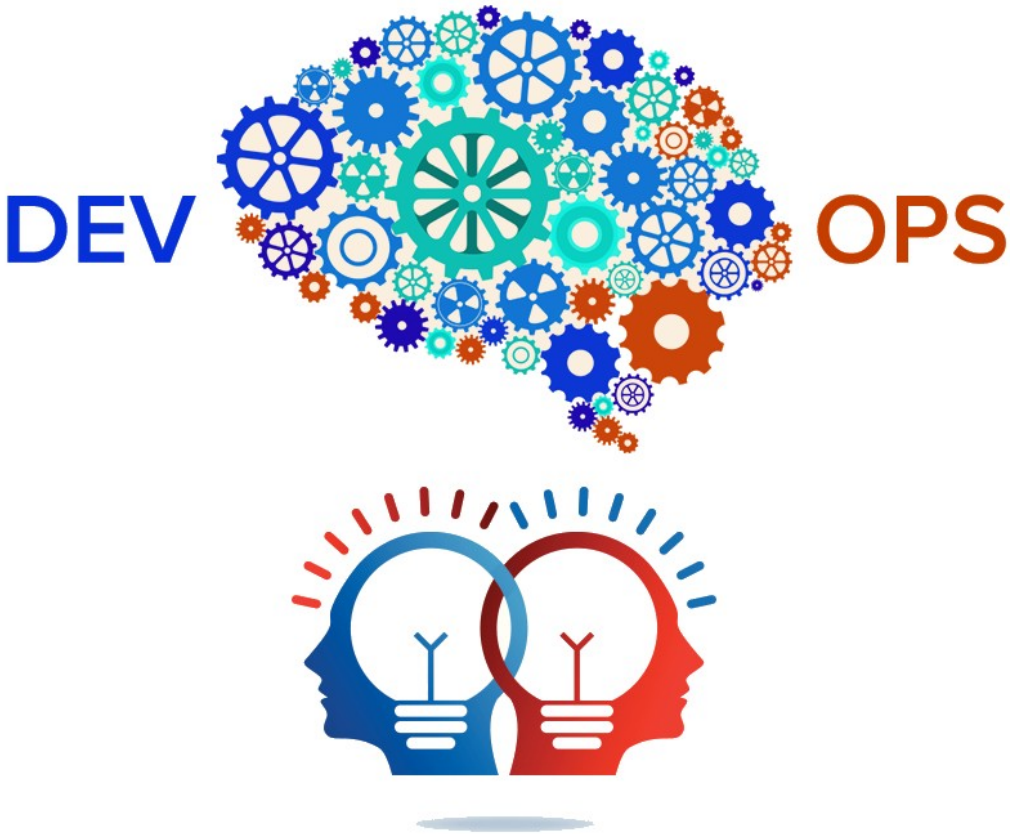
Implementando DevOps



Roberto Frutos Renedo
Octubre 2015

Índice

1. Acerca de mí

Me llamo Roberto Frutos Renedo. Nací un 20 de octubre de 1985 en Valladolid, España y ahora mismo estoy trabajando para ISBAN (Grupo Santander) en el área de Arquitectura Tecnología y Operaciones, en Madrid.

Mi andadura por Madrid comenzó siendo bastante joven, una vez que terminé la carrera de Ingeniería Técnica Informática. Desde ese momento, comencé a trabajar en un sector, el del software, que me apasiona, y que creo que tiene mucho que ofrecer a personas inquietas y entusiastas como yo.

Me entusiasma conocer y dominar cualquier tipo de tecnología, también motivado por la gran variedad de las mismas con las que me he tenido que enfrentar, unas veces por decisión propia, otras por las circunstancias del momento, lo cual me ha permitido acumular una gran experiencia a lo largo de estos años como profesional del mundo del software. Creo que esta experiencia, junto con mi carácter entusiasta por conocer y aprender, son las razones que me impulsan a continuar contando mi experiencia a la hora de diseñar e implementar la arquitectura de un gran sistema informático como es en este caso un ciclo de DevOps.

En cuanto a mis aficiones, todo lo relacionado con las dos ruedas. Cuando puedo trato de buscar un hueco para hacer MTB, una de mis pasiones desde que era bien pequeño y mi padre me compró mi primera bicicleta, desde ese momento no he parado. Además de la bicicleta, me encantan las motos, hasta hace un año practicaba moto-cross, pero por diversas razones lo fui abandonando y finalmente vendí la moto. Ahora estoy buscando una nueva, así que pronto retomaré esta excitante afición.

