Framework for availability assurance of services running on AWS EC2 Spot instances

MSc Research Project
Cloud Computing

Jerry Danysz
Student ID: 16148690

School of Computing
National College of Ireland

Supervisor: Victor Del Rosal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Jerry Danysz
Student ID:	16148690
Programme:	Cloud Computing
Year:	2019
Module:	MSc Research Project
Supervisor:	Victor Del Rosal
Submission Due Date:	12/08/2019
Project Title:	Framework for availability assurance of services running on AWS EC2 Spot instances
Word Count:	6537
Page Count:	31

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	6th August 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Framework for availability assurance of services running on AWS EC2 Spot instances

Jerry Danysz
16148690

Abstract

Amazon Web Services since early 2009 is offering its spare capacity at a discounted price. This discount comes at the price of unpredicted price changes and sudden instance termination. In this paper, we are presenting a solution to one of the challenges when using AWS Spot Instances, that is Termination of the instance on short notice.

Presented in this paper, Spot Instance Management System (SimS) can effectively manage spot instances and keep up the availability on the desired level. Thanks to machine learning, risk assessment mechanism and proactive actions system can assure required three nines SLA. The results are showing that usage of a proactive system like SimS proves to be valuable if our main goals are low running costs and availability of the systems running on the spot instances.

1 Introduction

Starting in 2009, Amazon as the first cloud provider started offering the new cost model for its flag service of providing virtual instances (EC2) In contrast to their other models ON-Demand and Reserved, Spot instances offered significant price drop per hour of usage. This offering was not without its hooks. The conditions for providing the spot instance were related to two factors, the availability of the spare capacity in the region and supply and demand.

One of the base assumptions behind spot instances is that price is dynamic and can change anytime based on available capacity and current demand for the type of instance.

Usage of spot instances requires from the customers to set maximum instance price (per hour) that they wish to pay and also to understand the risk that if the price of the instance changed above the maximum price the instance would be suddenly terminated and all data lost if it is not saved elsewhere.

Until 2015 the instance termination would be executed without prior notification to the customer, therefore usage of the spot instance even if its cost-effective could be limited only to applications that are either not critical or are design for interruptions.

Since 2015 Amazon improved their spot service by offering a different type of termination behaviours ranging from default termination through stopping the instance to hibernation of the instance.

All of those options have specific requirements that need to be taken into account before requesting the spot instance. None of those is offering actual prevention from the instance interruption itself.

The Spot Instance Management System that we are presenting in this research has the main goal to counter the sudden unexpected termination of the systems running on EC2 Spot instances.

This is achieved by the application of machine learning and prediction of the next price change. In case systems detect that within next hour price would change and risk of the instance termination is High, the appropriate actions are taken.

The actions could be live migration to another availability zone or redeployment of spot instance with a higher maximum bid. It is worth to notice that all actions are done proactively, while the risky instance is still running, therefore minimising the potential downtimes of any service using EC2 spot instances.

In this paper, we are investigating if it is possible to maintain the agreed service level of 99.9% that is common for a business-critical application while savings costs by running those applications on EC2 Instances.

Our goal is to prove or disprove that EC2 Spot instances can be used in combination with the management systems like The SimS as reliable hosting platforms for critical applications.

Research Question : For mission-critical services utilising EC2 Spot Instances, how can the service availability requirement of 99.9% be assured with spot price prediction using machine learning and automated system migration to avoid sudden interruptions

2 Related Work

In this section, we will look into past work in areas of price prediction, bidding strategies and interruption mitigation solutions.

2.1 Prediction of Spot Instances Pricing

We are starting a literature review section with the overview of publications presenting methods used in various research to predict the price of the AWS Spot Instances accurately.

Spot Price Prediction (SPP) has been a topic of research since the introduction of Amazon spot instances in 2009. Ben Yehuda in his paper Agmon Ben-Yehuda et al. (2013) approached topic of predicting spot price by deconstruction and reverse engineering of hypothetical spot instance algorithm that despite common assumption spot prices might not be supply and demand-driven but instead are randomly generated from dynamic hidden reserve price.

To confirm his hypothesis, he analysed the spot market and divided it to three pricing epochs with each epoch change at the significant change of SPP Pricing models. As the conclusion, the author is acknowledging that there is a market element in the price, but prices are still driven from the hidden reserved price.

Authors of Singh and Dutta (2015) seem not to fully agree with Ben's Yehuda conclusion as their algorithm model for Dynamic Price prediction is accounting for global market trends and local Seasonality.
The dynamic Price Prediction model is presenting two types of predictions short-term (hourly based) and long-term (a week ahead) based on analysis of 9 months of historical data for the top ten used spot instances.

In the conclusion of their work, authors present the prediction result of an average 9.4% prediction error in short-term prediction and 20% in long-term (five days ahead) price prediction.

In the work of Wallace et al. (2013), authors are presenting a novel approach to aws spot price prediction by using a moving simulation model to create an artificial neural network-based algorithm for price prediction.
Authors used historical data available for only the medium size instances in a period of 7 months to train the MLP model resulting in 4% prediction error in short-term prediction (hourly prediction) on average for medium size spot instances concluding that neural network models are well suited for the prediction of price changes of Spot Instances.

This conclusion has been initially confirmed by the researchers concluding the evaluation of algorithms used for spot price prediction Arévalos et al. (2016) who compared above prediction method (naming it a state-of-the-art method) with Support Vector POly Kernel Regression (SMOReg), Gaussian Process and Linear Regression.

Models have been trained for the month with 12 months of historical data for the three most used types of instances. The prediction has been generated for short (next hour), medium (half-day) and Long (next day) periods. As the conclusion, the neural network-based algorithms are performing better than others for medium (half-day predictions) where SMOReg is better suited for predictions with hight variable months.

Neural Network-based algorithms have also been evaluated in Amekraz and Youssef (2016), in this paper authors have used Adaptive neural fuzzy interface system (ANFIS) model that is the combination of neural networking and fuzzy interface system to analyse three months of historical data for one type of EC2 Instance.

In this work, authors are leaning towards Ben Yehuda view that behind the appearance of randomness, there is regularity and order that is hidden. In conclusion, authors are using the chaos theory principals, can conclude that spot pricing model is in fact, chaotic and models like ANFIS would be better suited for price prediction than stochastic forecasting presented by [7].

Authors of Zhao et al. (2015) are presenting a different approach to a prediction by using time-based series forecasting method, ARIMA Model (Auto-Regressive Integrated moving average) that is lighter compared to machine learning technics like neural networking.

Authors similar to researchers in Singh and Dutta (2015) have added season component to their ARIMA model effectively changing it to SARIMA model. SARIMA has been used to analyse five months of historical data and create a prediction that is close to the average price for a period of 48h.

From the above analysis, we can see that there are different approaches to prediction of the spot instances pricing ranging from reverse engineering and understanding what principals are behind the price level, through the classical statistical approach to the most modern use of artificial intelligence and machine learning techniques like neural networking.

Based on the above, we can state that all of those approaches are valid and have great potential if researched further. In our research proposal, we are planning the usage of machine learning algorithms to predict price and potential price change.

2.2 AWS Spot bidding strategies overview

Researches like in previous sections are trying to find the best strategy to find a golden bid to assure spot instance availability, Authors in Andrzejak et al. (2010) ask the question how to bid if there are strict target dates or SLA and they focus the investigation of their bidding strategies with that goal. Authors of Li et al. (2015) in their overview is dividing common biding approaches into three types:

- White box approach where bidding strategies are taking into account interactions between different market participants and effectively bidding can influence a spot price.
- Grey box approach has more individual biding strategies wherein contrary to white-box, market interactions are not taken into account, but strategies are focusing mostly on workload, cost and availability of the resources.
- Black Box approach, which consists most common strategies derives its biding from historical spot pricing data and do not focus so much on workload, cost and availability nor interactions between market participants.

Five most classic strategies in Black Box approach have been discussed in Li et al. (2015) and Voorsluys and Buyya (2012)
those strategies are:

- The minimum price, where the bid is based on spot historical minimum price
- Mean, the bid price is set as the mean of all values of the historical spot price
- High, the bid price on the maximal price observed in historical data
- Current, the bid price is set as the value of current spot price
- On-demand is the bid price equal to the on-demand price of the instance.

Above five strategies have also been incorporated to the solution for reliable provisioning of spot instances in Voorsluys and Buyya (2012), where authors are combining all five strategies with fault tolerance techniques like migration to assure the most reliable solution for the limitations of spot instances.

In the survey on the spot pricing Kumar et al. (2017) authors are presenting four strategies that are similar to earlier introduced five.

Those strategies are :

- Bidding on near to reserve price
- Bidding on above the average price calculated from the historical data
- Bidding close to the on-demand price
- Bidding over the on-demand price

We can see here that in both cases, we have bidding on-demand and above demand prices that are meant to assure the availability of the resource.

The biding strategies are divided into two types of the approach in Li et al. (2015), statistical-based bidding strategies that are independent of auction-based strategies and are adapted for Map-reduce jobs and economic-based strategy, which are the auction and game-based strategies.

BlackBox strategies (auction-based) in the work of Li et al. (2015) have been extended by a new type of Bidding, on a feedback control based mechanism. This is the mechanism using historical data as the input reference and analysing it using a proportional-integral controller to deliver the final bid.

The bid forecasting has also been investigated using time series forecasting in Chhetri et al. (2017) and algorithms like NATIVE, SEASONAL NATIVE (SNATIV), ARIMA,ETS, STL and TBATS to accurately predict most effective bid based on 110 days of historical data, in the conclusion authors admit that snative (seasonal native) algorithm had the best scores in terms of predicting bid price when ARIMA failed with all of the predictions.

According to researchers, the seasonal component and extreme price spikes have cause ARIMA not to be able to predict with much of the success.

To summarise there is a wide variety of approaches to biding for spot instances as we could see above some researchers are trying to use forecasting techniques to predict the best bid, others are approaching the Bidding with a strategy of looking into the past historical data and bid on the average or maximum bid price.

Cloud consumers that are constrained by the time or SLA can choose to bid according to on-demand or above on-demand price, while this will give them more certainty field of available, it may not necessarily bring cost savings foreseen by use of spot instances.
A. Andrzejak is his work Andrzejak et al. (2010) advice that in those cases the best is to bid on more powerful machines that may be more expensive but will execute job quicker hence filling the objecting of time and possible cost

2.3 Mitigation of interruption risk

During our search for relevant material, we have come across comparable work in all of the fields of Spot Pricing in the cloud Li et al. (2016). Base on that work, we have looked into two specific techniques to address the termination risk for workload running on the AWS Spot Instances.

Authors of Li et al. (2016) have defined checkpointing and migration among the different types of mitigating the risk of work interruption.

a) Checkpointing

In the overview work of Li et al. (2016), authors are pointing to four different approaches to the checkpointing,

- adaptive checkpointing
- application-centric scheme
- enhanced incremental adaptive scheme
- hourly scheme

Adaptive checkpointing was also a topic of the research done in Jangjaimon and Tzeng (2015). Authors used adjusted Markov Model (AMM) and implemented multi-level checkpointing for multithreaded applications. In their conclusion, authors state that adaptive checkpointing brings lower checkpoint overhead and may reduce the overall cost of the job run.

Hourly based checkpointing was one of the components of the bigger solution provided in Voorsluys and Buyya (2012).
In their work, authors claim that the use of hourly checkpointing guarantees that only used time for the processing is paid.

Authors of Nicolae and Cappello (2013) have presented checkpointing technique based on the incremental snapshot of a single disk image instead of a whole virtual machine, with their solution the dedicated repository is used.
Authors have concluded that disk image checkpointing results in a reduction of checkpointing downtime.

The exciting approach was also presented in the Neto et al. (2018). Authors have decided to use price prediction techniques to predict interruption and based on that prediction to set up checkpointing intervals. During the research, historical data of 6 months have been used to predict price changes in 7 days period.

b) Migration

Migration can be understood as a different type of checkpointing, like in work of Nicolae and Cappello (2013) the snapshot is taken, in this case of the whole instance and then used to resume work on another instance.

Authors of Voorsluys and Buyya (2012) have incorporated the migration of persistent VM State to their reliable provisioning solution; in their case, the snapshot is used to lease new instance under new biding terms.

The migration framework was described in the Huang et al. (2017). Authors created a framework that could be used to migrate virtual machines based on KVM. Their solution provides four modules

- performance migration
- geolocation migration
- encrypted migration
- random migration

Despite work being interesting is very unlikely that it could be easily adapted to the Amazon spot instance market.

Another way of performing migration mentioned in Li et al. (2016) is using nested virtualisation as a means to have greater control over the nested virtual machine and then migrate it if there is a risk of the host being interrupted, unfortunately, this solution currently would not be possible on AWS since Amazon EC2 Instances are not supporting nested virtualisation.

c) Other mitigation techniques

Apart one of the most popular methods described in previous point authors of full review Li et al. (2016) has taken note of other techniques mentioned in professional literature like :

- Duplication and redundancy
- Lineage-based recovery
- Stop and redo model
- Service Scaling Down
- Hybrid spot instance

Purpose of this section in the literature review is to give an overall overview of current techniques of mitigation the risk of service interruption.

Each of the mentioned techniques has its benefits but also costs.

Checkpointing and migration might have a cost of extra overhead due to the need for extra storage needed to store a snapshot of current work.

Duplication and redundancy would require additional computing power in order to assure that data are duplicated and consistent.

Nested virtualisation or hybrid spot instances would required change in a way how current services are provided by Amazon, while those changes could benefit the cloud consumer, it would be very likely against the business interest of cloud provider.

3 Methodology and Design

3.1 Case Scenario

For this research, we assume a scenario where IT Service Provider is responsible for providing a business-critical batch processing and ERP application for European Financial Institution with agreed Availability of the service on the level of 99.9%

The IT Service Provider and European Financial Institution has agreed upon a project to run the batch processing application using only AWS Spot Instances for the cost-effectiveness

3.1.1 Application Business Critically

The criticality matrix determines the application critically level with three main aspects

- Availability - Impact if information Availability is affected
- Integrity - Impact if information Integrity is affected
- Confidentiality - Information Confidentiality Level

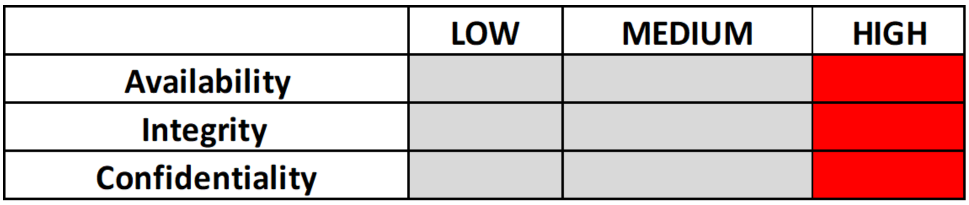


Figure 1: Business Application Critically Matrix

As seen above, the application is highly critical to the core business of the European Financial Institution.

If Availability of the system and therefore, the application is compromised, the ability of proper institution operations is degraded and therefore could lead to substantial losses in both profits and reputation.

In case when the integrity of the information in the system is affected; this could lead to substantial losses in profits as well as could be a risk to the confidentiality of the information.

Since system holds amounts of confidential information, including information that needs to be secured under the GDPR, in case of breach of confidentiality Institution could face legal actions based on GDPR provisions as well as the loss of customer trust, that could eventually lead to the loss of profits and the reputation.

3.1.2 Service Level Requirements (SLR)

Based on critically matrix following service level requirements have been presented

- Application Criticality : High
- AWS Region : eu-central-1
- Cummlative Downtime including maintance 3.65 days per year
- Mean Time to Respond : 2 minutes
- Mean Time to Resolve : 15 minutes

Based on the above Service Level Agreement has been agreed with Service Level of three nines (99.9%)

Service Level will be measured by using monitoring system site24x7, where the system will measure Availability of the system by checking HTTP response codes and general Availability of the host.

3.2 Project Development Steps

In this section, we are discussing our approach and steps taken during the development stage required for this research.

3.2.1 High-Level Project Plan

In this section, we presented in the form of bullet points the main milestones of this research project.

- Development of the Spot Instance Management System
- Deployment of SuiteCRM Application simulating the critical business application as spot instance controlled by the SiMS and uncontrolled
- Deployment of Site24x7 Monitoring system and configuration of availability monitoring of Control System (not manged by SiMS) and system under the control of the SiMS
- Execution of Interruption simulation of the control system
- Execution of the sudden price change to force SiMS to act in case that price in the real world is stable for a number of days

3.2.2 Limitations

During the development of the SiMS, we have encountered some of the limitations that we have not been aware during the project proposal and design phase.

Lambda Function Limitation Lambda functions where one of the key elements to be used by the SiMS system to be fully serverless, unfortunately, certain limitations such as duration of the lambda function execution and filesystem restrictions forced us to change the use of Lambda to be just a trigger for tasks executed now as docker containers using AWS Fargate

Computation Time Data Analysis module is the one requiring a lot of computation power due to the size of of the dataset and combination of all instances types and number of all availability zones. To save computing time and costs associated with it, we have decided to limit data analysis and machine learning to only EU-CENTRAL-1 region and three selected types of instances c5.xlarge, t3.micro and t3.medium.

