Sistemas Distribuidos: Parcial 3

Universidad Icesi
Departamento de Ingeniería
Ingeniería Telemática
Jaime Vélez Escandón
https://github.com/jaimevlz7/sd-parcial3-2017a

Objetivos

- Justificar los casos donde deban usarse máquinas virtuales o contenedores virtuales
- Comprender el concepto de microservicio y su relación con las tecnologías de aprovisionamiento automático

Descripción

Por medio de las referencias sugeridas y otros medios de información en Internet y literatura especializada complete las asignaciones que se presentan a continuación. Tenga en cuenta después de cada respuesta consignar las referencias empleadas.

1. **Realice un cuadro comparativo donde cite desventajas y ventajas con respecto al aprovisionamiento con máquinas virtuales y el aprovisionamiento con contenedores virtuales.**

Dependiendo del contexto en el que se está trabajando (producción, pruebas, desarrollo, etc.) una u otra opción de virtualización (Máquinas virtuales o contenedores virtuales) puede ser la más apropiada para resolver el problema.

Por ejemplo, la tecnología de contenedores virtuales ofrece una mayor flexibilidad y portabilidad debido a que pueden ser puestos en marcha en el lugar que deseemos, ya sea en las instalaciones físicas, la nube pública, nube privada, incluso en una computadora personal de un desarrollador. El nivel de flexibilidad permite que al usar tecnologías de paquetización (de los contenedores) una aplicación pueda ser trasladada de un ambiente a otro con mínimo impacto, permitiendo que en algunos escenarios de misión crítica y alta carga de procesamiento escalar, simplemente clonando nuevas instancias de la aplicación o simplemente clonar el contenedor, lo que hace el proceso de crecimiento rápido y seguro.

Por el lado de las máquinas virtuales al utilizar un software aislado que puede ejecutar sus propios sistemas operativos y aplicaciones como si fuera una computadora física. Se comporta exactamente como una máquina física, ya que contiene su propia CPU, memoria RAM, disco duro y tarjeta de interfaz de red (NIC) virtuales (Basados en software, Hardware digital o virtualizado).

La principal ventaja contra los contenedores virtuales es la seguridad que provee la capa lógica de hardware, ya que un ataque puede llegar únicamente a afectar la VM comprometida, aislando las demás que se encuentran en ese servidor físico.

Característica	**Máquinas Virtuales**	**Contenedores Virtuales**
Seguridad	Seguro (Hardening).	Add-on para seguridad.
Portabilidad	No.	Sí.
Escalable	Depende del HW.	Depende de las necesidades.
Tiempos de aprovisionamiento	Horas.	Segundos.
Soporte en nube	Sí.	Sí.
Aplicaciones	Multiples app.	Escenario común: una app.
Tiempo de clonación	Horas.	Segundos.
Dependencia de S.O	No.	Sí.
Compatibilidad	Sí.	No. Dependencia de S.O.
Aislamiento	Sí.	No.
Costos	Elevados (Licencias).	Tecnología Open Source.

2. **Realice un cuadro comparativo donde cite las diferencias entre vagrant y docker. El cuadro comparativo debe dar respuesta a los casos en que es útil emplear cada tecnología.**

Ambas herramientas son excelentes para entornos de desarrollo, pruebas y producción, ya que no debemos preocupándonos por las posibles diferencias entre tecnologías, software o dependencias o la falta de bibliotecas o librerías. Ofreciendo una mayor flexibilidad y automatización que una máquina virtual tradicional, lo que pueden suponer un ahorro enorme de problemas, dificultades de gestión, dinero, tiempo y alcance de muchos proyectos.

