2017/04/04 - Bigdata Use Case

Problem:

Design a Big Data architecture that can be used to process the clickstream data to extract insights for Business Analytics, Text Analytics, Recommendation Engines, Data Services, etc

Proposed Solution:

After digging into the current technologies for Bigdata, the solution I chose to try was send the logs clickstrem from multiple web server through Logstash into Kafka. From there we are going to use Spark Streaming to get the data into an Hadoop Cluster managed using the Cloudera tools.

Devops Tools

For the following slides, I'm going to use these tools:

- Invoke to create simple commands (fabric like)
- Ansible for host config/setup automation

Everything is under the `fdevops` directory.

Launch EC2 for Cloudera Director

To build our use case, let's start to Install and Setup the Hadoop Cluster. I'm going to use Cloudera Director on Amazon EC2 following the instructions here. I chose to use Ubuntu 16.04 because it's the Linux distribution that I most use.

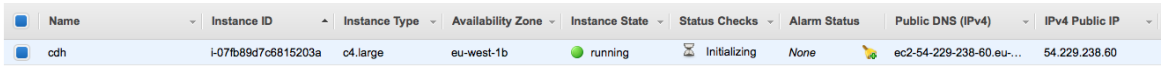
```
$ invoke prd.ec2_create c4.large cdh

PLAY [Create EC2] ***************************************************************

TASK [aws-ec2-instance : Create an EC2 instance] ********************************
changed: [localhost -> localhost]

PLAY RECAP **********************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0
```

And on AWS EC2:



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
cdh	i-07fb89d7c6815203a	c4.large	eu-west-1b	running	Initializing	None	ec2-54-229-238-60.eu-...	54.229.238.60

Configure EC2

Get the public IP

```
$ invoke prd.ec2_refresh_cache
$ invoke prd.ec2_list cdh
  hosts (1):
      54.229.238.60
```

Update the file `inventory/hosts` and run

```
$ invoke prd.cdh_update cdh
[...]
TASK [srv-cdh-apt : Setup cloudera director repo key] **************************
TASK [srv-cdh-apt : Setup cloudera director repo] ******************************
TASK [srv-cdh-apt : Apt Update] ************************************************
TASK [srv-cdh-apt : Apt install required packages] *****************************
TASK [srv-cdh-apt : Apt install cloudera] **************************************
TASK [srv-cdh-setup : Restart Cloudera Director Server] ************************
```

Access to Cloudera Director

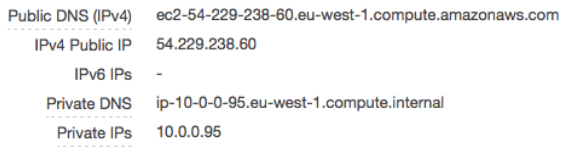
Cloudera Director is not accessible from the public, so I used the Step 1: Set Up a SOCKS Proxy Server with SSH suggested by Cloudera.

Open 2 terminals and run the following commands on each of them:

```
$ invoke prd.cdh_start_proxy
```

