AUTOMATIC DEPLOYMENT ON CLOUD USING SOFTWARE PRODUCT LINE PRINCIPLES

Ankur Jain[1], Shubham Gupta[2], Shefali Garg[3]

[1]*Ankur Jain Student, IT, BVCOE, Delhi, India*
[2]*Shubham Gupta Student, IT, BVCOE, Delhi, India*
[3]*Shefali Garg Student, IT, BVCOE, Delhi, India*

Abstract

In the modern age, web development is one of the fastest growing paradigms. Any developer who has worked on the web will agree that building an application is one thing and deploy the application on the server is another. There are many challenges in deploying the application on the web which includes selecting the cloud service provider, the type of hardware and software configuration needed, selection of a web and application servers and much more. First of all doing all theses thing is an ad hoc task and even after completion of all these tasks there is no guarantee that the application will run error free. In this paper, we propose a software pipeline that made the task of deployment a piece of cake. The pipeline consists of two configurations. In one configuration, the software decides the pipeline for the user and in the other configuration the user can set the pipeline by itself. With this approach, the deployment time is reduced many folds. Also, since our approach works on the application layer, it can be extended to a number of cloud environments.

Key Words: *Fabric, automatic deployment, Django, speed, cloud, software pipeline, ease of access, user friendly.*

1. INTRODUCTION

Cloud computing has emerged as a major trend in distributed computing environment. The cloud service usually works on three basic models i.e SaaS(Software as a service), PaaS(platform as a service), IaaS (infrastructure as a service)[1]. In IaaS, the service provider provides you with a basic physical server connected to the web with your choice of OS installed on it. On the other hand, the PaaS provides a method to interact with the high-level services like a database, web servers and caching without having to deal with low-level services like a type of server or ram needed. At last, the SaaS [2][3] is built on the platform as a service that provides, even more abstraction than PaaS. It offers a software on the web and made it available to the client. The amount of user work is less in SaaS.

Due to the presence of such variable platforms available for development, the developer faces three challenges at the time of deployment [4]. The first challenge is to select the functional and nonfunctional parameters. The functional parameters include the parameters that are essential for the application to run properly like the application servers, necessary libraries and much more. The non-functional parameters include the hardware components like ram and CPU power needed to support the functional parameters. The second challenge is the configuration of all these parameters. And the last challenge is to deploy it in such a way that all the modules work in a synchronous error free manner.

To address all these challenges we propose a software pipeline which is described in the research paper. The paper consists of the following sections. The section II explains the software pipeline in detail and how it is used to solve the above problems. In section II, we explain the technology used for building the software pipeline. We describe the evaluation and result done by us in the section IV of the paper.

2. PRODUCT LINE FOR DEPLOYMENT AND CONFIGURATION

The product line is aimed at deploying the software efficiently, reliably and in the least time with a reduction in cost and market efforts [7]. In order to build the product line, the user needs assets. The few assets are mandatory and few are optional. The mandatory assets are those assets which are necessary for building the product line for deployment. These assets are the backbone of the line and hence cannot be changed. But there are some optional assets which can be added or removed according to user wish or can even be replaced.

In the beginning of the product pipeline, the mandatory assets are triggered and almost remain constant. The mandatory assets are normally the code written to support the whole pipeline. After that, the developer selects the optional assets or features to build the desired pipeline according to its need. During selection of the features, their corresponding necessary configuration has also to be made. Now all the valid configuration and features are fed to the mandatory asset to complete the pipeline and deploy the code.

Our product pipeline distinguishes between two roles, domain experts and developers. The former are the cloud experts which have a very high understanding of the deployment and the cloud. These are those users which would want to configure the whole pipeline according to their own need. These users might also want to write their own configurations and select the modules if they feel necessary. On the other hand, the developers are those users that will be using the proposed approach. These users are those users that did not have any significant knowledge of deployment. These may not be interested in the configuration and would use automatically generated pipeline for deployment.

2.1 Cloud/Product Pipeline as Feature

The cloud pipeline can be easily depicted as a feature model. In this section, we will explain how the user selects or move through the product line and deploy the code on the web with the help of feature model. The pipeline consists of following sequential steps.

- Selection of the type of service: The user initially selects from two types of services available i.e. the cloud expert and the novice. The cloud expert service gives the developer the full access and control to change its configuration and even made it own. On the other hand, the novice service is for those developers who want to make the deployment quick and simple without getting into the details of configuration and setup. For such developers, the system takes decision for them.
- Selection of server: This stage is similar to both the services provided. This stage provides two options to the user i.e. either use our servers or provide the access to one of theirs. Normally the experts go with the second option and the novice with the first.
- Select the language and framework: Next stage makes the user select the language and framework to be deployed on the server. This stage is also identical for both the services provided. The selection of language and framework helps us to determine the best configuration for a given deployment.
- Selection of web servers, load balancers and application servers: This stage is limited to only cloud expert services. This allows the user to select his personal web and application servers from a list of available options. As far as other services are concerned, it is automatically decided by the software.
- Configuration: This is an important part of the pipeline. The configuration is automatic for the novice developers and manual as well as automatic for the expert developers. For the manual configuration, the testing is done by the software before finalizing the configuration.

