

CS 441

COURSE PROJECT DOCUMENTATION

Real time Log files processing pipeline using Akka, Apache Kafka and Spark deployed on EC2 instances

Team Members

Sai Nadkarni

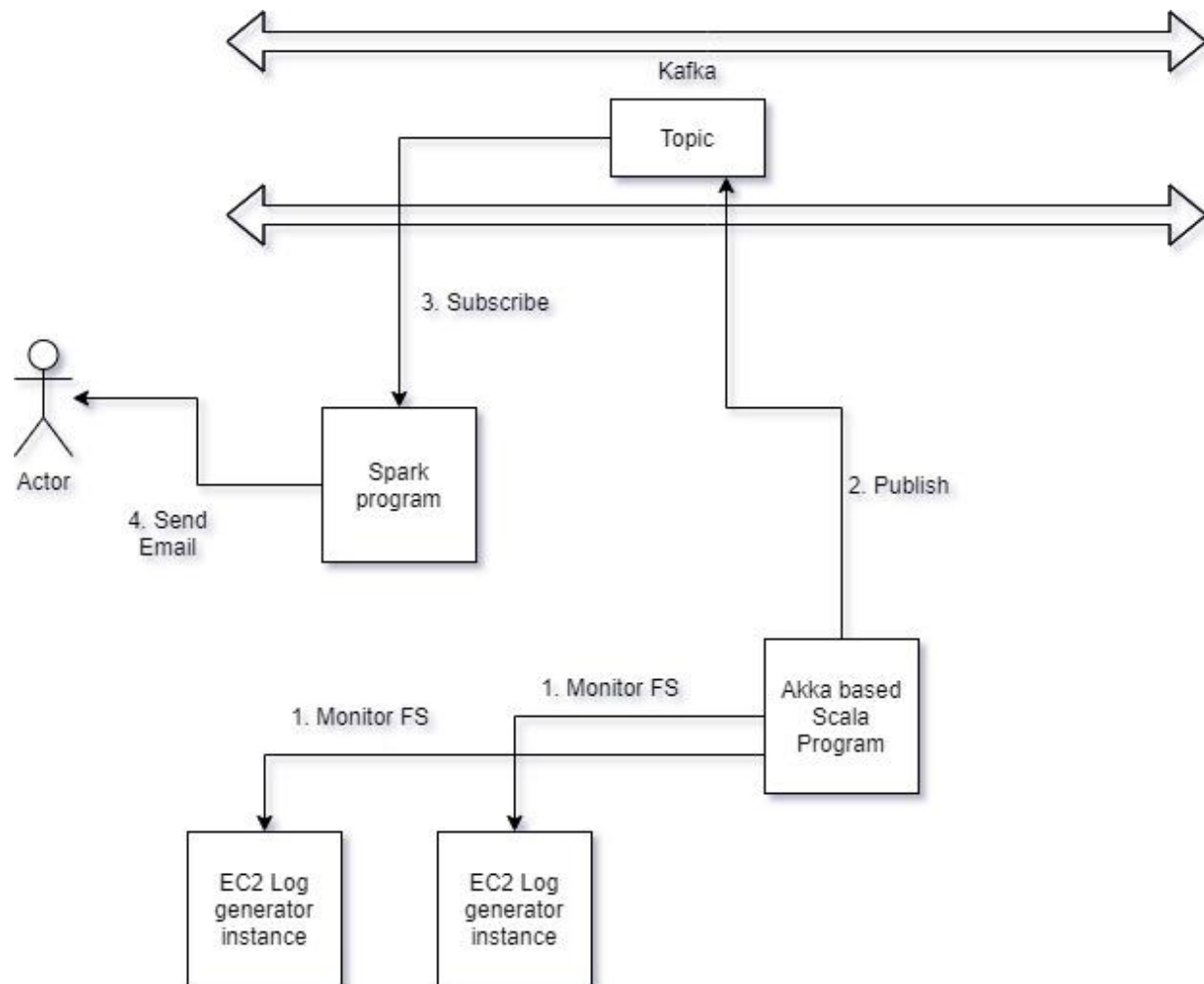
Ankit Kumar Singh

Sebastian Greczek

Sanchita Kapoor

Bhavana Nelakuditi

Basic Outline



Overview

In this project we have implemented a streaming data pipeline using cloud computing technologies like Amazon Elastic Compute Cloud (EC2) and by designing and implementing an actor - model service using Akka that is monitoring log files to which logs are added in real time and sends it to Spark using an event based Service Kafka. The project is deployed on AWS.