Docker es una tecnología Open Source para crear contenedores virtuales ligeros y portables para cualquier aplicación (Aplicación de un sistema operativo específico, hasta ahora Windows o Linux). Podemos empaquetar una aplicación en un contenedor Docker desde nuestro equipo de trabajo y moverla a un servidor, una máquina virtual o a la nube sin hacer ningún cambio. Con este tipo de tecnología nos olvidamos de dependencias, versiones del sistema operativo, bibliotecas y demás problemas de una virtualización tradicional. Es una herramienta muy poderosa, que sigue y va de la mano con la filosofía de integración continua, para evitar los problemas comunes de “¡en mi computador funcionaba!”.

Por otro lado Vagrant es otro tipo totalmente diferente de tecnología. Lo que proporciona Vagrant es la creación y gestión sencilla de entornos de trabajo "portables" y replicables que funcionan sobre tecnologías de virtualización conocidas (Virtualbox, VMware, etc.),

ofreciendo además un modo de trabajo claro para poder transportar dichos entornos y que funcionen sin problemas en otro lugar (Teniendo en cuenta las necesidades de hardware y software del equipo anfitrión).

Característica	**Docker**	**Vagrant**
Tipo de virtualización	Contenedores	Máquina virtual
Nivel de aislamiento	Débil	Muy alto
Tiempo de creación	Menos de 10 min	Más de 10 min
Tamaño del despliegue	Al menos 100MB	Al menos 1GB
Tiempo de arranque	Segundos	Minutos
Impacto en el sistema	Muy bajo	Alto
Garantiza recursos en el S.O.	No	Sí
Cantidad de nodos por máquina	Más de 50	Maximo 10 (Depende)
Ambiente de ejecución	Pruebas, dev y producción.	Pruebas, dev y producción.
Ventaja	Rápido y ligero.	Fácil de gestión

3. **¿Cuál es la relación entre la tecnología de contenedores virtuales y microservicios?**

La principal relación entre contenedores virtuales y los microservicios es la flexibilidad al momento de desarrollar una actividad. Por ejemplo para los microservicios se tiene un servicio (aplicación) para cada proceso o actividad (un servicio o nodo para base de datos, uno para envíos, otro para pagos, etc.) independientemente de cómo operen los demás servicios, el servicio que yo consulté debe funcionar de manera correcta estando completamente separado de los demás. En pocas palabras tengo una aplicación grande dividida en sub aplicaciones que cumplen tareas específicas, para dar solución al problema. En el caso de los contenedores virtuales es algo muy parecido solo que en este caso en vez de tener una aplicación grande tenemos un sistema operativo grande, que puede ser dividido o segmentado en un conjunto más pequeño de “mini” sistemas operativos (contenedores) con una aplicación (únicamente esa aplicación) para resolver un problema más grande. Por lo que en resumidas cuentas los contenedores y los microservicios siguen el enfoque de “divide y conquistaras”, separar problemas muy grandes, en problemas pequeños que puedan resolverse de manera fácil. Otra las principales relaciones entre contenedores y microservicios es el manejo de actualizaciones de código, versión o escalamiento, con una aplicación, servicio o infraestructura basada en microservicios es mucho más fácil, actualizar un nodo (aplicación o microservicio) que lo necesite, que actualizar toda una aplicación, teniendo en cuenta que usando cualquiera de estas tecnologías solo pararemos esa actividad (nodo que necesita actualizarse) y el resto de actividades funcionarán correctamente sin interferencias.

4. **¿Por qué los microservicios podrían requerir una aproximación DevOps?**

Los microservicios requieren de una aproximación de DevOps, primero para acelerar y automatizar la entrega de aplicaciones, adoptando los principios de integración continua y la entrega continua. Utilizando DevOps para facilitar la construcción, pruebas y orquestación de aplicaciones, así como la incorporación de otros servicios adicionales.

Segundo, los microservicios que normalmente siguen un patrón de diseño de desarrollo ágil de mantener las cosas muy pequeñas y repetitivas (“Divide y conquistaras”), lo que sigue la misma filosofía de DevOps.

Tercero, utilizar soluciones orientadas a los microservicios, ofrece una manera sencilla de empaquetar el código y enviarlo a los respectivos canales de entrega (Distribución y despliegue de las operaciones en ambientes de producción), lo que agiliza el flujo de trabajo.

