

ChordFileSystem Report

System components:

1 Uploadserver

Run on port 5058

2 Chord

Run on port 5057

3 Data migration server

Run on port 5059

4 User data (Something happen after my instance start)

```
user_data.txt
1  #!/bin/sh
2  sudo yum install -y python3-pip python3 python3-setuptools -y
3  sudo yum install amazon-cloudwatch-agent -y
4  /usr/bin/python3 -m pip install boto3 ec2_metadata uploadserver requests msgpack-rpc-python fastapi uvicorn pydantic
5  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json
6  /usr/bin/python3 /home/ec2-user/scripts/create_scaling_policy.py
7  /usr/bin/python3 -m uploadserver 5058 --directory /home/ec2-user/files &
8  /usr/bin/python3 /home/ec2-user/scripts/join_existing_chord_node.py
9  cd /home/ec2-user/scripts
10 uvicorn data_migration_server:app --host 0.0.0.0 --port 5059 &
11 /usr/bin/python3 /home/ec2-user/scripts/notify_join_system.py
12 /usr/bin/python3 /home/ec2-user/scripts/replication.py
13 --//--
```

4.1 First install python and amazon-cloudwatch -agent

4.2 Use pip install some python package needed following steps

4.3 Start the CloudWatch agent and create the metric by my config

file (/opt/aws/amazon-cloudwatch-agent/bin/config.json)

```

} config.json > {} metrics > {} metrics_collected > {} disk > [ ] resources
1  {
2      "agent": {
3          "metrics_collection_interval": 60,
4          "run_as_user": "cwagent"
5      },
6      "metrics": {
7          "append_dimensions": {
8              "InstanceId": "${aws:InstanceId}",
9              "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
10         },
11         "metrics_collected": {
12             "disk": {
13                 "measurement": [
14                     "used_percent"
15                 ],
16                 "metrics_collection_interval": 60,
17                 "resources": [
18                     "/"
19                 ]
20             }
21         }
22     }
23 }

```

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

4.4 接著會執行 create_scaling_policy.py

```

67 # Create CloudWatchWrapper
68 cloudwatchClient = boto3.client('cloudwatch', region_name='us-east-1')
69 cloudwatchResource = boto3.resource('cloudwatch', region_name='us-east-1')
70 cloudWatchWrapper = CloudWatchWrapper(cloudwatchResource)
71
72 # Create Autoscaling Client
73 autoscalingClient = boto3.client('autoscaling', region_name='us-east-1')
74 autoscalingWrapper = AutoScalingWrapper(autoscalingClient)
75
76 policy = autoscalingWrapper.put_scaling_policy(
77     "MyChordFileSystemASG", "disk_used_percent")
78 print(policy)
79
80
81 exist_instanceIDs = autoscalingWrapper.getAutoScalingGroupInstanceIDs(
82     "MyChordFileSystemASG")
83 print("instanceIDs: ", exist_instanceIDs)

```

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 instance 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。

```
85
86 metric_queries = []
87 for i in range(len(exist_instanceIDs)):
88     metric_queries.append(
89         {
90             "Id": f'm_{i}',
91             "MetricStat": {
92                 "Metric": {
93                     "Namespace": "CWAgent",
94                     "MetricName": "disk_used_percent",
95                     "Dimensions": [
96                         {
97                             'Name': 'AutoScalingGroupName',
98                             'Value': 'MyChordFileSystemASG'
99                         },
100                         {
101                             "Name": "InstanceId",
102                             "Value": exist_instanceIDs[i]
103                         },
104                         {
105                             'Name': 'device',
106                             'Value': 'xvda1'
107                         },
108                         {
109                             'Name': 'fstype',
110                             'Value': 'xfs'
111                         },
112                         {
113                             'Name': 'path',
114                             'Value': '/'
115                         },
116                     ],
117                 },
118                 "Period": 60,
119                 "Stat": "Average",
120                 "Unit": "Percent"
121             },
122             "ReturnData": False
123         }
124     )
125 expression_queries = [{
126     'Id': 'expr_1',
127     'Expression': 'AVG(METRICS())',
128     'ReturnData': True
129 }]
130
131
132 metrics = metric_queries + expression_queries
133 print("metrics: ", metrics)
```

