

Costa Rica 13 / Feb / 2019

Work Report

Rafael E Rumbobs S

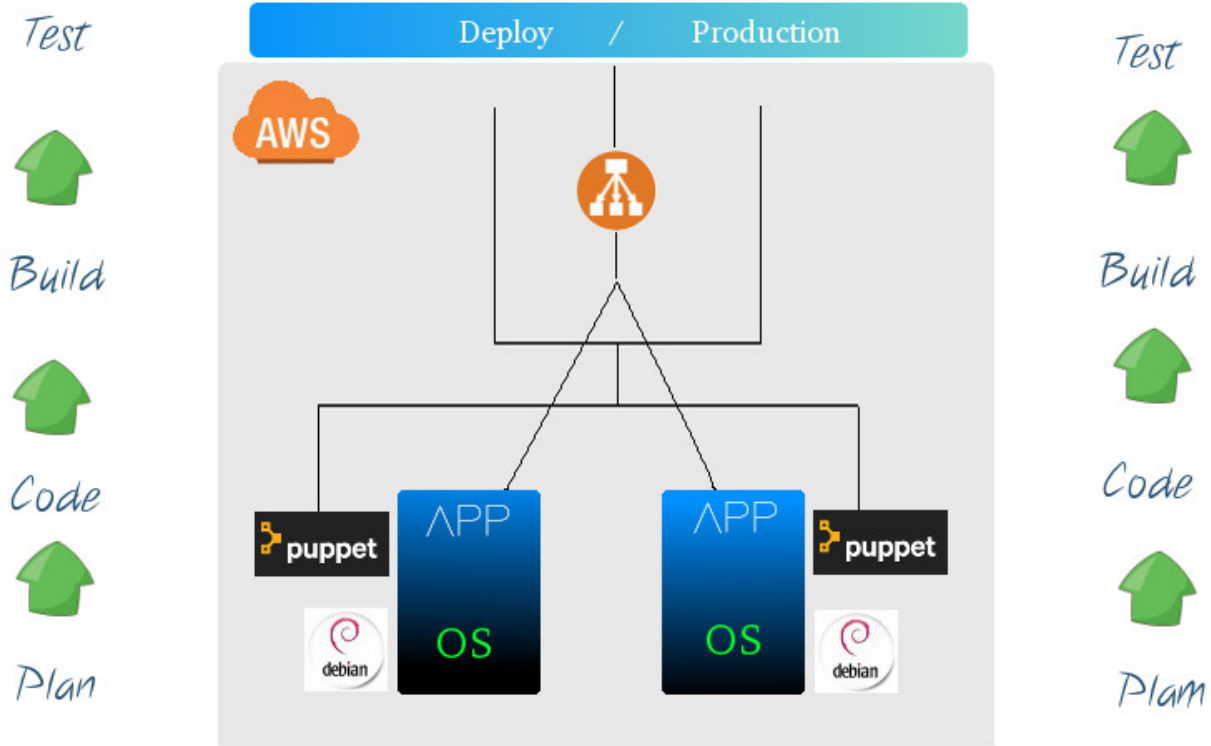
Acceptance Criteria:

- Architecture diagram demonstrating planned solution.
- Timeoff-management fork on local repository (Github, CodeCommit, BitBucket, GitLab, etc)
- Required infrastructure running on cloud provider of preference, provisioned using some sort of infrastructure as code solution.
- Application must be deployed using a fully automated continuous integration solution, triggered by a change in source control.
- Application must be secured from external access and the application should be serving via standard HTTP and HTTPS protocols.
- The application should be highly available and load balanced.

Solution Diagram

DEV OPS

Github



Components

- ec2 instances
- Debian GNU/Linux
- Aws Classic Balancer
- Puppet
- Apache2
- Cron Jobs
- Git - github
- Nodejs
- Bash

- Timeoff-management fork on local repository (Github, CodeCommit, BitBucket, GitLab, etc)

Solution: please review a copy the app at the following link:

<https://github.com/rufus88/deploy>

- Required infrastructure running on cloud provider of preference, provisioned using some sort of infrastructure as code solution.

Solution: cloud provider AWS

Infrastructure as code: Puppet - please review

<https://github.com/rufus88/production/blob/master/manifest/install.pp>

- Application must be deployed using a fully automated continuous integration solution, triggered by a change in source control.

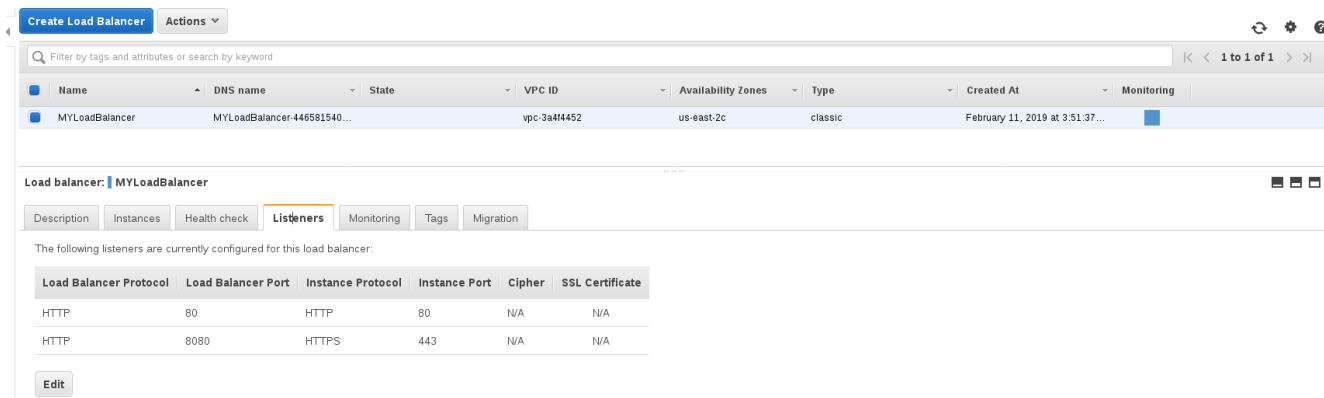
Solution: Using cron jobs and bash logic