3.2.3 Spot Instance Management System (SiMS)

We have started our project by developing the Spot Instance Management System composed of four main modules

- The Data Collection Module
- The Data Pump Module
- The Data Analysis Module
- The Risk Assessment and Automation module

The prime idea behind the system is to have automated mechanism that with use of machine learning, can predict the price of the spot instance in next hour and act accordingly by migrating the affected spot instance to the next availability zone. In case when all availability zones would be labelled as High Risk, System will redeploy spot instance with new bid price larger by 20%

In the worst-case scenario, where there would be no spare capacity to spin the machine, the system would run the on-demand machine only for the time while the capacity is not available.

3.2.4 SLA Monitoring System

To gather availability data and later use this data for SLA calculation, we have configured a cloud-based monitoring system site24x7. This system, compared to others like OP5, can generate SLA reports for custom periods and not just for a month. It is offered in SaaS model, therefore, saving time and effort required for setting up the system.

3.3 SiMS: Data Collection Module

The Data Collection Module is responsible for downloading and aggregating historical data for EC2 Spot prices.

The process is started by the Time Event configured in AWS Cloudwatch. This time event will execute the Lambda Function responsible for triggering the AWS Fargate Task definition.

During the task execution, the docker image is pulled from the AWS Docker registry and started with entry point pointing to the data collection script written in Powershell.

During the script execution, the module is running a query using AWS Powershell modules and is collecting full last month of historical data of all regions and all instance types, during the collection this data is stored in the memory.

One the collection is completed, the content of the memory is saved to a CSV File called master.csv and saved to S3 Bucket sims-data-collection/Landing Zone.

In the last step, the trigger file is saved in trigger bucket on S3 to allow Data Pump Lambda function to execute data pump tasks.

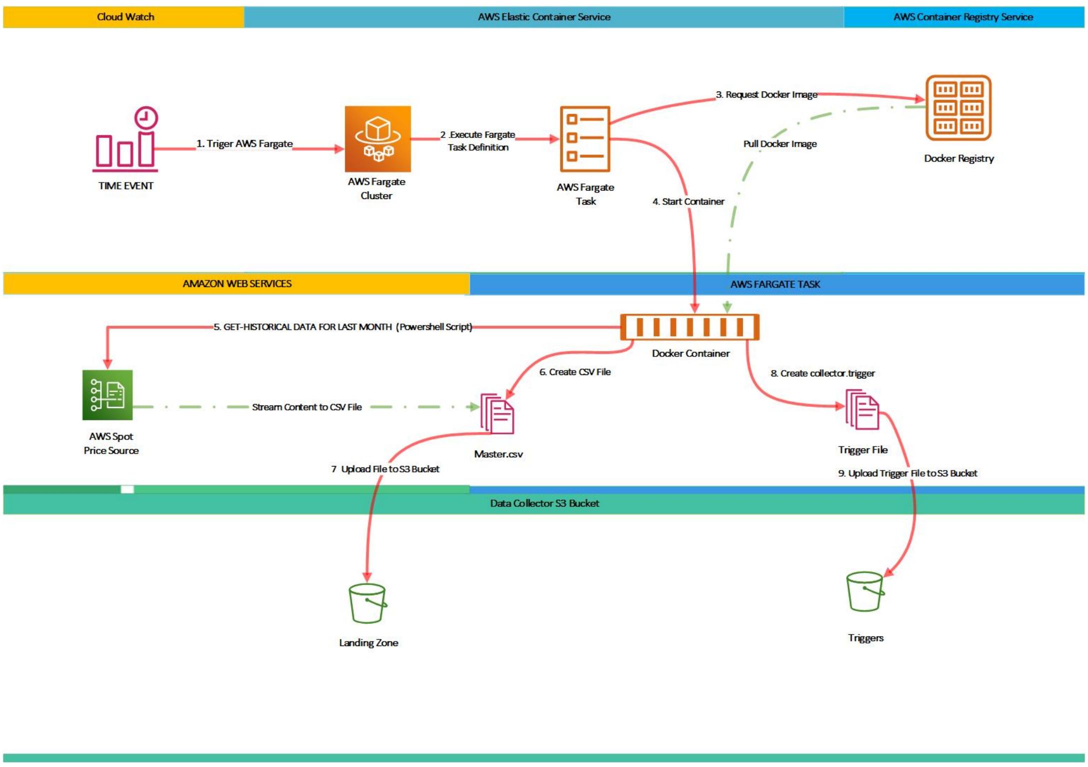


Figure 2: Data Collection Module

The module has been entirely developed in the Powershell using AWS Powershell modules and build using docker file and therefore delivered as a docker image.

Docker image is scheduled to be deployed on AWS Fargate service in EU-WEST-1 Region (Ireland) using Lambda Function and Cloud watch time event.

3.4 SIMS: Data Pump Module

The Data Pump module is small module mainly responsible for moving collected data from S3 Landing zone Bucket to S3 Staging Zone bucket after data collection, where later this data is used by Data Analysis Module for training the machine learning algorithm.

Launch mechanism is almost the same as in case of the data collector, in case of data pump, S3 put file event is triggering the Lambda function executing data pump task definition in AWS Fargate.

Once the docker container is started, the PowerShell script responsible for moving data from the landing zone to the staging zone is executed.

During the execution, the master.csv file is fetched from data collection landing zone s3 bucket and moved to stage zone in the same data collection bucket. This operation is required to avoid any potential issues if the master file was presented in a landing zone while the data collection is running.

Once the execution is completed, the trigger file is saved in trigger s3 bucket to allow the data analysis module to start.

The Data pump is executed by Lambda Function triggered by trigger file saved by the Data Collection module in S3 Triggers bucket which causes S3 New file Cloud watch event

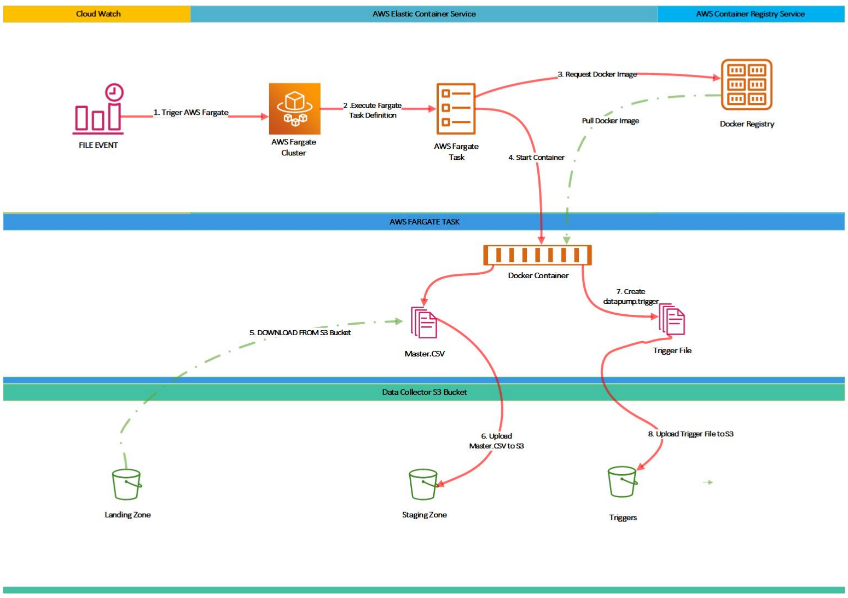


Figure 3: Data Pump Module

The module has been entirely developed in the Powershell using AWS Powershell modules and build using docker file and therefore delivered as a docker image.

3.5 SIMS: Data Analysis Module

The Data Analysis module is one of two key components of the SiMS system. Data Analysis module is responsible for analyzing historical data gathered by the data collection module and then is responsible for applying the machine learning model based on random forest regression

Data Analysis execution is triggered by the trigger file created after execution of the data pump. Execution is started via Amazon Lambda function that is responsible for starting the AWS Fargate Task.

In the Fargate task definition, three docker containers are defined. Each container is responsible for data analysis for configured availability zone. Since we have focused our efforts on eu-central-1 region, therefore docker containers are running the data analysis for Availability zones eu-central-1, eu-central-1b and eu-central-1c simultaneously.

During the Data analysis process data set is downloaded from the S3 bucket and analyzed using random forest regression, results in the form of price predictions for the next 48 hours are saved into DynamoDB Table.

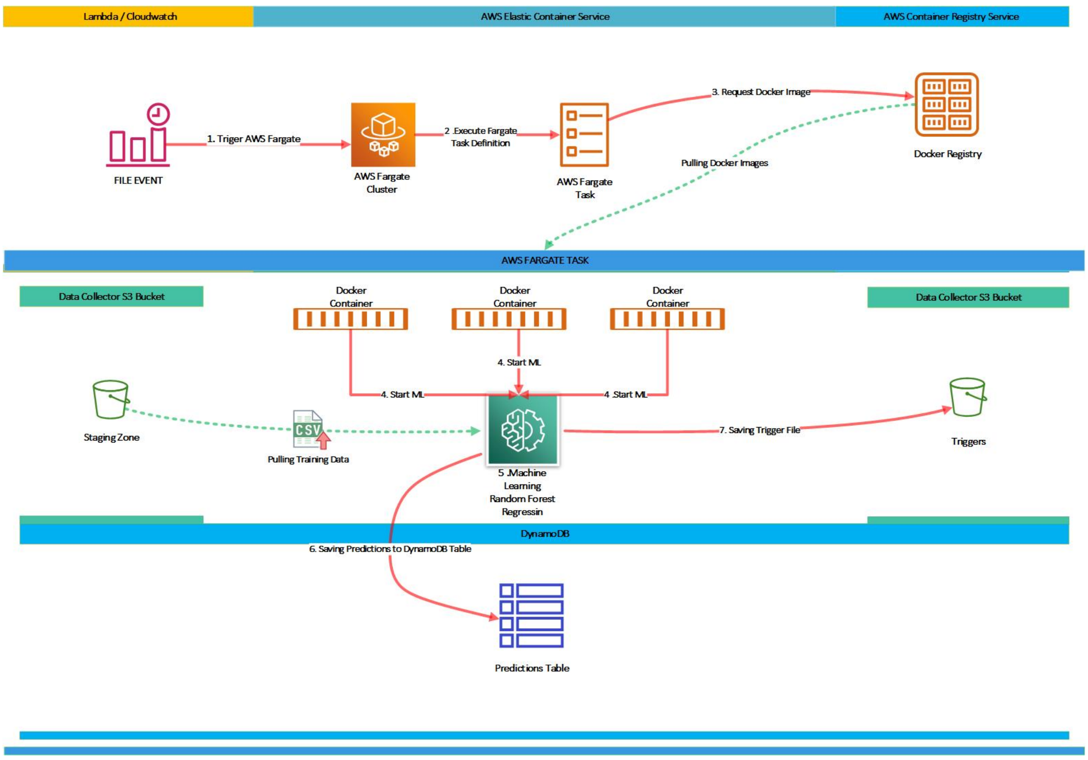


Figure 4: Data Analysis Module

The module has been entirely developed in the Python and build using docker file and therefore delivered as a docker image.

3.6 SIMS: Risk and Automation Module

The risk and automation module is one of two key modules of the SiMS solution. The module consists of two primary submodules, namely the risk engine and the automation engine.

This module is responsible for risk analysis of the instance interruption in each of availability zones in the configured region and next for taking the appropriate actions concerning the level of the risk.

All components of the module have been developed in Python and are delivered as one risk and automation docker container.

3.6.1 The Risk Engine

The risk engine is the part of the risk and automation module responsible for the calculation of the risk of instance interruption in each of the availability zones. The risk is calculated based on the maximum bid price, current price, predicted price and the threshold for each of the risks level LOW, MEDIUM, HIGH.

The risk is calculated for each of the availability zones in a given region with the following formula:

if Current price (Cp) is smaller than 55% of the Maximum Bid price (Mb) and Prediction price(Pp) is smaller than Maximum bid (Mb) the risk is calculated as LOW

$$Lr = Cp < \left(\frac{55}{100}x\ Mb\right)\ AND\ \ (Pp <\ Mb) \tag{1}$$

If Current Price (Cp) is larger than 55% and lower or equal 80% of Maximum bid price (Mb) and Prediction price (Pp) is smaller than Maximum bid (Mb) the risk is calculated as MEDIUM

$$Mr = Cp > \left(\frac{55}{100}x\ Mb\right)\ AND\ Cp < \left(\frac{8}{10}x\ Mb\right)\ AND\ \ (Pp <\ Mb) \tag{2}$$

If the current price (Cp) is larger than 80% of the maximum bid price (Mb) and prediction price is smaller than the maximum bid price risk is calculated as High

$$Hr = Cp > \left(\frac{8}{10}x\ Mb\right)\ \ AND\ \ (Pp <\ Mb) \tag{3}$$

If the case when prediction price (Pp) is larger than the maximum bid price risk is calculated as High.

$$Hr =\ Pp > Mb \tag{4}$$

3.6.2 The Automation Engine

The Automation Engine is responsible for the interruption mitigation actions based on the risk assessment for every availability zone in the region.

During the execution, the Automation Engine is comparing interaction risk level of each of the managed spot instances in their respective location to the risk assessment of other two availability zones. In case that current risk is assessed as LOW, there is no action taken.

In case that current risk is Medium and there is at least one Availability zone with LOW-Risk assessment than migration operation is starting to that zone. If there is no LOW zone, there is no action taken.

In case that current risk is HIGH and there is LOW or Medium Zone available than the migration process will move the instance to the lowest risk zone.

In case if the risk is HIGH in all availability zones, the machine is redeployed by automation engine to the same availability zone but with 20% higher maximum bid price.

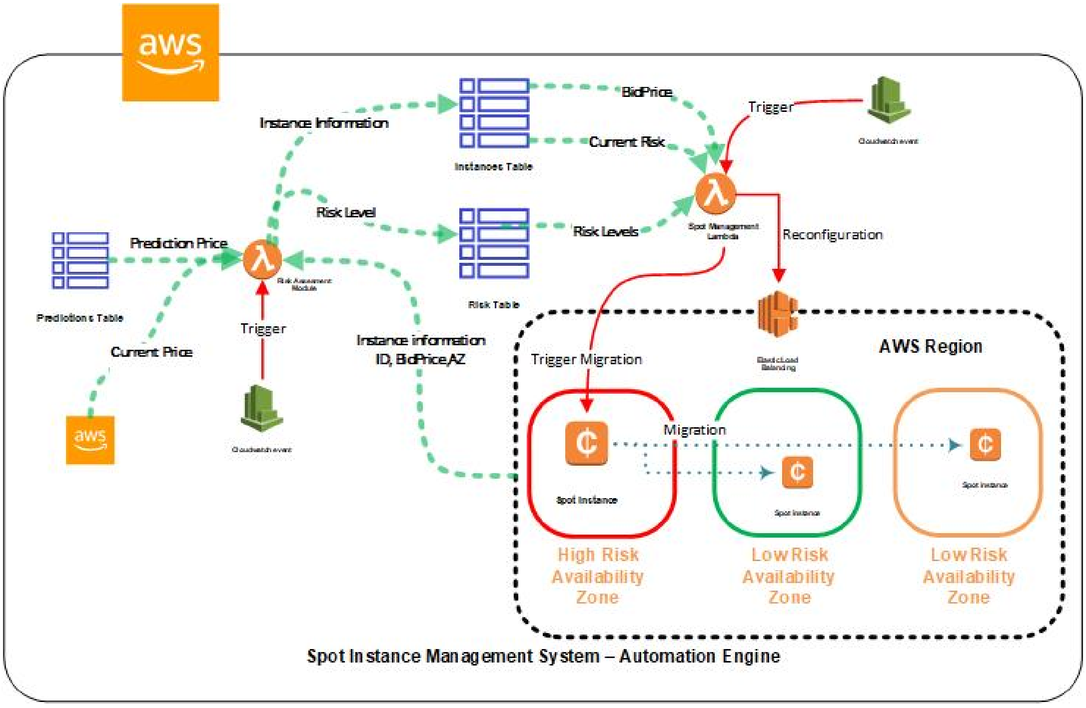


Figure 5: Risk and Automation Module

3.7 Simulation

For the simulation purposes, we have created a python script that during execution, would execute the following steps :

- Change Predicted price for next full hour to one of the values selected randomly on every execution (OnDemand price, predicted price, maximum bid price, current price, mean of all above)

- Execute Risk Recalculation

After risk is recalculated, we would rely on the automation module to evaluate and execute necessary actions against SimS Manged instance.

For our not managed instance (the control instance) script will execute the following steps:

- Evaluate if Maximum bid price is lower than the predicted price
- If the predicted price is higher, terminate instance after 2 minutes

At this stage, the script would simulate two minutes notification given by Amazon. After two minutes, the instance is terminated.

Based on our first experiment (described in the next section), we have updated simulation script to simulate the engineer acting on the incident created by the monitoring system.

We have calculated that manual intervention required to bring the system back on-line would take approx. Four minutes and 45 seconds.

