Automation of Root Volume Expansion on EC2 Instance - AWS Cloud

Rani Chowdary Mandepudi, Shreya Ponuganti,
Haripriya Pillalamari and D Mohan

July 14, 2023

Automation of Root Volume Expansion on EC2 Instance - AWS Cloud

Mandepudi Rani Chowdary
Department of Electronics and Computer Engineering,
Sreenidhi Institute of Science and Technology,
Hyderabad, India
ranichowdarymandepudi@gmail.com

Shreya Ponuganti
Department of Electronics and Computer Engineering,
Sreenidhi Institute of Science and Technology,
Hyderabad, India
19311a19e2@sreenidhi.edu.in

Haripriya Pillalamari
Department of Electronics and Computer Engineering,
Sreenidhi Institute of Science and Technology,
Hyderabad, India
19311a19e1@sreenidhi.edu.in

Dr. D. Mohan
Department of Electronics and Computer Engineering,
Sreenidhi Institute of Science and Technology,
Hyderabad, India
Mohan.aryan19@sreenidhi.edu.in

Abstract*—As their fastened and rapid growth in the technological world, it is highly recommended to automate the functionalities by managing the risks and overcoming the issue of delay due to manual designing. There is higher growth, especially in the cloud domain, so there are many challenges for cloud architects, in which one of that is managing the root volume basing the demand or requirement. In such an outline, to overcome the manual increment of EBS volume in an EC2 instance, we came up with automation where the architecture is designed once and can be used till their requirement. The EC2 instance will be continuously monitored by the cloud watch agent and when the agent observes the change that meets the threshold passes it to the Python function which is running as a lambda function in the Amazon Web Services(AWS), with the support of Event Bridge. Now the lambda function is designed to carry forward the information to the step function that provides the error-handling function. Then the step function manages the Python codes that are running in the three lambda functions based on their priority. This brings out faster and more efficient results than done manually as the manual one might produce errors. Deploying this architecture in the industries would provide low latency and scope of automating the services as per the requirement much faster.

Keywords: Cloud, AWS, Automation, Low-latency

I. INTRODUCTION

In the current fast-growing cloud world, there is a huge demand for storage and this makes the duty of managing the storage quite challenging. Monitoring the volume of the external hard disk is the key role as it leads to data corruption in the real world, which produces a great loss to the IT industries. Apart from this, balancing multiple disks of storage in a vast number of instances is a highly difficult part of managing in the world of big data. So, to solve this problem and achieve the efficiency of producing the interface to store data without any hanging up and managing them in a smoother way can be easily resolved by our one-time architecture setup. This architecture can be efficiently implemented in all the cloud platforms without errors [1]. Moreover, resolving this problem and setting up the architecture is initially developed in the Amazon Web Services (AWS) cloud platform especially focusing on Linux systems as it is used highly in the IT domain. To be more precise, Amazon Linux holds the file system of "xfs". As commands are to be executed automatically, this architecture is file-system dependent. We are managing this architecture in the AWS cloud by a supporting algorithm that triggers the peak point which is the threshold value that is set as 90% of the current EBS volume in the AWS cloud. So, as an alert when the storage reaches the threshold value, it alerts the device through an alarm and this will be carried over by a step function which is a part of the AWS cloud that would help in creating a junction to the Lambda services. Now, we have implemented the three lambda functions that would help in the proper management of the architecture. So, the first lambda service tries to collect the server-attached root volume Id, then followed by the second lambda function tries to increase the EBS volume which is currently done manually through commands or CLI and the last lambda function is helpful in internal system storage increment. The three lambda services get the trigger from the master lambda services through the step function. Once the third lambda service completes its action, there will be an automated increment of EBS volume in the instance that reached the peak point [2]. This automation makes the job easier and increases the volume based on the requirement without any manual assistance as shown in Fig 1.