Development Environment

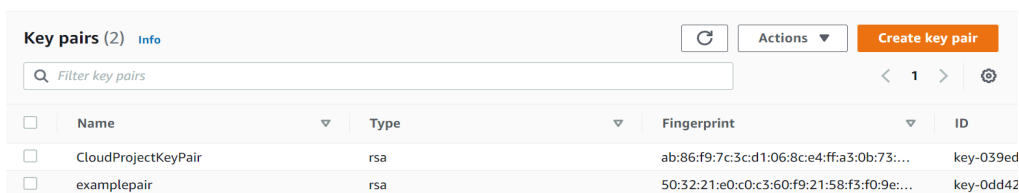
- Language: Scala v2.13.7
- IDE: IntelliJ IDEA 2021.3
- Build Tool: SBT v1.3
- Framework Used:
 - Akka 2.5.26
 - Akka HTTP 10.1.11
 - Kafka 3.0.0
 - Spark Core 3.0.3

Setting up the Environment

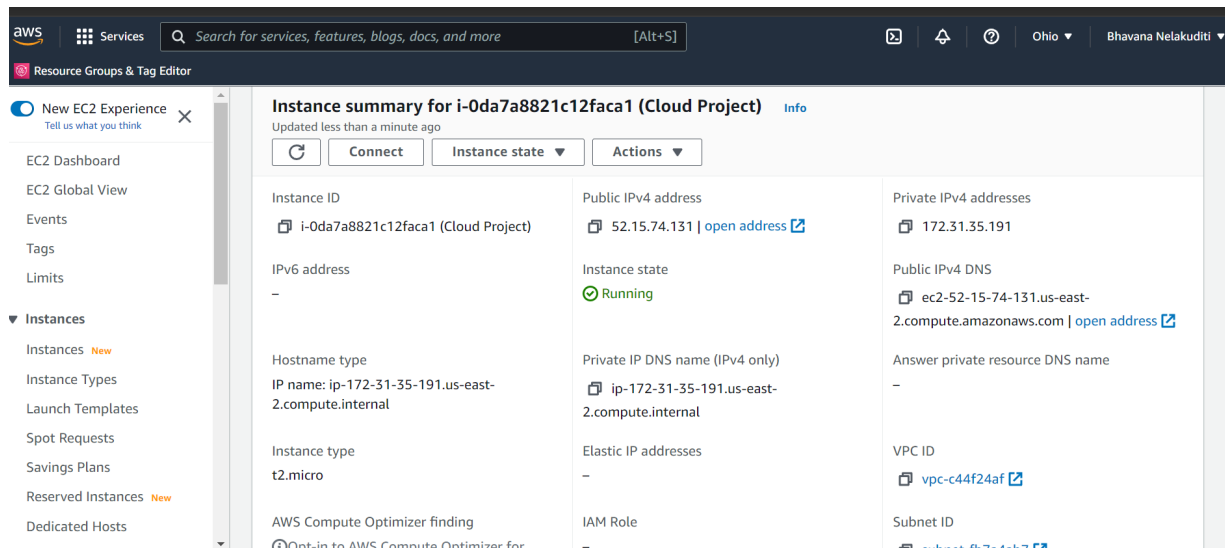
The Development Environment mentioned above is required to be set up in the EC2 instance before running the program. Different versions of the environment may lead to inconsistencies.

Setting up EC2:

Create EC2 instances with Amazon Linux 2, Key pairs and Security groups using AWS console. For our execution set up is as below.