3. TECHNOLOGIES USED

What technical things should the developer includes before making the site public? Imagine a web page without css or javascript, it will be just a page but not the responsive one. Similarly deployment needs the proper technology stack to work with. The outline when you are choosing a stack is thinking ahead. It basically concentrates on scalability, cost, efficiency, time, security and other major factors. So considering all these factors our paper focuses on

these technologies.

- *Django Framework*

  Django Framework is a lightweight and an open source license. More than thousands of packages and libraries are available to extend this massive framework. It has a understandable Model View Controller system. As compared to Flask Django provides more features like no forms processing. We need to install libraries for flask. Django follows the proper hierarchy for settings, authentications, view pages etc. Deployment through Django is quite simple as it provides various ways to deploy the code. Django uses wsgi platform standard for servers and web applications. Django uses the DJANGO_SETTINGS_MODULE[5] which is called when wsgi.py server runs. It also includes security features like secret key which is a large random value and must be kept secret.

- *Fabric*

  As we know that Django supports multiple libraries and packages, so fabric was the natural choice for deployment with Python. It is a command line utility with to run commands remotely over SSH for streamlining deployment. The fabric divides the deployment task into four pieces- 1) Connection to the server, 2) Virtual Environment Creation, 3) Code Transfer, 4) Installing required libraries. So, we have only one command to perform the deployment which will do the things right way instead of the quick way. Dealing with fabric is easy as we have to just use easy_install to initiate it. The following points are discussed previously. So we prefer Fabric as it gives the option of embedding all our commanding logic into python.

- *AWS services*

  As per Gartner research firm the cloud computing market spend will go from $23.8 billion to $39.4 billion from 2013 to 2015. It is expected that cloud computing will replace high infrastructure expenses. They can produce faster results by accessing thousands of virtual servers simultaneously through cloud. Nowadays web services are offered by AWS which is robust and secured. So we will use AWS as our host server for this paper as it compatible with any language and framework. To go ahead we can use ec2 instance variable or setting up a local virtual machine with the help of Vagrant. AWS supports tight security in which AWS access and secret keys are stored in AWS secret page.[6] AWS supports multiple features like management of domain name services, Amazon relational database service, Amazon API gateway, simple storage service and many more. AWS goes in hand in hand with the developer as it provides the features according to him. Developer can choose the instances type like T2, C4, I2, D2, T1, and then pay accordingly. Moreover it will show the performance and efficiency of your server with proper statistics and total data consumed. Connecting with the AWS instances is done with the IP address which is generated every time when it gets rebooted. With the benefits like Zero CapEx, procurement, pay per use, security, scalability and clean design of the high-level Django framework and object-oriented language makes this stack more rapid and simplifies to deploy the software.

4. EVALUATION

The automated system adopted in this project is implemented in Python, now we will evaluate our approach with various experiments we conducted with closs. The basis of our evaluation is totally dependent upon with the following criterias-

a) Time: Is closs cut-off the developers time to deploy the code?

2) Method: Are the developers could efficiently use methods to deploy the code like explained in 2(A) according to their requirements?

3) Experience: At what extent the closs is user-friendly?

4) Execution: Is app running on the server efficiently as expected?

This experiment was tested with the group of people who are from different fields like engineers, devops,

Participants	1	2	3	4	5	6	7	8
Field	E	N	E	N	N	E	E	N
Time(min)	15	72	31	41	52	19	21	65
Experience	5	1	4	2	2	4	3	1
Is running?	Yes	**No**	Yes	Yes	Yes	Yes	Yes	**No**

Fig. 1. Table 1 (N - Novice, E - Expert)

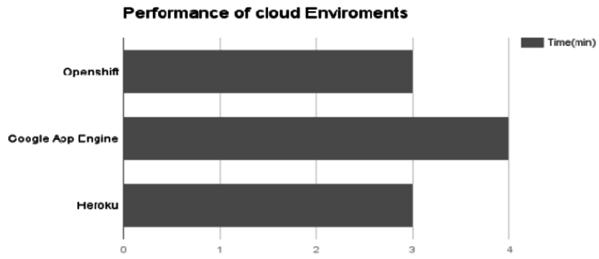

Fig. 3

Participants	1	2	3	4	5	6	7	8
Field	E	N	E	N	N	E	E	N
Time(min)	7	4	6	3	4	4	5	2
Experience	5	4	5	5	5	5	5	4
Is running?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Fig. 2. Table 2 (N - Novice, E - Expert)