Por otra parte, creo que soy un poco adicto a los retos, y así como hasta el momento había dejado de lado el aprendizaje de otras lenguas, el año pasado decidí apuntarme a la Escuela Oficial de Idiomas, y aunque nunca haya sido una de las materias que me haya apasionado especialmente, conseguí superar el primer curso de intermedio (B 1.1), por lo que si todo va bien este año conseguiré el certificado B1.

Como con esto parece que aún tengo algo de tiempo libre (esto no lo piensa así mi pareja), estoy trabajando/estudiando con el objetivo de conseguir la certificación en ArchiMate 2.1 antes de retomar la Escuela Oficial de Idiomas.

Este es un extracto de cómo soy, mis intereses personales, aficiones y carácter.

2. Motivaciones

Durante todos mis años como profesional, he estado inmerso en varias etapas del proceso de desarrollo de software, siempre con un denominador común, la implementación de componentes complejos. Es por ello, que además de realizar todo el diseño de la arquitectura DevOps que os he venido relatando hasta el momento, he llevado un gran peso en cuanto a la implementación de la misma. Debido a ello, mientras escribía el anterior artículo, no paraba de darle vueltas a la manera de transmitir, aunque nada más sean unas pocas líneas (desde el punto de vista más técnico) que te sirvan de orientación en el camino de llevar a cabo DevOps.

En este punto y a parte en cuanto a DevOps, me gustaría terminar hablando de lo comentado anteriormente. Aspectos importantes de la configuración, posibles tareas a crear en la herramienta de integración continua, necesidades del API para el delivery de aplicaciones, acciones a realizar desde el API, entorno inicial basado en contenedores y otros temas interesantes estarán aquí presentes de un modo u otro.

Todo esto, con un objetivo claro, que puedas tener una base técnica para poder afrontar la creación de un ciclo DevOps. No creo necesario mencionar, que me basaré en los modelos de arquitectura en los que he venido trabajando hasta el momento.

Cualquiera que tenga acceso a este documento puede contactarme con el objetivo de transmitirme sus valoraciones, críticas o comentarios acerca del mismo en la siguiente dirección de correo electrónico: robfrut@yahoo.es

3. Una mirada atrás

No tengo intención de que esto sea una guía paso a paso, ni tampoco un tutorial completo mediante el que consigas implementar, sin apenas error, un proceso DevOps. Eso sería algo descabellado y sin sentido. Lo que sí intentaré es que veas la utilidad de los modelos de cara a una implementación posterior y cómo pueden servirte de guía para llevar a cabo el ciclo DevOps.

Si no has podido leer mis anteriores artículos o no has podido acceder a ellos por otro motivo, aquí te dejo un referencia a los mismos.

- Modelando una gran arquitectura empresarial
  https://drive.google.com/open?id=0BymCGWR0IjzkbVY1MFlpaF9paE0

- Modelando procesos de negocio
  https://drive.google.com/open?id=0BymCGWR0IjzkbVY1MFlpaF9paE0

Creo que al menos una lectura por encima es necesaria para poder comprender lo que más adelante voy a ir comentando.

4. Implementación de procesos ¿por donde empiezo?

Por situarnos un poco, vamos a ver las consideraciones y pautas más importantes que hemos seguido en nuestro equipo hasta completar un ciclo DevOps. Es importante recalcar “el estado del arte” de nuestro DevOps: el proceso de integración continua ha alcanzado un estado medianamente productivo y la entrega continua y aceptación del software aún están en modo “prueba de concepto”. Además, sólo estamos gestionando un único entorno de aplicación (servidor de desarrollo). No obstante, diversas demos que hemos venido realizando con éxito, nos han permitido que estemos en proceso de conseguir infraestructura “de verdad” para estos procesos.

A partir de este momento, comentaré los principales detalles de la automatización de estos procesos partiendo de los modelos BPMN que tenemos. Ya que esta no ha sido una labor que haya llevado a cabo solo, me gustaría nombrar a los integrantes de este equipo.