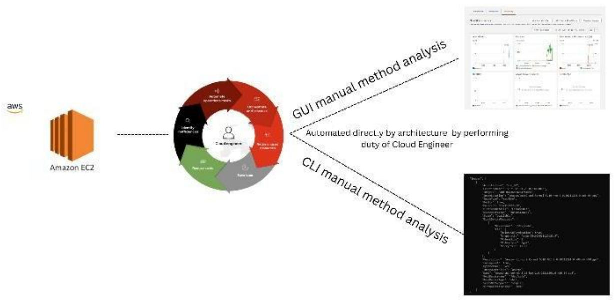

aws
Amazon EC2
GUI manual method analysis
Automated directly by architecture by performing duty of Cloud Engineer
CLI manual method analysis

Fig 1. Automating with the architecture without any manual assistance

II. LITERATURE SURVEY

The overview of cloud computing and its characteristics, such as on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. They also discuss the challenges associated with cloud data storage, including data security and privacy concerns. The authors then explore various cloud data storage models, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). They highlight the benefits of cloud data storage, such as cost-effectiveness, scalability, and accessibility. The article also discusses the architecture of cloud data storage and various approaches to implementing it. The authors conclude that cloud data storage technology has significant potential for improving data storage and management efficiency, but it requires a comprehensive understanding of cloud computing technology and appropriate implementation strategies to ensure its effectiveness. The article provides valuable insights into the potential benefits and challenges of cloud data storage and serves as a useful resource for researchers and practitioners interested in this field [3].

There is an in-depth analysis of cloud computing, its characteristics, and the challenges associated with cloud data storage. They also discuss various cloud data storage models and highlight their benefits and drawbacks. The article explores the architecture of cloud data storage and various approaches to implementing it, including centralized storage, distributed storage, and hybrid storage. The authors provide valuable insights into the potential benefits and challenges of cloud data storage, such as cost-effectiveness, scalability, accessibility, and data security. Overall, the article serves as an informative resource for researchers and practitioners interested in cloud data storage technology and its architecture implementation. It provides a clear understanding of the key concepts, challenges, and benefits associated with cloud data storage, and offers recommendations for effective implementation strategies [4].

The characteristics of cloud computing, including on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. They also explore the benefits of cloud storage, such as cost-effectiveness, scalability, and accessibility. The article focuses on online cloud storage technology and its architecture, including the components of online cloud storage systems, such as the storage service provider, user interface, data access protocol, and security mechanism. The authors also discuss the challenges associated with online cloud storage, such as data security, privacy, and compliance issues. Overall, the article provides a useful overview of online cloud storage technology and its applications. It serves as a valuable resource for researchers and practitioners interested in cloud storage technology and provides insights into the potential benefits and challenges of online cloud storage [5].

The increasing demand for virtual storage and the challenges associated with load balancing in virtual storage systems. They introduce the LBVS (Load Balancing for Virtual Storage) strategy, which is designed to optimize resource utilization and improve system performance. The article explains the architecture of the LBVS strategy and discusses its key components, including the storage pool manager, the load balancing algorithm, and the performance evaluation module. The authors provide a detailed analysis of the LBVS strategy's performance and compare it to other load balancing strategies. Overall, the article serves as a valuable resource for researchers and practitioners interested in load balancing strategies for virtual storage. It provides insights into the challenges associated with virtual storage and presents a promising solution to improve system performance and resource utilization [6]

III. IMPLEMENTATION

A. *Elastic Block Storage (EBS) Volume - Cloud Platform:*

As an initial setup, this is implemented in the Amazon Web Services (AWS) cloud platform due to its huge demand in the current market. But by following the same steps it can be very easily implemented in other cloud platforms like Microsoft Azure, and Google Cloud Platform by using their services itself. Amazon Web Services (AWS) provides a vast number of services that would help in building different architectures as per the requirement very effectively. So, the hard disk storage which is called Elastic Block Storage (EBS), is the external hard disk. EBS helps in providing block-level storage volumes that can be operationally attached to the EC2 instances in AWS. Increasing this volume can be done by the Command Line Interface or through AWS Graphical Interface. But to do this, there should be manpower for continuous watching of the increment in the volume space which is often difficult. This can be monitored easily once this setup is ready. Elastic Cloud Compute (EC2) is an instance for which EBS volume is attached as similarly hard disk is attached to our systems. So, while launching an instance there is an option of choosing the volume (hard disk space) which is static for a point in time till it is increased manually [7]. So, once the instance is launched it provides the details of it along with its CPU monitoring details. AWS has an extraordinary feature of displaying the hard disk space consumption, and CPU utilization of each instance through logs and metrics or can check them through command as shown in Fig 2. These metrics help us in knowing clearly the status of every instance running very clearly.

