A Novel Failure Recovery Technique On AWS Spot Instances With Optimal Cloud Cost

MSc Research Project
Cloud Computing

Sarath Ravichandran
Student ID: x18174311

School of Computing
National College of Ireland

Supervisor: Sean Heeney

Student Name:	Sarath Ravichandran
Student ID:	x18174311
Programme:	Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	17/08/2020
Project Title:	A Novel Failure Recovery Technique on AWS Spot Instance
Word Count:	6900
Page Count:	20

Signature:	
Date:	16th August 2020

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.	☐

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

A Novel Failure Recovery Technique On AWS Spot Instances With Optimal Cloud Cost

Sarath Ravichandran
x18174311

Abstract

The main purpose of this research is to guarantee the reliability of the service that is provided by the AWS(Amazon Web Services)spot instance. A novel spot instance failure recovery life-cycle is proposed to assure the reliability with the minimal cloud cost. This proposal comprises two main modules in it namely forecast and spot instance failure recovery. A unique spot instance failure recovery algorithm is also introduced as a part of this study. Data loss due to spot instance termination is overcome by the suggested algorithm. A web application is built in order to maintain the life cycle of the spot instance. The proposed algorithm would initiate the spot instance AMI creation when the forecast-ed spot price is higher than the user bid price. This AMI would be used to initiate an on-demand instance if the spot instance is terminated. Thus the data loss due to the spot instance termination by the vendor due to lower bid-price is resolved. Once the on-demand instance is created then the application would continuously monitor the user bid price and current spot price. Once the current spot price is equal to the user bid price then the application would generate an AMI for the running on-demand instance and this AMI would be used to create an equivalent spot instance. Thus the cost spent on the cloud is significantly decreased. This proposal is implemented and evaluated on the AWS cloud platform. The evaluation results have concluded that the business continuity is achieved even after the termination of spot instance.

1 Introduction

The principal intention of this research work is to address the absence of holistic reliable service in the cloud platform that is stated in this paperBuyya et al. (2019). Therefore, I proposed a unique spot instance rotation technique to ensure the business continuity of the service that is provided by the spot instance even after it's termination. The implementation and evaluation of this proposal are presented in the following sections of this report.

1.1 Overview on AWS EC2 Spot Instance

In this modern world, most of the enterprises are highly preferring cloud services that are provided by the cloud service provider instead of procuring physical servers. Pay as per the use is one of the main benefits to prefer a cloud platform. Amazon Web Service(AWS) is a leading cloud service provider in the cloud service provider market. AWS is providing the virtual server that could be used as the physical server. AWS is providing the virtual servers in the form of Elastic Cloud Computing(EC2) service. This service is provided in various modes namely reserved, on-demand, and spot instance. Spot instance is an unused spare compute capacity that is available at a steep discount often as much as 90% more inexpensive compared to on-demand prices. The reserved and on-demand instances would not be terminated by the cloud service provider. But spot instances could be terminated by the service provider with a two-minute warning when ec2 needs the capacity back. So the reliability of the service provided by the spot instance is not reliable. AWS spot instances are provided through the bidding approach. The user could provide the highest bid price in order to acquire a spot instance. If the user bid price is equal to or higher than the current spot price then the spot instance would be provided to the user for that hour. If the current spot price is higher than the user bid price then the spot instance could be terminated by the service provider. If the instance is terminated by the service provider then the data linked with that spot instance would be destroyed permanently. This would affect business continuity. There is no mechanism to recover the lost data due to the termination of the spot instance. Therefore, this study is highly stressed on increasing the reliability of the service provided by the spot instance through sustaining the business continuity even after the termination of the spot instance with the least cloud cost. A pictorial representation of the proposed spot instance life cycle is presented below:

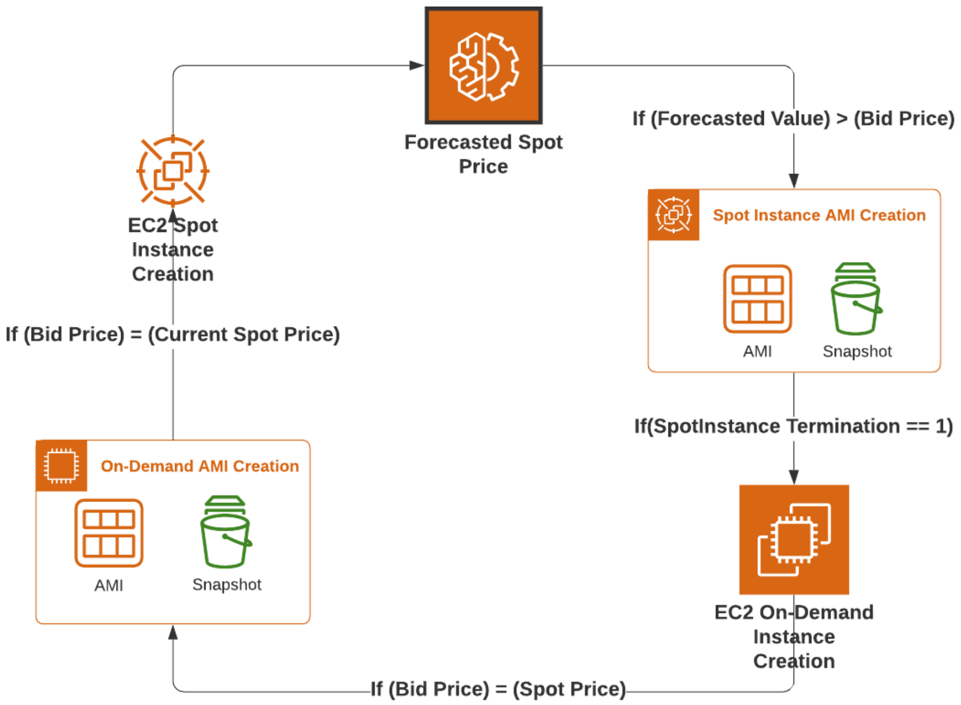

Forecasted Spot Price
If (Forecasted Value) > (Bid Price)
EC2 Spot Instance Creation
Spot Instance AMI Creation
AMI
Snapshot
If (Bid Price) = (Current Spot Price)
If(SpotInstance Termination == 1)
On-Demand AMI Creation
AMI
Snapshot
EC2 On-Demand Instance Creation
If (Bid Price) = (Spot Price)

Figure 1: Proposed Life Cycle

Forecast module is the first step and it is used to launch the back up(AMI) creation process in order to overcome the data loss that occurred due to the spot instance termination. This proposal makes use of both on-demand and spot instance to create the spot instance life cycle as shown in the figure to ensure the business continuity of the AWS spot instance. A new spot instance failure recovery algorithm4.2.1 is proposed to implement the spot instance life cycle.

1.2 Research Question

Since the chance of termination for spot instance is highly dependent on the ec2 instance need in the current market. So, the reliability of the service provided by the spot instance is not guaranteed. Therefore, this research work tries to address the following research question.

How to improve the reliability of the service provided by the AWS spot instance with an optimal price?

A critical investigation is performed between the proposed methodology and the current research works that are associated with the AWS spot instance in order to confirm the novelty of this proposal. The structure of this report is as follows Related Work, Methodology, Design Specification, Implementation, Evaluation, Conclusion, and Future Work.

2 Related Work

This part of the report would provide an overview of the critical analysis that is conducted over the different research works that are related to this research work. Several research works were discovered to be related to our question. But the ones that are published in the conferences and journals that are higher in the standard are considered for this critical analysis. Since our proposal has multiple modules like failure recovery and spot price forecast model. Therefore this critical analysis is segmented into two different segments namely Forecast Spot Price Models and Enhancement of Spot Instance Reliability.

