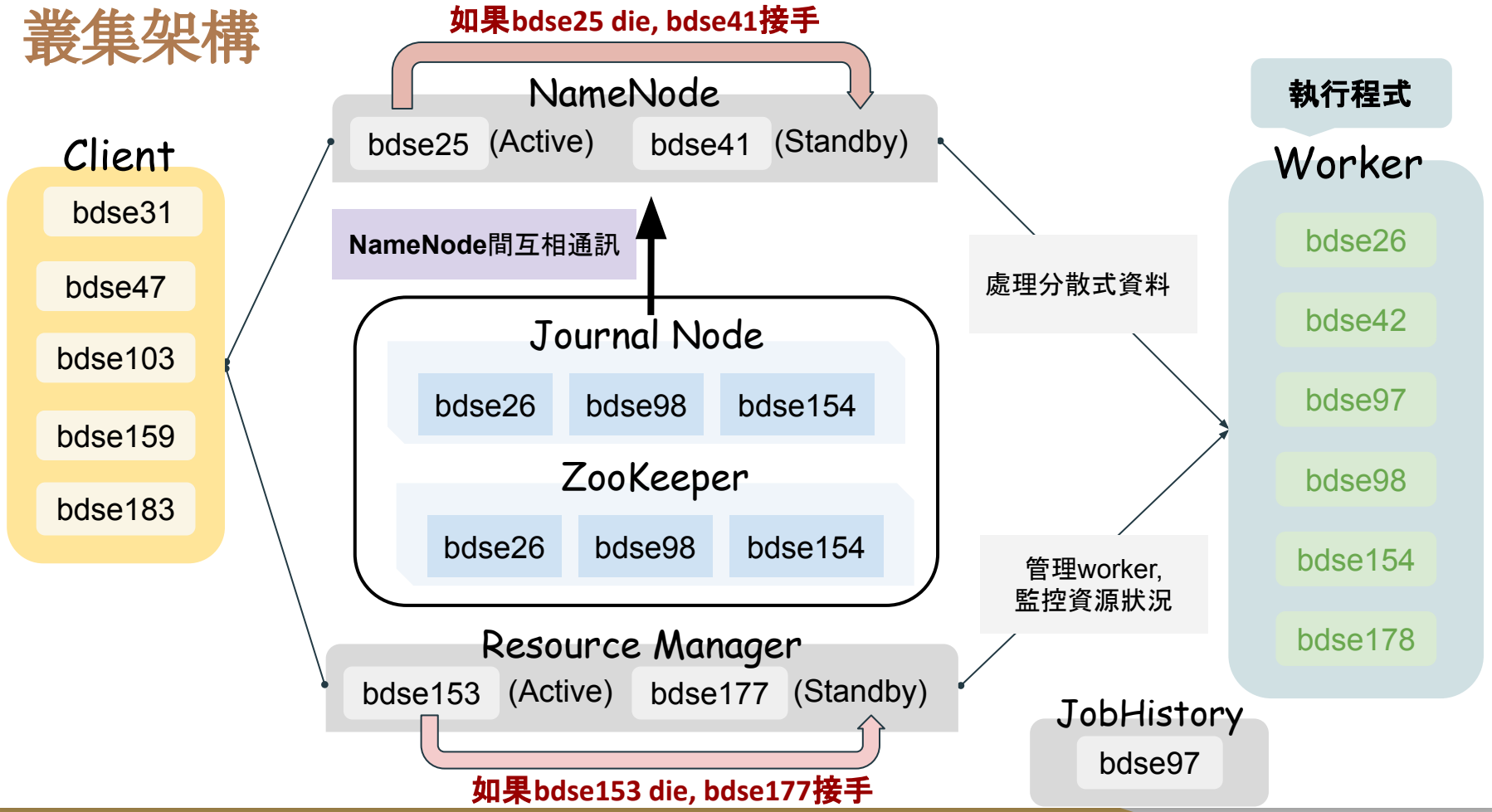
HADOOP叢集架設

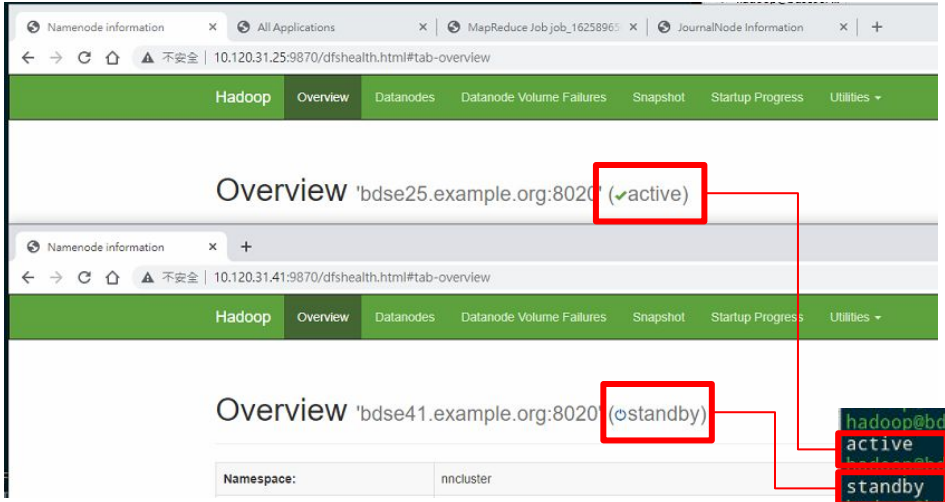
第三組　秦忻榆 林循恩 魏辰儒 徐佳芸 游敏卉

叢集架構


如果bdse25 die, bdse41接手
NameNode
bdse25 (Active)
bdse41 (Standby)
執行程式
Client
bdse31
bdse47
bdse103
bdse159
bdse183
NameNode間互相通訊
Journal Node
bdse26
bdse98
bdse154
ZooKeeper
bdse26
bdse98
bdse154
處理分散式資料
Worker
bdse26
bdse42
bdse97
bdse98
bdse154
bdse178
管理worker,
監控資源狀況
Resource Manager
bdse153 (Active)
bdse177 (Standby)
JobHistory
bdse97
如果bdse153 die, bdse177接手

Name Node	負責管理對用戶端的檔案存取、維護和管理 DataNodes並分配任務於其以及儲存metadata，如檔案名稱、block個數、block位置、replica個數等
Resource Manager	在 **YARN** 系統中所有應用程式之間決定資源如何分配
Job History Server	儲存**ResourceManager**分配資源的詳細資料(ex:啟動、完成時間)，使用者可查看過去Hadoop的使用訊息
Node Manager	管理每個**Worker**的運作，與**ResourceManager**保持聯繫，並監視資源使用狀況，回報(cpu、memory、disk、network等)情況
Data Node	執行**MapReduce**(運算處理層)所分配的任務。以及 HDFS(分散式檔案系統)指揮，完成檔案的建立、刪除、複製、存取等指令
Journal Node	同步**Active NameNode** 與 **Standby NameNode** 的資料

Hadoop叢集架構(HA)

