Cloud Foundry: PaaS

Juan Arteaga Carmona
Rafael Domínguez González
Aurora Gutiérrez Martínez
Sergio León Riego
Francisco Pardillo Castillo

CLOUDFOUNDRY

Índice

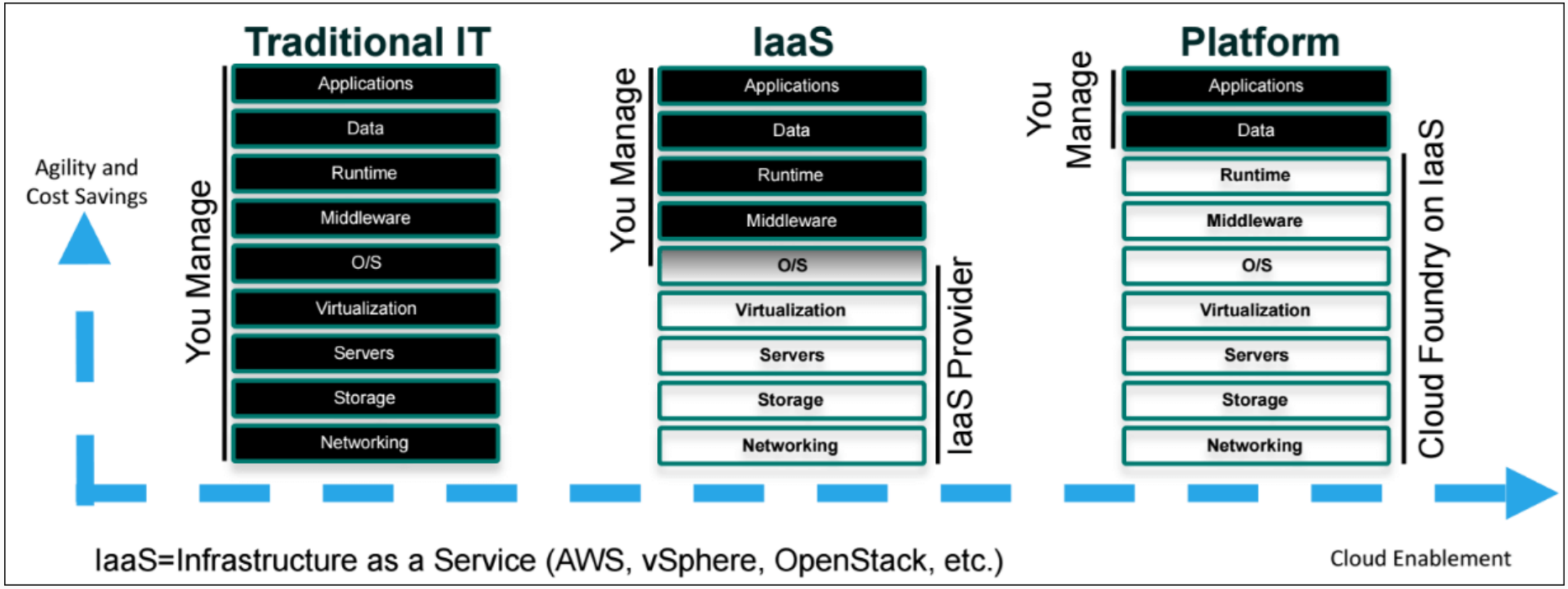
¿Qué es Cloud Foundry?

- Plataforma de desarrollo cloud-native
- Servicio PaaS
- Aísla configuración física del software

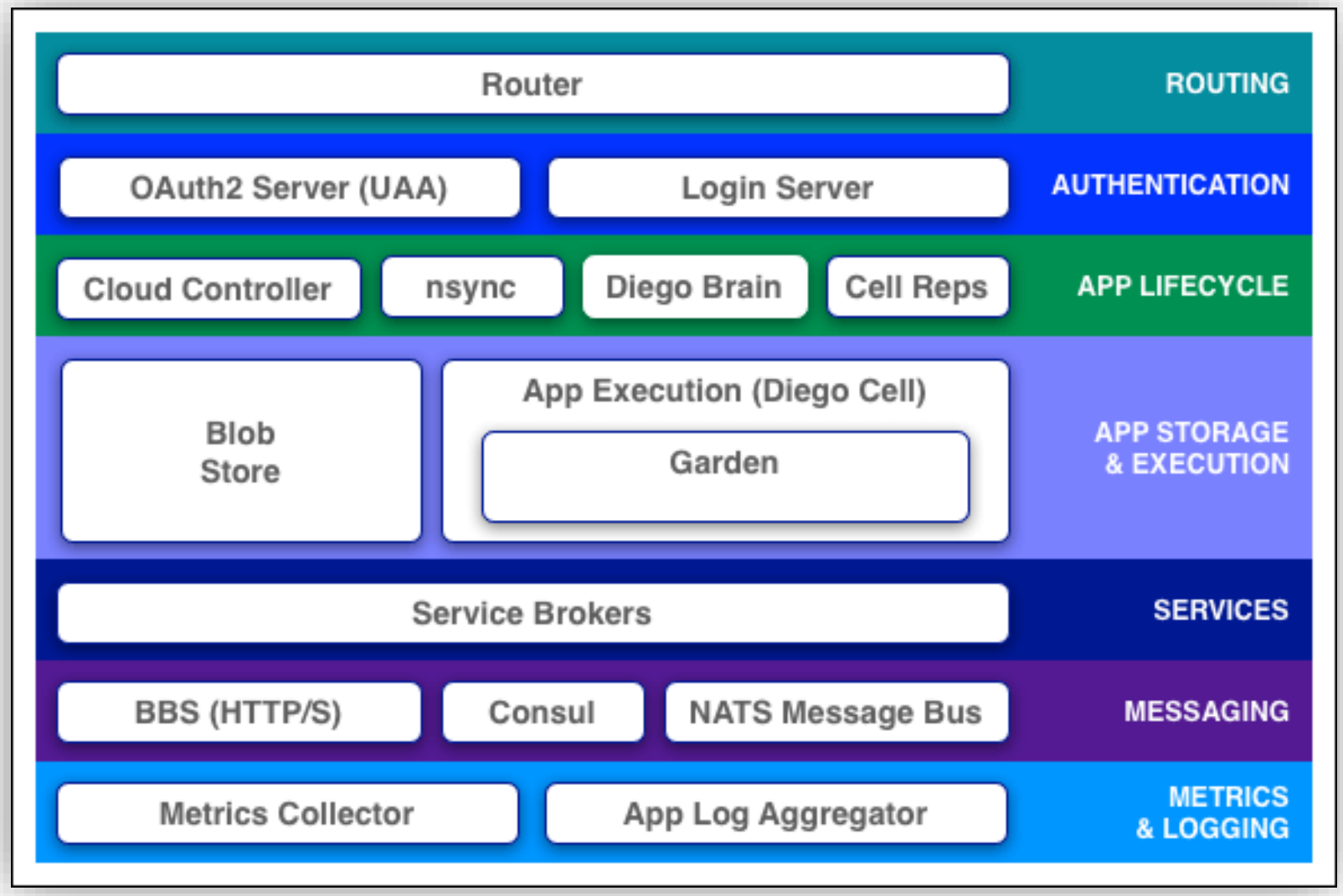
¿Qué ventajas tiene?