Therefore part of the script simulating the engineering work is taking to account two essential values :

- The time between interruption and incident in the Monitoring system
- The time required to bring system on-line

4 Implementation

This chapter is describing our implementation of the SimS Systems and all inputs and outputs of each model. We have started our implementation with setup of the cloud environment using AWS Cloud in region eu-west -1 (Ireland)

4.1 Cloud Setup

4.1.1 Simple Storage Service

Data Collection and Data Analysis are heavily depending on the CSV files generated by each of the modules. Those files need to be stored in accessible and reliable storage; therefore, we have configured an S3 bucket called SIMS-Data-Collection.

The data collection bucket contains three folders:

- **Landing Zone:** Where CSV File generated by Data Collection is stored
- **Staging Store :** Where thanks to data pump files are moved from the Landing zone and later accessed by the Data Analysis Module

- **Trigers:** Where trigger files at the end of the execution of data collection and data pump are stored.

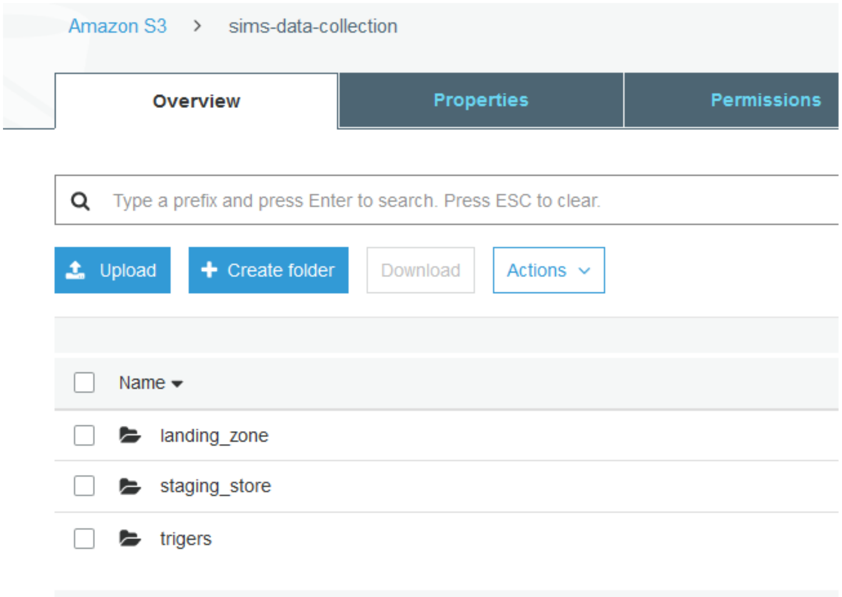


Figure 6: Data Collection S3 Bucket

4.1.2 DynamoDB

The DynamoDB is a crucial component used mainly by risk and automation module and as a data analysis outputs.

- **Project:** Table containing predicted Spot price
- **Risk:** Table containing calculated risk for each availability zone and used type of spot instances
- **Instances:** Table containing information such as current price, bid price, OS Type and Instance ID of available or past instances

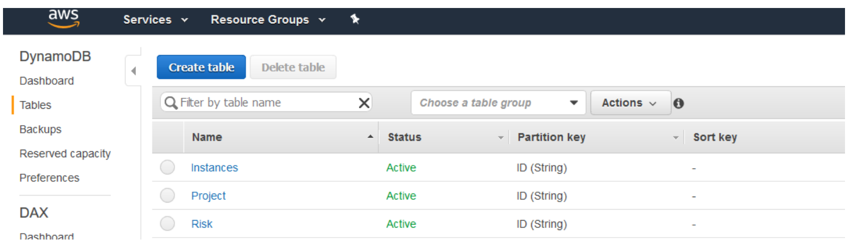


Figure 7: DynamoDB Tables

4.1.3 AWS Elastic Container Service - Fargate

Initially, the SimS was planned as completely serverless solutions, and unfortunately, due to limitations we described in the previous chapter, we had to adapt to semi-serverless by using AWS Fargate.

The Fargate Service is the docker container execution service not requiring underlying docker infrastructure. We have defined tasks for each of the modules using the build-in task definition.

Following tasks definitions have been configured:

- SimS Data Collection
- SimS Data Analysis
- SimS Automation
- SimS Risk Analysis
- SimS Simulation

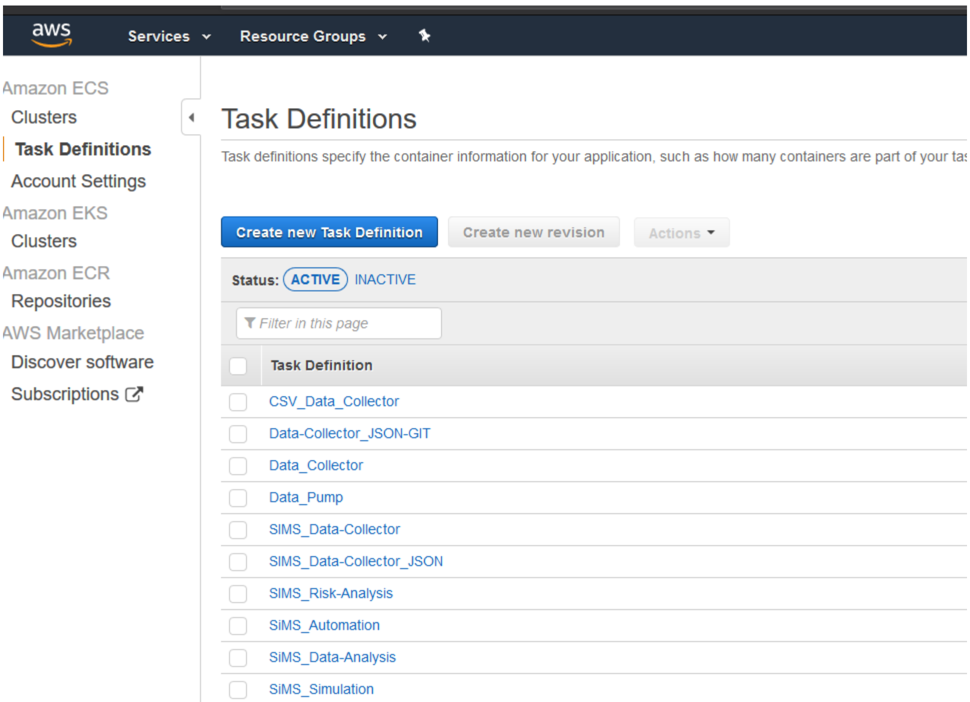


Figure 8: Fargate Task Definitions

In the scope of AWS ECS, we have also configured Docker Image Repositire using AWS Elastic Container Registry. The registry was automatically updated via CI/CD pipeline configured in GITLAB.

Following repositories have been configured :

- data-collector
- data-analysis
- data-pump
- risk-and-automation

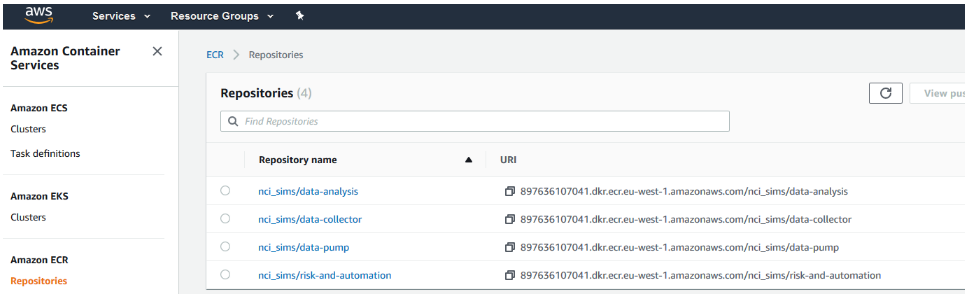


Figure 9: Amazon Docker Repositories

4.1.4 AWS Lambda

As mentioned in previous sections, we have planned for SimS to be based entirely on AWS Lambda functions, but due to its limitations, we had to adopt a new approach. Instead of relying on lambda to execute full code, we have decided to use lambda to execute triggering python code and execute AWS Fargate Tasks on a defined schedule.

We have configured the following functions:

- SIMS_Start_dataCollection
- SIMS-Start_DataPump
- SIMS_Risk-Analysis
- SIMS_Data-Analysis
- SIMS_Simulation

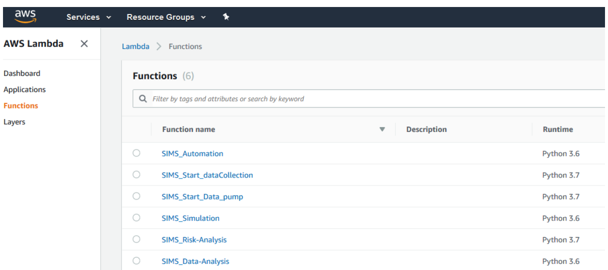


Figure 10: Lambda Functions

4.1.5 AWS Cloud Watch

AWS Cloud Watch triggering rules have been configured in conjunction with the AWS Lambda. We required a time trigger and S3 file trigger to be configured to execute required lambda functions. We have configured four time-triggers and two (using lambda interface) file events.

4.1.6 Route 53 DNS and Elastic IP

As part of the implementation, we have registered a domain name **EIF.SYSTEMS** using the amazon DNS service Route53 and Elastic IPs.

In DNS we have pointed the A records of DNS Server to our two test subjects (described below).

- **ba.eif.systems** - for our test subject fully controlled by our solution.
- **control.eif.systems** - for our control machine, a machine that is not controlled by our solution system

4.2 Test subject implementation

In order to have separate infrastructures between the SIMS system and test subjects, we have used the **eu-central-1** region for the two test machines running CRM software simulating business-critical application.

We have deployed two spot instances using instance type **t3.medium** and **SuitCRM** Amazon image from Amazon Marketplace.

Each of the test subjects had a list of mandatory tags associated with it to tell SimS system which of the two instances is the control instance and which is the one managed by the SimS System

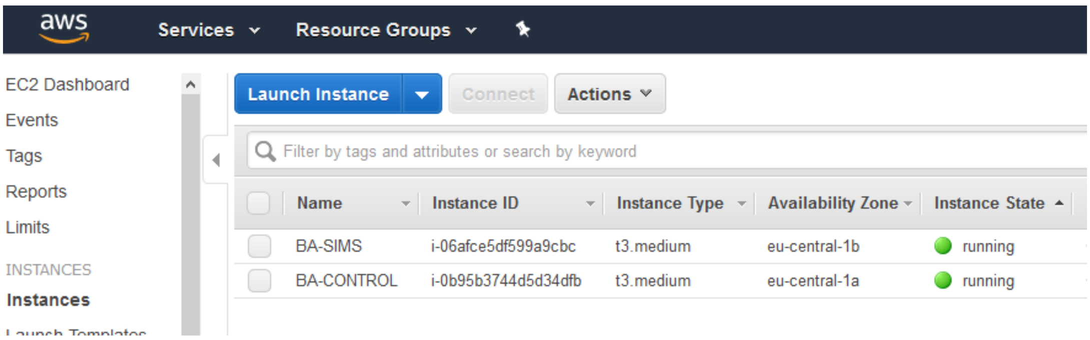


Figure 11: Test Machines running in eu-central-1

4.3 The Monitoring System

To gather availability data, we needed to find a monitoring system that would be independent of our solution and would provide SLA type reports and SLA Configuration.

We have found SaaS offering of Site24x7 brought by Zoho Corporation.

In the site24x7, we have configured monitoring for the websites that are checking in one minute's intervals connectivity to HTTP ports of defined targets, its response time, DNS response time and general availability by ping.

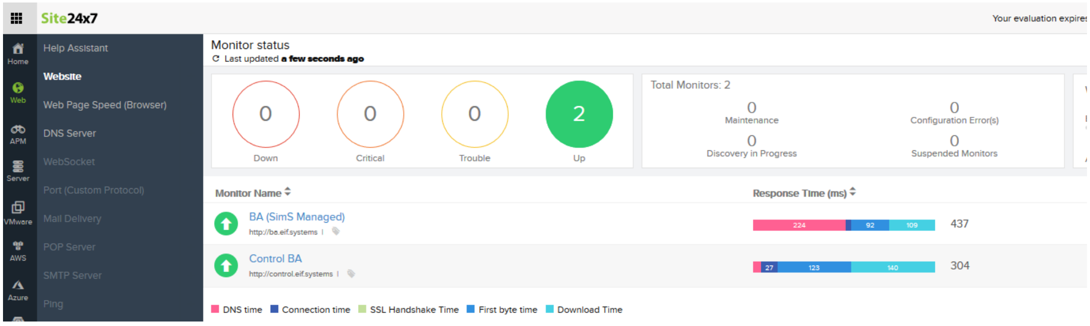


Figure 12: Site24x7 Monitoring System Dashboard

4.4 SimS System Inputs and Outputs

4.4.1 Data Collection Module

- **Input**: JSON Data gathered from Amazon
- **Output** : master.csv saved to S3 Bucket

4.4.2 Data Pump

- **Input**: master.csv from Data Collection Landing Zone S3
- **Output**: master.csv in Data Colelction Staging Store S3

4.4.3 Data Analysis

- **Input**: Master.csv file from Data Collection Staging Store S3
- **Output**: data in Project DynamoDB Table

4.4.4 Risk Engine

- **Input**: Prediction Price from Project Table, current spot price from Amazon and Maximum bid price from currently running instances
- **Output**: Risk Levels saved in Risk DynamoDB Data and Instance information saved to Instances Table in DynamoDB

4.4.5 Automation Engine

Input Risk Level from Risk dynamo DB table, Instance information from Instances DynamoDB

Output During the migration execution, the following outputs are generated :

- New Temporary Amazon Image (AMI) created based on the current instance
- New Instance deployed to target availability zone based on the created image
- Old Instance Data removal from DynamoDB Instances Table.

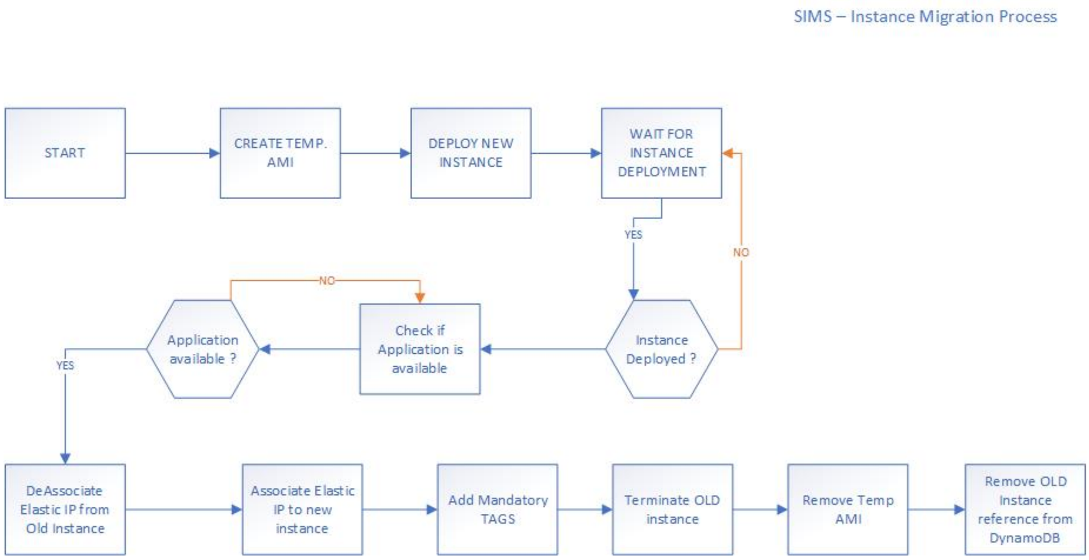


Figure 13: Instance Migration Process

5 Evaluation

5.1 Interuption of Control System

- Scenario: Classic Termination of EC2 Spot Instance
- Actors: Simulator
- Desired Outcome: System Terminated and restored

5.1.1 Test Steps

Simulation script to terminate EC2 Instance with 2 minutes delay simulating Amazon Notification grace period and later restoring the service simulating the system engineer.