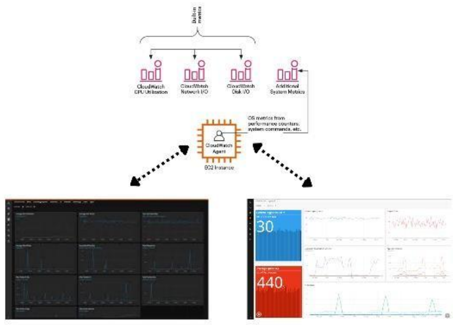

Built-in metrics
CloudWatch CPU Utilization
CloudWatch Network I/O
CloudWatch Disk I/O
Additional System Metrics
OS metrics from performance counters, system commands, etc.
CloudWatch Agent
EC2 Instance
30
440

Fig 2. Cloud Watch agent retrieves information and build up its metrics as per the logic

B. *Revolutionizing Business Operations with AWS Cloud Computing: Cloud Watch:*

As per the time, the required details of the instance are clearly provided by the EC2 instance. In the cloud watch, an alarm is created based on the condition that 'disk_used_percent is greater than 90 percent'. If the condition is true, then the alarm gets alerted which will be as an output state set to true. This alarm is the metrics alarm which will be analyzing the EC2 instances continuously. Once this alarm becomes high, this will be triggered by the Event Bridge service of Amazon. In Cloud Watch, the analysis graph helps us to exactly observe the percentage of the usage in an extraordinary view as shown in Fig 3. Initially when the data is below the threshold the color of the graph line will be blue and then once the threshold is reached, it changes to red as an alert signal [8]. This intelligent cloud watch alarm helps in detecting and acts faster than humans observing the peak point. Moreover, the cloud watch provides the Event Bridge rule for responding to the state change of the alarm.

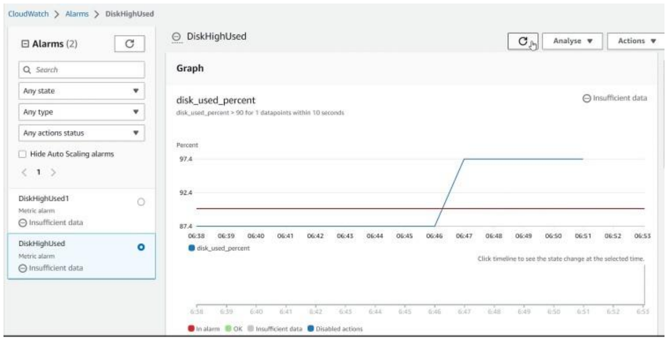
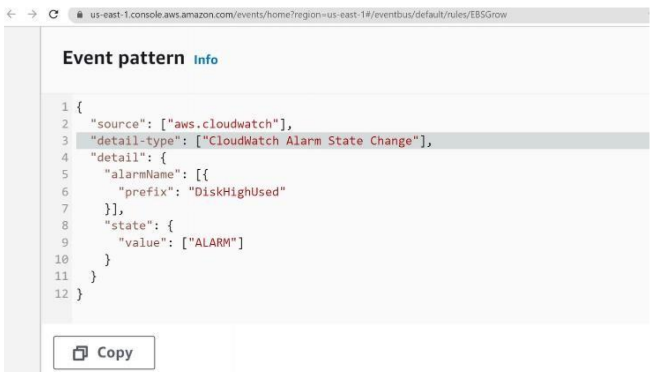


