2015 1st International Conference on Next Generation Computing Technologies (NGCT-2015)
Dehradun, India, 4-5 September 2015

Automated Provisioning of Application in IAAS Cloud using Ansible Configuration Management

Nishant Kumar Singh*, Sanjeev Thakur† Himanshu Chaurasiya‡ and Himanshu Nagdev§

* Computer Science and Engineering ASET, Amity University Noida(UP), INDIA nishant704@gmail.com
†Computer Science and Engineering ASET, Amity University Noida(UP), INDIA Sthakur3@amity.edu
‡Electronics and Communication Engineering ASET, Amity University Noida(UP), INDIA hchaurasiya@amity.edu
§Electronics and Communication Engineering ASET, Amity University Noida(UP), INDIA himanshunagdev.hn@gmail.com

***Abstract*—Cloud has become a dominant technology in todays information technology environment and the need arises to abide with the growing demand of clients. The pressure on information technology organizations is constantly increasing to deploy the customer's application over private cloud. This change has come into light since a large number of clients have directly started to contact the cloud venders for this support. Moreover, the DevOps teams are in much bigger focus now since they are responsible for the automation and provisioning of the whole environment along with the client application. This paper focuses upon the automation of customer application right from environment provisioning to application deployment. In this paper the whole architecture of automated application deployment assembled using amazon as IAAS provider and Ansible as the orchestration engine.**

Keywords:IAAS cloud, Provisioning, Ansible, AWS, DevOps

I. Introduction

In the current scenario, the importance of IAAS clouds has been realized beyond measures. The I.T industries are adopting accordingly with the growing need since the business requirement as well as market competition are all pointing in the direction of cloud. The reason why all this up scaling is being seen in information technology is due to the fact that customers are demanding swift service deployment over the cloud. With the advent of Amazon EC2 instances the environment creation takes less than a minute unlike earlier where the customer had to sit for weeks just to get started with the hardware [1]. The IAAS provider gives full flexibility to the customer to provision computational, storage as well as networking resources on cloud. The customer always welcomes this approach and demands this type of quick response from the IAAS provider. This direct interaction with the cloud provider makes the customer interested in one on one provisioning but due to the lack of proper tools, this gets hindered from customer end. Moreover, the clients who have a private cloud always have a complete control of the cloud space as compared with grid or cluster computing; so that their deployed application are customizable all the time. But to all this upper-hand, there comes a problem of complexity followed by additional endeavor towards setting up and deployment of application by the customer itself [2]. At this point of time, the need of automated provisioning comes into play along with the customer application deployment over the cloud. This rising need is satisfied by the DevOps teams which take care of all these operations over cloud. Automation can reduce infrastructure provisioning time from days to just few hours giving a critical advantage to the customer end. The need of a fully efficient DevOps team is also a prime focus in a highly dynamic environment of cloud computing since unlike the usual grid or cluster environment where an administrator can simply setup services; cloud requires more focus [3]. The applications running over cloud usually run for few hours and hence pay per use system is always kept in mind by DevOps teams. All these requirements need to be taken care by the DevOps team for increased efficiency of the customer application [4]. The need of automation doesn't just stop here, since a need arises to streamline and standardize the whole procedure right from customer request to service roll out. The old way of application deployment used to deal with batch scripts which basically were unreliable and lacked idem-potency. In this paper, the Ansible is used for configuration management tool over IAAS cloud. This paper reveals how application provisioning is done in the cloud using an architectural overview and gets to know more about the industrial ways of practice automating.

II. ARCHITECTURAL REQUIREMENTS

The maximum capacity of the IAAS cloud in automated provisioning and deployment accommodates satisfaction of the customer needs. This technique is the basic controlling rule behind fruitful execution of automated provisioning and deployment of the infrastructure as well as platform as requested by the customer. To be able to configure a fully organized automation system, the following requirements should be considered [5]

A. *Automated Application Deployments*

Most of the application that needs to be deployed by the DevOps team for the customer usually runs for a small duration of time on the cloud in order to complete a much bigger task. This is one of the most prominent reasons why IAAS cloud fits the bill for such application as they work to pay per use and on demand provisioning. Sadly, these applications frequently oblige complex environment in which they need to run. To build and set-up this sought of environment a DevOps team needs to be able to do this in a repeated manner whenever an application is being deployed. Earlier this was achieved using batch scripts which simply did the job by checking upon tmhe target machine. This was not considered to be a good practice. An automation engine is what a DevOps team is required at this stage. An automation tool needs to gather important facts about the target system and deploy the application whenever the DevOps team initiates the process.

