# Lab Three: Operational Report of Functionality

## Introduction:

The task of the Operations Team is to write the scripts and build the architectures to support the application and the changes to the application over time. The Operational Report of Functionality is the Operations Team’s way to say and show that a release candidate script will provide the functionality that is required.

## Deliverables:

The deliverables for this lab include the deployment of a load balanced architecture with the “Explore California” web site served as content. You should have developed a Visio diagram as part of your Project Plan.

Answers don’t need to be long, instead concentrate on being concise and effective.

# Operations Report of Functionality:

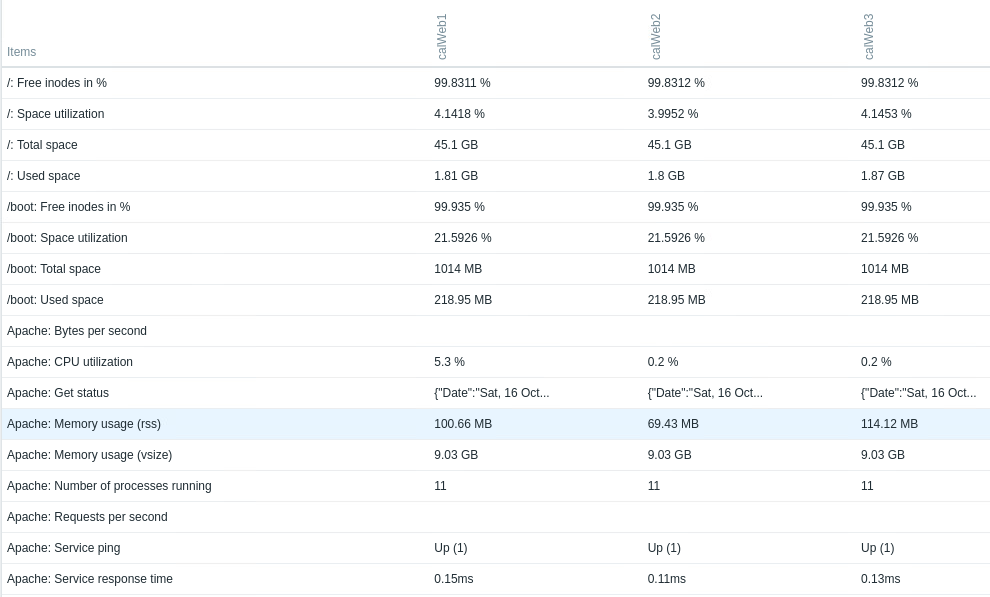
|  |  |
| --- | --- |
| Group Number: | 4 |
| Group Members: | Nate Bachelder (Formerly Williams), Steffen Barr, Eli Hopkins, Xavier Rivera |
|  |  |
|  |  |
| Version/Lab Number: | Lab 3 |
| Date: | 10/15/21 |

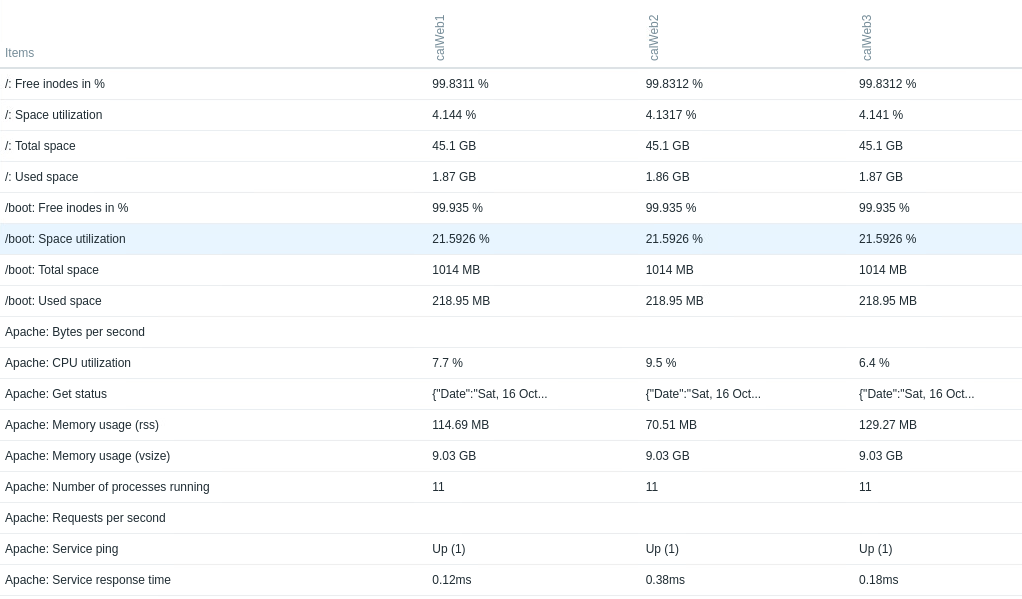
Fill in the table below with a short description that answers the question.

## Deliverables:

|  |  |
| --- | --- |
| What functionality is required of your architecture to satisfy the requirements of the lab? | Our architecture is required to include a load balancer which will direct requests between 3 web servers in a round robin fashion. |
| Show that your architecture satisfies those requirements. Screenshots can be very effective for this. | See below rest of document.  See sec report for tests proving that architecture is functional. |
| What additional services/resources did you have to add to provide the additional functionality? | To support this architecture we had to scale out 2 additional explore california web servers. Alongside this, we deployed a new host to act as the load balancer. Nginx needed to be set up on the loadbalancer host. Alongside this, Zabbix agents needed to be installed on the 2 new web servers. |
| Success/Definition of successful completion | The Ops team’s work is a success if the architecture as a whole functions and is able to handle an expected workload of 300 users. Ideally, the architecture should be able to handle an overload workload significantly larger than this. |
| Enough of a description of the system or script that a new user understands its purpose | In this system, a loadbalancer is set up to be the entrypoint for a grouping of 3 web servers. The loadbalancer receives http requests and forwards each request to a web server, choosing the next server each time and evenly distributing the workload. |
| Explanation of complicated or non-intuitive portions of code or process | All portions of the Ops team code for this lab is very intuitive and readable. Comments are written throughout the scripts. |
| Basic usage and operation | To run the playbook simply run ansible-playbook loadbalancer\_setup.yml |
| Which user to interact with the system or script as | The playbook can be run as the local user but will run as root on the target machine. |
| Where the script or system runs and what it touches or needs access to | The installation script needs to run as root on the target system in order to install packages and manage systemd. After the initial installation, Nginx will run as the nginx user. |
| Known issues | N/A |
| Future plans or features | Failsafe Secondary Loadbalancer |

**Expected Load Monitoring (300 Concurrent Users)**

**Overload Monitoring (500 Concurrent Users)**

****