Table of contents

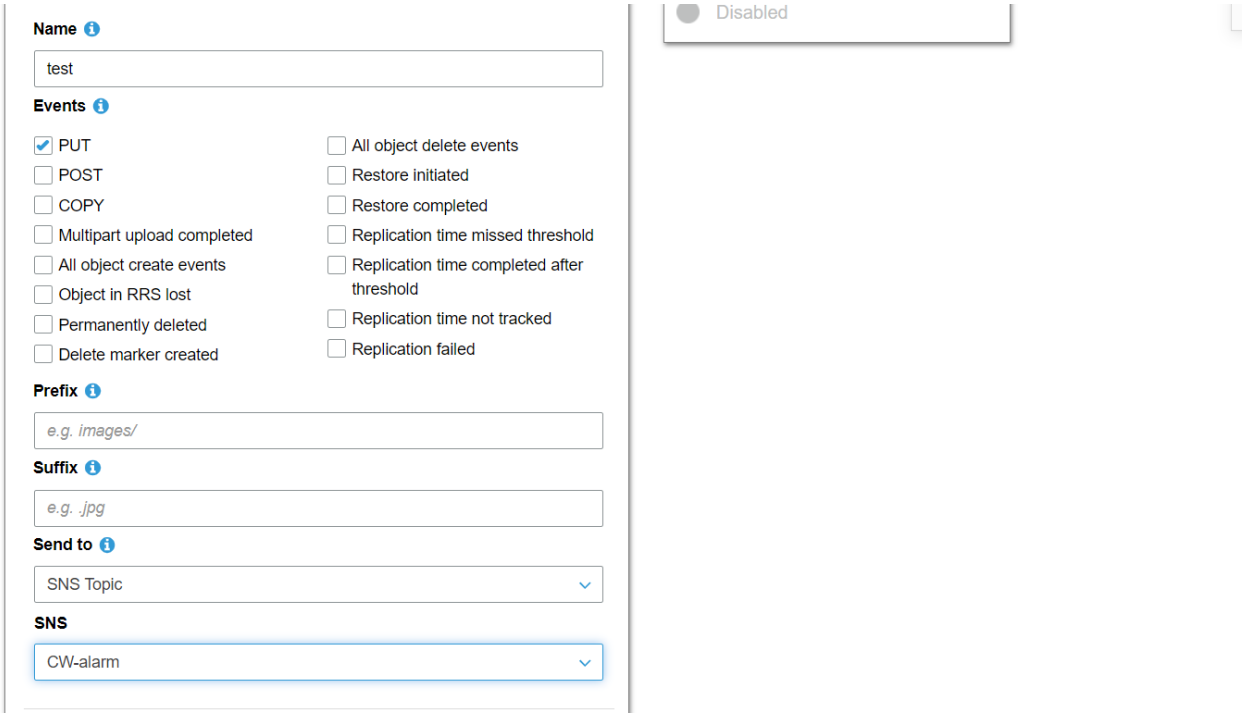
S3 events

- S3 events is similar to cloudwatch events . This is commonly used for object level activities like put,get etc
- S3 events sends out notification to SNS,lamba and SQS when the action is called.
- In order to use the SNS for S3 events however , we need to add access policy to the SNS topic.
- Navigate to SNS and click on edit the acces policy

```
{
  "Version": "2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish",
        "SNS:Receive"
      ],
      "Resource": "arn:aws:sns:us-east-1:384395217903:CW-alarm",
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "384395217903"
        }
      }
    },
    {
      "Sid": "bucketaccess",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
```

```
        },
        "Action": "SNS:Publish",
        "Resource": "arn:aws:sns:us-east-1:384395217903:CW-alarm",
        "Condition": {
          "ArnLike": {
            "AWS:SourceArn": "arn:aws:s3:::aws-devops-4"
          }
        }
      }
    ]
}
```

- Use policy as above, just make sure to replace te account id in source owner alng with topic arn and bucket arn
- Once done , navigate to S3 . Go to properties and click on s3 events
- Click on add notifications . Give it a appropriate name and select the event on which we want notification for ex put
- Select the sns topic and click on save



- Test the event by uploading an object in S3

Cloudwatch custom metrics :

- We have seen that cloudwatch gives us certain pre-defined metrics.
- Along with the above , we can also define some custom metrics .
- For example , even though we have CPU utlization metric by default, we dont have memory (RAM) utlization metric

Creating memory utlization metric

- Launch an ec2 instance

- With some versions of Linux, you must install additional Perl modules before you can use the monitoring scripts. For amazon linux use

```
sudo yum install perl-Switch perl-DateTime perl-Sys-Syslog perl-LWP-Protocol-https
-y
```

- At a command prompt, move to a folder where you want to store the monitoring scripts and run the following command to download them:

```
curl https://aws-
cloudwatch.s3.amazonaws.com/downloads/CloudWatchMonitoringScripts-1.2.2.zip -O
sudo yum -y install perl-Digest-SHA.x86_64
```