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_queries 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)。

```
134 tryCnt = 0
135 while (True):
136     list_metrics_result = list(cloudWatchWrapper.list_metrics(
137         "CWAgent", "disk_used_percent", ec2_metadata.instance_id))
138     if len(list_metrics_result) == 0:
139         tryCnt += 1
140         print("tryCnt: ", tryCnt)
141         time.sleep(20)
142     else:
143         print("list_metrics_result len: ", len(list_metrics_result))
144         disk_used_percent_metric = list_metrics_result[0]
145         print("disk_used_percent_metric: ", disk_used_percent_metric)
146         alarm = cloudwatchClient.put_metric_alarm(AlarmName='MyChordFileSystemASGDiskUsedAlarm',
147             AlarmActions=[
148                 policy['PolicyARN']],
149             Metrics=metrics,
150             EvaluationPeriods=2,
151             Threshold=30,
152             ComparisonOperator='GreaterThanThreshold')
153         print("alarm: ", alarm)
154         break
155
```

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。

4.5 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

4.6 接著執行 join_existing_chord_node.py

```
57 autoscalingClient = boto3.client('autoscaling', region_name='us-east-1')
58 autoScalingWrapper = AutoScalingWrapper(autoscalingClient)
59 exist_instanceIDs = autoScalingWrapper.getAutoScalingGroupInstanceIDs(
60     "MyChordFileSystemASG")
61 print("instanceIDs: ", exist_instanceIDs)
62
63 # Start Chord node
64 os.system("/home/ec2-user/scripts/chord {} 5057 {}".format(ec2_metadata.public_ipv4))
65 print("Chord Node start listen in {}:5057".format(ec2_metadata.public_ipv4))
66 time.sleep(5)
67 # Join existing Chord system
68 if len(exist_instanceIDs) == 1:
69     create(ec2_metadata.public_ipv4)
70 else:
71     if getPublicIP(exist_instanceIDs[0]) == ec2_metadata.public_ipv4:
72         join(ec2_metadata.public_ipv4, getPublicIP(exist_instanceIDs[1]))
73     else:
74         join(ec2_metadata.public_ipv4, getPublicIP(exist_instanceIDs[0]))
75
```

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。

4.7 接著將 data migration server 跑起來

```

43 @app.post("/notify_join_system", status_code=HTTPStatus.OK)
44 async def upload_file_to_predecessor(predecessor: Node):
45     source_path = '/home/ec2-user/files/'
46
47     files_to_upload = []
48
49     for f in listdir(source_path):
50         file_owner_ip = get_file_owner_ip(ec2_metadata.public_ipv4, f)
51         print("file_owner_ip: ", file_owner_ip)
52         if file_owner_ip == predecessor.ip:
53             files_to_upload.append(f)
54
55     for f in files_to_upload:
56         files = {
57             'files': open(source_path + f, 'rb'),
58         }
59         print("Uploading file to http://{0}".format(predecessor.ip))
60         requests.post(
61             'http://{0}:5058/upload'.format(predecessor.ip), files=files)
62
63     return {"files": files_to_upload}
64

```

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。

```

31
32 def get_file_owner_ip(ip, file_name):
33     h = hash(file_name)
34     client = new_client(ip, 5057)
35     file_onwer = client.call("find_successor", h)
36     print("get file {0} onwer.".format(file_name))
37     return file_onwer[0].decode()
38

```

4.8 接著執行 notify_join_system

```

23     time.sleep(20)
24
25     mynode = get_info(ec2_metadata.public_ipv4)
26     successor = get_successor(ec2_metadata.public_ipv4, 0)
27     successor_ip = successor[0].decode()
28     print("successor: ", successor)
29     node = {
30         'ip': mynode[0].decode(),
31         'port': mynode[1],
32         'id': mynode[2]
33     }
34     print('http://{0}:5059/notify_join_system'.format(successor_ip))
35     print("node: ", node)
36     response = requests.post(
37         'http://{0}:5059/notify_join_system'.format(successor[0].decode()), json=node)
38
39     print(response.content)
40