Por último, con DevOps y microservicios, los nuevos miembros del equipo pueden construir sus soluciones más rápidamente si solo tienen que familiarizarse con el servicio en el que están trabajando y no con lenguajes de programación, codigo, pseudo código, y versiones de la aplicación, simplemente trabaja en el servicio asignado y se “despreocupa” de las otras actividades. Es decir, pueden implementar de forma independiente cada servicio, lo que ayuda con el aislamiento de fallos.

5. **¿Por qué la arquitectura de una aplicación orientada a microservicios requiere herramientas de integración continua?**

Una arquitectura orientada a microservicios requiere herramientas de integración continua principalmente, para acelerar y automatizar la entrega de aplicaciones, ya que la integración continua, se suele utilizar para facilitar la construcción de pruebas, código y orquestación de aplicaciones, así como la incorporación de otras aplicaciones o servicios. Adicionalmente, gracias al manejo de versiones que una herramienta de integración continua nos brinda, los desarrolladores, podrán darse cuenta en tiempo real sobre todas las versiones o modificaciones del código y saber si ese cambio afecta o no a la aplicación. Debido a que al ser una plataforma de microservicios, el manejo de versiones y actualizaciones de código del servicio es muy común.

6. **Es aconsejable el uso de contenedores dentro de contenedores para el caso de integración continua? Justifique su respuesta.**

El uso de contenedores dentro de contenedores es recomendable para ambientes de pruebas, para acceder a ciertas APIs de manera local alojadas en algunos contenedores y además de para hacer uso de algunas herramientas de integración continua teniendo en

cuenta que hay que seguir ciertos pasos y tener un conocimiento acertado y detallado de los demonios de docker para controlar en qué momento publicar o no una versión de nuestro servicio o aplicación. Usar un contenedor dentro de un contenedor no traería problemas al usarlo para herramientas de integración continua, debido a que el segundo contenedor será usado para las veces de herramienta de integración para publicar, validar y probar las versiones del código, servicio, sistema o máquina, realizando consultas básicas al servidor (publicando avances y versiones) donde se realizan las pruebas y procesos ya elaborados anteriormente.

El problema de usar contenedores dentro de contenedores, existe a la hora de usar volúmenes o a la hora de almacenar información debido a que los filesystem del segundo contenedor no corren sobre un formato comúnmente usado como NFS, EXT4, EXT3 etc (Los contenedores corren en sus filesystem unos formatos de archivos propios, diferentes a los usados normalmente), sino que corre en otros formatos que no son compatibles entre sí (Compatibles entre capas). Por lo que el manejo del almacenamiento es un gran impedimento a la hora de usar la información (copiar, editar, escribir, etc.).

7. **¿Cómo se realizan las pruebas automáticas en una arquitectura de microservicios?**

Lo primero que se debe tener en cuenta para la automatización de pruebas en una arquitectura de microservicios es contar con las herramientas técnicas, tecnológicas y teóricas para implementarlas. Por ejemplo, una de las claves para agilizar ese procesos, detectar errores en puntos del desarrollo en los que nos cueste menos solucionarlos y así desarrollar con más seguridad, es la optimización y automatización de ciertos procesos y **pruebas**. Algunas de estos procesos pueden ser agilizados usando herramientas como la integración continua.

Para realizar las pruebas automáticas se debe tener en cuenta:

1. **Herramientas:** Las herramientas de automatización de pruebas, como una herramienta de integración continua.
2. **Metodología:** Se debe definir una metodología apropiada y acorde al proyecto o necesidades del equipo.
3. **Diseño apropiado:** Se debe diseñar de manera adecuada las pruebas o el conjunto de pruebas para cada uno de los niveles o puntos de evaluación.
4. **Definir niveles de pruebas:** Que nivel de pruebas, se hacen manuales, pruebas de GUI, nivel de pruebas de integración, nivel de pruebas unitarias, pruebas de API, de conexión, etc.
5. **Estrategía:** Y finalmente definir una estrategia de desarrollo de pruebas que permita dar solución teniendo en cuenta los niveles definidos anteriormente.

