

Satellite Ads, Inc.

Auto-Scaling Satellite Control System

D085:CKM2

Ryan Peterson

Student ID: #000980429

April 23, 2022

Version 1.0



WESTERN GOVERNORS UNIVERSITY®

A. INTRODUCTION OF SOLUTION

A.1. PURPOSE

With the release of their newest generation of space marketing satellites, Satellites Ads Inc. anticipates a spike in advertising demand, which this document aims to address by providing a solution for scaling with increased advertising demand.

A.2. GOALS AND OBJECTIVES

Satellites Ads Inc. must achieve two goals and objectives. The first is an automated scaling solution for deploying satellite clusters as demand dictates. Four servers that can manage advertisements for up to 300 satellites make up these clusters. Satellite Ads may use up to 60 clusters to meet advertisement capacity during peak demand. The second requirement is monitoring the systems and using elastic services to scale the number of clusters required based on need.

A.3. SCOPE

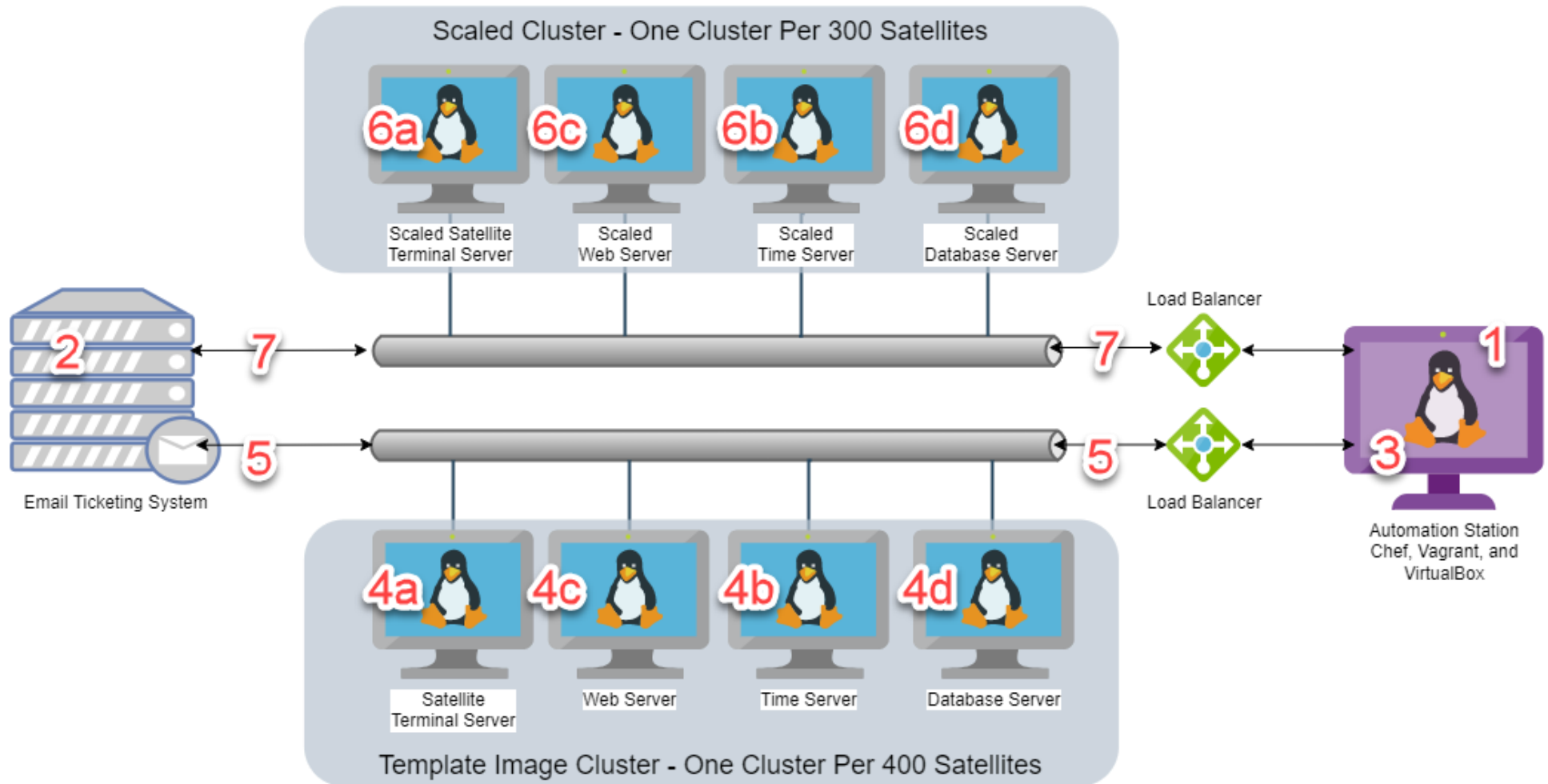
The proposed AWS Auto-Scaling Groups will monitor cluster metrics to determine when to scale up or scale down. First, the actual demand for core systems plus 10% and 10% core systems should always be available. Second, when the cluster reaches 300 satellites, a new scaled cluster is created to meet the demand. Furthermore, at least one scaled cluster should always be available. Third, as satellite demand declines, automation will decommission the clusters. Finally, whenever clusters are scaled up or down or errors occur, the system should email helpdesk@satelliteads.com.