```

這就是上一步驟說當新 instance 加入 ASG 時會執行的程式，call successor 的/notify_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 stabilize 所以等 20 秒再要資料。

4.9 接著執行 replication.py

```

99     files_set = set()
100     old_files_set_size = 0
101     neighbor = Neighbor([predecessor_ip='', first_successor_ip='',
102                          second_successor_ip=''])
103     while True:
104         time.sleep(8)
105         print("start replication")
106         neighbor_same = check_neighbor_same()
107         replication(neighbor_same)

```

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。

```

46 def check_neighbor_same():
47     global neighbor
48     print("check_neighbor_same")
49     new_predecessor_ip = get_predecessor_ip(ec2_metadata.public_ipv4)
50     new_first_successor_ip = get_successor_ip(ec2_metadata.public_ipv4, 0)
51     new_second_successor_ip = get_successor_ip(ec2_metadata.public_ipv4, 1)
52     alive = ((neighbor.predecessor_ip == new_predecessor_ip) & (neighbor.first_successor_ip ==
53         new_first_successor_ip) & (neighbor.second_successor_ip == new_second_successor_ip))
54     neighbor.predecessor_ip = new_predecessor_ip
55     neighbor.first_successor_ip = new_first_successor_ip
56     neighbor.second_successor_ip = new_second_successor_ip
57     return alive

```

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。

```

60 def replication(neighbor_same: bool):
61     global old_files_set_size
62     global files_set
63     source_path = '/home/ec2-user/files/'
64     files = [f for f in listdir(source_path)]
65     print("old_files_set_size: ", old_files_set_size)
66     print("new_files_set_size: ", len(files))
67     need_check_files = []
68     if (len(files) == old_files_set_size) & neighbor_same:
69         return
70     elif neighbor_same:
71         need_check_files = [f for f in files if f not in files_set]
72     else:
73         need_check_files = files
74

```

```

75     for f in need_check_files:
76         files_set.add(f)
77         file_owner_ip = get_file_owner_ip(ec2_metadata.public_ipv4, f)
78         print("file_owner_ip: ", file_owner_ip)
79         if file_owner_ip == ec2_metadata.public_ipv4:
80             files = {
81                 'files': open(source_path + f, 'rb'),
82             }
83
84             response = requests.get(
85                 "http://{}:5058/{}".format(neighbor.first_successor_ip, f))
86             if (response.status_code != 200):
87                 print("Uploading file to http://{}/".format(neighbor.first_successor_ip))
88                 requests.post(
89                     'http://{}:5058/upload'.format(neighbor.first_successor_ip), files=files)
90             response = requests.get(
91                 "http://{}:5058/{}".format(neighbor.second_successor_ip, f))
92             if (response.status_code != 200):
93                 print("Uploading file to http://{}/".format(neighbor.second_successor_ip))
94                 requests.post(
95                     'http://{}:5058/upload'.format(neighbor.second_successor_ip), files=files)
96         old_files_set_size = len(files_set)

```

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，如果沒有就 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。

2 File Chunks:

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

2.1 upload.py

```

29 chunk_size = 4000 # 4KB
30
31 # for i in range(1, 20):
32 with open(filepath, 'rb') as f:
33     chunk_num = 1
34     while True:
35         # chunk_stream = io.BytesIO()
36         chunk_data = f.read(chunk_size)
37         if not chunk_data:
38             break
39         chunk_file_name = '{}_chunk_{}'.format(filename, chunk_num)
40         h = hash(chunk_file_name)
41         print("Hash of {} is {}".format(chunk_file_name, h))
42         node = client.call("find_successor", h)
43         node_ip = node[0].decode()
44         chunk_stream = io.BytesIO(chunk_data)
45         chunk_stream.name = chunk_file_name
46         files = {
47             'files': io.BufferedReader(chunk_stream),
48         }
49         print(files)
50         print("Uploading file to http://{}".format(node_ip))
51         response = requests.post(
52             'http://{}/5058/upload'.format(node_ip), files=files)
53         chunk_num += 1