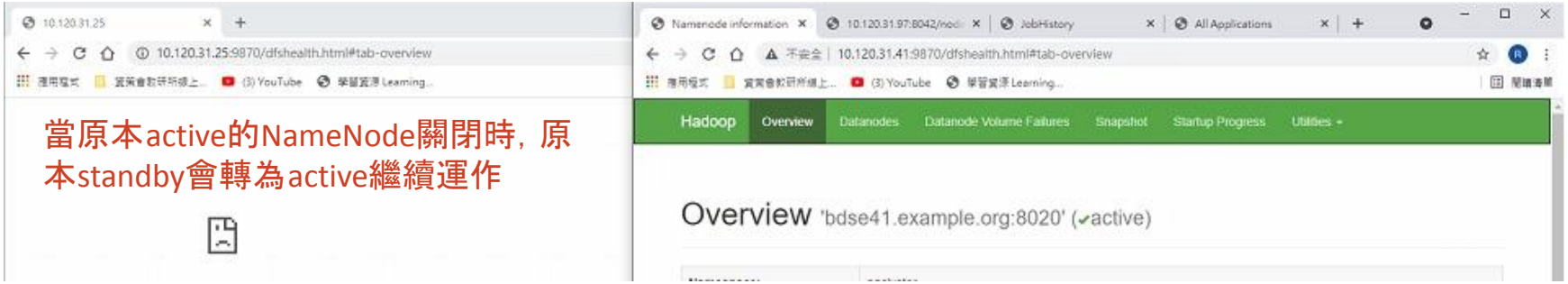


連接至9870確認, active及standby狀況
http://10.120.31.25:9870/
http://10.120.31.41:9870/

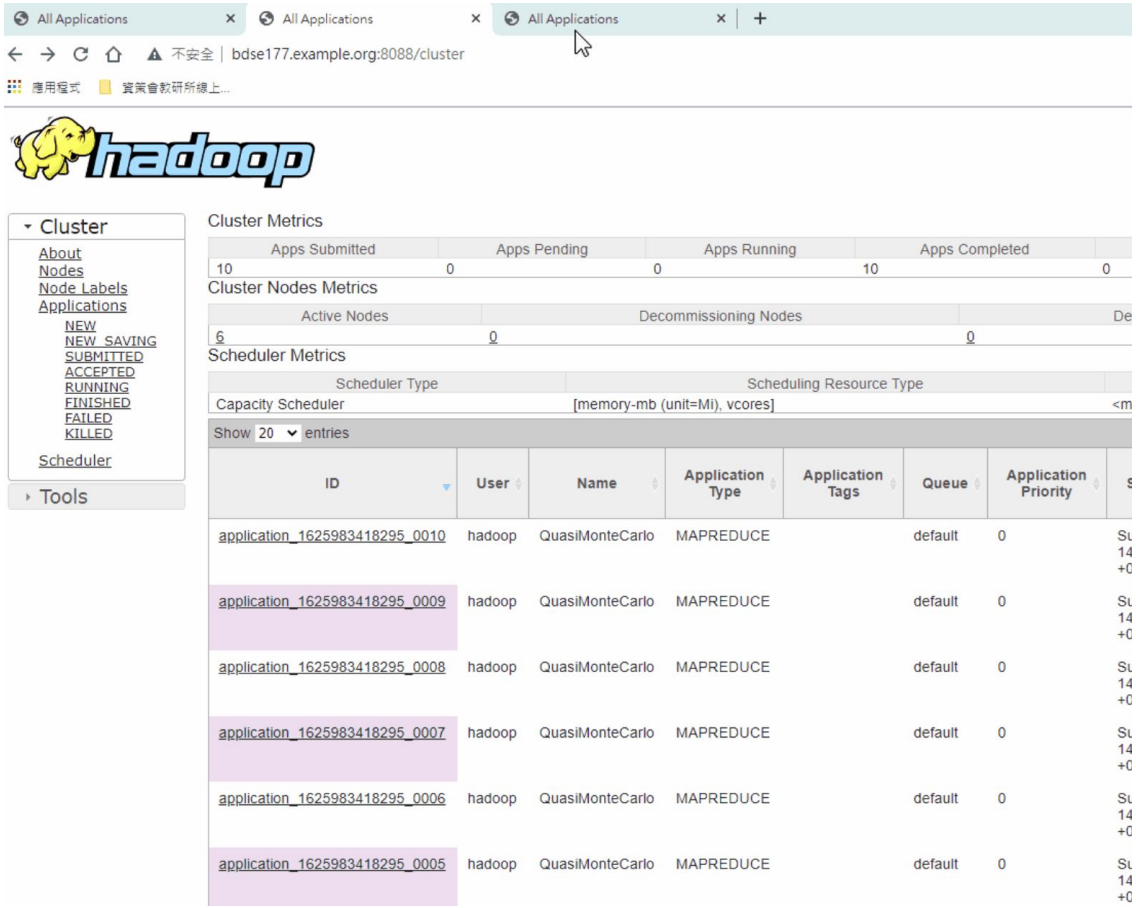
或者

下指令確認NameNode其中一個啟動

```
hadoop@bdse98:/usr/local/zookeeper/conf$ hdfs haadmin -getServiceState nn1
active
hadoop@bdse98:/usr/local/zookeeper/conf$ hdfs haadmin -getServiceState nn2