A.4. DESCRIPTION OF FUNCTIONALITY

Scaling up this method will necessitate the usage of a few different tools. Initially, the Automation Station will execute The Chef cookbooks. After the initial cluster is live, the EC2 instances will be enabled as members of the AWS Auto-Scaling Group and CloudWatch rules. For tracking and informational purposes, alerts will be provided to the support desk for any problems and scale events.



Satellite Ads Inc.



1. The Automation Station is deployed with Chef and Vagrant installed to deploy the initial cluster and scale automatically as demand increases.
2. The Email Ticketing system is created to properly route messages for scaling actions and errors.
3. The kitchen.yml file on the Automation Station has been configured to deploy all servers in the clusters and will be implemented with the kitchen create and kitchen converge commands.
4. For the initial cluster, these servers are created:
 - a. Satellite Terminal Server
 - b. Master Time Server
 - c. Web Server
 - d. Database Server
5. A message is passed to the Automation Station and the email ticketing system with a status update.
6. The Automation Station deploys satellite control cluster and these servers are created:
 - a. Scaled Satellite Terminal Server
 - b. Scaled Master Time Server
 - c. Scaled Web Server
 - d. Scaled Database Server
7. After 300 Satellites are deployed, a message is sent to the Automation Station to scale up another cluster as in step 6.

C. AUTOMATION SCRIPT

```
# Ryan Peterson Student ID: 000980429
---
# Developed Locally using Vagrant driver
driver:
  name: vagrant

# Using Chef Zero as an in-memory Chef server for testing purposes
provisioner:
  name: chef_zero
  product_name: chef
  product_version: 14.12.9

# Using Chef InSpec as the automated testing framework
verifier:
  name: inspec

# Operating System that will be used by the virtual machines
platforms:
  - name: centos-7

# For testing purposes, all servers will be built with the learn_chef_httpd
# cookbook sourced from GitHub at: https://github.com/chef/learn_chef_httpd
suites:

# Satellite Terminal Server
- name: satellite_terminal_server
  driver:
    vm_hostname: terminal.satelliteads.local
  run_list:
    - recipe[learn_chef_httpd::default]
  attributes:
```



```

# Master Time Server
- name: master_time_server
  driver:
    vm_hostname: time.satelliteads.local
  run_list:
    - recipe[learn_chef_httpd::default]
  attributes:

# Web Server
- name: web_server
  driver:
    vm_hostname: web.satelliteads.local
  run_list:
    - recipe[learn_chef_httpd::default]
  attributes:

# Database Server
- name: database_server
  driver:
    vm_hostname: database.satelliteads.local
  run_list:
    - recipe[learn_chef_httpd::default]
  attributes:

# Scaled Satellite Terminal Server
- name: scaled_satellite_terminal_server
  driver:
    vm_hostname: scaled-terminal.satelliteads.local
  run_list:
    - recipe[learn_chef_httpd::default]
  attributes:

# Scaled Web Server
- name: scaled_web_server
  driver:
    vm_hostname: scaled-web.satelliteads.local
  run_list:
    - recipe[learn_chef_httpd::default]
  attributes:

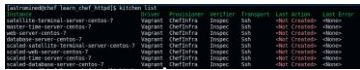


# Scaled Time Server
- name: scaled_time_server
  driver:
    vm_hostname: scaled-time.satelliteads.local
  run_list:
    - recipe[learn_chef_httpd::default]
  attributes:

# Scaled Database Server
- name: scaled_database_server
  driver:
    vm_hostname: scaled-database.satelliteads.local
  run_list:
    - recipe[learn_chef_httpd::default]
  attributes:


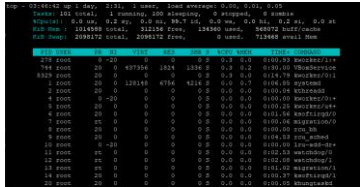
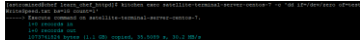
```



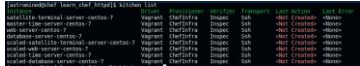
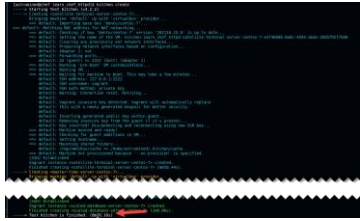