- Fácil escalabilidad
- Fiabilidad del servicio
- Balanceo de carga sencillo
- Gestión más simple de los recursos

Introducción

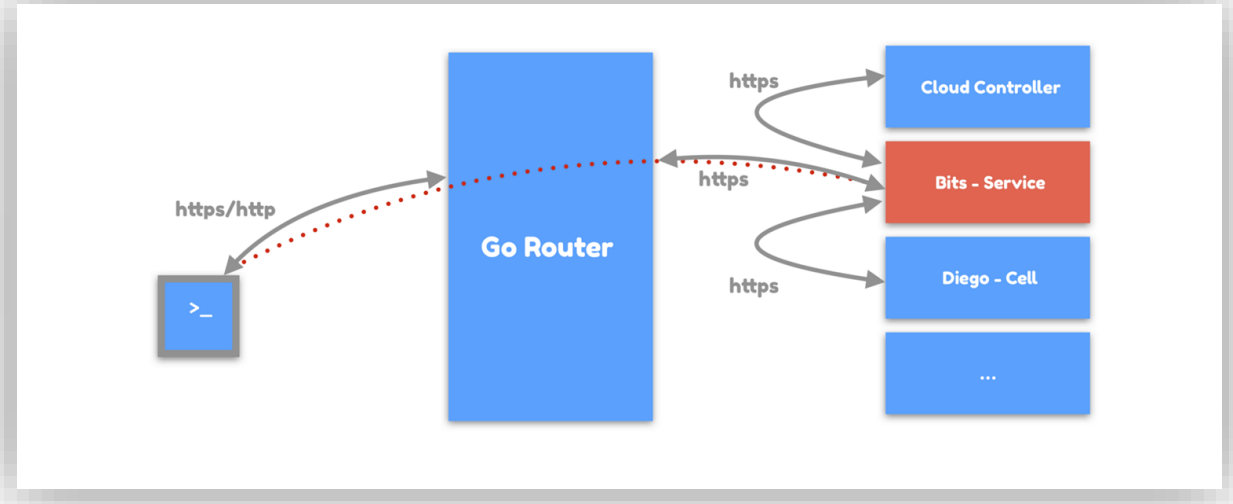


Arquitectura 


Router
ROUTING
OAuth2 Server (UAA)
Login Server
AUTHENTICATION
Cloud Controller
nsync
Diego Brain
Cell Reps
APP LIFECYCLE
Blob Store
App Execution (Diego Cell)
Garden
APP STORAGE & EXECUTION
Service Brokers
SERVICES
BBS (HTTP/S)
Consul
NATS Message Bus
MESSAGING
Metrics Collector
App Log Aggregator
METRICS & LOGGING

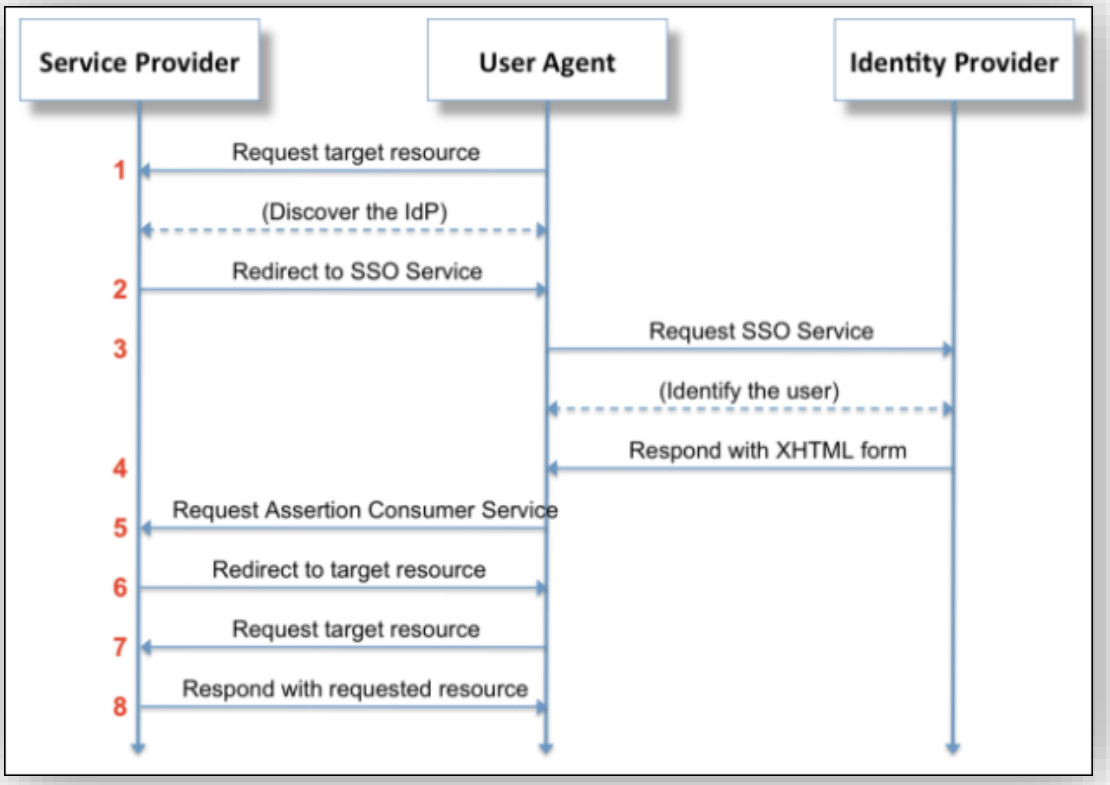
Gorouter:

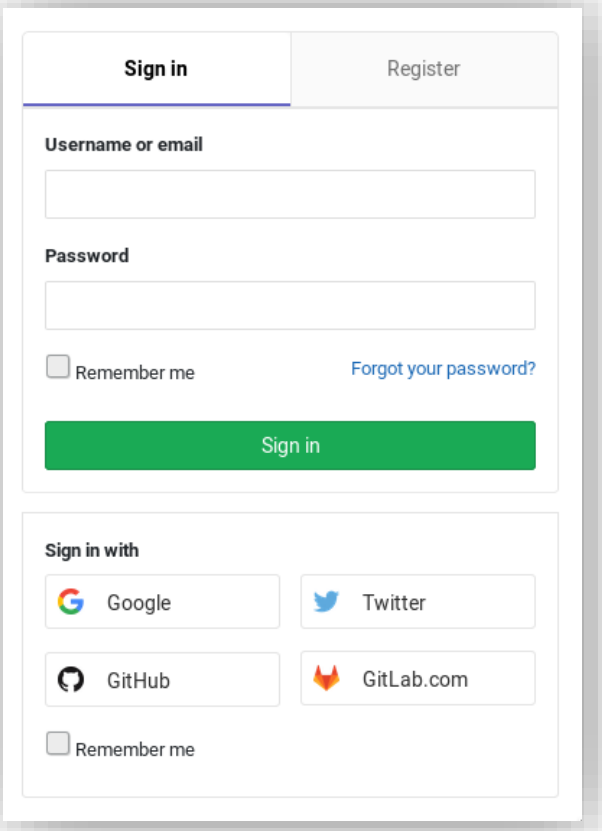
- Software
- Encargado de enrutar el tráfico entrante
- Implementado en Go
- Actualización de los enrutamientos en 2 min



Autentificación:

- UAA (OAuth server)
- Servidor de inicio de sesión de CF





Cloud Controller y Diego Brain:

- CC
  - Endpoint al que se conectan los desarrolladores
  - Necesita de una DB Postgres/MySQL

- Diego Brain
  - Encargado de poner en funcionamiento las VMs
  - Intenta mantener las especificaciones del desarrolla

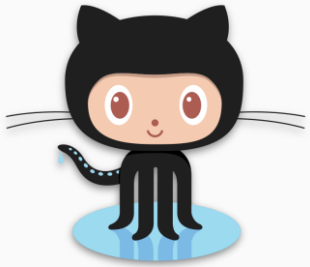
nsync, BSS, Cell Reps:

Nota:
LRP - Long Running Processes.
Instancias de la app que quiere ejecutar el desarrollador

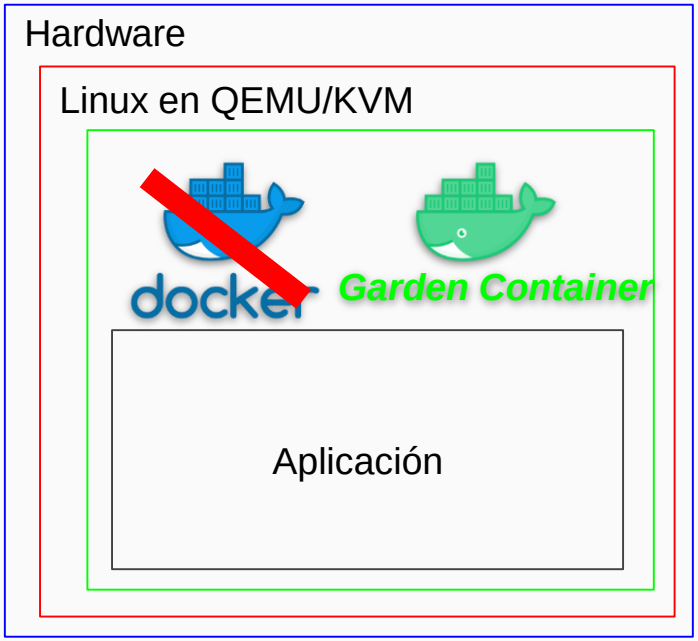
- nsync
  - Encargado de modificar ‘Desired LRP’ en la BSS
- Cell Rep
  - Encargado de actualizar ‘ActualLRP’
- BSS
  - Base de datos de Diego
  - ‘DesiredLRP’ = ‘ActualLRP’

Blobstore y Diego Cell:

- Blobstore
  - Repositorio de binarios grandes
  - Archivos que github no puede guardar eficientemente.



- Diego Cell
  - Máquinas virtuales en las que se ejecutan los contendores Garden



Loggregator y Recolector de métricas:

- Loggregator
  - “Log” + “Aggregator”
  - Se encarga de recoger y entregar los logs de la app a los desarrolladores.

- Recolector de métricas
  - Recoge las métricas de los componentes de CF para monitorizar y aplicar mejora continua a la infraestructura.

Distribución

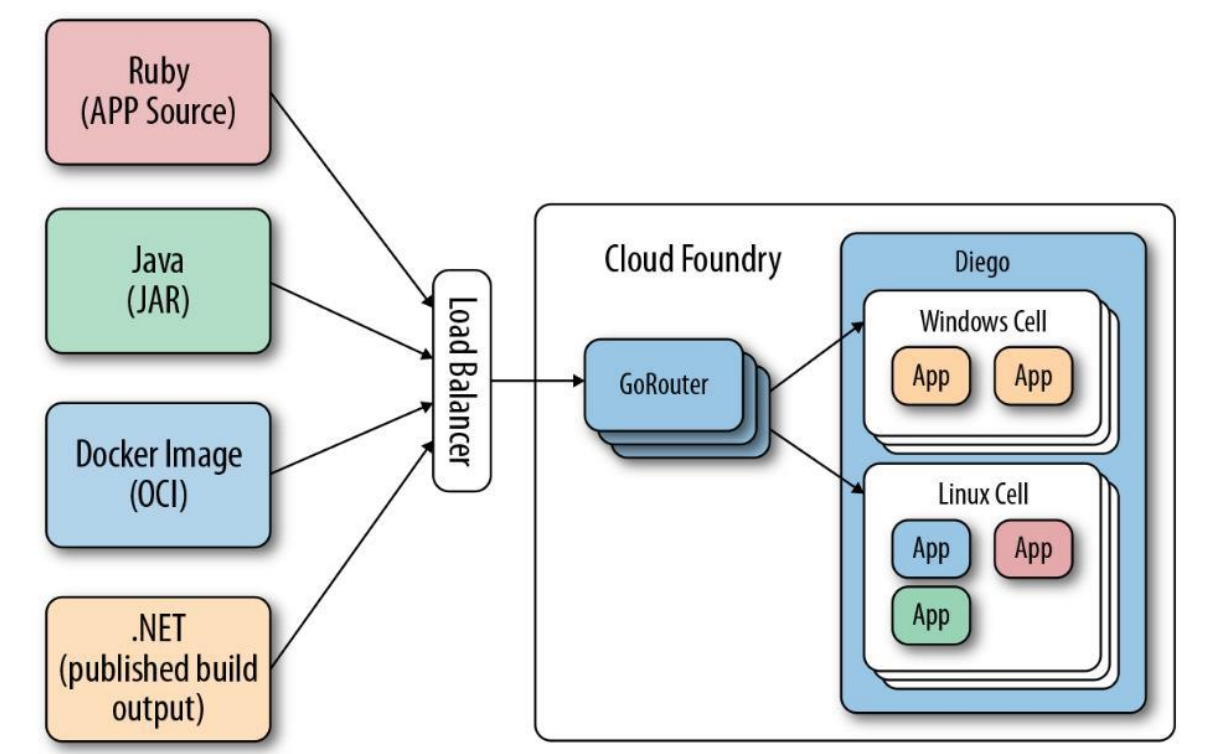
Diego: sistema distribuido que permite ejecutar y escalar aplicaciones y tareas en contenedores.

