Exercise on pushing Custom Metrics to CloudWatch

Taken from : https://aws.amazon.com/premiumsupport/knowledge-center/cloudwatch-custom-metrics/

You can create a custom CloudWatch metric for your EC2 instance statistics by creating a script through the AWS Command Line Interface (AWS CLI). Then, you can monitor that metric by pushing it to CloudWatch.
Be sure that you configure the AWS CLI for use with the instance that you want to monitor. This requires your AWS account's *Access Key ID* and *Secret Access Key*, which are available on the RosettaHub console – expand the *AWS* menu on the left and click on *IAM Users. https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html*

To create your custom metric:
1. Copy or download (from here) the following Bash script, and then save it to your instance (for example, mem.sh). This example script shows the values that you can publish in CloudWatch. In this example, we are using the put-metric-data API call to push the following values to CloudWatch: the percentage of used memory, the number of total and port 80 TCP connections, the number of users currently logged in, and the percentage of I/O wait time.

```
#!/bin/bash
INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-data/instance-id)
USEDMEMORY=$(free -m | awk 'NR==2{printf "%.2f\t", $3*100/$2 }')
TCP_CONN=$(netstat -an | wc -l)
TCP_CONN_PORT_80=$(netstat -an | grep 80 | wc -l)
USERS=$(uptime |awk '{ print $5 }')
IO_WAIT=$(iostat | awk 'NR==4 {print $5}')

aws cloudwatch put-metric-data --metric-name memory-usage --dimensions Instance=$I
NSTANCE_ID --namespace "Custom" --value $USEDMEMORY
aws cloudwatch put-metric-data --metric-name Tcp_connections --dimensions Instance
=$INSTANCE_ID --namespace "Custom" --value $TCP_CONN
aws cloudwatch put-metric-data --metric-name TCP_connection_on_port_80 --dimension
s Instance=$INSTANCE_ID --namespace "Custom" --value $TCP_CONN_PORT_80
aws cloudwatch put-metric-data --metric-name No_of_users --dimensions Instance=$IN
STANCE_ID --namespace "Custom" --value $USERS
aws cloudwatch put-metric-data --metric-name IO_WAIT --dimensions Instance=$INSTAN
CE_ID --namespace "Custom" --value $IO_WAIT
```

2. After creating the Bash script, give execute permissions to the file.

```
$ chmod +x mem.sh
```

3. Run the Bash script to check that it works. **Note** the script above differs slightly from the script that is available at the link – the USERS awk statement prints $5 not $6. Also you must change the script above to reference YOUR instance-id. If you are placing such a custom metric push in UserData when launching an instance you can retrieve the instance-is with `curl -s http://169.254.169.254/latest/meta-data/instance-id`

Push your metric to CloudWatch

1. Create a cron job:

```
$ env EDITOR=nano crontab -e
```

2. Add this line to execute your script every minute:

```
*/1 * * * * /home/ec2-user/mem.sh
```

3. Save and exit.

When the crontab is saved, `crontab: installing new crontab` is displayed.

Monitor your EC2 instance

Find your custom metric in the CloudWatch console:

1. Sign in to the CloudWatch console.
2. Choose Metrics.
3. Choose the All Metrics view.
4. Choose Custom.
5. Choose the dimension Instance.
6. Select your custom metric by its `InstanceId` and Metric Name.
7. View the graph of your metric.

Other Uses

You can use this example to build your own logic to process multiple dimensions (memory, CPU, swap, and so on), and then push that metric data to CloudWatch.
For example, suppose that you benchmark your application. Then, you discover that when the I/O wait time and percentage memory usage reach a certain threshold, the system stops functioning properly. To address this problem, you monitor both values simultaneously in a script, storing the logical AND of the values in a third variable that you push to CloudWatch.

```
c=0
if [[  $IO_WAIT > 70 && $USEDMEMORY > 80 ]]
then
  c=1
fi
aws cloudwatch put-metric-data --metric-name danger --dimensions Instance=i-0c51f9
f1213e63159  --namespace "Custom" --value $c
```

For normal conditions, this variable is 0 (zero). For situations when both conditions are met, the value is set to 1 (one). You can then build custom alarms around these parameters to warn of problematic situations for your system. For more information see : Publish Custom Metrics