5.1.2 Results

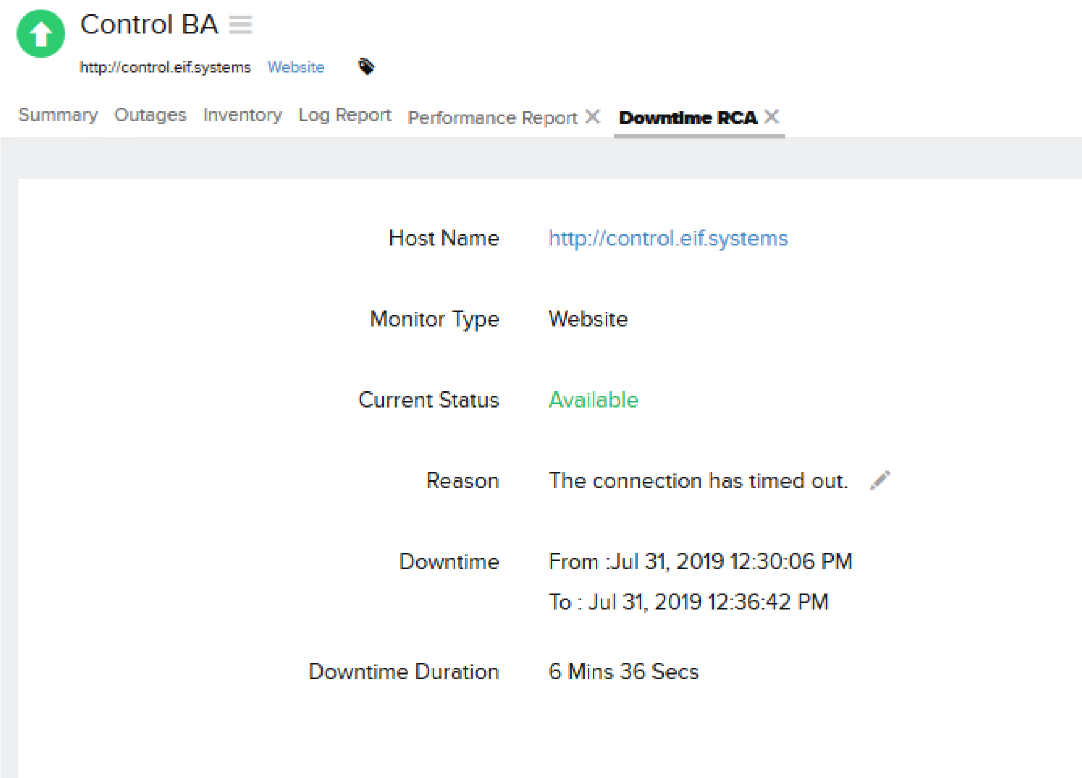


Figure 14: Root Cause Analysis Report from Monitoring System

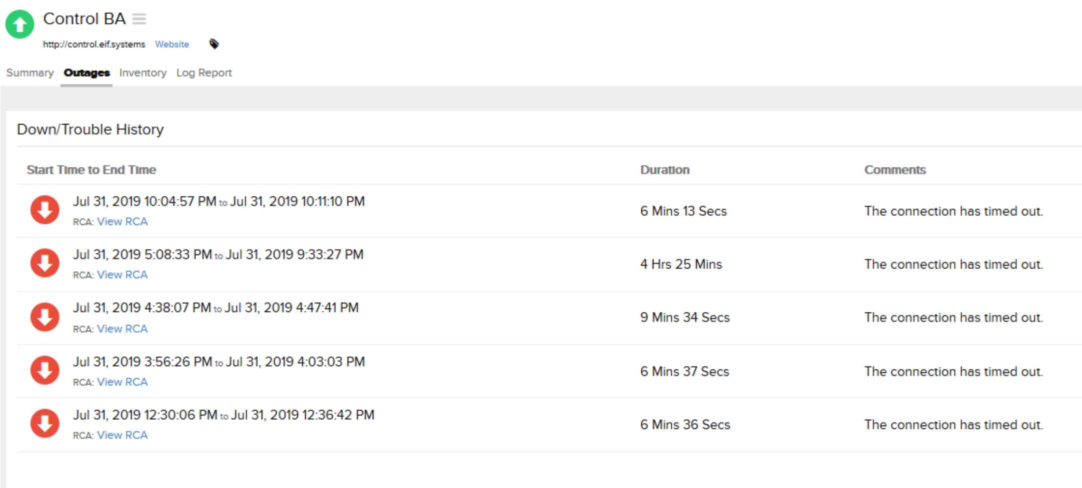


Figure 15: Outage reports of control system

The control system during the experiment has been terminated a number of times causing reported outage and unavailability of the application.

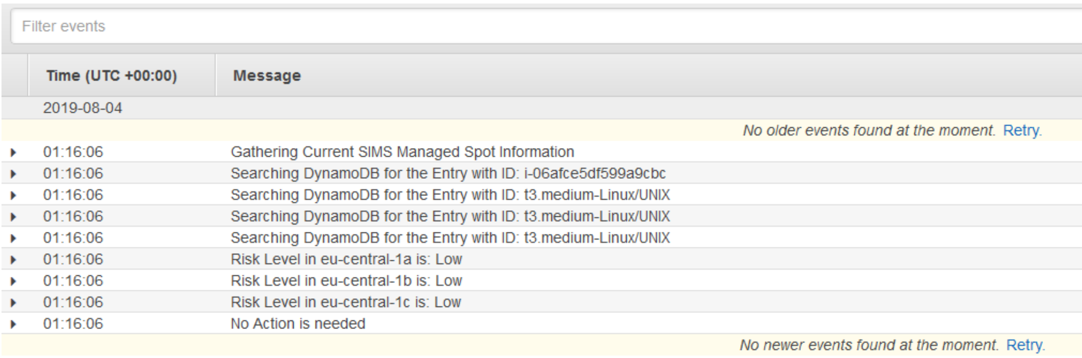
5.2 SimS Automated Actions LOW Risk

- Scenario: Risk Level: LOW LOW LOW
- Actors: SimS System
- Desired Outcome: No migration initiated

5.2.1 Test Steps

Automated Execution of Automation Module

5.2.2 Results


Filter events

Time (UTC +00:00)	Message
2019-08-04	
	No older events found at the moment. Retry.
01:16:06	Gathering Current SIMS Managed Spot Information
01:16:06	Searching DynamoDB for the Entry with ID: i-06afce5df599a9cbc
01:16:06	Searching DynamoDB for the Entry with ID: t3.medium-Linux/UNIX
01:16:06	Searching DynamoDB for the Entry with ID: t3.medium-Linux/UNIX
01:16:06	Searching DynamoDB for the Entry with ID: t3.medium-Linux/UNIX
01:16:06	Risk Level in eu-central-1a is: Low
01:16:06	Risk Level in eu-central-1b is: Low
01:16:06	Risk Level in eu-central-1c is: Low
01:16:06	No Action is needed
	No newer events found at the moment. Retry.

Figure 16: Cloud Watch Log representing No Actions

Expected behaviour, if all availability zones are LOW risk, is not to take action. As presented above, the automation module evaluated the risk and did not perform any actions.

5.3 SimS Automated Actions High Risk

- Scenario: Risk Level: High High High
- Actors: SimS System
- Desired Outcome: Instance Redeployed with higher bid price, no system downtime reported by monitoring system

5.3.1 Test Steps

During the execution, the Automation Module will evaluate risk in all availability zone and based on the result will move the instance to another availability zone or redeploy to current with the higher bid price.

5.3.2 Results

Filter events

Time (UTC +00:00)	Message
2019-08-03	
16:52:05	Searching DynamoDB for the Entry with ID: t3.medium-Linux/UNIX
16:52:05	Searching DynamoDB for the Entry with ID: t3.medium-Linux/UNIX
16:52:05	Checking if there is LOW Risk AZ Available
16:52:05	Adding 50% to Maximum Bid Price
16:52:05	Searching DynamoDB for the Entry with ID: i-094ea2c8903d23bc3
16:52:05	Redeployment of the Instance with new BidPrice
16:52:05	Creating Image
16:52:05	Preparation : Creating IMAGE.
16:52:05	Preparation : Image Created
16:52:05	Deployment: Getting current Bid price for the instance id i-094ea2c8903d23bc3
16:52:05	Searching DynamoDB for the Entry with ID: i-094ea2c8903d23bc3
16:52:05	Deployment: Preparing move of the instance to new availability zone eu-central-1b
16:52:05	Deployment: Spot Request Created. Waiting for fulfillemend
16:52:05	Deployment: Instance Deployed to new availability Zone eu-central-1b
16:52:05	Deployment: New Instance Deployed in target AZ with id i-06afce5df599a9cbc
16:52:05	Deployment: Waiting for Instance to Complete the Boot
16:52:05	Configuration: Switching Traffic from old to new instance
16:52:05	Configuration: Reassining IP Address 35.157.195.247 from old instnace to new
16:52:05	Configuration: Setting SIMS Tags on instance i-06afce5df599a9cbc
16:52:05	Name BA-SIMS
16:52:05	domain eif.systems
16:52:05	SIMS_Managed True
16:52:05	FQDN ba.eif.systems
16:52:05	Cleanup: Terminating old Instance i-094ea2c8903d23bc3
16:52:05	Cleanup: Removing Image ami-0c35a61c510d42afb
16:52:05	Clenup:Image ami-0c35a61c510d42afb removed
16:52:05	Clenup: Removing old instanceID i-094ea2c8903d23bc3 from DynamoDB reference Table

Figure 17: Redeployment of Instance with new Bid Price

In this scenario, The Sims System detected that all availability zones in the region are High Risk. In this situation, the system is designed to redeploy the instance with a higher bid price to mitigate the high risk of interruption and therefore to maintain desired availability level by setting up (migrating new instance) and reassigning the elastic IP from Risky instance to newly deployed one, therefore allowing traffic to reach the system without any issue.

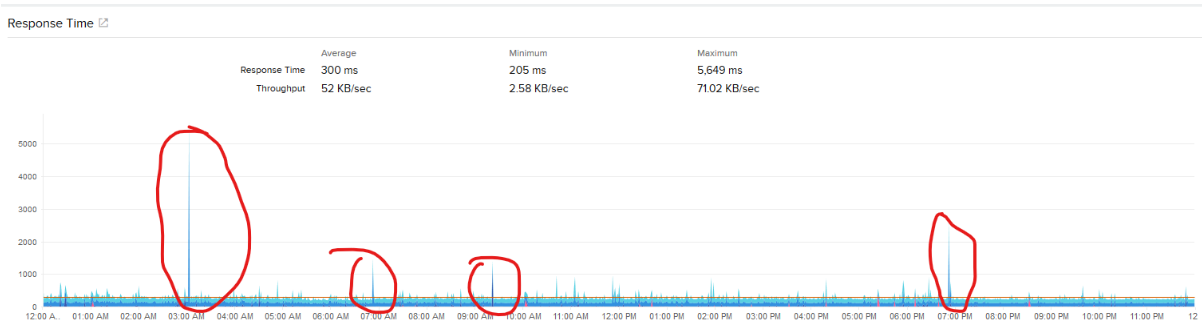


Figure 18: Response time monitoring of the System

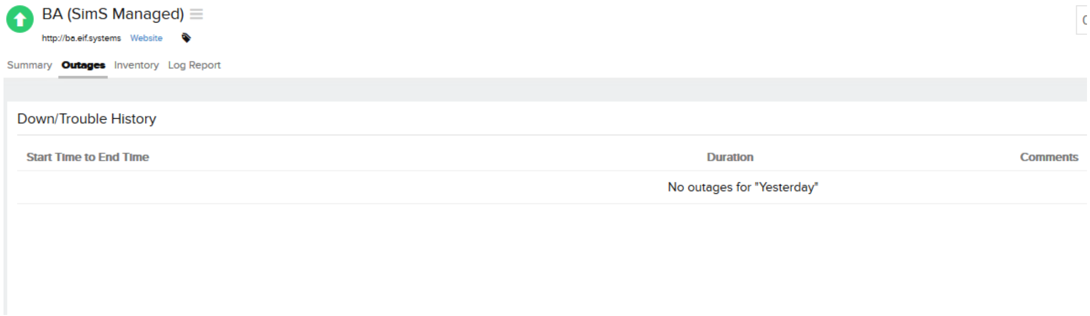


Figure 19: No outage reported by the system

Monitoring system did not report system outage, the only indication of migration is a short spike in response time

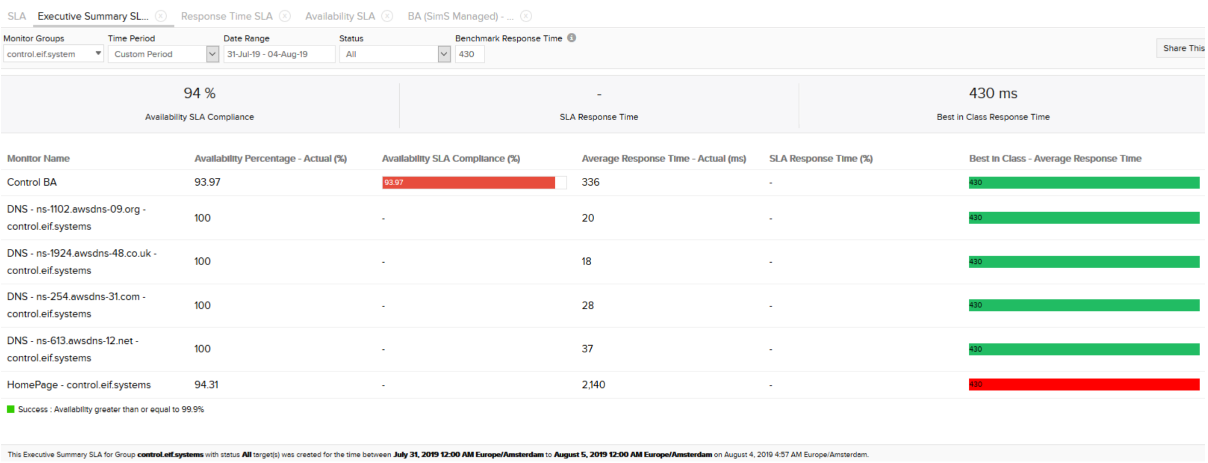
5.4 Five Days SLA Monitoring

- Scenario: 5 Days SLA Monitoring
- Desired Outcome: SLA Level of 99.9%

5.4.1 Test Steps

Simulation script has been scheduled and running in four hours intervals affecting the classic EC2 Spot instance as well as risk analysis data for the SimS System. Due to random price selection, exact time and date of the next interruption were not known.

5.4.2 Results



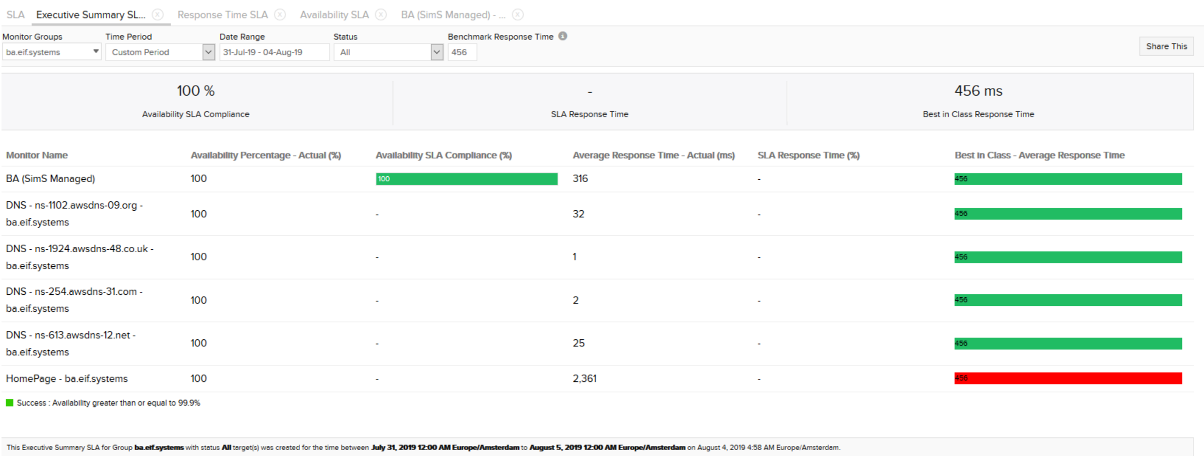
Monitor Name	Availability Percentage - Actual (%)	Availability SLA Compliance (%)	Average Response Time - Actual (ms)	SLA Response Time (%)	Best in Class - Average Response Time
Control BA	93.97	93.97	336	-	430
DNS - ns-1102.awsdns-09.org - control.eif.systems	100	-	20	-	430
DNS - ns-1924.awsdns-48.co.uk - control.eif.systems	100	-	18	-	430
DNS - ns-254.awsdns-31.com - control.eif.systems	100	-	28	-	430
DNS - ns-613.awsdns-12.net - control.eif.systems	100	-	37	-	430
HomePage - control.eif.systems	94.31	-	2,140	-	430

Success : Availability greater than or equal to 99.9%

This Executive Summary SLA for Group **control.eif.systems** with status **All** target(s) was created for the time between **July 31, 2019 12:00 AM Europe/Amsterdam** to **August 5, 2019 12:00 AM Europe/Amsterdam** on August 4, 2019 4:57 AM Europe/Amsterdam.

Figure 20: 5 Days SLA Report for Control System

Figure 21: 5 Days availability report for Control System



Monitor Name	Availability Percentage - Actual (%)	Availability SLA Compliance (%)	Average Response Time - Actual (ms)	SLA Response Time (%)	Best in Class - Average Response Time
BA (SimS Managed)	100	100	316	-	456
DNS - ns-1102.awsdns-09.org - ba.eif.systems	100	-	32	-	456
DNS - ns-1924.awsdns-48.co.uk - ba.eif.systems	100	-	1	-	456
DNS - ns-254.awsdns-31.com - ba.eif.systems	100	-	2	-	456
DNS - ns-613.awsdns-12.net - ba.eif.systems	100	-	25	-	456
HomePage - ba.eif.systems	100	-	2,361	-	456

Success : Availability greater than or equal to 99.9%

This Executive Summary SLA for Group **ba.eif.systems** with status **All** target(s) was created for the time between **July 31, 2019 12:00 AM Europe/Amsterdam** to **August 5, 2019 12:00 AM Europe/Amsterdam** on August 4, 2019 4:58 AM Europe/Amsterdam.