```

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。

2.2 download.py

```

20 chunk_num = 1
21 with open(output_file_name, 'wb') as f:
22     while True:
23         chunk_file_name = output_file_name + '_chunk_' + str(chunk_num)
24         h = hash(chunk_file_name)
25         print("Hash of {} is {}".format(chunk_file_name, h))
26         node = client.call("find_successor", h)
27         node_ip = node[0].decode()
28         print("Downloading file from http://{}".format(node_ip))
29         response = requests.get("http://{}/5058/{}".format(node_ip, chunk_file_name))
30         print(response)
31         if(response.status_code != 200):
32             break
33         f.write(response.content)
34         chunk_num += 1
35

```

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。

3 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。

The screenshot shows the AWS Management Console interface for the 'MyChordFileSystemASG' Auto Scaling group. The top navigation bar includes tabs for Name, Launch template/configuration, Instances, Status, Desired capacity, Min, Max, and Availability Zones. The main content area displays the 'Group details' for 'MyChordFileSystemASG'. The details include:

Property	Value
Auto Scaling group name	MyChordFileSystemASG
Desired capacity	1
Status	Updating capacity...
Amazon Resource Name (ARN)	arn:aws:autoscaling:us-east-1:231931584364:autoScalingGroup:844103b4-7960-4279-901b-35e0b088756f:autoScalingGroupName/MyChordFileSystemASG
Date created	Sun Apr 30 2023 15:54:11 GMT+0800 (台北標準時間)
Minimum capacity	1
Maximum capacity	10

過陣子後發現 scaling policies 有新的 policy 出現

The screenshot shows the AWS Management Console interface for the 'MyChordFileSystemASG' Auto Scaling group, focusing on the 'Dynamic scaling policies' section. The top navigation bar is the same as the previous screenshot. The main content area displays the 'Dynamic scaling policies (1)' section. The details include:

Policy Name	Policy Type	Policy Status
disk_used_percent	Simple scaling	Enabled

Additional details for the 'disk_used_percent' policy:

- No alarm selected
- Add 1 capacity units
- 180 seconds before allowing another scaling activity

此時去 cloudwatch 找對應 instance 的 metric

Instances (1)							
<input type="text" value="Filter instances"/>							
<input type="checkbox"/>	Instance ID	Lifecycle	Instance type	Weighted capacity	Launch template...	Availability Zone	Health status
<input type="checkbox"/>	i-0b92f25f807542f14	InService	t2.micro	-	MyChordFileSystem-finz	us-east-1a	Healthy

Metrics (1) Info							
<input type="text" value="Search for any metric, dimension, resource id or account id"/>							
<input type="checkbox"/>	Instance name 1/1	AutoScalingGroupName	InstanceId	device	fstype	path	Metric name
<input type="checkbox"/>	No name specified	MyChordFileSystemASG	i-0b92f25f807542f14	xvda1	xfs	/	disk_used_percent

