Amazon Elastic Beanstalk

SHREE GOVIND MISHRA[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: shremish@indiana.edu

project-000, April 30, 2017

Amazon Elastic Beanstalk is a service provided by the Amazon Web Services which allow developers and engineers to deploy and run web applications in the cloud, in such a way that these applications are highly available and scalable. Elastic Beanstalk manages the deployed application by reducing the management complexities as it automatically handles the capacity provisioning, load balancing, scaling, and application health monitoring. Elastic BeanStalk also provisions one or more AWS Resources such as Amazon EC2 instances when an application is deployed.

Keywords: Cloud, I524, Amazon Web Services, AWS Elastic Beanstalk, Load Balancing, Cloud Watch, AWS Management Console, Auto Scaling

https://github.com/cloudmesh/sp17-i524/raw/master/paper2/S17-IR-2021/report.pdf

1. INTRODUCTION

Cloud Computing can be defined as an abstraction of services from infrastructure (i.e. hardware), platforms, and applications (i.e. software) by virtualization of resources [1]. The different form of cloud computing services include IaaS, PaaS, and SaaS which stands for (Infrastructure-as-a-Service), (Platform-as-a -Service), and (Software-as-a-Service) respectively. Elastic Beanstalk is a PaaS offered from the AWS that allows users to create applications and push them over the cloud, wherein creating an environment where the features and services of AWS are used such as Amazon EC2, Amazon RDS, etc. to maintain and scale the application without the need for continuous monitoring.

AWS Elastic Beanstalk is one of the many services provided by the AWS with its functionality to manage the Infrastructure. Elastic beanstalk provides a quick deployment and management of the applications on the Cloud as it automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring. Elastic beanstalk uses highly reliable and scalable services [2].

2. NEED FOR AWS ELASTIC BEANSTALK

Cloud Computing shifts the location of resources to the cloud to reduce the cost associated with over-provisioning, under-provisioning, and under-utilization [3]. When more than required resources are available it is called over-provisioning. When the resources are not used adequately it is known as under-utilization. When the resources available are not enough is called under-provisioning. With Elastic Beanstalk, users can prevent the under-provisioning, under-utilization and over-overprovisioning of computing resources, as it keeps a close check on how the workload for the application is varying with time. The information thus obtained, helps in automatically scaling the application up and down by provisioning the adequate required resources to the application. Elastic Beanstalk also reduces the average response time for the application as it automatically provides the needed resources without manual monitoring and decision making. A user uses one or more of these three scaling techniques to manage and scale the application:

2.1. Manual Scaling in Cloud Environments

In traditional applications, scalability is achieved by predicting the peak loads, then purchasing, setting up and configuring the infrastructure that could handle this peak load [4]. With Manual Scaling, the resources are provisioned at the deployment time and the application servers are added to infrastructure manually, thus there is high latency.

2.2. Semi-automatic Scaling in Cloud Environments

In Semi-automatic Scaling the resources are provisioned dynamically (i.e. at runtime) and automatically (i.e. without user intervention) [1]. Thus, the mean time until when the resources are provisioned are short. However, manual monitoring of resources are still necessary. Since resources are provisioned by request, the problem of the unavailability of an application at peak loads is not completely eliminated.

2.3. Automatic Scaling in Cloud Environments

Elastic Beanstalk uses the Automatic Scaling which overcomes the drawbacks of both the Manual Scaling as well as Semi-automatic Scaling by allowing the users to closely follow the

workload curve of their application, by provisioning of the resources on demand. The user owns the choice that the number for resources these applications are using increases automatically during the time when the demand of resources are high to handle the peak load and also automatically decreases when the minimal resources are needed, to minimize the cost so that the user only pays for what they used. Automatic Scaling also predicts the peak load that the applications may require in the future and provisions these required resources in advance, proving the elasticity of the cloud [1].

