

# **Comparison of Software Orchestration Performance Tools and Serverless Web Application**

Project report submitted in fulfilment of the requirement for the degree of  
Bachelor of Technology

In

**Computer Science and Engineering**

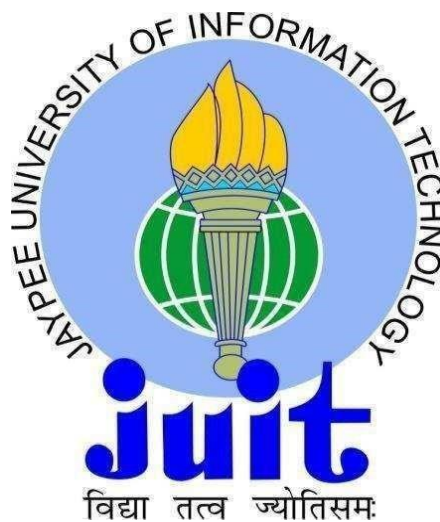
By

Kanav Singla (151397)

Under the supervision of

(Mr. Pristley Sathyaraj)

To



Department of Computer Science & Engineering and Information Technology

**Jaypee University of Information Technology Waknaghat, Solan-173234,  
Himachal Pradesh**

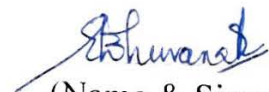
**Project Certificate**  
**Cognizant Technology Solutions(CTS)**

This is to certify that the report entitled “Comparison of software orchestration performance tools” has been <sup>undergoing</sup> completed by **Mr. Kanav Singla** bearing enrollment no **151397** who is bonafide student of Computer Science Department at Jaypee University of Information Technology, Wakanaghat.

He is pursuing training at “Cognizant Technology Solutions.” He is now working on project “**Comparison of software orchestration performance tools**” under my guidance & supervision.

This is to certify that the above statements are correct to the best of my knowledge & belief.

Date: 9<sup>th</sup> May 2017



(Name & Signature)

Cognizant Academy  
Cognizant technology Solutions

## **DECLARATION**

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

**Kanav Singla, 151397**

This is to certify that the above statement made by the candidates is true to the best of my knowledge.

## **ACKNOWLEDGEMENT**

First, I would like to thank **Jaypee University of Information Technology** who provided me an opportunity to sit in the placement drive. I am very thankful to **Cognizant Technology Private Limited**. For having given me the opportunity to undertake my four months internship in their working area. It was a very good learning experience for me. I would like to convey my heartiest thanks to my manager and my mentor for their endless support in office, all teachers of Cognizant Technology Private Limited who brought me to my present performance and shape me like this during the last three successive years. Before I finish I would like to give my deepest thanks to all workers from the contractor side. Also for those who are not listed above but supported me in different areas, I would like to thank you all.

**Date:**

CHAPTER 1 .....	1
Introduction .....	1
1.1 DevOps .....	1
1.2 How DevOps work .....	3
1.3 Scope of the Project .....	4
CHAPTER 2 .....	8
Puppet .....	8
2.1 Introduction to Puppet .....	8
2.2 Puppet Code Basics .....	10
2.3 Puppet Architecture .....	12
CHAPTER 3 .....	15
Ansible .....	15
3.1 Introduction to Ansible .....	15
3.2YAML .....	15
3.3 Ansible Architecture .....	15
CHAPTER 4 .....	25
Installation and Configuration .....	25
4.1 Puppet Installation and Configuration .....	25
4.2 Establish communication between Puppet Master and Agents .....	27
4.3 Ansible Installation and Configuration .....	29
4.4 Establish communication between Ansible Master and Hosts.....	29
4.5 Comparison .....	29
CHAPTER 5 .....	30
Deployment .....	30
5.1 Creating Red Hat Linux Virtual Machine .....	30
5.2 Installation of Puppet Configuration Tool .....	30
5.3 Installation of Ansible Configuration Tool .....	30
5.4 Installation of LAMP Stack Using Puppet .....	31
5.5 Installation of LAMP Stack Using Ansible .....	32

CHAPTER 6 .....	35
Deploying Serverless Website .....	35
6.1 Serverless Computing .....	35
6.2 Building Blocks of serverless Computing .....	35
6.3 Benefits of Serverless Computing/Tradeoffs .....	40
6.4 AWS API .....	40
6.5 Steps for creating the serverless Website .....	43
CHAPTER 7 .....	47
Conclusions .....	47

## TABLE OF FIGURES

S.No.	Figures	Page No.
1	Fig 2.1: Architecture of Puppet	7
2	Fig 3.1: Architecture of Ansible	10
3	Fig 4.1: Hostnames for Instances in puppet	11
4	Fig 4.2: Installing Ansible	12
5	Fig 4.3: PHP version	14
6	Fig 4.4: Comparison of Ansible and Puppet	15
7	Fig 4.5: Index of Deployed Website	16
8	Fig 5.1: AWS Console	17
9	Fig 5.2: AWS Instance Console	18
10	Fig 5.3: Choosing AMI	18
11	Fig 5.4: Configuring Instance	19
12	Fig 5.5: Reviewing Instance	19
13	Fig 5.6: Creating Key-Value pair	19
14	Fig 5.7: Instance Created	20
15	Fig 5.8: Viewing Running Instances	20
16	Fig.5.9: Connecting to Instance	21
17	Fig.5.10: Installation Commands	21
18	Fig 5.11: Installing Puppet on master	22
19	Fig 5.12: Installing Puppet on Client	22
20	Fig 5.13: Configuring Puppet	22
21	Fig 5.14: List clients within master	23
22	Fig 5.15: Installation Commands	24
23	Fig 5.16: Testing the Webpage	27
24	Fig 5.17: Pinging Client	27
25	Fig 5.18: Testing the Webpage	29
26	Fig 5.19: Changing Client Directory	30

27	Fig 5.20: Sample html page	31
28	Fig 5.21: Page Running	31
29	Fig 6.1: Deployed Website Architecture	35
30	Fig 6.2: Website Running	38



## **Abstract**

In today's IT world with increasing computing power and demands of the world the server power and so does the manpower need to be increased. With increase in server counts the failure chances increase and so does the responsibility of humans to maintain them and keep them running at all costs. This is too much work and solving this automation is the best way to keep in control the demands of today and our project running at all times. The problem is solved by automating the deployment process. The general sending process comprises of a few interrelated exercises with potential changes between them. These exercises can happen at the maker side or at the buyer side or both. Since each product framework is interesting, the exact procedures or methodology inside every action can barely be characterized. Along these lines, "sending" ought to be decrypted as a general procedure that must be redone as per explicit necessities or qualities. In this project, we will be using two very popular configuration tools for deploying namely, Ansible and Puppet. Puppet is master server architecture that has a master which has configuration info that can be sent to nodes for them to be in a required state. It uses Ruby language which is not easy to learn. Ansible is same as puppet but has an agentless arch that helps them to deploy on the host machine. It has playbooks which are maintained so that target machines can be configured. It has YAML language. (It is a comprehensible information serialization language and is usually utilized for setup of documents).

# CHAPTER 1

## INTRODUCTION

Increasing demands of businesses leads to more number of servers and high work pressure from clients requires quicker work which in turn requires quicker server management today as compared to past. Cloud Computing and virtualization is gaining popularity in these times because of the exponential growth in computing for organizations.

Cloud Computing refers to the delivery of IT services and resources over the internet. It is a model for empowering whenever, anyplace, on-request organize access to a mutual or devoted pool of promptly accessible assets. This empowers an organization to store and access information or projects practically, for example in a cloud, as opposed to on nearby hard drives or servers.

DevOps is a process of utilizing the cloud-computing paradigms to bring more agility to development team. Most organizations comprehend that to build aggressiveness in today is quickly evolving world, they can't overlook computerized change. Devops and cloud are one of the sought after ways by which the companies can achieve the needed transformation.

As a consequence, apps like Puppet and Ansible which are known as mobile execution and deployment tools are required for this. These tools make work of admin servers as easy as it can be because it makes configuration of infra much easier as we only have to write code for server deploy. With a small code snippet we can deploy multiple servers and even configure them with a simple click of mouse as these tools also provide interfaces to interact and user can also write commands to do the desired work.

