ChordFileSystem Report

System components:

1. Uploadserver

Run on port 5058

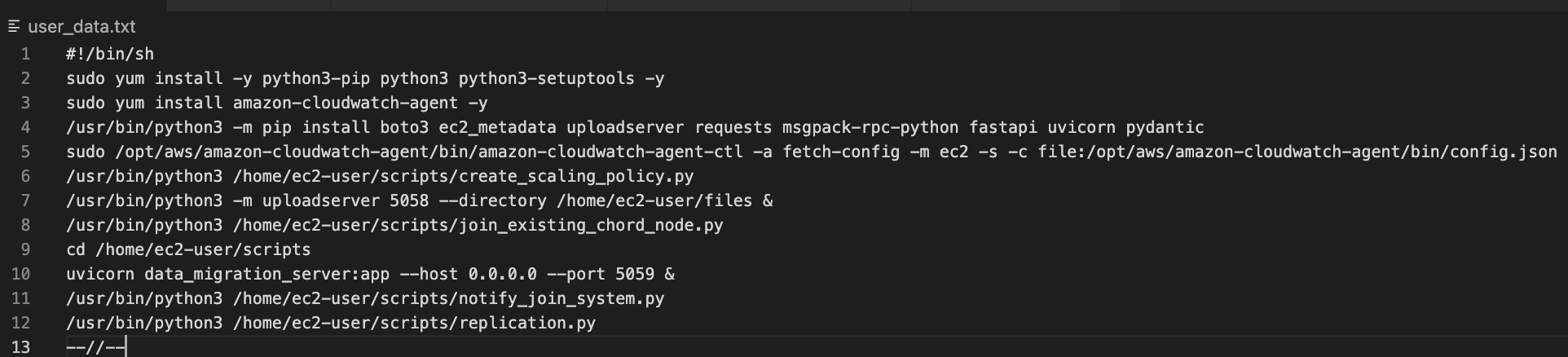
1. Chord

Run on port 5057

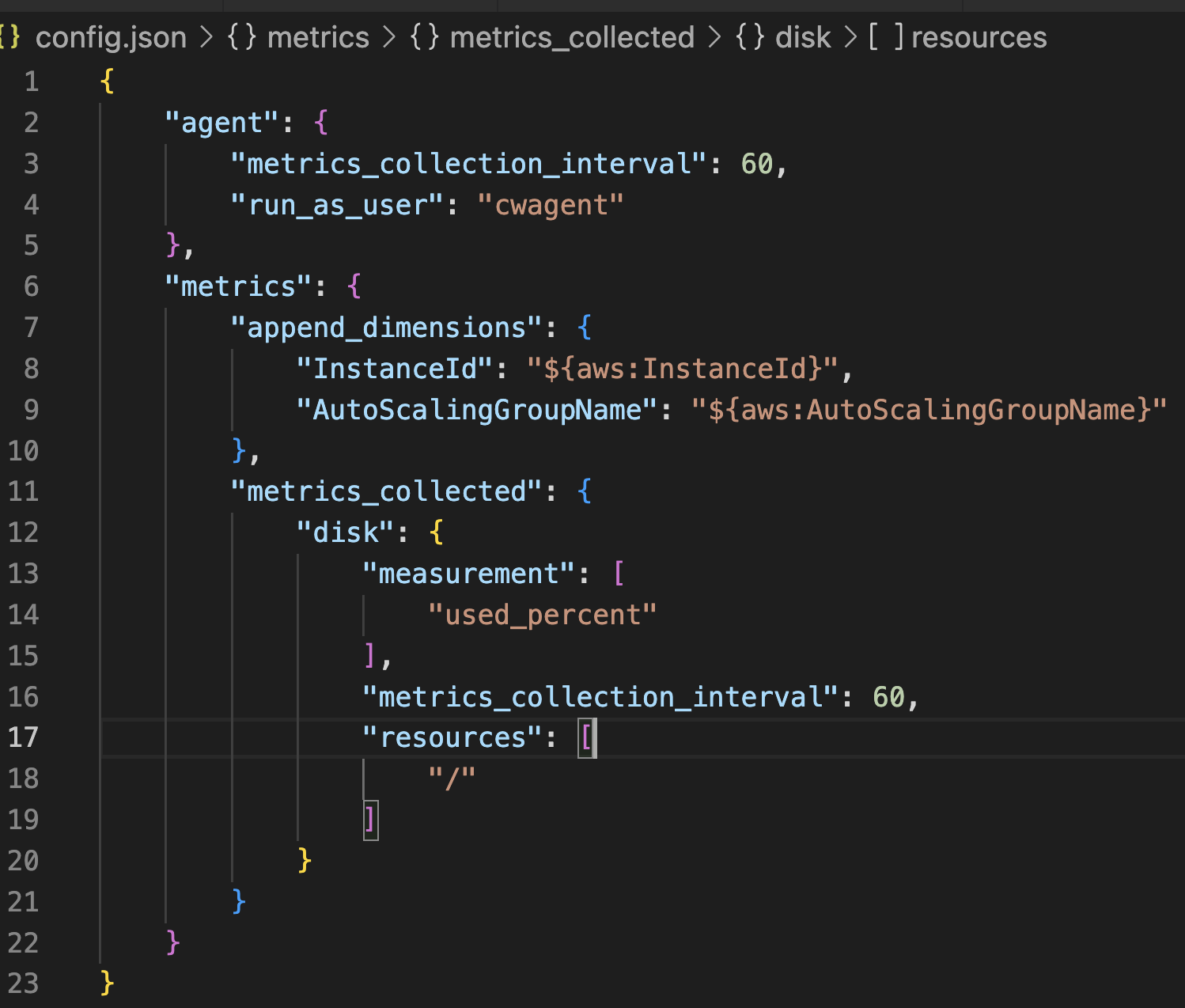
1. Data migration server

Run on port 5059

1. User data (Something happen after my instance start)

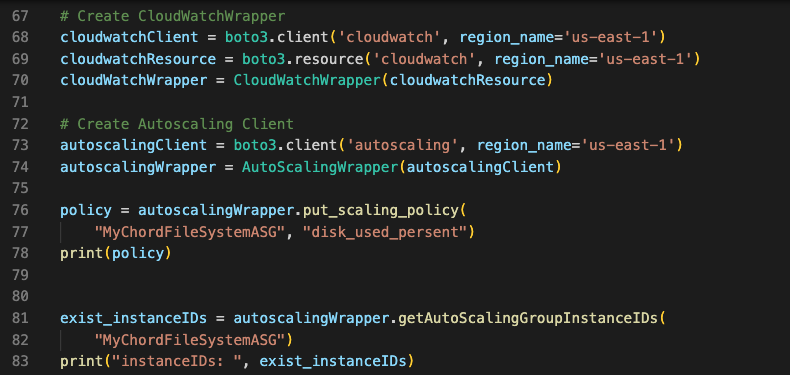


* 1. First install python and amazon-cloudwatch -agent
  2. Use pip install some python package needed following steps
  3. Start the CloudWatch agent and create the metric by my config file (/opt/aws/amazon-cloudwatch-agent/bin/config.json)



這個metric 之後可用instance\_id、所屬ASG來找到、且會監聽 / 目錄下的disk用量百分比

* 1. 接著會執行create\_scaling\_policy.py



首先找到ＭyChordSystemASG這個ASG並用put\_scaling\_policy建立policy。（之後會用到該policy的ARN來attach alarm）

接著，每個在ASG中的instamce都有一個前一步驟建立的metric來監聽，所以我們需要拿到ASG中所有的instance\_id來取得這些metric。





接著我們用剛剛得到的instance\_id找到對應的metric放入metric\_query中，也定義我們的expression\_querys來算所有metric的回傳值平均。並把他們合併變成最終的metrics參數（put\_metric\_alarm會用到）。



因為前一步驟建立cloudwatch metric會需要一些時間，我這裡跑while迴圈一直去用instance\_id來list metrics看看該instance是否成功建立的metric。