B. Complicated Conditions and Dependencies

It is a common practice that the application is generally not going to be deployed at one single machine. Most of the times the database is set up on a different machine, the application server on another machine and the web server on some other host, all of these host machines depend on each other for communications as well as correct functioning of the whole application. With such a straight forward goal, the hosts must be configured in a well-defined order such that all of this gets executed in specific order along with exclusive dependencies.

C. Active Provisioning

The applications which are deployed over the cloud are dynamic in nature, i.e. the resource requirements of such applications amend very quickly. Lets say that java heap memory size might need to change if application data increases or maybe the load on application server increases at a certain hour of time. DevOps need to support this change in deployment conditions at run time. Ansible is one of the prominent answers to such kind of dynamic behavior of applications.

D. Monitoring

To achieve an instant attention of information technology framework issue, such that downtime doesn't antagonistically influence the customer is a vital part of DevOps deployment provisioning. Tasks which run for a long time may experience issues that oblige client's attention at extreme levels. A monitoring service like Nagios is employed by the DevOps reams to take care of any such problems such that if an incident occurs, an alert can be triggered instantly to the team.

E. Numerous Cloud Providers

A very few of the times, a service requested by the customer side might not be available with a cloud provider and need might arise to shift the deployed application to other cloud providers. A configuration management tool like Ansible is highly capable of doing the same with quite an ease. Apart from the above mentioned requirements, the DevOps teams need to follow a model which help them better understand automated provisioning and deployment. This understanding will serve as a guide to help the organizations to evaluate current state with respect to individual, technology, process and business administrator [6] [7].

III. DESIGN AND IMPLEMENTATION

The information technology organizations around the globe have different and vivid necessities which require a lot concerns while designing an end to end automated provisioning and deployment solution. Despite that, a certain number of guidelines need to be followed by the DevOps teams before automations can begin. The principle guidelines for designing and implementation of a fully automated provisioning system are [8]

A. Characterize and Document the Existing Processes

Prior to the DevOps can make headway with automating the provisioning, deployment process, the current deployment document must exist which describe the whole process thoroughly, portray every methodology regarding key measurements, and reduce manual methods however much as could reasonably be expected. This is a primarily and important task since an astounding number of information technology methods are inadequately characterized and reported. Most of the time, there is no formal definition during the request phase which results in including conflicting results, wide difference in labor expenses and procedure span, inconsistent or missing measurements, and poor administration.

B. Recognize the Owners of Process

It is a very difficult to own a process on the off chances that it does not have an exact and exhaustive definition. This is why it is quite an important step to identify the process owners whenever a DevOps team starts to automate provisioning and deployment. Moreover the owners must be included in all periods of automations, so that they can provide profitable knowledge into the needs of business clients and also make sure that the quality assurance phase is checked upon constantly.

C. Optimized Before it Gets Automated

A DevOps team must make sure that the process is already optimized before it can be automated since a non-optimized process is infective and will result in low productivity.

D. Define a Benchmark

A benchmark basically helps the DevOps team to help it through decision making before automation can begin. A benchmark sets standard time duration from the current end-to-end time of customer request to final service activation. The advantage of setting a threshold determines the gain of automation versus previous batch deployment. There are many different matrices involved with system crashes, provisioning etc. The standard must be calculated even after automated provisioning is successfully completed so that the comparison between expected and actual advantages can be made.

E. Create Strategy to Operate Automation

A strategy based governance workflow helps an organization to combat the hindrance created by a manual approval scheme. As discussed earlier, standardization helps to start the request process easily. DevOps teams usually take care of this hindrance by a customer portal to handle the request initiation phase by the potential client. As soon as the customer fills up and submits the requirement form, the later process is governed with the help of pre-defined strategies.

