Odin Hugo Daniel Orozco Camacho

Universidad Oberta de Catalunya
Proyecto Final de Master

2 de enero de 2017

Resumen

La administración de servidores es una actividad creada en el momento en el que comenzamos a extender y agregar complejidad a estos, los cuales exigen actividades y revisiones constantes para un funcionamiento optimo, y para esto existen herramientas que facilitan estas tareas al ingeniero o informático.

Palabras Clave -- xencli, xen, administración de servidores

Glosario

Introducción

XenCli es un software el cual esta orientado a la administración de maquinas virtuales dentro de un ambiente controlado como lo es el propio Dom0 o Xen[1].

En este documento se explica y detalla las características de xenCli, proyecto desarrollado para la empresa Oracle[7] en su área de Cloud Operation Services con base en la ciudad de Guadalajara, Jalisco Mexico.

Antecedentes

Actualmente la administración de sistemas operativos es de forma manual y por medio de una interface CLI, esto para ser fácilmente manipulable dentro de algún script y poder ser escalable dependiendo de la tarea; pero en ocaciones, simplemente se necesita la intervención de un SA para tareas previas o tareas en las cuales simplemente no se pueden automatizar.

En cierto modo esto no se percibe correctamente agregar a la lista una herramienta más, ya que una interface como “xenCLI” podría llegar a complicar un simple script, pero al momento en el cual una tarea tan simple como ver el uptime de 17 maquinas virtuales resulta algo complejo, ya que estaríamos buscando y accediendo 17 veces un root/pwd para poder escribir uptime y poder generar una conclusión rápida.

Siguiendo el ejemplo anterior, si contamos con un contrato con un SLA[11] critico, la verificación de la integridad de 17 maquinas virtuales puede que de “una por una” no sea la mejor opción.

Motivación

Motivación es un término bastante amplio, básicamente son las condiciones que proporcionan al medio la energía para implementar acciones tendientes a obtener algún fin; y ese fin está relacionado directamente con las necesidades. En otras palabras el desarrollo de una herramienta de administración masiva, es una necesidad en las tareas y acciones realizadas diariamente.

Esta necesidad latente puede ser satisfecha con el desarrollo planteado durante todo este tiempo del PFM y que en este documento lo llamamos xenCli.

Propósito

Este documento tiene como propósito el dar a conocer a la interface xenCli funcionamiento, planeación del desarrollo y el histórico de este software el cual esta dirigido al equipo de administradores en sistemas para facilitar de manera considerable la administración de la infraestructura basada en xenproject[1].

Objetivos

El objetivo es contar con una herramienta interna eficiente, en el cual el tiempo de reacción se el mínimo, donde no se gasten recursos desmesurados para tareas cotidianas y sencillas.

Objetivos Específicos

- Diseño e implementación de una herramienta para la administración de maquinas virtuales basadas en Xen.
    - Basado en C.
    - Daemonizado para su uso en sistemas Linux.
- Diseño e implementación de un repositorio para el almacenamiento del desarrollo de software.
    - Sistema de versión, (Major, Menor, RC) Ejemplo. Xencli 1.1.beta
    - Aplicación de CVS para control de versiones
- Diseño e implementación de un repositorio para el almacenamiento de los entregables
    - Manual de usuario
    - Errata

Funciones generales

- Administración segura dentro del Dom0.
- Manejo y administración comenzando desde un solo DomU.
- Sistema de log para errores y/o salidas de estándar

Actores

- Miguel Martín Mateo (Profesor UOC)
- Horacio Olivares Montoya (Manager Oracle Cloud Services)
- Odin Orozco Camacho (Estudiante UOC, Empleado de Oracle y desarrollador del Proyecto Xencli)

Requerimientos

Para el funcionamiento de Xencli se supone lo siguiente:

- Se esta trabajando bajo un derivado de RHEL[4].
- Xen Kernel deberá estar instalado correctamente.
- La administración de usuarios y el manejo del xencli en el Dom0 hacia los DomU es un proceso transparente, esto es no es necesario un método de autenticación, ya que se esta trabajando en un ambiente con mecanismos de seguridad ajenos a este desarrollo.
- La interfaz de Xencli deberá ser diseñada para ser un CLI, el cual da la flexibilidad de poder ser utilizado de forma dinámica y/o automatizada.
- Xencli sera una posible adición a la paqueteria de scritps que automatización que actualmente cuenta OVS.

Definición de requerimientos

- Software tiene que estar tanto en el Servidor como en el Cliente.
- Debe de poder tener la opción de enviar comandos de uno a muchos clientes.
- Debe de poder hacer la comunicación sin necesidad de autenticarse.
- En caso de que el servicio SSH[14] este apagado, xenCli deberá de funcionar.
- Contara con una bitácora donde se guardaran todas las operaciones realizadas.
- Contara con una presentación clara donde se especifique el servidor y los comandos que se ejecutaron junto con su resultado.