standby
hadoop@bdse98:/usr/local/zookeeper/conf$
```



當原本active的NameNode關閉時，原本standby會轉為active繼續運作



有兩台Resource Manager, 分別為active(bdse177), standby(bdse153)

開啟網頁bdse177(8088), bdse153(8088), 其中bdse153(8088)會自動redirect到bdse177(8088)

bdse177: yarn --daemon stop resourcemanager

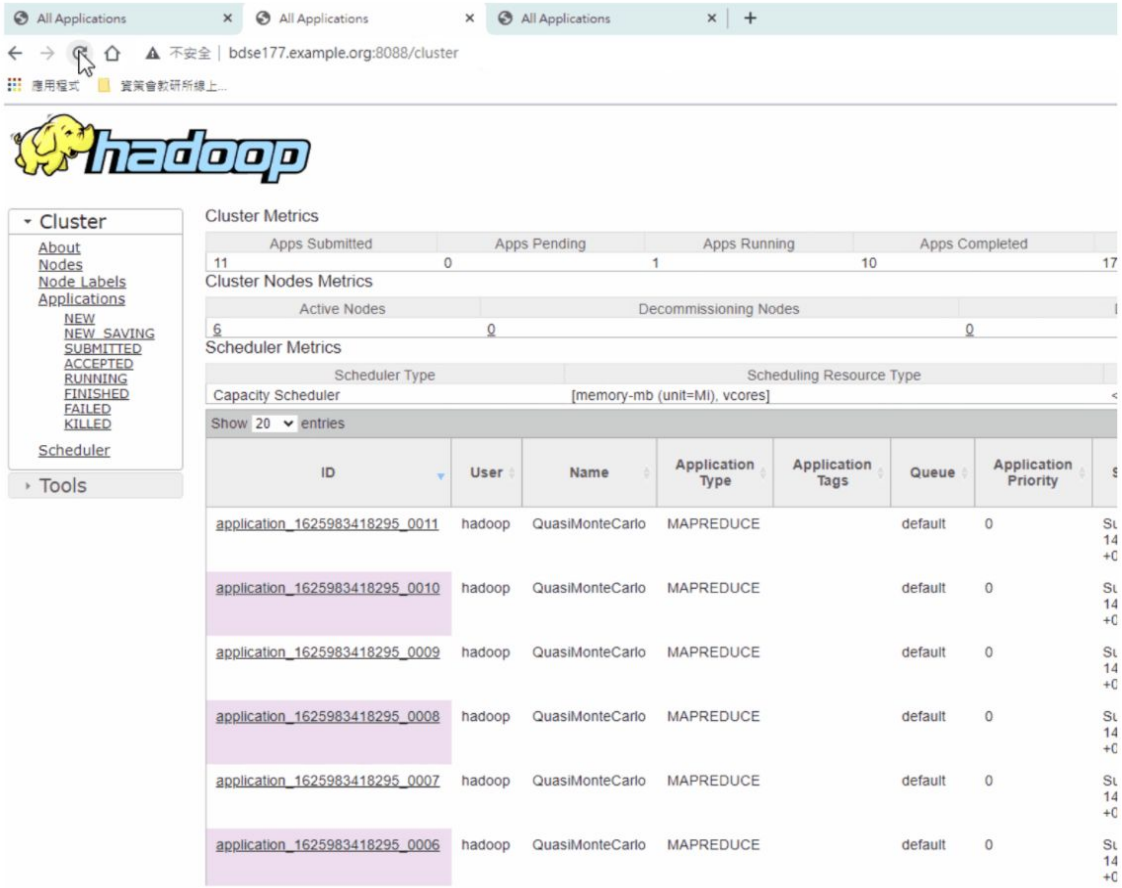
自動跳轉到standby 的RM(bdse153)

