For this project, please think about how you would architect a scalable and secure static web

application in AWS.

• Create and deploy a running instance of a web server using a configuration management

tool of your choice. The web server should serve one page with the following content.

<html>

<head>

<title>Hello World</title>

</head>

<body>

<h1>Hello World!</h1>

</body>

</html>

To Solve this problem we have to make use of configuration management tools which works with AWS, there are many configuration management tools in the market like Ansible, Puppet, Saltstack and Chef.

I am using Chef to demonstrate bootstrapping and apply the same principles.

First we will create a simple Chef Cookbook that installs an apache web server and deploys a “Hello World” site. The Chef recipe for apache cookbook looks like this

#

# Cookbook Name:: apache

# Recipe:: default

#

# Copyright 2014, WEBSERVER.com

#

# All rights reserved - Do Not Redistribute

# package "httpd"

#Allow Apache to start on boot service "httpd" do action [:enable, :start] end

#Add HTML Template into Web Root

template "/var/www/html/index.html" do

source "index.html.erb"

mode "0644"

end

In this recipe, we install, enable, and start the HTTPD (HTTP daemon) service. Next, we render a template for index.html and place it into the /var/www/html directory. The index.html.erb template in this case is a very simple HTML page:

<h1>Hello World </h1>

Next the Cookbook is uploaded to the Chef server. In our example, we add this cookbook to a role called ‘webserver’.

Now we are ready to launch EC2 instances (nodes), we can provide EC2 user data to bootstrap them by using Chef. To make this as dynamic as possible, we can use an EC2 tag to define which Chef role to apply to our node. This allows us to use the same user data script for all nodes, whichever role is intended for them. For example, a web server and a database server can use the same user data if you assign different values to the ‘role’ tag in EC2. We also need to consider how our new instance will authenticate with the Chef server. We can store our private key in an encrypted Amazon S3 bucket by using Amazon S3 server side encryption, 5 and we can restrict access to this bucket by using IAM roles. The key can then be used to authenticate with the Chef server. The chef-client uses a validator.pem file to authenticate to the Chef server when registering new nodes. We also need to know which Chef server to pull our configuration from. We can store a pre-populated client.rb file in Amazon S3 and copy this within our user data script. You might want to dynamically populate this client.rb file depending on environment, but for our example we assume that we have only one Chef server and that a pre-populated client.rb file is sufficient. You could also include these two files into your custom AMI build.

The user data would look like this:

#!/bin/bash

cd /etc/chef

#Copy Chef Server Private Key from S3 Bucket

aws s3 cp s3://s3-bucket/orgname-validator.pem orgnamevalidator.pem

#Copy Chef Client Configuration File from S3 Bucket

aws s3 cp s3://s3-bucket/client.rb client.rb

#Change permissions on Chef Server private key.

chmod 400 /etc/chef/orgname-validator.pem

#Get EC2 Instance ID from the Meta-Data Service

INSTANCE\_ID=`curl -s [http://169.254.169.254/latest/metadata/instance-id`](http://169.254.169.254/latest/metadata/instance-id%60)

#Get Tag with Key of ‘role’ for this EC2 instance

ROLE\_TAG=$(aws ec2 describe-tags --filters "Name=resourceid,Values=$INSTANCE\_ID" "Name=key,Values=role" --output text)

#Get value of Tag with Key of ‘role’ as string

ROLE\_TAG\_VALUE=$(echo $ROLE\_TAG | awk 'NF>1{print $NF}')

#Create first\_boot.json file dynamically adding the tag value as the chef role in the run-list

echo "{\"run\_list\":[\"role[$ROLE\_TAG\_VALUE]\"]}" > first\_boot.json

#execute the chef-client using first\_boot.json config

chef-client -j first\_boot.json

#daemonize the chef-client to run every 5 minutes

chef-client -d -i 300 -s 30

As shown in the previous user data example, we copy our client configuration files from a private S3 bucket. We then use the EC2 metadata service to get some information about the instance (in this example, we are using Instance ID).

Next, we query the Amazon EC2 API for any tags with the key of ‘role,’ and dynamically configure a Chef run-list with a Chef role of this value.

Finally, we execute the first chef-client run by providing the first\_boot.json options, which include our new run-list.

We then execute chef-client once more; however, this time we execute it in a daemonized setup to pull configuration every 5 minutes.

We now have some re-usable EC2 user data that we can apply to any new EC2 instances.

As long as a ‘role’ tag is provided with a value that matches a role on the target Chef server, the instance will be configured using the corresponding Chef cookbooks.

Below worflow shows how from a instance launch to a fully configured instance that is ready to serve the traffic.

Diagram

Description automatically generated