Especificación de requerimientos funcionales

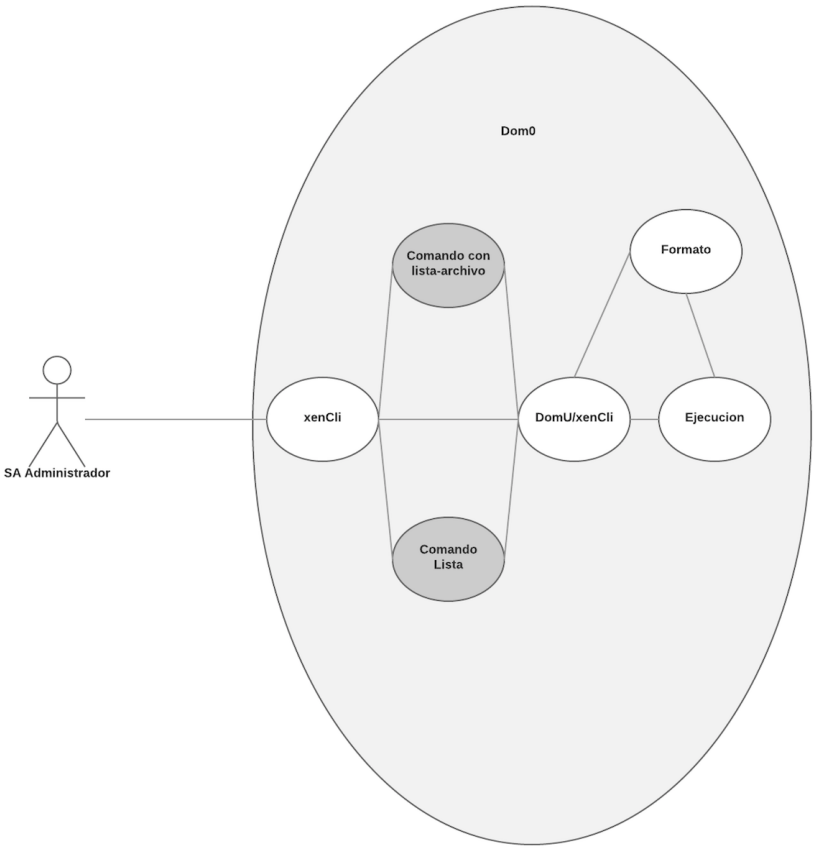
- El software estará basado en el esquema cliente-servidor y de igual forma, este estará siendo ejecutado como un demonio en el sistema operativo tanto en el Dom0 como en las DomU.
- XenCli deberá tener la capacidad de poder utilizar una lista de DomU's desde un archivo o poder introducir los nombres de las DomU a administrar desde una opción del mismo comando.
  - -f (file)
  - -h (manualmente separado por espacios en blanco)
- xenCli sera independiente del servicio de openSSH[14], pero de igual forma estará basado en protocolos de red.
- Contara con un sistema de logeo en /var/log/xencli, en donde se detalle las transacciones realizadas.
- El STOUT del comando xencli deberá regresar lo siguiente:

  ********* Nombre DomU *********

  Comando ejecutado
  Resultado del mismo
  ****
- El software sera desarrollado en leguaje de programación C.

Especificación de requerimientos NO Funcionales

- El sistema deberá de capas de detectar si una maquina virtual esta en algún tipo de error que no permita que se ejecute el comando y notificarlo.
- Este software solo podrá ser instalado y ejecutado desde un Dom0 (servidor Xen) hacia servidores DomU (maquinas virtuales).



Metodología

La metodología utilizada en este proyecto de desarrollo fue una metodología clásica, en este caso la de cascada, ya que el proyecto no contaba con etapas complejas y/o colaboraciones con otros equipos o aspectos que justificara una metodología diferente.

Riesgos

Los riesgos detectados del proyecto durante su desarrollo fueron pocos, pero el mas grande riesgo fue el cambio de lenguaje de programación.

Taxonomía de riesgos

Taxonomía de riesgos

ID	Elemento	Riesgo
RI-01	Planificación	Errores en la toma de requerimientos
RI-02	Planificación	Cambio en el lenguaje de programación
RI-03	Desarrollo	Dificultad del desarrollo e implementación del proyecto

Declaración de riesgos

RI01 Errores en la toma de requerimientos

Condición:

Errores en el entendimiento de los requerimientos, no estimar los factores principales como funcionamiento.

Consecuencia:

No disponer del tiempo necesario para poder generar las correcciones pertinentes y/o adecuadas para cumplir con el requerimiento.

Efecto:

Un entregable no adecuado en función de los requerimientos iniciales.

RI02 Cambio de lenguaje de programación

Condición:

El cliente cambie de opinión con respecto al lenguaje de programación en el que sera desarrollado.

Consecuencia:

No disponer del tiempo necesario para hacer un software robusto y/o estable.

Efecto:

Posible retraso en la entrega del proyecto.

RI03 Dificultad del desarrollo e implementación del proyecto

Condición:

Falta de experiencia en la codificación de este, falta de metodología o técnicas de programación.

Consecuencia:

Doblar tiempo y esfuerzos para logar este cometido.

Efecto:

Posible consulta con personal con experiencia en el ramo.