Key pairs (2) Info					
<input type="text" value="Filter key pairs"/>					
< 1 > ⚙					
<input type="checkbox"/>	Name	Type	Fingerprint	ID	
<input type="checkbox"/>	CloudProjectKeyPair	rsa	ab:86:f9:7c:3c:d1:06:8c:e4:ff:a3:0b:73:...	key-039ed	
<input type="checkbox"/>	examplepair	rsa	50:32:21:e0:c0:c3:60:f9:21:58:f3:f0:9e:...	key-0dd42	



Connect to the instances with SSH client and your private Key file using the command:

```
ssh -i <"PrivateKey.pem"> <Public DNS of the instance>
```

Setting up Git:

1. `sudo yum update -y`
2. `Sudo yum install git -y`

Setting up SBT

SBT can be set up along with JDK using SDKMAN

1. `curl -s "https://get.sdkman.io" | bash`
2. `source "$HOME/.sdkman/bin/sdkman-init.sh"`
3. `sdk install java $(sdk list java | grep -o "8\.[0-9]*\.[0-9]*\hs-adpt" | head -1)`
4. `$ sdk install sbt`

Setting up Kafka

Download Kafka Binaries from [here](#) and unpack them:

- ```
1. tar -xzf kafka 2.13-3.0.0.tgz
2. cd kafka 2.13-3.0.0
```

---

## Setting up Spark

Make sure Java and Python are set up since Apache Spark requires Java 8.

Download Spark from [here](#) and its [winutils.exe](#) file.

Choose a Spark release and the Pre-built for Apache Hadoop 2.7 package.

Add the winutils.exe file to the /hadoop/bin folder after extracting spark. Once Spark is installed go to the spark directory bin folder and run `spark-shell.cmd`

