524)

[3.1 Kafka 3](#_Toc451162525)

[3.1.1 單機安裝測試 3](#_Toc451162526)

[3.2 Spark 9](#_Toc451162527)

[3.3 Mesos 10](#_Toc451162528)

[4 Setup 10](#_Toc451162529)

[4.1 Kafka 10](#_Toc451162530)

[4.2 Spark 10](#_Toc451162531)

[4.3 Mesos 10](#_Toc451162532)

[5 Operation 10](#_Toc451162533)

[6 Cloud 10](#_Toc451162534)

[6.1 佈署流程 10](#_Toc451162535)

[6.2 所需項目與費用（2016/4/18 更新） 10](#_Toc451162536)

[6.2.1 隨選型（前期測試用，預估三天） 11](#_Toc451162537)

[6.2.2 預付型（長期使用） 11](#_Toc451162538)

[7 Tasks 12](#_Toc451162539)

Table of Figures

[Figure 1: 1](#_Toc94586926)

[Figure 2: 1](#_Toc94586927)

[Figure 3: 1](#_Toc94586928)

[Figure 4: 1](#_Toc94586929)

# Introduction/Purpose

## Purpose

This document provides the detailed design of Flume, Kafka and Scalability Spark Streaming. The document should provide sufficient information to carry out the development of the component.

## Scope

This design document will describe the procedure of setting up Flume, Kafka and Scalability Spark Streaming.

## Overview

Version

| Software | | |
| --- | --- | --- |
| Name | Version | Date |
| Flume |  |  |
| Kafka | 0.9.1.0 | 2016/04/28 |
| Spark | 1.6.1 |  |
| JDK | 1.7 u55 |  |

# Architecture and Setup

## Kafka

### 單機安裝測試

#### 作業環境準備

1. 安裝作業系統

* Centos 6.7 x64

1. 安裝 Java

* wget --no-cookies --no-check-certificate --header "Cookie: oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/7u55-b13/jdk-7u55-linux-x64.rpm" -O jdk-7-linux-x64.rpm
* rpm -ivh jdk-7-linux-x64.rpm

1. 安裝 Java\_Home

* vim /etc/profile
* export JAVA\_HOME=/usr/java/jdk1.7.0\_55
* export PATH=$PATH:/usr/java/jdk1.7.0\_55/bin

#### 系統架設（3 個 broker）

1. 下載 Kafka

* wget http://apache.stu.edu.tw/kafka/0.9.0.0/kafka\_2.11-0.9.0.0.tgz
* tar -xzf kafka\_2.11-0.9.0.0.tgz
* mv kafka\_2.11-0.9.0.0 /usr/local/ kafka\_2.11-0.9.0.0
* cd /usr/local/kafka\_2.11-0.9.0.0

1. 修改設定檔

* vim /usr/local/kafka\_2.11-0.9.0.0/config/server.properties
* advertised.host.name=10.144.30.31 # 加入此行

1. 新增其他兩個 broker

* cp config/server.properties config/server-1.properties
* cp config/server.properties config/server-2.properties
* 分別將內容修改如下：
  + config/server-1.properties:
  + broker.id=1
  + port=9093
  + log.dir=/tmp/kafka-logs-1
  + config/server-2.properties:
  + broker.id=2
  + port=9094
  + log.dir=/tmp/kafka-logs-2

1. 啟動 zookeeper

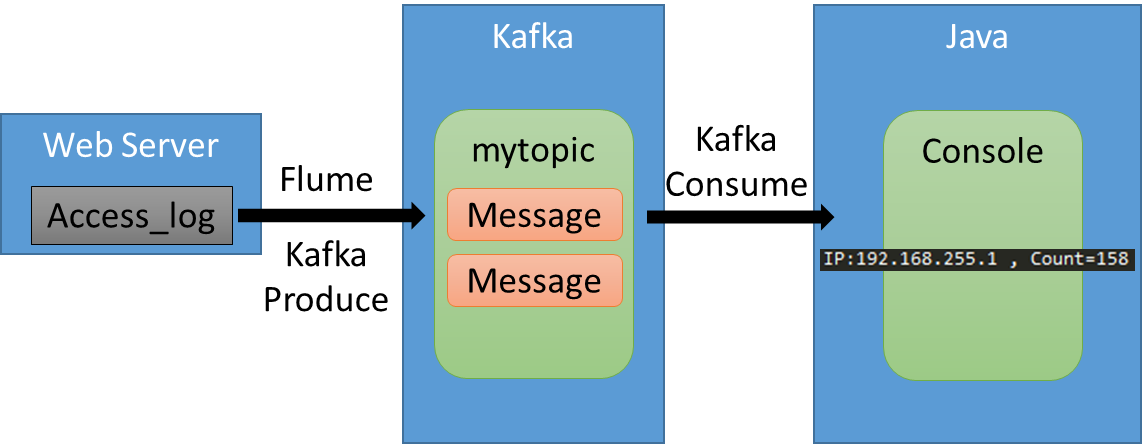
* bin/zookeeper-server-start.sh config/zookeeper.properties &