### 1.1 DevOps

DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably.

It is the mix of social rationalities, practices, and apparatuses that builds an association's capacity to convey applications and administrations at high speed: advancing and improving items at a quicker pace than associations utilizing conventional programming improvement and foundation the executives' forms.

While it's anything but a fixed philosophy, robotization and a community culture are the establishment for acknowledged DevOps rehearses, which include:

- Measuring the things that affect your association's objectives
- Making those estimations unmistakable to everybody
- Using a mutual arrangement of programming advancement devices and best practices
- Including all groups in the product conveyance process from intending to generation
- Automating the advancement pipeline and expelling bottlenecks for quicker conveyance

## **1.2 How Devops works**

In this software method, improvement and undertakings bunches are not any more drawn out "seloed". Every so often, these two gatherings are combined into a singular gathering where the engineers work over the entire application lifecycle, from progression and test to sending to exercises, and develop an extent of capacities not obliged to a lone limit.

These gatherings use practices to automate shapes that irrefutably have been manual and moderate. They use an advancement stack and tooling which help them work and create applications quickly and constantly.

## **1.3 Scope of the project**

Puppet and Ansible present various ways to accomplish a shared objective of overseeing huge scale server foundation productively, with insignificant contribution from designers and sys admins. The arrangement of config and executives apparatuses are intended to diminish the multifaceted nature of designing circulated foundation assets, empowering speed, and guaranteeing unwavering quality and consistence. The project aims at establishing a comparison between the two configuration management tools Ansible and Puppet, which are widely, used automation tool in the Devops culture.

The target audience includes members both from development team as well as from IT operations who are responsible for managing and configuring large-scale server infrastructure.

The serverless technology explained at the end allows developers to refrain from the task of managing database server or application server with improved scalability and speed.

## **SYSTEM REQUIREMENTS**

### **Hardware Requirements**

□ Four virtual machines each with following specifications

○ Red hat Enterprise Linux (RHEL HVM) ○

Processor 1GHZ ○ RAM min 4GB ○ Hard

Drive min 201GB

### **Software Requirements**

- Web Browser – Firefox or Chrome
- AWS Console Account

# Chapter 2

## Puppet

### 2.1 Puppet

Puppet is the mainstay of the software market. It has largest market percentage ratio in terms of usage which it should be as it has been around the longest i.e. since 2005 which is very long in case of CM industry. Many big companies like Google, Reddit, Paypal, Dell, oracle database operate their vast datacentres at various locations using puppet which gives them some sort of credible effect and let users trust this software easily. It has most dynamic and up to date GUI that does many tasks and runs on major OS' like Windows, Unix and Mac OS and is compatible with major cloud opeartors like AWS, Azure.

### 2.2 Puppet code basics

Puppet code consists of manifests and classes. Some of the important terminologies associated with puppet are as follows-

#### 1. Resources

Puppet has declarations which are resources that need to be installed on that particular machine.

Resources are declared as follows:

```
resource_type      {'resource_name'}

attribute => value

...

}
```

#### 2. Manifests

The code run in puppet are called manifests. The codes end with ‘.pp’ extension.

### 3. Classes

The classes used in puppet are code blocks that are reusable code snippets to be used again with some manifests.

## 2.3 Puppet Architecture-

Puppet Configuration Tool refers to client-server model where one virtual machine runs Puppet Master (as shown in fig. 1). Application that controls the configuration information & managed client server machines running the task of Puppet Agent Application fulfilling the background service while requesting its own configuration catalog by sending facts to Puppet Master.

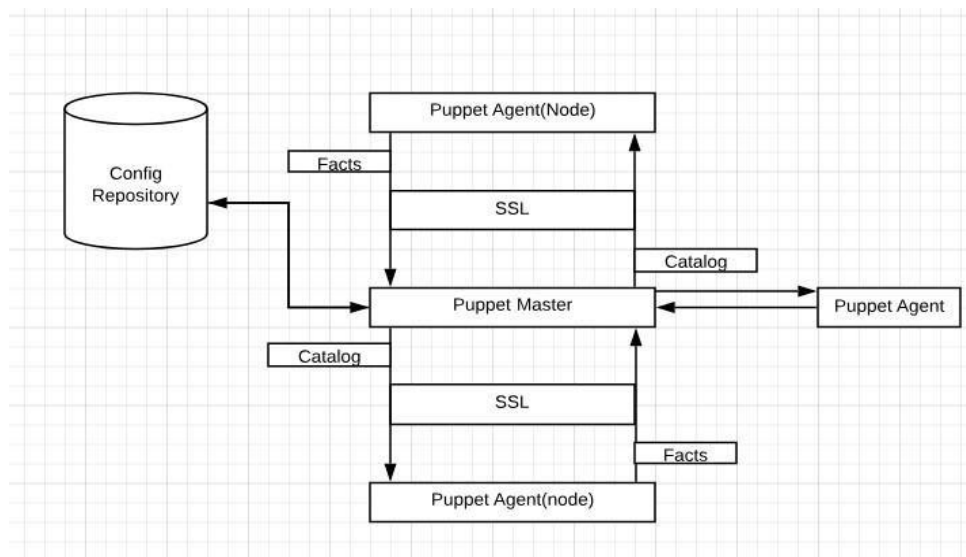


Fig 2.1: Architecture of Puppet

## Chapter 3

### Ansible

#### 3.1 Ansible

Ansible is an automation engine. It allows us to run tasks on automation servers. If we are using AWS and do not setup infrastructure such as EC2, DB, S3, etc. than ansible will set up all these for us automatically.

#### Use cases for Ansible:

- Provisioning environments: It can automate setting up and tearing down environments.
- Configuring operating system: It install and remove operating system.
- Deploying application: It performs blue/green deployment.
- Performing compliance check:

It uses no agents and no additional custom security infrastructure, so it's easy to deploy and most importantly, it uses a very simple language (YAML).

#### 3.2 YAML

It stands for "*YAML Ain't Markup Language*". It is a data serialization language, which uses .yaml or .yml file extension. It is white space sensitive; if you insert white space at wrong place then whole file will be corrupted. Types used various data in YAML such as Integers, Strings, Null and Boolean.

YAML includes block collections, which can have distinguished from other quantities with an identification of key value pair included in them.

- Mapping: It is the representation of key value as included in JSON structure. Its basic representation is given below.

Key: value

- Nested mapping:

Outerkey:

InnerKey: innervalue

- Inline syntax:

Outerkey: {innerkey: innervalue}

□Sequence (list or array): It represents a series of nodes. Each item is denoted by a leading “-” indicator separated with a white space from the nodes. Its basic representation is given below: -

-value1

-value2

-value3

### 3.3 Ansible Architecture

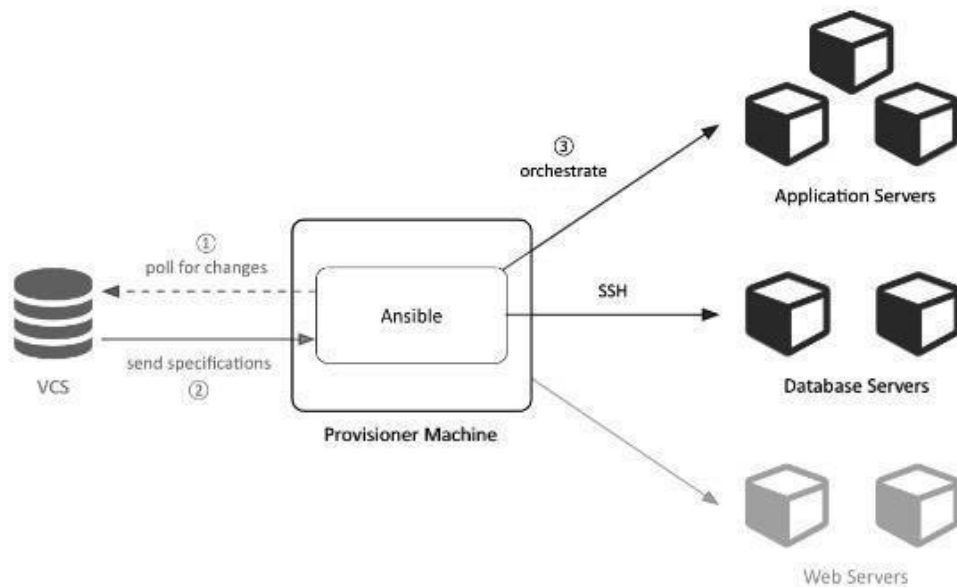


Fig 3.1: Architecture of Ansible



- Inventories: Agentless Automation Configuration Tool running on servers that automates cloud provisioning, configuration management, application deployment & other requirements in IT Sector.
- Modules: Ansible connects to instances by pushing out program called modules that is executed in desired state of system & terminated after accomplishment of task.
- API: used as communication medium to cloud services.
- Plugins: An add-on functionality which is offered by Ansible while allowing you to create own plugin with customized code.
- Playbook: Instruction Manual written in YAML human readable language describing tasks to be done with declaration of configuration without need to remembering any special syntax.