2.1 Forecast Spot Price Models

Research works that are highly focused on guaranteeing the reliability of the spot instance by using a forecast model are critically analyzed in this section.

The research work that was submitted by Baughman et al. (2018) has used the Long Short Term Memory Recurrent Neural Network algorithm to predict the spot price in advance to overcome the spot instance termination. They used the Root Mean Square Error(RMSE) and Mean Absolute Error(MAE) to measure the correctness of the prediction. But still, the reliability of the spot instance is unreliable because if the prediction goes wrong and the spot instance is terminated. Then in that case it does not have any kind of spot instance failure recovery technique. Therefore this work could be further improved by owning a recovery model with it as proposed in our proposal.

A framework was introduced by Zheng et al. (2015), which has the ability to provide how the spot instance pricing is done with the user in order to bypass the spot instance termination due to lower spot price. The major interest of this research work is to achieve the maximum cost reduction by using the spot instance instead of on-demand and reserved instances. Therefore the reliability of the spot instance is still unpredictable because the spot instance may terminate at any time due to lower bid price and the service provided by the spot instance would be suspended. So this work could be further refined by collaborating the spot instance failure recovery techniques that are proposed in this research work to assure the reliability even after the termination of the spot instance.

The work submitted by Khandelwal et al. (2017) is extremely concentrated on preventing the spot instance termination by predicting the spot price in advance with the help of the Regression Random Forest Algorithm. They evaluated the model results in terms of MAPE and MCPE. This work does not guarantee the business continuity of the service provided by spot instance after it's termination. Therefore this gap could be fulfilled by having a failure recovery module as proposed in our research work.

A framework was recommended by Fabra et al. (2019) that would predict the spot price by characterizing each availability zones behavior. Since they are categorizing the availability zones based on the spot price behavior it could be not that accurate as the result produced by machine learning algorithms. So the forecast result could be enhanced by using more than one machine learning algorithm to get more accurate results. Therefore, in my proposal, we used LSTM and ARIMA for the forecast model and the result which has the lowest variation with actual spot price would be considered and presented in my application. Thus the accuracy of the forecast is raised and the reliability is assured by having the failure recovery model.

Since the demand of the ec2 instance modifies with the region, so the spot price for the same instance type could also vary with region. Research work was published by Ekwe-Ekwe and Barker (2018), to explore the relationship between the spot price and the location of the instance instead of predicting the spot price in advance. They have classified the spot instance based on the spot price in different regions. By which they were able to recognize the region that has the lowest deviation in the spot price. By selecting a stable region could prevent the termination of spot instance. But again the reliability of the spot instance is not completely ensured, it might be terminated in the worst case. Therefore a failure recovery module4.2.1 could be collaborated with this idea to assure the business continuity even after the instance termination. In addition, a forecast module could be implemented instead of categorizing the regions to assure reliability.

A new algorithm was introduced by Song et al. (2012), that would update the bid price based on the current bid price in order to avoid the termination of that instance. This research work is highly directed on improving the profit by making use of the spot instance rather than an on-demand instance. Since the forecast module and failure recovery are not considered in this research, the reliability of the service is still unreliable. Therefore a forecast could be made more reliable by using two machine learning algorithms as proposed in our proposal. Further, reliability could be enhanced by implementing the failure recovery module of our proposal.

2.2 Enhancement in The Reliability of Spot Instance

The following research works are utilizing various techniques to improve the reliability of the spot instance.

A detailed characteristics analysis of spot instances for three various regions was presented by Pham et al. (2018) in order to block the spot instance termination. This characterization would help them to recognize a region that has the lowest spot price variation. If the spot price of a region is not varying often then the chance of spot instance termination is also reduced. Thus they are assuring the spot instance reliability. Again the chance of spot instance terminated termination is high once the user bid price is lower than the current spot price. This could suspend the business continuity of the service provided by the spot instance. So this gap could be fulfilled by considering machine learning algorithms like LSTM and ARIMA for making the forecast more accurate as proposed in our work. My proposal would ensure the business continuity even if the worst-case(spot instance termination) occurs. The business continuity is assured by the novel spot instance failure recovery algorithm 4.2.1 which is introduced in this paper.

The research work was written by Khatua and Mukherjee (2013) has introduced a new check-pointing algorithm to increase the reliability of the spot instance. In this method, they created two bid prices in order to increase reliability. Two bid prices would result in two spot instance and this would reduce the cost optimization. This method also produces two snapshots every hour. This would raise the cost further by holding more storage. Therefore this proposal's gap could be fulfilled by considering my proposal. The storage cost is reduced by creating the snapshot only if the forecast-ed value is higher than the bid price and we do not use two spot instance strategies. This would result in higher cost optimization. Further, reliability is also assured by owning the failure recovery module of our proposed application.

A model was introduced by Qu et al. (2016), that assures the reliability of the spot instance by deliberating the workload of that spot instance between the mix of on-demand instance and spot instance. This would improve the business continuity but the user has to compromise with the cost optimization. Therefore this gap could be fulfilled by having only one instance, either spot instance or on-demand instance based on the spot price condition as proposed in the algorithm 4.2.1. This would improve reliability and improves the cost optimization aspect as well.

A scheduler was recommended by He et al. (2015), that highly stressed on enhancing the reliability and cost. This proposal has stated that they are using on-demand instance in the event of spot instance termination. It clearly states that due to the migration the business continuity of the spot instance would be interrupted during these times. But this proposal does not have the idea of returning back to the spot instance once the spot price is equal to the current bid price as proposed in this algorithm4.2.1. So by considering the forecast model we could have the updated work-space of the spot instance even after the termination and the cost could be further reduced by returning back the on-demand instance to spot instance as mention in the life-cycle.

A framework was recommended by Sabyasachi et al. (2017) that enables the user to modify the user spot price based on the importance of the task which is running in the spot instance. An algorithm is also introduced in this research for producing the checkpoints in the running spot instance. The evaluation of this research work is conducted through simulation. The results of the simulation were satisfyingly achieved. By using this proposal the user would be able to improve the business continuity but the cost optimization aspect would be compromised. By collaborating the proposed spot instance life-cycle the user would be able to maximize the cost optimization.

Various checkpointing strategies were introduced by Yi et al. (2012), that would start the creation of checkpoints based on the conditions of each strategy. The principal aim of this work is to assure business continuity by persisting in the latest workspace. The major disadvantage of this proposal is that they do not have any kind of threshold for checkpoint creation. This would end up creating multiple checkpoints for each instance. Therefore the cost spent on storage for those checkpoints would affect the cost optimization aspect. Therefore the gap in this work could be fulfilled by our AMI creation algorithm4.2.2. In this algorithm, only one AMI would be created and the snapshot of that AMI would be updated in an incremental manner to improve the cost optimization.

An overview of the entire critical analysis of the related works that are considered is presented as a table. In which the forecast column represents whether a forecast module is used in that related work or not, Algorithms column describes how many algorithms were used in the forecast model, Failure recovery column express whether a failure recovery module is present or not in that particular related work, and reliability enhancement column indicates whether that related work enhances the reliability of the spot instance.

