Importance of switching on/ off the instance using a scheduler: In our early days, we were running our instances 24/7. One fine day we realized that we needed the instances to be up and running only for 10-12 hours per day i.e during our normal working hours. Thereby we were paying unnecessarily for the unused 12-14 hours.

Our first approach was to turn instances on/off manually, but we later realized that this was prone to human errors. Additionally as we were expanding , the manual effort required to turn them on/off was quite high and unnecessary too.

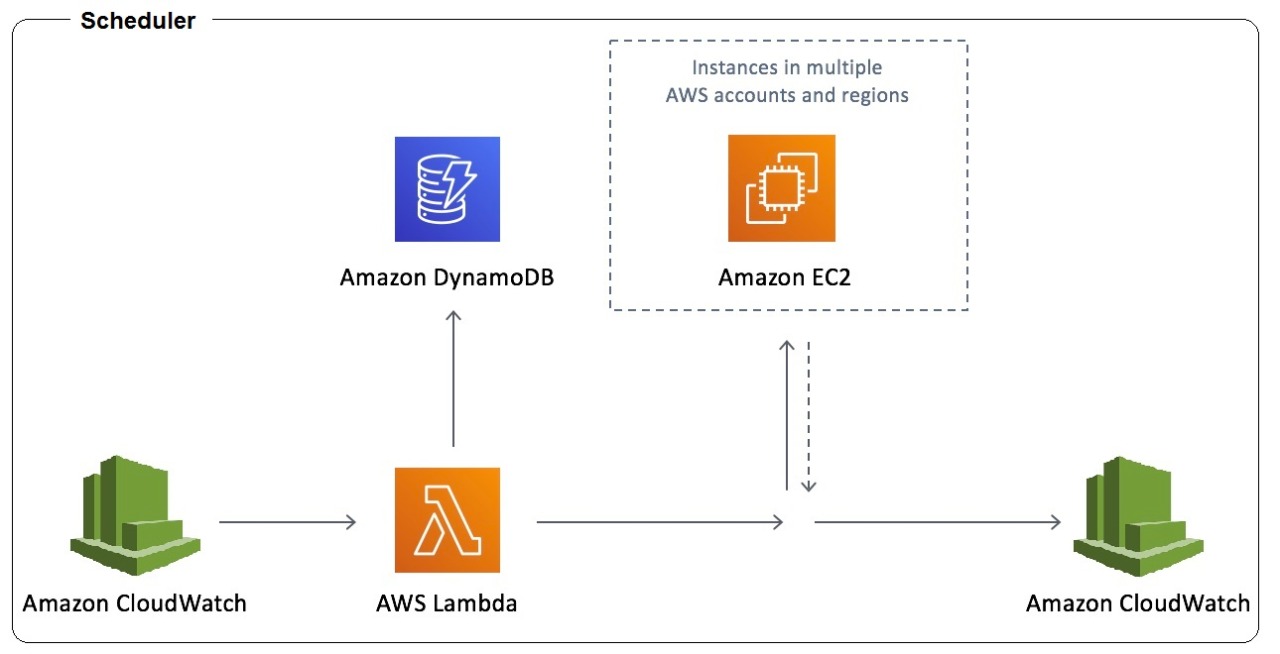
With all this in mind we moved on to schedulers and this step has given us a huge cost saving of around 70 %(approximately) and also saves us approximately 10 man hours per week.

**Scheduler:**

The Scheduler is a solution that enables customers to easily configure custom start and stop schedules for their Amazon EC2. The solution is easy to deploy and can help reduce operational costs for both development and production environments.