1. 啟動 kafka

* bin/kafka-server-start.sh config/server.properties &
* bin/kafka-server-start.sh config/server-1.properties &
* bin/kafka-server-start.sh config/server-2.properties &

#### Flume with Kafka

以下為用 Flume 將 Web Server 的 Access\_log 傳送到 Kafka 上，再以 Java取得 Message 後進行後續處理



1. Flume設定

* vim flume-conf.properties

agent1.sources=source1

agent1.sinks=sink1

agent1.channels=channel1

#Spooling Directory

#set source1

agent1.sources.source1.type=exec

agent1.sources.source1.command=tail -f /var/log/httpd/access\_log

agent1.sources.source1.batchSize = 1

agent1.sources.source1.channels=channel1

agent1.sources.source1.fileHeader = false

agent1.sources.source1.interceptors = i1

agent1.sources.source1.interceptors = itime ihost itype

agent1.sources.source1.interceptors.itime.type = timestamp

#agent1.sources.source1.channels = memoryChannel

agent1.sources.source1.interceptors = itime ihost itype

# http://flume.apache.org/FlumeUserGuide.html#timestamp-interceptor

agent1.sources.source1.interceptors.itime.type = timestamp

# http://flume.apache.org/FlumeUserGuide.html#host-interceptor

agent1.sources.source1.interceptors.ihost.type = host

agent1.sources.source1.interceptors.ihost.useIP = false

agent1.sources.source1.interceptors.ihost.hostHeader = host

# http://flume.apache.org/FlumeUserGuide.html#static-interceptor

agent1.sources.source1.interceptors.itype.type = static

agent1.sources.source1.interceptors.itype.key = log\_type

agent1.sources.source1.interceptors.itype.value = apache\_access\_combinedFe

#set sink1

agent1.sinks.sink1.type=hdfs

agent1.sinks.sink1.hdfs.path=/home/hdfs/flume/logdfs

agent1.sinks.sink1.hdfs.fileType=DataStream

agent1.sinks.sink1.hdfs.writeFormat=TEXT

agent1.sinks.sink1.hdfs.rollInterval=1

agent1.sinks.sink1.channel=channel1

agent1.sinks.sink1.hdfs.filePrefix=%Y-%m-%d

agent1.sinks.sink1.type = org.apache.flume.sink.kafka.KafkaSink

agent1.sinks.sink1.topic = mytopic

agent1.sinks.sink1.brokerList = 10.144.30.31:9092

agent1.sinks.sink1.requiredAcks = 1

agent1.sinks.sink1.batchSize = 20

agent1.sinks.sink1.channel = channel1

#set channel1

agent1.channels.channel1.type=file

agent1.channels.channel1.checkpointDir=/root/flume/point

agent1.channels.channel1.dataDirs=/root/flume/logdfstmp

* 啟動 Flume
  + bin/flume-ng agent -c conf -f conf/flume-conf.properties -n agent1

