¿Por qué tener las imágenes Docker cacheadas en Cloud Foundry? (Cacheo de imágenes docker)

Teniéndolas cacheadas en Cloud Foundry, le ahorramos tiempo al desarrollador cada vez que éste quiera ejecutar en la nube Cloud Foundry la aplicación de la imagen Docker, desplegada en un contenedor(contenizada).

Es decir, de no haber caché, cada vez que se quiera ejecutar en la nube Cloud Foundry la aplicación del desarrollador, Cloud Foundry tendría que llevar a cabo el proceso de autentificarse en el registro Docker(si es privado) donde se haya la imagen Docker que contiene la aplicación para posteriormente traérsela a la nube Cloud Foundry y ponerla en funcionamiento en tiempo de ejecución a través de su despliegue en un contenedor(contenización), que se lleva a cabo por el backend "Garden-runC" de Diego que es el que crea los contenedores.

Sin embargo gracias a la caché, concretamente la Diego-Docker-Caché, se trae la imagen una primera vez a Cloud Foundry desde el registro Docker donde se haya la imagen y se cachea, para que las posteriores veces que haga falta se acceda directamente a ella desde la caché.

Las imágenes Docker se cachean subdividiéndolas en capas. Cuando el backend garden quiera recuperar la imagen, sólo necesita recuperar todas las capas de la caché y luego usar las bibliotecas de Docker para unirlas y montarlas como un sistema de archivos raíz.

Otra ventaja de esto es que garantiza la disponibilidad de las imágenes sin depender de la disponibilidad de DockerHub o cualquier otro registro/repositorio de imágenes Docker donde se hallase inicialmente la imagen Docker.

Si éste está caído/inaccesible o cualquier otro problema de disponibilidad que pudiese surgir, siempre se puede recuperar de la caché de Diego-Docker.

También se garantiza mayor facilidad de escalado para nuestras aplicaciones, ya que al tenerlas cacheadas no tenemos que depender del registro docker para poder instanciarlas y llevar a cabo un escalado, tan solo hace falta recuperarla de la cache e instanciarlas.

Diego será el encargado de indicarle a Garden que la imagen a extraer es la que se halla en caché en vez de la que está en el registro remoto. Esto tiene la ventaja adicional de asegurarse de que siempre esté ejecutando exactamente la imagen Docker que el desarrollador montó, en lugar de algo que puede haber cambiado en el registro remoto.

¿A qué buscamos dar respuestas?

- ¿Su aplicación no se inició porque Docker Hub no estaba disponible?
- ¿Alguien cambió una imagen de la que dependes y estropeó tu aplicación?
- ¿Cada nueva instancia de su aplicación Docker tarda años en descargarse?

Caché de recursos y caché de buildpacks (Cacheo de archivos de aplicación y cacheo de paquetes de compilación)

Sabemos que el Cloud Controller (CC) contiene un BLOBSTORE, esto es un almacén para objetos binarios de gran tamaño.

Aquí se guardan droplets, imágenes conteneizadas, buildpacks (paquetes de compilación), resource files(archivos de recursos) y "Apps Code Packages"(Páquetes de código de aplicación).

Cuando los archivos de recursos (application files -> archivos de aplicaciones), son cargados al Cloud Controller, justamente después son cacheados en la BLOBSTORE haciendo uso del algoritmo de hasheo "SHA" (HASH -> archivo), de esta manera se garantiza la reusabilidad de dichos archivos sin necesidad de tener que volver a subirlos al controlador.

El funcionamiento de la caché de recursos es el siguiente: antes de cargar todos los archivos de la aplicación, el CLI de Cloud Foundry emite un archivo de solicitud de coincidencias de recursos al Cloud Controller para determinar si alguno de los archivos de la aplicación ya existe en la caché de recursos (blobstore) del Cloud Controller.

Cuando se cargan los archivos/ficheros de la aplicación, la CLI de Cloud Foundry omite los archivos que existen en el caché de recursos proporcionando el resultado de la solicitud de coincidencia de recursos.