Estimación de la probabilidad

La estimación esta calculada con base en las siguientes tablas comparativas

Rango	Promedio	Estimación
1% - 10%	5%	Baja
11% - 25%	18%	Poco probable
26% - 55%	40%	Media
56% - 80%	68%	Alta
81% - 99%	90%	Segura

Impacto

El en proyecto se detecto 3 riesgos importantes, de los cuales se cuenta con uno critico como se demuestra a continuación

ID	Riesgo	Estimación	Probabilidad
RI-01	Errores en la toma de requerimientos	Poco probable	18%
RI-02	Cambio en el lenguaje de programación	Alta	68%
RI-03	Dificultad del desarrollo e implementación del proyecto	Media	40%

Criterio	Retraso	Valor
Insignificante	1 Hora	1
Marginal	1 Dia	2
Medio	3 Dias	3
Critico	1 Semana	4
Severo	Mas de 1 semana	5

ID	Impacto
RI-01	Marginal
RI-02	Critico
RI-03	Medio

Plan de acción

- RI-01: Revisiones periódicas al termino de cada etapa, incluyendo en esta etapas anteriores.
- RI-02: Contar con elementos necesarios para adquirir las habilidades necesarias, ya sean, libros, manuales, personal experto.
- RI-03: Buscar segundos enfoques, opiniones externas para generar lluvia de ideas y soluciones.

Coste total estimado

A continuación se muestra una tabla con la relación de costo-equipo que contemplado para este proyecto.

Servidor Sun Server X3-2 (antes Sun Fire X4170 M3)		
Procesador	Intel 8-Core	
RAM	256Gb	Maximo 512Gb
Almacenaje	2 bahias 200Gb	Maximo 8 Bahias
Ehternet	4 Gigabit	
USB	6 Bahias	

Costo Instalación	0 USD
Costo Adecuación	0 USD
Costo Cableado	0 USD
Costo Rack Unit	0 USD
Costo Desarrollo	0 USD

El coste de instalación de este proyecto es muy bajo, ya que se cuenta con todo lo necesario para instalar el equipo en un RackUnit local, este es usado para propósitos educativos y/o pruebas de software en las instalaciones Oracle MDC, cabe mencionar que se cuentan con precios fijos con respecto al servidor de pruebas, resultado de proyectos terminados y depreciados.

Costo-beneficio

El costo esta basado en la siguientes tablas tanto de salarios como de tiempo de mantenimiento y precios basados en costos publicados por Oracle[13]

xenCli		
Costo de desarrollo	$0.00 MXN	49 dias
Costo de entrenamiento	$468.75 MXN	2 horas
Mantenimiento	$45,000.00 MXN	1 Mes

ClusterSSH		
Costo de desarrollo	0 MXN	0 dias
Costo de entrenamiento	$468.75 MXN	2 horas
Mantenimiento	No medible con exactitud, ya que en caso de que este cuente con algún bug, no depende de nosotros y debemos esperar a un nuevo release	
Mantenimiento estimado	La frecuencia con que se actualiza este software es de 1 mes a 11 meses aproximadamente en su ultimo evento	

openSSH		
Costo de desarrollo	0 MXN	0 dias
Costo de entrenamiento	$468.75 MXN	2 horas
Mantenimiento	No medible con exactitud, ya que en caso de que este cuente con algún bug, no depende de nosotros y debemos esperar a un nuevo release	
Mantenimiento estimado	La frecuencia con la que este software es liberado aproximadamente 5 meses en su ultimo evento	

Con base a tiempos/corrección tenemos las siguientes comparaciones, estas simulan una situación en el 4 mes en el que se ve la necesidad de hacer una corrección al software.

Notas a contemplar:

- La mensualidad esta basada en la lista de precios de Oracle para servicios Cloud[13]
- La inversión se esta tomando el coste del servidor y gastos varios.
- Se calcula un salario de 234.38 MXN como base x 6 días a la semana x 4 semanas al mes.
- 15% de tasa de descuento para calcular el Costo-Beneficio.

		Mensualidad(cloud adapters)	$73,150.00 MXN
Tasa de Descuento	15.00%	Inversion Inicial	$65,000.00 MXN
		Salario basado en horas	$234.38 MXN

xenCli

Mes	Ingresos	Costos
0	$73,150.00	$45,468.75
1	$73,150.00	$45,000.00
2	$73,150.00	$45,000.00
3	$73,150.00	$45,000.00

Ingresos	$208,841.67
Costos	$128,881.64
Costos+Inversión	$193,881.64
B/C	1.077160646

clusterSSH

Mes	Ingresos	Costos	
0	$73,150.00	$45,468.75	
1	$73,150.00	$45,000.00	
2	$73,150.00	$45,000.00	
3	$73,150.00	$90,000.00	Simulando falla a 1 mes

Ingresos	$208,841.67
Costos	$154,610.53
Costos+Inversión	$219,610.53
B/C	0.9509638095

openSSH

Mes	Ingresos	Costos	
0	$73,150.00	$45,468.75	
1	$73,150.00	$45,000.00	
2	$73,150.00	$45,000.00	
3	$73,150.00	$225,000.00	Simulando falla a 5 meses

Ingresos	$208,841.67
Costos	$231,797.22
Costos+Inversión	$296,797.22
B/C	0.703651024

Alternativas

Para este proyecto se han considerado las 2 siguientes alternativas, de las cuales se detallan a continuación:

- OpenSSH[14]
  Con esta herramienta que se encuentra por default en cualquier instalación de Linux podemos configurar y utilizar las llaves o Key Authentication para un propósito similar y simplificar el estar introduciendo las contraseñas de cada maquina virtual por acceder.
  Detalles a observar
  - Se tendría que desarrollar un script para la automatización y envió de comandos remotos.
  - Un aspecto a tomar en cuenta es la contraseña de root, el cual cada determinado tiempo tiene que ser cambiada.
  - Existen ocaciones en las cuales el hardware sufre daños, y la tarjeta de red puede ser afectada, en este caso se tendría que reemplazar y generar nuevamente las llaves.
  - Si por alguna razón el servicio de sshd se termina de forma inesperada la comunicación estaría perdida.
  - Se generaría un log general en /var/log/messages y en HISTORY.

- ClusterSSH[15]
  Es una herramienta formidable para la administración masiva de servidores, su manejo es idéntico al comando ssh, el cual podemos administrar n cantidad de servidores.

  Detalles a observar
  - ClusterSSH esta diseñado para masivamente enviar comandos y este sera ejecutado de forma simultanea, si algún error por ejemplo de un comando no se encuentre exportado dentro del PATH, este se tendrá que tratar de forma independiente, por mencionar un ejemplo.
  - Cada maquina que se accede, se tendrá que introducir manualmente la contraseña, ya que ninguna contraseña debe de ser igual.
- Se generaría un log general en /var/log/messages y en HISTORY.

Solución Planificada

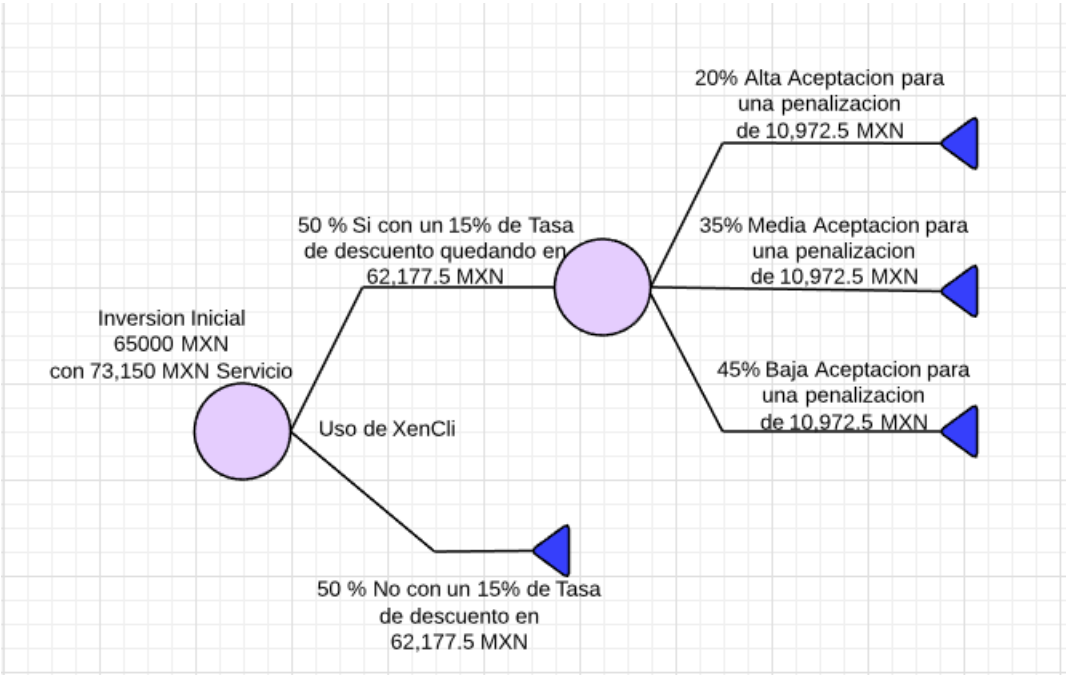
La solución planificada para este proyecto se establece a raíz de la creación de xenCli, que manejara de forma individual o masiva la administración de maquinas virtuales dentro de un Dom0 o servidor Xen; esta herramienta que maneja un diseño simple de cliente-servidor cuenta con las cualidades necesarias para el cumplimiento de los requerimientos anteriormente planteados.

Determinación del enfoque

Gracias al análisis costo beneficio podemos asegurarnos que el desarrollo de xenCli fue mas viable que clusterSSH y openSSH con respecto al tiempo de resolución de algún problema que necesite una actualización.

Valoración Económica

Hablando de la valoración económica, me apoye en los mismos datos ya anteriormente proporcionados en el análisis de costo-beneficio.



El árbol de decisiones esta orientado a la probabilidad de penalización en función a un SLA vencido, el cual es un 15% de acuerdo al contrato. En este análisis no puedo calcular una valoración económica con datos duros, ya que, siendo una empresa de servicios, este varia en función del cliente, servicio proporcionado y penalizaciones estipuladas. El árbol esta basado en el servicio “Cloud Adapters” con una renta mensual de servicio de 73,150 MXN.