Table 1: Overview of Related Work

Related Works				
Reference	Forecast	Algorithms	Failure Recovery	Reliability Enhancement
Baughman et al. (2018)	Yes	1	No	Yes
Zheng et al. (2015)	No	0	No	Yes
Khandelwal et al. (2017)	Yes	1	No	Yes
Fabra et al. (2019)	No	0	No	Yes
Pham et al. (2018)	No	0	No	Yes
Khatua and Mukherjee (2013)	No	0	Yes	Yes
Qu et al. (2016)	No	0	Yes	Yes
He et al. (2015)	No	0	Yes	Yes
Sabyasachi et al. (2017)	Yes	1	No	Yes
Ekwe-Ekwe and Barker (2018)	No	0	No	Yes
Song et al. (2012)	Yes	1	No	Yes
Yi et al. (2012)	No	0	Yes	Yes
Proposed Research	Yes	2	Yes	Yes

2.3 Novelty of This Proposal

Numerous research works were discovered in the current state of the art that is related to the AWS spot instance. Most of the researches is extremely concerned about cost optimization by preventing spot instance termination by foretelling the spot price in advance. Some of the research works are highly concentrated on assuring business continuity through different check-pointing strategies like the ones that are explained in the critical analysis. In the current state of the art, the user can either obtain cost optimization or service reliability. But with this proposal, it has the failure recovery module to assure the service reliability of the spot instance, and the proposed algorithm4.2.1 would guarantee the cost optimization. Further, this proposal has used a couple of machine learning algorithms namely LSTM and ARIMA to make the spot price forecast to be more precise, which is not done in any of the related works. A web application that has both the capabilities like projecting the spot price in advance and spot instance failure recovery is not present in any of the related works. This provides evidence of the novelty of this proposal. Each module of this proposal is described in detail in the subsequent section of this report.

3 Methodology

A web application is created in order to implement the proposed spot instance life cycle. The principal objective of this web application is to guarantee the business continuity of the AWS spot instance even if it terminates and the secondary focus is to assure cost optimization throughout the life cycle. This application has two major modules in it namely spot price forecast module and spot instance failure recovery module. Spot price forecast module is used to project the predicted spot price for the end-user through the web application home screen. Failure recovery module is implemented with the help of this novel algorithm4.2.1. This algorithm would ensure both the objectives. The detailed figure of my entire proposal is presented in the figure2.

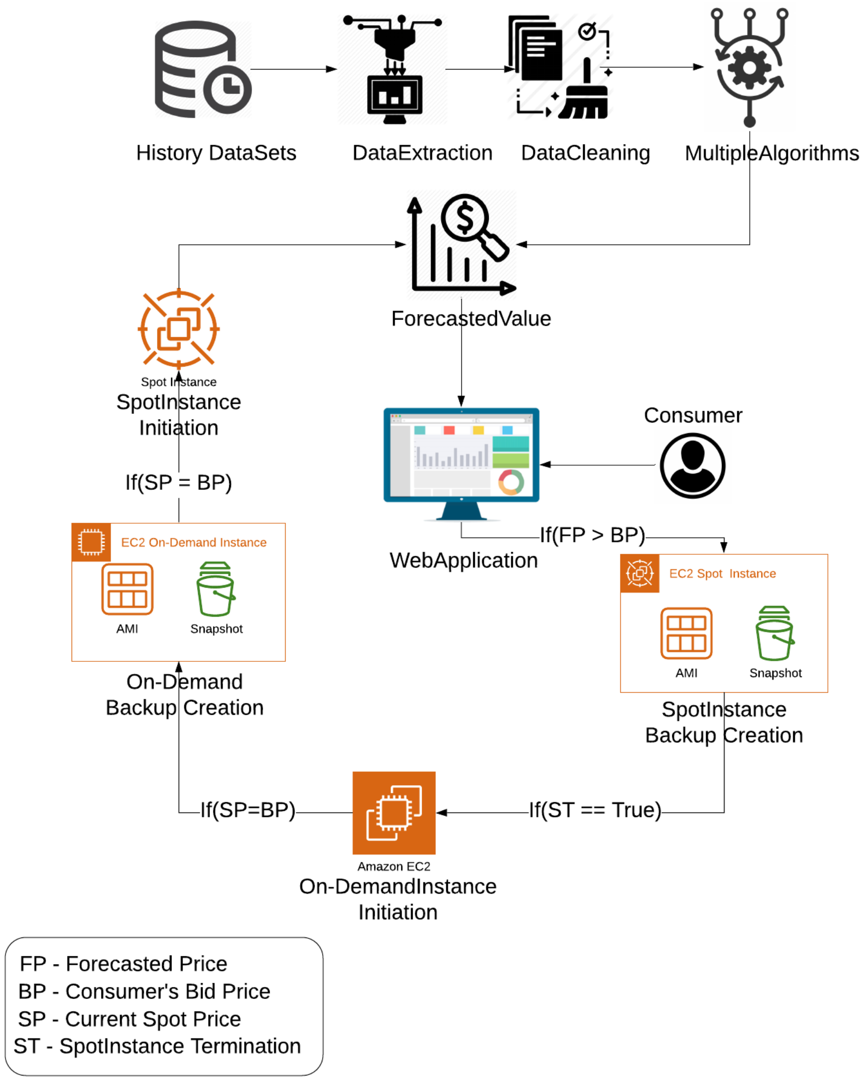


Figure 2: Proposed Methodology

As presented in the above figure the end user could make use of our proposal through the web application that is hosted in the link that is given in the following section5.

3.1 Spot Price Forecast

The chief objective of the spot price forecast module is to initiate the AMI creation process before the spot instance termination. This module is the initial step of the proposed life cycle. This module is responsible for AMI creation of the running spot instance. If the predicted spot price is higher than the user bid price then the chance of spot instance termination is very high. Therefore, this module would initiate the AMI creation of the running spot instance. This spot instance AMI would be used for the corresponding equivalent on-demand instance creation in case of spot instance termination. Thus the data of spot instance is persisted even after its termination. A detailed description of this forecast module is provided in the design section4.1.

3.2 Spot Instance Failure Recovery

The main agenda of the spot instance failure recovery module is to initiate the corresponding alternative instances based on the algorithm4.2.1 conditions. This module would make use of the created AMI to initiate the corresponding instance creation process. If the spot instance is terminated then an equivalent on-demand instance would be created. Thus the business continuity aspect is achieved. Once the user bid price is equal to the current spot price. Then running an on-demand instance could cost more to the user. Therefore the running on-demand instance is terminated and an equivalent spot instance is created only if the user bid price is equal to the current spot price. Therefore the cost optimization aspect is also enhanced. A detailed explanation of this module is presented in the following section 4.2

4 Design Specification

This section of the report would give a detailed design of each module that is present in this proposal. The forecast module, of this proposal, has used two machine algorithms namely Long Short Term Memory(LSTM) and Auto-Regressive Integrated Moving Average(ARIMA). The results of both the algorithm would be compared with the current spot price and the result with the lowest error rate would be presented to the end-user through the created web application. As a part of the spot instance failure recovery module, two new algorithms like spot instance failure recovery algorithm4.2.1 and AMI creation algorithm4.2.2 are proposed.

4.1 Spot Price Forecast Module

The forecast module is the initial step in our proposed spot instance life cycle process. If the forecast-ed value is higher than the user bid price then the chance of spot instance termination is very high. Therefore, this module would start the backup creation process(AMI creation) once the forecast-ed value is higher than the user bid price. To forecast the spot price in advance we need some sample data to train and create a model to make the predictions. AWS is providing the spot price history for each instance type, OS type, and regions that are available. They are providing the spot price history for the last ninety days. This data was retrieved in JSON format using AWS CLI. Therefore, a script was created a script to retrieve the spot price history for the last 90 days and it would write that JSON output to the text file. Then this text file is converted into CSV file using the python program. This CSV file is used to train and test the created model to forecast the spot price in advance. Before creating the model it is mandatory to clean the dataset. Cleaning of the dataset is performed with the help of the pandas library which is available in the python. This cleaning process includes clearing the empty rows and filling the data between the time range. Once the data is cleaned then the models could be created to forecast the spot price in advance. In our proposal, the forecast module is used to forecast the spot price in advance. That implies the user would be able to see the spot price for each hour of the current day. As per our proposal, I have created two models using two different machine learning algorithms namely LSTM and ARIMA. In which LSTM has shown the lowest error rate so we projected the result of LSTM in the created web application. Forecast models are created with the help of the Jupyter Notebook. A detailed diagram of our forecast model is shown in the below figure:

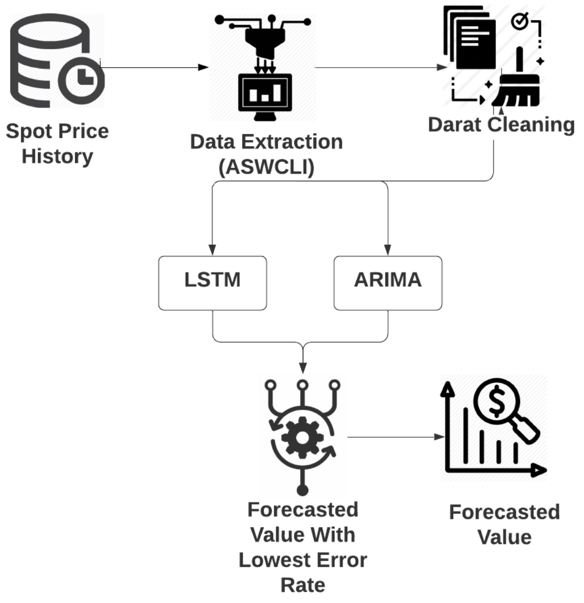


Figure 3: Forecast Module

4.2 Spot Instance Failure Recovery Module

Two novel algorithms were created for this module. The web application would execute these algorithms for every ten seconds in order to initiate the corresponding process like instance creation and AMI creation based on the algorithm's conditions. Those two algorithms are mentioned below:

- Spot Instance Failure Recovery Algorithm
- AMI Creation Algorithm

4.2.1 Spot Instance Failure Recovery Algorithm

A novel algorithm is proposed as a part of this research to assure the reliability of the service that is provided by the spot instance. If the forecast-ed spot price is higher than the user bid price. This would begin the AMI creation of the running spot instance. Every AMI is backed up with the snapshot of the corresponding instance. If a running spot instance is terminated by the vendor due to out of bid then the created web application would create a corresponding on-demand instance of the same type with the help of the AMI that is created previous step. Therefore, the business continuity is ensured even after the termination of the spot instance. The created on-demand instance will be alive until the spot price reaches the user bid price. Once the user bid price is equal to the current spot price then this algorithm would generate an AMI of the running on-demand instance. Once the on-demand AMI is created then that on-demand instance would be terminated automatically. After the on-demand instance termination, an equivalent spot instance would be created with the on-demand AMI. This would enhance the cost optimization. The logic of this algorithm is shown below:

Algorithm 1: Spot Instance Failure Recovery Algorithm

```
Result: Instance Creation
Predicted_SpotPrice, User_BidPrice, Termination_Check, OnDemand_Instance,
  Spot_Instance, Current_SpotPrice;
FailureRecovery(Predicted_SpotPrice, User_BidPrice, Termination_Check,
  OnDemand_Instance, Spot_Instance, Current_SpotPrice);
if Predicted_SpotPrice >= User_BidPrice then
    SpotInstanceAMI_Creation() ;
end
if (Termination_Check = True) AND (Spot_Instance = True) then
    On-DemandCreation(SpotInstanceAMI);
end
if (Current_SpotPrice = User_BidPrice) AND (OnDemand_Instance = True)
  AND (Termination_Check = True) then
    OnDemandAMI_Creation();
    OnDemand_Termination();
    SpotInstance_Creation(OnDemandInstanceAMI);
end
```

4.2.2 AMI Creation Algorithm

A novel algorithm is created in order to persist the updated data of the running instance. This algorithm would produce an AMI for each instance based on the conditions that are mentioned in the Algorithm4.2.2. Every AMI is associated with a snapshot of the running instance. Therefore with the help of the AMI's Snapshot, the content of the terminated instance would be persisted even after the termination of the instance. This algorithm is highly focused on avoiding the creation of Duplicate AMI. If an AMI is not created for an instance then it would create a new AMI for that instance. If the AMI is already created for an instance then this algorithm would update the snapshot associated with that AMI incrementally. In order to have the updated data for instance creation. Through which duplication in the creation of AMI is prevented. The logic of this algorithm is shown below:

Algorithm 2: Spot Instance Check-Pointing

```
Result: AMI Creation
Predicted_SpotPrice, User_BidPrice, AMI_Check ;
AMI_Ceation(Predicted_SpotPrice, User_BidPrice, AMI_Check);
if AMI_Check = False then
    AMICreation() ;
    else

    end
    Incremental_SnapshotCration(AMI);
end
```

5 Implementation

AWS Instance Management(AIM) is a web application that was designed in order to implement the proposal and it would enable the end-user to make use of our proposal. Various technologies were used in the created web application namely Python, Django, HTML, Jquery, AJAX, Postgres, AWS CLI, and etc. The web application was created using the Django framework which is a python web application framework. This web application is deployed in Heroku. Heroku is a cloud platform that provides Platform as a service. In the web application, the back-end code was written in python and the front-end was written in HTML. Postgresql is used as the database to keep track of instance creation and AMI creation. All the AWS interaction in our application is performed through the boto3 python library. The technical specification of used technologies are mentioned below.

Technical Specifications:

- Programming Language - Python 2.8.1
- Web Application Framework - Django 3.0.7
- Database - Postgres 12.3
- AWS Library - boto3 1.14.12
- Machine Learning Library - Keras 2.3.1
- Machine Learning Library - Tensorflow 2.1.0
- Machine Learning Library - Pandas 1.0.3

Web Application Link: https://awsinstancemanagement.herokuapp.com/

5.1 AWS Instance Management Functionalities

AWS Instance Management(AIM) was built with the above specifications and hosted in Heroku and the link is mentioned above. To make use of this application the user needs to have an AWS account and these following parameters from that account AWS_Access_Key, AWS_Secret_Key, and Region. Once the user has this information then they would be able to make use of our proposal by logging in to our application. This web application has various modules in it namely Registration, Login, Home Page, Admin Page, and Instance Monitor Screen.

5.1.1 Registration

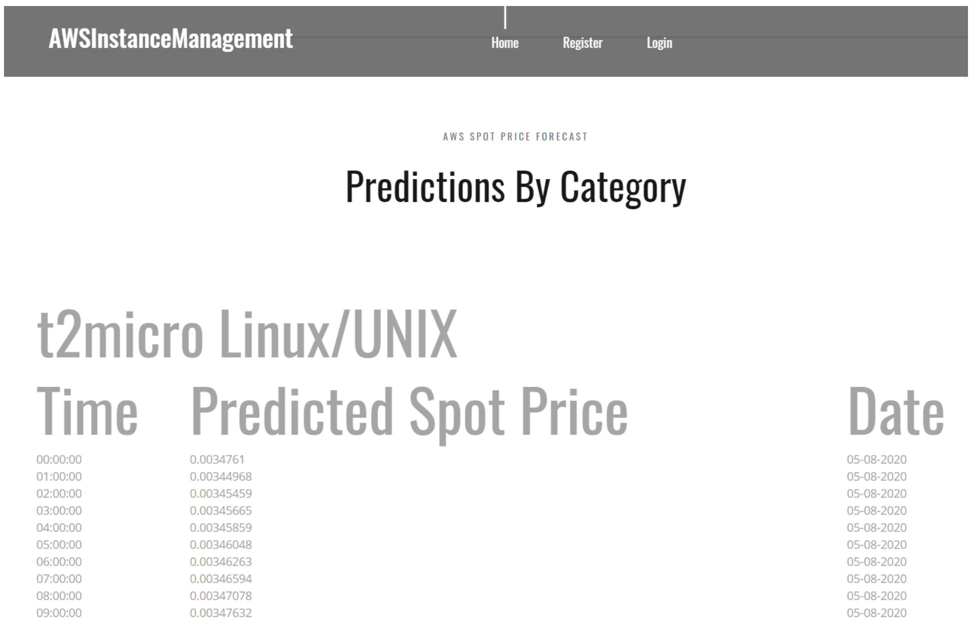
Registration functionality of the AIM application would allow the end-user to register themselves into this application. The user can get into the registration screen by clicking on the register tab on the home screen. The user needs to provide the FirstName, LastName, Username, Email, Password, and confirm password information to register with the AIM application.