Fig 3 Cloud Watch analysis of disk usage

By using the rule provided in the Cloud Bridge as shown in Fig 4, we can create the Event Bridge rule to carry the state change information further.


us-east-1.console.aws.amazon.com/events/home?region=us-east-1#/eventbus/default/rules/EBSGrow
Event pattern Info
Copy

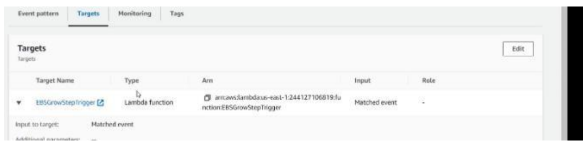
```
{
  "source": ["aws.cloudwatch"],
  "detail-type": ["CloudWatch Alarm State Change"],
  "detail": {
    "alarmName": [{
      "prefix": "DiskHighUsed"
    }],
    "state": {
      "value": ["ALARM"]
    }
  }
}
```

Fig 4 Cloud Watch event pattern analyzed from EC2 instance

C. *Amazon Event Bridge:*

The Amazon Event Bridge acts as a serverless service that uses the events information to connect the various components/services together that is basically acting as a bridge between applications by bridging the applications making us easier to drive out the scalable methodologies. By acting as a central hub for all events occurring within an AWS environment. EventBridge makes us easier to construct event-driven architectures.

In the Event Bridge, by creating the rule, we need to add the pattern of the event that is to be performed which will be in the JSON format [9]. In this event pattern, once the value it needed comes from the Cloud Watch Alarm, it starts to perform its action that is synced in its targets. So, in the targets, we add a lambda function that helps in passing out the information. The event targets look as shown in Fig 5.



Targets
Targets

	Target Name	Type	Arn	Input	Role
▼	EBSGrowStepTrigger	Lambda function	arn:aws:lambda:us-east-1:244127106819:function:EBSGrowStepTrigger	Matched event	-

Input to target: Matched event

Fig 5 Event targets that are connected with Lambda Function

D. *Lambda Function and Step Function:*

AWS Lambda is an event-driven, serverless computing technology that is part of Amazon Web Services. With Lambda, there is no need to concern yourself with selecting the appropriate AWS resources or managing them efficiently [10]. Instead, you can simply upload your code to Lambda, where it will be executed automatically in response to events in various AWS services, such as file uploads to S3 buckets or HTTP requests via Amazon API Gateway.

It is worth noting that AWS Lambda is designed primarily for background tasks. Although it can respond to events triggered by different AWS services, it is not suitable for running interactive or long-running applications.

In the current project, the lambda function of AWS is widely used for code management. The wider advantage of using this lambda function is its functionality which runs our code when the event is triggered and stays ideal at rest of the time. Overall, four lambda functions are used in the complete architecture [11]. The first one is used for transmission of data from Event Bridge to Step Function as we cannot send data or boolean value directly to the step function [12]. So, this lambda function triggers the step function whenever the event bridge passes and pushes this lambda to run. The lambda function code is written in Python V3 and the whole information of EBS will be attached to this code.

Now the step function is used especially for error handling and proper management of the three lambda functions. The step function handles the lambda function and helps in triggering them one after the other without destroying or hanging up the architecture [13]. The three lambda functions are basically designed to perform the actions that are manually done by humans currently in real-time and that is operated by the Python code by using the Linux commands as shown in Fig 6.