# Chapter 4

## Installation and Configuration

### 4.1 Puppet installation and Configuration

After ensuring that all the prerequisites are met such as creation of four virtual machines with specified configurations begin with installation of Puppet master server

1. Install Puppet server on the Puppet master server. Any one of the virtual machine can be referred to as puppet master server with hostname puppet master.
2. Configure memory allocation and Start Puppet Server.
3. Installing the Puppet Agent

The agent can be configured on any server to let it communicate with the master so that the master could manage it. Here the other three virtual machines are used as a host for the puppet master.

4. After installing the puppet-agent package start the puppet agent.

The puppet uses certificate signing method so that master and agent can communicate. The first time installed agent creates the SSL certificate which is sent to be signed by master and after the signing it can communicate with agent.

The following infrastructure has been created with designated hostnames-

Hostname	Role	Security Group
Puppetmaster	Puppet master	Launch-wizard-13
Client1	Puppet agent	Launch-wizard-14
Client2	Puppet agent	Launch-wizard-15
Client2	Puppet agent	Launch-wizard-15

Fig 4.1: Hostnames for Instances in puppet

## 4.2 Establish communication between Puppet Master and Agents

### Sign certificates on Puppet Master

Initially when agent runs the request to sign goes to particular master for the communication to start. The certificate signing process is as follows:

- Firstly, currently certificate requests to be signed are noted which are pending requests from the puppet nodes.
- Sign the certificate request with hostname
- For modification on hosts like removing or regenerating them particular host's certificate is needed.

After execution of the above steps, Ansible master and ansible client can communicate with each other successfully.

## 4.3 Ansible Installation and Configuration

- Connect to your instance with key pair & public DNS extracted from AWS ➤ Enable ansible package repository in virtual dedicated machine.
- Install ansible package on the Master Server Virtual machine, which has Puppet master already installed in it.
- Verify if ansible is successfully installed. The following screen conforms the installation of Ansible on the master server.

```
root@puppetmaster:~  
[root@puppetmaster ~]# ansible --version  
ansible 2.7.10  
  config file = /etc/ansible/ansible.cfg  
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']  
  ansible python module location = /usr/lib/python2.7/site-packages/ansible  
  executable location = /bin/ansible  
  python version = 2.7.5 (default, Mar 26 2019, 22:13:06) [GCC 4.8.5 20150623 (Red Hat 4.8.5-36)]  
[root@puppetmaster ~]#
```

Fig 4.2: Installing Ansible

## 4.4 Establishing communication between ansible master and hosts

- Enter the inventory of the ansible by entering command to list the hosts, which need to be connected to the master.
- Edit the file by making entry of the hosts and save the file. Create a group [test] and add hosts into it.

[test]

Client1

Client2

Client3

## 4.5 Comparison task

In order to compare the performance of both Ansible and Puppet we try to carry out the following tasks which ensures both Ansible and Puppet are used on the same testbed

- Installation of Lamp stack using Puppet
  - Create new module structure to lamp on puppet master server.
  - Create a manifest file init.pp in the manifests folder with specification to install lamp stack under the class.
  - Now use the module lamp by editing main manifest file of puppet server to install lamp stack in specified agent client servers.
  - Puppet client server will be installing lamp stack by pulling configuration from master server with command puppet agent -test.
  - Verify installation by executing the following link on web browser.

[http://puppetclient\\_ipaddress/info.php](http://puppetclient_ipaddress/info.php)



Fig 4.3 : PHP version

➤ Installation of Lamp stack using Ansible

○ Adding hosts to the Ansible Master Server

- Generate the public key of ansible master using sshkeygen
- Copy the public key by entering the following directory- vi /root/.ssh/id\_rsa.pub.
- Login to the host vm and paste the copied key in authorized\_keys folder.

○ Final setup to lamp stack with Ansible tool

Create a lamp\_stack.yml playbook with required specification for the installation of the lamp stack on the Ansible master

- Execute the above created lamp stack playbook with the ansible-playbook command.
- Verify the installation by visiting the same link as above for the puppet.

➤ Deploying a web application on the lamp stack.

- Create a sample web application for deploying on both Ansible and Puppet hosts.
- Change the client server directory to /html where the web application can be copied. This step is carried out for both Ansible and Puppet.

- Open the hostname with specified name of the web page to ensure successful deployment of the application.

Comparison Factor	Puppet	Ansible
❖ <b>Ease Of Setup</b>	Driven on Client-Server Model where puppet server runs on master machine & puppet client runs as agent on client machine.	✓ Agentless based model where client machine requires no special setup therefore faster to install.
❖ <b>Management</b>	Not easy to manage as using own language Puppet DSL where client pulls configuration from server on non-immediate basis of execution.	✓ Easy to manage as master server pushes configuration on all client servers under immediate remote execution.
❖ <b>Dependency</b>	Puppet requirement could be more depending upon configuration management.	✓ Ansible will need minimal requirement for controller node with ssh direct access.

Fig 4.4: Comparison of Ansible and Puppet

After successful deployment of the web application, we can see the following by visiting the browser for both Ansible and Puppet hosts.

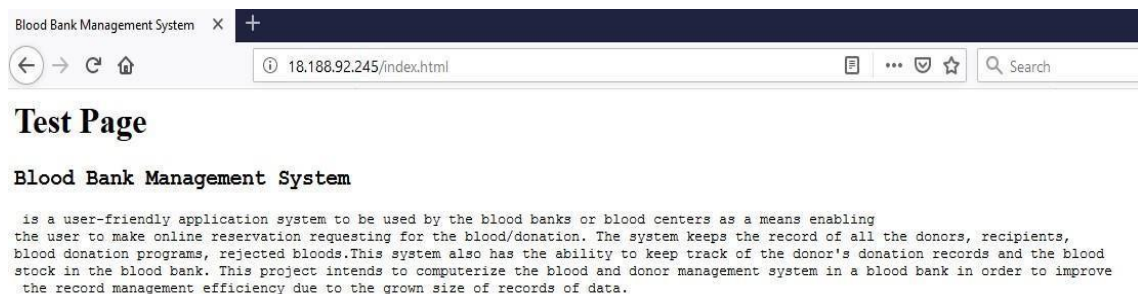


Fig 4.5: Index of Deployed Website

# Chapter 5

## Installation and Deployment

Puppet and Ansible do the same thing of automating software installation and deployment very easily and effectively with code. These different tools like Puppet, Ansible and many others are designed to achieve this aim with increased trust, less prone to human error and increased performance of our apps and projects. This project is aimed at comparing the performance of both the tool over same testbed.