直到我成功建立metric後，我會用上個步驟的metrics參數和一開始拿到的policy ARN來建立alarm。

* 1. 接著執行/usr/bin/python3 -m uploadserver 5058 --directory /home/ec2-user/files &

把upload server跑起來在5058 port 且之後上傳的檔案都會放到

/home/ec2-user/files這個目錄

* 1. 接著執行join\_existing\_chord\_node.py



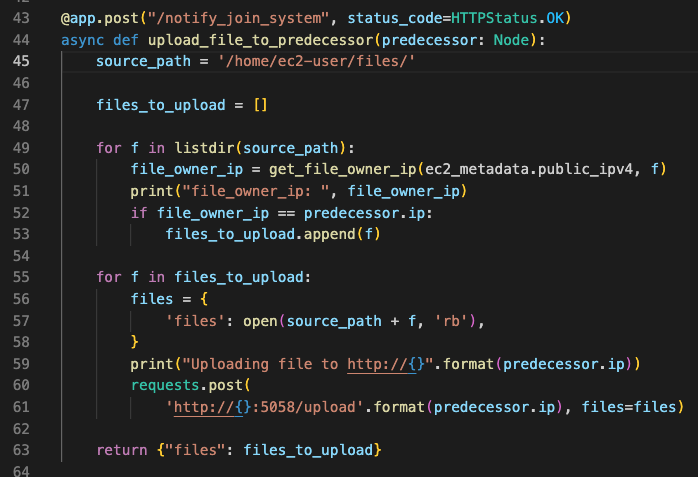
首先取得所有在MyChordFileSystemASG裡的instanceID，並把助教提供的chord執行檔跑起來在5057 並使用自己的public\_ip

因為chord跑起來要時間所以等5秒

接著若自己是ASG中第一個instance，則用自己的public\_ip create chord system

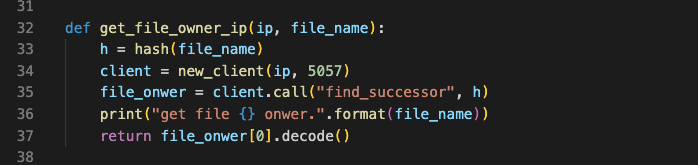
若已經有其他instance在ASG中，則用他的public\_ip將自己的public\_ip join進chord system。

* 1. 接著將data migration server 跑起來



我用fastapi簡單建一個server，裡面有一隻api叫做/notify\_join\_system，當有新instance join ASG時他會對他的successor打這個API並將自己的Node資訊(ip以及chord id)傳給successor。

這裡收到訊息後用會掃描所有存放在該instance中的檔案，把新instance應該負責的檔案upload 給新的instance。

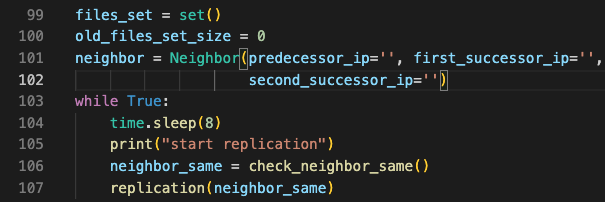


* 1. 接著執行notify\_join\_system



這就是上一步驟說當新instance加入ASG時會執行的程式，call successor 的/nptify\_join\_system api，和他要應該存在自己這裡的file。因為新instance join chord system需要時間stabalize所以等20秒再要資料。