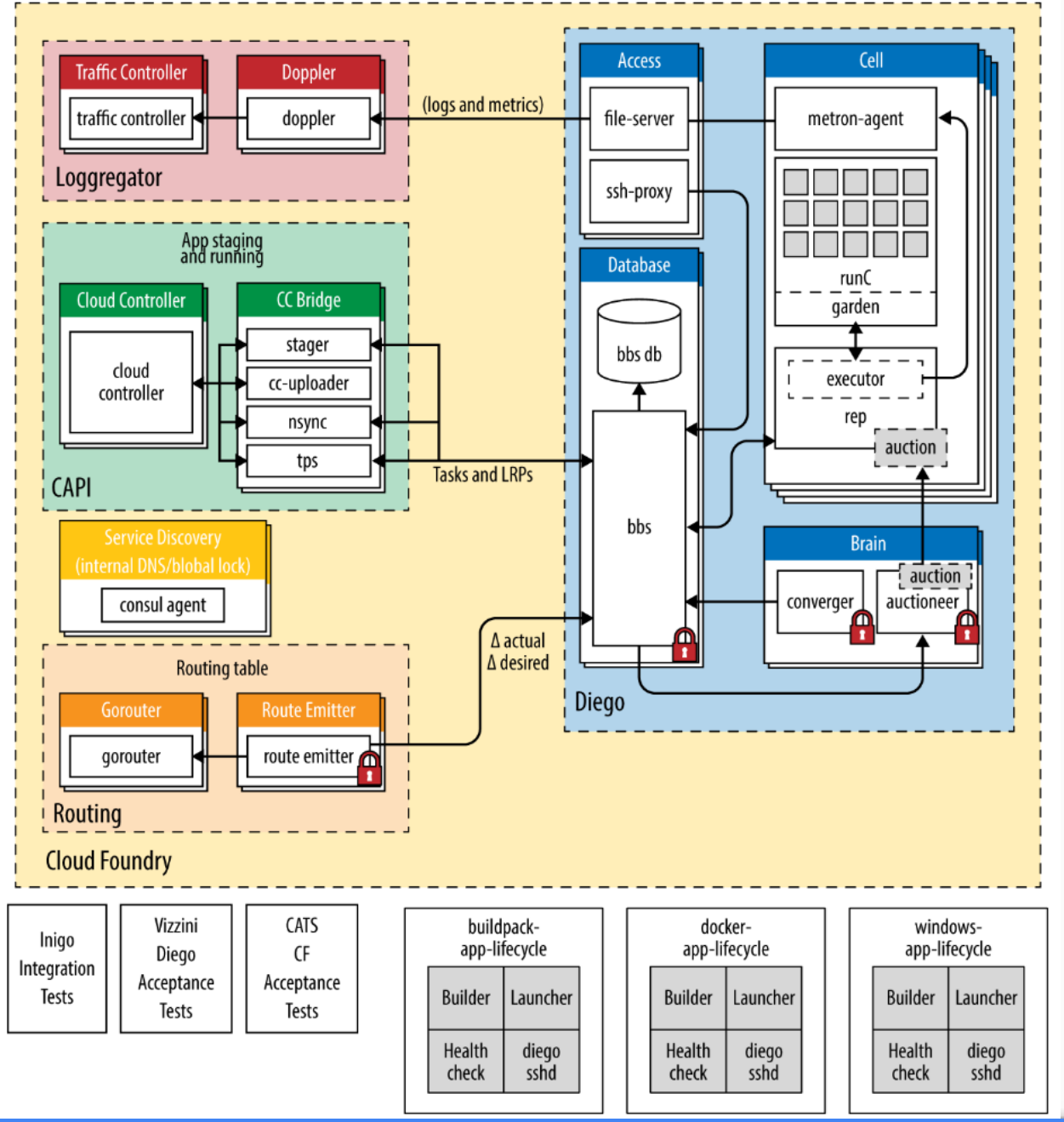
- ➔ Programa, ejecuta y supervisa tareas y aplicaciones.
- ➔ Asegura que las aplicaciones sigan ejecutándose al conciliar el estado deseado con el estado real.
- ➔ Entorno de ejecución genérico que permite el soporte de múltiples cargas de trabajo basadas en Windows y Linux.

Cloud Foundry ⟶ Garden

A través de Garden, Diego puede:

- Admitir cualquier formato de imagen de contenedor.
- Agregar soporte para ejecutar contenedores en cualquier tecnología de contenedor basada en Garden, incluidos los backends de contenedor basados en Linux y Windows.



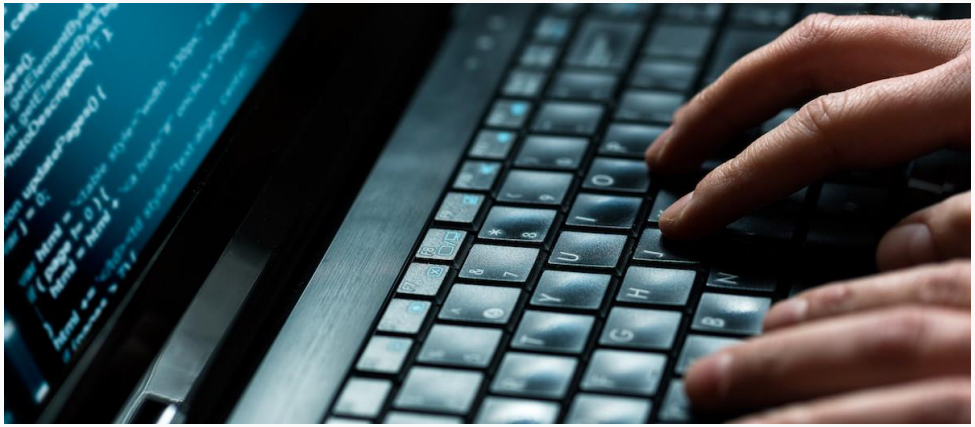


Distribución: arquitectura de Diego

➔ Los usuarios de Cloud Foundry no interactúan con Diego.

➔ Componentes orientados al usuario de Cloud Foundry:
- Componentes CAPI: la API de Cloud Foundry.
- El sistema de registro definido por los agentes Loggregator y Metron.
- Enrutamiento (GoRouter, TCPRouter y el Emisor de ruta).

