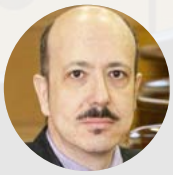
Una aproximación ágil a TaaS basada en contenedores


Joaquín Salvachúa
Departamento de Ingeniería de Sistemas Telemáticos UPM
joaquin.salvachua@upm.es


Gabriel Huecas
Departamento de Ingeniería de Sistemas Telemáticos UPM
gabriel.huecas@upm.es


Pedro Verdugo
Doctorando en Ingeniería de Sistemas Telemáticos UPM
pmverdugo@dit.upm.es

Los escenarios actuales de despliegue en nube implican la necesitad de un testeo rápido de software orientado a usuario, en entornos de hardware y red diversos, heterogéneos y muchas veces desconocidos, haciendo difícil asegurar un testing in-site óptimo o reproducible. El artículo actual propone el uso de virtualización ligera basada en contenedores con una estrategia de despliegue ready-to-run, just-in-time, con objeto de minimizar el tiempo y los recursos necesarios para el prototipado de componentes estándares en sistemas PaaS. Con tal fin, estudiaremos un caso de uso específico, que consiste en proveer a los usuarios de software previamente testeado, empaquetado y configurado, garantizando la viabilidad de dicho software, la integridad sintáctica del sistema de despliegue provisto y la disponibilidad de dependencias necesarias, así como la comprobación del software específico una vez desplegado y ejecutándose.

Desde un punto de vista arquitectural, usando paquetes de despliegue comunes y de uso estándar como Chef o Puppet, alojados en cargas paralelizables sobre imágenes Docker listas para el despliegue, podemos minimizar el tiempo requerido para las pruebas y validación del despliegue completo de sistemas multicomponente, así como envolver las características provistas de forma común a través de una API RESTful accesible por el usuario.

La infraestructura propuesta está actualmente disponible de forma libre como parte de la iniciativa FIWARE de la UE, y está abierta a colaboraciones y extensiones de terceras partes desde una perspectiva FOSS.

En los últimos años, la idea de emplear tecnologías en la nube para el testing funcional y de rendimiento se ha revisitado en numerosas ocasiones, y la enorme mayoría de los autores coinciden en las ventajas económicas, organizativas y de seguridad de una plataforma de pruebas basada en la nube. Aquí propondremos una implementación de una de las susodichas plataformas de pruebas basada en cloud, funcional y orientada a objetivos. Definiremos algunos de los términos más prominentes empleados en este estudio, como es el caso de los paradigmas de Testing as a Service (TaaS)y Virtualización Ligera. Como ya se ha señalado, hay un interés vigente, a la vez con avances constantes en el campo del cloud testing, de forma que analizaremos a continuación las contribuciones más relevantes y prominentes de nuestro campo de estudio

¿Qué es el TaaS, Testing as a Service?

Los modelos de despliegue en cloud tradicionales puede definirse brevemente como sigue:

- Software as a Service (SaaS): Provee paquetes de software listos para desplegarse en una plataforma cloud.

- Platform as a Service (PaaS): Provee un entorno software construido previamente, habitualmente un sistema operativo con librerías y dependencias opcionales.

- Infrastructure as a Service (IaaS): Provee hardware virtualizado o bare metal bajo demanda

Pero para nuestro caso de estudio, un modelo recientemente adoptado de especial interés es el de Testing as a Service (TaaS), definido como la comprobación de software automatizada a través de un servicio basado en nube.

Hay varias ventajas bien conocidas derivadas de adoptar una perspectiva TaaS para el testing en cloud, las más importantes de las cuales enunciaremos a continuación:

- Entorno de pruebas escalable: podemos adaptar de forma elástica la cantidad y potencia de recursos hardware a los requisitos de pruebas.

- Reducción de costes: Reutilizando entornos de hardware y recursos, podemos minimizar el número de usos de infraestructura bloqueados para un periodo de tiempo determinado.

- Modelos de servicio basados en utilidad: Personalizar los servicios de pruebas ofrecidos de forma que se adapten a aquellos requeridos por las utilidades alojadas

- Servicios de pruebas bajo demanda: Nuestros servicios de pruebas pueden estar disponibles plenamente todo el tiempo.

Las capacidades habituales para un sistema TaaS se organizan como sigue:

- Administración de procesos TaaS: Entendida como el control del flujo de pruebas.

- Administración de requisitos de calidad de servicio (QoS): Delimitación de los parámetros requeridos para un objetivo de QoS determinado.

- Servicio de entorno de pruebas: Dedicado a habilitar y ofrecer entornos virtualizados para pruebas.