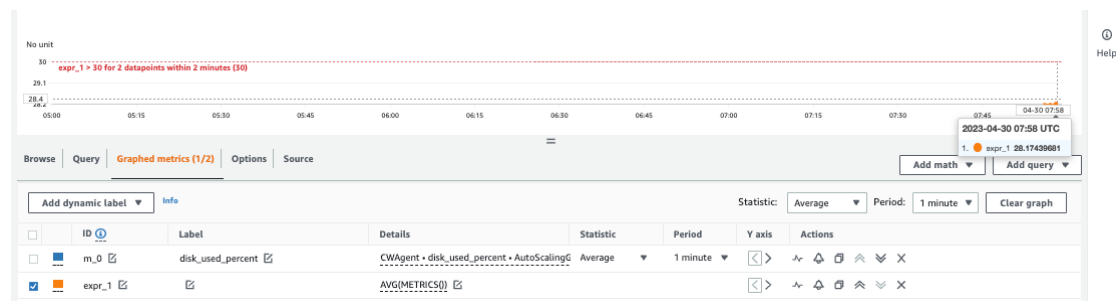
發現 alarm 也出現了

Alarms (1)						
<input type="text" value="Search"/>						
<input type="checkbox"/>	Name	State	Last state update	Conditions	Actions	
<input type="checkbox"/>	MyChordFileSystemASGDiskUsedAlarm	Insufficient data	2023-04-30 15:57:17	expr_1 > 30 for 2 datapoints within 2 minutes	Actions enabled	

過陣子狀態變 ok

Alarms (1)						
<input type="text" value="Search"/>						
<input type="checkbox"/>	Name	State	Last state update	Conditions	Actions	
<input type="checkbox"/>	MyChordFileSystemASGDiskUsedAlarm	OK	2023-04-30 15:58:36	expr_1 > 30 for 2 datapoints within 2 minutes	Actions enabled	

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

```
[ec2-user@ip-172-31-14-115 files]$ ls
[ec2-user@ip-172-31-14-115 files]$
```

從 client 用 upload.py 上傳幾個檔案試試

```

> cd uploadserver
> python3 upload.py b.txt 3.215.184.16
filename: b.txt
Hash of b.txt_chunk_1 is 3239240481
{'files': <_io.BufferedReader name='b.txt_chunk_1'>}
Uploading file to http://3.215.184.16
> python3 upload.py c.txt 3.215.184.16
filename: c.txt
Hash of c.txt_chunk_1 is 415054527
{'files': <_io.BufferedReader name='c.txt_chunk_1'>}
Uploading file to http://3.215.184.16
> vim d.txt
> python3 upload.py d.txt 3.215.184.16
filename: d.txt
Hash of d.txt_chunk_1 is 4055362934
{'files': <_io.BufferedReader name='d.txt_chunk_1'>}
Uploading file to http://3.215.184.16

```

結果如下：

```

[ec2-user@ip-172-31-14-115 files]$ ls
[ec2-user@ip-172-31-14-115 files]$ ls
b.txt_chunk_1 c.txt_chunk_1 d.txt_chunk_1
[ec2-user@ip-172-31-14-115 files]$

```

此時把 ASG Desired capacity 設為 2

The screenshot shows the AWS Management Console for the **MyChordFileSystemASG**. The **Details** tab is selected, showing the following information:

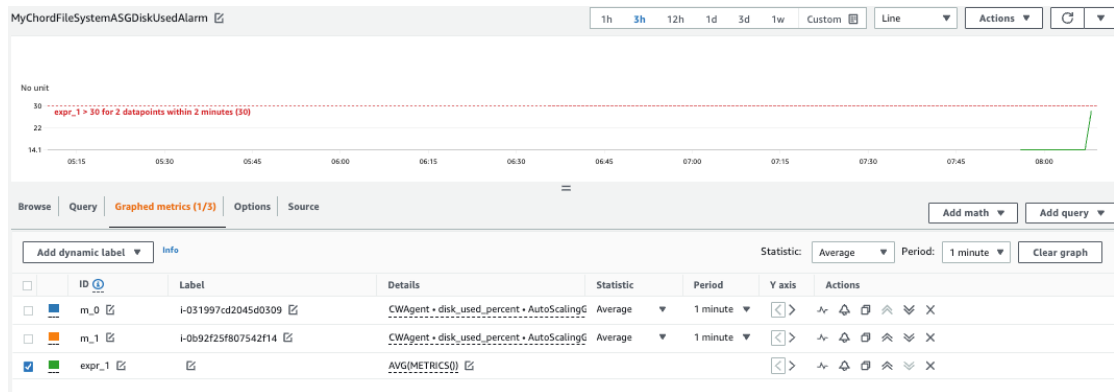
- Auto Scaling group name:** MyChordFileSystemASG
- Desired capacity:** 2
- Status:** Updating capacity
- Amazon Resource Name (ARN):** arn:aws:autoscaling:us-east-1:231931584364:autoScalingGroup:844103b4-7960-4279-901b-35e0b088756f:autoScalingGroupName/MyChordFileSystemASG
- Date created:** Sun Apr 30 2023 15:54:11 GMT+0800 (台北標準時間)
- Minimum capacity:** 1
- Maximum capacity:** 10