**Normal instance scheduler**

**Scheduler implementation in IaaS(Infrastructure as a service) :**



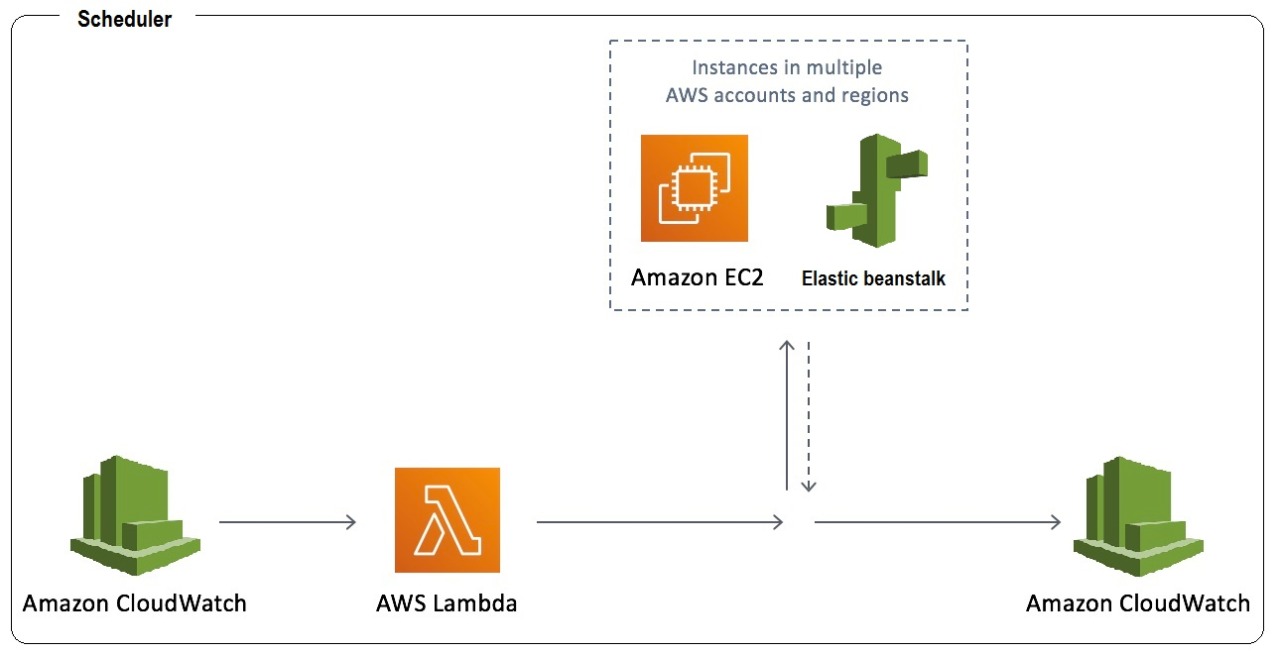
* When we create an instance on EC2, a unique instance ID gets created and this doesn’t change throughout the life of the instance.
* EC2 instance id is passed to the lambda function to start/stop EC2 instance as per schedule.

**Scheduler implementation in PaaS (Platform-as-a-service):**

* AWS Elastic Beanstalk Environment has unique id for each application (which is different from the instance id).
* Initially we configured our scheduler in the same way as Iaas i.e scheduler pointing to the instance ID. However we faced our 1st challenge when we deployed a new version of our application.

**New way of implementation for scheduler (PaaS):**

* Due to the deployment a new instance ID had got generated but the scheduler was pointing to the old instance ID. Hence the scheduler didn’t work.
* We tackled this issue by using the environment ID instead of Instance ID. This way the scheduler was always pointing to the intended environment id.



**Components of the solution:**

1. Cloud watch - triggers the script to start and stop the services as per schedule
2. Lambda function - executes the script file
3. Cloudwatch - stores the log files

**Benefit of new solution:**

1. Customers who use this solution to run instances only during regular business hours can save up to 70%(approximately) compared to running those instances 24 hours a day.
2. Easy to configure and monitor,
3. using the same trigger script both Environment and EC2 instance can be started/stopped
4. No need to configure DynamoDb
5. No manual efforts required and hence lot of time saved from a Programmer’s perspective

**Challenge from usability:**

* Server/Application should be up and running during the standard office hours on all working days.i.e Monday to Friday 9 AM to 10 PM IST.
* Hence 2 lambda functions have been created - one to start the server and the other to stop the server. Appropriate log files have been created for monitoring server on and off times.
* So if someone wants to use the application during the scheduled down time, the application needs to be brought up manually by logging in to AWS. To make it easier we are trying to create a mobile service to perform the same on adhoc basis.

**Conclusion:**

* Input parameters to the cloud watch / lambda utilities should be determined based on the implementation model i.e Iaas or Paas. Only this will ensure successful functioning of the scheduler.