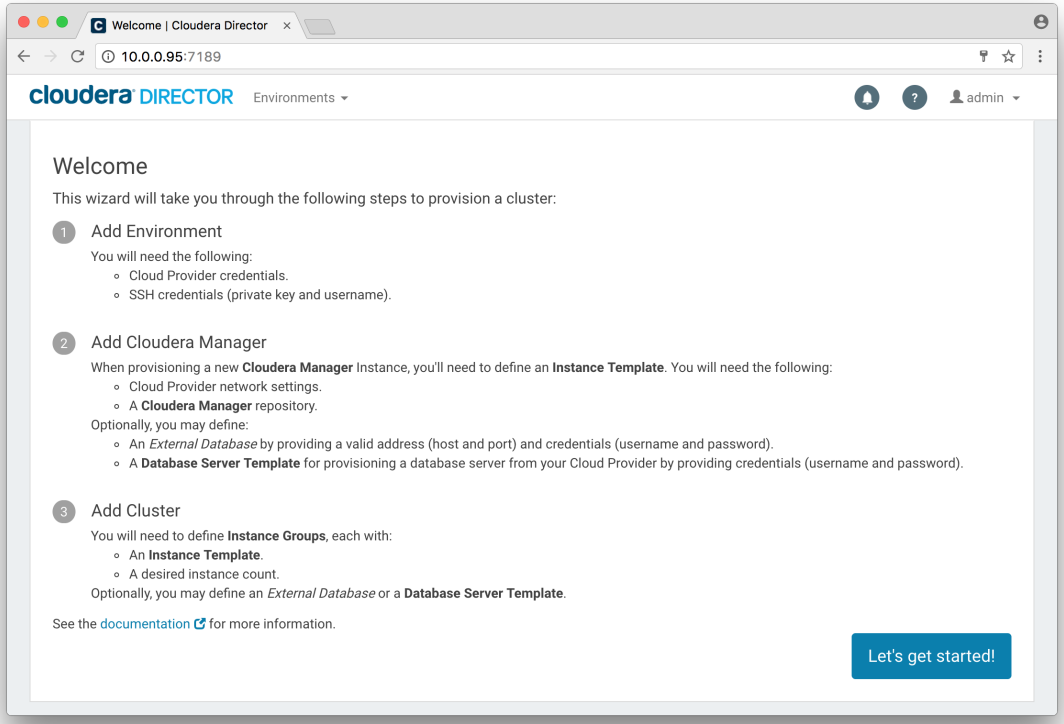
```
$ invoke prd.cdh_start_chrome
```

The last command launches a Chrome browser able to connect to Cloudera Director, for this you need to go to the EC2 Console, and search for the private IP of the machine.



Public DNS (IPv4)	ec2-54-229-238-60.eu-west-1.compute.amazonaws.com
IPv4 Public IP	54.229.238.60
IPv6 IPs	-
Private DNS	ip-10-0-0-95.eu-west-1.compute.internal
Private IPs	10.0.0.95

Cloudera Director



Cloudera Manager

Following the documentation on [Deploying Cloudera Manager and CDH on AWS]

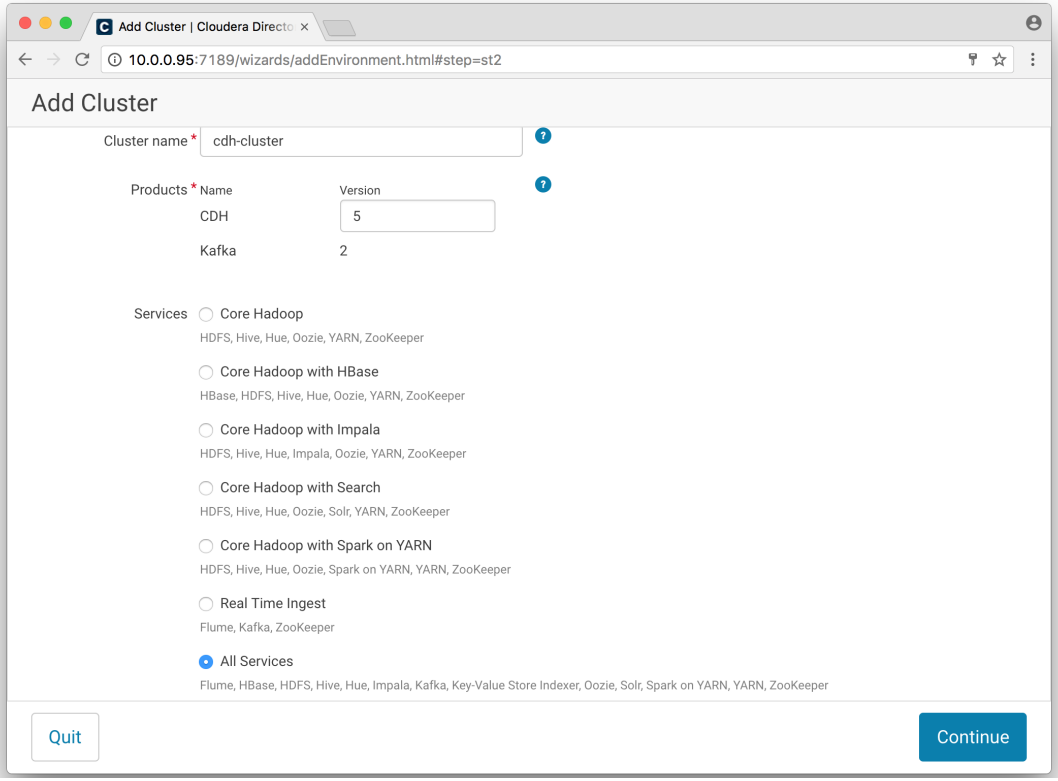
1. We need to create an environment with our AWS credentials
2. Configure an Instance Template
3. Choose a license, in my case the trial license