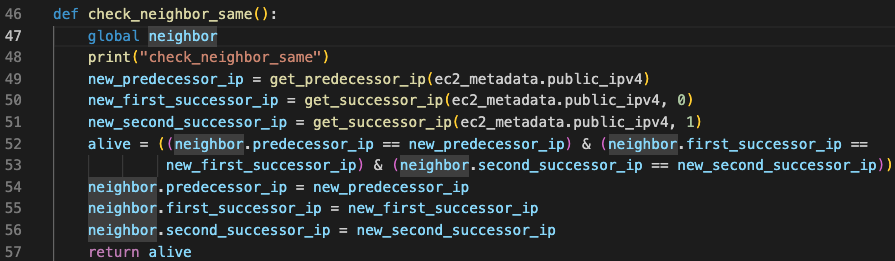
* 1. 接著執行replication.py



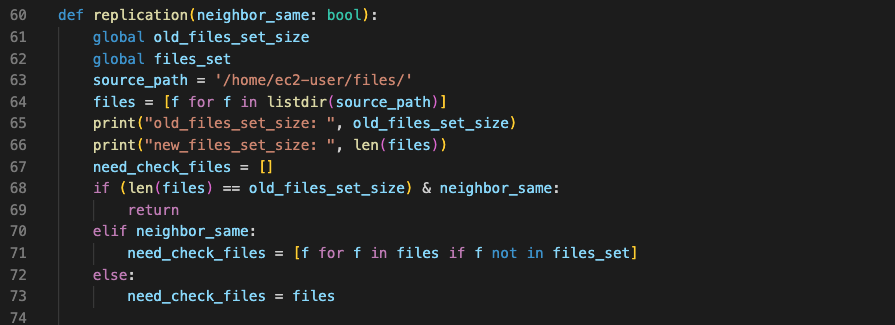
首先會用files\_set來記錄已經被check過的file，用old\_files\_set\_size來紀錄已經被check過的file的數量。

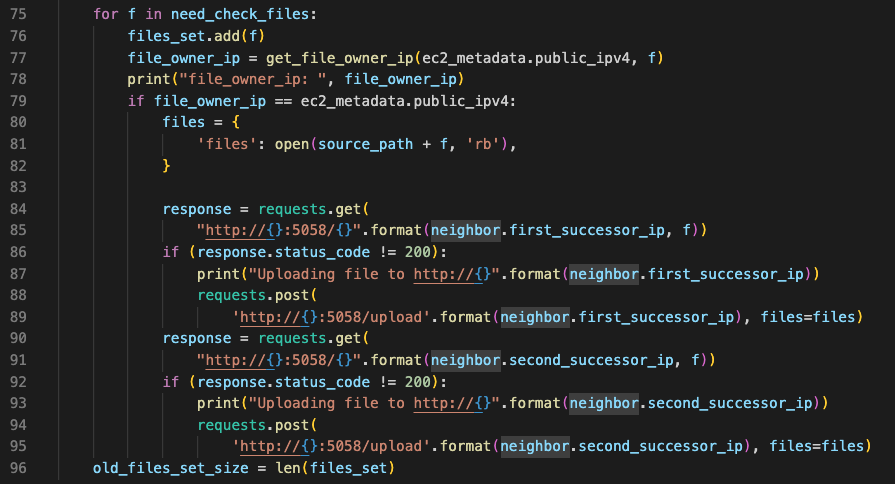
另外還需要紀錄Neighbor的ip資訊(predecessor、first\_successor、second\_successor)

每隔8秒會check一次neighbor是否有變動，並且檢查是否要replication。



Check\_neighbor\_same會去用rpc call來檢查自己的neighbor是否有變，並更新neighbor資訊。





Replication中首先判斷instance中是否有新的file，若沒有新file且neighbor資訊沒變，就什麼都不用做。

若neighbor沒變，但有新file則只需檢查新的files

若neighbor資訊變了則全部file都需要檢查

接著將需要檢查的files都加進files-set中，變看看這個file是不是自己要負責replication，如果是，則檢查後兩個successor有沒有這個file資料，如果沒有救upload給他們。

最後更新old\_files-set\_size資訊

System functionalities:

1. Data Migration：

詳細步驟請看System components的4.7, 4.8。

1. File Chunks:

這裡主要在client這裡處理。分upload.py和download.py討論。

* 1. upload.py



首先定義chunk\_size為4KB。

接著每次會讀chunk\_size的檔案並將檔名命名為filename\_chunk\_i (filename為原始檔名，i為index)

把他hash後用find\_successor找到要upload的ip做upload。

直到讀不到資料後結束迴圈。

* 1. download.py



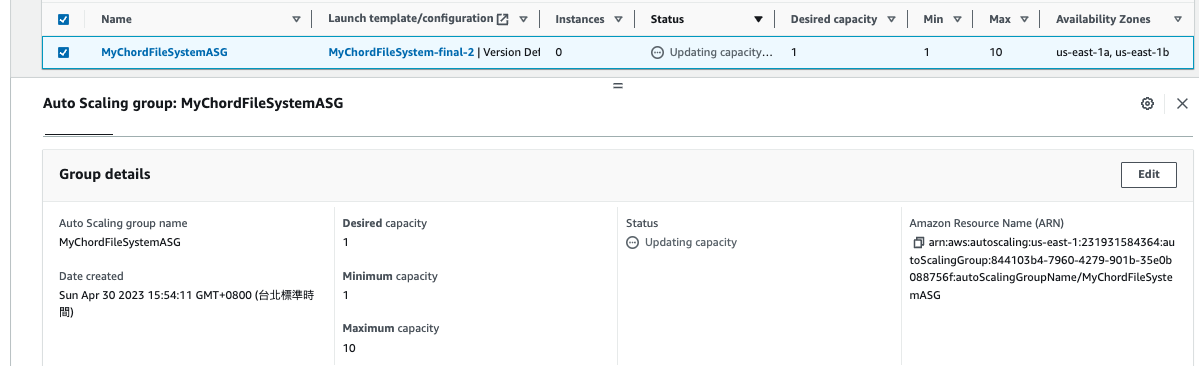
Download時則按照chunk id一個一個下載直到下載時找不到檔案每次載完會把下載到的結果寫入下來。