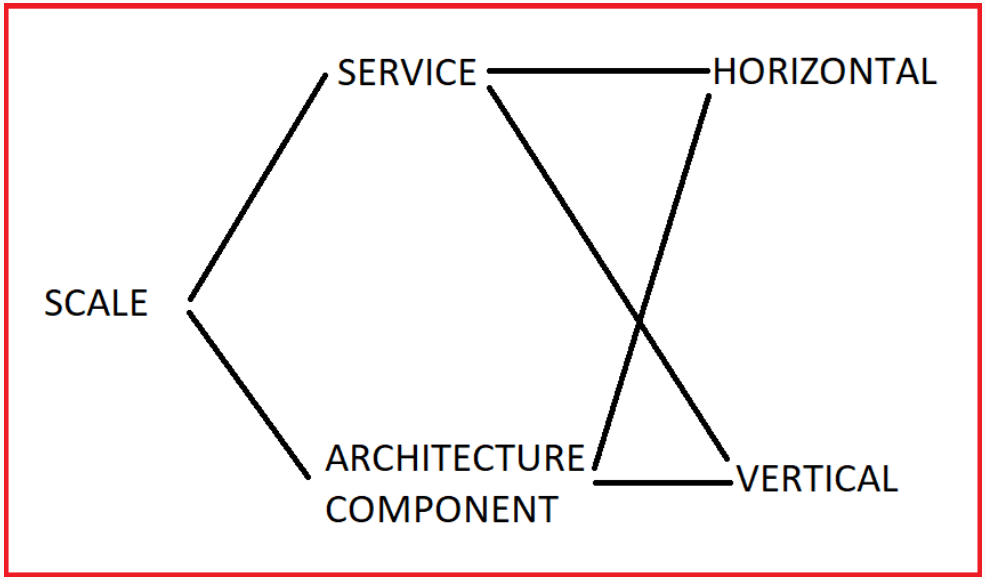
➔ Son responsables:
- Política de aplicación.
- Cargar artefactos y metadatos de la aplicación en un blobstore.
- Tráfico de enrutamiento y manejo de tráfico de aplicaciones.
- Gestión de usuarios.



ESCALADO

AL IGUAL QUE LO VISTO EN LA TEORÍA DE LA ASIGNATURA, EN CLOUD FOUNDRY CONTAREMOS CON LOS 2 TIPOS DE ESCALADO HABITUALES:

- **HORIZONTAL**
- **VERTICAL**



AMBOS APLICABLES TANTO A LOS SERVICIOS DESPLEGADOS EN CF, COMO A LOS COMPONENTES QUE CONFORMAN LA ARQUITECTURA

ESCALADO

¿QUÉ BUSCAMOS?

- GARANTIZAR UNA ALTA DISPONIBILIDAD DE NUESTRAS APLICACIONES DESPLEGADAS EN LA NUBE CLOUD FOUNDRY, ASÍ COMO UN AUMENTO DEL RENDIMIENTO EN EL SERVICIO

¿CÓMO LO HAREMOS?

- HORIZONTALMENTE: CREAREMOS NUEVAS INSTANCIAS DE NUESTRO SERVICIO DESPLEGADO EN LA NUBE

- VERTICALMENTE: AUMENTAREMOS EL RENDIMIENTO DE CADA INSTANCIA DOTÁNDOLA DE MEJORES PRESTACIONES

ESCALADO

ESCALADO HORIZONTAL

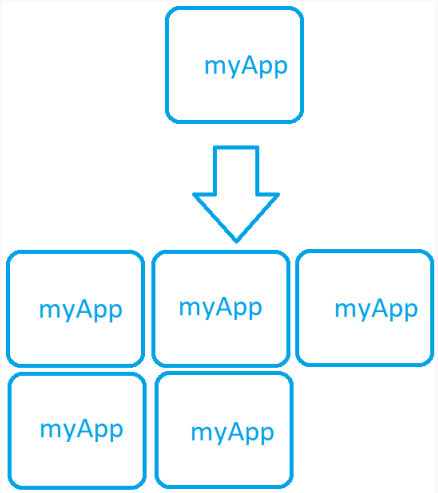
- HORIZONTAL POSITIVO/HACIA ARRIBA: CREAR NUEVAS INSTANCIAS.

- HORIZONTAL NEGATIVO/HACIA ABAJO: DESTRUIR ALGUNAS DE LAS INSTANCIAS YA CREADAS.

COMANDO UTILIZADO

```
$ cf scale APP -i INSTANCES
```

EJEMPLO: $ cf scale myApp -i 5



ESCALADO

ESCALADO VERTICAL

¿QUÉ PRESTACIONES DE LA INSTANCIA SE PUEDEN MEJORAR?

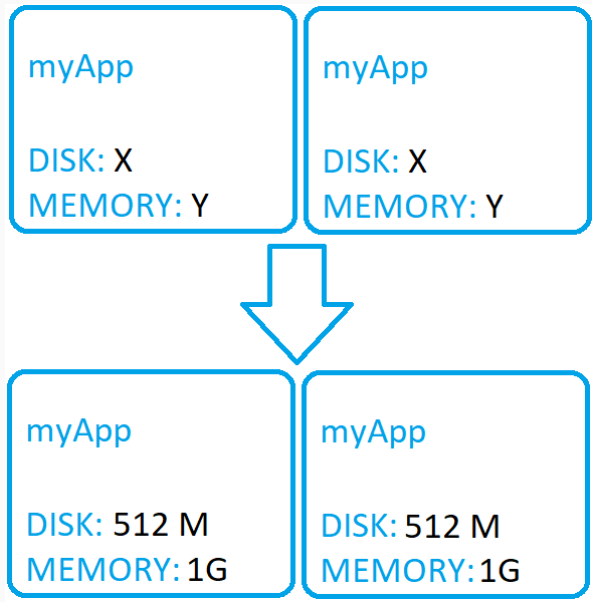
- *MEMORY*
- *DISK*

COMANDO UTILIZADO

```
$ cf scale APP -k DISK
$ cf scale APP -m MEMORY
```

EJEMPLO: $ cf scale myApp -k 512M

$ cf scale myApp -m 1G



ESCALADO

ESCALADO DE LOS COMPONENTES DE CLOUD FOUNDRY

¿QUÉ BUSCAMOS?

- ESCALAR LA PLATAFORMA EN SÍ MISMA, LA IMPLEMENTACIÓN DE CLOUD FOUNDRY.