Setup the Cluster 1/4

Time to setup and run the Cluster

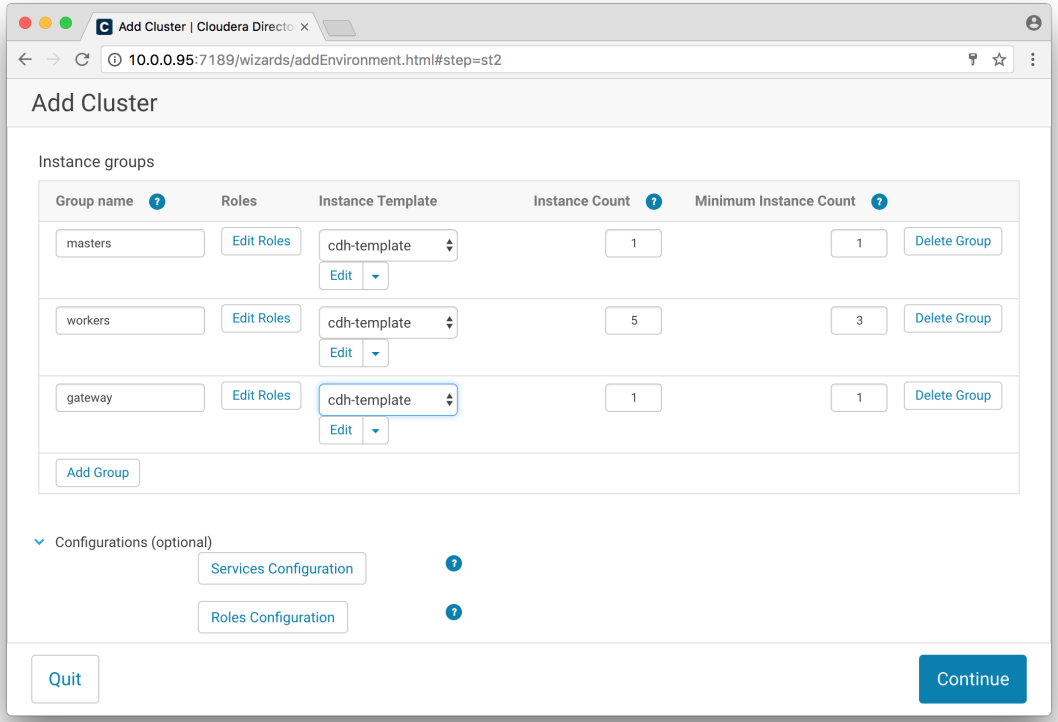
1. Configured 'cdh-cluster' as the cluster name
2. Chose to install all services
3. For the instance groups "workers" I chose only 5 instances
4. Configure the instance templates for the instance groups