D. DIAGNOSIS REPORT

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	One cluster = 8:28 200 clusters = 28:13:20	kitchen list kitchen create kitchen converge	kitchen-list.png  kitchen-create-full.png kitchen-create-reduced.png  kitchen-converge-full.png kitchen-converge-reduced.png 



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	One cluster = 00:00:44	kitchen destroy	kitchen-destroy.png 
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: 0.00, 0.01, 0.05	kitchen exec satellite-terminal-server-centos-7 -c 'top'	kitchen-exec-top.png 
Write times to the diagnostic data drive	<30 milliseconds	1073741824 bytes (1.1 GB) copied, 35.5089 s, 30.2 MB/s	kitchen exec satellite-terminal-server-centos-7 -c 'dd if=/dev/zero of=testWriteSpeed.txt bs=1G count=1'	kitchen-exec-dd.png 

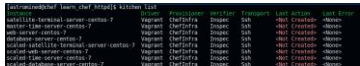




<p>Pull time from the game instances (1 Satellite Terminal Server, 1 Web Server, 1 Database, and 1 time server) and initialization time</p>	<p>Part of cluster launch 15–30 minutes</p>	<p>One cluster = 8:28</p>	<p>kitchen list kitchen create kitchen converge</p>	<p>kitchen-list.png</p>  <p>kitchen-create-full.png</p> <p>kitchen-create-reduced.png</p>  <p>kitchen-converge-full.png</p> <p>kitchen-converge-reduced.png</p> 
<p>*Average messaging service (queue) time</p>	<p><1 minute in queue</p>	<p>N/A</p>	<p>N/A</p>	<p>N/A</p>



Average latency for the Time server	<30 milliseconds	rtt min/avg/max/mdev = 16.673/17.933/19.118/0.8 95 ms	kitchen exec master-time-server-centos-7 -c 'ping -c 5 google.com'	kitchen-exec-time.png
Average latency of each cluster	<30 milliseconds	rtt min/avg/max/mdev = 0.032/0.049/0.094/0.025 ms	kitchen exec master-time-server-centos-7 -c 'ping -c 5 localhost'	kitchen-exec-ping.png
Network data in and out for each cluster	<1 second	rtt min/avg/max/mdev = 16.673/17.933/19.118/0.8 95 ms	kitchen exec master-time-server-centos-7 -c 'ping -c 5 google.com'	kitchen-exec-time.png
Overall CPU utilization of the environment for each cluster	Not >60%	load average: 0.00, 0.01, 0.05	kitchen exec satellite-terminal-server-centos-7 -c 'top'	kitchen-exec-top.png
*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	rtt min/avg/max/mdev = 14.553/16.411/18.780/1.4 78 ms	kitchen exec scaled-time-server-centos-7 -c 'ping -c 5 google.com'	kitchen-exec-scaled.png
Scaled Satellite Cluster latency between gateway/scaled clusters and core	<30 milliseconds	rtt min/avg/max/mdev = 14.553/16.411/18.780/1.4 78 ms	kitchen exec scaled-time-server-centos-7 -c 'ping -c 5 google.com'	kitchen-exec-scaled.png
Scaled Satellite Cluster latency between scaled clusters and environment	<30 milliseconds	rtt min/avg/max/mdev = 14.553/16.411/18.780/1.4 78 ms	kitchen exec scaled-time-server-centos-7 -c 'ping -c 5 google.com'	kitchen-exec-scaled.png



<p>Pull time from the scaled clusters and initialization time</p>	<p>15–30 minutes for each cluster</p>	<p>One cluster = 8:28</p>	<p>kitchen list kitchen create kitchen converge</p>	<p>kitchen-list.png</p>  <p>kitchen-create-full.png</p> <p>kitchen-create-reduced.png</p>  <p>kitchen-converge-full.png</p> <p>kitchen-converge-reduced.png</p> 
---	---------------------------------------	---------------------------	---	--



E. WEB SOURCES

chef. (n.d.). *GitHub - chef/learn_chef_httpd: Cookbook for the Learn Chef tutorials*. GitHub; github.com. Retrieved April 22, 2022, from https://github.com/chef/learn_chef_httpd

top(2) - Linux manual page. (n.d.). *Top(1) - Linux Manual Page*; man7.org. Retrieved April 22, 2022, from <https://man7.org/linux/man-pages/man1/top.1.html>

dd(3) - Linux manual page. (n.d.). *Dd(1) - Linux Manual Page*; man7.org. Retrieved April 22, 2022, from <https://man7.org/linux/man-pages/man1/dd.1.html>

ping(8) - Linux manual page. (n.d.). *Ping(8) - Linux Manual Page*; man7.org. Retrieved April 22, 2022, from <https://man7.org/linux/man-pages/man8/ping.8.html>

kitchen (executable). (2022, February 24). *Kitchen (Executable)*; docs.chef.io. https://docs.chef.io/workstation/ctl_kitchen/#kitchen-exec

F. SOURCES

No other sources were used other than web sources above