The **Instance management** tab is also visible, showing a table of instances:

Instance ID	Lifecycle	Instance type	Weighted capacity	Launch template...	Availability Zone	Health status	Protected from
i-031997cd2045d0309	InService	t2.micro	-	MyChordFileSystem-fina	us-east-1b	Healthy	
i-0b92f25f807542f14	InService	t2.micro	-	MyChordFileSystem-fina	us-east-1a	Healthy	

At the bottom, there is a section for **Lifecycle hooks** with a button to **Create lifecycle hook**.

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



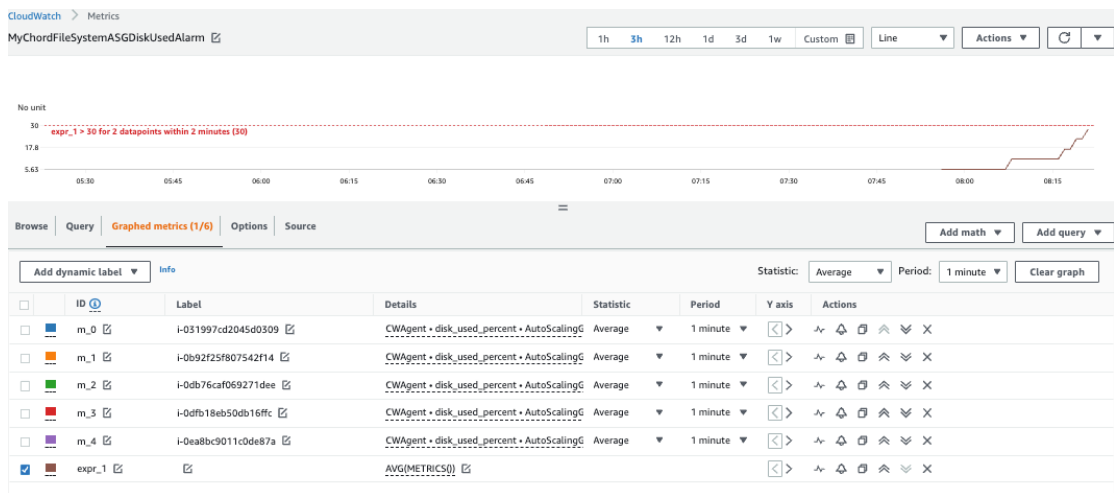
連進第二個 instance 看發現 flie 有被 migration 和 replica 進來

```
[ec2-user@ip-172-31-82-215 files]$ ls
b.txt_chunk_1 c.txt_chunk_1 d.txt_chunk_1
[ec2-user@ip-172-31-82-215 files]$
```

接著逐步把 instance 擴充到 5 個

Instances (5)								
<div>Filter instances</div>								
<input type="checkbox"/>	Instance ID	Lifecycle	Instance type	Weighted capacity	Launch template...	Availability Zone	Health status	Protected from
<input type="checkbox"/>	i-031997cd2045d0309	InService	t2.micro	-	MyChordFileSystem-fins	us-east-1b	Healthy	
<input type="checkbox"/>	i-0b92f25f807542f14	InService	t2.micro	-	MyChordFileSystem-fins	us-east-1a	Healthy	
<input type="checkbox"/>	i-0db76caf069271dee	InService	t2.micro	-	MyChordFileSystem-fins	us-east-1b	Healthy	
<input type="checkbox"/>	i-0dfb18eb50db16ffc	InService	t2.micro	-	MyChordFileSystem-fins	us-east-1a	Healthy	
<input type="checkbox"/>	i-0ea8bc9011c0de87a	InService	t2.micro	-	MyChordFileSystem-fins	us-east-1a	Healthy	
Lifecycle hooks (0)								