Mejoras y oportunidades

En esta sección mencionare las mejoras y oportunidades de este proyecto

- Generar un paquete para su instalación, RPM. **(Mejora)**
- Optimización de código, no que sea necesario, pero se podría mejorar o simplificar. **(Mejora)**
- Agregar la opción de (-a | - -all), el cual pueda tomar directamente del *xm*|*xl list* la lista de maquinas. **(Mejora)**
- A xenCli, se le puede agregar una funcionalidad para escalaciones verticales, el cual se basaría en función del hardware con el que se cuenta, y se necesitaría un análisis de recursos muy detallado de recursos para poder tener esta ventaja, la cual se podría agregar recursos de manera inmediata si una maquina virtual necesitara. Este esquema no es nuevo, y existen muchas empresas desarrolladoras de hardware como IBM, Oracle, etc . Que cuentan con esta solución pero es a nivel servidores, esta seria como una implementación para maquinas virtuales como xenCli lo es. **(Oportunidad)**

Resultados

Xencli es un software diseñado para facilitar las tareas administrativas de maquinas virtuales dentro de un dom0. Las funciones principales han sido completadas de manera satisfactoria.

¿Como funciona xencli?

Xencli valida 3 opciones tanto en formato estándar como en formato posix, F H C, que corresponden a:

Opcion	Estandar	Posix	Argumento
F	-f	--file	Nombre de archivo
H	-h	--hosts	IP o Hostname
C	-c	--commands	Comando a ejecutar

cabe mencionar que siempre debe de usarse en duplas, es decir, -f -c o -h -c, de lo contrario marcara error.

Ejemplos del funcionamiento de xencli

$xencli - -file hosts - -commands ‘df -h | grep nfs’

$xencli -h ‘192.168.1.15 192.168.1.49’ -c ‘cat /etc/redhat-release’

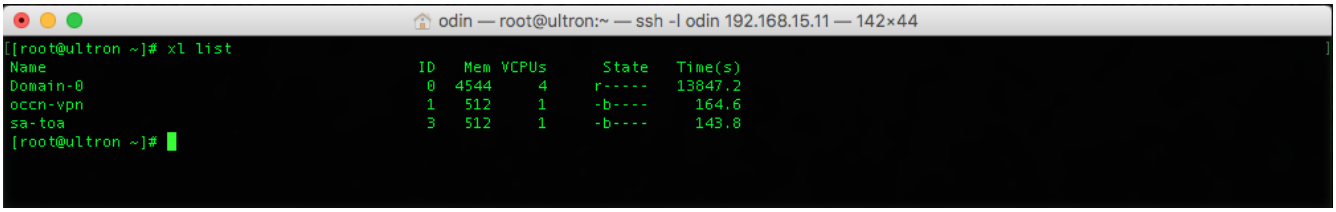
Cuando xencli es ejecutado genera un output directamente a la terminal y de la misma manera, este es almacenado en /var/log/xencli.log para un control detallado de las acciones realizadas.

El formato de salida es el siguiente:

------→ IP / HOST | Comando ejecutado | Fecha y hora

De esta forma tenemos un registro detallado de actividades

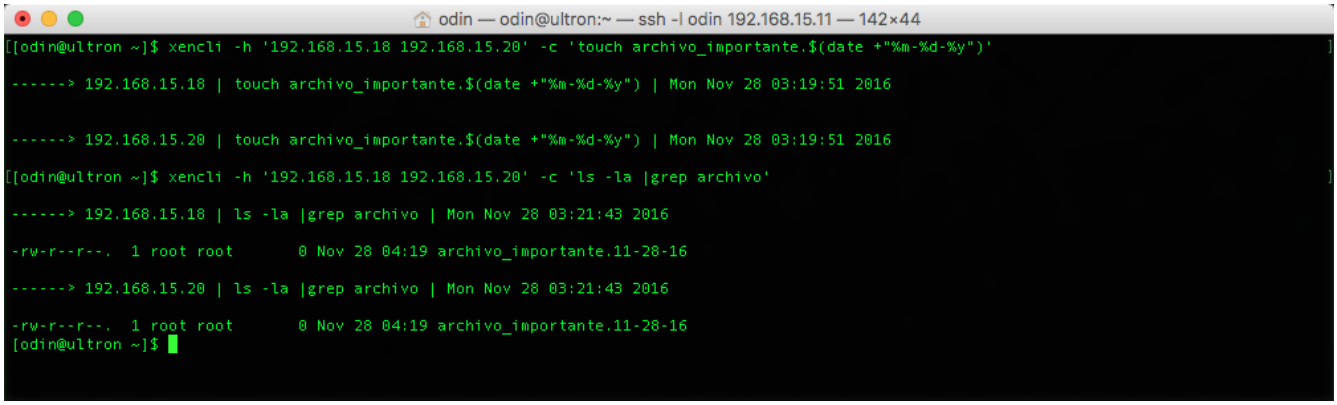
Pero veamos un ejemplo real, en el cual creamos un archivo llamado archivo_importante, este para hacerlo mas interesante, se le agrego la fecha. Este comando es ejecutado en las 2 maquinas virtuales corriendo en este momento.