Figure 22: 5 Days SLA Report for SiMS Managed System

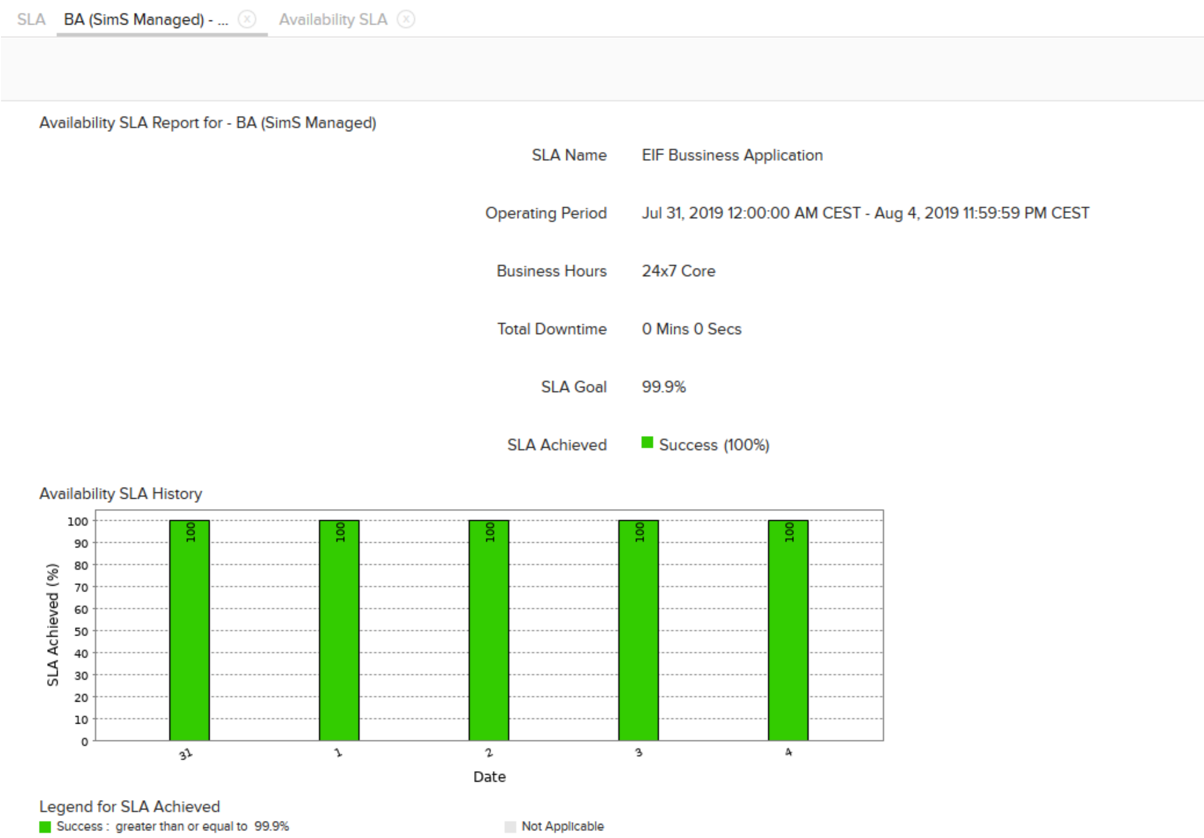


Figure 23: 5 Days availability report for SimS Managed System

5.5 Discussion

The results are showing that usage of a proactive system that is managing spot instances can prove to be valuable if our main goals are low costs and availability of the system using the spot instances.

We can see that if automation is designed to act while a risky instance is still running, automated switch over is almost seamless but at the price of a drop in performance during the migration of the instance.

For applications that are very response time-sensitive this could be still the issue, as well as for the application where data is written continuously as during the migration the checkpoint (image) of the machine is created and any data written in the time between image creation and switch over of the traffic to the newly deployed machine would be lost.

The question in here would be the trade-off, in case of classic spot instance customer can face unexpected termination and if they do not have any solution that would periodically save the data from that instance while running they could face complete data loss in compering to few seconds of loss in case of SimS Managed System, therefore automated system as one presented in this research can significantly expand possibilities of the application of spot instances as well as can help with reduction of operational costs

The Spot Instance Management system for its risk analysis is using Random Forest regression-based machine learning model, while this model during our research proved sufficient, the model itself was not a part of in-depth testing and therefore could be not as accurate as we could hope. Necessary comparison testing showed us that price predicted are the same as current prices or difference in price is minimal.

Our prediction machine learning model requires a lot of computing power, to calculate price predictions for 150 types of EC2 spot instances for every availability zone in EU Region would require over 26 hours running in 4CPU docker containers provided by the Amazon. Even with those limitations, we have been able to prove that smart, proactive system that can move around spot instances based on the predicted price and therefore risk calculated from it can be effective in achieving not only 99.9% availability but even 100%.

In our research question we are asking How can we assure SLA Level of 99.9% for service running on EC2 Spot instance, as mentioned above and shown in the evidence The SimS System is able to effectively managed spot instances and keep up the availability on the desired level and achieve required three nines SLA thanks to its Risk assessment mechanism and proactive actions before any terminate can happen.

With automatic bid rise in case of High risk in all availability zone at the current stage, this could lead to higher than desired bid price and therefore not always keeping cost-effectiveness.

6 Conclusion and Future Work

In our research question, we are asking How can we assure SLA Level of 99.9% for service running, In our paper, we show the Spot instance Management system in principle can manage spot instances effectively to keep up availability level on the 99.9%

A system like SimS to be effective requires a good and reliable machine learning module to predict spot price in next hour accurately. In hour five-day test we have simulated number of interruptions of classic ec2 spot instance, and we put this together with the change of the risk level in availability zones to force the SimS system to act and migrate affected machine if are located in a high availability zone. Migration is using image creation (checkpointing).

Currently, due to the limitation in computing power and desire to run the system as serverless as possible the support for more just a subset of instance is limited by the computation power of AWS Fargate docker containers.

Future researchers could take this solution step further by selecting more sophisticated machine learning model that would take into account not only the historical data but also seasonality, and it would perform better, therefore, allowing support for a more significant number of instances and availability zones.

For limitations regarding potential data loss during the migration process and switch over, further research could be done in the field of vMotion style migration where performance nor data is affected.

References

Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A. and Tsafrir, D. (2013). Deconstructing amazon ec2 spot instance pricing, *ACM Transactions on Economics and Computation* **1**(3): 16.

Amekraz, Z. and Youssef, M. (2016). Prediction of amazon spot price based on chaos theory using anfis model, *Computer Systems and Applications (AICCSA), 2016 IEEE/ACS 13th International Conference of*, IEEE, pp. 1–6.

Andrzejak, A., Kondo, D. and Yi, S. (2010). Decision model for cloud computing under sla constraints.

Arévalos, S., López-Pires, F. and Baran, B. (2016). A comparative evaluation of algorithms for auction-based cloud pricing prediction, *2016 IEEE International Conference on Cloud Engineering (IC2E)*, IEEE, pp. 99–108.

Chhetri, M. B., Lumpe, M., Vo, Q. B. and Kowalczyk, R. (2017). On estimating bids for amazon ec2 spot instances using time series forecasting, *Services Computing (SCC), 2017 IEEE International Conference on*, IEEE, pp. 44–51.

Huang, K., Gao, X., Zhang, F. and Xiao, J. (2017). Coms: Customer oriented migration service, *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 692–695.

Jangjaimon, I. and Tzeng, N.-F. (2015). Effective cost reduction for elastic clouds under spot instance pricing through adaptive checkpointing, *IEEE Transactions on Computers* **64**(2): 396–409.

Kumar, D., Baranwal, G., Raza, Z. and Vidyarthi, D. (2017). A survey on spot pricing in cloud computing, *Journal of Network and Systems Management* **26**.

Li, Z., Kihl, M. and Robertsson, A. (2015). On a feedback control-based mechanism of bidding for cloud spot service, *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 290–297.

Li, Z., Zhang, H., OBrien, L., Jiang, S., Zhou, Y., Kihl, M. and Ranjan, R. (2016). Spot pricing in the cloud ecosystem: A comparative investigation, *Journal of Systems and Software* **114**: 1 – 19.
URL: *http://www.sciencedirect.com/science/article/pii/S0164121215002332*

Neto, J. P. A., Pianto, D. M. and Ralha, C. G. (2018). A prediction approach to define checkpoint intervals in spot instances, *International Conference on Cloud Computing*, Springer, pp. 84–93.

Nicolae, B. and Cappello, F. (2013). Blobcr: Virtual disk based checkpoint-restart for hpc applications on iaas clouds, *Journal of Parallel and Distributed Computing* **73**(5): 698–711.

Singh, V. K. and Dutta, K. (2015). Dynamic price prediction for amazon spot instances, *2015 48th Hawaii International Conference on System Sciences (HICSS)*, IEEE, pp. 1513–1520.

Voorsluys, W. and Buyya, R. (2012). Reliable provisioning of spot instances for compute-intensive applications, *Advanced information networking and applications (aina), 2012 ieee 26th international conference on*, IEEE, pp. 542–549.

Wallace, R. M., Turchenko, V., Sheikhalishahi, M., Turchenko, I., Shults, V., Vazquez-Poletti, J. L. and Grandinetti, L. (2013). Applications of neural-based spot market prediction for cloud computing, *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013 IEEE 7th International Conference on*, Vol. 2, IEEE, pp. 710–716.

Zhao, H., Pan, M., Liu, X., Li, X. and Fang, Y. (2015). Exploring fine-grained resource rental planning in cloud computing, *IEEE Transactions on Cloud Computing* **3**(3): 304–317.

Configuration Manual

MSc Research Project
Cloud Computing

Jerry Danysz
Student ID: 16148690

School of Computing
National College of Ireland

Supervisor: Victor Del Rosal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	JERRY DANYSZ
Student ID:	16148690
Programme:	MSc Cloud Computing **Year:** 2
Module:	RESEARCH PROJECT
Lecturer:	Victor Del Rosal
Submission Due Date:	12/08/2019
Project Title:	Spot Instance Management System Configuration Manual
Word Count:2065	**Page Count: 24**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ………

Date: ………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	

Penalty Applied (if applicable):	

Configuration Manual

Jerry Danysz
Student ID: 16148690

1 Introduction

1.1 Purpose of this document

This Configuration Manual is based on the NCI Research project requirements described in the Project Handbook. Main purpose of this document is to describe the required software tools and settings in order to successfully deploy the Spot instance Management System

1.2 Document Structure

Section	Purpose
General Information and Prerequisites	This chapter describes general information and prerequisites needed for the installation of the system
Development Environment Requirements	This chapter describes steps required for successful setup of development environment used for development and update of the solution
Solution Deployment Procedure	This chapter describes step by step procedure of deployment of the SiMS Solution
Validation	This chapter describes steps required to validate successful deployment of the solution

2 General Information

2.1 Objectives

The main objective of Spot Instance Management System is to implement platform for effective and automatic management of AWS Spot Instances and to prevent unplanned interruptions of services depending on those services by usage of machine learning for the prediction of next hour spot price and automation for mitigation of the high risk of interruption by migration of potentially risky instance to another region where the interruption of the risk is lower.

2.2 Solution Summary

The Spot Instance Management System solution consists of four separate modules interlinked and acting as one ecosystem.
The Data Gathering module is responsible for gathering historical Spot price data for past month and saving it to AWS S3 bucket for later consumption by other modules.

The Data Pump module is the storage (s3) management module that is responsible for moving gathered data to respective folders in S3 Buckets.

The Data Analysis module is responsible for analysing gathered data by application of machine learning algorithms (Random Forest Regression) and resulting with two days of spot instance pricing per every hour

The Risk and Automation module is responsible for the risk assessment of the availability zones where current spot instances are residing and if there is a need (high risk) executing automated mitigation actions such as migration to lower risk zone or redeployment of instance with higher maximum bid price.
Automation module during the migration will also reconfigure additional services such as elastic IP

2.3 Architectural Requirements

In this section we describe base architectural requirement.

2.3.1 Cloud Platform

The Spot Instance Management System is the cloud management platform designed to explicitly manage Amazon Web Services Spot Instance; therefore, it is required that solution is deployed in Amazon Web Services Cloud.

2.3.2 Docker Environment

Solution is designed to run independent scripts in Docker containers for each of the modules.
It is required to use Elastic Container Service provided by the Amazon Web Services.
Use of the AWS Faregate is recommended but other offering in scope of ECS such as EC2 Container cluster or AWS Managed Kubernetes could be also accepted but at this current stage solution was not tested on those services.

2.3.3 Simple Storage Service (S3)

Simple storage service (S3) provided by the Amazon is required for the data gathering and data pump operations and are used to store trigger files therefore properly configured S3 Buckets have to be deployed.

2.3.4 Cloud Watch Monitoring

Time events configure in the Cloud Watch service provided by Amazon are required in order to trigger and execute different modules operations as given time.
Output logs of the each of the operations are automatically stored in Log Trail of Cloud Watch.

2.3.5 AWS Lambda

AWS Lambda functions written in Python 3.6/3.7 are responsible for executing AWS Faregate tasks definitions connected to each of the solution’s modules.

2.4 Required Skills

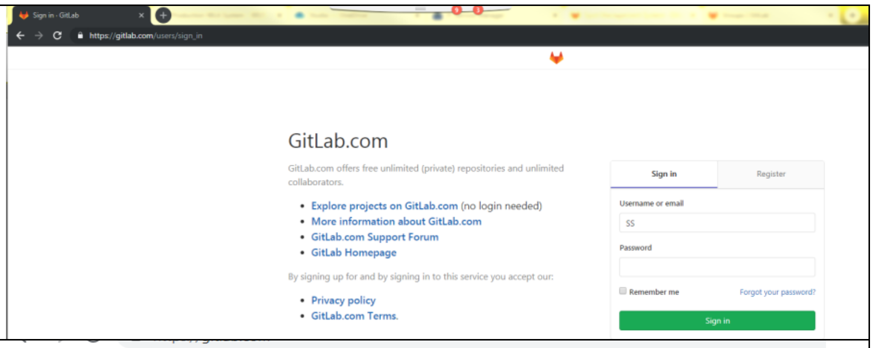
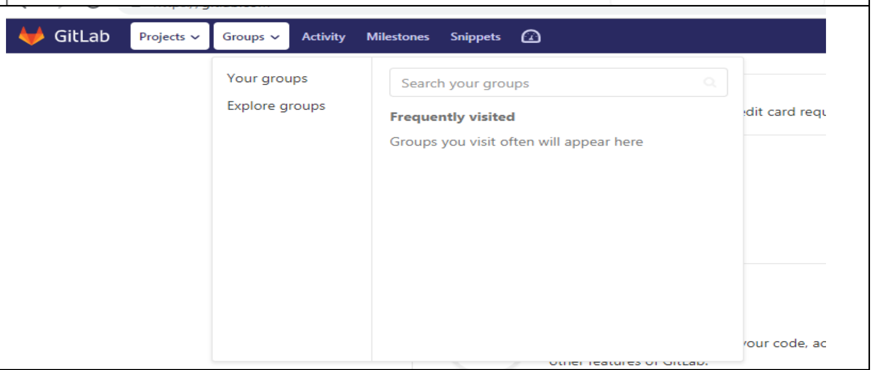
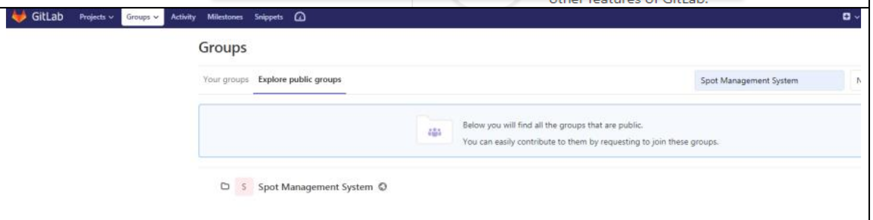
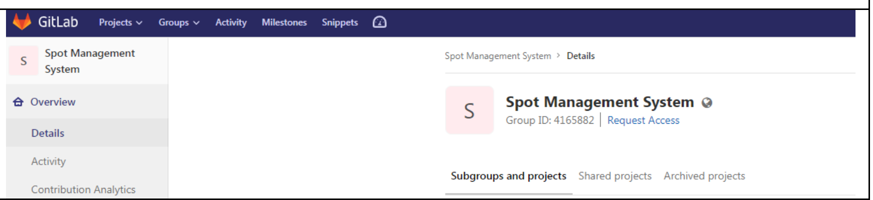
In this guide we assume that operator following this guide possess all basic system administration skills including basic knowledge of usage of git commands and basic knowledge of Amazon Web Services.