- Rubén de Dios Barbero
- Sergio Merino Merino
- Francisco Javier de la Morena

Imagina que en este momento te entregan los modelos de arquitectura y procesos de negocio que componen DevOps. Es vital tener claro qué procesos de alto nivel hay definidos y cuál es la delimitación de cada uno, es decir, dónde comienza y dónde termina cada etapa de DevOps y más importante aún qué cometido tiene cada una dentro del proceso global. Como has podido ver, los que he detallado son seis, tres de ellos, sin ninguna duda, automatizables por completo:

- Gestión del proyecto (especificaciones, tareas, documentación, criterios de aceptación...)
- Desarrollo de software (codificación de software y pruebas asociadas)
- **Integración continua** (desde la construcción del software hasta la publicación de artefactos en nexus, pasando por el versionado, ejecución de pruebas que incluya la aplicación...)
- **Entrega continua** (despliegue de la aplicación, notificación de entorno...)
- **Aceptación del software** (pruebas sobre el software, notificación de aceptación...)
- Soporte y Mantenimiento (notificación de incidencias, alertas...)

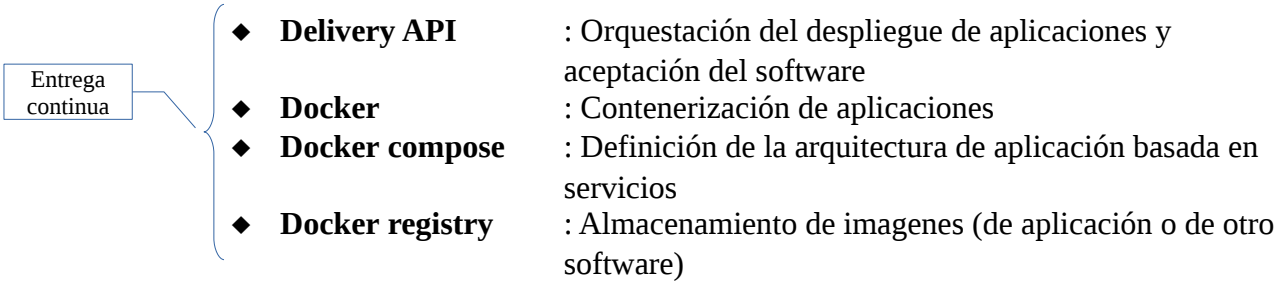
En los procesos que he modelado, he intentado remarcar dos bloques clave: uno con el nombre “Pipeline integración continua” y otro con el nombre “Delivery API”. La razón de esto, no es otra que intentar destacar los bloques que debemos automatizar.

4.1. Herramientas básicas

Estas son las pricipales herramientas que hemos empleado para implementar la parte automatizable en nuestro proceso de DevOps. No documentaré la configuración de cada una, pero si comentaré algunos aspectos destacados de las que considero más importantes.

Integración continua

- **GitLab** : Interfaz web para la gestión de Git
- **Git** : Repositorio de código fuente colaborativo
- **Jenkins** : Gestión de tareas (compilación, publicación...)
- **Nexus** : Repositorio de artefactos
- **Sonar** : Análisis estático de código



En este punto me gustaría hacer referencia a unos artículos muy interesantes, en los que un compañero mío explica la instalación de un entorno de integración continua paso a paso, centrándose en las herramientas y en la configuración de las mismas.

- Aprende a montar un entorno de integracion continua I
- Aprende a montar un entorno de integracion continua II - SonarQube
- Aprende a montar un entorno de integracion continua III - Nexus OSS
- Aprende a montar un entorno de integracion continua IV - Jenkins
- Aprende a montar un entorno de integracion continua V - Jobs

4.2. Jenkins

Mediante Jenkins, automatizamos el proceso de integración continua y las diferentes tareas que engloba, con todas las garantías que ofrece esta herramienta. No entraré a comentar si es mejor o peor que otras, o por qué se han descartado otras (Bamboo, Go CD...) y tampoco a detallar la configuración completa de nuestras tareas. Es una implementación particular nuestra y creo no te aportarían demasiado, ya que sin duda, el DevOps de tu organización tendrá sus particularidades y necesidades propias. Con esto me refiero a: frecuencia de ejecución de tareas, versionado, nomenclatura de artefactos, etc.