Setup the Cluster 2/4

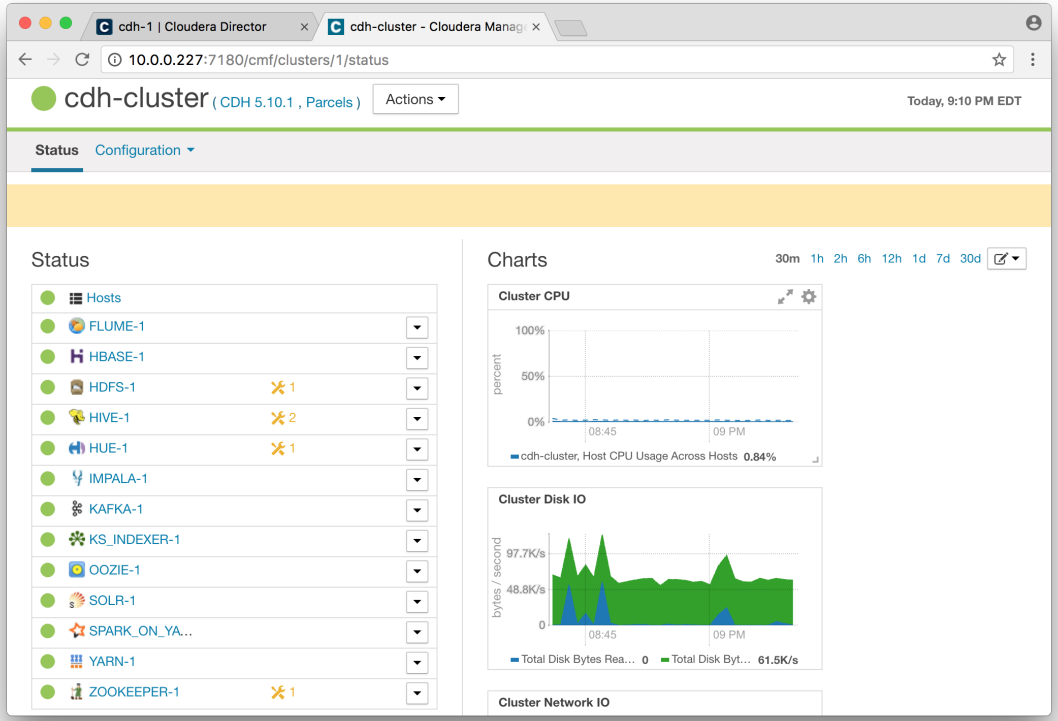


10 / 22

Setup the Cluster 3/4



Setup the Cluster 4/4



Cluster is on AWS

All instances are up and running

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
cdh-instance	i-04f771b005f225732	m4.large	eu-west-1b	running	2/2 checks ...
cdh	i-07fb89d7c6815203a	c4.large	eu-west-1b	running	2/2 checks ...
cdh-instance	i-09369731cd5513377	m4.large	eu-west-1b	running	2/2 checks ...
cdh-instance	i-0b42731dbd05258...	m4.large	eu-west-1b	running	2/2 checks ...
cdh-instance	i-0b4d80ebd7d0a8d...	m4.large	eu-west-1b	running	2/2 checks ...
cdh-instance	i-0d6d8b4d6eeace119	m4.large	eu-west-1b	running	2/2 checks ...
cdh-instance	i-0d745125e27b93d...	m4.large	eu-west-1b	running	2/2 checks ...
cdh-instance	i-0e4c08ace2b282563	m4.large	eu-west-1b	running	2/2 checks ...
cdh-instance	i-0e80095efbd7d9651	m4.large	eu-west-1b	running	2/2 checks ...

Logstash

To start, let's create an instance with Logstash installed. After that we can send logs into Logstash and then connect it with Kafka.

```
$ invoke prd.ec2_create t2.medium logstash_1

PLAY [Create EC2] **************************************************************

TASK [aws-ec2-instance : Create an EC2 instance] *******************************
changed: [localhost -> localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0
```

Get the public IP

```
$ invoke prd.ec2_refresh_cache
$ invoke prd.ec2_list logstash_0
  hosts (1):
    54.246.237.120
```

And then Update the file `inventory/hosts`

Setup Logstash - generate logs

Before we run the `invoke` command to setup Logstash, let's just take a moment to think about it.

We don't have a fully working web server to feed logs to logstash

Using a sample log file let's make a python script to generate logs forever

```
def print_file(sleep=5, limit=10):
    with open('/home/ubuntu/ctlogs-.1438663216674') as fh:
        count = 0
        for line in fh:
            sys.stdout.write(line)
            count += 1
            if count >= limit:
                count = 0
                time.sleep(sleep)


def main():
    while True:
        print_file(5, 10)
```

Setup Logstash - clickstream.conf

For the `input` we use the `pipe` plugin to run our python script

Because we already have log sample shaped to our needs, we don't need to configure filters, and we just send the log lines directly to kafka.

```
input {
    pipe {
        command => "/home/ubuntu/gen_logs.py"
    }
}
```

And then configure the `output` to send to kafka, and for that we need to find the private IP in **Cloudera manager** > **KAFKA-1** > **instances**

```
output {
    kafka {
        bootstrap_servers => "10.0.0.151:9092"
        topic_id => "clickstream"
        codec => plain {
            format => "%{message}"
        }
    }
    stdout { }
}
```

Setup Logstash - nuisances

The Logstash output plugin is not compatible with the Kafka version instaled on the Cloudera Cluster.