3. ELASTIC BEANSTALK SERVICES

Elastic Beanstalk supports applications developed in Java, PHP, NET, Node.js, Python, and Ruby, and also in different container types for each language. A container defines the infrastructure and software stack to be used for a given environment. When an application is deployed, Elastic Beanstalk provisions one or moreAWS resources, such as Amazon EC2 Instances [5]. The software stack that runs on the instances depends on the container type, where two container types are supported by the Elastic Beanstalk Node.js: a 32-bit Amazon Linux Image and a 64-bit Amazon Linux Image. Where each of them runs the Software stack tailored to the hosted Node.js application. Amazon Elastic Beanstalk can be interacted using the AWS Management Console, the AWS Command Line Interface(AWS CLI), or a high-level CLI designed for Elastic BeanStalk [5].

The following web service and features of AWS are used by the Amazon Elastic Beanstalk for the Automatic Scaling of Applications:

3.1. AWS Management Console

AWS Management Console is a browser-based graphical user interface(GUI) for Amazon Web Services. It allows users to configure an automatic scaling mechanism of AWS Elastic Beanstalk as well as other services of AWS. From the Management console, the user can decide about how many instances does the application require. When the application must be scaled up and down [1].

3.2. Elastic Load Balancing

Amazon Elastic Beanstalk uses the Elastic Load Balancing service which enables the load balancer to automatically distributes the incoming application traffic across all running instances in the auto-scaling group based on metrics like request count and latency tracked by Amazon Cloud Watch. If an instance is terminated, the load balancer will not route requests to this instance anymore. Rather, it will distribute the requests across the remaining instances [6].

3.3. Auto Scaling

Auto Scaling automatically launches and terminates instances based on metrics like CPU and RAM utilization of the application and are tracked by Amazon CloudWatch and thresholds called triggers. Whenever a metric crosses a threshold, a trigger is fired to initiate automatic scaling. For example, a new instance will be launched and registered at the load balancer if the average CPU utilization of all running instances exceeds an upper threshold

3.4. Amazon Cloud Watch

Amazon CloudWatch enables the application to monitor, manage and publish various metrics. It also allows configuring alarms based on the data obtained from the metrics to make operational and business decisions. Elastic Beanstalk automatically monitors and scales the application using the CloudWatch.

4. AMAZON ELASTIC BEANSTALK DESCRIPTION

An Elastic Beanstalk Application is a collection of Elastic Beanstalk components which include the application versions, environments, and the environment configurations. Application Version is a deployable code for the web application and it alson implements the deployable code via Amazon Simple Storage Service (Amazon S3) which contains the code. An application may have many different application versions [7].

An environment is a version that is deployed onto AWS resources. At the time of environment creation, Elastic Beanstalk provisions the resources needed to run the application version specified. Where, the environments include an environment tier, platform, and environment type. The environment tier chosen determines whether Elastic Beanstalk provisions resources to support a web application that handles HTTP(S) requests or web application that handles the background processing task. AWS resources created for an environment include one elastic load balancer, an Auto Scaling group, and one or more Amazon EC2 instances [8].

The software stack that runs on the Amazon EC2 instances is dependent on the container type. Where a container type defines the infrastructure topology and software stack to be used for that environment. For example, an Elastic Beanstalk environment with an Apache Tomcat container uses the Amazon Linux operating system, Apache web server, and Apache Tomcat software. In addition, a software component called the host manager. HM runs on each Amazon EC2 server instance. The host manager reports metrics, errors and events, and server instance status, which are available via the AWS Management Console, APIs, and CLIs [8]. The important aspects of Elastic Beanstalk such as security, management version updates, and database and storage are discussed below:

4.1. Security

The application on the Elastic Beanstalk cloud is available publically at myapp.elasticbeanstalk.com for anyone to access. The user can control what other incoming traffic, such as SSH, is delivered or not to your application servers by changing the EC2 security group settings [9]. The IAM (Identity and Access Management) allows the user to manage users and groups in a centralized manner, such as the user can control which IAM users have access to AWS Elastic Beanstalk, and limit permissions to read-only access to Elastic Beanstalk for operators who should not be able to perform actions against Elastic Beanstalk resources [9].