5.1.2 Login

The login functionality of the AIM application would allow the user to login to the application. To log in to the application, the user needs to provide the username and password. Once the user is authenticated then they would be forwarded to the homepage. If the user is not valid then the user would be asked to log in again.

5.1.3 Home Page

The home page of the AIM application would update on its own based on the user login status. If the user is not logged in then they would be able to only see the predicted spot price4. But if the user is login then they would be able to see the search form5. Using this search form the user can monitor their AWS instances in our application by providing the following parameters namely AWS_Access_Key, AWS_Secret_Key, Region, and User Bid Price. Based on these parameters my application would interact with the corresponding AWS account through the boto3 library. The home screen would look like as shown before for the not logged in user and logged in user.



Predictions By Category

t2micro Linux/UNIX

Time	Predicted Spot Price	Date
00:00:00	0.0034761	05-08-2020
01:00:00	0.00344968	05-08-2020
02:00:00	0.00345459	05-08-2020
03:00:00	0.00345665	05-08-2020
04:00:00	0.00345859	05-08-2020
05:00:00	0.00346048	05-08-2020
06:00:00	0.00346263	05-08-2020
07:00:00	0.00346594	05-08-2020
08:00:00	0.00347078	05-08-2020
09:00:00	0.00347632	05-08-2020

Figure 4: Home Screen of Not Logged in User

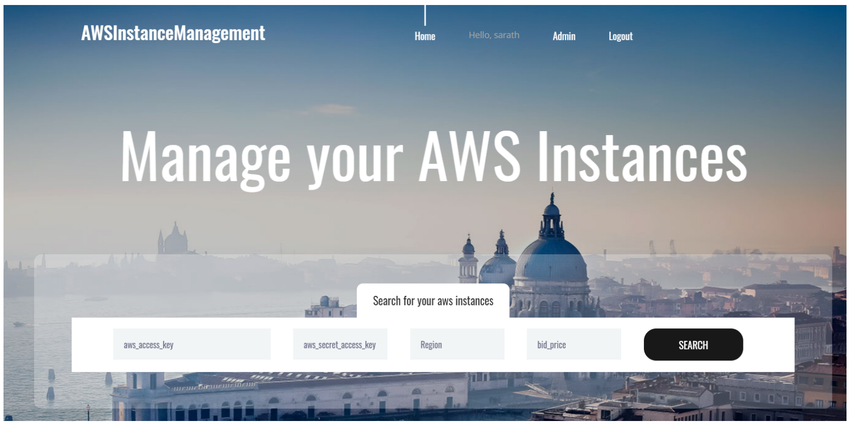
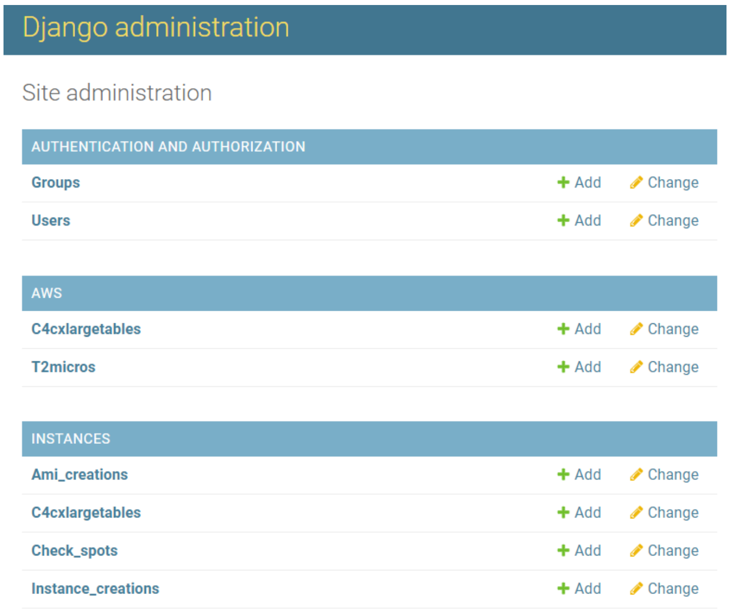


Figure 5: Home Screen of Logged in User

5.1.4 Admin Page

If the logged-in user is admin user then that user can see the admin tab in the navigation bar of the application. Once the user clicks on the admin tab then the user would be able to see all the tables that are present in the database. Using that screen the user could perform the Create, Read, Update, and Delete(CRUD) operations on any available table. The admin page of our application is shown below:



Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

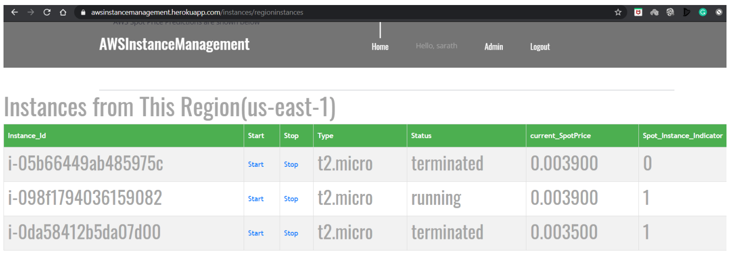
AWS		
C4cxlargetables	+ Add	Change
T2micros	+ Add	Change

INSTANCES		
Ami_creations	+ Add	Change
C4cxlargetables	+ Add	Change
Check_spots	+ Add	Change
Instance_creations	+ Add	Change

Figure 6: Admin Page

5.1.5 Instance Monitor Screen

Once the user provide a valid search parameters then this screen would appear to them as the search result. This screen would display a table with the status of every instance that is associated with the provided parameter's AWS account. Using this screen the user could monitor the running instances. The logic of this algorithm4.2.1 would be performed every ten seconds on this screen in order to enhance the reliability of the spot instance with optimal cost. Various parameters are presented in the table namely instance id, instance type, instance state, start, stop, current spot price, and spot instance indicator. Instance-id represents the id of that particular instance. Instance type would present the type of that particular instance. The instance state represents the current state of that instance. Start link would allow the user to start the instance if it is in the stopped state. Stop link would enable the user to stop the running instance. The current spot price would present the current spot price for that particular type of instance. The spot instance indicator would present whether the instance is a spot instance or not. The figure for instance monitor screen is shown below:


awsinstancemanagement.herokuapp.com/instances/regioninstances
AWSInstanceManagement
Home
Hello, sarath
Admin
Logout

Instances from This Region(us-east-1)

Instance_Id	Start	Stop	Type	Status	current_SpotPrice	Spot_Instance_Indicator
i-05b66449ab485975c	Start	Stop	t2.micro	terminated	0.003900	0
i-098f1794036159082	Start	Stop	t2.micro	running	0.003900	1
i-0da58412b5da07d00	Start	Stop	t2.micro	terminated	0.003500	1

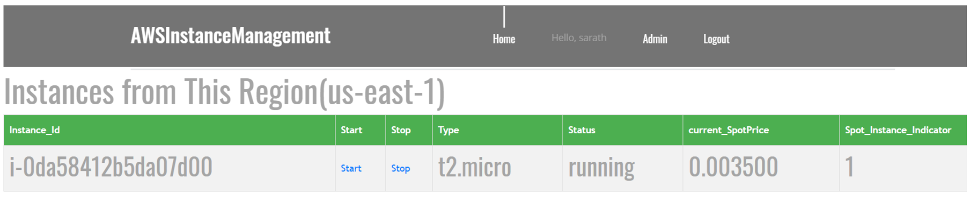
Figure 7: Instance Monitor Screen

6 Evaluation

This proposal was implemented with the AWS Instance Management web application and various evaluation was performed on it to ensure that the implementation is working as expected.