### 5.1 Creating Red Hat Enterprise Linux Virtual Machine

1. Open the Console -

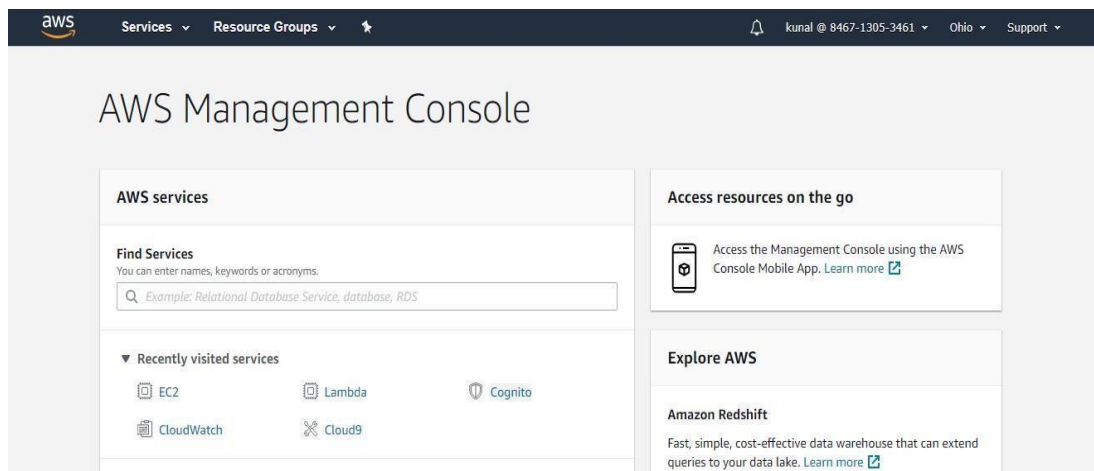


Fig 5.1: Management Console

2. Select compute Service & click on Launch to power up irtual machine.

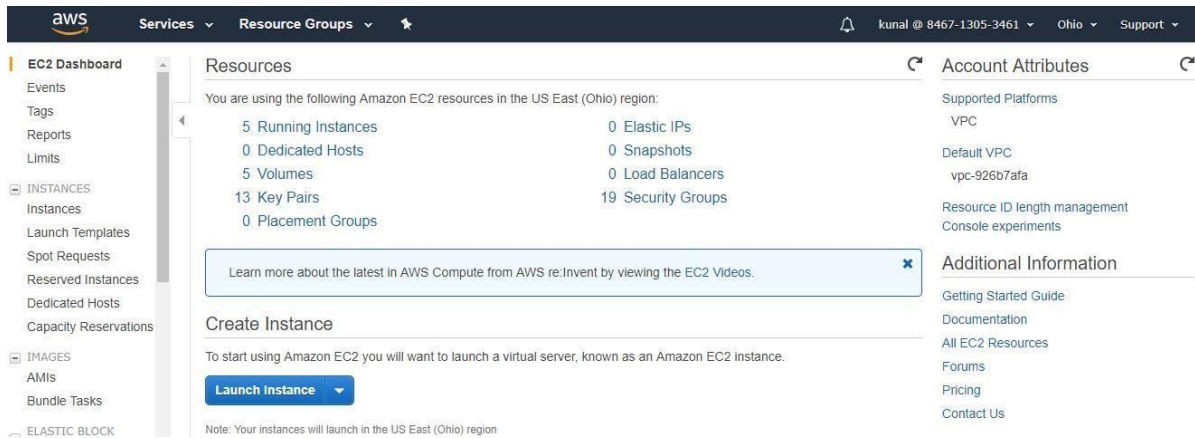


Fig 5.2: AWS Instance Console

### 3. Choose the required AMI.

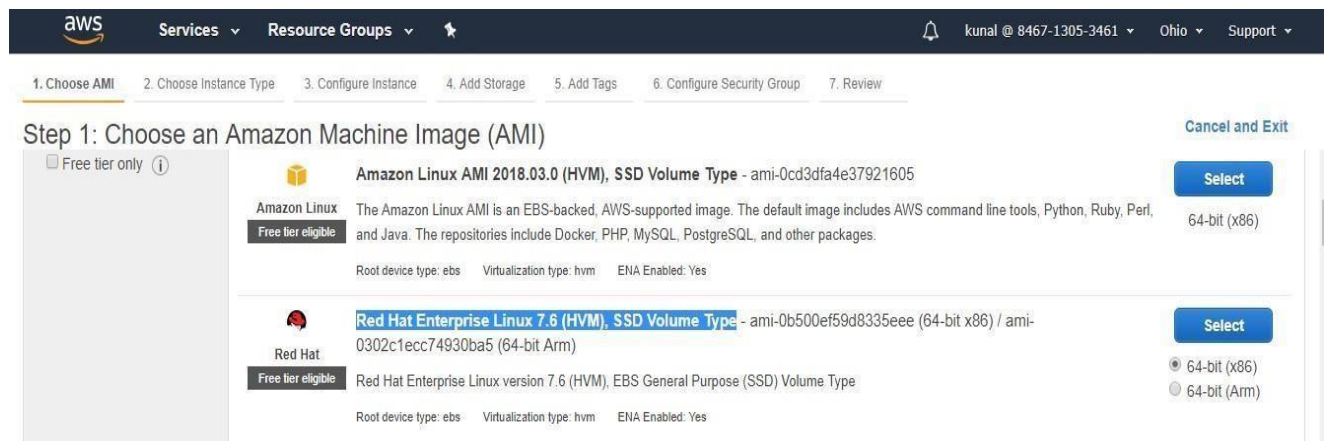


Fig 5.3: Choosing AMI

### 4. Choose **t2.medium** Instance Type to meet required computing needs for the project.



**Step 2: Choose an Instance Type**

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.medium (Variable ECUs, 2 vCPUs, 2.3 GHz, Intel Broadwell E5-2686v4, 4 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes

Fig 5.4: Configuring Instance

5. Click on Review & Launch instances to review instance details.

**Step 7: Review Instance Launch**

AMI Details: Red Hat Enterprise Linux 7.6 (HVM), SSD Volume Type - ami-0b500ef59d8335eee

Free tier eligible: Red Hat Enterprise Linux version 7.6 (HVM), EBS General Purpose (SSD) Volume Type

Root Device Type: ebs | Virtualization type: hvm

Instance Type: t2.medium

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.medium	Variable	2	4	EBS only	-	Low to Moderate

Fig 5.5: Reviewing Instance

6. Click Launch to create a new key pair and connect via SSH.

Select an existing key pair or create a new key pair

A key pair consists of a public key that AWS stores, and a private key file that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

Key pair name: projectkey

Download Key Pair

You have to download the private key file (\*.pem file) before you can continue. Store it in a secure and accessible location. You will not be able to download the file again after it's created.

Cancel | Launch Instances

Fig 5.6: Creating Key-Value pair

7. Click on Launch Instances & your instance is now launching under the environment.

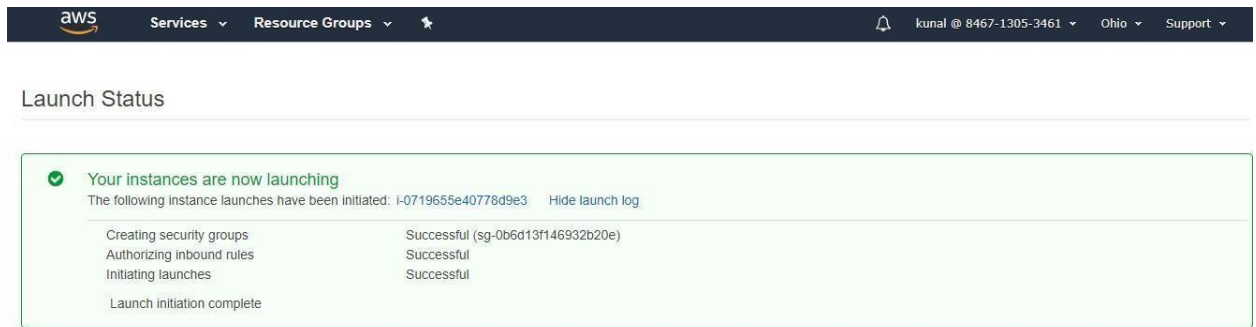


Fig 5.7: Instance Created

8. Virtual machine is successfully created & running under instances.

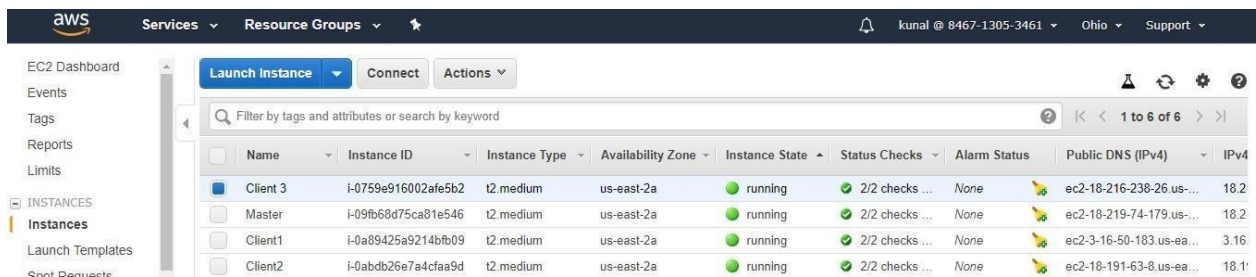


Fig 5.8: Viewing Running Instances

## 5.2 Installation of Puppet Configuration Tool

### □ Architecture

Puppet Configuration Tool refers to a client-server model where one virtual machine runs the Puppet Master Application that controls the configuration information & manages client server machines running the task of Puppet Agent Application fulfilling the background service while requesting its own configuration catalog by sending facts to Puppet Master.