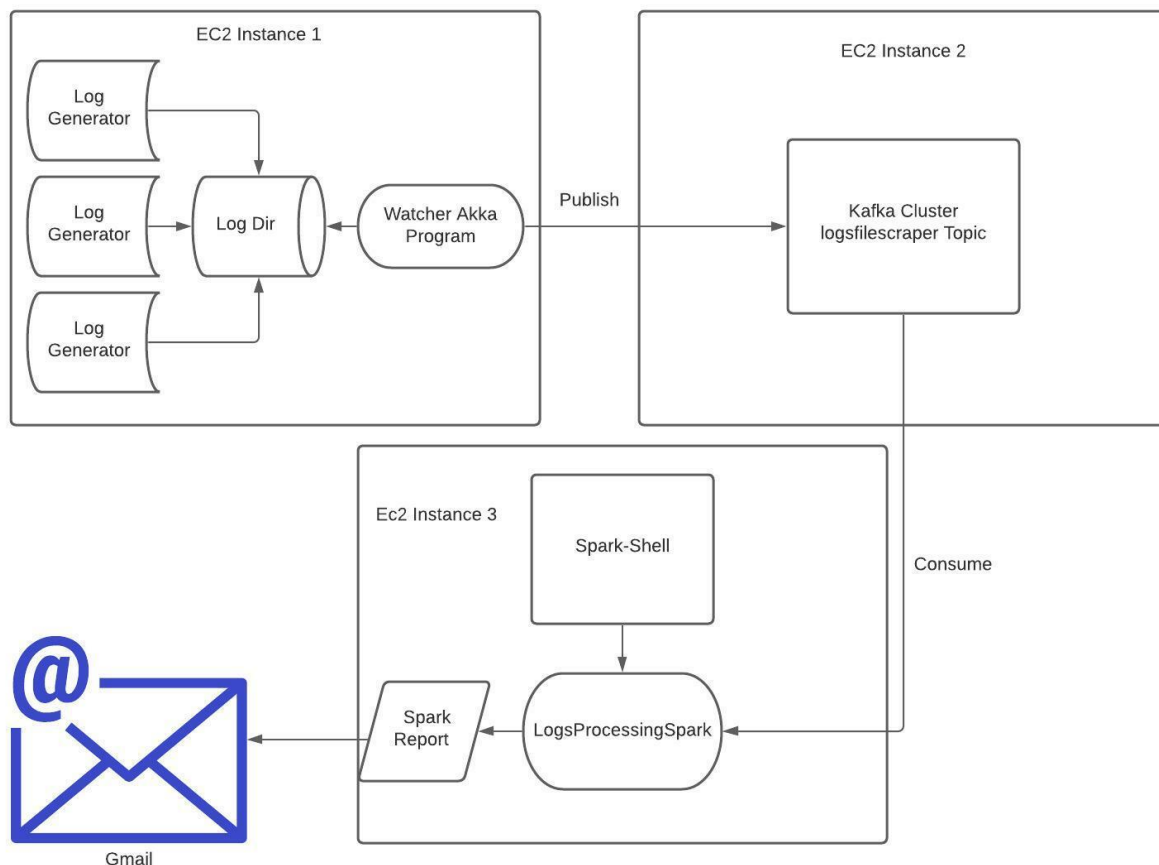
## Steps to Run

- Clone this repository and navigate to the directory its been stored in each of the EC2 instances and run one module in each.
- Go to KafkaConsumer Directory and run
  - `sbt clean compile`
  - `sbt test`
  - `sbt run`
- Go to Watcher Directory and run
  - `sbt clean compile`
  - `sbt test`
  - `sbt run`
- Go to LogFileGenerator Directory and run
  - `sbt clean compile`
  - `sbt test`
  - `sbt run`

---

## Code Structure and Flow:

### Basic Pipeline:



### Akka:

The Akka modules provide Akka Actors Model which allow a higher level of abstraction for writing concurrent and distributed programs. Akka Actors which are implemented in the Watcher directory are looking out for changes in the Log files generated by the Log File Generator. Java NIO is implemented in the Watcher program to watch for the changes in the Log Directory. Akka Actor Model checks the logs generated in real time and if it receives more than four ERROR or WARN messages, these logs are sent to the Kafka Cluster.

---

## Kafka:

Kafka is a distributed event streaming platform that lets us process, read, write and store events i.e messages across machines. Log messages are sent as events from the Akka actor system to the kafka cluster. We have Zookeeper running on this instance that keeps track of a node's status and maintains a list of topics. Zookeeper is used for service synchronisation in distributed systems and in Apache Kafka, Zookeeper is used to track the status of nodes in the Kafka cluster and maintain the list of Kafka topics and messages.

In Kafka, Events are stored and organised in topics which is like a file system with topics being the folders and events being the files.

Our events are under the topic "logfilescrapper".

Kafka Broker, which works as a controller, allows consumers to fetch messages by topic and can create a kafka cluster by sharing information between each other directly or indirectly using zookeeper. We have our kafka broker running on this instance where the Watcher program acts as producer and the "LogProcessingSpark" Program acts as a consumer. "KafkaConsumer" and "KafkaProducer" modules contain this implementation.

## Spark:

Spark, which is an extension of the map/reduce computational model with considerable scheduling and performance improvements and works on the concept of datasets that can be reused several times, unlike key-value pairs in map/reduce. Apache Spark is a distributed processing system for big data workloads that makes use of in-memory caching and optimized query execution.

Spark shell running in the EC2 instance provides the spark context that connects to the spark cluster. In the LogProcessingSpark module, we utilize the spark context to process our logs. We count the number of ERROR and WARN log messages, provide the time frame of the logs and provide a message of the first and last log. This processing is bundled into a report and is sent as an email to the address "[sparkreport8@gmail.com](mailto:sparkreport8@gmail.com)" created for the project using Javax Mail API.

---

## Results:

Email sent to “[sparkreport8@gmail.com](mailto:sparkreport8@gmail.com)” which contains the processed report of the log files generated in a time frame with ERROR and WARN messages.