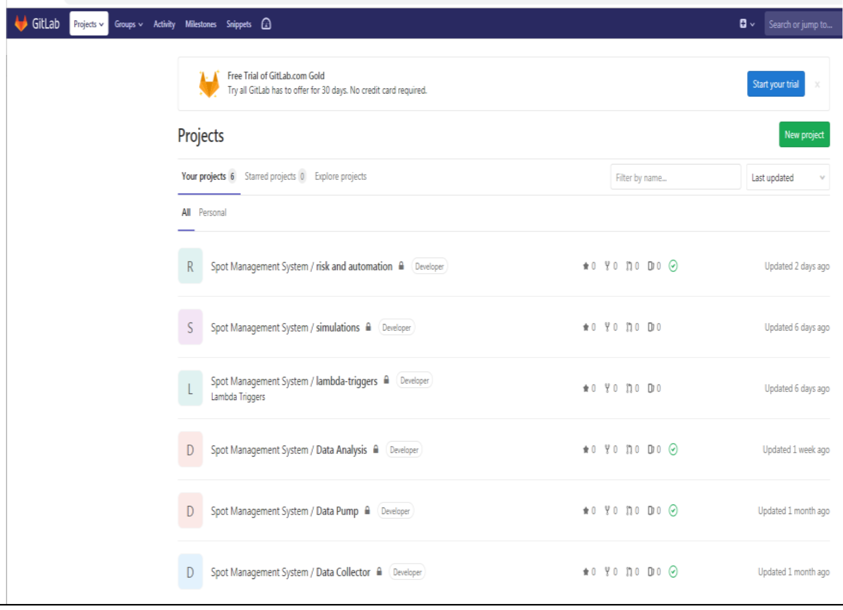
3 Development Environment Requirements

3.1 Code Repository

3.1.1 GIT LAB Access

In order to have access to the developed code you need first request access to the Spot Management System group in GIT LAB. To request access please follow steps outlined below:

1 Go to https://Gitlab.com and login with your existing account or register for the new account.	
2. Once Logged on, select Groups and Explore groups	
3. In the Groups screen type in Search box: Spot Management System	
4. Click on the group and in group screen click on request access	

5. Once Access is given you should see list of the projects	

3.1.2 Downloading Code from GIT LAB

In order to download code for the GitLab user should have GitLab installed on his PC.
If its Windows PC than please download GITBASH from https://gitforwindows.org/
In case its Linux or Mac please use build in package manager to install the GIT software.

Please follow GIT LAB Documentation for configuring the SSH Keys
https://docs.gitlab.com/ee/gitlab-basics/create-your-ssh-keys.html

Once the GIT has been configured on the PC and GITLAB Account configured with SSH Keys of the user
You can now clone (download required repositories)

Run following commands in the Windows PowerShell or Linux / Mac Terminal to get all required code packages.

Data Collector:
git clone **git@gitlab.com:sport-management-system/data-collector.git**

Data Pump
git clone git@gitlab.com:sport-management-system/data-pump.git

Data Analysis
git clone git@gitlab.com:sport-management-system/data-analysis.git

Risk and Automation
git clone git@gitlab.com:sport-management-system/risk-and-automation.git

Lambda Triggers
git@gitlab.com:sport-management-system/lambda-triggers.git

3.2 Required Programming Languages

3.2.1 PowerShell Core v 6.1

PowerShell core is available for platforms (Windows, Linux, OSX).
For this project following PowerShell modules are required:

- AWSPowerShell.NetCore == 3.3.390.0

3.2.2 Python v 3.6 or above

Install Python 3.6 or above using Anaconda 3 Distribution.
Once installed install following required packages:

- numpy==1.16.4
- pandas==0.24.2
- boto3==1.9.111
- python_dateutil==2.8.0
- joblib==0.13.2
- scikit_learn==0.20.3
- requests==2.21.0

3.2.3 Update to wrapper.ps1 in Data Collector

Before progressing to creation of docker images, please make sure to update script wrapper.ps1
In data-collector and data-pump repositories

In both scripts please update line 2 and replace Access Key and Secret Key with keys configured for your Amazon Account

```
Initialize-AWSDefaultConfiguration -AccessKey <update> -SecretKey <update> -
Region eu-central-1
```

3.3 Creating Docker Images

To create docker images please make sure that you have installed latest docker environment on your development machine.

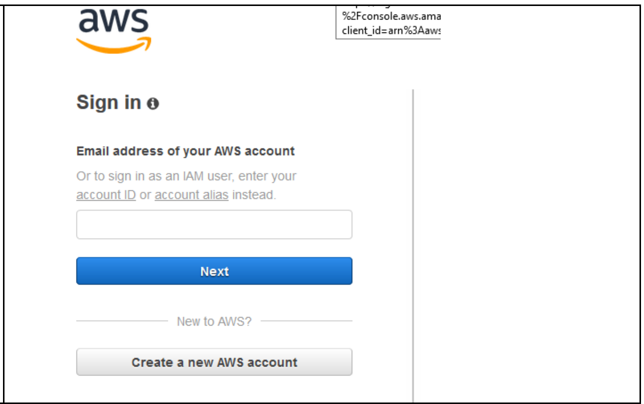
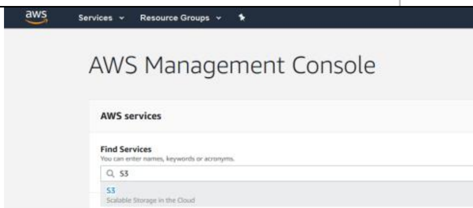
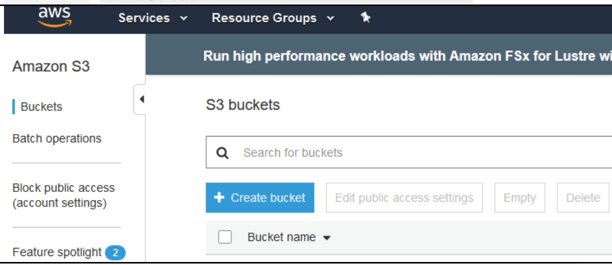
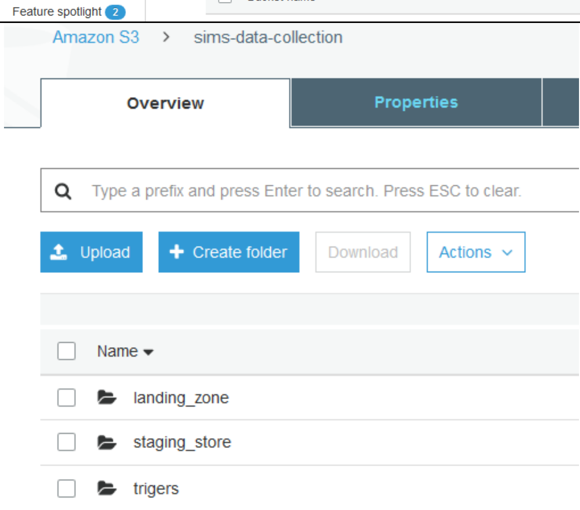
In order to create the docker image of each of the modules please do following:

1. Navigate to root folder of each module (i.e. /opt/scripts/data-analysis)
2. Run following command: **docker build .**

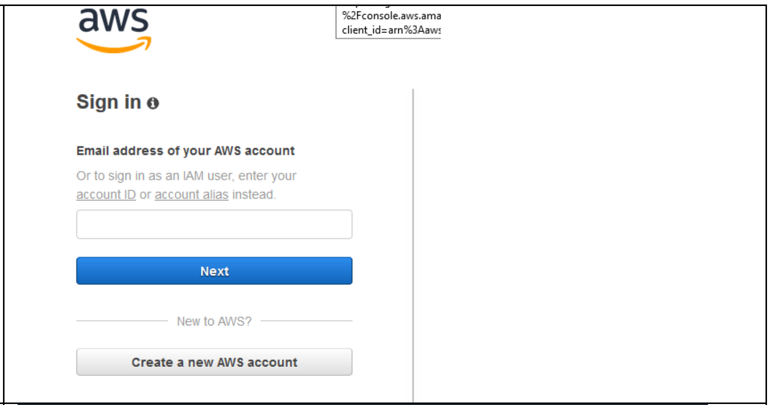
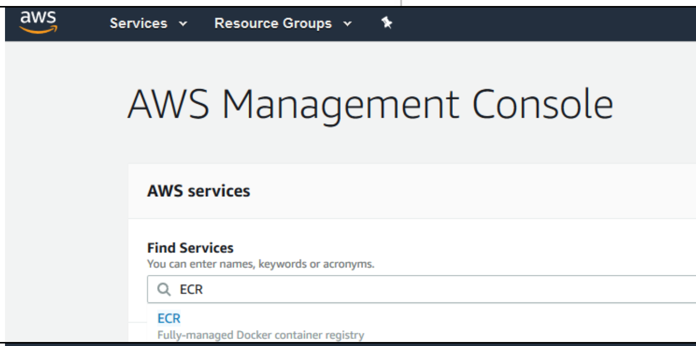
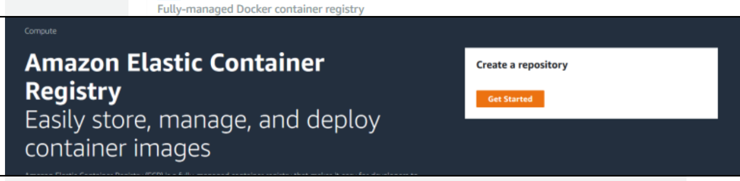
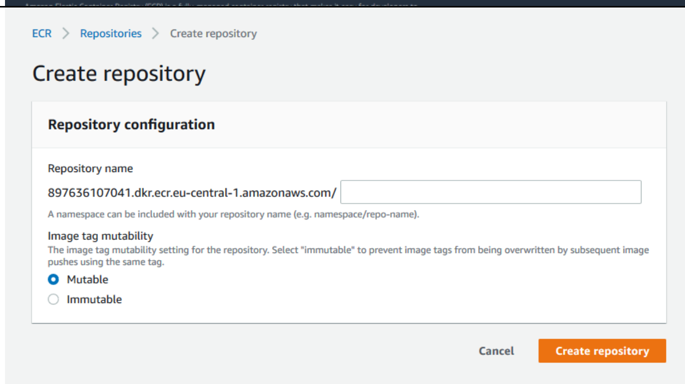
Once docker image is created you can push it to docker repository of your choice (we recommend using AWS ECR (elastic container registry)

4 SimS Deployment Procedure

4.1.1 AWS S3 Bucket Configuration

Login to AWS Console with your username and password	
In AWS management console Find Service S3	
In S3 Console create bucket sims-data-collection by clicking on blue button create bucket,	
Enter the sims-data-collection buck and create following folders: • Landing zone • Staging store • trigers	

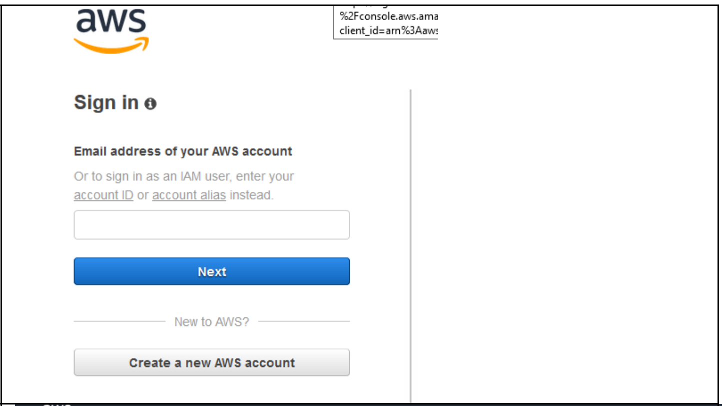
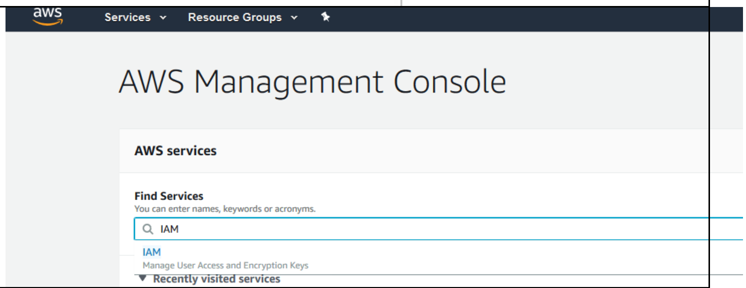
4.1.2 Amazon Elastic Container Registry Setup

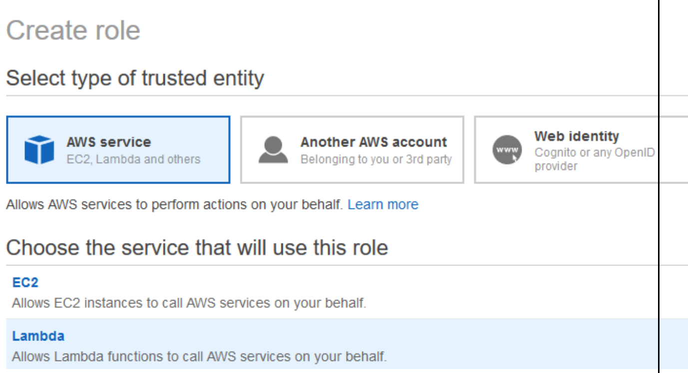
Login to AWS Console with your username and password	
In AWS management console Find Service ECR	
In ECR website click Get-started	
Select name of the repository and press create repository button please create separate repository for each of the modules: - Data-Collector - Data-Analysis - Data-Pump - Risk-and-automation	

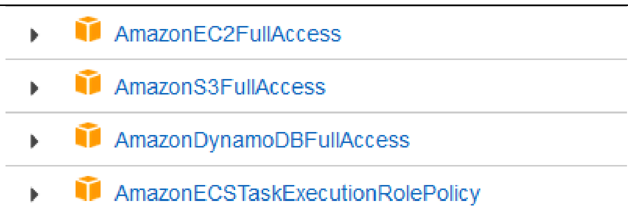
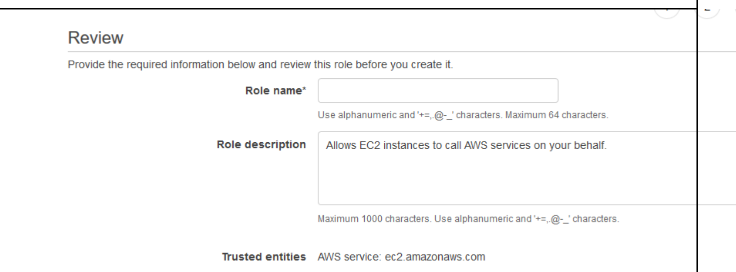
Once the container registry is completed please use command Docker login and docker push in order to push docker images

4.1.3 Create AWS IAM Role for Faregate Execution

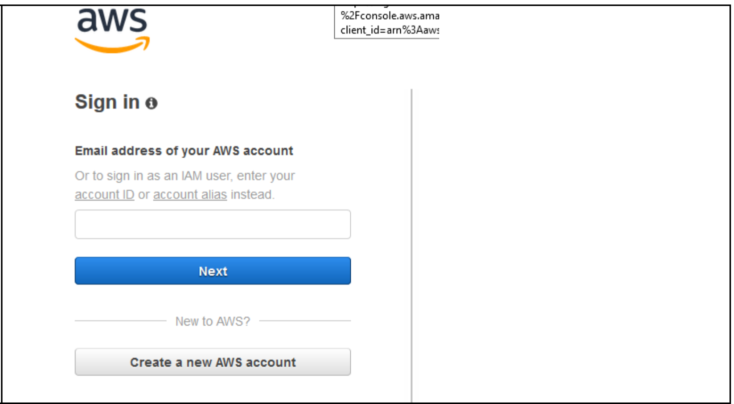
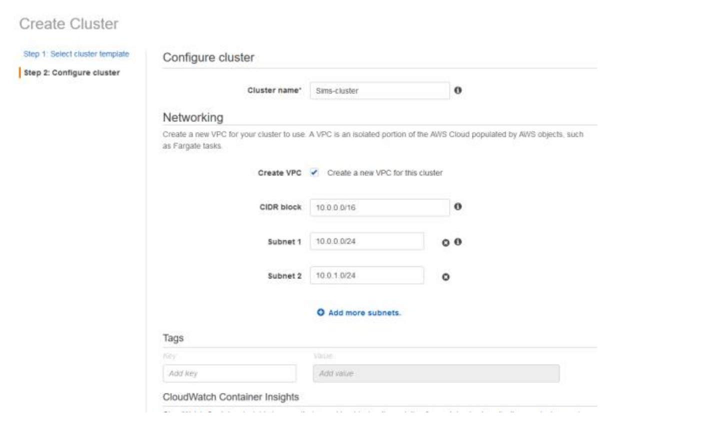
In our example we have given to our role full administrative access to services that we required,
This is not recommended and if possible, you should limit this to only actions that the role really requires.