6.1 Spot Instance AMI Creation

To evaluate the spot instance AMI creation scenario the user should have a spot running spot instance in their AWS account. AWS instance management application would create the AMI of the running spot instance if the predicted spot price for the current hour of the particular instance is greater than the user bid price. After providing a valid search parameter the user would see the following result:

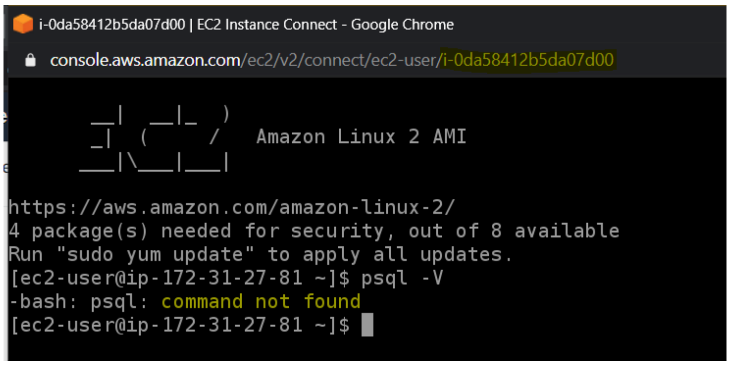


Instances from This Region(us-east-1)

Instance_Id	Start	Stop	Type	Status	current_SpotPrice	Spot_Instance_Indicator
i-0da58412b5da07d00	Start	Stop	t2.micro	running	0.003500	1

Figure 8: Running Spot Instance

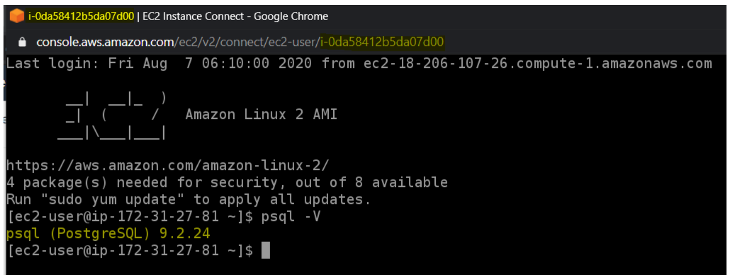
To evaluate the business continuity let us install the Postgres database in the running spot instance. The following figures will show the before and after installation of Postgres in the spot instance.



```
i-0da58412b5da07d00 | EC2 Instance Connect - Google Chrome
console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0da58412b5da07d00

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
4 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-27-81 ~]$ psql -V
-bash: psql: command not found
[ec2-user@ip-172-31-27-81 ~]$
```

Figure 9: Before Postgres Installation on Spot Instance

The above image shows the status of the spot instance, before installing the Postgres database. This command psql -V shows an error of command not found. Therefore Postgres is not present in the spot instance.

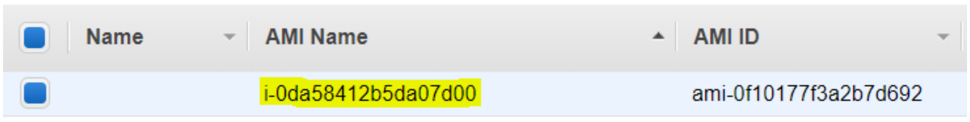


```
i-0da58412b5da07d00 | EC2 Instance Connect - Google Chrome
console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0da58412b5da07d00
Last login: Fri Aug  7 06:10:00 2020 from ec2-18-206-107-26.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
4 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-27-81 ~]$ psql -V
psql (PostgreSQL) 9.2.24
[ec2-user@ip-172-31-27-81 ~]$
```

Figure 10: After Postgres Installation on Spot Instance

The above image shows the status of the spot instance after installing the Postgres database. This command psql -V shows the version of the Postgres as Postgres 9.2.24. Therefore Postgres is present in the spot instance.

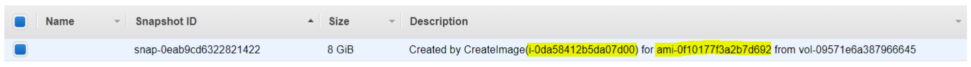
For the spot instance, AMI creation process evaluation let us consider the predicted spot price for t2.micro Linux/Unix is 0.0035 for that particular testing hour. The user provides a bid value of 0.0033. Since (0.0033¡0.0035) which is the user bid price is lower than the forecasted spot price. So AMI with a snapshot of the running spot instance should be created in AWS as shown as follows.



	Name	AMI Name	AMI ID
		i-0da58412b5da07d00	ami-0f10177f3a2b7d692

Figure 11: Spot Instance AMI Creation

The above image shows the created AMI id for the running spot instance. From the AMI name, we could find for which instance the AMI was created.



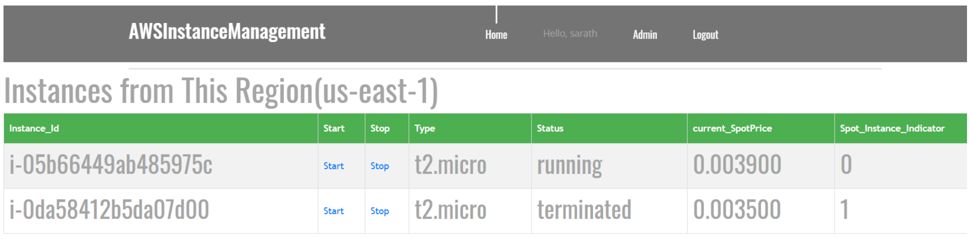
Name	Snapshot ID	Size	Description
	snap-0eab9cd6322821422	8 GiB	Created by CreateImage(i-0da58412b5da07d00) for ami-0f10177f3a2b7d692 from vol-09571e6a387966645

Figure 12: Spot Instance Snapshot Creation

The above image shows the created Snapshot id for the running spot instance. From the snapshot id and description column, we could identify which snapshot is associate with which AMI id.

6.2 On-Demand Instance Creation

To evaluate the on-demand instance creation process the running spot instance should be terminated. This would initiate a corresponding on-demand instance. Since the running spot instance of t2.micro is terminated AWS Instance Management application should create an equivalent t2.micro on-demand instance with the created spot instance AMI as follows:

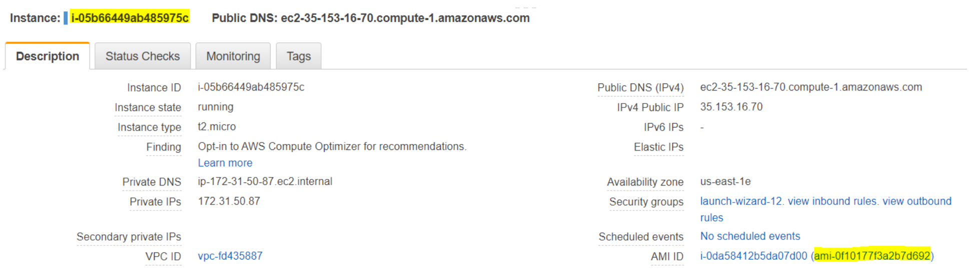


Instances from This Region(us-east-1)

Instance_Id	Start	Stop	Type	Status	current_SpotPrice	Spot_Instance_Indicator
i-05b66449ab485975c	Start	Stop	t2.micro	running	0.003900	0
i-0da58412b5da07d00	Start	Stop	t2.micro	terminated	0.003500	1

Figure 13: On-demand Instance Creation

From the above Figure, we could clearly see that an equivalent t2.micro on-demand instance is created for the terminated spot instance.