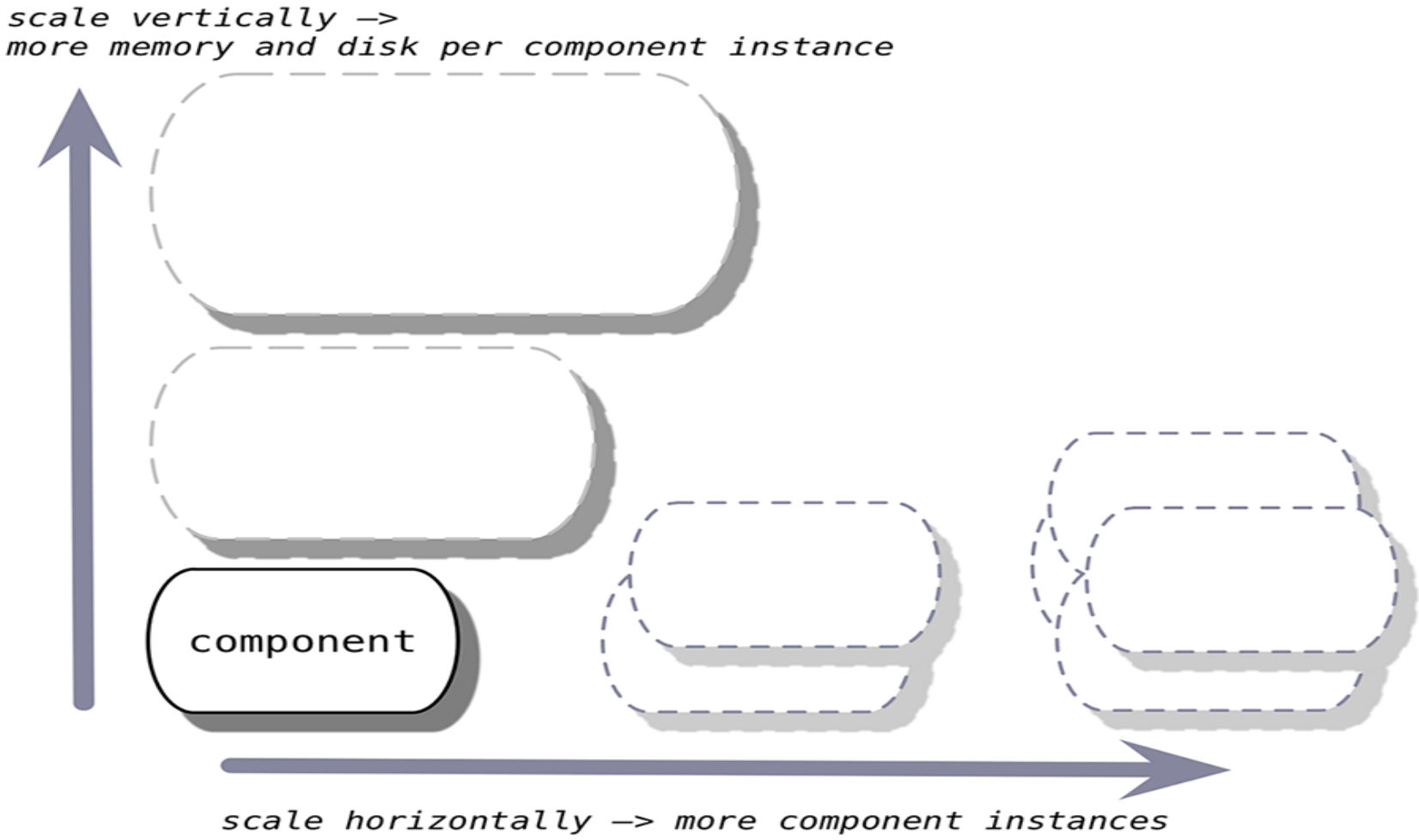
¿CÓMO LO HAREMOS?

A TRAVÉS DEL ESCALADO DE LOS COMPONENTES QUE LA CONFORMAN:

- HORIZONTALMENTE: LEVANTANDO UN MAYOR NÚMERO DE MÁQUINAS VIRTUALES (VM's) QUE EJECUTEN LAS NUEVAS INSTANCIAS DE LOS COMPONENTES QUE SE VAYAN A CREAR.

- VERTICALMENTE: AGREGANDO MÁS MEMORIA Y DISCO A CADA NUEVA INSTANCIA DEL COMPONENTE.

ESCALADO


scale vertically –>
more memory and disk per component instance
component
scale horizontally –> more component instances

ESCALADO

¿POR QUÉ ESCALAR LOS COMPONENTES?

- DURANTE UN UPGRADE O ACTUALIZACIÓN DE UN PRODUCTO, SE VENGA ABAJO UNA DE LAS VM's QUE EJECUTAN INSTANCIAS, PONIENDO EN RIESGO LA DISPONIBILIDAD DEL SERVICIO.

¿CÓMO APROVECHAR EL ESCALADO DE COMPONENTES PARA PALIARLO?

- INSTANCIAR LOS COMPONENTES DEL DESPLIEGUE CLOUD FOUNDRY.
- DISTRIBUIR CORRECTAMENTE LOS COMPONENTES QUE CONFORMAN EL DESPLIEGUE CLOUD FOUNDRY A LO LARGO DE MÚLTIPLES AVAILABILITY ZONES.
- LO CORRECTO SERÍA UN DESPLIEGUE CLOUD FOUNDRY BASADO EN 3 O MÁS "AZ", CONFORMADO EN BASE A MÚLTIPLES INSTANCIAS DE LOS COMPONENTES DE LA CF DISTRIBUIDOS A LO LARGO DE LAS "AZ's".

¿QUÉ EVITAMOS?

- EL RIESGO DE QUE SE CAIGA EL SERVICIO SI SE CAE UNA AVAILABILITY ZONE, EN CASO DE UN DESPLIEGUE CLOUD FOUNDRY BASADO EN MULTI-AVAILABILITY_ZONE.

ESCALADO

CONSIDERACIONES ESCALADO HORIZONTAL

- LEVANTAR UN MAYOR NÚMERO DE VM’s -> MAYOR CAPACIDAD PARA ALOJAR APLICACIONES

CONSIDERACIONES ESCALADO VERTICAL

A LA HORA DE ESCALAR VERTICALMENTE, TRATAR DE GARANTIZAR CON EL ESCALADO:

- ESPACIO LIBRE EN LAS CELDAS DEL SISTEMA DIEGO DE LAS VM’s ANFITRIONAS PARA EJECUTAR INSTANCIAS DE APLICACIONES, PUESTO QUE CUANDO SE DETECTE UNA DIFERENCIA ENTRE LO ACTUAL Y LO DESEADO, SE GENERARÁN NUEVAS INSTANCIAS Y HARÁ FALTA ESPACIO EN LAS CELDAS PARA EJECUTARLAS.

- MEMORIA Y ESPACIO EN DISCO DE FORMA QUE SI UNA VM QUEDA INACTIVA, LAS INSTANCIAS DE APLICACIONES QUE MANEJABA PUEDAN DELEGARSE A OTRA MÁQUINA VIRTUAL.

- ESPACIO LIBRE PARA MANEJAR EL EFECTO DE QUE SE CAIGA UNA AVAILABILITY ZONE, EN CASO DE QUE HAYAMOS HECHO UN DESPLIEGUE CLOUD FOUNDRY BASADO EN MÚLTIPLES “AZ”.

ESCALADO

AUTOESCALADOR DE PIVOTAL CLOUD FOUNDRY

ESCALADO MANUAL FRENTE A ESCALADO AUTOMÁTICO BASADO EN REGLAS Y MÉTRICAS.