- Run below commands to install the monitoring scripts which we just downloaded

```
unzip CloudWatchMonitoringScripts-1.2.2.zip && \
rm CloudWatchMonitoringScripts-1.2.2.zip && \
cd aws-scripts-mon
```

- Make sure your Instance has permissions to publish metrics in cloudwatch (This can be acheived using an IAM role)
- The following example collects all available memory metrics and sends them to CloudWatch, counting cache and buffer memory as used

```
./mon-put-instance-data.pl --mem-used-incl-cache-buff --mem-util --mem-used --mem-
avail
```

- Above script will send data to cloudwatch everytime it is executed
- We can scheudle it to send data periodically by using cronjob as well
- in order to set the cronjob use

```
crontab -e
```

- Enter the cron expression as below

```
*/1 * * * * /home/ec2-user/aws-scripts-mon/mon-put-instance-data.pl --mem-used-
incl-cache-buff --mem-util --mem-used --mem-avail
```

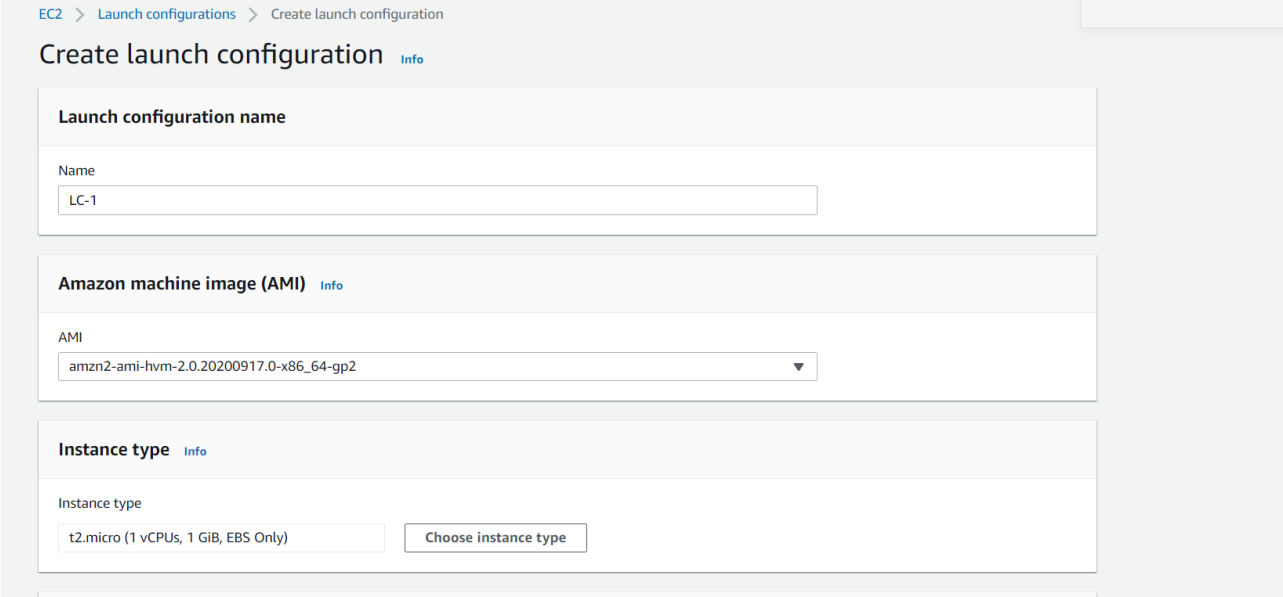
- save the file and exit
- Above setting will make sure the script is run every minute
- You should be able to see the graph in cloudtwahc metrics under custom namespaces . Or you can search using the instance id