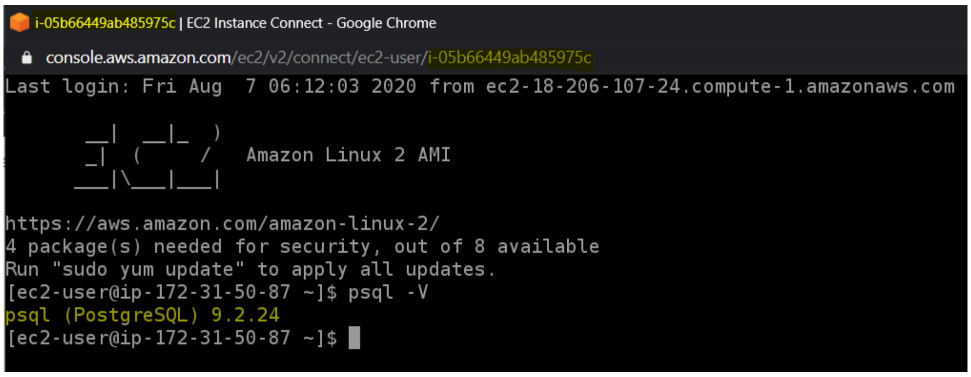
Instance: i-05b66449ab485975c **Public DNS: ec2-35-153-16-70.compute-1.amazonaws.com**

Instance ID	i-05b66449ab485975c	Public DNS (IPv4)	ec2-35-153-16-70.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	35.153.16.70
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-172-31-50-87.ec2.internal	Availability zone	us-east-1e
Private IPs	172.31.50.87	Security groups	launch-wizard-12. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-fd435887	AMI ID	i-0da58412b5da07d00 (ami-0f10177f3a2b7d692)

Figure 14: AMI of On-Demand Instance

From the above figure, we could cross-check whether the newly created spot instance AMI is used for created on-demand instance. The AMI used for this instance creation could be revealed from the above figure.

To evaluate the business continuity we could check for the Postgres installation in the newly created on-demand instance. The following figure provides a piece of evidence that the newly created on-demand instance has the Postgres database in it:



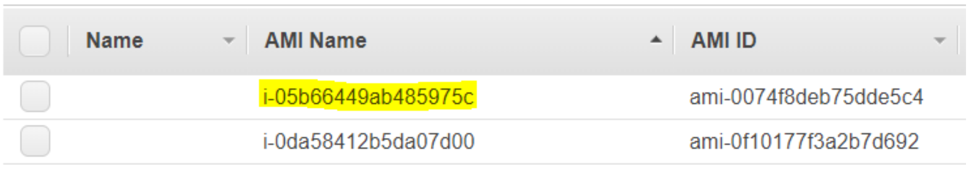
```
i-05b66449ab485975c | EC2 Instance Connect - Google Chrome
console.aws.amazon.com/ec2/v2/connect/ec2-user/i-05b66449ab485975c
Last login: Fri Aug  7 06:12:03 2020 from ec2-18-206-107-24.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
4 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-50-87 ~]$ psql -V
psql (PostgreSQL) 9.2.24
[ec2-user@ip-172-31-50-87 ~]$ 
```

Figure 15: Postgres Check in On-Demand Instance

The above image shows the status of the on-demand instance. This command psql -V shows the version of the Postgres as Postgres 9.2.24. Therefore Postgres is present in the on-demand instance. Hence the business continuity is achieved in a newly created on-demand instance.

6.3 Spot Instance Creation

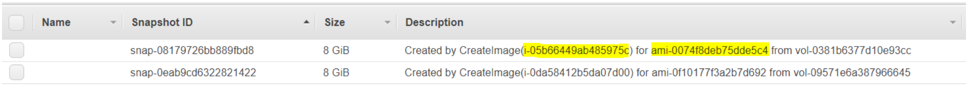
To evaluate spot instance creation condition the user needs to have a running on-demand instance as shown before. Once the current spot price is equal to the user bid price then AIM application would create an AMI of the running on-demand. For example, the value of the current spot price column of the running on-demand instance is 0.0039 and the user bid price is also 0.0039. Since (0.0039(current spot price)=0.0039(user bid price)) AIM application should create an AMI with a snapshot of the running on-demand instance as shown below:



Name	AMI Name	AMI ID
	i-05b66449ab485975c	ami-0074f8deb75dde5c4
	i-0da58412b5da07d00	ami-0f10177f3a2b7d692

Figure 16: On-Demand Instance AMI Creation

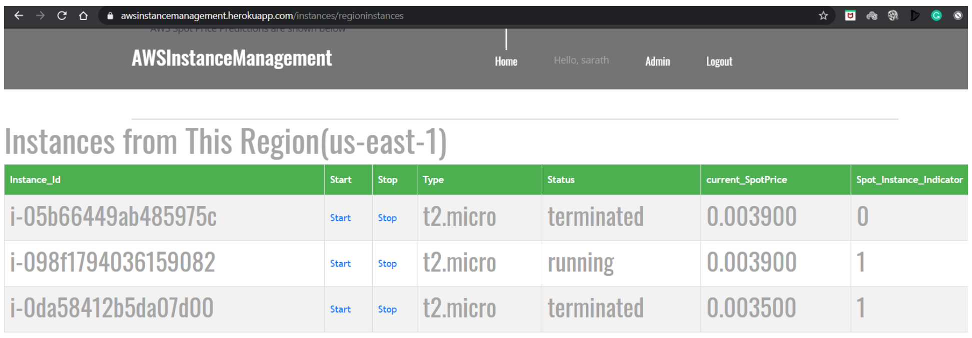
The above image shows the created AMI id for the running on-demand instance. From the AMI name, we could find for which instance the AMI was created.



Name	Snapshot ID	Size	Description
	snap-08179726bb889fbd8	8 GiB	Created by CreateImage(i-05b66449ab485975c) for ami-0074f8deb75dde5c4 from vol-0381b6377d10e93cc
	snap-0eab9cd6322821422	8 GiB	Created by CreateImage(i-0da58412b5da07d00) for ami-0f10177f3a2b7d692 from vol-09571e6a387966645

Figure 17: On-Demand Instance Snapshot Creation

The above image shows the created Snapshot id for the running on-demand instance. From the snapshot id and description column, we could identify which snapshot is associate with which AMI id.


awsinstancemanagement.herokuapp.com/instances/regioninstances

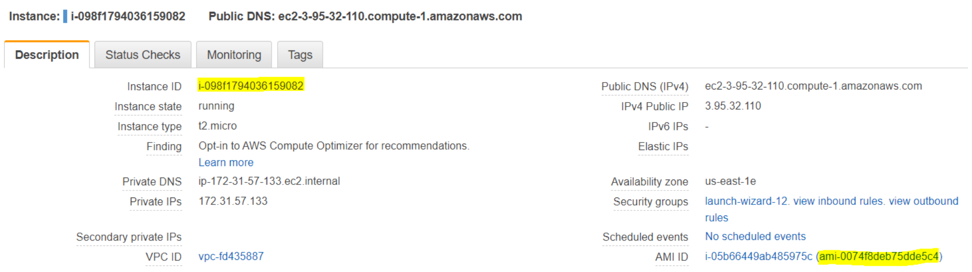
AWSInstanceManagement — Home — Hello, sarath — Admin — Logout

Instances from This Region(us-east-1)