¿QUÉ BUSCAMOS?

- CONTROLAR EL COSTE DE EJECUTAR APLICACIONES.
- AL MISMO TIEMPO, MANTENER EL RENDIMIENTO DE LA APLICACIÓN.

¿QUÉ NOS PERMITE?

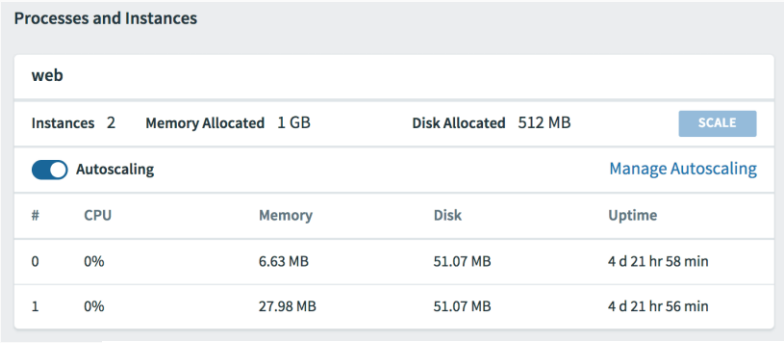
- CONFIGURAR REGLAS QUE AJUSTEN LOS RECUENTOS DE INSTANCIAS SEGÚN UMBRALES Y MÉTRICAS, COMO PUEDE SER EL USO DE LA CPU.
- MODIFICAR EL NÚMERO MÍNIMO Y MÁXIMO DE INSTANCIAS PARA UNA APLICACIÓN, YA SEA MANUALMENTE O SIGUIENDO UN PROGRAMA.

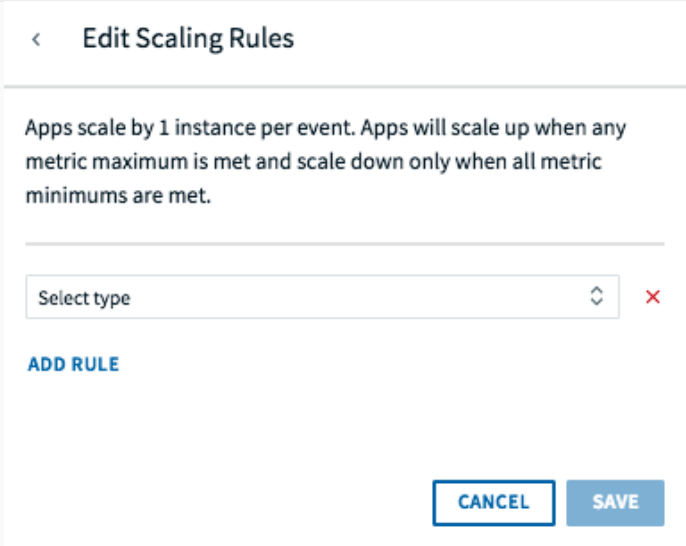
ESCALADO

AUTOESCALADOR DE PIVOTAL CLOUD FOUNDRY

PRIMEROS PASOS

1. CREAR INSTANCIA DEL SERVICIO “APP AUTOESCALER”.
1. VINCULARLA CON LA APLICACIÓN QUE QUERAMOS ESCALAR.
1. IR AL “APP MANAGER” Y SELECCIONAR UNA DE LAS APLICACIONES.
1. IR A LA PESTAÑA “PROCESOS E INSTANCIAS”.
1. ACTIVAR “AUTOSCALING”.
1. IR A “MANAGE AUTOSCALING” Y CONFIGURAR A NUESTRO GUSTO.





ESCALADO

AUTOESCALADOR DE PIVOTAL CLOUD FOUNDRY

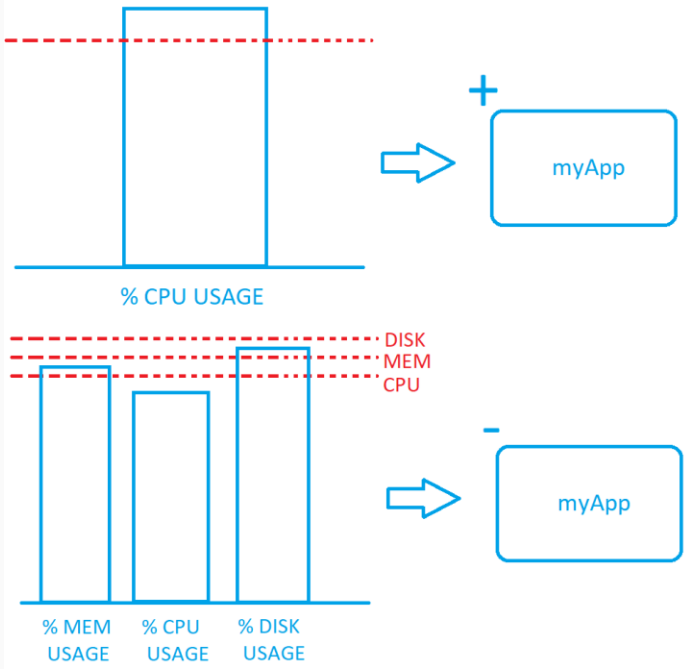
¿CÓMO FUNCIONA?

INCREMENTA O DISMINUYE INSTANCIAS EN BASE A UNA COMPARACIÓN ENTRE LA MÉTRICA ELEGIDA (RAM, CPU, DISCO) Y EL UMBRAL MÍNIMO/MÁXIMO ESTABLECIDO.

EJEMPLO:

- INCREMENTO EN UNA INSTANCIA
  CUALQUIER MÉTRICA
  UMBRAL MÁXIMO ESPECIFICADO.

- DISMINUYE EN UNA INSTANCIA
  SÓLO CUANDO TODAS LAS MÉTRICAS
  CAEN POR DEBAJO DEL UMBRAL MÍNIMO ESPECIFICADO.