4.2. Management Version Updates

The user can opt to update the AWS Elastic Beanstalk environment automatically to the latest version of the underlying platform running the application during a specified maintenance window [9]. The managed platform updates use an immutable deployment mechanism to perform these updates, where the applications will be available during the maintenance window and consumers will not be impacted by the update.

4.3. Database and Storage

AWS Elastic Beanstalk stores the application files, deployable codes, and server log files in Amazon S3. If the user is using the

AWS Management Console, the AWS Toolkit for Visual Studio, or AWS Toolkit for Eclipse, an Amazon S3 bucket will be created in the user's account for the user, and the files uploaded by the user will be automatically copied from your local client to Amazon S3 [9]. AWS Elastic Beanstalk does not restrict the user to use any particular data persistence technology. The user can choose to use Amazon Relational Database Service (Amazon RDS) or Amazon DynamoDB, or use Microsoft SQL Server, Oracle, or other relational databases running on Amazon EC2. The user can configure AWS Elastic Beanstalk environments to use different databases by specifying the connection information in the environment configuration. The connection string will be extracted from the application code and thus Elastic Beanstalk can configure different databases to work together [9].

5. CONCLUSION

"There is an observation that in many companies the average utilization of application servers ranges from 5 to 20 percent, meaning that many resources like CPU and RAM are idle at peak times" [10]. Thus, Amazon Elastic Beanstalk helps to provide automatic scaling of the application which efficiently utilizes the expensive computing resources and makes the application able to manage the peakload at all times [9].

Other platforms like Google App Engine also let users have their applications to automatically scale both up and down according to demand but with even more restrictions on how users should develop their applications [11]. Wheras, with AWS Elastic Beanstalk there is more control with the user as the user can manage all the elements of the infrastructure or choose to go with Automatic Scaling.

REFERENCES

[1] D. Bellenger, J. Bertram, A. Budina, A. Koschel, B. Pfänder, C. Serowy, I. Astrova, S. G. Grivas, and M. Schaaf, "Scaling in cloud environments," *ACM Digital Library*, pp. 145–150, 2011. [Online]. Available: http://www.wseas.us/e-library/conferences/2011/Corfu/COMPUTERS/COMPUTERS-23.pdf

[2] Amazon Web Services, "Waht is aws elastic beanstalk," Web page, 2017, online; accessed 19-Mar-2017. [Online]. Available: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html

[3] S. Tai, J. Nimis, C. Baun, and M. Kunze, *Cloud Computing: Web-Based Dynamic IT Services 1st*. Springer Publishing Company, Incorporated ©2011. [Online]. Available: http://www.springer.com/us/book/9783642209161

[4] J. Yang, J. Qiu, and Y. Li, "A profile-based approach to just-in-time scalability for cloud applications." IEEE Computer Society Washington, DC, USA ©2009, 2009, pp. 9–16.

[5] J. Vlie, F. Paganelli, S. van Wel, and D. Dowd, *Elastic Beanstalk*, 1st ed. O'Reilly Media, Inc., 2011, online; accessed 18-Mar-2017.

[6] Amazon Web Services, "Elastic load balancing," Web page, online; accessed 20-Mar-2017. [Online]. Available: https://aws.amazon.com/elasticloadbalancing/

[7] Amazon Web Services, "Elastic beanstalk components," Web page, online; accessed 2-Apr-2017. [Online]. Available: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.components.html

[8] Amazon Web Services, "Elastic beanstalk architecture," Web page, online; accessed 30-Mar-2017. [Online]. Available: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.concepts.architecture.html

[9] Amazon Web Services, "Frequently asked questions," Web page, online; accessed 2-Apr-2017. [Online]. Available: https://aws.amazon.com/elasticbeanstalk/faqs/

[10] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Communications of the acm," *ACM Digital Library*, vol. 53, pp. 50–58. [Online]. Available: http://cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext

[11] Google Inc., "Google cloud platformGoogle App Engine," Web page, online; accessed 18-Mar-2017. [Online]. Available: https://cloud.google.com/appengine/