網頁bdse177將無法連線

網頁bdse153畫面將會出現

bdse177: yarn --daemon start resourcemanager

bdse177則變為standby



有兩台Resource Manager, 分別為active(bdse177), standby(bdse153)

開啟網頁bdse177(8088), bdse153(8088), 其中bdse153(8088)會自動redirect到bdse177(8088)

bdse177: yarn --daemon stop resourcemanager

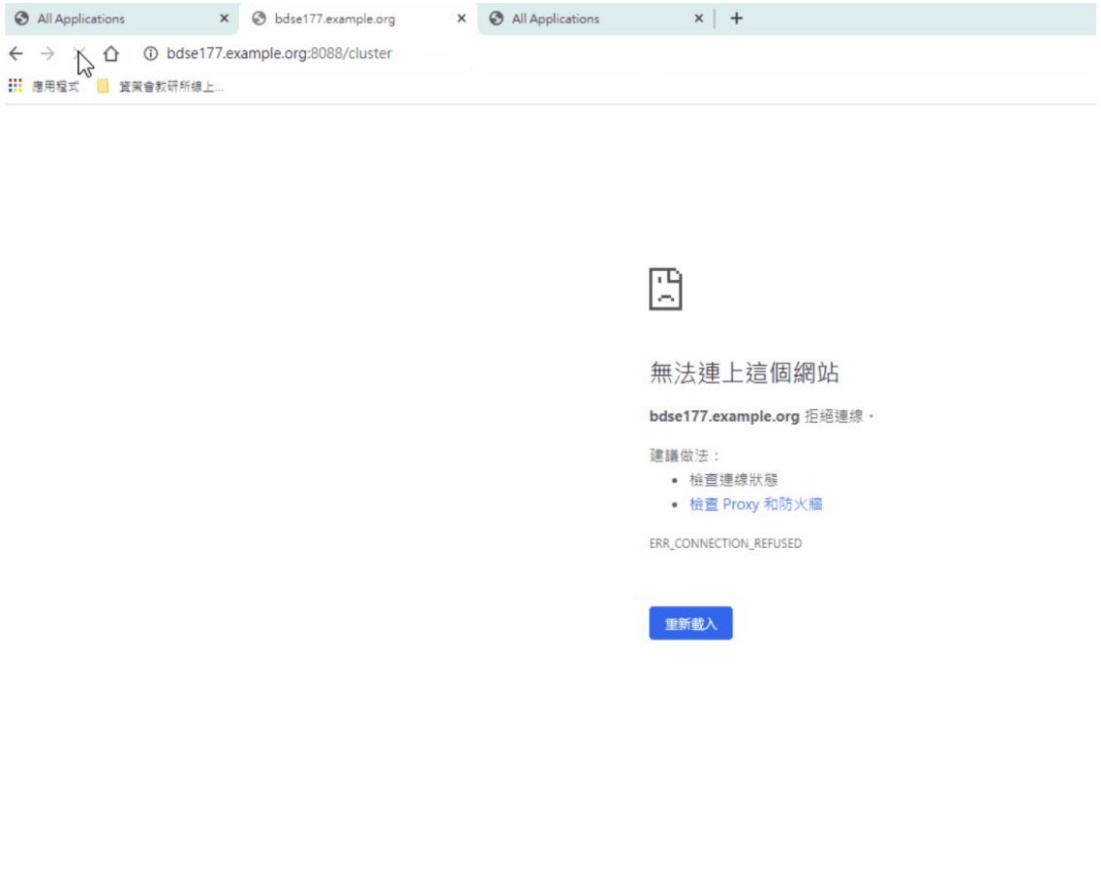
自動跳轉到standby 的RM(bdse153)