F. Revise the Economic Authority

To make sure that desired workflow velocity is achieved during the request phase, it is highly recommended to reassess the economical control. One methodology is to give business clients a preapproved "line of credit" which they can spend for required administrations without unequivocal support for individual appeals. Different associations depend on month to month reporting with show back frameworks to instruct business clients about the money related effect of their administration utilization and make responsibility.

IV. ARCHITECTURE AND DEPLOYMENT

There is an application which basically responsible for the management and enforcement of entitlements for SAAS as well as on premises applications. This application is delivered as a product to the end customer. The product is a security service for feature based authorization and also serves as a licensing model. DevOps are interested in the deployment of this application over the cloud. The previous deployment techniques make the use of batch script which lack a lot of features including idem potency and is considered an ill methodology. The goal is to automate the whole process of deploying using a configuration management tool. For this purpose, here in this paper Ansible is used to perform automated provisioning. The goals which Ansible persuade basically revolve around simplicity and maximizing the ease of usability. Before automating, set the outline of the architecture for the deployment first. The components of this architecture include:

A. Scamper Point

It is a customer side part that interfaces with cloud-empowered parts, for approval of authorizing choices and gathering of use information. Scamper point gives an arrangement of API calls that one have to embed in application code to inquiry rights for a client to every individual highlight of the application. The assignments that scamper points perform are:

1) User authorization: Allows or limits a client from getting to the highlights of your application, as characterized by the administration assertion and the representing permit model. Scamper point corresponds with cloud connect for permitting choices.
2) Use information gathering: Collects information identified with utilization of your application by every client, at the highlight level. This information can be further utilized by charging and business insight frameworks. Scamper point stores this information mainly and occasionally exchanges it to the usage recorder for metering and information collection purposes.

B. Usage Recorder

It is a segment that approves permit demands and methodologies utilization records got from scamper point. License authorization is attuned to when scamper point, coordinated in your application, queries usage recorder for a permit serving choice. When your client endeavors utilize a highlight of your application. Both the segment interfaces with one another utilizing predefined solicitation reaction messages. This segment is additionally in charge of enacting the client administration understanding, authorizing permissive terms, and a worldwide choice making.

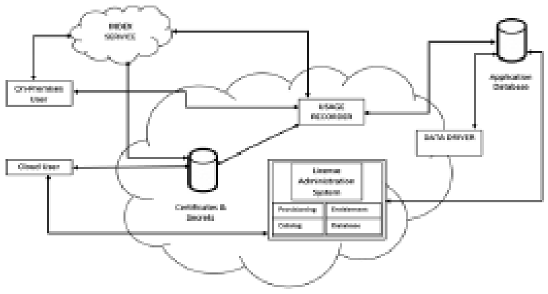

Fig. 1. Application Component and Architecture

C. License Administration System

It is an online administration stage that aides in administration list definition permit provisioning and administration, reporting capacities, and client administration. It gives both an online interface and a web administrations interface for making and looking after qualification.

D. Index Service

This part aides interface a scamper point case to a usage recorder hub. It is responsible for:

1) Maintains and deals with a rundown of scamper point cases and usage recorder hubs.
2) Authenticates usage recorder hubs and scamper point.
3) It sends the data of the enrolled usage recorder hubs to scamper point on solicitation. After a positive reaction is gotten from Index Services, Scamper point sends all further solicitations to usage recorder for permit utilization and use information transfer.

E. Data Driver

The data driver stores, courses of action, and total use records of the application. This information is utilized by license administration system for preparing utilization and charging reports. Apart from the above mentioned components, a usage recorder web service shows the licensing as well as monetization abilities of the application. All of these components are shown in figure 1. To begin with the actual automated provisioning, first prerequisite to understand the flow of data and component information using figure 2. To understand the working of this application which needs to be automated, the independent software vendor (ISV) will require. The ISV is responsible for creating a product definition along with the feature and license type in license administration system. This can be done by the ISV using the web UI as well as the web services. This paper focuses on the automated provisioning of license administration system since it simplifies the understanding process in this type of complex deployment.

