The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 8 Issue 1, 2020
pp: 05 - 08

News Magazine

[1]D. Apoorva, [2]B. Anjali, [3]G. Swapna, [4]K. Shireesha
[1,2,3,4] UG Student, Department of CSE,
Dr K V Subba Reddy College Of Engineering For Women, Kurnool, Andhra Pradesh, India

Abstract
Python can be used to build server-side web applications. While a web framework is not required to build webapps. Python is not used in a web browser. The language executed in browsers such as chrome, Firefox and internet explorer is JavaScript. Most Python developers write their web applications using a combination of Python and JavaScript.
How browsers work provides an overview with solid detail on how browsers take the HTML,CSS, JavaScript, images and other files as input and render web pages as output. Web application development development is different and better provides some context for how web development has evolved from writing static HTML files into the complex JavaScript client-side applications produced in now a days we development involves HTTP communication between the server, hosting a website or web application, and the client, a web browser. Web development is the umbrella term for conceptualizing, creating, deploying and operating web applications and application programming interfaces for the web

1. Introduction

Amazon Web Services (AWS) comprises over one hundred services, each of which exposes an area of functionality. While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them. With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.
Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python, and Ruby. When you deploy your application, Elastic Beanstalk builds the selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application. You can interact with Elastic Beanstalk by using the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or eb, a high-level CLI designed specifically for Elastic Beanstalk.
To learn more about how to deploy a sample web application using Elastic Beanstalk, see Getting Started with AWS: Deploying a Web App. You can also perform most deployment tasks, such as changing the size of your fleet of Amazon EC2 instances or monitoring your application, directly from the Elastic Beanstalk web interface (console)

2. Literature Review

Because applications deployed using Elastic Beanstalk run on Amazon cloud resources, you should keep several things in mind when designing your application: scalability, security, persistent storage, fault tolerance, content delivery, software updates and patching, and connectivity. For a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security and economics, go to AWS Cloud Computing Whitepapers.

When you're operating in a physical hardware environment, as opposed to a cloud environment, you can approach scalability two ways—you can scale up (vertical scaling) or scale out (horizontal scaling). The scale-up approach requires an investment in powerful hardware as the demands on the business increase, whereas the scale-out approach requires following a distributed model of investment, so hardware and application acquisitions are more targeted, data sets are federated, and design is service- oriented. The scale-up approach could become very expensive, and there's still the risk that demand could outgrow capacity. Although the scale-out approach is usually more effective, it requires predicting the demand at regular intervals and deploying infrastructure in chunks to meet demand. This approach often leads to unused capacity and requires careful monitoring.

By moving to the cloud you can bring the use of your infrastructure into close alignment with demand by leveraging the elasticity of the cloud. Elasticity is the streamlining of resource acquisition and release, so that your infrastructure can rapidly scale in and scale out as demand fluctuates. To implement elasticity, configure your Auto Scaling settings to scale up or down based on metrics from the resources in your environment (utilization of the servers or network I/O, for instance). You can use Auto Scaling to automatically add compute capacity when usage rises and remove it when usage drops