1) Connect to your instance with key pair & public DNS extracted from AWS.

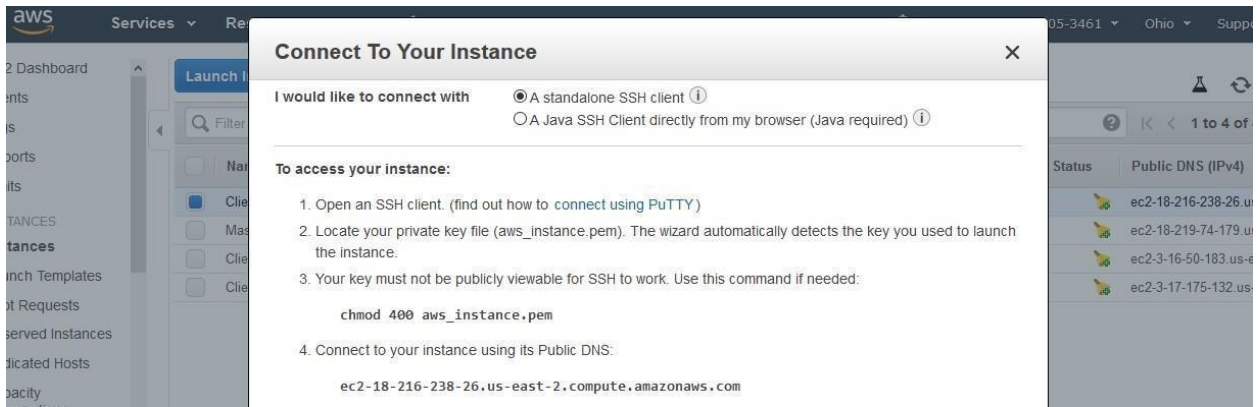


Fig.5.9: Connecting to Instance

- 2) Enable puppet package repository in all virtual machine dedicated to master & client servers as pre-requisite.

```
[root@client3 ~]# sudo rpm -ivh https://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm
Retrieving https://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm
Preparing... ##### [100%]
package puppetlabs-release-22.0-2.noarch is already installed
[root@client3 ~]#
```

Fig.5.10: Installation Commands

- 3) Execute the following command to install puppet on puppet master server.

```
[root@puppetmaster ~]# sudo yum install puppetserver
Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
Package puppetserver-1.2.0-1.el7.noarch already installed and latest version
Nothing to do
[root@puppetmaster ~]#
```

Fig 5.11: Installing Puppet on master

- 4) Execute the following command to install puppet agent application on dedicated client server/virtual machines with init puppet script.

```
[ec2-user@client3 ~]$ sudo yum install puppet
Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
Package puppet-3.8.7-1.el6.noarch already installed and latest version
Nothing to do
[ec2-user@client3 ~]$
```

Fig 5.12: Installing Puppet on Client

## Basic Configuration

o Puppet configuration tool is now successfully installed on Master & Client Server, add following inbound rules under dedicated instances security groups on AWS for enabling restriction free communication between Master & Client Servers.

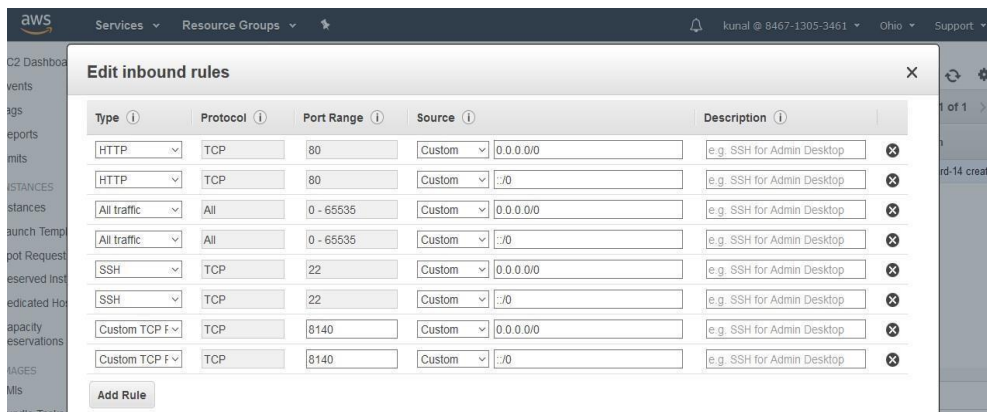


Fig 5.13: Configuring Puppet

-Run `vi /etc/hosts` in master server & list hosts which needs to be communicated as client with master server. Follow the same task in client server before starting with configuration of generating & assigning the signature to server requests & communication.

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
172.31.6.42 puppet puppetmaster.example.org

172.31.7.177 puppetclient3.example.org
172.31.1.239 puppetagent2.example.org
172.31.4.196 puppetclient2.example.org
172.31.0.219 client3.example.org
3.16.50.183 client1.example.org
```

Fig 5.14: List clients within master

## 5.3 Installation of Ansible Configuration Tool

### □Architecture

- Inventories: Agentless Automation Configuration Tool running on servers that automates cloud provisioning, configuration management, application deployment & other requirements in IT Sector.
- Modules: Ansible connects to instances by pushing out program called modules that is executed in desired state of system & terminated after accomplishment of task.
- API: used as communication medium to cloud services.
- Plugins: An add-on functionality that is offered by Ansible while allowing you to create own plugin with customized code.
- Playbook: Instruction Manual written in YAML human readable language describing tasks to be done with declaration of configuration without need to remember any special syntax.

1) Connect to your instance with key pair & public DNS extracted from AWS.



Fig 5.15: Installation Commands

2) Enable Ansible package repository in virtual dedicated machine.

```
[root@puppetmaster ~]# rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
Retrieving https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
Preparing... ##### [100%]
package epel-release-7-11.noarch is already installed
[root@puppetmaster ~]#
```

- 3) Execute following command to install Ansible on your master server.

```
root@puppetmaster:~
[root@puppetmaster ~]# sudo yum -y install ansible
Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
[root@puppetmaster ~]#
```

- 4) Ansible is successfully installed & you can verify the same by executing following command.

```
root@puppetmaster:~
[root@puppetmaster ~]# ansible --version
ansible 2.7.10
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /bin/ansible
  python version = 2.7.5 (default, Mar 26 2019, 22:13:06) [GCC 4.8.5 20150623 (Red Hat 4.8.5-36)]
[root@puppetmaster ~]#
```

## Basic Configuration

- 1) Enter the inventory of the Ansible by entering the following command to list the hosts, which need to be connected to the master.

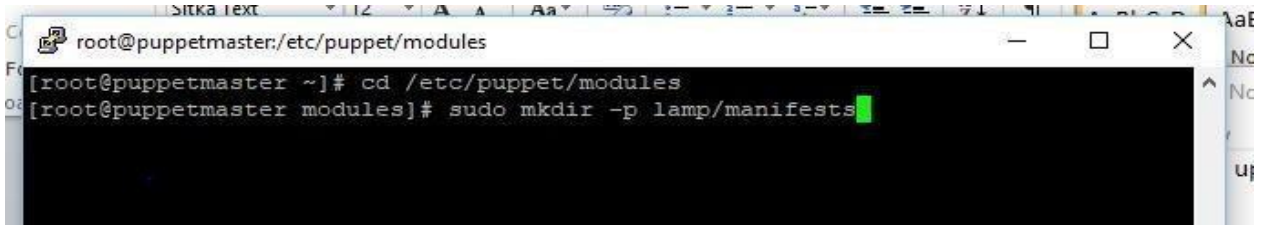
```
root@puppetmaster:~
[root@puppetmaster ~]# vi /etc/ansible/hosts
```

- 2) Edit the file by making entry of the hosts and save the file. Create a group [test] and add hosts into it.