After the accomplishment of entitlement in license administration system, it automatically gets registered in usage

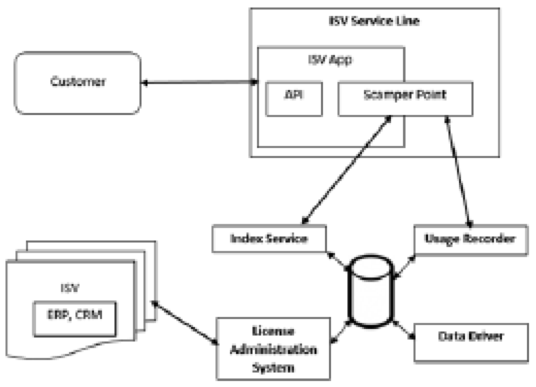

Fig. 2. Component Overview

recorder database. The application which is integrated with the cloud service firstly makes a call to index service to authenticate the ISV. If the ISV is registered, the authentication result in a pass, then the response from index service is the URL of usage recorder. After this process the customer makes a request to ISV application, the API call is made to scamper Point for customer entitlement. When usage records kicks off, it registers with index service. The group id of usage is sent to index service. The index service replies with a list of ISVs with their usage record database configuration back to usage record, so that the license request can be supported. The usage data is stored inside scamper point and is pushed to usage recorder when either the scamper point cache is full or old. Usage recorder takes the raw usage data and stores in its database. The data driver processes raw data on a regular scheduled basis to create some pre-aggregated report. Then the ISV user requests a report from license administration system, which in return requests the usage recorder database for raw data to generate the report. The report is generated back to the ISV user.

V. AUTOMATED PROVISIONING USING ANSIBLE

Since cloud infrastructure is a very large environment consisting of big data, different architectures and more, the information technology industry has realized the need of reliable code for deployment of customer's need. The code's reliability is considered to be an important issue since the end target environment may dynamically change and if the code dissolves at such a high stake environment, the customers expectations won't meet and they shall be disappointed. A DevOps team, intend to deliver the application via high end code. A tool is needed for configuration management along with continuous delivery. For such a requirement, Ansible is used as an information technology automation engine that automates the cloud provisioning. The main reason using Ansible as our configuration management tool is the fact that it is designed for multi-tier deployment. Ansible represents the information technology infrastructure by narrating how the servers interrelate rather than just one single server at a time. A key concept of Ansible is the use of inventory files which is considered to be a single point of truth. Moreover, cloud infrastructure consists of virtual machines which come and go very frequently, so maintaining a single inventory file in such a dynamically provisioned environment is cumbersome. But Ansible has all the features to handle such situations right from dynamic inventory files to full automation modules. The amazon web services (AWS) is accountable for the cloud infrastructure. Ansible has a huge number of AWS modules. All the modules require the famous boto library. Since Ansible is written in python, it makes it extremely portable and flexible. In this work, limit will be set for deployment to provisioning of the license administration system component only; before this the overview of the deployed services and dependencies for license administration system have initialized [9] [10]. The typical production deployment should contain redundant services, load balancing application, clustered back-end databases and deployment performance for centralized repository. The additional support repository includes a central repository of script, tools and application along with monitoring at system and service level. In this new architecture Ansible playbooks are responsible for replace the script for idem potency and much more. To start with the deployment process, Ansible plays important roles. Roles can be understood as include directives for automation purpose. In this scenario, a number of roles are written which deploy our application in cloud environment. The AWS used for most of our deployment and hence the deployment of amazon relational database services (RDS) begin. The RDS followed by the provisioning of EC2 instance. After the EC2 instance is sprung the basic deployment prerequisites introduce. To start the deployment of databases of different components in one single instance i.e. the databases of index service; license administration system and usage recorder are on one single EC2 instance. In the same way the application wise roles played for all components [11] [12]. Now start with the deployment of license administration system on the amazon EC2 instance. The two major roles basically do the deployment of license administration system. The first role is the database role which provisions the database over the instance. As a prerequisite the sql database is install using the sql role and then starts up with our application role. The database role, copies four specifically written sql scripts on the target instance for customer database provisioning. After this script gets copied on the target machine, send the specific customer reference id script to the target. Right after the completion of this task check if a database already exists in the instance and if it does make sure it gets deleted by using the absent tag. Then move on to create the user for our new database and create the application specific database with the requested database name. Once this is accomplished, the task to run sql scripts against database get executed and also the conditional cloud deployment request is also taken care off in the playbook. Later update the customer reference id task which configures the database for that particular customer. After this, the second role which is application specific, call four roles in sequence which deals with this deployment namely:

- Base Role The base role is application specific for the deployment of license administration system. It first sets the x 86 architecture types on the target system. The requirement of setting a 32 bit architectural point is to get the jdk of 32 bit since the license administration system needs this for licensing cloud provisions. Some other dependencies which need to be installed along with jdk are libc6-i386, python-dev, libmysqlclient-dev, python-pip, and python-mysqldb. Once all these dependencies are installed on target

Fig. 3. License administration system accessed over a web browser

machine the required fonts will help in the report generation of the license administration system.

- Tomcat role This role basically installs tomcat 7 on the target machine and restarts the same. The role can be further customized.
- Application Role This is the main role which is responsible for the .war deployment of license administration system. The removals of the previous .war file with same package name and then ensure that the deployment directory is present. After this, copy the .war at specific destination usually at /var/lib/tomcat7. This process set up all the logging options by creating log4j. Once this is accomplished, setup the cloud revocation tool which is yet another licensing option. After updating all XML and .properties file restart the tomcat engine so that the changes can take place. At this point of time the license administration system is up and can be accessed via browser.
- SO or Security certificates role This is again a customizable role which basically involves the transfer of a .dll file or more specifically shared objects. Moreover, security certificates can also be transferred to the target system such that the license administration system can be accessed over https also. After this there is a setup of tomcat 7 to use specific setting by java_home. Now one can access the deployed application using the public IP address of instance and the corresponding port number. At the first call the page throws a slow response time as the application is loading and connections are being established with the database. Figure 3 shows the license administration system accessed over a web browser.

VI. CONCLUSION

With the advent of cloud computing, new dimensions of automation deployment are coming into the picture. The features offered by cloud providers are becoming more vivid as technology progresses. More number of customers demand cloud deployment of requested application for the purpose of increased flexibility. This paper presented the new way of deploying applications on cloud using automated provisioning. there is a use of Ansible, a 4rth generation configuration management tool for automated provisioning over the AWS infrastructure. It is shown how flexibility of Ansible makes it easy to automate the whole procedure of deployment of applications. It is also investigated the working of DevOps teams and showed the code followed by DevOps teams.

REFERENCES

[1] Elastic Compute Cloud (EC2), Amazone Web Services, Amazon.com, http://aws.amazon.com/ec2.

[2] FutureGrid, https://portal.futuresystems.org. W. Gentzsch,Sun Grid Engine: towards creating a compute power grid, 1st IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGrid 01), pp. 35 36, May 2001.

[3] G. Juve, E. Deelman, K. Vahi, and G. Mehta, Scientific Workflow Applications on Amazon EC2, 5th IEEE Int. Conf. on e-Science Workshop (e-Science 2009), pp. 59-66, Dec 2009.

[4] C. David, Introduction to DevOps on AWS, AWS, pp. 2, Dec. 2014 https://d0.awsstatic.com/whitepapers/AWS_DevOps.pdf

[5] Sentinel Cloud Services, Run-time Guide, Safenet, 2014

[6] Sentinel Cloud Service, Sentinel Cloud V3.5 Quick Start Guide, Sentinel, Jun 2014

[7] Automated Provisioning and Deployment, Vmware White Papers,

[8] M. A. Murphy, B. Kagey, M. Fenn, and S. Goasguen, Dynamic Provisioning of Virtual Organization Clusters, 9th IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGrid 09), pp. 364-371, May 2009. doi: 10.1109/CCGRID.2009.37

[9] Ansible, http://docs.ansible.com.

[10] VLDB Endowment, vol.2, Isss.1, pp. 253-264, Aug. 2009.

[11] P. Shivam, A. Demberel, P. Gunda, D. Irwin, L. Grit, A. Yumerefendi,S. Babu, and J. Chase, Automated and On-Demand Provisioning of Virtual Machines for Database Applications, In ACM Proc. Int. Conf. of Management of Data (SIGMOD 2007) pp. 1079-1081, 2007