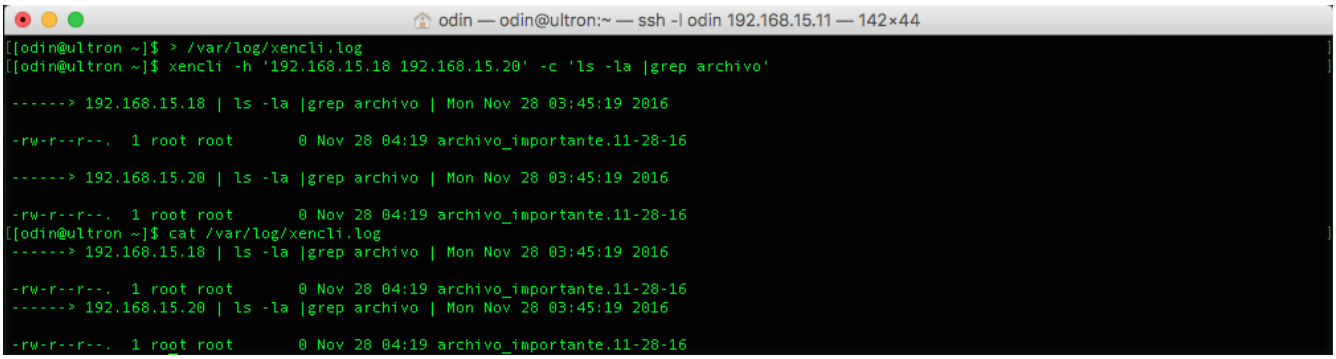
Aquí, quiero hacer énfasis en algo muy importante, un estándar para el uso de maquinas virtuales es nombrarlas de acuerdo a un FQDN y no como se encuentran en esta toma de pantalla.

En lugar de occn-vpn y sa-toa las maquinas virtuales deberían de llamarse occn-vpn.mx.oracle.com y sa-toa.mx.oracle.com, pero como no se cuenta con los nombres de dominio se estarán tomando las capturas de pantalla con IP’s

Continuando con el ejemplo, se puede observar en la primera interacción con xencli que, cuando se mandan a ejecutar no contamos con ningún output de regreso en la terminal ya que de hecho la creación de archivos no regresa ningún caracter



En la segunda parte, contamos con un simple *ls -la* | *grep archivo* el cual si genera información de regreso como se puede observar.



Conclusiones

En el desarrollo de cualquier proyecto, no importando el índole, por mas clara que sea la idea y como dede de ser el resultado al final, necesita ser estudiado y analizado de tal modo que podamos contar con toda la información de como y porque de cada suceso que surja durante el desarrollo del mismo; ¿a que me refiero con esto?, simplemente mirar al rededor, y contemplar opciones ya existentes y analizar si es viable su uso, ya que un desarrollo implica tiempo/dinero, e invertir para desarrollar algo ya existente que satisface las necesidades que actualmente se cuentan no es bueno para el negocio.

Durante el desarrollo de xenCli las necesidades fueron cambiando, no drásticamente pero si de tal forma que sus repercusiones afectaron en tiempo y en el resultado, ya que no es el pensado en un principio. El cambio mas grande fue el lenguaje de programación, el cual paso de ser de un lenguaje interpretado (Python) a uno compilado (C,C++). La justificación resulto muy valida ya que no todos los clientes cuentan con Python[2] e incluso, los clientes que cuentan con este, puede darse el caso de no contar con la misma versión, lo cual generaría errores o un funcionamiento errático; en su contraparte con un binario ejecutable como resultado de la compilación de C, no contaría con esta posibilidad.

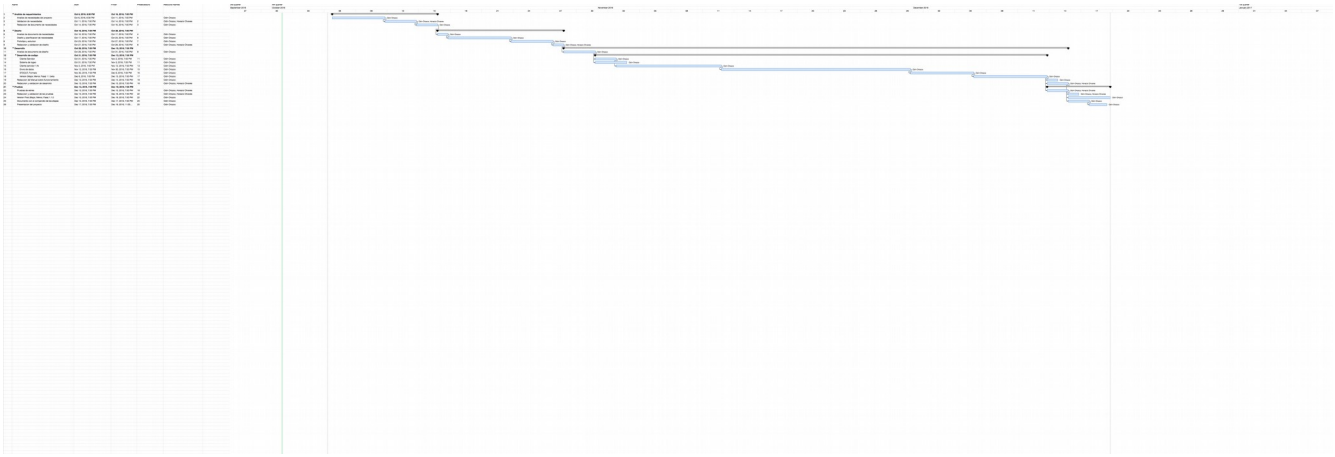
Los resultados actuales son muy satisfactorios, xenCli es una herramienta fácil de utilizar y servirá para garantizar los requerimientos estipulados dentro de los SLA en la empresa.