```
[test]
Client1
Client2
Client3
```

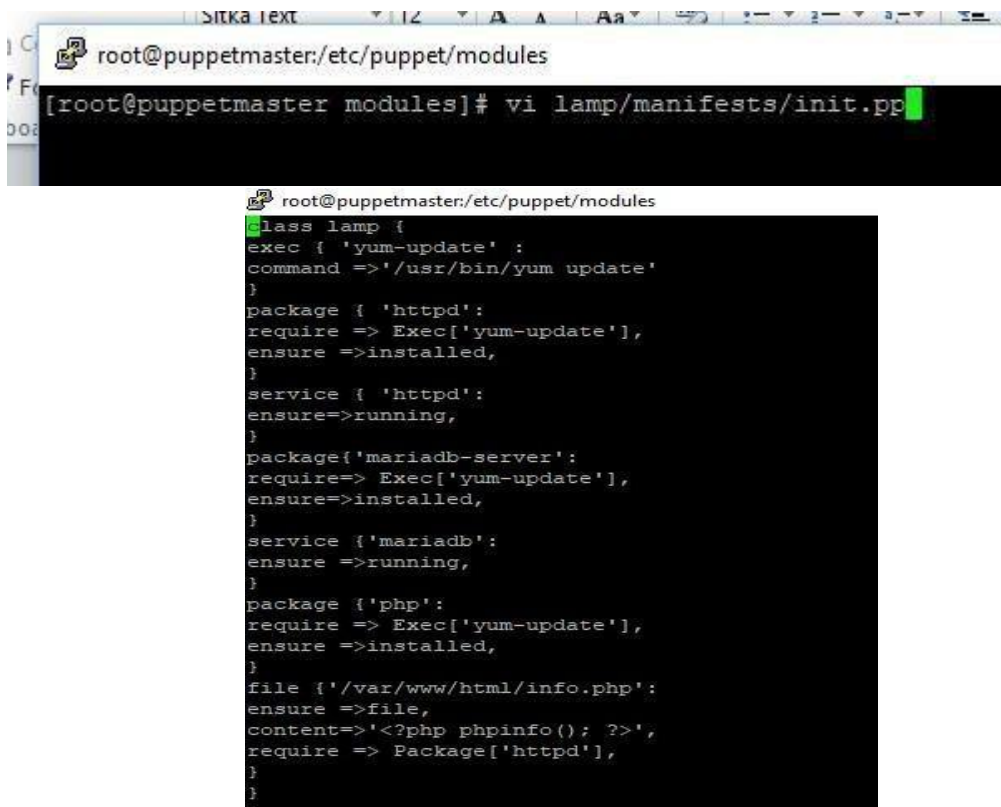
## 5.4 Installation of LAMP Stack using Puppet

- 1) Create new module structure to lamp on puppet master server.



```
root@puppetmaster:/etc/puppet/modules
[root@puppetmaster ~]# cd /etc/puppet/modules
[root@puppetmaster modules]# sudo mkdir -p lamp/manifests
```

- 2) Create manifest file init.pp in module with specification to installation execution of LAMP Stack under the class.



```
root@puppetmaster:/etc/puppet/modules
[root@puppetmaster modules]# vi lamp/manifests/init.pp

class lamp {
  exec { 'yum-update' :
    command => '/usr/bin/yum update'
  }
  package { 'httpd':
    require => Exec['yum-update'],
    ensure => installed,
  }
  service { 'httpd':
    ensure => running,
  }
  package { 'mariadb-server':
    require => Exec['yum-update'],
    ensure => installed,
  }
  service { 'mariadb':
    ensure => running,
  }
  package { 'php':
    require => Exec['yum-update'],
    ensure => installed,
  }
  file { ['/var/www/html/info.php']:
    ensure => file,
    content => '<?php phpinfo(); ?>',
    require => Package['httpd'],
  }
}
```

- 3) Now use the module **lamp** by editing main manifest file of puppet server to install lamp stack in specified puppet agent client servers.

```
root@puppetmaster:/
[root@puppetmaster /]# vi /etc/puppet/manifests/site.pp
```

4) Puppet client server will be installing lamp stack by pulling configuration from master server with command puppet agent --test

```
root@ip-172-31-4-196:~
Info: Loading facts in /etc/puppet/modules/java/lib/facter/java_default_home.rb
Info: Loading facts in /etc/puppet/modules/java/lib/facter/java_libjvm_path.rb
Info: Loading facts in /etc/puppet/modules/java/lib/facter/java_major_version.rb
Info: Loading facts in /etc/puppet/modules/java/lib/facter/java_patch_level.rb
Info: Loading facts in /etc/puppet/modules/java/lib/facter/java_version.rb
Info: Loading facts in /etc/puppet/modules/archive/lib/facter/archive_windir.rb
```

Verify installation by executing following link convention on web browser -  
[http://puppetclient\\_ipaddress/info.php](http://puppetclient_ipaddress/info.php)



Fig 5.16: Testing the Webpage

## 5.5 Installation of LAMP Stack using Ansible ○

### Adding hosts to the Ansible Master Server

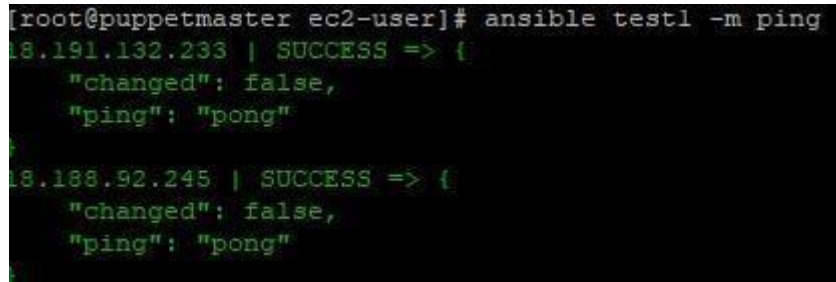
- 1) Generate the public key of ansible master ssh-keygen
- 2) Copy the public key by entering the following directory - **vi /root/.ssh/id\_rsa.pub.**
- 3) Login to the host vm and paste the copied key into the following folder **vi /root/.ssh/authorized\_keys.**



To verify the connection between ansible master and its hosts enter the following command on to the ansible master-

- ansible <hostname/hostgroup> -m ping.

After successful communication, we see the following screen.



```
[root@puppetmaster ec2-user]# ansible test1 -m ping
18.191.132.233 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
18.188.92.245 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

Fig 5.17: Pinging Client

## 5.5 Final setup to LAMP Stack with Ansible Tool

1) Create a **lamp\_stack.yml** playbook with the following content

```
hosts: test tasks:
```

```
# Apache installation -name: install Apache yum:
```

```
pkg=httpd state=latest - name: ensure Apache is
```

```
running service: name=https state=started
```

```
enabled=yes - name: install Hello World Php script
```

```
copy: src=index.php dest=/var/www/html/info.php #
```

### **Mysql installation**

```
- name: Install mariaDb server package yum:
```

```
name=mariadb-server
```

```
state=present
```

```
- name: Start Mysql service
```

```
service: name=mariadb state=started
```

Execute the above created playbook using following command

– **ansible-playbook lamp\_stack.yml**

```

ok: [18.191.132.233]
ok: [18.188.92.245]

TASK [install Apache] *****
ok: [18.188.92.245]
ok: [18.191.132.233]

TASK [ensure apache is running] *****
ok: [18.188.92.245]
ok: [18.191.132.233]

TASK [install Hello World PHP script] *****
ok: [18.188.92.245]
ok: [18.191.132.233]

TASK [Install mariaDb server package] *****
ok: [18.191.132.233]
ok: [18.188.92.245]

TASK [Start Mysql service] *****
changed: [18.191.132.233]
changed: [18.188.92.245]

TASK [Install python Mysql package] *****
ok: [18.191.132.233]
ok: [18.188.92.245]
    to retry, use: --limit @/etc/ansible/lamp_stack.retry

PLAY RECAP *****
18.188.92.245      : ok=7    changed=1    unreachable=0    failed=0
18.191.132.233   : ok=7    changed=1    unreachable=0    failed=0

```

- Verify installation by executing following link convention on web browser **<http://18.188.92.245/info.php>**



Fig 5.18: Testing the Webpage

## □ Copy files from Master to Client Server

The copy module copies a file from the local or remote machine to a location on the remote machine.

