JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Automated Deployment on Multiple instances using Auto Scaling

Aditya Narayan Mundhe[1], Atharva Atul Yeole[2], Rushikesh Umesh Kanhegaokar[3], Dr.Jayashri Shinde[4]

Department of Information Technology,

G.H.Raisoni College of Engineering and Management, Pune, India

Abstract: In this Project We have worked on automated deployment on multiple instances using auto scaling. Here we used AWS Cloud Computing Platform.Autoscaling is one of the service of AWS.AWS Auto Scaling is a service that helps the user to monitor applications and automatically adjusts the capacity to maintain steady, predictable performance at the lowest possible cost.We are going to achieve Scalability by increasing the numbers of amazon ec2 instances when customer demand will increase unexpectedly.

Index Terms **– AWS(Amazon Web Service), Auto Scaling**

1.INTRODUCTION

In this Project we are using Cloud Computing Platform. Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS). Organizations of every type, size, and industry are using the cloud for a wide variety of use cases, such as data backup, disaster recovery, email, virtual desktops, software development and testing, big data analytics, and customer-facing web applications. For example, healthcare companies are using the cloud to develop more personalized treatments for patients. Financial services companies are using the cloud to power real-time fraud detection and prevention. And video game makers are using the cloud to deliver online games to millions of players around the world.

1.1 OVERVIEW

❖ Existing System and Need for System

This project requires highly available systems because they are reliable in the sense that they continue operating even when critical components fail. They are also resilient, meaning that they are able to simply handle failure without service disruption or data loss, and

seamlessly recover from such failure. So, the following elements help you implement highly available systems:

• Redundancy—ensuring that critical system components have another identical component with the same data, that can take over in case of failure.

• Monitoring—identifying problems in production systems that may disrupt or degrade service. Failover—the ability to switch from an active system component to a redundant component in case of failure, imminent failure, degraded performance or functionality.

• Failback—the ability to switch back from a redundant component to the primary active component, when it has recovered from failure. To achieve this, we used services of AWS.

❖ Scope of Work

As we store all the data of our clients, we have to make sure that they are secure and highly available. Highly available systems are reliable in the sense that they continue operating even when critical components fail. They are also resilient, meaning that they are able to simply handle failure without service disruption or data loss, and seamlessly recover from such failure. High availability is commonly measured as a percentage of uptime. The number of "nines" is commonly used to indicate the degree of high availability. For example, "four nines" is indicative of a system that is up 99.99% of the time, meaning it is down for only 52.6 minutes during an entire year. The following elements help you implement highly available systems:

• Redundancy—ensuring that critical system components have another identical component with the same data, that can take over in case of failure.

• Monitoring—identifying problems in production systems that may disrupt or degrade service.

• Failover—the ability to switch from an active system component to a redundant component in case of failure, imminent failure, degraded performance or functionality.

• Failback—the ability to switch back from a redundant component to the primary active component, when it has recovered from failure. AWS helps you achieve high availability for cloud workloads, across three different dimensions:

• Compute—Amazon EC2 and other services that let you provision computing resources, provide high availability features such as load balancing, auto-scaling and provisioning across Amazon Availability Zones (AZ), representing isolated parts of an Amazon data center.

• SQL databases—Amazon RDS and other managed SQL databases provide options for automatically deploying database with a standby replica in a different AZ. 10

• Storage services—Amazon storage services, such as S3, EFS and EBS, provide built-in high availability options. S3 and EFS automatically store data across different AZs, while EBS enables deployment of snapshots to different AZs. If you are running instances on Amazon EC2, Amazon provides several built-in capabilities to achieve high availability:

• Elastic Load Balancing—you can launch several EC2 instances and distribute traffic between them.

• Availability Zones—you can place instances in different AZs.

• Auto Scaling—use auto-scaling to detect when loads increase, and then dynamically add more instances. These capabilities are illustrated in the diagram below. The Elastic Load Balancer distributes traffic between two or more EC2 instances, each of which can potentially be deployed in a separate subnet that resides in a separate Amazon Availability Zone. These instances can be part of an Auto-Scaling Group, with additional instances launched on demand.

1.2 WHAT IS HAPPENING HERE?

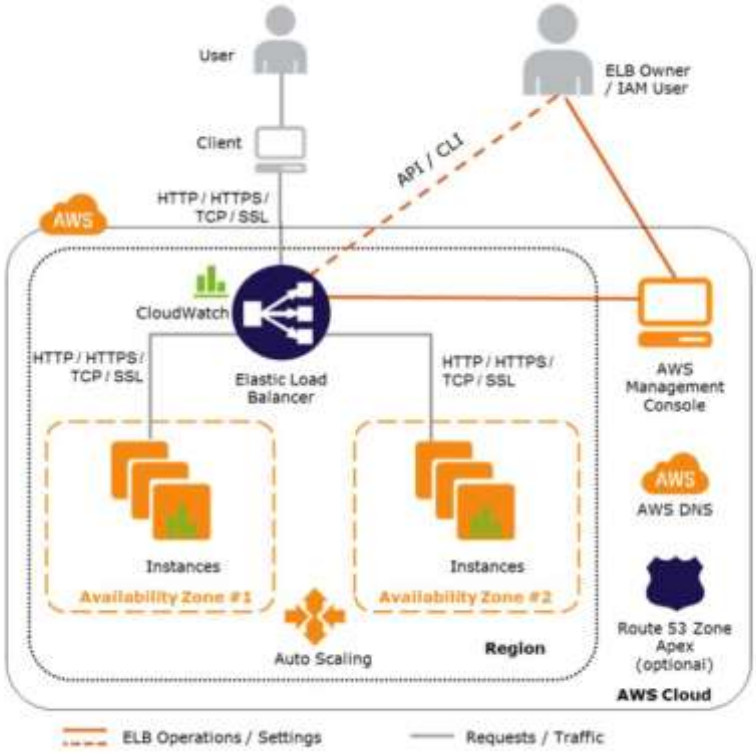

Fig.: System Architecture

Following is the step-by-step deployment of Application/ Website which will be in highly available environment using automated deployment of instances using auto scaling.