- Servicio de soluciones de pruebas: Paquetes de modelos de pruebas estandarizados, conocidos como soluciones.

- Servicio de simulación de pruebas: ofrece la capacidad de simulación de entornos externos.

- Servicio de pruebas bajo demanda: Provee un entorno de ejecución continuo para las pruebas.

- Servicio de monitorización y trazado: Responsable de la monitorización y cuantificación de resultados y comportamientos de pruebas.

Virtualización ligera

Las técnicas de virtualización ligera llevan algunos años siendo foco de estudio, pero no han sido populares hasta el advenimiento de los sistemas de máquina virtual basada en contenedores

Esta aproximación propone el uso de los espacios de nombres y grupos de control del kernel de Linux como objetos similares a los de una MV, y presenta varias ventajas frente a la virtualización de pila completa:

- Rendimiento: Comparativamente similar a la ejecución de procesos nativos, las MV clásicas necesitan soporte de virtualización de hardware para conseguir resultados cercanos.

- Utilización de recursos: La compartición de kernel y módulos, así como opcionalmente de espacios de memoria, permite una huella del contenedor más pequeña en proporción a la de una MV completa.

- Funcionalidad: los puntos anteriormente mencionados permiten sustituir casi completamente a las aplicaciones de SO comunes, simplificando la gestión de paquetes y actualización de versiones.

Recientemente ha habido preocupación sobre la seguridad de los pro-

cesos basados en contenedores, pero en el momento de escribir este artículo, los avances en los módulos de seguridad de control de grupos (GRSEC) y administración de espacios de nombres en memoria, han relegado dichas preocupaciones a la obsolescencia.

Trabajos relacionados

Actualmente el campo del cloud testing se basa principalmente en la validación de componentes de la nube externa principal, como ponen de manifiesto iniciativas como Open Cloud Lab. También es de reseñar que las propuestas de estructuras como Progress Cloud Test Framework tienden a integrar las suites en el propio framework de pruebas. Últimamente esta idea ha sido parcialmente relegada a un segundo plano, para permitir cargas de pruebas más heterogéneas, como muestra la propuesta FCHTS.

Desde el punto de vista de la virtualización, la principal preocupación con el despliegue de MV ha sido sobre las estrategias de planificación, pero no tanto sobre la tecnología de virtualización a emplear. Encontramos una excepción a esto en la propuesta de Kuo, que basa sus despliegues en el entorno OpenStack. Muy cerca de nuestra área de interés, el entorno de pruebas CUTEi propone una solución de pruebas basada en contenedores, si bien orientada a problemas de simulación de redes.

También ha habido soluciones proponiendo la generación inteligente de casos de uso de pruebas, así como la adaptación a aplicaciones variables, pero sin una vista extendida de la implementación, resulta difícil medir las tasas de éxito y rendimiento de dichas propuestas

Otra línea de trabajo prometedora y que goza actualmente de popularidad es la aplicación de estas técnicas a las pruebas de aplicaciones móviles, específicamente la plataforma Android.

Caso de estudio: Definición

Para definir los usos aplicables de nuestro sistema, comenzaremos detectando y definiendo los requisitos de uso. En el entorno FIWARE se ofrece a los usuarios una selección de infraestructuras base para sus despliegues en nube, así como la personalización de dichos despliegues mediante la instalación de software previamente empaquetado. Los ciclos de desarrollo de dicho software habitualmente son cortos, siguiendo un paradigma ágil.

Por lo tanto, surge la necesidad de comprobar constantemente nuevos paquetes disponibles, así como nuevas versiones y parches de los existentes para todas las infraestructuras provistas.

El sistema actualmente bajo estudio propone la automatización de dichas tareas mediante un entorno rápido, orientado a recursos, bajo demanda y siempre disponible. Para alcanzar dichos objetivos, definiremos un ciclo de trabajo principal, englobando las capacidades TaaS previamente vistas tal como se relacionan con nuestro presente escenario.

Figura 1 Pasos del proceso de pruebas

Validación de elementos de despliegue

Definiremos un elemento de despliegue con el conjunto de instrucciones y estados necesarios para la instalación y configuración de un paquete de software determinado. El objetivo principal de nuestro sistema será por consiguiente el instanciar un elemento de despliegue determinado en un sistema operativo dado.