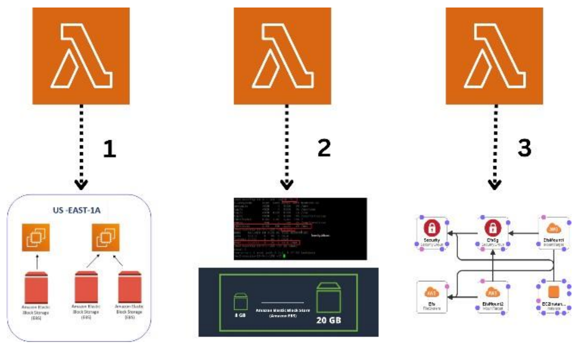


Fig 6. Lambda Functionality in increasing the root volume

IV. ARCHITECTURE

A. *Revolutionizing Business Operations with AWS Cloud Computing:*

The whole architecture starts with connecting the EC2 instance to the cloud watch alarm for monitoring the instance. The cloud watch alarm continuously captures the CPU consumption, storage space metrics and displays them, and notifies when the threshold value is reached [14]. This notification is carried over to the lambda function by the event bridge. The lambda function analyses the EC2 and carries forwards it to the next step which will be triggered by the step function. This step function now actually manages the error handling and efficiently manages the steps to perform one after the other. The increase of root volume is done manually in the current days through commands [15]. So, now these commands will be managed by the lambda function with the help of the step function which automatically increases the root volume and helps in the partitioning of the instance without any help of manual power as shown in Fig 7.

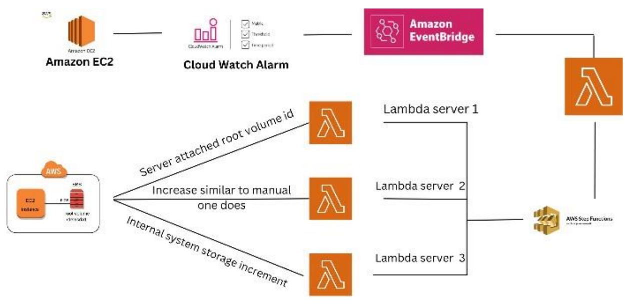


Fig 7. Automation Architecture of the system

B. *Flow Chart - Error Handling:*

To handle the errors, the step function is used. The step function manages the three lambda functions for which each one does its functionality as mentioned. The first lambda function that the step function handles is for ebs_id capturing, the second function is for growing the ebs_space and the third one is for growing fs(internal partition). With the help of the pipeline, we manage the step function as shown below [16]. Moreover, the architecture of the step function sends a failure message when the event fails or moves to the end when succeeded as shown in Fig 8.

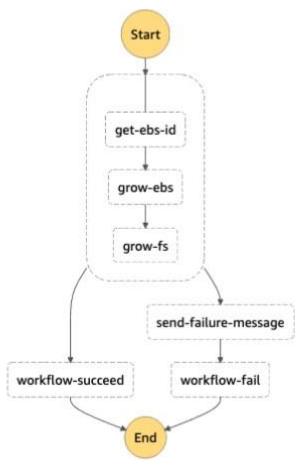


Fig 8. Step Function Error Handling Flow chart

C. *Flow Chart:*

The complete process begins with installing the cloud watch agent inEC2 as that is the one that monitors the processing of the instance. The agent creates a pattern in the cloud watch for the condition of threshold reach and once it occurs, the event bridge triggers and move that to the lambda functionality for the further process. This lambda pipeline of Python code set up and connects by triggering the step function [17]. To the step function, the SSM document is created and populated for ebs id and fs. The EBS details are as shown in Fig. Once the step function completes its process, either there will be the root volume expansion done or else it populates the failure message and comes to the end as shown in the flowchart Fig 8.