網頁bdse177將無法連線

網頁bdse153畫面將會出現

bdse177: yarn --daemon start resourcemanager

bdse177則變為standby



有兩台Resource Manager, 分別為active(bdse177), standby(bdse153)

開啟網頁bdse177(8088), bdse153(8088), 其中bdse153(8088)會自動redirect到bdse177(8088)

bdse177: yarn --daemon stop resourcemanager

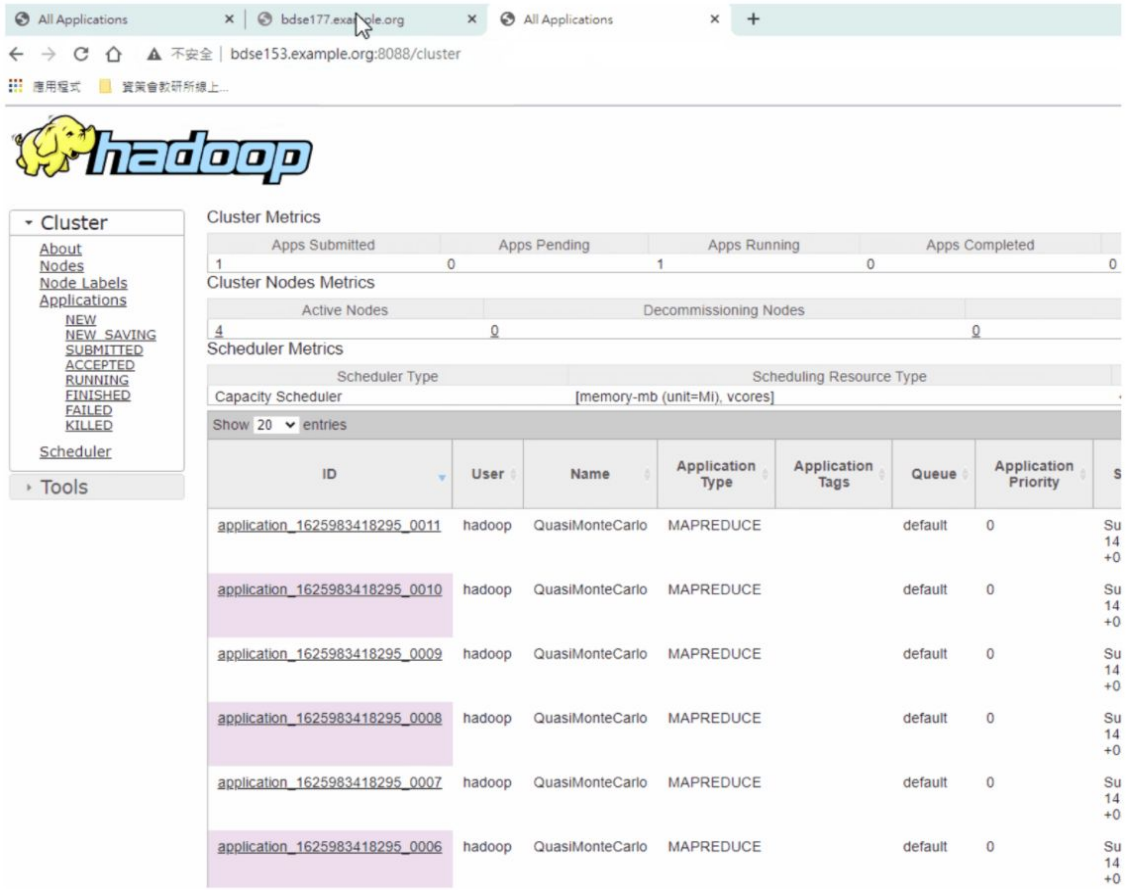
自動跳轉到standby 的RM(bdse153)