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

```

> python3 upload.py a.txt 3.215.184.16
filename: a.txt
Hash of a.txt_chunk_1 is 2756827572
{'files': <_io.BufferedReader name='a.txt_chunk_1'>}
Uploading file to http://54.152.226.234
Hash of a.txt_chunk_2 is 1370087605
{'files': <_io.BufferedReader name='a.txt_chunk_2'>}
Uploading file to http://3.215.184.16
Hash of a.txt_chunk_3 is 2707073571
{'files': <_io.BufferedReader name='a.txt_chunk_3'>}
Uploading file to http://54.152.226.234
Hash of a.txt_chunk_4 is 3608324371
{'files': <_io.BufferedReader name='a.txt_chunk_4'>}
Uploading file to http://3.91.178.60
Hash of a.txt_chunk_5 is 844703452
{'files': <_io.BufferedReader name='a.txt_chunk_5'>}
Uploading file to http://3.91.178.60
Hash of a.txt_chunk_6 is 1307373017
{'files': <_io.BufferedReader name='a.txt_chunk_6'>}
Uploading file to http://3.215.184.16
Hash of a.txt_chunk_7 is 2238907549
{'files': <_io.BufferedReader name='a.txt_chunk_7'>}
Uploading file to http://3.85.99.175

~/Desktop/專案/分散式系統/chord-part-2/uploadserver ma

```

Ssh 進每個 instance 看

```

[ec2-user@ip-172-31-14-115 files]$ ls
[ec2-user@ip-172-31-14-115 files]$ ls
b.txt_chunk_1 c.txt_chunk_1 d.txt_chunk_1
[ec2-user@ip-172-31-14-115 files]$ ls
b.txt_chunk_1 c.txt_chunk_1 d.txt_chunk_1
[ec2-user@ip-172-31-14-115 files]$ ls
a.txt_chunk_1 a.txt_chunk_3 a.txt_chunk_5 b.txt_chunk_1 d.txt_chunk_1
a.txt_chunk_2 a.txt_chunk_4 a.txt_chunk_6 c.txt_chunk_1
[ec2-user@ip-172-31-14-115 files]$ █

```

```

[ec2-user@ip-172-31-82-215 files]$ ls
b.txt_chunk_1 c.txt_chunk_1 d.txt_chunk_1
[ec2-user@ip-172-31-82-215 files]$ ls
a.txt_chunk_2 a.txt_chunk_6 a.txt_chunk_7 b.txt_chunk_1 c.txt_chunk_1 d.txt_chunk_1
[ec2-user@ip-172-31-82-215 files]$ █

```

```
[ec2-user@ip-172-31-1-79 files]$ ls
a.txt_chunk_1 a.txt_chunk_3 a.txt_chunk_7 b.txt_chunk_1
[ec2-user@ip-172-31-1-79 files]$
```

```
[ec2-user@ip-172-31-11-94 ~]$ cd files/
[ec2-user@ip-172-31-11-94 files]$ ls
a.txt_chunk_2 a.txt_chunk_5 b.txt_chunk_1 d.txt_chunk_1
a.txt_chunk_4 a.txt_chunk_6 c.txt_chunk_1
[ec2-user@ip-172-31-11-94 files]$
```

```
[ec2-user@ip-172-31-82-172 ~]$ cd files/
[ec2-user@ip-172-31-82-172 files]$ ls
a.txt_chunk_1 a.txt_chunk_4 a.txt_chunk_7 c.txt_chunk_1
a.txt_chunk_3 a.txt_chunk_5 b.txt_chunk_1 d.txt_chunk_1
[ec2-user@ip-172-31-82-172 files]$
```

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