Hay distintas variables a considerar que serán de interés para la monitorización de nuestro despliegue, y que detallaremos como sigue:

- Sistema operativo: Se garantizará el funcionamiento de los elementos de despliegue al menos sobre las versiones actuales de Ubuntu/ RedHat linux
- Software de Aprovisionamiento: Se soportarán elementos de despliegue basados en Chef, Puppet y opcionalmente Murano
- Gestión de dependencias: Se aprovisionarán e instalarán todas las dependencias necesarias para cada paquete.
- Comprobación de sintaxis: Todos los elementos de despliegue provistos cumplirán con la sintaxis del sistema de aprovisionamiento seleccionado.
- Comprobación de despliegue: Todos los elementos de despliegue provistos completarán el proceso de despliegue sin errores
- Prueba de salud (Health Check): Opcionalmente, se ejecutará una comprobación posterior simple de la instalación y configuración correcta del paquete

Siguiendo la problemática arriba marcada, podemos definir el ciclo de trabajo principal de nuestro proceso de pruebas como se indica en la Figura 1.

Caso de estudio: Implementación del sistema

API Restful

El componente principal de nuestro sistema es una API RESTful basada en RESTfulFramework 3.3.3 y alojada en un servidor Django versión 1.8.3 El código de la API está escrito en el interprete de Python 2.7.11, y cumple las guías de estilo de OpenStack así como el estándar pep8.

La API es la única interfaz externa del sistema, y está documentada de forma completa en el servicio *APIary*, siendo comprobable su funcionamiento en el mismo.

Para cubrir los casos de uso definidos previamente se ha implementado una arquitectura de sistema de referencia. La Figura 2 ofrece una vista simplificada de la estructura actual del sistema, que estudiaremos en detalle en los siguientes párrafos.

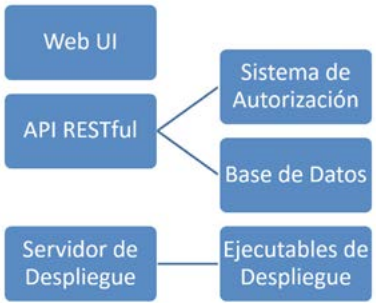


Figura 2. Componentes del sistema

Siguiendo el concepto de aplicación universal y uso libre, nuestra pila de software estará íntegramente compuesta por software libre de código abierto (FOSS) y se liberará completamente bajo la licencia *Apache License Version 2.0*.

Interfaz Web

La interfaz de usuario está escrita en javascript mediante el framework *Backbone.js*, y puede trabajar independientemente, siendo una solución íntegramente basada en navegador. La administración de dependencias de javascript se realiza mediante *Require.js* 2.2.0, permitiendo cumplir con la definición de módulos asíncronos y carga bajo demanda *AMD*.

La UI está basada en el patrón de aplicación de página única y está enfocada al uso por personal técnico, por lo que es extremadamente simple y diseñada para una experiencia de usuario rápida y orientada a flujo de trabajo gracias a la estructura AMD y el framework elegidos, el uso de recursos es extremadamente bajo.

La Figura 3 muestra el estado actual de la UI del Validador.

Base de Datos

Los usos de objetos de la base de datos y administración están cubier-

tos por modelos Django, que tendrán un rol importante en el flujo de trabajo de la aplicación:

- Almacenamiento de imágenes de sistema disponibles.
- Almacenamiento de repositorios de software y versiones externos.
- Almacenamiento de historial de uso y monitorización.

API de Autenticación

Los servicios de autenticación y autorización se proveerán de forma externa por el FIWARE Identity Manager, una extensión del entorno de autorización Keystone de OpenStack. Se ha adaptado al mismo un plugin de autorización de Django personalizado para garantizar el cumplimiento con el esquema de autenticación seleccionado, el cual está disponible en el repositorio github de la aplicación.

Servicio de Despliegue

El servicio de despliegue de imágenes está basado en un servidor Docker con nivel de api 1.23, y es responsable de la generación de imágenes, instanciación de las imágenes en contenedores flexibles y ejecución, configuración y monitorización de dichos contenedores. Tras una valoración racional del bajo número de usuarios y tareas concurrentes, se ha implementado un planificador simple bajo demanda

Entorno Hardware

Como se observa en la Tabla 1, los requisitos hardware para nuestra implementación son considerablemente bajos. La separación de funcionalidades en máquinas distintas puede evitarse si se requiere integración completa, como en el caso de la infraestructura de validación de medidas que se detalla en la sección siguiente.