Instance_Id	Start	Stop	Type	Status	current_SpotPrice	Spot_Instance_Indicator
i-05b66449ab485975c	Start	Stop	t2.micro	terminated	0.003900	0
i-098f1794036159082	Start	Stop	t2.micro	running	0.003900	1
i-0da58412b5da07d00	Start	Stop	t2.micro	terminated	0.003500	1

Figure 18: Spot Instance Creation

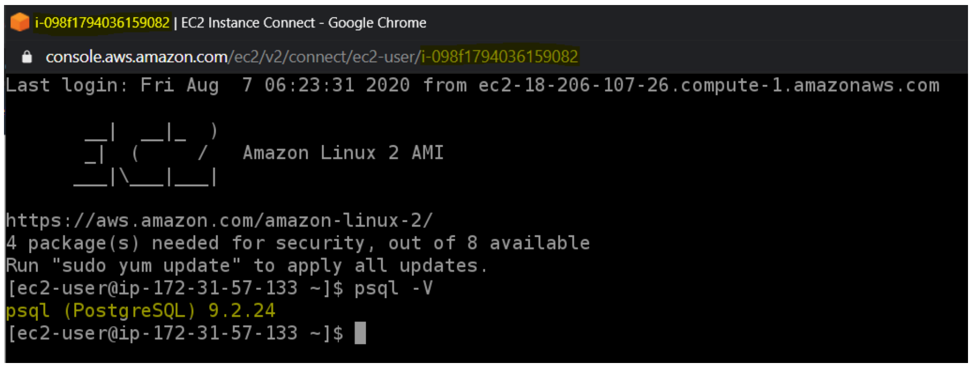
Once the AMI and snapshot are created then the AIM application should terminate the running on-demand instance and create an equivalent spot instance as shown above.


Instance: i-098f1794036159082 Public DNS: ec2-3-95-32-110.compute-1.amazonaws.com

Instance ID	i-098f1794036159082	Public DNS (IPv4)	ec2-3-95-32-110.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	3.95.32.110
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-172-31-57-133.ec2.internal	Availability zone	us-east-1e
Private IPs	172.31.57.133	Security groups	launch-wizard-12. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-fd435887	AMI ID	i-05b66449ab485975c (ami-0074f8deb75dde5c4)

Figure 19: Spot Instance Creation

From the above figure, we could cross-check whether the newly created on-demand instance AMI is used for created spot instance. The AMI used for this instance creation could be revealed from the above figure.

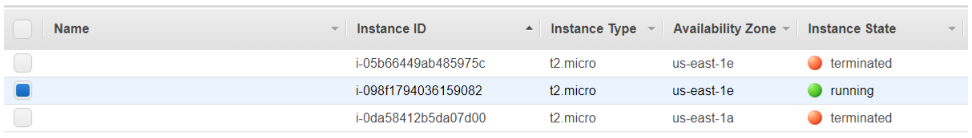

i-098f1794036159082 | EC2 Instance Connect - Google Chrome

console.aws.amazon.com/ec2/v2/connect/ec2-user/i-098f1794036159082

```
Last login: Fri Aug  7 06:23:31 2020 from ec2-18-206-107-26.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
4 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-57-133 ~]$ psql -V
psql (PostgreSQL) 9.2.24
[ec2-user@ip-172-31-57-133 ~]$
```

Figure 20: Spot Instance Creation

The above image shows the status of the spot instance. This command psql -V shows the version of the Postgres as Postgres 9.2.24. Therefore Postgres is present in the spot instance. Hence the business continuity is achieved in the newly created spot instance.



	Name	Instance ID	Instance Type	Availability Zone	Instance State
☐		i-05b66449ab485975c	t2.micro	us-east-1e	🔴 terminated
☑		i-098f1794036159082	t2.micro	us-east-1e	🟢 running
☐		i-0da58412b5da07d00	t2.micro	us-east-1a	🔴 terminated

Figure 21: Spot Instance Creation

The above figure could provide the status of each instance that is present in the AWS console. To cross-check whether every instance is created in the AWS environment or not.

6.4 Discussion

If the AWS Instance Management application is not used then the data associated with the spot instance would be lost and the business continuity of the service provided by the spot instance would be interrupted. But by using this proposal the end-user would be able to have the business continuity even after the termination of spot instance. The evaluation results have provided evidence to achieved business continuity. Spot instance would be created if the user bid price is equal to the current spot price in order to achieve higher cost optimization. Therefore with the evidence of each evaluation result I could conclude that the proposed idea could be implemented in the real cloud platform and it would enhance the business continuity and cost optimization.

7 Conclusion and Future Work

The principal intention of this research work is to assure the business continuity of the service that is provided by the AWS spot instance with the minimal possible cost. Therefore various modules like forecast module and spot instance failure recovery module are proposed. Two novel algorithms were proposed as a part of our research namely spot instance failure recovery algorithm and AMI creation algorithm. A web application was created with Django(python web application framework) to implement the proposed idea. This research work is evaluated on the AWS cloud platform. The evaluation section would provide evidence to the business continuity of the spot instance even after the termination with the possible minimal cost. This proposal has some limitations like it would be applicable only for the AWS users. AWs_access_key and AWS_secret_key are mandatory to make use of this web application. This application would work for the instance type that has the predicted value. In this research work, only a few variants of instance's spot price are forecast-ed. But this could be enhanced by forecasting the spot price for each possible instance type and availability zones.

References

Baughman, M., Haas, C., Wolski, R., Foster, I. and Chard, K. (2018). Predicting amazon spot prices with lstm networks, *Proceedings of the 9th Workshop on Scientific Cloud Computing*, ScienceCloud'18, Association for Computing Machinery.

Buyya, R. et al. (2019). A manifesto for future generation cloud computing: Research directions for the next decade, *ACM Comput. Surv.* **51**(5): 105:1–105:38.

Ekwe-Ekwe, N. and Barker, A. (2018). Location, location, location: Exploring amazon ec2 spot instance pricing across geographical regions, *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 370–373.

Fabra, J., Ezpeleta, J. and Álvarez, P. (2019). Reducing the price of resource provisioning using EC2 spot instances with prediction models, *Future Gener. Comput. Syst.* **96**: 348–367.

He, X., Shenoy, P., Sitaraman, R. and Irwin, D. (2015). Cutting the cost of hosting online services using cloud spot markets, *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '15, Association for Computing Machinery, p. 207–218.

Khandelwal, V., Chaturvedi, A. and Gupta, C. P. (2017). Amazon ec2 spot price prediction using regression random forests, *IEEE Transactions on Cloud Computing* pp. 1–1.

Khatua, S. and Mukherjee, N. (2013). A novel checkpointing scheme for amazon ec2 spot instances, *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pp. 180–181.

Pham, T., Ristov, S. and Fahringer, T. (2018). Performance and behavior characterization of amazon ec2 spot instances, *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 73–81.

Qu, C., Calheiros, R. N. and Buyya, R. (2016). A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances, *J. Netw. Comput. Appl.* **65**: 167–180.

Sabyasachi, A. S., Kabir, H. M. D., Abdelmoniem, A. M. and Mondal, S. K. (2017). A resilient auction framework for deadline-aware jobs in cloud spot market, *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 247–249.

Song, Y., Zafer, M. and Lee, K. (2012). Optimal bidding in spot instance market, *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, pp. 190–198.

Yi, S., Andrzejak, A. and Kondo, D. (2012). Monetary cost-aware checkpointing and migration on amazon cloud spot instances, *IEEE Transactions on Services Computing* **5**(4): 512–524.

Zheng, L., Joe-Wong, C., Tan, C. W., Chiang, M. and Wang, X. (2015). How to bid the cloud, *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, Association for Computing Machinery, p. 71–84.