Y haciendo mención a una conclusión en un entregable anterior, …. *pero para que todo esto se vea reflejado, y cumpla con el objetivo para el cual este fue creado, se necesita una implementación exitosa y esto depende de la aceptación de xencli por parte de la empresa ...[18]*

Definiciones, Acrónimos y Abreviaturas

XenCli	→ Nombre asignado al desarrollo del proyecto
Dom0	→ Nombre asignado a los servidores basados en Xen[1]
DomU	→ Nombre asignado a las maquinas virtuales[1]
CLI	→ Command Line Interface
SA	→ System Administrator
CVS	→ Concurrent Version System[16]
Daemonizado	→ Termino que hace referencia a procesos que corren en background[17]
RHEL	→ Abreviatura de Redhat Enterprise Linux[4]
OVS	→ Abreviatura de Oracle Virtual Server[7]
SLA	→ Service Level Agreement[11]
Errata	→ Documento de errores conocidos
FQDN	→ Fully Qualified Domain Name

Diagrama de Gantt



El diagrama mostrado anteriormente es la planeación original, en si, esta presento atrasos, en tiempo, pero muy pocas modificaciones.

1	**▼ Analisis de requerimientos**
2	Analisis de necesidades del proyecto
3	Validacion de necesidades
4	Redaccion de documento de necesidades
5	Creacion de repositorio para documentos y entregables
6	**▼ Diseño**
7	Analisis de documento de necesidades
8	Diseño y planificacion de necesidades
9	Prototipo y solucion
10	Redaccion y validacion de diseño
11	**▼ Desarrollo**
12	Analisis de documento de diseño
13	Creacion de repositorio CVS para control de versiones
14	**▼ Desarrollo de codigo**
15	Cliente-Servidor
16	Sistema de logeo
17	Cliente-servidor 1-N
18	Envio de datos
19	STDOUT, Formato
20	Version (Major, Menor, Fase) 1.1.beta
21	Redaccion de Manual sobre funcionamiento
22	Redaccion y validacion de desarrollo
23	**▼ Pruebas**
24	Pruebas de estres
25	Redaccion y validacion de las pruebas
26	Version Final (Major, Menor, Fase) 1.1.0
27	Documento con el compendio de las etapas
28	Presentacion del proyecto

Referencias

[1] Xenproject (2013) https://www.xenproject.org

[2] Python (2016) https://www.python.org

[3] CentOS (2016) https://www.centos.org

[4] RHEL (2016) https://www.redhat.com

[5] Diagrama de Gantt (2016, Sep) https://es.wikipedia.org/wiki/Diagrama_de_Gantt

[6] UOC (2016) http://www.uoc.edu

[7] Oracle (2016) http://www.oracle.com

[8] Lutz, M. (2011). Learning Python. Sebastopol, CA: O'Reilly

[9] Apache (2016) http://www.apache.org/licenses/

[10] Python (2016) https://www.python.org

[11] ITIL (2016) https://www.axelos.com/best-practice-solutions/itil

[12] Oracle Docs (2016) http://docs.oracle.com/cd/E22368_01/

[13] Oracle Price List (2016) http://www.oracle.com/us/corporate/pricing/price-lists/index.html

[14] OpenSSH (2016) http://www.openssh.com

[15] ClusterSSH (2015) https://github.com/duncs/clusterssh

[16] CVS (2016) https://en.wikipedia.org/wiki/Concurrent_Versions_System

[17] Daemon (2016) https://en.wikipedia.org/wiki/Daemon_(computing)

[18] Orozco, O. (2016, November 29). Resultados, Valoración Económica y Conclusiones. Conclusiones