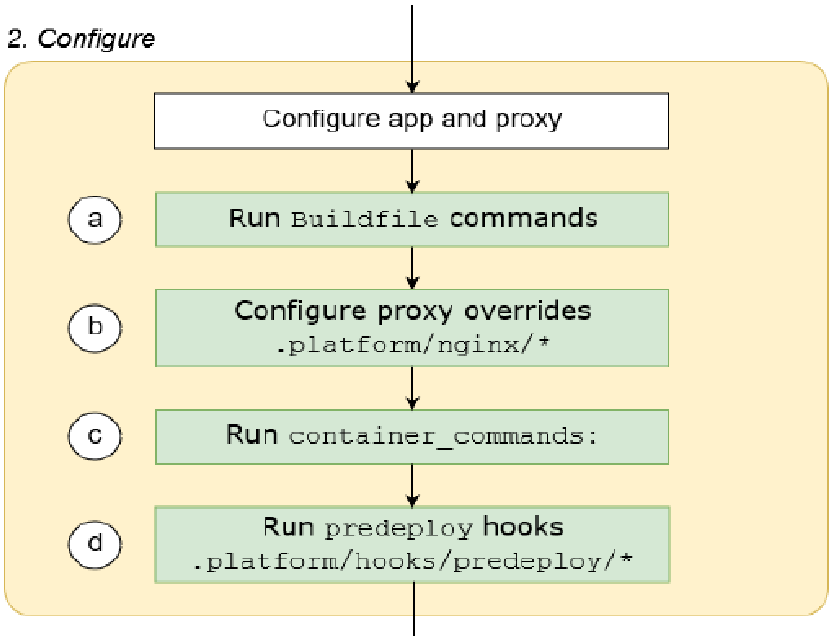

Fig.1 Existing System

3. Proposed System

An entire new platform from scratch, customizing the operating system, additional software, and scripts that Elastic Beanstalk runs on platform instances. This flexibility enables you to build a platform for an application that uses a language or other infrastructure software, for which Elastic Beanstalk doesn't provide a managed platform. Compare that to custom images, where you modify an Amazon Machine Image (AMI) for use with an existing Elastic Beanstalk platform, and Elastic Beanstalk still provides the platform scripts and controls the platform's software stack. In addition, with custom platforms you use an automated, scripted way to create and maintain your customization, whereas with custom images you make the changes manually over a running instance. To create a custom platform, you build an AMI from one of the supported operating systems— Ubuntu,
RHEL, or Amazon Linux (see the flavorentry in Platform.yaml file format (p. 46) for the exact version numbers)—and add further customizations. You create your own Elastic Beanstalk platform using Packer, which is an open-source tool for creating machine images for many platforms, including AMIs for use with Amazon Elastic Compute Cloud (Amazon EC2). An Elastic Beanstalk platform comprises an AMI configured to run a set of software that supports an application, and metadata that can include custom configuration options and default configuration option settings. Elastic Beanstalk manages Packer as a separate built-in platform, and you don't need to worry about Packer configuration and versions.

You create a platform by providing Elastic Beanstalk with a Packer template, and the scripts and files that the template invokes to build an AMI. These components are packaged with a platform definition file (p. 39), which specifies the template and metadata, into a ZIP archive, known as a platform definition archive (p. 43).

When you create a custom platform, you launch a single instance environment without an Elastic IP that runs Packer. Packer then launches another instance to build an image. You can reuse this environment for multiple platforms and multiple versions of each platform.

Detects text in the input image and converts it into machine-readable text.Pass the input image as base64-encoded image bytes or as a reference to an image in an Amazon S3 bucket. If you use the AWS CLI to call Amazon Rekognition operations, you must pass it as a reference to an image in an Amazon S3 bucket. For the AWS CLI, passing image bytes is not supported. The image must be either a .png or .jpeg formatted file.

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 8 Issue 1, 2020
pp: 05 - 08

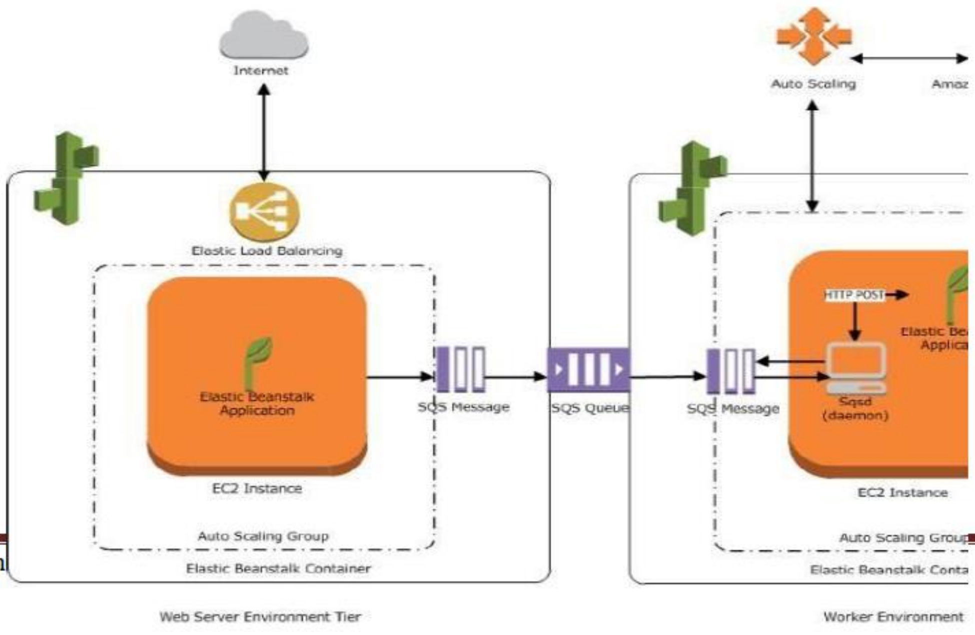

Fig.2 Proposed System

4. Conclusion

AWS Elastic Beanstalk is a good and quick way of deploying and monitoring web application bulit using python or using other technologies supported. We can upload our code and Elastic Beanstalk automatically handles the deployment. Elastic Beanstalkautomatically scales our application up and down based on our needs in python web Easily integrate powerful image in mobile or desktop application - Eliminates the time-consuming complexity associated with creating capacity for image recognition in your apps with this simple API.

References

1. https://aws.amazon.com/elasticbeanstalk/
2. https://aws.amazon.com/ec2/
3. https://aws.amazon.com/getting-started/projects/deploy-python-appliction/servicecosts/
4. https://aws.amazon.com/getting-started/hands-on/deploy-python-application/
5. https://docs.aws.amazon.com/rekognition/index.html.
6. https://www.linkeit.com/blog/what-is-aws-rekognition.
7. https://www.novetta.com/wpcontent/uploads/2017/08/NovettaBiometrics_AmazonRekognitionBiometricPerformanceAssess
8. https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html
9. https://aws.amazon.com/rekognition/
10. https://en.m.wikipedia.org/wiki/Amazon_Rekognition
11. https://aws.amazon.com/rekognition/image-features/
12. https://docs.aws.amazon.com/rekognition/latest/dg/text-detecting-text-procedure.html