Adicionalmente para una serie de pruebas en una arquitectura de microservicios se debe tener en cuenta que hay distintos tipos de pruebas; se pueden realizar pruebas a nivel de procesos, actividades, servicios, integración y comunicación. Debido a que como cada parte

de la solución se comporta como un servicio independiente se deben (pueden) realizar pruebas a cada uno de estos servicios de manera separada o a cada una de las partes de la comunicación. Siguiendo o no un patrón de pruebas automáticas.

8. **¿Qué beneficios trae para una datacenter emplear la tecnología de microservicios? (incluya en su respuesta un aspecto relacionado con la migración en vivo)**

Para un datacenter es de gran importancia, teniendo en cuenta que por motivos de escalamiento o crecimiento del servicio a nivel de recursos (Almacenamiento, procesamiento, etc) es necesario mudar ciertos servicios de una máquina a otra más potente. Por lo que implementar tecnología basada en microservicios que es portable, eficiente y de fácil clonación. Por ejemplo tener una solución de microservicios de compra, ventas e intercambios como OLX y que por modelo del negocio la parte de ventas se la de más alta concurrencia y uso, tener implementada esta solución en microservicios nos permite unicamente mover o hacer migración del componente de ventas de este servicio sin afectar los otros servicios que están corriendo y funcionando paralelamente, además sería solo mover clonar, mover y reiniciar, o en su defecto usar un balanceador de carga y ahora tener dos nodos para ventas, independientes y que trabajan exactamente igual.

Otro de los aspectos más relevantes es el hecho de que al ser los microservicios tan portables en un datacenter podemos tener un centenar de soluciones basadas en microservicios, que pueden crecer teniendo en cuenta HW y necesidades.

Referencias

- http://martinfowler.com/articles/microservices.html
- http://www.martinfowler.com/articles/continuousIntegration.html
- http://martinfowler.com/articles/microservice-testing/#conclusion-summary
- https://jpetazzo.github.io/2015/09/03/do-not-use-docker-in-docker-for-ci/
- http://stackoverflow.com/questions/16647069/should-i-use-vagrant-or-docker-for-creating-an-isolated-environment
- http://dev.otto.de/2015/09/30/on-monoliths-and-microservices/
- http://www.slideshare.net/ewolff/microservices-architecture-for-agile-software-development
- http://www.infoworld.com/article/2906362/data-center/dawn-of-the-data-center-operating-system.html
- https://github.com/stevenalexander/docker-authentication-authorisation
- https://www.paradigmadigital.com/dev/devops-microservicios-autoescalado-desarrollo-apps-las-claves-del-exito-amazon/
- http://blogs.itpro.es/maitchovcow/archivos/33

- http://searchdatacenter.techtarget.com/es/consejo/Los-secretos-de-la-migracion-en-vivo-RHEV
- https://lapastillaroja.net/2014/07/microservicios/
- https://innovacionactiva.ieci.es/2015/02/10/nuevas-tendencias-contenedores-y-microservicios/
- http://www.javiergarzas.com/2015/01/testing-agil.html
- http://www.javiergarzas.com/2015/12/y-si-queremos-orquestar-varios-contenedores-docker-microservicios-docker-compose-yaml-que-jaleo-22.html
- http://asesoftware.com/site/blog/2016/03/08/la-arquitectura-microservicios/
- https://www.zankuda.com/2015/08/01/Micro-Servicios-Con-NodeJS/
- https://blog.docker.com/2013/09/docker-can-now-run-within-docker/
- https://forums.docker.com/t/how-to-run-docker-inside-a-container-running-on-docker-for-mac/16268/3
- http://container-solutions.com/running-docker-in-jenkins-in-docker/