RETCHAT

Flume, Kafka and Scalability Spark Streaming

Aiden HuangTable of Contents

[1 Introduction/Purpose 3](#_Toc456036620)

[1.1 Purpose 3](#_Toc456036621)

[1.2 Scope 3](#_Toc456036622)

[1.3 Overview 3](#_Toc456036623)

[2 Version 3](#_Toc456036624)

[3 Setup 3](#_Toc456036625)

[3.1 Kafka 3](#_Toc456036626)

[3.1.1 單機安裝測試 3](#_Toc456036627)

[3.2 Spark 8](#_Toc456036628)

[3.3 Mesos 8](#_Toc456036629)

[3.4 DC/OS 9](#_Toc456036630)

[3.4.1 系統環境準備 9](#_Toc456036631)

[DC/OS Environment Settings 12](#_Toc456036632)

[4 Operation 13](#_Toc456036633)

[5 Cloud 13](#_Toc456036634)

[5.1 佈署流程 13](#_Toc456036635)

[5.2 所需項目與費用（2016/4/18 更新） 13](#_Toc456036636)

[5.2.1 隨選型（前期測試用，預估三天） 13](#_Toc456036637)

[5.2.2 預付型（長期使用） 14](#_Toc456036638)

[6 Tasks 15](#_Toc456036639)

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 | 0.9.0.0 | 2016/04/28 |
| Spark |  |  |

# 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

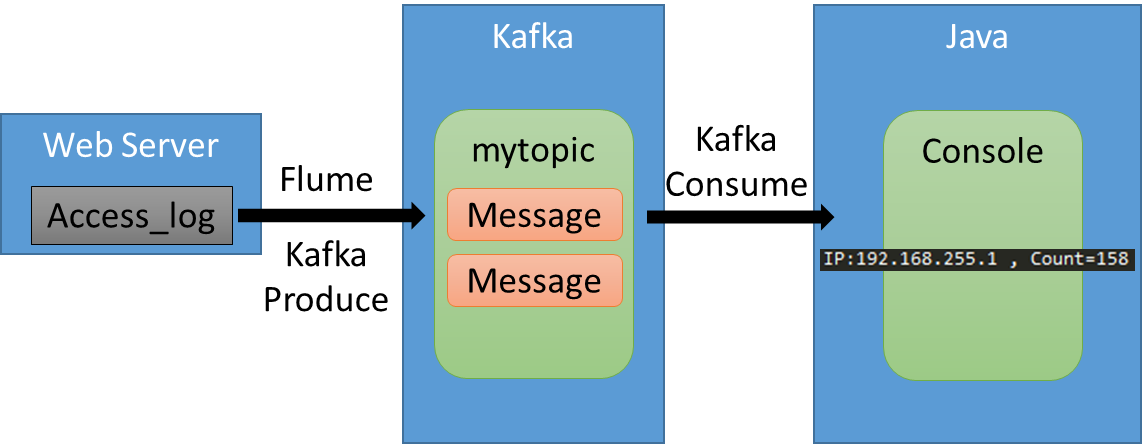
* bin/zookeeper-server-start.sh config/zookeeper.properties &

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)));  
 }  
 }  
 }  
 }  
}

#### Run test code

1. 請先 check out 專案（kafka\_test\_eclipse）
2. 開啟 consumer\_test.java
3. 確認bootstrap.servers 所指定的 kafka server 的 ip 跟 port 正確
4. 確認要存取的 topic 名稱正確（範例為 mytopic）
   * consumer2.subscribe(Arrays.*asList*("mytopic"));
5. 執行consumer\_test.java，如果有訊息進入到 topic，就會持續印出來