網頁bdse177將無法連線

網頁bdse153畫面將會出現

bdse177: yarn --daemon start resourcemanager

bdse177則變為standby



有兩台Resource Manager, 分別為active(bdse177), standby(bdse153)

開啟網頁bdse177(8088), bdse153(8088), 其中bdse153(8088)會自動redirect到bdse177(8088)

bdse177: yarn --daemon stop resourcemanager

自動跳轉到standby 的RM(bdse153)

網頁bdse177將無法連線

網頁bdse153畫面將會出現

bdse177: yarn --daemon start resourcemanager

bdse177則變為standby

Hadoop叢集架構(HA)

```
hadoop@bdse98:~$ yarn rmadmin -getServiceState rm1
standby
hadoop@bdse98:~$ yarn rmadmin -getServiceState rm2
active
hadoop@bdse98:~$ yarn rmadmin -getServiceState rm1
active
hadoop@bdse98:~$ yarn rmadmin -getServiceState rm2
2021-07-11 14:23:09,760 INFO ipc.Client: Retrying connect to server: bdse177.example.org/10.120.31
.177:8033. Already tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries
=1, sleepTime=1000 MILLISECONDS)
Operation failed: Call From bdse98.example.org/10.120.31.98 to bdse177.example.org:8033 failed on
connection exception: java.net.ConnectException: Connection refused; For more details see:  http:/
/wiki.apache.org/hadoop/ConnectionRefused
```

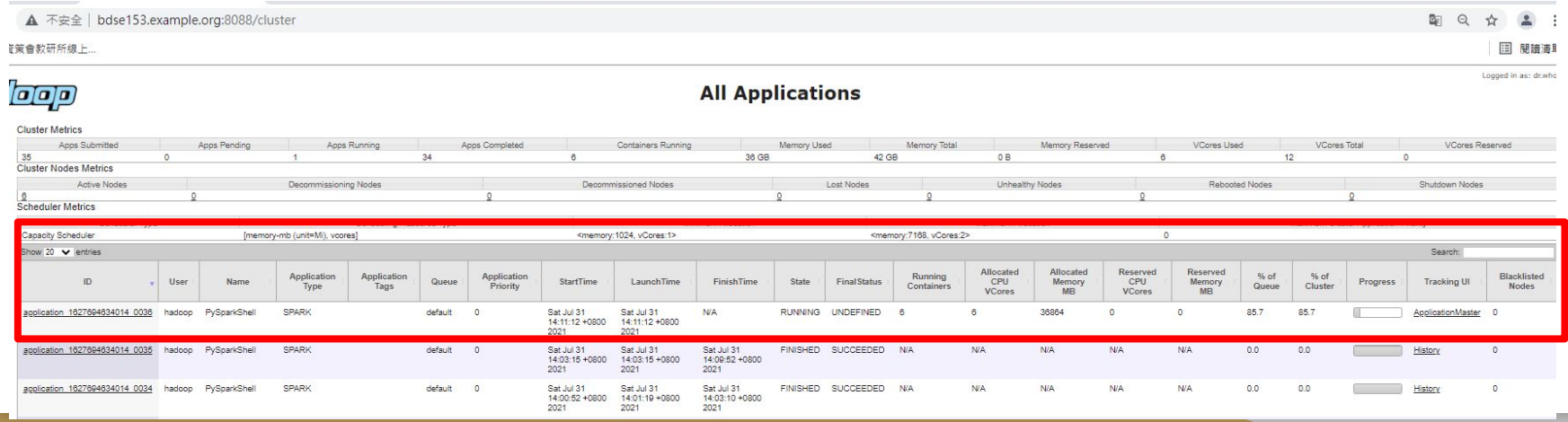
```
hadoop@bdse98:~$ yarn rmadmin -getServiceState rm1
active
hadoop@bdse98:~$ yarn rmadmin -getServiceState rm2
standby
hadoop@bdse98:~$
```