1. Create a playbook named copy\_files.yml with following content in it-

```
- name: Ansible copy file to a remote hostgroup.

copy:
  hosts: test
  src: /etc/ansible/sample.txt
  dest: /tmp

tasks:
```

2. Execute the above created playbook- ansible-playbook copy\_files.yml

## Deployment Phase

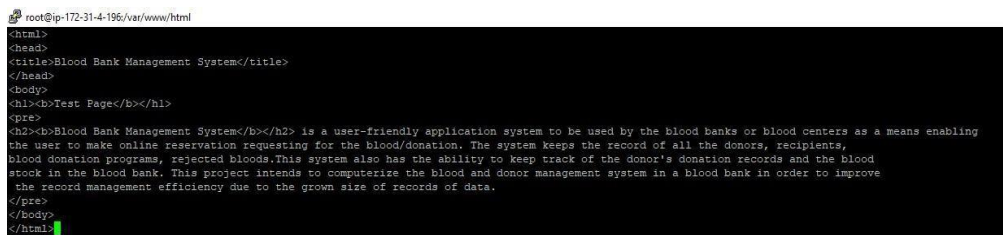
1. Change client server directory to /var/www/html folder.



```
root@ip-172-31-4-196:/var/www/html
[root@ip-172-31-4-196 ~]# cd /var/www/html/
[root@ip-172-31-4-196 html]# ls
index.html  index.php  info.php
[root@ip-172-31-4-196 html]#
```

Fig 5.19: Changing Client Directory

2. Create sample web page file for deployment using command vi filename



```
root@ip-172-31-4-196:/var/www/html
<html>
<head>
<title>Blood Bank Management System</title>
</head>
<body>
<h1>Test Page</h1>
<p>
<h2>Blood Bank Management System</h2> is a user-friendly application system to be used by the blood banks or blood centers as a means enabling
the user to make online reservation requesting for the blood/donation. The system keeps the record of all the donors, recipients,
blood donation programs, rejected bloods.This system also has the ability to keep track of the donor's donation records and the blood
stock in the blood bank. This project intends to computerize the blood and donor management system in a blood bank in order to improve
the record management efficiency due to the grown size of records of data.
</p>
</body>
</html>
```

Fig 5.20: Sample html page

3. Open `puppetipaddress/webpagename` for testing to successful deployment.

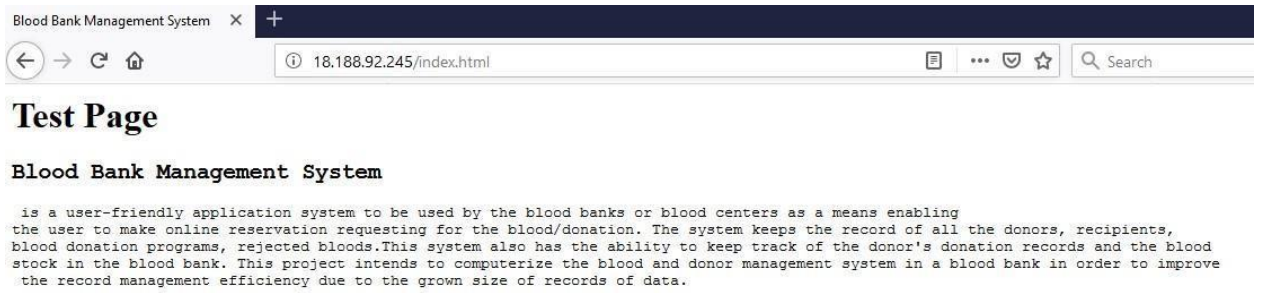


Fig 5.21: Page Running

# CHAPTER 6

## Serverless Computing

### 6.1 Serverless Computing

It is first started by amazon lambda, in this server are still involved but these resources are managed by the cloud provider. It works as function as a server. There are number of cloud platforms available such as Microsoft Azure, AWS, Google cloud platform (GCP). We only pay for the time our function runs, we do not manage or own the infrastructure but we still need to manage some aspects of serverless function, it is serverless developer friendly. All platform will provide us better availability and scalability.

In this aws lambda manages creation and deletion of containers, there is no actual API for us to actually manage the container. Lambda function tries to reuse the container for subsequent invocation of lambda function. If we want to add some storage, so we can check if the action is already maintained by adding the logical code so if connection is already there in that case.

We can reuse it instead of making a new connection. It supports various Languages:

- Node.js version – v0.10.36, v4.3.2
- Java – Java8
- Python – Python 2.7
- .Net Core - .Net Core 1.0.1(C#)

It supports invoke method to trigger. There are three types of invocation method:

- X- AMZ-Invocation-Type: DryRun/RequestResponse(Default)/Event
- X-AMZ: LogType
- X-AMZ-Client-Context: ClientContext

We can trigger a lambda function by using the 'INVOKE' method by setting up our code to trigger from our other AWS Services, HTTP endpoint or in-app activity. So while selecting the 'PUSH' event mode should be synchronous and in 'PULL' event mode is defined by the lambda server.

Another type is DryRun that indicates if anything else except the code is run:

We can invoke the function by HTTPS or by REST API or one can setup its own gateway by the help of Amazon API gateway. REST stands for Representational State Transfer, which makes

request unicorn system to have and manipulate textual web representation using predefined code and functions.

Rest interface say nothing about the structure of the API, It makes accessing web application much more flexible and maintainable.

## 6.2 Building blocks of serverless computing

- 1.) **Restful API:** It is based on HTTP/HTTPS as a transfer protocol, used to exchange textual representation of web services to other web applications using predefined methods such as get, post, push, patch, delete, etc. Standard exchange format for this is JSON (JavaScript Object Notation), is a lightweight data interchange format. It is easy for human to read and write and also easy for machine to parse and generate. It is based on the subset of the JavaScript programming language.
- 2.) **Stateless Function:** It can be invoked manually using command line interface (CLI). We can estimate the concurrent execution in lambda function by the help of this formula:  
$$\text{Event per second} * \text{function duration}$$
- 3.) **Micro Service design pattern:** State Independence, Decoupled Layers, No dependency on other services, function written in native language, Restful API, Enhanced client.

Lambda functions are stateless, fast, execute the code in milliseconds, manages compute resource requirement for your function, provide built in logging and monitoring using CloudWatch used for real time stream processing. By using lambda function, all the base infrastructure is provided for us by AWS as a service provider, service provider runs our code on a highly available compute infrastructure and performs all the infrastructure and performs all the administrator task of the compute resources and that is going to include server itself, OS, maintenance impaction, capacity provisioning.

We can create our own backend that impact operates with AWS doing all the scale, performance and security; we can even set the memory allocation. The larger the memory allocation also represents an

increase in processor and network performance, so we have enough control over the performance, which enables us to increase the performance of the base layer if needed. Each instance comes with up to 500 megabytes of temporary storage.

### 6.3 Benefits of Serverless computing/Tradeoffs

- Zero Administration
- Easily Scaled
- High availability and fault tolerance
- Compatible with micro services.
- Not compatible with all applications
- Reduces costs

### 6.4 AWS API

AWS API gateway service provides an easy way to create RESTful application programmable ends. It makes it easy to design your own resources and structure, add dynamic routing parameters and develop custom authorization logic. Each API resource can be configured independently, while each stage can have a specific cache and logging specifications.