Auto scaling

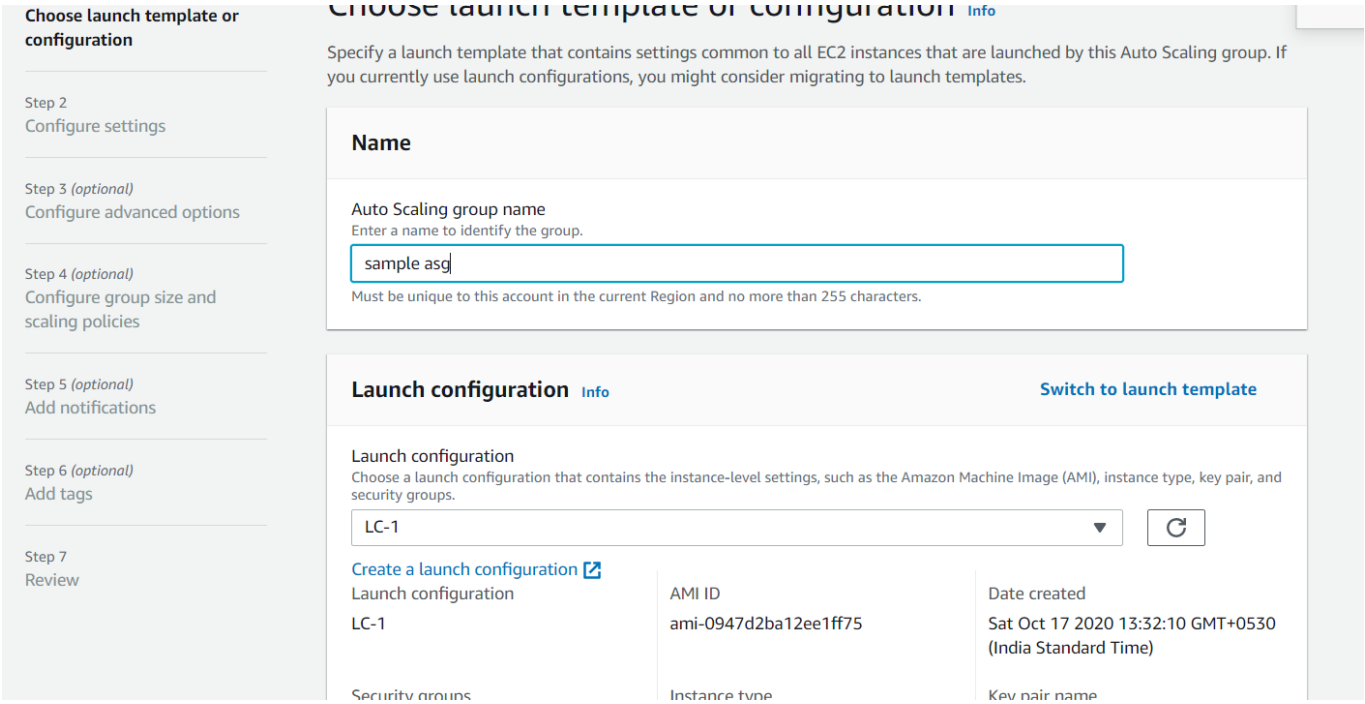
- Auto scaling is a part of horizontal scaling stratergy wherein we scale the instances based on scaling policies
- These scaling policies are derived over any of the cloudwatch metric
- For ex. scale the instances if the cpu utlization goes beyond 50%
- These scaling policies can be configured for scaling in or scaling out actions as well

Configuring auto scaling group

- Navigate to EC2
- Go to launch configurations - This decides the ec2 instances which will be launched ,should have which configuration
- Select the AMI - Mostly in indiustry , we use custom AMI where our application is already installed
- Select the instance type



- Under additional configuration we have an option of using psort request . which we can skip for now
- Under IAM role, if we require any access to other service we can create and select the role
- We can enable detailed monitoring as of now as we want to test the scaling results asap
- Select appropriate amount of volume . Can be kept as default
- Configure the security group and keyname as we do it for EC2 and click on create launch cofiguration
- Once launch configuration is created , we can move to auto scaling group
- Select the launch configuration which was just created and click on create auto scaling group
- Give to auto scaling group a name and click on next



- Select the VPC and subnet in which we want our instances to launch . We can select multiple subnets or can stick with one. Cick next
- Under advanced options we have the ability to put the auto scaled instances behind an ELB . We can select a target group under which these instances will be registered
- Health checks is based on which the auto scaling group decides if more instances are needed . WE can keep them as default which is EC2 status check . Adiitionaly we can also add elb health check
- Click next
- Under group size we can select the desired, minimum and maximum limit we want our instance count to be
- Let us keep desired as 3 . Minimum as 1 and maximum as 5 . ASG will try and keep the count till 3 . where minimum count will be 1 and maximum will be 5
- For scaling policies ,we can define if we want to dynamically scale our ec2 instances based on metrics. Which we do . Hence select the "target tracking scaling policy "
- Let us keep the scaling policy name as it is . Chose the metric name as Average CPU utlization
- Target value can kept as 50 .
- keep other options as default and click next
- Under notifications we can add notification for each action performed to the SNS topic. This is optional
- Add tags if required and go ahead and create the auto scaling group
- Since we kept the minimum value as 1 , we can see the instance getting launched in the instances tab
- If we need to test the auto scaling however , we need to load the instance in such a way that cpu utlization goes beyond 50%
- In order to acheive that , we have a linux utiltiy called stress . Stress hogs the resources for a certain amount of time and then releases them.
- Log in the instance which has been launched and run below commands

```
sudo amazon-linux-extras install epel -y
sudo yum install stress -y
stress --cpu 1 --timeout 300
stress --vm 4 --vm-bytes 102M
```

- Above commmands will instal stress utility and hog the resources
- You can monitor the same from the metrics
- Observe that once the CPU utlization graph goes beyond 50 , other instances will be launched as auto scaling group will be triggered