<table>
<tr><td>Login to AWS Console with your username and password</td><td>
</td></tr>
<tr><td>In AWS management console Find Service IAM</td><td>
</td></tr>
<tr><td>In Create Role screen select following services that will use the role:

• EC2
• Lambda</td><td>
</td></tr>
<tr><td>In permission screen select following Policies:

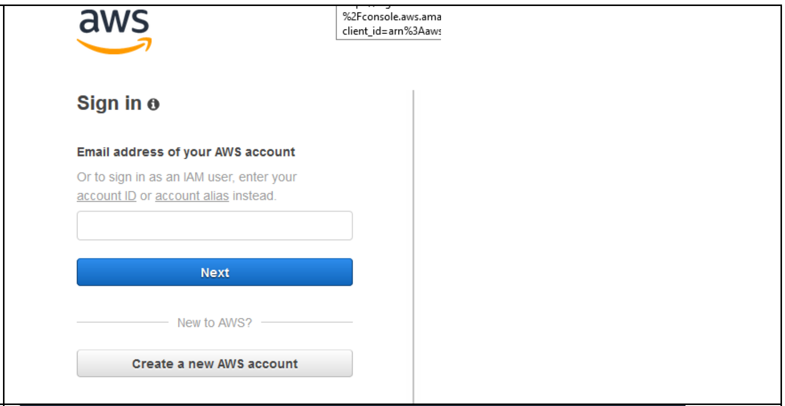
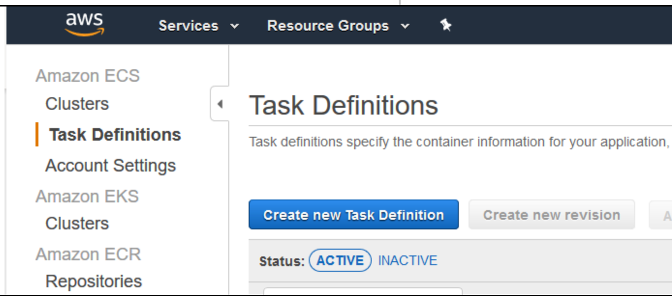
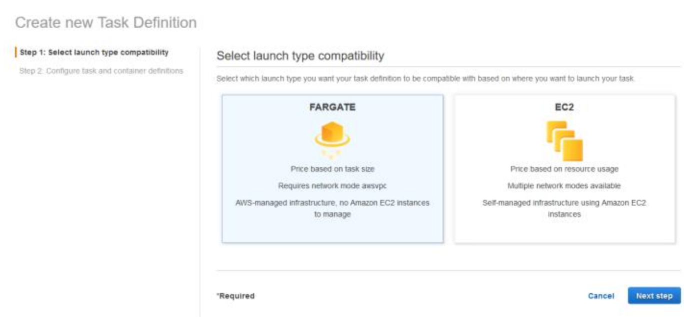
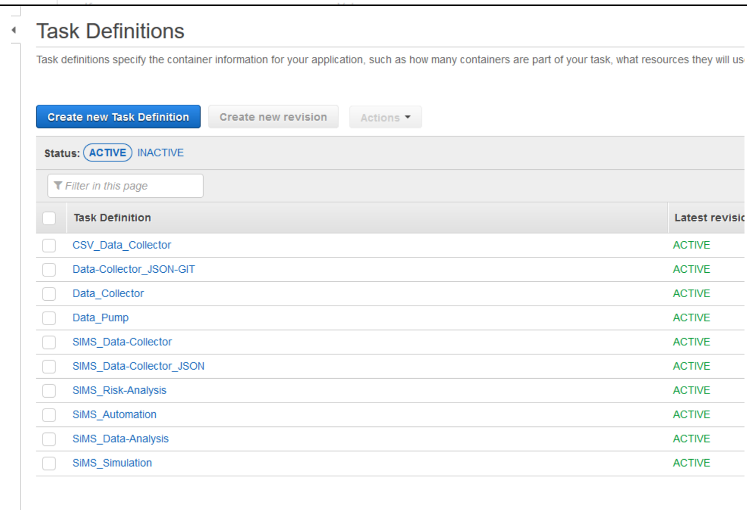
• AmazonEC2FullAccess
• AmazonS3FullAccess
• AmazonDynamoDBFullAccess
• AmazonECSTaskExecutionPolicy</td><td>
</td></tr>
<tr><td>In Tags Screen Click Next</td><td></td></tr>
<tr><td>In Review Screen Type ecsExeutionRole in role name field and click create role.</td><td>
</td></tr>
</table>

4.1.4 Setup AWS Faregate Cluster

Login to AWS Console with your username and password	
On Faregate Control panel, click Clusters under Amazon ECS and press blue button Create Cluster	
In the Create Cluster screen at step one selects Networking Only and at step2 name the cluster Sims-cluster (in the cluster name field) and tick Create VPC	

4.1.5 Setup AWS Faregate Task Definition

In the task definition section, we need to create task definition for each of the core modules of the SiMS Following the same procedure for each

Login to AWS Console with your username and password	
On Faregate Control panel, click Task Definitions under Amazon ECS and press blue button Create new task definition.	
In the Create new Task definition screen select FARGATE and click next Step (repeat this and below steps for each of the tasks mentioned in next table)	
Scroll down and select button Configure via JSON. Paste content of the table below	
Once Configuration is completed you should have number of tasks created	

Task definition:

Please replace mark red IAM and Image sections of each json and replace with your own:

<table>
<tr><th>Data Collection</th><th>Data Pump</th></tr>
<tr><td>
{
"ipcMode": null,
"executionRoleArn":
"arn:aws:iam::897636107041:role/ecsTaskExecutionRole",
"containerDefinitions": [
{
"dnsSearchDomains": null,
"logConfiguration": {
"logDriver": "awslogs",
"secretOptions": null,
"options": {
"awslogs-group": "/ecs/SIMS_Data-Collector",
"awslogs-region": "eu-west-1",
"awslogs-stream-prefix": "ecs"
}
},
"entryPoint": [
"pwsh -c ./get-data-csv.ps1"
],
"portMappings": [
{
"hostPort": 22,
"protocol": "tcp",
"containerPort": 22
}
],
"command": [],
"linuxParameters": null,
"cpu": 0,
"environment": [],
"resourceRequirements": null,
"ulimits": null,
"dnsServers": null,
"mountPoints": [],
"workingDirectory": null,
"secrets": null,
"dockerSecurityOptions": null,
"memory": null,
"memoryReservation": 4096,
"volumesFrom": [],
"stopTimeout": null,
"image": "897636107041.dkr.ecr.eu-west-1.amazonaws.com/nci_sims/data-collector",
"startTimeout": null,
"dependsOn": null,
"disableNetworking": null,
"interactive": null,
"healthCheck": null,
"essential": true,
"links": [],
"hostname": null,
"extraHosts": null,
"pseudoTerminal": null,
"user": null,
"readonlyRootFilesystem": null,
"dockerLabels": {
"Stage": "FAT",
"System": "SIMS"
},
"systemControls": null,
"privileged": null,
"name": "SIMS_DC"
}
],
"placementConstraints": [],
"memory": "4096",
"taskRoleArn": null,
"compatibilities": [
</td><td>
{
"ipcMode": null,
"executionRoleArn":
"arn:aws:iam::897636107041:role/ecsTaskExecutionRole",
"containerDefinitions": [
{
"dnsSearchDomains": null,
"logConfiguration": {
"logDriver": "awslogs",
"secretOptions": null,
"options": {
"awslogs-group": "/ecs/Data_Pump",
"awslogs-region": "eu-west-1",
"awslogs-stream-prefix": "ecs"
}
},
"entryPoint": [
"pwsh"
],
"portMappings": [],
"command": [
"./move-to-stage.ps1"
],
"linuxParameters": null,
"cpu": 0,
"environment": [],
"resourceRequirements": null,
"ulimits": null,
"dnsServers": null,
"mountPoints": [],
"workingDirectory": null,
"secrets": null,
"dockerSecurityOptions": null,
"memory": null,
"memoryReservation": null,
"volumesFrom": [],
"stopTimeout": null,
"image": "897636107041.dkr.ecr.eu-west-1.amazonaws.com/nci_sims/data-pump",
"startTimeout": null,
"dependsOn": null,
"disableNetworking": null,
"interactive": null,
"healthCheck": null,
"essential": true,
"links": null,
"hostname": null,
"extraHosts": null,
"pseudoTerminal": null,
"user": null,
"readonlyRootFilesystem": null,
"dockerLabels": {
"Project": "SIMS"
},
"systemControls": null,
"privileged": null,
"name": "data-pump"
}
],
"placementConstraints": [],
"memory": "4096",
"taskRoleArn":
"arn:aws:iam::897636107041:role/ecsTaskExecutionRole",
"compatibilities": [
"EC2",
"FARGATE"
],
"taskDefinitionArn": "arn:aws:ecs:eu-west-1:897636107041:task-
</td></tr>
</table>

"EC2",
"FARGATE"
],
"taskDefinitionArn": "arn:aws:ecs:eu-west-1:897636107041:task-definition/SIMS_Data-Collector:1",
"family": "SIMS_Data-Collector",
"requiresAttributes": [
{
"targetId": null,
"targetType": null,
"value": null,
"name": "ecs.capability.execution-role-ecr-pull"
},
{
"targetId": null,
"targetType": null,
"value": null,
"name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
},
{
"targetId": null,
"targetType": null,
"value": null,
"name": "ecs.capability.task-eni"
},
{
"targetId": null,
"targetType": null,
"value": null,
"name": "com.amazonaws.ecs.capability.ecr-auth"
},
{
"targetId": null,
"targetType": null,
"value": null,
"name": "ecs.capability.execution-role-awslogs"
},
{
"targetId": null,
"targetType": null,
"value": null,
"name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
},
{
"targetId": null,
"targetType": null,
"value": null,
"name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"
},
{
"targetId": null,
"targetType": null,
"value": null,
"name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
}
],
"pidMode": null,
"requiresCompatibilities": [
"FARGATE"
],
"networkMode": "awsvpc",
"cpu": "1024",
"revision": 1,
"status": "ACTIVE",
"proxyConfiguration": null,
"volumes": []
}
	definition/Data_Pump:2",
	"family": "Data_Pump",
	"requiresAttributes": [
	{
	"targetId": null,
	"targetType": null,
	"value": null,
	"name": "ecs.capability.execution-role-ecr-pull"
	},
	{
	"targetId": null,
	"targetType": null,
	"value": null,
	"name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
	},
	{
	"targetId": null,
	"targetType": null,
	"value": null,
	"name": "ecs.capability.task-eni"
	},
	{
	"targetId": null,
	"targetType": null,
	"value": null,
	"name": "com.amazonaws.ecs.capability.ecr-auth"
	},
	{
	"targetId": null,
	"targetType": null,
	"value": null,
	"name": "com.amazonaws.ecs.capability.task-iam-role"
	},
	{
	"targetId": null,
	"targetType": null,
	"value": null,
	"name": "ecs.capability.execution-role-awslogs"
	},
	{
	"targetId": null,
	"targetType": null,
	"value": null,
	"name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
	},
	{
	"targetId": null,
	"targetType": null,
	"value": null,
	"name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
	}
	],
	"pidMode": null,
	"requiresCompatibilities": [
	"FARGATE"
	],
	"networkMode": "awsvpc",
	"cpu": "1024",
	"revision": 2,
	"status": "ACTIVE",
	"proxyConfiguration": null,
	"volumes": []
	}

Data Analysis	**Automation**
{ "ipcMode": null, **"executionRoleArn": "arn:aws:iam::897636107041:role/ecsTaskExecutionRole",**	{ "ipcMode": null, "executionRoleArn": "arn:aws:iam::897636107041:role/ecsTaskExecutionRole",

```
"containerDefinitions": [
  {
    "dnsSearchDomains": null,
    "logConfiguration": {
      "logDriver": "awslogs",
      "secretOptions": null,
      "options": {
        "awslogs-group": "/ecs/SiMS_Data-Analysis",
        "awslogs-region": "eu-west-1",
        "awslogs-stream-prefix": "ecs"
      }
    },
    "entryPoint": [
      "python3",
      "spot-predict.py",
      "eu-central-1a"
    ],
    "portMappings": [],
    "command": [],
    "linuxParameters": null,
    "cpu": 0,
    "environment": [],
    "resourceRequirements": null,
    "ulimits": null,
    "dnsServers": null,
    "mountPoints": [],
    "workingDirectory": "/opt/data-analysis",
    "secrets": null,
    "dockerSecurityOptions": null,
    "memory": null,
    "memoryReservation": null,
    "volumesFrom": [],
    "stopTimeout": null,
    "image": "897636107041.dkr.ecr.eu-west-1.amazonaws.com/nci_sims/data-analysis",
    "startTimeout": null,
    "dependsOn": null,
    "disableNetworking": null,
    "interactive": null,
    "healthCheck": null,
    "essential": true,
    "links": null,
    "hostname": null,
    "extraHosts": null,
    "pseudoTerminal": null,
    "user": null,
    "readonlyRootFilesystem": null,
    "dockerLabels": {
      "Name": "DA-Central"
    },
    "systemControls": null,
    "privileged": null,
    "name": "SIMS_DA-Central-1a"
  },
  {
    "dnsSearchDomains": null,
    "logConfiguration": {
      "logDriver": "awslogs",
      "secretOptions": null,
      "options": {
        "awslogs-group": "/ecs/SiMS_Data-Analysis",
        "awslogs-region": "eu-west-1",
        "awslogs-stream-prefix": "ecs"
      }
    },
    "entryPoint": [
      "python3",
      "spot-predict.py",
      "eu-central-1b"
    ],
    "portMappings": [],
    "command": [
      ""
    ],
    "linuxParameters": null,
    "cpu": 0,
```

```
"containerDefinitions": [
  {
    "dnsSearchDomains": null,
    "logConfiguration": {
      "logDriver": "awslogs",
      "secretOptions": null,
      "options": {
        "awslogs-group": "/ecs/SiMS_Automation",
        "awslogs-region": "eu-west-1",
        "awslogs-stream-prefix": "ecs"
      }
    },
    "entryPoint": [
      "python3",
      "automation.py"
    ],
    "portMappings": [],
    "command": null,
    "linuxParameters": null,
    "cpu": 0,
    "environment": [],
    "resourceRequirements": null,
    "ulimits": null,
    "dnsServers": null,
    "mountPoints": [],
    "workingDirectory": null,
    "secrets": null,
    "dockerSecurityOptions": null,
    "memory": null,
    "memoryReservation": null,
    "volumesFrom": [],
    "stopTimeout": null,
    "image": "897636107041.dkr.ecr.eu-west-1.amazonaws.com/nci_sims/risk-and-automation:latest",
    "startTimeout": null,
    "dependsOn": null,
    "disableNetworking": null,
    "interactive": null,
    "healthCheck": null,
    "essential": true,
    "links": null,
    "hostname": null,
    "extraHosts": null,
    "pseudoTerminal": null,
    "user": null,
    "readonlyRootFilesystem": null,
    "dockerLabels": null,
    "systemControls": null,
    "privileged": null,
    "name": "SIMS_AUTOMATION"
  }
],
"placementConstraints": [],
"memory": "4096",
"taskRoleArn": "arn:aws:iam::897636107041:role/ecsTaskExecutionRole",
"compatibilities": [
  "EC2",
  "FARGATE"
],
"taskDefinitionArn": "arn:aws:ecs:eu-west-1:897636107041:task-definition/SiMS_Automation:1",
"family": "SiMS_Automation",
"requiresAttributes": [
  {
    "targetId": null,
    "targetType": null,
    "value": null,
    "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
  },
  {
    "targetId": null,
    "targetType": null,
    "value": null,
    "name": "ecs.capability.execution-role-awslogs"
```

```
    "environment": [],
    "resourceRequirements": null,
    "ulimits": null,
    "dnsServers": null,
    "mountPoints": [],
    "workingDirectory": "/opt/data-analysis",
    "secrets": null,
    "dockerSecurityOptions": null,
    "memory": null,
    "memoryReservation": null,
    "volumesFrom": [],
    "stopTimeout": null,
    "image": "897636107041.dkr.ecr.eu-west-1.amazonaws.com/nci_sims/data-analysis",
    "startTimeout": null,
    "dependsOn": null,
    "disableNetworking": null,
    "interactive": null,
    "healthCheck": null,
    "essential": true,
    "links": null,
    "hostname": null,
    "extraHosts": null,
    "pseudoTerminal": null,
    "user": null,
    "readonlyRootFilesystem": null,
    "dockerLabels": {
      "Name": "DA-WEST"
    },
    "systemControls": null,
    "privileged": null,
    "name": "SIMS_DA-Central-1b"
  },
  {
    "dnsSearchDomains": null,
    "logConfiguration": {
      "logDriver": "awslogs",
      "secretOptions": null,
      "options": {
        "awslogs-group": "/ecs/SiMS_Data-Analysis",
        "awslogs-region": "eu-west-1",
        "awslogs-stream-prefix": "ecs"
      }
    },
    "entryPoint": [
      "python3"
    ],
    "portMappings": [],
    "command": [
      "spot-predict.py",
      "eu-central-1c"
    ],
    "linuxParameters": null,
    "cpu": 0,
    "environment": [],
    "resourceRequirements": null,
    "ulimits": null,
    "dnsServers": null,
    "mountPoints": [],
    "workingDirectory": "/opt/data-analysis",
    "secrets": null,
    "dockerSecurityOptions": null,
    "memory": null,
    "memoryReservation": null,
    "volumesFrom": [],
    "stopTimeout": null,
    "image": "897636107041.dkr.ecr.eu-west-1.amazonaws.com/nci_sims/data-analysis",
    "startTimeout": null,
    "dependsOn": null,
    "disableNetworking": null,
    "interactive": null,
    "healthCheck": null,
    "essential": true,
    "links": null,
    "hostname": null,
      },
      {
        "targetId": null,
        "targetType": null,
        "value": null,
        "name": "com.amazonaws.ecs.capability.ecr-auth"
      },
      {
        "targetId": null,
        "targetType": null,
        "value": null,
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
      },
      {
        "targetId": null,
        "targetType": null,
        "value": null,
        "name": "com.amazonaws.ecs.capability.task-iam-role"
      },
      {
        "targetId": null,
        "targetType": null,
        "value": null,
        "name": "ecs.capability.execution-role-ecr-pull"
      },
      {
        "targetId": null,
        "targetType": null,
        "value": null,
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
      },
      {
        "targetId": null,
        "targetType": null,
        "value": null,
        "name": "ecs.capability.task-eni"
      }
    ],
    "pidMode": null,
    "requiresCompatibilities": [
      "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "2048",
    "revision": 1,
    "status": "ACTIVE",
    "proxyConfiguration": null,
    "volumes": []
}
```