To solve this I needed to downgrade `logstash-output-kafka` plugin to version 4.0.4

Setup Logstash - Invoke command

```
$ invoke prd.log_update logstash_1
PLAY [Logstash Setup] **********************************************************
TASK [setup] *******************************************************************
TASK [srv-log-apt : Setup logstash repo key] ***********************************
TASK [srv-log-apt : Setup logstash repo] ***************************************
TASK [srv-log-apt : Apt Update] ************************************************
TASK [srv-log-apt : Apt install required packages] *****************************
TASK [srv-log-apt : Apt install logstash] **************************************
TASK [srv-pyenv : Install pyenv] ***********************************************
TASK [srv-pyenv : Set bash_aliases] ********************************************
TASK [srv-pyenv : Set .profile] ************************************************
TASK [srv-pyenv : Install python] **********************************************
TASK [srv-log-setup : Copy ctlogs] *********************************************
TASK [srv-log-setup : Copy gen_logs.py] ****************************************
TASK [srv-log-setup : Copy clickstream.conf] ***********************************
TASK [srv-log-setup : Stop logstash to remove plugin] **************************
TASK [srv-log-setup : Remove logstash-output-kafka plugin] *********************
TASK [srv-log-setup : Install logstash-output-kafka plugin version 4.0.4] ******
TASK [srv-log-setup : Start logstash] ******************************************
```

Kafka

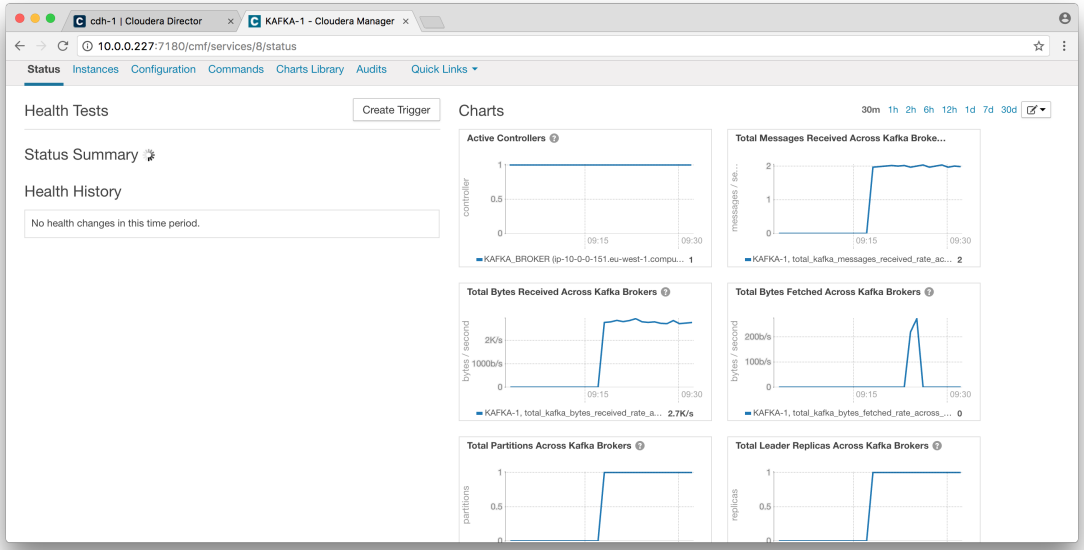
To check if the logs are be sent to Kafka we can start a consumer in the shell with the command:

```
kafka-console-consumer --zookeeper 10.0.0.110:2181 --topic clickstream
```

And the logs being fed to logstash should now being showed on the screen.

Kafka - Status

Cloudera Kafka status page



This is a Work In Progress!

Next steps:

1. Configure a schema in HBase
2. Configure Spark Streaming to consume kafka clickstream into HBase
3. Work on the Exploration by Data Scientist and Data Analysts
4. Create real-time use case
5. Think about use cases and to do queries to fetch relevant data from HBase
6. Test the scalability of the architecture, and think about automation and strategies

References

- https://www.elastic.co/webinars/getting-started-logstash?baymax=rtp&elektra=products&iesrc=ctr
- https://kafka.apache.org/documentation/
- https://www.elastic.co/guide/en/logstash/current/plugins-outputs-kafka.html
- https://github.com/logstash-plugins/logstash-output-kafka
- https://www.elastic.co/guide/en/logstash/current/plugins-inputs-pipe.html
- http://eagle.apache.org/docs/cloudera-integration.html
- https://github.com/dpkp/kafka-python
- http://docs.ansible.com/ansible/intro.html
- https://mapr.com/blog/guidelines-hbase-schema-design/