**sparkreport8@gmail.com**

to me ▾

This report is generated for logs between the time frame: 01:10:01.075 --- 01:10:59.488

There were: 27 ERROR messages

There were: 10 WARN messages

First 10 logs in this time period:

```
01:10:01.075 [scala-execution-context-global-178] WARN HelperUtils.Parameters$ - JeRo)-dH{b3(R)ui3+INR^wG9l(gD*s/uOH$~
01:10:01.692 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - OUshZ0)O6^*B#DZmKMx|nkY(=Y-$*DI<+ISBsRo8*{rm7dhk"F5f}aq+@
01:10:02.450 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - _*7OXObcz(qYb4s,"pO%#*@"u87.]
01:10:03.353 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - QWl)-IQV*,p|Ym;a5LYs@F]V#)bBe,XyTW>6gbKZuwEz3z{
01:10:04.239 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - E"_[P[d<FULG>>)9[V>K&2D3wh,C_Y</JON}q
01:10:06.918 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - waRGlcpFB6S,B|ixXMJnahe>X0f>`16F&?8)O=lb~*1g*Q||GL0FU
01:10:07.901 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ -)RA, .VT9#&|1szp,%95NmpUQjBC"CxD#qcO
01:10:08.547 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - Lr8biaXhJitPLh`c|QB9lc,z2,Rh8>kD}R,RJ2*J0cf=SL
01:10:09.332 [scala-execution-context-global-178] WARN HelperUtils.Parameters$ - *j(YRk_Vi2B,+*ag3@hacoXt*{hf~h{ab
01:10:10.126 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - -Q}zZz{,9~I<TL
```

Last 10 Logs in this time period

```
01:10:44.263 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - tAlzIG5L0"dU|U&|NR1'mhf~{,RX 6=nkqsHG:|uZwDK23XS
01:10:45.106 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - .ay>Q%#ke:Q&*6Wi)%zDCS9NK|+IQ04I7Ss3(BB.|UrTY3WqsVfF0
01:10:52.474 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - z,ZeFmmlBz9q
01:10:53.112 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - EX[_GHH2w/gxYAcf0Z8uR5uV9wgc2Ko'+Fg1|uTe&%)
01:10:55.280 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - 3<y>mNW|x@?$tb3@S@Q0MP{HP|IG
01:10:56.365 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - e@cWw!N.rP@,.$2"J-[$38u^UaJQB.X
01:10:56.971 [scala-execution-context-global-178] WARN HelperUtils.Parameters$ - <g#Hqo'0&K2b(LastOmLG5iaf1N5mag3Y8pce2bg0be370g'OA/4\aoX8LV+6RJdJ
01:10:57.739 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - IR1QLs.KIDL|pv|JuC(EN~A>_]be2cg0P5gcf2J8kae0cg3be3bg0cg3bf3K8o'Fx9K-U2>NeRh%Uu'd!{Z#
01:10:58.680 [scala-execution-context-global-178] ERROR HelperUtils.Parameters$ - :(7I'j|(<U#LU@A(|,FqNtpin5TSLTuiH/NbP@DVweS'
01:10:59.488 [scala-execution-context-global-178] WARN HelperUtils.Parameters$ - \Zo<9al8O/rRJAf)P8jL7icf3ae3cf1S5k&kvI7gdafe@Az0W4
```

AWS Deployment video is hosted on youtube at <https://youtu.be/vj4cN4khn8A>

## Future Work:

Akka Actor system can be made to implement AWS API with an EC2 Tag-Based Directory with tags marking instances that belong to the same cluster with required IAM roles and access.



---

## References:

<https://www.linuxtutorial.co.uk/install-sbt-on-amazon-linux/>

<https://cloudaffaire.com/how-to-install-git-in-aws-ec2-instance/>

<https://www.scala-sbt.org/1.x/docs/Installing-sbt-on-Linux.html>

<https://sdkman.io/>

<https://phoenixnap.com/kb/install-spark-on-windows-10>

<https://github.com/cdarlint/winutils>

<https://doc.akka.io/docs/akka/current/typed/guide/introduction.html>

<https://jaceklaskowski.gitbooks.io/apache-kafka/content/kafka-brokers.html>

<https://databricks.com/spark/about>

Mark Grechanik, Cloud Computing Theory and Practice, 2020