      "extraHosts": null,
      "pseudoTerminal": null,
      "user": null,
      "readonlyRootFilesystem": null,
      "dockerLabels": {
        "Name": "DA-WEST"
      },
      "systemControls": null,
      "privileged": null,
      "name": "SIMS_DA-central-1c"
    }
  ],
  "placementConstraints": [],
  "memory": "8192",
  "taskRoleArn":
"arn:aws:iam::897636107041:role/ecsTaskExecutionRole",
  "compatibilities": [
    "EC2",
    "FARGATE"
  ],
  "taskDefinitionArn": "arn:aws:ecs:eu-west-1:897636107041:task-definition/SiMS_Data-Analysis:5",
  "family": "SiMS_Data-Analysis",
  "requiresAttributes": [
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "ecs.capability.execution-role-awslogs"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.ecr-auth"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.task-iam-role"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "ecs.capability.execution-role-ecr-pull"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
    },

```
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "ecs.capability.task-eni"
    }
  ],
  "pidMode": null,
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "networkMode": "awsvpc",
  "cpu": "4096",
  "revision": 5,
  "status": "ACTIVE",
  "proxyConfiguration": null,
  "volumes": []
}
```


Risk Analysis

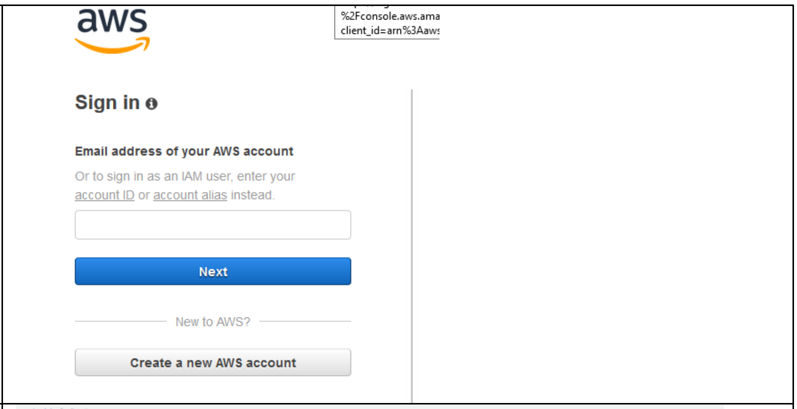
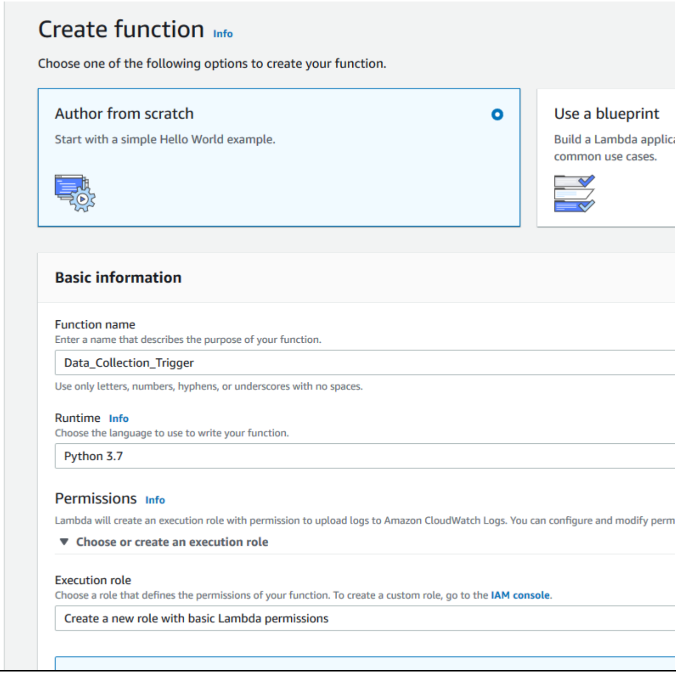

```
{
  "ipcMode": null,
  "executionRoleArn": "arn:aws:iam::897636107041:role/ecsTaskExecutionRole",
  "containerDefinitions": [
    {
      "dnsSearchDomains": null,
      "logConfiguration": {
        "logDriver": "awslogs",
        "secretOptions": null,
        "options": {
          "awslogs-group": "/ecs/SIMS_Risk-Analysis",
          "awslogs-region": "eu-west-1",
          "awslogs-stream-prefix": "ecs"
        }
      },
      "entryPoint": [
        "python3",
        "risk_analysis.py"
      ],
      "portMappings": [],
      "command": null,
      "linuxParameters": null,
      "cpu": 0,
      "environment": [],
      "resourceRequirements": null,
      "ulimits": null,
      "dnsServers": null,
      "mountPoints": [],
      "workingDirectory": null,
      "secrets": null,
      "dockerSecurityOptions": null,
      "memory": null,
      "memoryReservation": null,
      "volumesFrom": [],
      "stopTimeout": null,
      "image": "897636107041.dkr.ecr.eu-west-1.amazonaws.com/nci_sims/risk-and-automation:latest",
      "startTimeout": null,
      "dependsOn": null,
      "disableNetworking": null,
      "interactive": null,
      "healthCheck": null,
      "essential": true,
      "links": null,
      "hostname": null,
      "extraHosts": null,
      "pseudoTerminal": null,
      "user": null,
      "readonlyRootFilesystem": null,
      "dockerLabels": null,
      "systemControls": null,
      "privileged": null,
      "name": "SIMS_RA"
```

```
    }
  ],
  "placementConstraints": [],
  "memory": "4096",
  "taskRoleArn": "arn:aws:iam::897636107041:role/ecsTaskExecutionRole",
  "compatibilities": [
    "EC2",
    "FARGATE"
  ],
  "taskDefinitionArn": "arn:aws:ecs:eu-west-1:897636107041:task-definition/SIMS_Risk-Analysis:1",
  "family": "SIMS_Risk-Analysis",
  "requiresAttributes": [
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "ecs.capability.execution-role-awslogs"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.ecr-auth"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.task-iam-role"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "ecs.capability.execution-role-ecr-pull"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
    },
    {
      "targetId": null,
      "targetType": null,
      "value": null,
      "name": "ecs.capability.task-eni"
    }
  ],
  "pidMode": null,
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "networkMode": "awsvpc",
  "cpu": "2048",
  "revision": 1,
  "status": "ACTIVE",
  "proxyConfiguration": null,
  "volumes": []
}
```

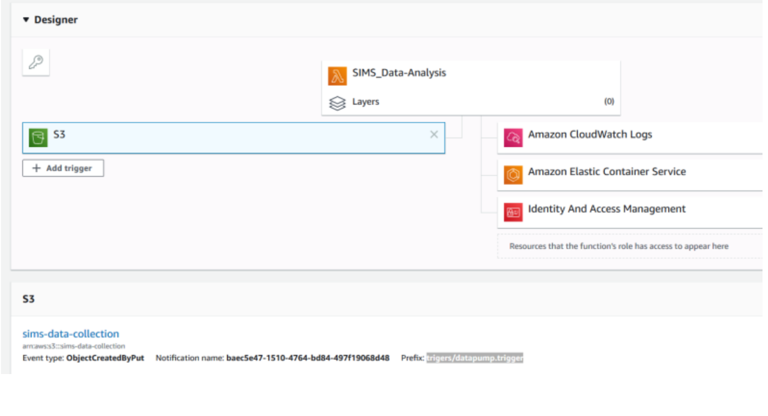
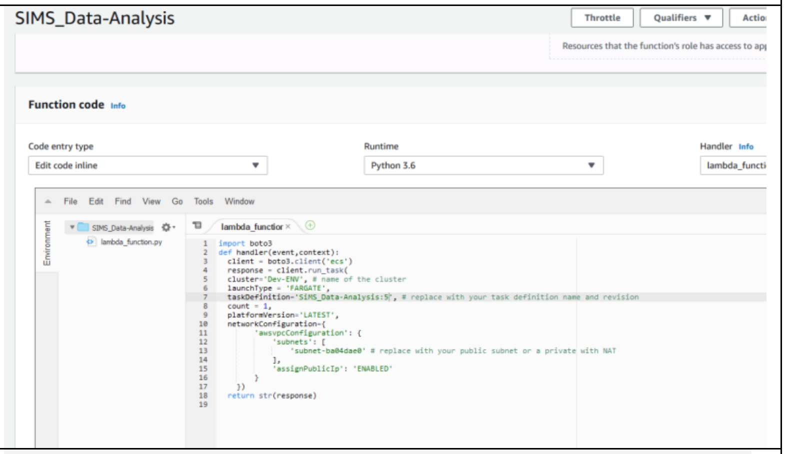
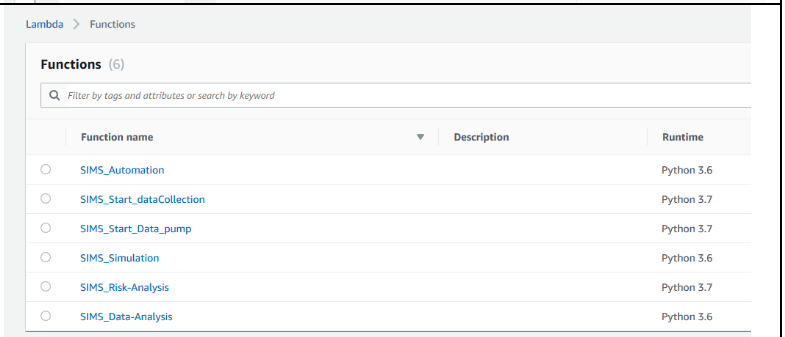
4.1.6 AWS Lambda Configuration

Lambda is used to trigger Faregate tasks at scheduled time or via file trigger
Please follow below steps for each of the actions:

- Data Collection
- Data Analysis
- Data Pump
- Automation
- Risk Analysis

Login to AWS Console with your username and password	
Go to AWS Lambda (Service / Lambda) and click on Create Function	
• In Create Function scream select Author from scratch, • in function name type the name of the action triggered, • in runtime select Python 3.6 or Python 3.7, • in execution role select create new role with basic Lambda permissions	

<table>
<tr>
<td>For Data-Analysis only
On new screen under designer please click on + Add Triger and add S3 File event as a trigger,
In Bucket please select sims-data-collection bucket and in the prefix type trigers/datapump.trigger

For Data-Pump only
Follow above steps, in prefix filed type trigers/collection.trigger</td>
<td>
</td>
</tr>
<tr>
<td>FOR ALL FUNCTIONS
Go to your local folder with trigger scripts
Open a trigger file current stage and copy content of that file.
In Lambda Editor paste it</td>
<td>
</td>
</tr>
<tr>
<td>Repeat above steps for each of the steps and you should have number of functions created</td>
<td>
</td>
</tr>
</table>

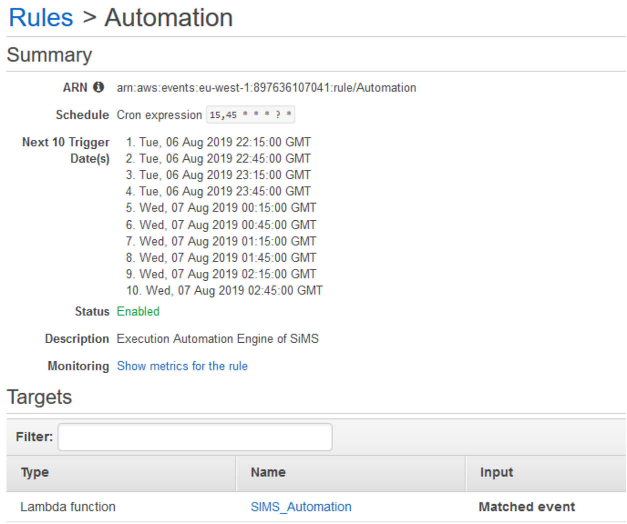
4.1.7 Setup CloudWatch time events

From Amazon Console select service in top menu and type Cloud Watch

<table>
<tr>
<td>In Cloud Watch console select Rules on left panel</td>
<td>
</td>
</tr>
<tr>
<td>Create Rule by clicking Create Rule Button</td>
<td></td>
</tr>
</table>

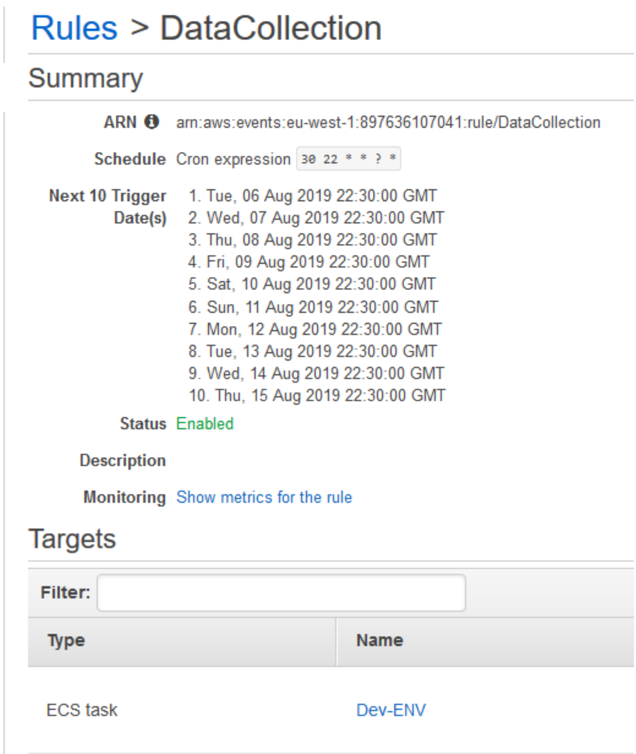
Create **Automation Rule** with following:

- Cron expression 15,45 * * * ? *
- Target: Automation Lambda



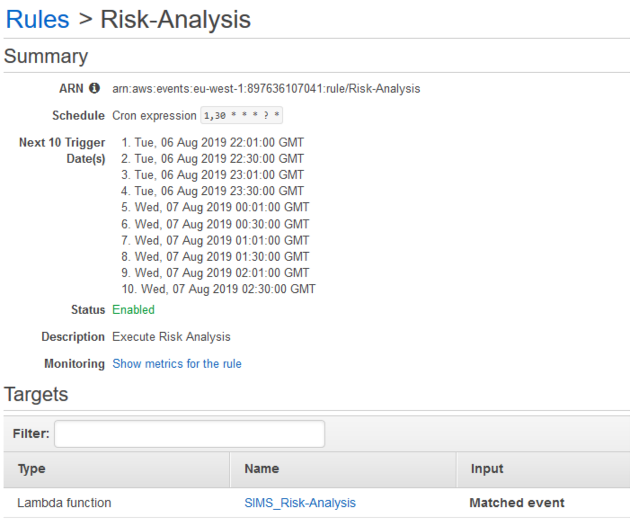
Create **Data Collection Rule**

- Cron expression 30 22 * * ? *
- Target: ECS Task



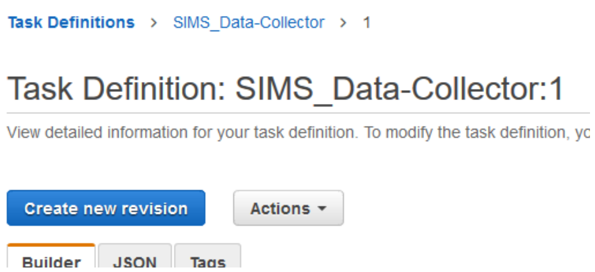
Create **Risk-Analysis Rule**

- Cron expression 1,30 * * * ? *
- Target Risk Analysis Lambda



5 Validation

In order to validate the implementation, please go to AWS Console / Service / ECS
Click on Task definition and open Data Collection task definition.
From Actions menu select RUN TASK



In next screen select :

- Launch type: Fargate
- Cluster VPC : Default
- Subnets Default
- Auto-Assign Public IP: Enable

Click Execute

Observer the task TAB in cluster Screen.
Once tasks are started the Last Status will be Running