## Spark

## Mesos

### 持久化設計

## DC/OS

### 系統環境準備

#### Install Docker on CentOS

1. Upgrade CentOS to 7.2:

$ sudo yum upgrade --assumeyes --tolerant

$ sudo yum update –assumeyes

1. Verify that the kernel is at least 3.10:

$ uname -r

3.10.0-327.10.1.el7.x86\_64

1. Enable OverlayFS:

$ sudo tee /etc/modules-load.d/overlay.conf <<-'EOF'

overlay

EOF

1. Reboot to reload kernel modules:

$ reboot

1. Verify that OverlayFS is enabled:

$ lsmod | grep overlay

overlay

1. Configure yum to use the Docker yum repo:

$ sudo tee /etc/yum.repos.d/docker.repo <<-'EOF'

[dockerrepo]

name=Docker Repository

baseurl=https://yum.dockerproject.org/repo/main/centos/$releasever/

enabled=1

gpgcheck=1

gpgkey=https://yum.dockerproject.org/gpg

EOF

1. Configure systemd to run the Docker Daemon with OverlayFS:

$ sudo mkdir -p /etc/systemd/system/docker.service.d && sudo tee /etc/systemd/system/docker.service.d/override.conf <<- EOF

[Service]

ExecStart=

ExecStart=/usr/bin/docker daemon --storage-driver=overlay -H fd://

EOF

1. Install the Docker engine, daemon, and service:

$ sudo yum install --assumeyes --tolerant docker-engine

$ sudo systemctl start docker

$ sudo systemctl enable docker

When the process completes, you should see:

Complete!

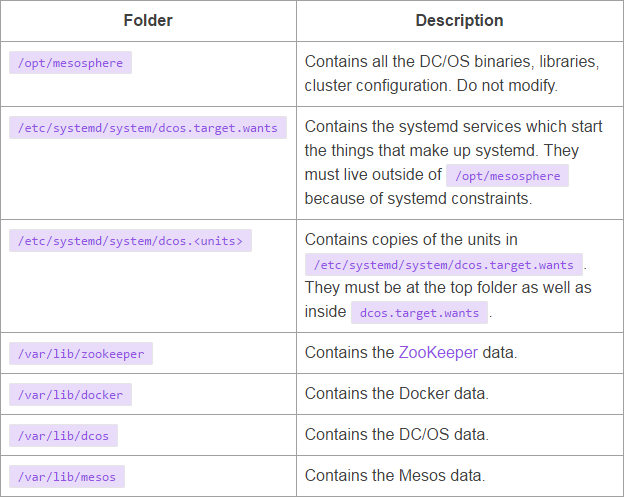
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

1. Test that Docker is properly installed:

$ sudo docker ps

#### DC/OS Installation

相關安裝路徑一覽：



1. Download the [DC/OS installer](https://downloads.dcos.io/dcos/EarlyAccess/dcos_generate_config.sh) to your bootstrap node.

$ curl -O <https://downloads.dcos.io/dcos/EarlyAccess/dcos_generate_config.sh>

1. From your terminal, start the DC/OS GUI installer with this command.

$ sudo bash dcos\_generate\_config.sh --web

Here is an example of the output.

Running mesosphere/dcos-genconf docker with BUILD\_DIR set to /home/centos/genconf

16:36:09 dcos\_installer.action\_lib.prettyprint:: ====> Starting DC/OS installer in web mode

16:36:09 root:: Starting server ('0.0.0.0', 9000)

**Tip:** You can add the verbose (-v) flag to see the debug output:

$ sudo bash dcos\_generate\_config.sh --web -v

1. Launch the DC/OS web installer in your browser at: http://<bootstrap-node-public-ip>:9000.

#### DC/OS SETUP

**Deployment Settings**

* **Master Private IP List** Specify a comma-separated list of your internal static master IP addresses.
* **Agent Private IP List** Specify a comma-separated list of your internal static agent IP addresses.
* **Master Public IP** Specify a publicly accessible proxy IP address to one of your master nodes. If you don’t have a proxy or already have access to the network where you are deploying this cluster, you can use one of the master IP’s that you specified in the master list. This proxy IP address is used to access the DC/OS web interface on the master node after DC/OS is installed.
* **SSH Username** Specify the SSH username, for example centos.
* **SSH Listening Port** Specify the port to SSH to, for example 22.
* **SSH Key** Specify the private SSH key with access to your master IPs.