1. Kafka 設定

（參考上一章節）

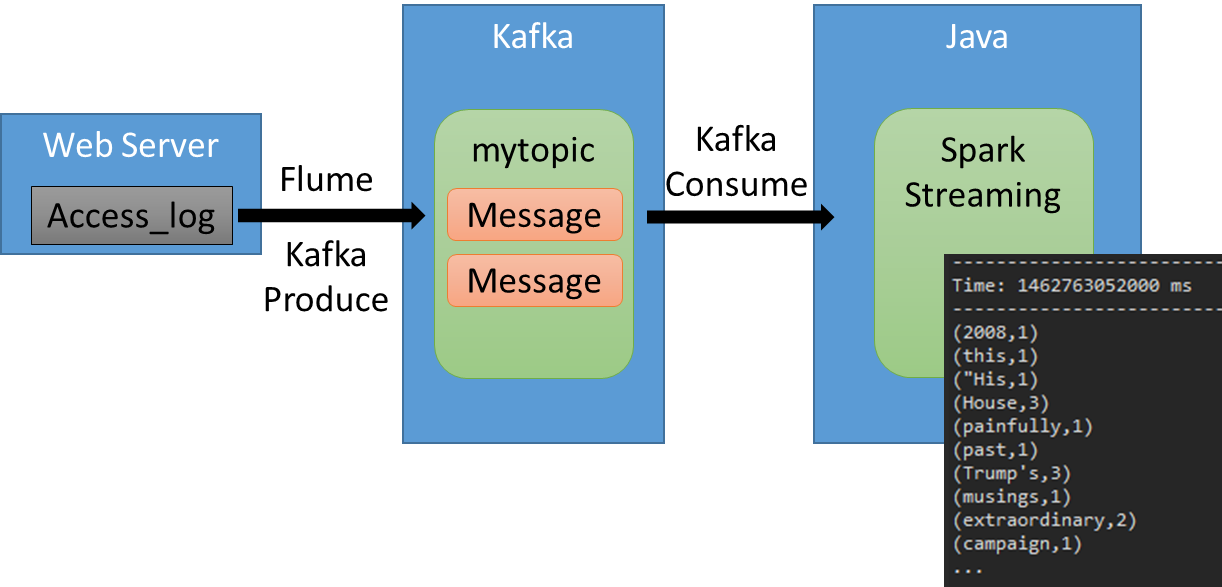
1. Java Consumer

import com.fasterxml.jackson.databind.deser.DataFormatReaders;  
import org.apache.kafka.clients.consumer.ConsumerRecord;  
import org.apache.kafka.clients.consumer.ConsumerRecords;  
import org.apache.kafka.clients.consumer.KafkaConsumer;  
  
import java.util.Arrays;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Properties;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
/\*\*  
 \* Created by aiden on 2016/4/28.  
 \*/  
public class Consumer\_Test {  
  
 public static void main(String[] args){  
  
 Properties props = new Properties();  
 props.put("bootstrap.servers", "10.144.30.31:9092");  
 props.put("group.id", "test");  
 props.put("enable.auto.commit", "true");  
 props.put("auto.commit.interval.ms", "1000");  
 props.put("session.timeout.ms", "30000");  
 props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");  
 props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");  
 KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);  
 consumer.subscribe(Arrays.asList("mytopic"));  
   
 HashMap<String, Integer> list = new HashMap<String, Integer>();  
  
 while (true) {  
 ConsumerRecords<String, String> records = consumer.poll(100);  
 for (ConsumerRecord<String, String> record : records) {  
  
 String value = record.value();  
 //System.out.printf("offset = %d, key = %s, value = %s \n", record.offset(), record.key(), value);  
  
 String ipkey = value.split("-")[0];  
  
 Integer i = -1;  
 if(list.containsKey(ipkey)){  
 i = list.get(ipkey);  
 }  
 list.put(ipkey, ++i);  
  
 for (String s:list.keySet()) {  
 System.***out***.println(String.format("IP:%s, Count=%d", s, list.get(s)));  
 }  
 }  
 }  
 }  
}

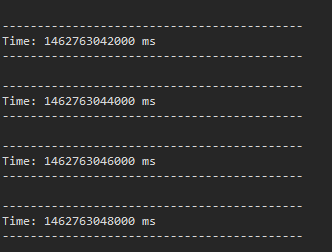
## Spark

### Flume with Kafka with Spark Streaming

以下為用 Flume 將 Web Server 的 Access\_log 傳送到 Kafka 上，再以 Java取得 Message 後進行後續處理



1. 請先 check out 「Kafka\_test\_eclipse」專案，在專案中開啟JavaKafkaWordCount.java
2. 根據相關環境變數修改對應之變數內容（IP, Topic, group, zookeeper）
3. 執行專案，執行成功 console 畫面會持續出現，代表執行成功



1. 此時對 topic produce 資料，畫面會立即顯示處理後的資料結果（範例是對每個 message 進行 word count）

## Mesos

## Ambari

### 簡介

Apache Ambari 提供簡單易用的 Web UI，以供用來佈建、管理及監視 Hadoop 叢集，讓 Hadoop 管理起來更為簡單。開發人員可以使用 Ambari REST API 將這些功能整合到應用程式。重要功能如下：

* Provision a Hadoop Cluster
  + Ambari provides a step-by-step wizard for installing Hadoop services across any number of hosts.
  + Ambari handles configuration of Hadoop services for the cluster.
* Manage a Hadoop Cluster
  + Ambari provides central management for starting, stopping, and reconfiguring Hadoop services across the entire cluster.
* Monitor a Hadoop Cluster
  + Ambari provides a dashboard for monitoring health and status of the Hadoop cluster.
  + Ambari leverages [Ambari Metrics System](https://issues.apache.org/jira/browse/AMBARI-5707) for metrics collection.
  + Ambari leverages [Ambari Alert Framework](https://issues.apache.org/jira/browse/AMBARI-6354) for system alerting and will notify you when your attention is needed (e.g., a node goes down, remaining disk space is low, etc).

