**Solution Design and Implementation**

1. **Proposed Solution**

***Purpose:***

The purpose of this document is to identify and provide a business solution for Satellite Ads Inc.’s (Sad\_INC) projected growth. Sad\_INC is launching a 2nd wave of satellites and is looking to implement a new cloud-based automation system. The core requirements are that the system handle expected workloads and peak workloads throughout the year.

***Goals and Objectives:***

The new cloud-based system will have the ability to scale to critical events based on projections and data, with the ability to monitor the capability of systems while using elastic services. The ability to scale will allow Sad\_INC to adequately handle peak business hours with minimal manual labor involved, saving money and time.

***Scope:***

The new cloud-based system will consist of one Base Station made up of one Satellite Terminal, one Web Server, one Database Server, and one Master Time Server. The system will auto scale using Chef, to create a cluster consisting of a Satellite Terminal, Web Server, Database Server, and a Time server. Each cluster will handle 300 satellites, with new clusters being built for every 270 satellites put into use.

1. **Visual Representation**

**Graphical user interface

Description automatically generated**

**Automation Script**

1. **Automation Script**

# Name: Alexander Fluhr

# Student ID: 001354345

---

driver:

  # Driver specifies the software that manages the machine, per instructions I am using Vagrant.

  name: vagrant

  network:

    # creates a second interface used to allow connectivity between VMs and to other devices on the network

    - ["public\_network"]

  customize:

    # Sets the amount of Memory for each vm to 512mb for testing purposes.

    memory: 512

provisioner:

  # Name specifices how chef will be simulated during

  name: chef\_zero

  product\_name: chef

  product\_version: 14.12.9

verifier:

  name: inspec

# Specifies the the OS's that Chef will use when creating the instances

platforms:

  - name: centos-7

suites:

# Each section creates a server cluster, name specifies Name in Chef, and vm\_hostname specifies name of created VM, Other than name, settings are same for every suite

  - name: Satellite-Terminal-Server

    driver:

      vm\_hostname: Satellite-Terminal-Server

    # run list specifies which recipes to run, for this test we are defaulting to chef's cookbook for all servers

    run\_list:

      - recipe[learn\_chef\_httpd::default]

  - name: Master-Time-Server

    driver:

      vm\_hostname: Master-Time-Server

    run\_list:

      - recipe[learn\_chef\_httpd::default]

  - name: Web-Server

    driver:

      vm\_hostname: Web-Server

    run\_list:

      - recipe[learn\_chef\_httpd::default]

  - name: Database-Server

    driver:

      vm\_hostname: Database-Server

    run\_list:

      - recipe[learn\_chef\_httpd::default]

  - name: Scaled-Satellite-Terminal-Server

    driver:

      vm\_hostname: Scaled-Satellite-Terminal-Server

    run\_list:

      - recipe[learn\_chef\_httpd::default]

  - name: Scaled-Time-Server

    driver:

      vm\_hostname: Scaled-Time-Server

    run\_list:

      - recipe[learn\_chef\_httpd::default]

  - name: Scaled-Web-Server

    driver:

      vm\_hostname: Scaled-Web-Server

    run\_list:

      - recipe[learn\_chef\_httpd::default]

  - name: Scaled-Database-Server

    driver:

      vm\_hostname: Scaled-Database-Server

    run\_list:

      - recipe[learn\_chef\_httpd::default]