1. Replication

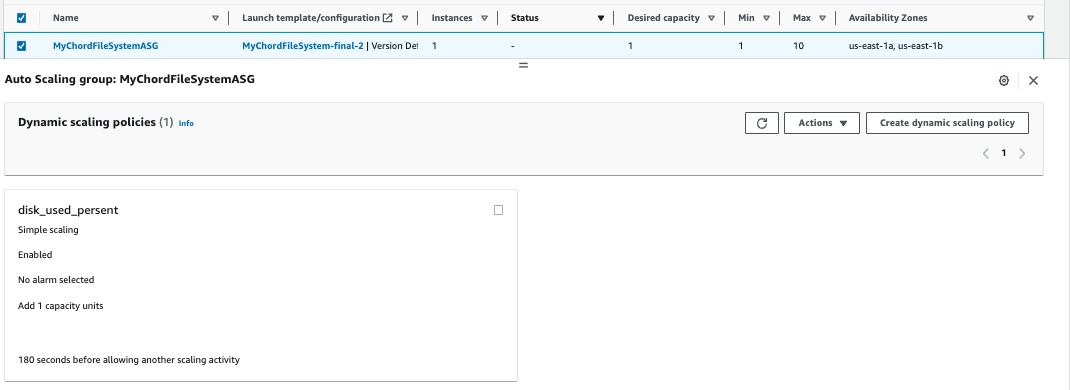
詳細步驟請看System components的4.9。

Experiment:

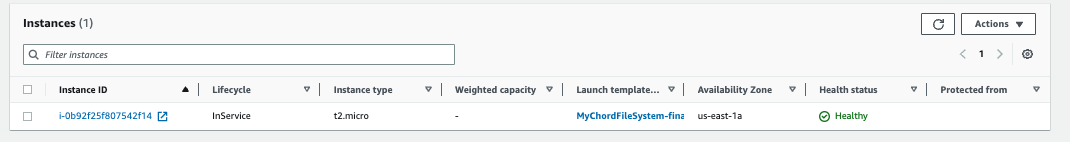
首先先建立一個ASG把Desired capacity設為1。

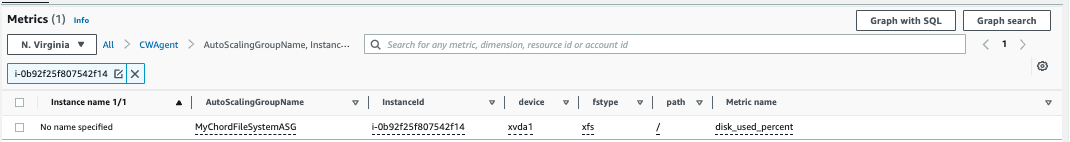


過陣子後發現scaling policies有新的policy出現

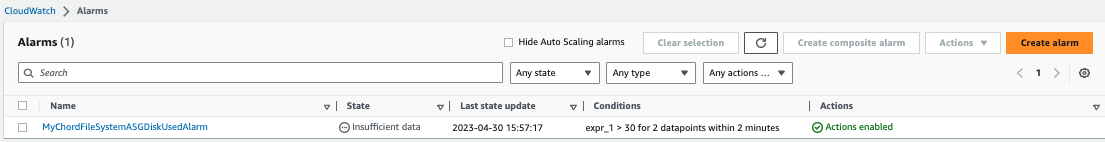


此時去cloudwatch找對應instance的metric

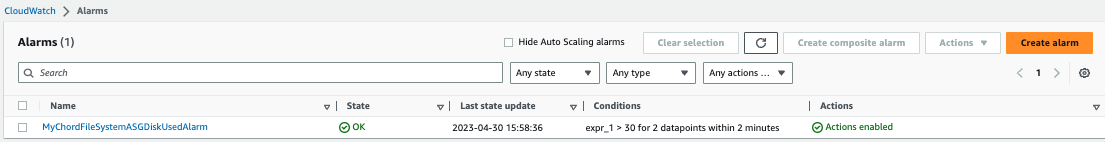




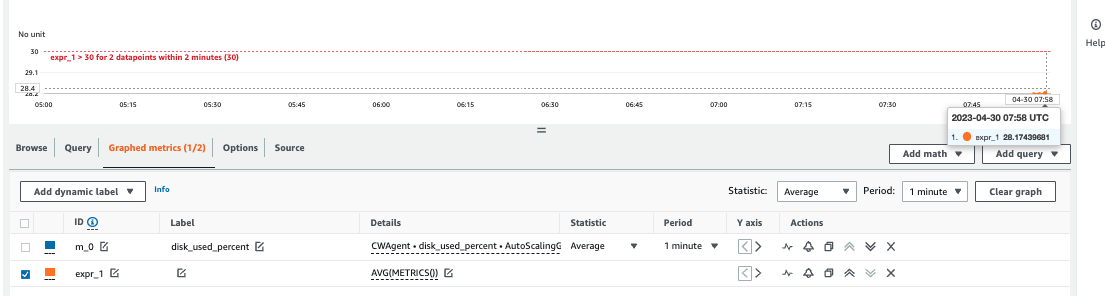
發現alarm也出現了



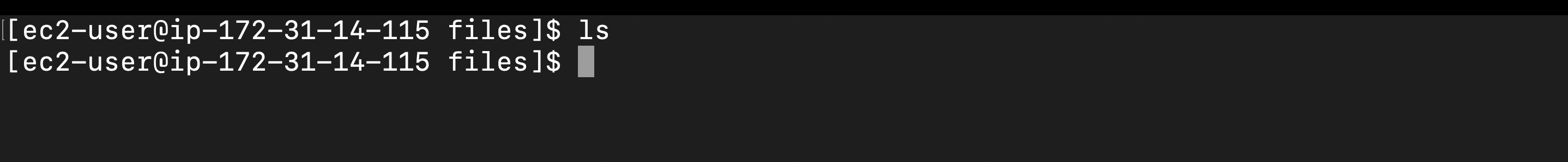
過陣子狀態變ok



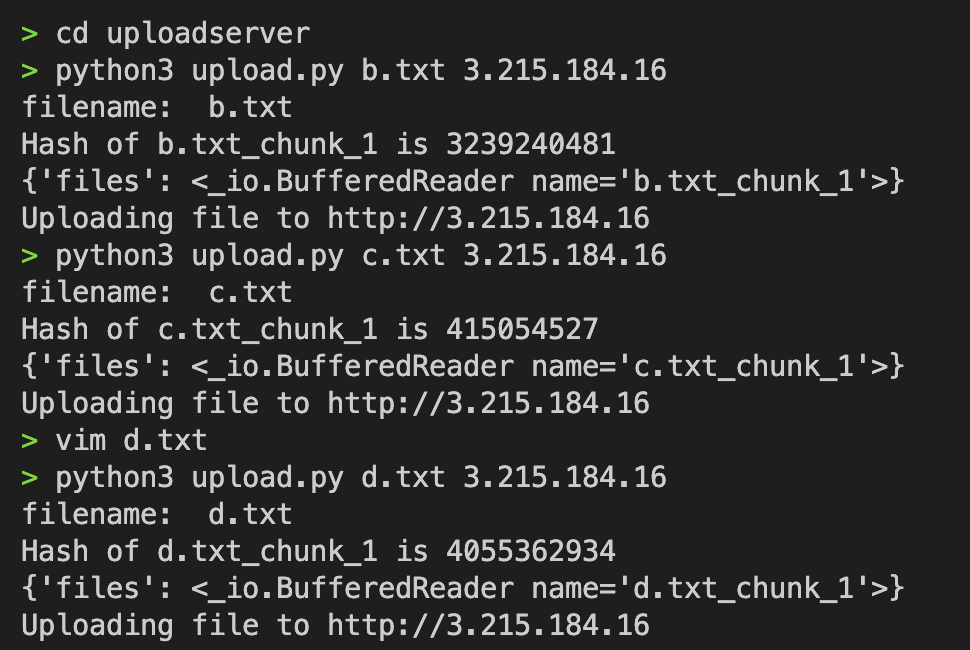
可以看到metric中現在有一個metric和一個expression