Step 1. Create Bucket: to upload application/code. In the Buckets list, choose the name of the bucket that you want to upload your folders or files to. To upload your objects, choose Upload. Amazon S3 uploads your object. When the upload completes, you can see a success message on the Upload: status page.

Step 2. Create IAM Role For ec2 and s3 connection give s3 full access policy. One service(ec2) can call another service(s3) on behalf of you.

Step 3. In the Amazon EC2 Dashboard, choose "Launch Templates" to create a launch template, specifying a name, AMI, instance type, and other details. launch configuration -> select aim -> select t2 micro -> click on advanced setting and upload your script in user data section. Create a shell script to automate instance configuration, put the script in user data. we can use user data to auto configure the instances.it supports text/scripts. assign security group -> provide key details -> and create launch configuration.

Step 4. Using the Auto Scaling wizard, create an Auto Scaling group specifying a name, size, and network for your Auto Scaling group. auto scale group-> provide an auto scale group name -> click on "switch to launch configuration " -> then select previously created launch conf template from drop down - > select all the subnets (to manage HA) - > choose existing load balancer -> Group size -> max count: 5, min& desired count: 2 -> Scaling policies -> Target tracking scaling policy. select CPU utilization -> target value: 70, Instance need:60 second.

2.PROPOSED SYSTEM

2.1 PROPOSED SYSTEM

This application is used to managed auto scaling the application. The client can access the website. Everything will be managed by AWS itself. This application provides the administrator with a facility to add new exams. This application provides the instructor add questions to the exam, modify questions in the exam in a particular exam. This application takes care of authentication of the administrator, Instructor as well as the student.

2.2 OBJECTIVES OF SYSTEM

To Manage load between the servers.
Servers can present between application load balancer so it can manage properly.
Automatically scale up and scale down the resources depending on CPU utilizations.
For hosting the application, we can use lambda function as well as other AWS services.
AWS account will only access by otherized person because of the security reason.

2.3 TOOL and TECHNOLOGIES

2.3.1 MODULES

- Login/Authentication
- Application
- Coding

2.3.2 SOFTWARE

- AWS(Amazon Web Services)

2.3.3 TECHNOLOGIES

- Cloud Computing
- Auto Scaling

3. EFFECT OF YOUR PROJECTION ENERGY CONSERVATION, ECONOMY & ENVIRONMENT

The most important is that every client will interact with computer, and they can easily access websites.
The most important is that cutting paper is saving because no use of paper is doing here.
Therefore, millions of tons of paper will save, if paper save then trees are also saved.
Nearly future online exam system portal will be easily saved time of students and other issues will be solved. Today every government exam or any other competitive exam is held by online this indicated that nearly future demand of online exam system website is increased.
And another issue solved by using online exam system is that billions of rupees are saved, because making a paper are complex issue and time consuming also.

4. LIMITATIONS AND DRAWBACKS

Amazon Elastic Compute Cloud service has a defined limit on the number of EC2 instances that a user can group under a particular auto scaling group. Users should be constantly keeping a check on this limit so that it does not get crossed before they could get hold of it.

The following are the default limits for EC2 auto scaling groups, defined by AWS:

- Auto Scaling groups per region: 200.
- Scaling policies per Auto Scaling group: 50.
- Scheduled actions per Auto Scaling group: 125.
- Lifecycle hooks per Auto Scaling group: 50.
- SNS topics per Auto Scaling group: 10.
- Classic Load Balancers per Auto Scaling group: 50.
- Target groups per Auto Scaling group: 50.

5. CONCLUSION

Hence High availability is an important subset of reliability engineering, focused towards assuring that a system or component has a high level of operational performance in a given period of time.
The primary goal of a high availability system is to prevent and eliminate all single points of failure. This should include multiple action plans that have been tested and in place, ready to react independently and immediately to any and all service disturbances, disruptions, and failures.
This includes hardware, software, and application irregularities. The eradication of downtime can be accomplished with the composed, skilled planning and implementation of a system. A critical eye is required to envision and prepare for any occurrence or disaster, which could impede the primary objective of the stated and expected uptime goal. A well instituted High Availability system can achieve this target with proper planning and design, reducing or eliminating disruptions and maximizing availability.
Careful Planning + Reliable Implementation Methodologies + Stable Software Platforms + Sound Hardware Infrastructure + Smooth Technical Operations + Prudent Management Goals + Consistent Data Security + Predictable Redundancy Systems + Robust Backup Solutions + Multiple Recovery Options = 100% Uptime. At a first glance, its implementation might seem quite complex; however, it can bring tremendous benefits for systems that require increased reliability.

6. ACKNOWLEDGMENT

We would like to thank our teacher Prof.Apashabi Pathan for helping us with this project.
We would also want to thank G.H.Raisoni College of Engineering and Management, Pune and Our parents for their moral support.

7. REFERENCES

- https://stackoverflow.com/questions/57563603/lambda-script- update-to-change-instance-size
- https://aws.amazon.com/elasticloadbalancing/sla/
- https://www.bmc.com/blogs/aws-ebs-elasticblock-store/
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using- volumes.html

8. BIBLIOGRAPHY

https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-healthchecks.html
https://stackoverflow.com/questions/57563603/lambda-script-update-to-change-instance-sizehttps://aws.amazon.com/elasticloadbalancing/sla/
https://www.bmc.com/blogs/aws-ebs-elastic-block-store/
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html