1. **Screenshot showing that the automation script executes without errors:**

**![Graphical user interface, text

Description automatically generated]()**

**Diagnostic Report**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Description** | **Optimal Range** | **Data and Results** | **Automation Script Used to Extract Data (text only)** | **Screenshot of Result of Script** |
| Time to scale from 1 cluster to 200 clusters  (60,000 advertisements expected at peak global usage) based on 300 satellites per cluster (subject to change based on load testing) | 15–30 minutes for each cluster | **Data:**  7 Minutes to launch each cluster of 4 servers.  200 Clusters = 7m23\*200 = 1446 Minutes in total to launch 200 clusters. | Kitchen Create,  Kitchen Converge | ­ |
| Time to register a cluster and then quench connections to the load balancer, taking the cluster off-line (start-up, operation, shutdown) | 1 minute per connection quench, start of cluster launch, and part of time to scale cluster, can be tracked separately as a quench | **Data:**  0m45s to shut down a cluster | Kitchen Destroy |  |
| Peak load averages per system at 200, and 300, satellites per cluster | 60% of CPU triggers new cluster launch; if reaching core load at 200 satellites, launch new cluster on 60% CPU loads | **Load Average:** 0.01, 0.03, 0.06 | kitchen exec Satellite-Terminal-Server -c 'top' |  |
| Write times to the diagnostic data drive | <30 milliseconds | **Disk Performance:** 1073741824 bytes (1.1 GB) copied, 5.74997 s, 187 MB/s | kitchen exec Satellite-Terminal-Server -c 'dd if=/dev/zero of=testWriteSpeed.txt bs=1G count=1' |  |
| Pull time from the game instances (1 Satellite Terminal Server, 1 Web Server, 1 Database, and 1 time server) and initialization time | Part of cluster launch 15–30 minutes | **Time to Start Cluster:**  4m25.88s | Kitchen create,  kitchen converge |  |
| \*Average messaging service (queue) time | <1 minute in queue | N/A | N/A | N/A |
| Average latency for the Time server | <30 milliseconds | **Data:**  rtt min/avg/max/mdev = 4.192/4.446/5.187/0.382 ms | kitchen exec Master-Time-Server -c 'ping -w 5 google.com' |  |
| Average latency of each cluster | <30 milliseconds | **Data:**  rtt min/avg/max/mdev = 0.009/0.095/0.403/0.154 ms | kitchen exec Satellite-Terminal-Server -c 'ping -w 5 localhost' |  |
| Network data in and out for each cluster | <1 second | **Data:**  rtt min/avg/max/mdev = 4.194/4.389/4.603**/0.145 m** | kitchen exec Satellite-Terminal-Server -c 'ping -w 5 -c 4 google.com' |  |
| Overall CPU utilization of the environment for each cluster | Not >60% | **Load Average:**  0.00, 0.01, 0.05 | kitchen exec Satellite-Terminal-Server -c 'top' |  |
| \*Diagnostic data able to be written by the automation to the correct cloud bucket storage space | Show read/write times <1 second | N/A | N/A | N/A |
| Scaled Satellite Cluster latency | <30 milliseconds | **Time to Reach Cluster:**  rtt min/avg/max/mdev = 4.192/4.446/5.187/0.382 ms | kitchen exec Master-Time-Server -c 'ping -w 5 google.com' |  |
| Scaled Satellite Cluster latency between gateway/scaled clusters and core | <30 milliseconds | **Latency:**  rtt min/avg/max/mdev = 0.009/0.020/0.043/0.011 ms | kitchen exec Database-Server -c 'ping -w 5 localhost' |  |
| Scaled Satellite Cluster latency between scaled clusters and environment | <30 milliseconds | **Latency:**  rtt min/avg/max/mdev = 0.009/0.013/0.015/0.004 ms | kitchen exec Scaled-Time-Server -c 'ping -w 5 localhost' |  |
| Pull time from the scaled clusters and initialization time | 15–30 minutes for each cluster | **Time to Start Cluster:**  7m23s  +  4m25s  =  11.48m | Kitchen Create,  Kitchen Converge, |  |

1. Web Sources

Learn.chef.io ​- Getting Started with Test Kitchen - Used as guidance when configuring test kitchen.

<https://learn.chef.io/modules/local-development/ubuntu#/infrastructure-automation>

Chef GitHub Repo

<https://github.com/learn-chef>

Diagram.net

<https://app.diagrams.net/>