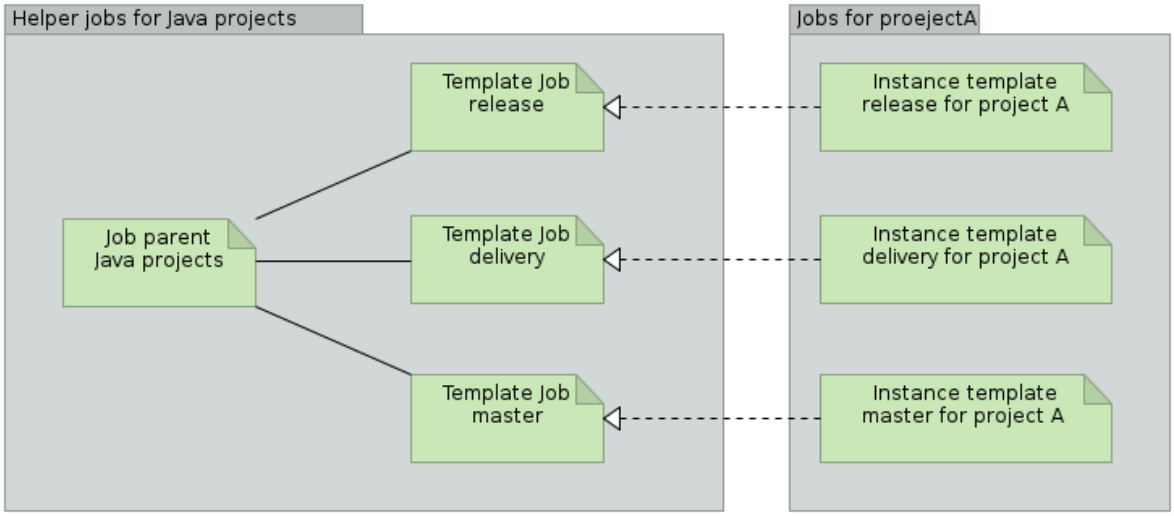
Por otra parte, en la red puedes encontrar gran cantidad de información y documentación de referencia relacionada con Jenkins: cómo crear tareas, construcción de software, integración con repositorios de artefactos, gestión de esclavos... Sí me gustaría indicar brevemente algunas de las razones de por qué hemos optado por esta herramienta:

- ✔ Es una herramienta open-source
- ✔ Amplia comunidad soportando el proyecto
- ✔ Gran variedad de plugins y extensiones
- ✔ Configuración realmente flexible
- ✔ Integración con Maven, Ant, Gradle, MSBuild, Grails, Ruby, Shell, batch...
- ✔ Numerosos sitemas de gestión de fuentes: AccuRev SCM, Bazaar, BitKeeper, Clearcase, CMVC, CVS, Git, Mercurial, Perforce, PVCS, SourceSafe, StarTeam, Subversion, TFS
- ✔ Conocimiento previo de la misma

En cuanto al uso de Jenkins, es importante la creación de tareas, partiendo de plantillas preparadas previamente para cada tipo de proyecto que debamos incluir en nuestro DevOps. Esto facilitará enormemente la configuración, proporcionando una base estable sobre la que ir perfilando

los detalles de cada proyecto en particular.

Este sería un esquema de alto nivel de la relación entre las plantillas y las tareas que se ejecutarían finalmente, tomando como ejemplo un proyecto de tipo Java que queremos desarrollar utilizando nuestro ciclo.



4.3. Delivery API

El “Delivery API”, inicia sus acciones en el momento en que nuestra integración continua ha terminado de construir nuestro software (aplicación y los diferentes componentes de pruebas). En este punto, debemos hacer el despliegue en el primer entorno que hemos definido y arrancar la fase de aceptación del software, proceso en el cual ejecutaremos las pruebas, reportaremos los resultados, pararemos la aplicación en el caso en que corresponda y comprobaremos si podría realizarse una promoción al siguiente entorno. En todas estas acciones, el API jugará un papel fundamental.

La versión que actualmente he terminado del API, es una versión inicial de un componente que ya comienzo a vislumbrar que debo evolucionar a otra tecnología bien sea Go (por su cercanía y buen rendiminento en tareas de este tipo) o Java (por versatilidad a la hora de cubrir nuevas funcionalidades que tenemos en mente).

Como mecanismo para la ejecución de pruebas en paralelo y ahorro de infraestructura, se decide investigar la linea de Docker, y cómo nos podría beneficiar en ese sentido. Además de lo comentado, esta tecnología será vital para alcanzar un circuito de DevOps suficientemente ágil, ya que con la paralelización de pruebas en contenedores conseguiremos reducir el tiempo en la fase de integración continua (ya que no lanzaremos las pruebas más costosas en ese momento), reduciremos la infraestructura necesaria para este proceso (ya que reduciremos la carga de trabajo en los nodos de Jenkins) y por tanto, reduciremos el tiempo total de cada ciclo DevOps.