The application architecture uses AWS Lambda, Amazon API Gateway, Amazon S3, Amazon Dynamo DB, and Amazon Cognito as pictured below:



Fig 6.1: Deployed Website Architecture

## 6.5 Steps for creating the serverless Website

- (i) Steps for hosting the static website.
- (ii) Steps for managing the users.
- (iii) Steps for building a server less backend.
- (iv) Steps for deploy a RESTful API.

### Step 1. Hosting a static website

To host the site the first step is to create the S3 bucket for HTML files, js files and other vital parts of the website for storage. In the other consequent steps the sites dynamic capabilities are explored using Lambda functions and amazon's API service.

We make the static web content including HTML, CSS, JavaScript, images and other files that will be stored in amazon S3, after that the end user will then access the site using the public website.

- a. Select the region.
- b. Create the bucket.
- c. Upload the content in the bucket

Add the Bucket policy because the bucket policies are JSON documents which are used for access permissions and various requests on the contents in the bucket.

- i. By adding the bucket policy allows for public Reads.
- ii. Enable website hosting. (To enable website host service)
- iii. Validate the previous steps by writing the website URL in the browser.

### Step: 2 Managing the users

In these steps we are going to manage the users by this way, whenever new users access our app we let them create new id. For the same, are providing the two input field to register i.e. (name and password). Then Amazon Cognito provides the additional attribute in the Applications. When new and first time users register, Amazon Cognito mails them a secret digit code which customers fill to continue. However, Amazon Cognito provides new and more attributes in the apps of users.



When users continue after their registration form submission process the in talk Amazon service will sends a secret code to the particular user which submitted their form. To see whether their account is confirmed or not they have to return to their site and proceed further on with their email id and code sent to them by entering it in the form. The admin can himself manually confirm their id's by going to console and himself confirming the user.

After the verification is completed they are enabled access to sign in to their account, when users slog in they enter their user id and security password. A JavaScript function is used to communicate with Amazon Cognito, it can communicate with Cognito using Secure Remote Password Protocol (SRP) and they received back the set of JSON Web Tokens (JWT).

The JWTs contains the identity about the user and will be used in the next step to authenticate against the RESTful API that is used with Amazon API Gateway.

- i. Create an new Cognito Pool.
- ii. Add the required app to user Pool.
- iii. Update the config.js file in the js folder where update the user pool Id, user pool client Id and region.
- iv. Test the implementation by going to the user registration page.

### **Step: 3**

Amazon DynamoDB and AWS Lambda is used for handling the backend process the web application. In the previous step unicorn is used to send to the location of their choice. In order to complete that request, the JAVASCRIPT running in the browser is used to invoke a service running in the cloud. Lambda function is used to invoke the request each time the user requested for unicorn. The function records the request from the DynamoDB table and then responds to the front-end application with details about the dispatched unicorn.

With the Amazon API Gateway, the function is invoked from the browser. We are going to implement this in next step.

- i. Create the database table.
- ii. Create the role for the lambda function.
- iii. Create a lambda function for handling the request.

- iv. Validate the implementation to check the browser based application.

#### **Step:4**

In this step, we are using Amazon API Gateway as a RESTful API by which we will get to know about lambda function we used in the previous step. The url is accessed through outside internet. Amazon Cognito user pool to secure them.

With this configuration, our static app turned into dynamic site application with the help of js functions in lambda which dynamically calls the API of our earlier step.

We are using Amazon API Gateway to use the Lambda function that we built in the previous module. With this configuration, our static app turned into dynamic site application with the help of js functions in lambda which dynamically calls the API of our earlier step.

The main aim of this step is used to build the cloud components of the API.

- i. REST API is created.
- ii. Create a user pool Authorizer. iii. Create a new method and resource. iv. Install the API.
- v. The config file is updated.
- vi. Update the invoke URL in config file.
- vii. Validate the implementation by running the ride.html in the browser to check whether it pick up the user location or not.

After the successful deployment of the serverless web, Application this screenshot is captured:

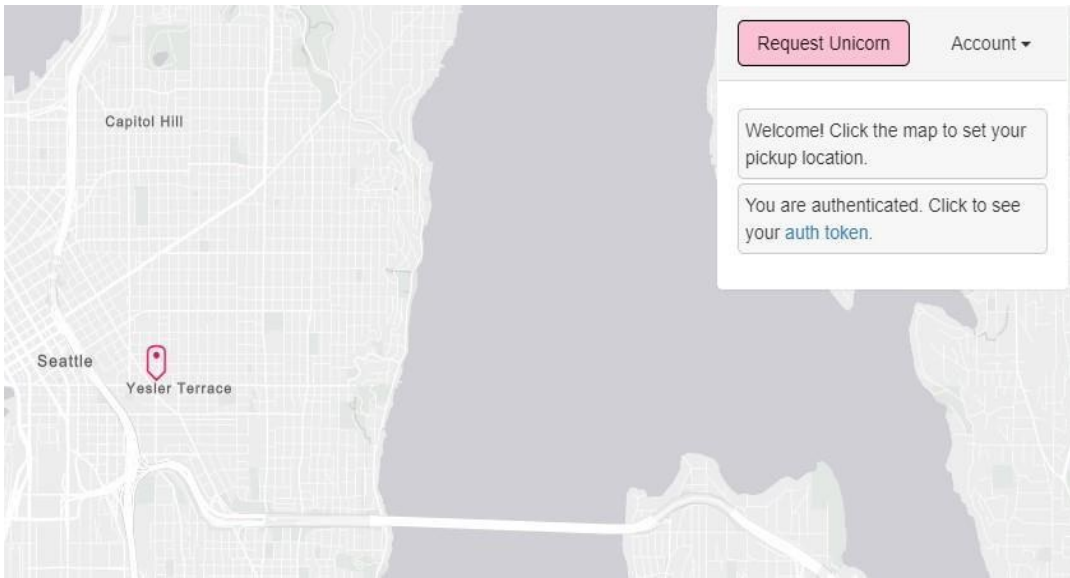


Fig 6.2: Website Running

## **CHAPTER 7**

### **CONCLUSION**

In this Project, the concept of automation using shell scripting is executed. The Configuration Framework Used i.e., Ansible and Puppet are used to make the deploy the web app onto a server. Using automation, it is made sure that repetition of code is not done in all the clients. All the repetition is eliminated using a single code from Ansible via Playbooks and Puppet via Manifests. At the end, it is consequential that both the tools that are used for automation and deployment should be used throughout the whole server life from powering up to installing different required software and ultimately giving up a server. It makes life easier for server administrator.

## References

[1] Ansible Simple IT Automation <<https://docs.ansible.com>>

[2] Puppet Documentation <<https://puppet.com/docs>>

[3] AWS Resource Documentation <<https://docs.aws.amazon.com>>

[4] Getting Started with Ansible <<https://app.pluralsight.com/library/courses>>

[5] Getting Started with Puppet <<https://app.pluralsight.com/library/courses>>

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: .....

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: Kanav Singla Department: CSE Enrolment No 151397

Contact No. 98153-76371 E-mail. singlakanav203@gmail.com

Name of the Supervisor: Mr. Bristley Sathiyaraj & Dr. Hemraj Saini

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): COMPARISON OF SOFTWARE ORCHESTRATION PERFORMANCE TOOLS AND SERVERLESS WEB APPLICATION

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages = 45
- Total No. of Preliminary pages = 35 8
- Total No. of pages accommodate bibliography/references = 1

(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at 21.....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
<u>23/5/2019</u>	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• <u>14</u> Words String</li> </ul>	<u>12 %</u>	Word Counts	<u>5,035</u>
Report Generated on			Character Counts	<u>28,116</u>
<u>23/5/2019</u>		Submission ID	Total Pages Scanned	<u>41</u>
		<u>1134833085</u>	File Size	<u>4.5819</u>

Checked by [Signature]  
Name & Signature 23/5/19

[Signature]  
23/05/2019  
LIBRARIAN

LEARNING RESOURCE CENTER  
Jaypee University of Information Technology

**Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)**