Fuentes

En esta sección se muestra el código cliente.c y server.c

Cliente.c

```
/** Estructura arg_struct
 *  Esta estructura nos ayuda a manipular el contenido tanto el host como el
 *  comando y posteriormente lo escribe en un archivo
 * msg1 --> hostname/ip
 * msg2 --> comando
 * msg3 --> path
 * msg4 --> fecha-hora
 */

struct arg_struct
        {
                char msg1[1024];
                char msg2[1024];
                char msg3[1024];
                char msg4[1024];
        };


** Creacion de log
 *  el log se crea de los argumentos 1 y 3 que son host y path respectivamente, args4 es el tiempo
 *  el archivo /var/log/xencli.log ya debe de estar creado y con los permisos adecuados para lectura y escritura
 * msg1 --> hostname/ip
 * msg2 --> comando
 * msg3 --> path
 * msg4 --> fecha-hora
 */

std::ofstream logfile;
logfile.open (args->msg3, std::ofstream::out | std::ofstream::app);
logfile << "------> "<<args->msg1<<" | "<<args->msg2<<" | "<<args->msg4<<std::endl<<recvBuff;
logfile.close();


/** Funcion getFileContent
 *  Esta funcion lee de un archivo los hostnames o IP de los equipos a administrar
```

```
*/



const char* getFileContent(const std::string path)
        {
                std::ifstream file(path.c_str());
                std::string content((std::istreambuf_iterator<char>(file)), std::istreambuf_iterator<char>());
                return content.c_str();
        }



/** Funcion principal
 *  Esta funcion lee los parametros y ejecuta todas las funciones creadas para el correcto funcionamiento de xencli
 */



int main (int argc, char **argv)
        {

                char ch = 0;
                char flagSource = 0;
                char flagCmd = 0;
                char hosts[1024];
                char hostsFile[30];
                char cmd[1024];

                static struct option long_options[] =
                        {
                                {"file",     required_argument, 0, 'f'},
                                {"hosts",    required_argument, 0, 'h'},
                                {"commands", required_argument, 0, 'c'}
                        };

                bzero(cmd, 1024);
                while ((ch = getopt_long(argc, argv, "f:c:h:", long_options, NULL)) != -1)
                        {

                           switch (ch)
                                {

                                   case 'f':
                                        if(flagSource == 0)
                                                {
                                                        if(optarg[0]=='-')
                                                                {
```

```
                                                                                printf("Option requires an argument -- '%c'\n", ch);
                                                                                break;
                                                                            }
                                                                        flagSource+=2;
//                                                                      printf("EFE %s\n", optarg);
                                                                        strcpy(hostsFile, optarg);
                                                                    }
                                                    break;

                                                    case 'c':
                                                        if(optarg[0]=='-')
                                                                {
                                                                    printf("Option requires an argument -- '%c'\n", ch);
                                                                    break;
                                                                }
                                                            flagCmd = 1;
                                                    strcpy(cmd, optarg);
//                                          printf("SE %s\n", optarg);
                                                            break;

                                                    case 'h':
                                                            if(flagSource==0)
                                                                {
                                                                    if(optarg[0] == '-')
                                                                        {
                                                                            printf("Option requires an argument -- '%c'\n", ch);
                                                                            break;
                                                                        }

                                                                    flagSource++;
//                                                                  printf("ACHE %s\n", optarg);
                                                                    strcpy(hosts, optarg);

                                                                }
                                                            break;
                                            }
                                }



//                              printf("Validando banderas...\n");
                                    if(flagSource == 0 || flagCmd == 0)
                                            {
                                                    printf("Must use -f or -h and -c option.\n");
                                                    return 1;
                                            }
```

```
                                        else if(flagSource == 2)
                                                {
//                                                      printf("Abriendo archvo...\n");
                                                      strcpy(hosts, getFileContent(hostsFile));
                                                }


                                char *ptr;
                                time_t rawtime;
                                struct arg_struct args;
                                ptr = strtok(hosts,"\n\r ");
                                while(ptr!=NULL)
                                        {


                                                time (&rawtime);
                                                strcpy(args.msg1, ptr);
                                                strcpy(args.msg2, cmd);
                                                strcpy(args.msg3, "/var/log/xencli.log");
                                                strcpy(args.msg4, ctime(&rawtime));
                                                printf("\n------> %s | %s | %s\n",ptr, cmd,args.msg4 );
                                                Procesa(&args);
                                                ptr = strtok(NULL,"\n\r ");
                                        }
                        exit (0);
}
```

Server.c

```
std::string exec(const char* cmd) {
char buffer[128];
std::string result = "";
std::tr1::shared_ptr<FILE> pipe(popen(cmd, "r"), pclose);
if (!pipe) throw std::runtime_error("popen() failed!");
while (!feof(pipe.get())) {
                        if (fgets(buffer, 128, pipe.get()) != NULL)
                        result += buffer;
                        }
return result;
}

int main(int argc, char *argv[])
{
    int listenfd = 0, connfd = 0;
    struct sockaddr_in serv_addr;

    char sendBuff[1025];

    int n=0;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sendBuff, '0', sizeof(sendBuff));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(5000);

    bind(listenfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));

    listen(listenfd, 10);

   std::string response = "";
   while(1)
   {
      connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);

   bzero(sendBuff, 1025);
```

```
  n = read(connfd, sendBuff, 1024);
  if(n < 0)
     printf("Error al leer\n");

     response = exec(sendBuff);

     bzero(sendBuff,1025);
     strcpy(sendBuff, response.c_str());
     write(connfd, sendBuff, strlen(sendBuff));

  close(connfd);
    }
}
```

Man de Xencli

El manual de xenCli es el que se muestra a continuación:

NAME

xc - herramienta administrativa para maquinas virtuales bajo xen.

SYNOPSIS

xc [OPCION] ... [OPCION] ...

DESCRIPTION

xc funciona con POSIX y el estándar, se necesita que el Servidor este

funcionando dentro del DomU

-h, --hosts

lista de IP o hostname a administrar

-f, --file

lista de IP o hostname a administrar, un host o ip por linea

-c, --command

comando a enviar

EXAMPLES

xc -h '192.168.1.30 my.host.com' -c 'tail /var/log/messages |grep cpu'

AUTHOR

Desarrollado por Odin Orozco.