Además de esto, es sumamente importante que tengas presente que el tiempo de levantar una aplicación que ejecute sobre contenedores, con respecto a una que esté en una máquina virtual o en un host directamente, es tremendamente inferior. Puedes investigar este aspecto y otros relacionados con Docker en la red. Si no conoces esta tecnología, creo que te sorprenderá gratamente.

Obviaré los detalles más técnicos y me centraré en lo que nos implicó de cara al proceso

DevOps.

- ✔ Añadir un fichero de definición de infraestructura en el software
- ✔ Ejecución de pruebas dentro de un contenedor Docker
- ✔ Creación de imagen Docker para la ejecución de pruebas
- ✔ Levantar varias instancias de una aplicación simultáneamente
- ✔ Registro de imágenes de aplicación privado
- ✔ Registro de imágenes base privado
- ✔ Control de versiones de las imágenes de aplicación que construimos
- ✔ Cierto rechazo a su implantación dentro de la organización

Estas nuevas consideraciones, junto con las necesidades que se han mostrado a través de los modelos de arquitectura, derivan en diferentes funcionalidades que fui integrando en el API con el fin de llegar a una versión inicial y poder tener una prueba de concepto que enseñar antes de que cortaran “el grifo” al proyecto.

Comentar las funciones más importantes del API, creo podrían servirte de guía si decides implementar por tu cuenta la entrega continua y aceptación del software dentro del circuito de DevOps.

```
./delivery-api.sh <id-process> <action> <project name> <option>

      Parameters execution
            (1)
                  <id-process>     : [id current process]
                  <action>         : [start|stop|test|clean]
                  <project name>   : [your name project]
            (2)
                  <action>    : [--help]
            (3)
                  <action>    : [install|remove]
            (4)
                  <id-process>      : [id current process]
                  <action>          : [check]
                  <project name>    : [your name project]
                  <option>          : [application|tests|save]"
```

Como has podido ver a lo largo de este apartado, nuestro API nos ofrece las funcionalidades apropiadas para poder abarcar el proceso de despliegue de aplicaciones y aceptación de software sin ningún tipo de problema.

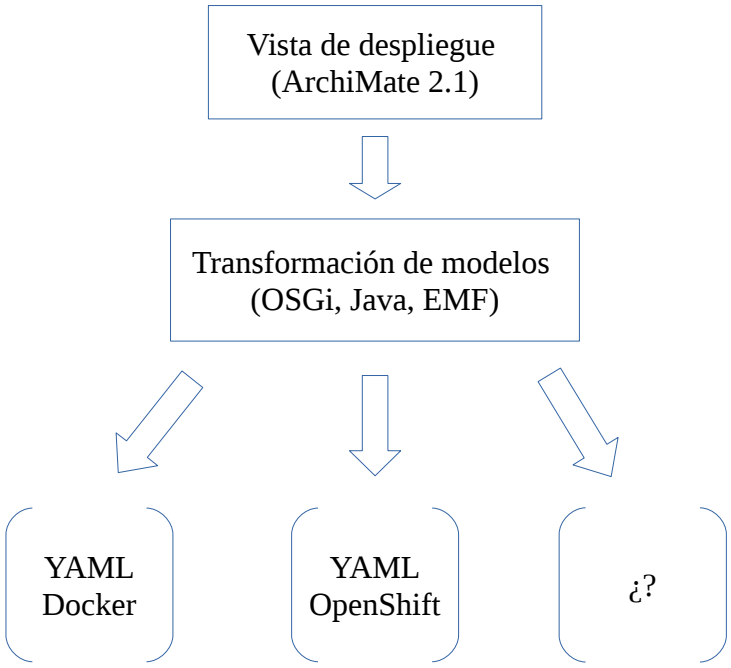
4.4. Infraestructura de aplicación, docker-compose

Qué decir que no esté ya escrito acerca de esta tecnología. En el momento en que vemos la necesidad de tener algo que defina la arquitectura de la aplicación dentro del software, pensamos en un fichero YAML más genérico donde se especificase la información necesaria para poder generar posteriormente el fichero específico de cada tecnología en particular.