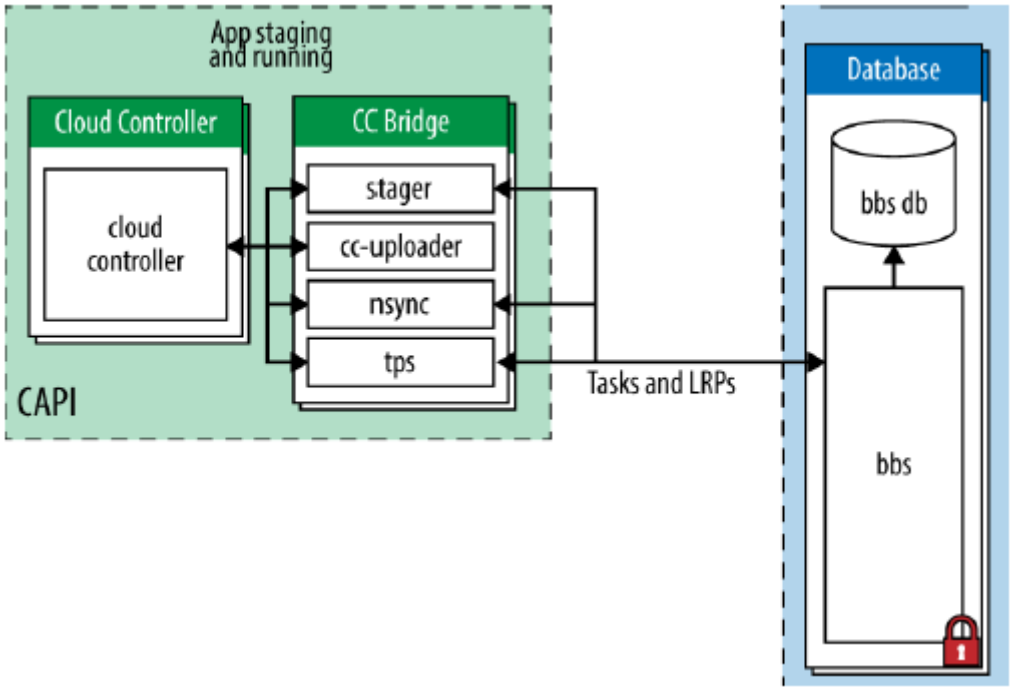
Archivos grandes de paquetes de aplicaciones que Cloud Controller almacena con un SHA para su reutilización posterior. Para ahorrar ancho de banda, la Interfaz de línea de comandos de Cloud Foundry (cf CLI) solo carga archivos de aplicaciones grandes que el Cloud Controller aún no ha almacenado en la caché de recursos.

Antes de cargar todos los archivos de origen de la aplicación, la CLI de Cloud Foundry emite una solicitud de coincidencia de recursos al controlador de la nube para determinar si alguno de los archivos de la aplicación ya existe en el caché de recursos. Cuando se cargan los archivos de la aplicación, la CLI de Cloud Foundry omite los archivos que existen en el caché de recursos al proporcionar el resultado de la solicitud de coincidencia de recursos. Los archivos de aplicación cargados se combinan con los archivos del caché de recursos para crear el paquete de la aplicación.

Los ficheros de aplicación cargados se combinan con el fichero de la caché de recursos para crear el paquete completo de la aplicación.

En la blobstore también se cachean buildpacks, concretamente archivos grandes que los paquetes de compilación generan durante la preparación, almacenados para su posterior reutilización. Este caché permite que los paquetes de compilación se ejecuten más rápidamente cuando se almacenan aplicaciones que se han almacenado previamente.

Caché en la BBS (Cacheo de DesiredLRP's y cacheo de ActualLRP's)



LRP: Long-Range Process /Procesos de larga duración

Task*:* Tareas

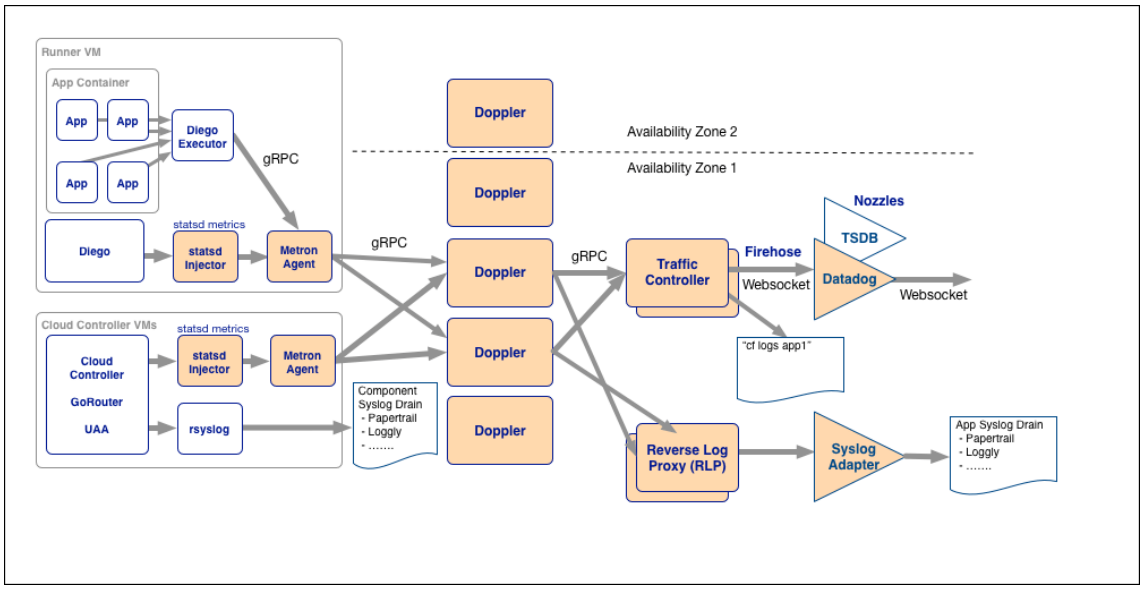
El BBS (Bulletin Board System/Sistema de tablón de anuncios) maneja la base de datos de Diego, manteniendo una caché actualizada del estado en tiempo real del cluster de Diego, incluyendo una representación del momento, de todos los DesiredLRPs (Los procesos de larga duración deseados), las instancias ActualLRP (Los procesos de larga duración que están actualmente en ejecución) que se están ejecutando, y las Tareas a bordo.