錯誤發生

```
hadoop@bdse154:/usr/local/hive/data$ jps
1872 NodeManager
16613 Jps
16060 RunJar
1502 DataNode
hadoop@bdse154:/usr/local/hive/data$ jobs -l
[1]+ 16060 Running                 hiveserver2 &
hadoop@bdse154:/usr/local/hive/data$ kill 16060
hadoop@bdse154:/usr/local/hive/data$ jobs -l
[1]+ 16060 Exit 143                hiveserver2
hadoop@bdse154:/usr/local/hive/data$ jps
1872 NodeManager
16644 Jps
1502 DataNode
hadoop@bdse154:/usr/local/hive/data$ stop-yarn.sh
Stopping nodemanagers
Stopping resourcemanager
hadoop@bdse154:/usr/local/hive/data$ jps
17029 Jps
1502 DataNode
hadoop@bdse154:/usr/local/hive/data$
```

```
hadoop@bdse154: ~/data
<1> hadoop@bdse154:...
total 2201984
drwxrwxr-x 5 hadoop hadoop       4096 Jun  8 13:50 data
drwxrwxr-x 3 hadoop hadoop       4096 Jul 10 11:58 journalnode
-rw-rw-r-- 1 hadoop hadoop 2254818039 Jun  9 09:27 testdata.csv
hadoop@bdse154:~$ cd data
hadoop@bdse154:~/data$ hdfs haadmin -getServiceState nn1
active
hadoop@bdse154:~/data$ hdfs haadmin -getServiceState nn2
standby
hadoop@bdse154:~/data$ jps
17904 DataNode
18058 JournalNode
20557 Jps
hadoop@bdse154:~/data$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@bdse154:~/data$ jps
17904 DataNode
20904 NodeManager
18058 JournalNode
21070 Jps
hadoop@bdse154:~/data$ hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoo
p-mapreduce-examples-3.3.0.jar pi 3 1000
```

```
[9]: %%timeit
     # Read CVS file
     df_train = ks.read_csv('/user/spark/share/performance_detail.csv')

     14 s ± 246 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

[10]: # Read CVS file
      df_train = ks.read_csv('/user/spark/share/performance_detail.csv')

[12]: %%timeit
      # Read CVS file
      sdf_train = spark.read.csv('/user/spark/share/performance_detail.csv',inferSchema=True, header=True)

      13.9 s ± 202 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

[13]: %%timeit
      df_train.shape

      2.53 s ± 93.9 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```



```
from pyspark.ml import Pipeline
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.linalg import Vector
from pyspark.ml.evaluation import RegressionEvaluator
import time
```

```
%%timeit
df = spark.read.csv('/user/spark/share/testData.csv', inferSchema=True, header=True)
```

```
497 ms ± 86.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
df = spark.read.csv('/user/spark/share/testData.csv', inferSchema=True, header=True)
```

```
feature_cols = df.columns[:-1]
assembler = VectorAssembler(inputCols=feature_cols, outputCol='features')
features_df = assembler.transform(df)
```

```
model_df = features_df.select('features', 'houseprice')
```

```
trainingData, testData = model_df.randomSplit([0.7, 0.3],seed=10)
```

```
%%time
rf = RandomForestRegressor(featuresCol='features',labelCol='houseprice', numTrees=200)
rf_model = rf.fit(trainingData)
train_pred = rf_model.transform(trainingData)
test_pred = rf_model.transform(testData)
regressionEvaluator = RegressionEvaluator(
  predictionCol="prediction",
  labelCol="houseprice",
  metricName="mse")
r2_train = regressionEvaluator.setMetricName("r2").evaluate(train_pred)
r2_test = regressionEvaluator.setMetricName("r2").evaluate(test_pred)
print(f"R2 train: {r2_train}")
print(f"R2 test: {r2_test}")
```

```
R2 train: 0.8047031370388252
R2 test: 0.5923063564072693
CPU times: user 24.3 ms, sys: 1.72 ms, total: 26 ms
Wall time: 9.5 s
```