VERSION 1.0

SEPT 17 2019

AMDOCS OS PATCHING

SIU CAPSTONE PROJECT SUMMARY

PRESENTED BY: ALAN J ENNEN
AMDOCS
CHAMPAIGN ILLINOIS

AMDOCS OS PATCHING

INTRODUCTION

A Key challenge we are facing in our datacenter today is the amount of time and effort it takes to apply OS patches to all our Linux Production and Non-Production servers. Over the years this has become an ever-increasing burden due in part to the increasing rate in which security patches are becoming available.

Our data center consists of:

We have ~5000 Linux server hosted in 2 DC's. The production is hosted in Champaign and Disaster Recovery/Non-Production is hosted in Moses Lake. These servers are running different versions of Linux RedHat (RHEL). Most of the infrastructure is virtualized on VMware hypervisor.

The current patching process uses primarily Redhat Satellite and Ansible tools to apply patches to servers.

1. Patching is done using RedHat Satellite.
2. Satellite sync with Redhat repository
3. Satellite Publish Content Views (a collection of RedHat repositories)
4. Satellite Promote Content Views to Environment Lifecycle
5. Environment Lifecycles Content sync with Capsules
6. Unix Admin patch RHEL clients by running yum update
7. Servers are taken out of production before starting the patching
8. After patching is completed all teams perform the sanity and servers go back to production

OBJECTIVE

Our goal is to improve our ability to quickly deploy OS patches with little to no human interaction through intelligent automation of the deployment process.

Key Challenges

This project has 3 major areas to consider each with its unique challenges.

1) Access to topology information of all the servers running that include not just server specifications but information on the products running on them.

2) An intelligent orchestrator that can take the topology information and determine the proper sequence and steps to execute on a server for the update

3) A flexible execution layer that will be able to take and execute instructions from an orchestrator.

SUMMARY

TOPOLOGY:

Access to data center topology information is critical for the proper execution for patching a server.

A topology information repository will be needed to provide information not just for server specifications such as OS version and patch level but also to provide information on the application(s) running on the server. Information such as the proper application shutdown and startup commands and any runtime dependencies such as needing to take a service out of a load balancer before a clean shutdown.

For any given server you may have 1 to many applications running on it and each application requires a clean shutdown, additionally an application may be accessed through a load balancer and if so would need to be removed from the load balancer to properly stop traffic to it before shutting down.

ORCHESTRATION:

The patching process depending on application being hosted has certain rules that need to be followed such as Zero down time. In this case the orchestrator would determine that a rolling bounce of logical groups of servers in such a way as to prevent outage would be needed. The orchestrator would analyze the topology and determine the proper sequence of events that needs to occur to make this happen.

EXECUTION:

The execution phase simply takes the direction from the orchestrator to execute the necessary steps to cleanly bring down the applications and execute the patching on all servers. Given the number of servers in the datacenter parallel processing will be required to patch the servers in any reasonable time frame.