### DC/OS Environment Settings

* **Upstream DNS Servers** Specify a comma-separated list of DNS resolvers for your DC/OS cluster nodes. Set this parameter to the most authoritative nameservers that you have. If you want to resolve internal hostnames, set it to a nameserver that can resolve them. If you have no internal hostnames to resolve, you can set this to a public nameserver like Google or AWS. In the example file above, the [Google Public DNS IP addresses (IPv4)](https://developers.google.com/speed/public-dns/docs/using) are specified (8.8.8.8 and 8.8.4.4). If Google DNS is not available in your country, you can replace the Google DNS servers 8.8.8.8 and 8.8.4.4 with your local DNS servers.

*Caution:* If you set this parameter incorrectly you will have to reinstall DC/OS. For more information about service discovery, see this [documentation](https://dcos.io/docs/1.7/overview/service-discovery/).

* **IP Detect Script** Choose an IP detect script from the dropdown to broadcast the IP address of each node across the cluster. Each node in a DC/OS cluster has a unique IP address that is used to communicate between nodes in the cluster. The IP detect script prints the unique IPv4 address of a node to STDOUT each time DC/OS is started on the node.

*Important:* The IP address of a node must not change after DC/OS is installed on the node. For example, the IP address must not change when a node is rebooted or if the DHCP lease is renewed. If the IP address of a node does change, the node must be wiped and reinstalled.

## Cassandra

本章節說明 Cassandra 相關佈署架設研究。

### Cluster Setup

#### 規劃

1. 在每個節點安裝Cassandra
2. 為集群選擇一個合適的名字
3. 為每個節點分配一個IP
4. 確定種子節點，但不要把所有節點都作為種子
5. 如果是多數據中心，為每個數據中心和機架確定命名約定，比如：DC1, DC2 或 100, 200 和 RAC1, RAC2 或 R101, R102。

#### 安裝

1. 安裝節點：

node0 192.168.0.111(seed1)

node1 192.168.0.112(seed2)

node2 192.168.0.113

注意：每個數據中心擁有多個種子節點是一條最佳實踐。

1. 如果 cluster 中有防火牆，確保相對應的 port 可以通過，以便節點之間能互相訪問。
2. 如果Cassandra在運行中，必須先停掉，並且清理數據：

停止 Cassandra:

$ ps -ef | grep cassandra

$ sudo kill pid

清理數據:

$ sudo rm -rf /var/lib/apache-cassandra-2.1.9/data /\*

1. 修改cassandra.yaml中的屬性：
   * -seeds: 每個種子節點的內部IP。種子節點不做引導，它負責處理現有集群中新加入的節點。對於新的集群，引導處理會跳過種子節點。
   * listen\_address：如果不設置的話，Cassandra會從系統讀取與主機名對應的本地地址，某些情況下Cassandra獲取的地址並不正確，此時必須手動指定。如：  
       
     cluster\_name: 'monitorCluster'  
     seed\_provider:  
     - class\_name: org.apache.cassandra.locator.SimpleSeedProvider  
       
     parameters:  
     - seeds: "192.168.0.111,192.168.0.112"  
       
     listen\_address:192.168.0.111  
     rpc\_address: 192.168.0.111
2. 配置完所有節點後，先啟動種子節點，一次一個，然後再啟動其他節點

$ cd /var/lib/apache-cassandra-2.1.9

$ sudo bin/cassandra

1. 最後檢查集群運行狀態：

$ cd /var/lib/apache-cassandra-2.1.9

$ bin/nodetool status （這個命令可以不加sudo）

# 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 | 5/7 |
| Spark Streaming Scalability | 5/14 |
| Spark Auto Scaling | 7/2 |
| DC/OS Installition | 7/10 |