Este trabajo, tanto la definición de qué debería contener este fichero como el componente

necesario para generar los ficheros necesarios en formato directamente ejecutable (Ej. formato OpenShift, formato docker-compose, formato CloudFoundry...) considero que es enormemente costoso para el reducido grupo de personas que éramos en el equipo. De este modo, se decide aparcar temporalmente y utilizar como primera aproximación el formato de docker-compose.

Aquí te muestro un esquema del objetivo a largo plazo a conseguir por nuestro equipo. Creo que explica claramente nuestro objetivo final.



4.5. Integración continua y entrega continua

De acuerdo, pero...estas dos fases, ¿cómo se integran? En nuestro caso, mediante la creación de una tarea en Jenkins que sea llamada al final de la integración continua, es decir, la última acción de ese pipeline. De este modo, lanzamos la parte de entrega continua y ejecución paralela de pruebas sobre infraestructura basada en Docker. Esta tarea encapsula las llamadas al API que consideremos oportunas, que en nuestro caso particular, son los distintos comandos que hemos desarrollado.

En este proceso de integración de fases (integración continua y entrega continua) nos encontramos con diferentes problemas a la hora de configurar Jenkins. En este sentido, debes prestar especial atención a:

- ✔ la sincronización de tareas
- ✔ la recepción de los informes en los esclavos
- ✔ la publicación de informes de las pruebas en el panel de Jenkins
- ✔ la transferencia de informes de pruebas entre esclavos
- ✔ la parametrización del job de entrega continua (delivery)
- ✔ la ejecución de tareas en diferentes esclavos de Jenkins

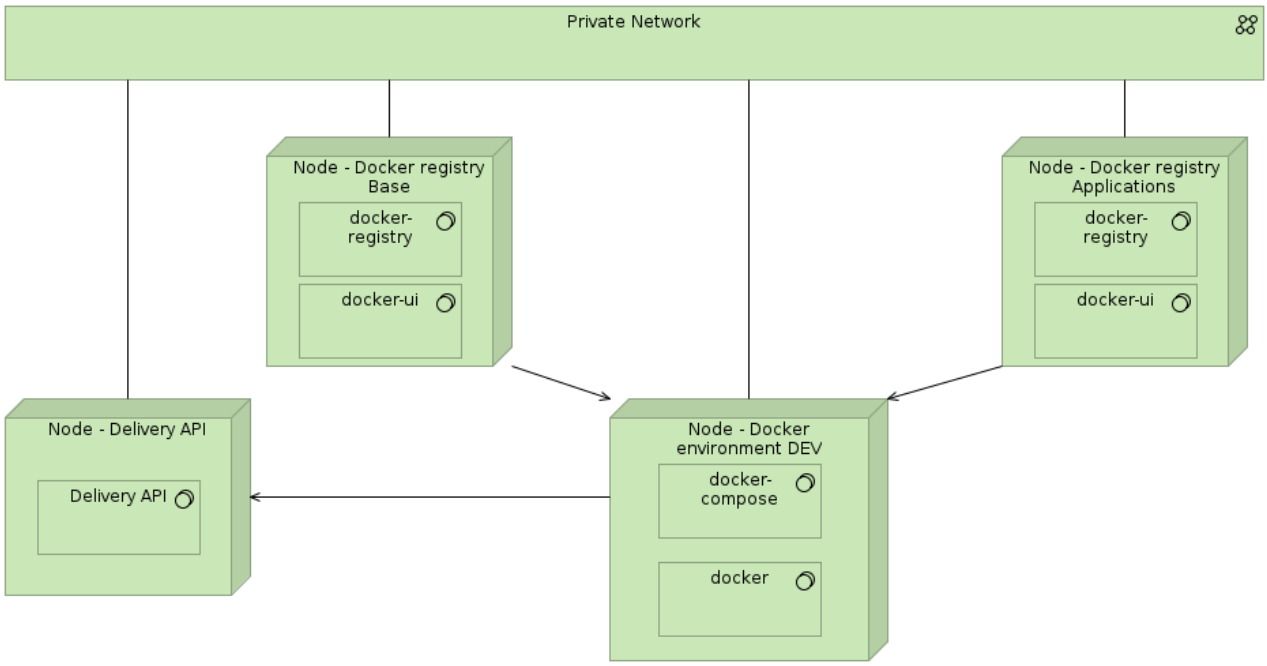
Este apartado daría para un artículo nuevo, pero creo que ya he mencionado las consideraciones más importantes al respecto, y espero te sean de utilidad si te embarcas en un proyecto similar a este.