BBS está constantemente monitoreando si las LRP actuales son iguales que las LRPs deseadas. Cuando las actuales son menores que las deseadas, existe una deficiencia y por tanto BBS se pone en funcionamiento llevando a cabo una nueva subasta de tareas y LRPs. Cuando existe un excedente, mata instancias de tareas o LRPs.

En cualquier caso, BBS tiene que estar constantemente monitoreando estos números, de ahí la importancia de que estén cacheadas para poder consultarse más rápidamente y saber si hay una deficiencia o un excedente.

El Cloud Controller se comunica con el BBS a través del CC-Bridge.

Caché de logs (registros) y cacheo de métricas en la arquitectura loggregator y sus dependencias



La arquitectura loggregator es un subsistema de la implementación Cloud Foundry que se encarga de recopilar logs y métricas de aplicaciones de usuario que estén corriendo en una nube de Cloud Foundry. También registra logs del funcionamiento de los componentes de la implementación de Cloud Foundry.

Los logs y métricas son generados por los agentes Metrón, que están colocados en los VM's contenedoras de aplicaciones en funcionamiento

Los logs son reportes de eventos detectados, eventos que requieren tomar alguna medida, errores que surgen y cualquier otro mensaje que el desarrollador ha querido que se generase por alguna razón intrínseca a éste.

Las métricas son mediciones que se hacen sobre VM's contenedoras de aplicaciones, así como el estado de dichos contenedores (Docker y esas mierdas).

Toda estos logs y métricas se mandan a los servidores dropplers que contienen la cache, se cachean los logs y tras esto se envía al controlador de tráfico.

El controlador de tráfico recibe los logs de todos los dropplers y maneja las solicitudes de logs llevadas a cabo por el cliente interesado en dichos logs. Proporciona una API externa.

Luego la manguera que streamea (flujo) dichos logs y métricas provenientes del controlador de tráfico.

Los datos del flujo de la manguera no son constantes, van pasando y se van borrando, de ahí la importancia de la caché de logs ubicada en los servidores dropplers y que almacena los datos provenientes del loggregator. Dicha caché permite ver/consultar los datos desde la manguera y durante un periodo de tiempo. La cache proporciona una interfaz RESTful para recuperar los logs.

Todos estos datos pueden ser de gran utilidad, y es interesante el hecho de que puedan ser analizados por algún servicio que se encargue de ello.

La caché de registro está situada en la máquina virtual Doppler. Las dependencias de dicha caché son las siguientes:

- La caché de logs depende de **Loggregator** para ver y filtrar los logs, por lo que su fiabilidad depende del rendimiento de Loggregator. La caché de logs utiliza la memoria disponible en un dispositivo para almacenar logs y, por lo tanto, puede afectar al rendimiento del dispositivo durante períodos de alta contención de memoria.
- La caché de logs está ubicada en las **máquinas virtuales Dopplers.** Acelera la recuperación de datos del sistema Loggregator, especialmente para implementaciones Cloud Foundry que cuentan con un gran número de Dopplers.

Escalar la caché de logs (registros) ESTA MIERDA TAMBIÉN SE PUEDE PONER EN SCALING CLOUD FOUNDRY

Tal y como hemos mencionado anteriormente, se consigue una aceleración en la recuperación de logs en aquellos despliegues Cloud Foundry que cuentan con muchas VM's Dopplers, ergo escalar la caché de logs consiste en incrementar el número de máquinas virtuales Dopplers, de esta manera aceleramos la recuperación de logs.

REFERENCIAS

https://docs.cloudfoundry.org/concepts/diego/diego-architecture.html

https://docs.cloudfoundry.org/concepts/architecture/cloud-controller.html

https://docs.cloudfoundry.org/concepts/how-applications-are-staged.html **(Punto 4, para cacheo de archivos de aplicación y paquetes de compilación[buildpacks])**

https://assets.dynatrace.com/en/docs/report/cloud-foundry-definitive-guide.pdf **(Páginas 34, 70, 87)**

https://www.altoros.com/blog/how-to-push-private-docker-images-and-enable-caching-on-cloud-foundry-diego/ **(Cacheo de imágenes docker)**

https://docs.cloudfoundry.org/loggregator/architecture.html **(Cacheo de logs y métricas)**