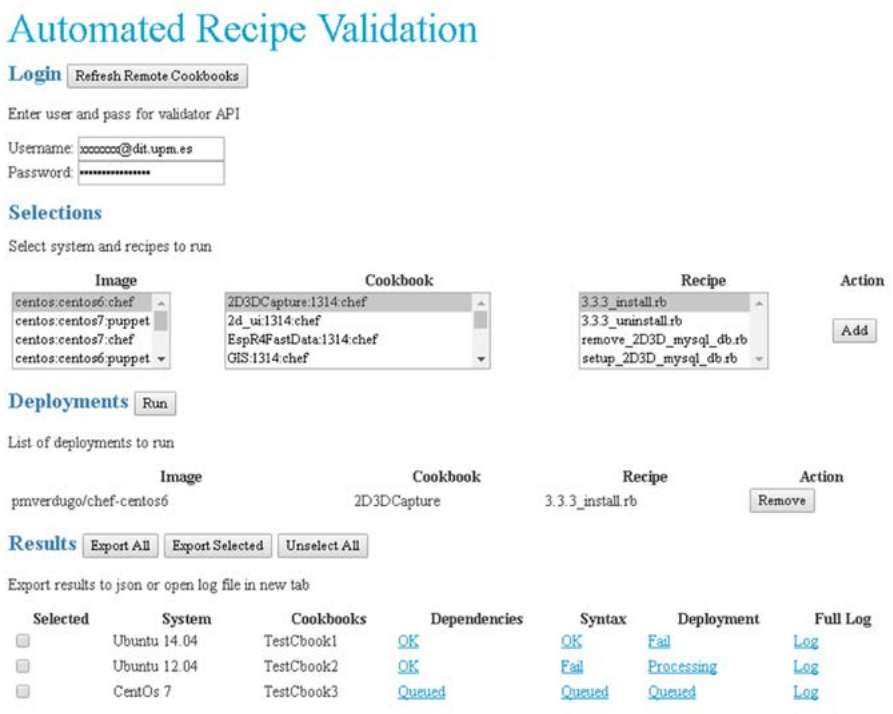


Figura 3 Interfaz Web

Tabla 1: Entorno Hardware

Rol	CPUs	Ram	Red	OS
API Server	2	2048	1000	Ubuntu 14.04
Servidor de Despliegue	4	4096	1000	Ubuntu 14.04
Servidor Web	2	2048	100	Windows 7 SP

Conclusiones

El sistema como se presenta, cubre de forma suficiente los requisitos fijos iniciales. Una breve validación de las características del sistema principal se incluye en la siguiente lista:

- Gestión de procesos TaaS: El proceso de pruebas se ha unificado en un único componente.
- Gestión de requisitos de calidad de servicio: El sistema cubre los requisitos de calidad de servicio para las cargas de trabajo actuales.
- Servicio de entorno de pruebas: Los entornos de virtualización ligera implementados como imágenes Docker están disponibles al vuelo para despliegue inmediato.
- Servicio de solución de pruebas: Dado el contexto de comprobación de elementos de despliegue, el sistema implementa las soluciones de aprovisionamiento de uso más extendido.
- Servicio de simulación de pruebas: La simulación de entornos de pruebas está limitada por la disponibilidad de contenedores Docker base, pero cubre de forma suficiente las cargas de trabajo actuales.
- Servicio de pruebas bajo demanda: El sistema depende de un número mínimo de servicios externos y está disponible de forma continua.

- Servicio de monitorización y trazado: La unificación de la recolección de datos en una sola base de datos simplifica la monitorización y trazado de usos de sistema.

Para cubrir de forma más completa las capacidades TaaS arriba citadas, el sistema tal como se presenta puede ser mejorado significativamente en los siguientes campos:

- Incrementar la disponibilidad de variables para testeo de entornos externos.
- Incrementar el número de soluciones de pruebas estándar soportadas.

*El artículo completo estará disponible en las actas del *Congreso "The Digital Transformation: A Challenges for ICT Engineers"*.

An Agile Container-based Approach to TaaS

Current cloud deployment scenarios imply a need for fast testing of user oriented software in diverse, heterogeneous and often unknown hardware and network environments, making it difficult to ensure optimal or reproducible in-site testing. The current paper proposes the use of container based lightweight virtualization with a ready-to-run, just-intime deployment strategy in order to minimize time and resources needed for streamlined multicomponent prototyping in PaaS systems. To that end, we will study a specific case of use consisting of providing end users with pre-tested custom prepackaged and preconfigured software, guaranteeing the viability of the aforementioned custom software, the syntactical integrity of the provided deployment system, the availability of needed dependencies as well as the sanity check of the already deployed and running software. From an architectural standpoint, by using standard, common use deployment packages as Chef or Puppet hosted in parallellizable workloads over readyto-run Docker images, we can minimize the time required for full-deployment multicomponent systems testing and validation, as well as wrap the commonly provided features via a user-accessible RESTful API. The proposed infrastructure is currently available and freely accessible as part of the FIWARE EU initiative, and is open to third party collaboration and extension from a FOSS perspective.