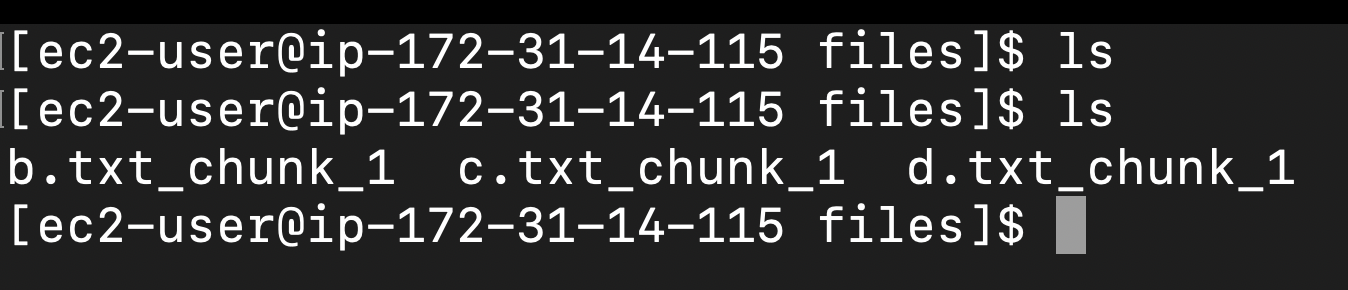
連線進ec2看看發現files目錄下沒東西



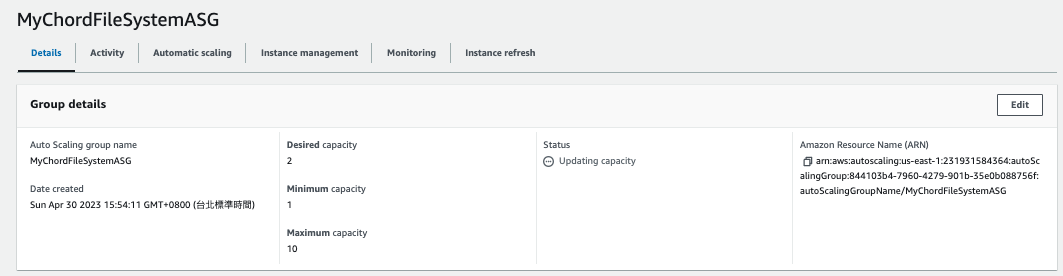
從client 用upload.py上傳幾個檔案試試

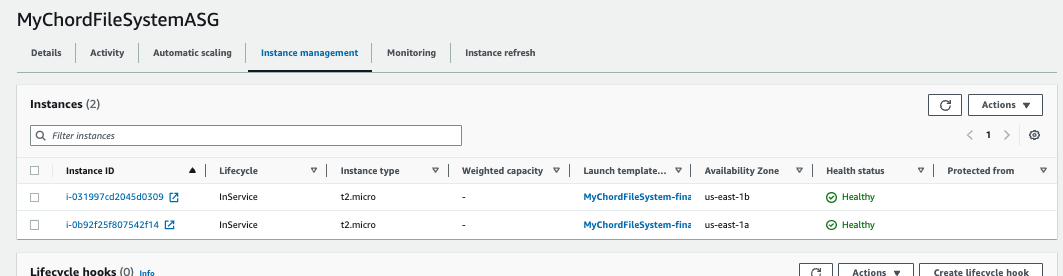


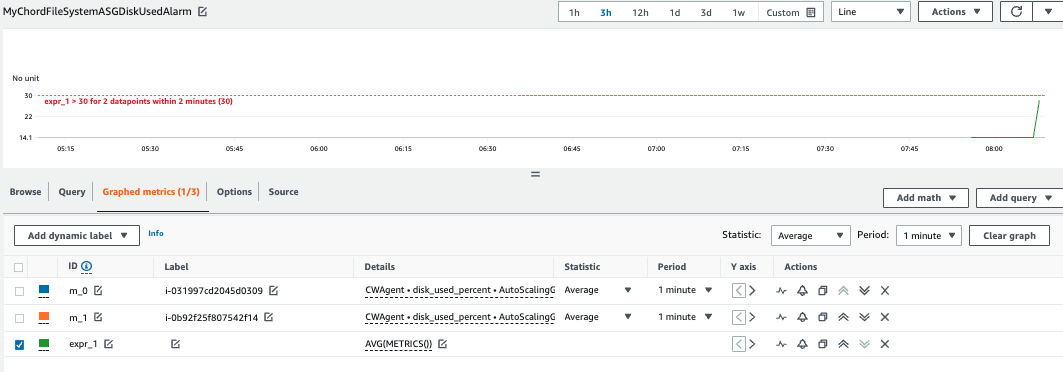
結果如下：



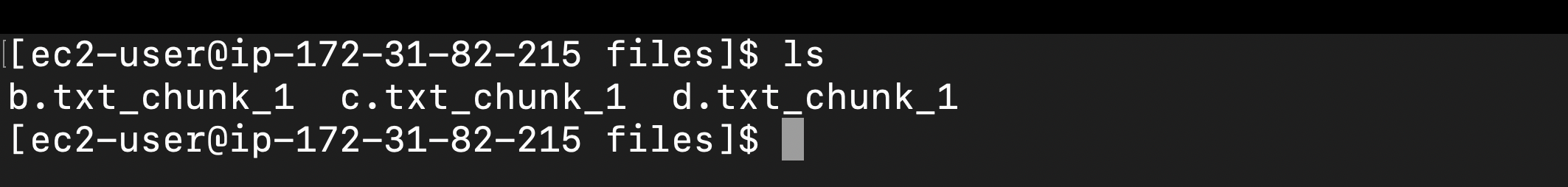
此時把ASG Desired capacity設為2



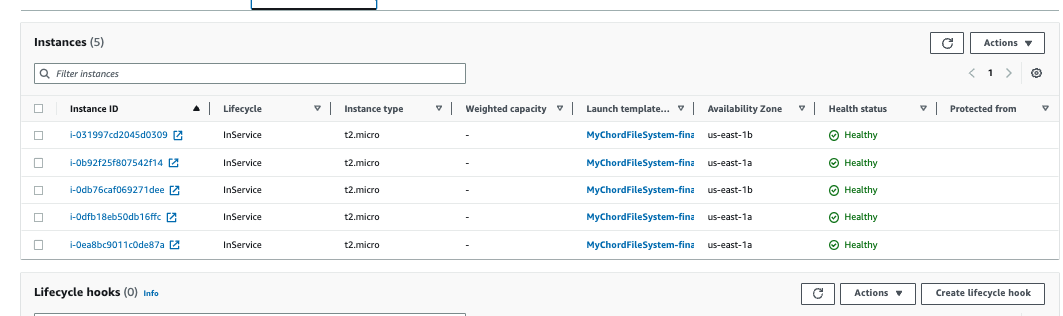
去cloudwatch看alarm的詳細資訊，發現有兩個metric和一個expression



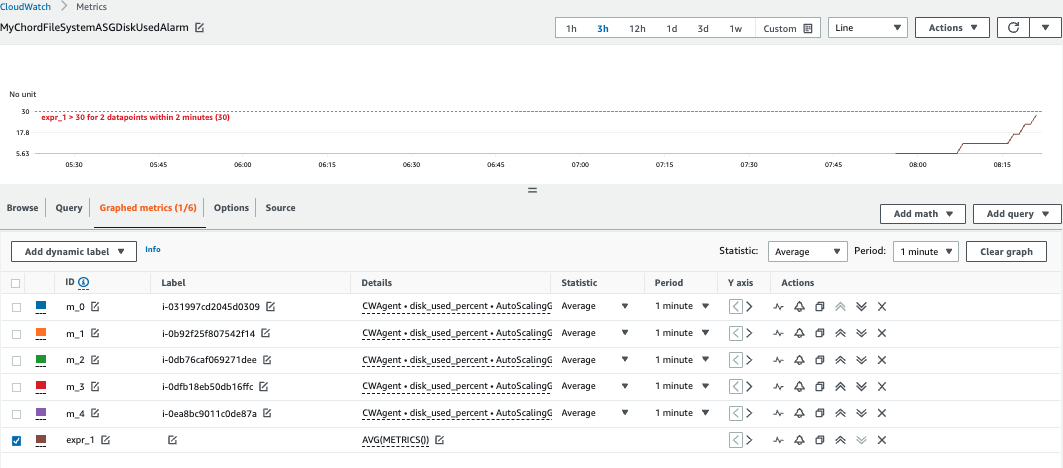
連進第二個instance看發現flie有被migration和replica進來