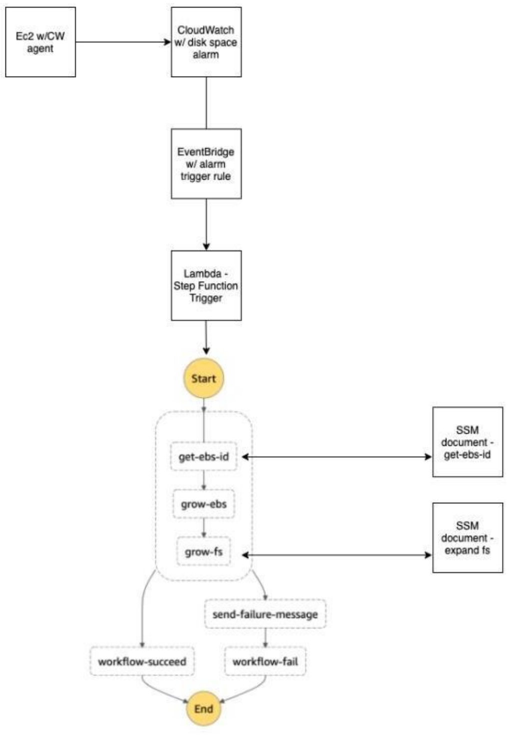
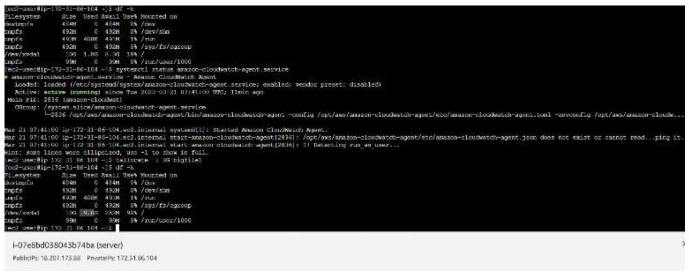


Fig 9. Architectural Flow chart – analysis of the design

V. RESULTS

Once the installation of the alarm agent in the EC2 and the architecture are set up then the automation gets started. So, initially, when we run the command in the instance and check the hard disk (EBS) volume space is ….. Now using the command 'fallocate' we have added the dummy data and then tested it as shown in the Fig 9.



```
[ec2-user@ip-172-31-86-104 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        484M     0  484M   0% /dev
tmpfs           492M     0  492M   0% /dev/shm
tmpfs           492M  408K  491M   1% /run
tmpfs           492M     0  492M   0% /sys/fs/cgroup
/dev/xvda1       10G  1.8G  8.3G  18% /
tmpfs            99M     0   99M   0% /run/user/1000
[ec2-user@ip-172-31-86-104 ~]$ systemctl status amazon-cloudwatch-agent.service
● amazon-cloudwatch-agent.service - Amazon CloudWatch Agent
   Loaded: loaded (/etc/systemd/system/amazon-cloudwatch-agent.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2023-03-21 07:41:00 UTC; 11min ago
 Main PID: 2836 (amazon-cloudwat)
   CGroup: /system.slice/amazon-cloudwatch-agent.service
           └─2836 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml -envconfig /opt/aws/amazon-cloudw...

Mar 21 07:41:00 ip-172-31-86-104.ec2.internal systemd[1]: Started Amazon CloudWatch Agent.
Mar 21 07:41:00 ip-172-31-86-104.ec2.internal start-amazon-cloudwatch-agent[2836]: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json does not exist or cannot read...ping it.
Mar 21 07:41:00 ip-172-31-86-104.ec2.internal start-amazon-cloudwatch-agent[2836]: I! Detecting run_as_user...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-86-104 ~]$ fallocate -l 8G bigfile1
[ec2-user@ip-172-31-86-104 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        484M     0  484M   0% /dev
tmpfs           492M     0  492M   0% /dev/shm
tmpfs           492M  408K  491M   1% /run
tmpfs           492M     0  492M   0% /sys/fs/cgroup
/dev/xvda1       10G  9.8G  253M  98% /
tmpfs            99M     0   99M   0% /run/user/1000
[ec2-user@ip-172-31-86-104 ~]$
```

i-07e8bd038043b74ba (server)
PublicIPs: 18.207.173.88   PrivateIPs: 172.31.86.104

Fig 9. Initial EBS volume and volume after addition of dummy data

Now when we check the analysis graph, it reaches the peak point, which is the threshold as per our condition as shown in Fig 10.