challenge: improve code to catch all possible case scenarios

<https://github.com/rufus88/production/blob/master/manifest/env.pp>

- Application must be secured from external access and the application should be serving via standard HTTP and HTTPs protocols.

Solution: applications is now currently running on port 80 and 8080 as an attempt to configure SSL.



The screenshot shows the AWS Management Console interface for a Load Balancer named MYLoadBalancer. The console displays a table of listeners with columns for Load Balancer Protocol, Load Balancer Port, Instance Protocol, Instance Port, Cipher, and SSL Certificate. Two listeners are configured: one for HTTP on port 80 and one for HTTP on port 8080.

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Cipher	SSL Certificate
HTTP	80	HTTP	80	N/A	N/A
HTTP	8080	HTTPS	443	N/A	N/A

```
<VirtualHost *:443>
SSLEngine On
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
SSLCertificateFile /etc/ssl/localcerts/apache.pem
SSLCertificateKeyFile /etc/ssl/localcerts/apache.key
ProxyPreserveHost On
ProxyRequests Off
ProxyPass / http://localhost:3000/
ProxyPassReverse / http://localhost:3000/
</VirtualHost>
```

Nmap scan report for ec2-3-16-1-186.us-east-2.compute.amazonaws.com
(3.16.1.186)

Host is up (0.11s latency).

Not shown: 996 filtered ports

PORT	STATE	SERVICE
------	-------	---------

22/tcp	open	ssh
--------	------	-----

80/tcp	open	http
--------	------	------

443/tcp	open	https
---------	------	-------

3000/tcp	open	ppp
----------	------	-----

Nmap scan report for ec2-18-222-240-252.us-east-2.compute.amazonaws.com (18.222.240.252)

Host is up (0.10s latency).

Not shown: 996 filtered ports

PORT	STATE	SERVICE
------	-------	---------

22/tcp	open	ssh
--------	------	-----

80/tcp	open	http
--------	------	------

443/tcp	open	https
---------	------	-------

3000/tcp	open	ppp
----------	------	-----

Challenge: without an FQDN for the project is hard to setup a valid SSL service as note want to state that try to use Lets Encrypt service but the DNS name of the AWS classic load balancer was not accepted.

- The application should be highly available and load balanced.

Solution: Using AWS classic load Balancer on two EC2 instances please see the following screenshot.

The screenshot displays the AWS Management Console interface for a Classic Load Balancer. At the top, there's a navigation bar with 'Create Load Balancer' and 'Actions' buttons. Below this is a search bar and a table listing the load balancer. The table has columns: Name, DNS name, State, VPC ID, Availability Zones, Type, Created At, and Monitoring. The entry for 'MYLoadBalancer' is shown with its DNS name, VPC ID 'vpc-3a4f4452', and availability zones 'us-east-2c'. Below the table, the 'Load balancer: MYLoadBalancer' section is active, showing tabs for Description, Instances, Health check, Listeners, Monitoring, Tags, and Migration. The 'Instances' tab is selected, showing 'Connection Draining: Enabled, 300 seconds (Edit)'. Below this is an 'Edit Instances' button and a table of instances. The instances table has columns: Instance ID, Name, Availability Zone, Status, and Actions. Two instances are listed: 'i-0444dd863538e6527' (serverA) and 'i-06c3003b651b7c9ab' (serverB), both in 'us-east-2c' and 'InService' status. Below the instances table is an 'Edit Availability Zones' button and a table of availability zones. The availability zones table has columns: Availability Zone, Subnet ID, Subnet CIDR, Instance Count, Healthy?, and Actions. One zone is listed: 'us-east-2c' with Subnet ID 'subnetfd75a4b1', Subnet CIDR '172.31.32.0/20', and an instance count of 2.

Name	DNS name	State	VPC ID	Availability Zones	Type	Created At	Monitoring
MYLoadBalancer	MYLoadBalancer-446581540...		vpc-3a4f4452	us-east-2c	classic	February 11, 2019 at 3:51:37...	

Load balancer: MYLoadBalancer

Instances

Connection Draining: Enabled, 300 seconds (Edit)

Edit Instances

Instance ID	Name	Availability Zone	Status	Actions
i-0444dd863538e6527	serverA	us-east-2c	InService ⓘ	Remove from Load Balancer
i-06c3003b651b7c9ab	serverB	us-east-2c	InService ⓘ	Remove from Load Balancer

Edit Availability Zones

Availability Zone	Subnet ID	Subnet CIDR	Instance Count	Healthy?	Actions
us-east-2c	subnetfd75a4b1	172.31.32.0/20	2	Yes	-

Challenge: application is using SQLite3 as data backend - the backup of this element should be also automated in case of emergency.