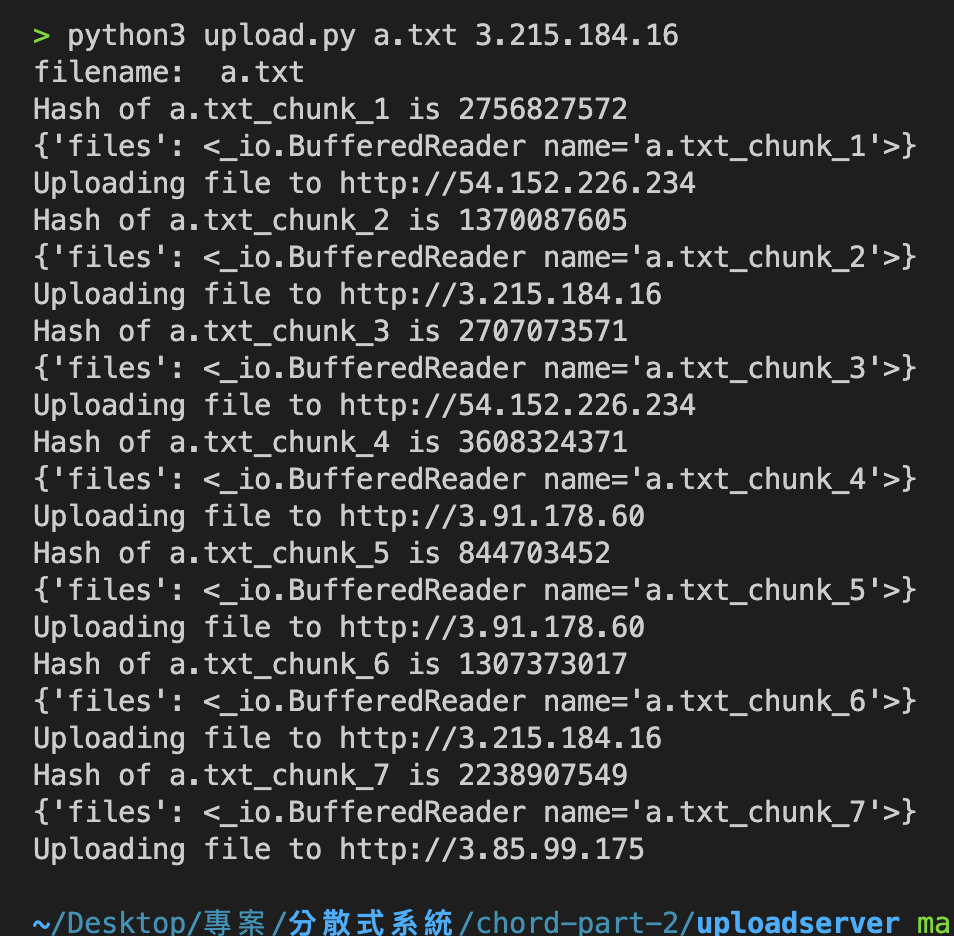
接著逐步把instance擴充到5個



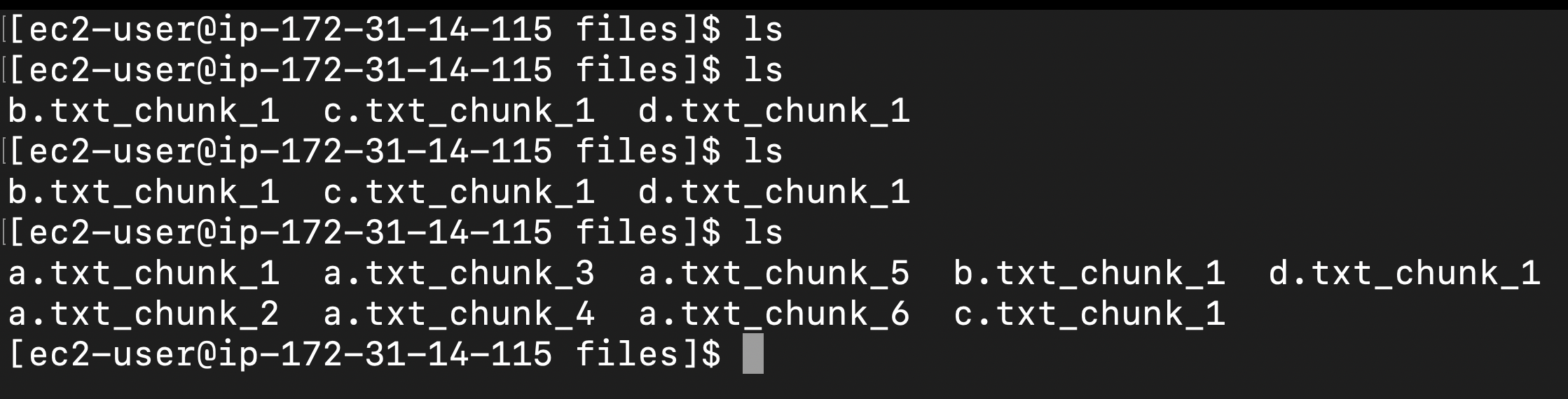
去cloudwatch看alarm結果

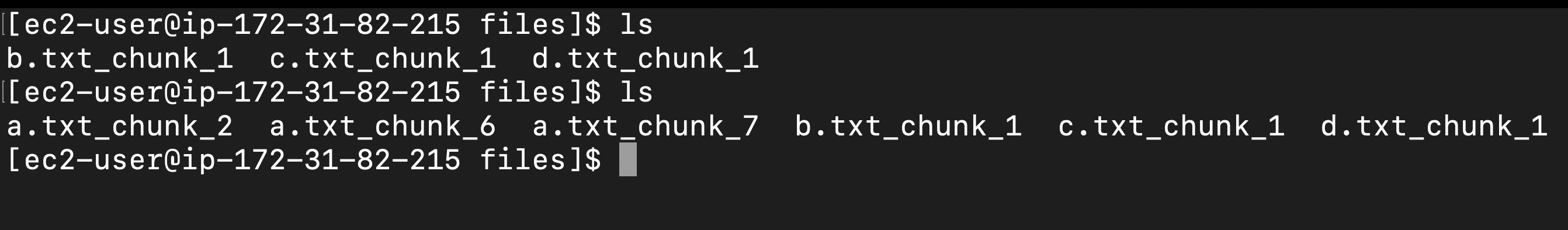


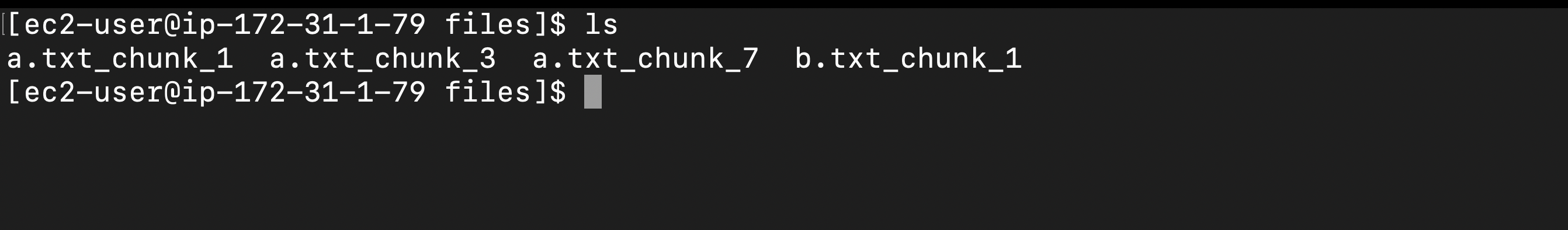
把大檔案a.txt從client端上傳發現有被切成7個chunk上傳到不同地方

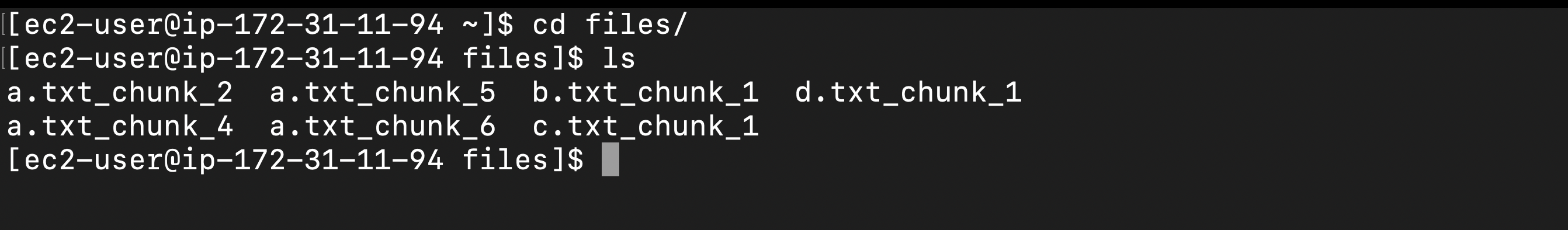


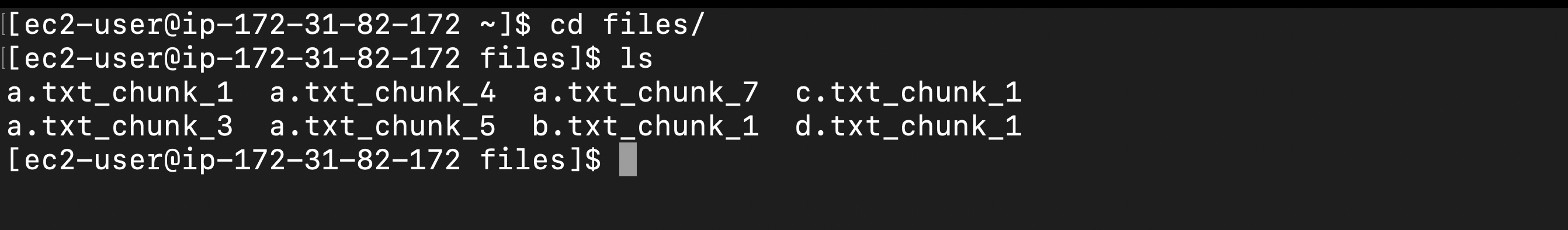
Ssh 進每個instance看











發現所有file都有三個replica

References:

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/autoscaling.html>

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/cloudwatch.html>

<https://pypi.org/project/ec2-metadata/>

<https://github.com/awsdocs/aws-doc-sdk-examples/blob/main/python/example_code/cloudwatch/cloudwatch_basics.py>

<https://docs.aws.amazon.com/zh_tw/code-library/latest/ug/python_3_auto-scaling_code_examples.html>

<https://docs.aws.amazon.com/zh_tw/code-library/latest/ug/python_3_cloudwatch_code_examples.html>

<https://docs.aws.amazon.com/zh_tw/autoscaling/application/userguide/create-step-scaling-policy-cli.html>