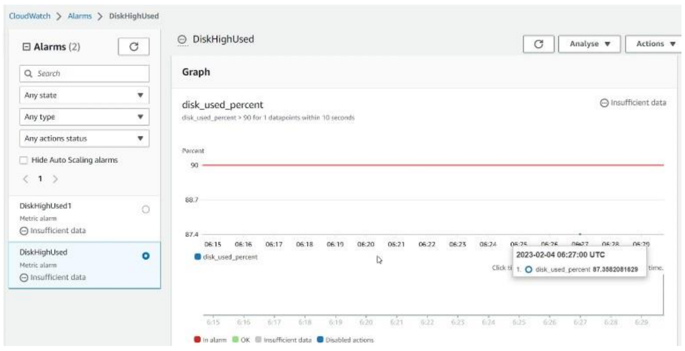
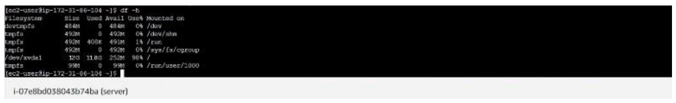


Fig 10. Cloud Watch analysis when threshold value is obtained

So, as it crossed 90% of usage the cloud agent triggers and then alerts lambda which then triggers and connects to the step function and further completes the process successfully. Now when we check through command 'df -h' in the console, we observe the increment of 20% of external space in the EBS volume which is 1.2 times of actual hard disk volume but it is dynamic which can be changed as per the requirement as in the Fig 11.



```
[ec2-user@ip-172-31-86-104 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        484M     0  484M   0% /dev
tmpfs           492M     0  492M   0% /dev/shm
tmpfs           492M  408K  491M   1% /run
tmpfs           492M     0  492M   0% /sys/fs/cgroup
/dev/xvda1       12G   11.8G 252M  98% /
tmpfs            99M     0   99M   0% /run/user/1000
[ec2-user@ip-172-31-86-104 ~]$
```

i-07e8bd038043b74ba (server)

Fig 11. Space increment and parallel high volume for expanding

The challenge with this automation in the AWS cloud is, the EBS volume of AWS can be only increased once after it completes 24 hours of the previous increase. This is the restriction from AWS cloud that cannot be solved at this point in time as their services are designed and can be used by following the restrictions.

VI. CONCLUSION

The design is exclusively designed and developed for upgrading and enhancing automation in the domain of the cloud by meeting the standards of the emerging IT world. As there are many challenges to the cloud for proper monitoring and managing we have come up with solving one of them by producing effective results. Even more, the proposed architecture has the capability to deploy in a fraction of a second and much faster than manual increasing of root volume as per the tests performed. The Architecture provides a smoother, more efficient, reliable way of analyzing the storage and automating the extension of root volume. There is no requirement for cloud admins for managing this in the real world and also stores the snapshot as a safety measure. Snapshots provide an additional advantage as there will be no fear of data loss to the companies as data is very precious. Snapshots are an add-on advantage that could be a support. The usage of this automation would make AWS a much wider extension compared to the present analysis.

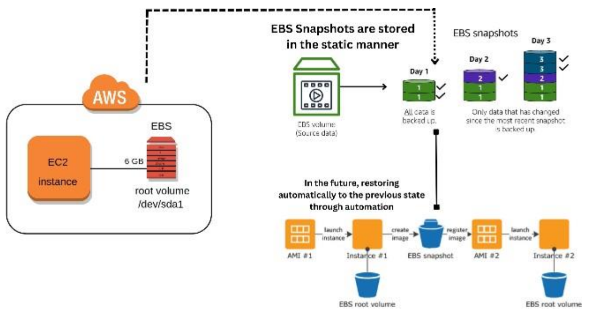


Fig 13. Current snapshot storage in database & future automation using AWS services

As per the future improvements, we will provide the design of the same architecture for all types of file systems and operating systems as we got effective results for this project. Additionally, replicates the whole architecture in different cloud service providers like Azure and GCP. Moreover, if the architecture gets corrupted in any of the cases, basing the snapshots, we try getting the whole instance to its previous state by providing the report for the failure where the testing is also automated and produces the test results to correct those test cases to get the action to be performed.

References

[1] M. M. Rovnyagin, V. V. Odintsev, D. Y. Fedin and A. V. Kuzmin, "Cloud computing architecture for high-volume monitoring processing," 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow and St. Petersburg, Russia, 2018, pp. 361-365, doi: 10.1109/EIConRus.2018.8317107.