5. Infraestructura inicial

Esta es la primera aproximación a un entorno de desarrollo basado en Docker hecha por nuestro equipo. Se trata de una arquitectura con cuatro nodos, en la que cada uno se destina a una funcionalidad diferente:

- **Nodo – Delivery API**. Donde instalaremos el API y será el punto de acceso de la integración continua con la entrega continua.
- **Nodo – Docker environment DEV**. Nodo de simulación de un servidor de desarrollo. Necesitaremos tener corriendo Docker y un HA-Proxy para el balenceo de aplicaciones.
- **Nodo – Docker registry base**. Nodo donde alojaremos un registro de docker con imágenes de diferentes tipos de software (Tomcat, MySQL...)
- **Nodo – Docker registry applications**. Nodo donde instalaremos un registro docker para almacenar la imagen de cada aplicación particular, imagen que utilizaremos para ejecutar la aplicación en cada entorno que hayamos definido.

Estos nodos se encuentran alojados en una red privada, es decir, sin acceso público y por tanto bajo las medidas de seguridad proporcionadas por nuestra compañía. Aquí te dejo una vista de la infraestructura, cuya notación creo que te resultará familiar.



Como puedes apreciar es una infraestructura extremadamente simple. Nuestro objetivo no era otro que tener un acercamiento a una tecnología como Docker (en cremiento exponencial) y a su vez completar un ciclo DevOps con despligue de aplicaciones sobre una infraestructura basada en contenedores Docker.

Si tienes cierto conocimiento de las arquitecturas Cloud, has utilizado algún SaaS o algún PaaS, podrás entender que esta infraestructura es válida para una prueba de concepto y no puede ser un entorno real para la ejecución de aplicaciones.

De este modo, para ser trasladado a un entorno real, estamos trabajando en incorporar otros

componentes clave que solventen algunas de las principales problemáticas que se nos están planteando, como pueden ser la gestión de la configuración, el descubrimiento de servicios o la orquestación de contenedores. En esta línea estamos trabajando sobre:

- **Consul** como servidor de configuración y descubrimiento de servicios.

- **Kubernetes** como orquestador de contenedores.

6. Conclusión final

Llegados a este punto, destacar la importancia del proceso automatizado por encima de las herramientas que empleemos para implementarlo. Aunque utilices estas herramientas u otras, la clave de DevOps es la automatización del ciclo y la eliminación cualquier tipo de acción manual en el proceso. Como consecuencia de esto, subyace que es clave tener totalmente claros los pasos de nuestro proceso DevOps antes de automatizarlo. Indudablemente, en función de las herramientas y tecnologías que decidas emplear para la automatización del mismo, encontrarás unos problemas u otros a solventar en el camino.

Destacar, que el uso de contenedores Docker en DevOps es otra pieza clave que nos permitirá conseguir unos tiempos de ciclo reducidos, y como consecuencia, desplegar aplicaciones o actualizaciones de las mismas con una frecuencia bastante elevada. Estos tiempos de ciclo se antojan imposibles en entornos aprovisionados con máquinas virtuales. Además de este beneficio, reduciremos la infraestructura de ejecución, necesitaremos menos infraestructura para soportar nuestro ciclo DevOps, tendremos un mejor aislamiento de aplicaciones, podremos garantizar que el software es el mismo en cada entorno y otras muchas ventajas derivadas de su adopción.

He de agradeceros que hayáis terminado de leer el artículo, espero que haya sido de vuestro agrado y, que desde un punto de vista más técnico, tengáis una visión más nítida de cómo funciona un proceso de DevOps completamente automatizado.

Por mi parte, confesar que ha sido un reto difícil enfrentarme al hecho de intentar comunicar diferentes aspectos complejos de un proceso DevOps, desde la definición hasta la implementación del mismo, no obstante, creo estar satisfecho con el resultado. Espero haber sido suficientemente claro en los aspectos que considero más relevantes en DevOps y que no hayas sentido más confusión que claridad.

Por último, no puedo terminar este artículo sin dar las gracias a la persona con la que he estado trabajando codo con codo los últimos años, Ruben De Dios Barbero. De él, he aprendido cosas que no aparecen en este artículo, ni en los anteriores, pero que para mí albergan una importancia vital, ya que me han permitido crecer enormemente tanto a nivel personal como profesional. Gracias.