Apache Ambari 專案為 Apache License，故在本次測試中，會先以 apache 版本為主，而先不以 Hortornworks 版本，目前初步研究並無相關授權問題。

### Install

1. yum install epel-release
2. yum repolist
3. yum install ambari-server

### Setup

1. <http://10.144.30.26:8080>
2. 確認是否 Hosts 已經被辨識成功
3. 確認要安裝的服務是否正常
4. 指定 Master
5. 只應 Slave 與 Clients

# Operation

# Cloud

本章節說明 Amazon EC2 費用評估流程與結果。

## 佈署流程

* 系統移轉測試（3天）
  + 移轉佈署（1天）：將原系統轉移至 Amazon EC2 約一天，流程內容包括：
    - 準備相關環境資料設定檔（IP, 帳號密碼, Disk Partition…..etc）
    - 系統建置設定
    - 系統功能測試
    - 自動化備份、備援測試
    - Cloudwatch告警通知測試
  + 觀察兩天
    - 確認上述測試結果能穩定運作兩天
  + 費用選擇
    - 隨選型主機
* 系統完整移轉（1年）
  + 費用選擇
    - 預付型主機
    - 預付時間有分一年跟三年，規劃以一年為期因應客戶變動

## 所需項目與費用（2016/4/18 更新）

* + 費用選擇：有分隨選、預付、競價、專用四種型態
  + 我們沒有專用需求，競價型雖然便宜但隨時會被停機，故先以隨選型當作系統測試，再用預付型可享七折折扣最符合需求
* 地區選擇：美國西部（奧勒剛）

### 隨選型（前期測試用，預估三天）

* 主機規格與費用

|  |  |  |  |
| --- | --- | --- | --- |
| 項目 | 內容 | 費用 | |
| 單價 | 總價 |
| EC2  （t2.medium） | * CPU：2 Core * Memory：4G | $ 0.052 每小時 | $ 3.744 每三天 |
| EBS | 20G | $ 0.10 每月佈建儲存的 GB 數 | $ 2.00 每三天 |
| Network in |  | $ 0 | $ 0 |
| Network out |  | $ 0.090 每 GB  （每月前1GB 免費） | $ 9.00 / 100GB / 月 |
| Cloudwatch | Dashboards | $ 3.00 per dashboard per mont | $ 3.00 |
| 執行個體的詳細監控 | $ 3.50 按 1 分鐘頻率 | $ 3.50 |
| 警示 | $ 0.10 每月每警示 | $ 0.10 |
| Events | $ 1.00 per million custom events | $ 1.00 |
| 總價 | | | $ 22.344 |

### 預付型（長期使用）

* 主機規格與費用

|  |  |  |  |
| --- | --- | --- | --- |
| 項目 | 內容 | 費用 | |
| 單價 | 總價 |
| EC2  （t2.medium） | * CPU：2 Core * Memory：4G | $ 0.034 每小時 | $ 302每一年 |
| EBS | 20G | $ 0.10 每月佈建儲存的 GB 數 | $ 2.00 每三天 |
| Network in |  | $ 0 | $ 0 |
| Network out |  | $ 0.090 每 GB  （每月前1GB 免費） | $ 9.00 / 100GB / 月 |
| Cloudwatch | Dashboards | $ 3.00 per dashboard per mont | $ 3.00 |
| 執行個體的詳細監控 | $ 3.50 按 1 分鐘頻率 | $ 3.50 |
| 警示 | $ 0.10 每月每警示 | $ 0.10 |
| Events | $ 1.00 per million custom events | $ 1.00 |
| 總價 | | | $ 320.26 |

# Tasks

| Task | Due Date |
| --- | --- |
| Amazon Evaluation | 4/16 |
| Kafka and Flume Setting | 4/30 |
| Ambari Evaluation | 5/14 |
| Spark Streaming Scalability | 5/7 |
| Spark Cluster AutoScaling with Mesos | 5/28 |
| Spark Streaming integration with Mllib | 6/11 |