[2] Yuan, Zizhen & Wen, Chengyu & Zhang, Xiaoli. (2019). Research on Cloud Hard Disk Capacity Prediction Scheme Based on Time Series Model. International Journal of Computer Applications Technology and Research. 8. 29-32. 10.7753/IJCATR0801.1006.

[3] Liu, Kun & Dong, Long-jiang. (2012). Research on Cloud Data Storage Technology and Its Architecture Implementation. Procedia Engineering. 29. 133–137. 10.1016/j.proeng.2011.12.682.

[4] Liu, K. and Dong, L.J., 2012. Research on cloud data storage technology and its architecture implementation. Procedia Engineering, 29, pp.133-137.

[5] S. -H. Zou, N. -S. Fang and W. -J. Gao, "Research on online cloud storage technology," 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Xuzhou, China, 2020, pp. 62-65, doi: 10.1109/DCABES50732.2020.00025.

[6] Liu, H., Liu, S., Meng, X., Yang, C. and Zhang, Y., 2010, May. LBVS: A load balancing strategy for virtual storage. In 2010 International Conference on Service Sciences (pp. 257-262). IEEE.

[7] He, Q., Li, Z. and Zhang, X., 2010, October. Analysis of the key technology on cloud storage. In 2010 International Conference on Future Information Technology and Management Engineering (Vol. 1, pp. 426-429). IEEE.

[8] Liang, X.Y., 2014. Analysis on Non-Center Cloud Storage Architecture of Gluster. In Applied Mechanics and Materials (Vol. 587, pp. 2346-2349). Trans Tech Publications Ltd.

[9] Hai, J., 2016, March. Network cloud storage service architecture analysis and research. In 2016 Eighth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA) (pp. 413-416). IEEE.

[10] K. Zhou et al., "LEA: A Lazy Eviction Algorithm for SSD Cache in Cloud Block Storage," 2018 IEEE 36th International Conference on Computer Design (ICCD), Orlando, FL, USA, 2018, pp. 569-572, doi: 10.1109/ICCD.2018.00091.

[11] B. Ravandi and I. Papapanagiotou, "A Self-Learning Scheduling in Cloud Software Defined Block Storage," 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honololu, HI, USA, 2017, pp. 415-422, doi: 10.1109/CLOUD.2017.60.

[12] J. Spillner and J. Müller, "Tutorial on Distributed Data Storage: From Dispersed Files to Stealth Databases," 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, London, UK, 2014, pp. 535-536, doi: 10.1109/UCC.2014.80.

[13] F. Jiang, Y. Cheng, B. Qin and Z. Hui, "Who owns the address? A performant block storage model for hybrid cloud," 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS), Takamatsu, Japan, 2022, pp. 1-6, doi: 10.23919/APNOMS56106.2022.9919977.

[14] D. Yang and C. Ren, "VCSS: An Integration Framework for Open Cloud Storage Services," 2014 IEEE World Congress on Services, Anchorage, AK, USA, 2014, pp. 155-160, doi: 10.1109/SERVICES.2014.36.

[15] H. T. Lu, C. H. Kao, P. H. Wu and Y. H. Lee, "Towards a hosted private cloud storage solution for application service provider," Proceedings of 2014 International Conference on Cloud Computing and Internet of Things, Changchun, China, 2014, pp. 82-84, doi: 10.1109/CCIOT.2014.7062510.

[16] J. Spillner and J. Müller, "PICav: Precise, Iterative, and Complement-Based Cloud Storage Availability Calculation Scheme," 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, London, UK, 2014, pp. 443-450, doi: 10.1109/UCC.2014.56.

[17] S. Nepal, C. Friedrich, L. Henry and S. Chen, "A Secure Storage Service in the Hybrid Cloud," 2011 Fourth IEEE International Conference on Utility and Cloud Computing, Melbourne, VIC, Australia, 2011, pp. 334-335, doi: 10.1109/UCC.2011.55.