ESCALADO

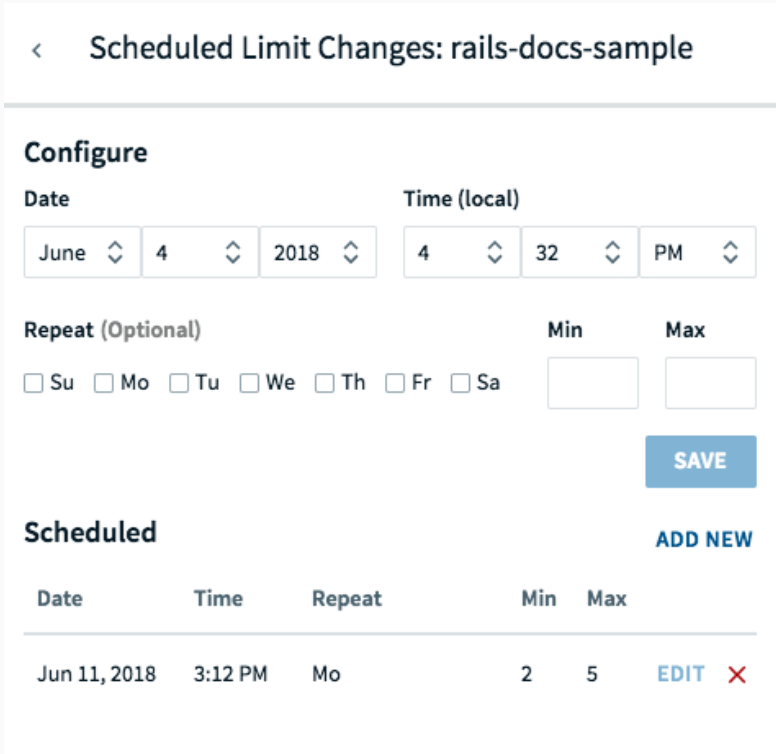
AUTOESCALADOR DE PIVOTAL CLOUD FOUNDRY

ESCALANDO EN DIFERENTES MOMENTOS DEL TIEMPO

- **FECHA Y HORA:** ESTABLEZCA LA FECHA Y LA HORA DEL CAMBIO DE NÚMERO DE INSTANCIAS.

- **REPETIR:** ESTABLEZCA EL DÍA DE LA SEMANA EN EL QUE SE QUIERE REPETIR EL CAMBIO.

- **MÍNIMO Y MÁXIMO:** ESTABLEZCA EL RANGO PERMITIDO DENTRO DEL CUAL LA APLICACIÓN AUTOESCALER PUEDE CAMBIAR EL NÚMERO DE INSTANCIAS PARA UNA APLICACIÓN.



CACHEO

TEMAS A TRATAR:

- CACHEO DE IMÁGENES DOCKER.
- CACHEO DE RECURSOS Y BUILDPACKS.
- CACHEO DE LRP’s EN LA BBS.
- CACHEO DE LOG’s Y MÉTRICAS EN LOGGREGATOR.
- ESCALADO DE LA CACHÉ DE LOG’s.
- CACHEO DE MICROSERVICIOS EN PIVOTAL CLOUD CACHE.

CACHEO

1. CACHEO DE IMÁGENES DOCKER

- CLOUD FOUNDRY DA SOPORTE A DESPLIEGUE DE APLICACIONES EN LA NUBE A TRAVÉS DE IMÁGENES DOCKER.
- PODEMOS TRAERNOS LA IMAGEN DOCKER DE UN REPOSITORIO A LA NUBE CLOUD FOUNDRY, Y “GARDEN-RUNC”, EL BACKEND DEL SUBSISTEMA DIEGO, SE ENCARGA DE MONTAR UN CONTENEDOR PARA EJECUTAR LA APLICACIÓN EN LA NUBE.
- PARA SU REUTILIZACIÓN EN EL FUTURO SIN TENER QUE DEPENDER DE LA DISPONIBILIDAD DE LA MISMA EN EL REPOSITORIO DONDE SE HAYASE, SE PUEDE CACHEAR LA IMAGEN EN **DIEGO-DOCKER-CACHÉ**, DIVIDIÉNDOLA EN CAPAS.
- SI EL BACKEND QUIERE RECUPERAR LA APP A TRAVÉS DE LA CACHÉ, SÓLO HA DE RECUPERAR TODAS LAS CAPAS, UNIRLAS CON SUS BIBLIOTECAS Y MONTARLA EN UN SISTEMA DE ARCHIVOS RAÍZ.
- LO MISMO EN CASO DE QUERER ESCALARLA A BASE DE INSTANCIACIONES DE LA APP, NO DEPENDEMOS DEL REPOSITORIO, SÓLO HABRÍA QUE COGERLA DE LA CACHÉ Y VOLVERLA A INSTANCIAR.
- TAMPOCO TENDRÍAMOS QUE PREOCUPARNOS DE QUE LA IMAGEN QUE HUBIESE EN EL REPOSITORIO PUDIESE HABER SIDO ALTERADA, LA TENEMOS INALTERADA EN CACHÉ.

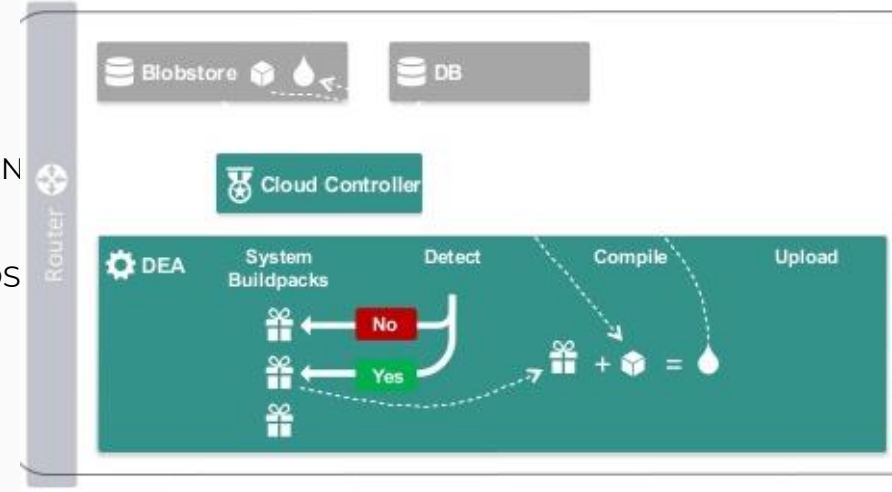
CACHEO

2. CACHEO DE RECURSOS Y BUILDPACKS

- “BLOBSTORE”, ALMACÉN DE OBJETOS DE GRAN TAMAÑO, EN EL CLOUD CONTROLLER (CC).
- CUANDO LOS ARCHIVOS DE RECURSOS DE LAS APLICACIONES SE CARGAN EN EL CLOUD CONTROLLER, SE CACHEAN EN LA BLOBSTORE MEDIANTE UN ALGORITMO HASH, EL “SHA”, PARA GARANTIZAR SU REUTILIZACIÓN FUTURA.

¿CÓMO FUNCIONA?

- ANTES DE CARGAR LOS ARCHIVOS DE LA APP, SE EMITE UNA PETICIÓN DE COINCIDEN
  Y SE DETERMINA SI ALGUNO YA EXISTE EN LA “BLOBSTORE”.
- CUANDO SE VAN A CARGAR LOS ARCHIVOS NECESARIOS, LOS QUE COINCIDEN QUE ESTÁN EN LA CACHÉ SE OMITEN.
- LOS QUE SÍ SE HAN CARGADO Y NO SE HAN OMITIDO,
  SE COMBINAN CON LOS OMITIDOS DE LA CACHÉ
  PARA CREAR EL PAQUETE COMPLETO DE LA APLICACIÓN.

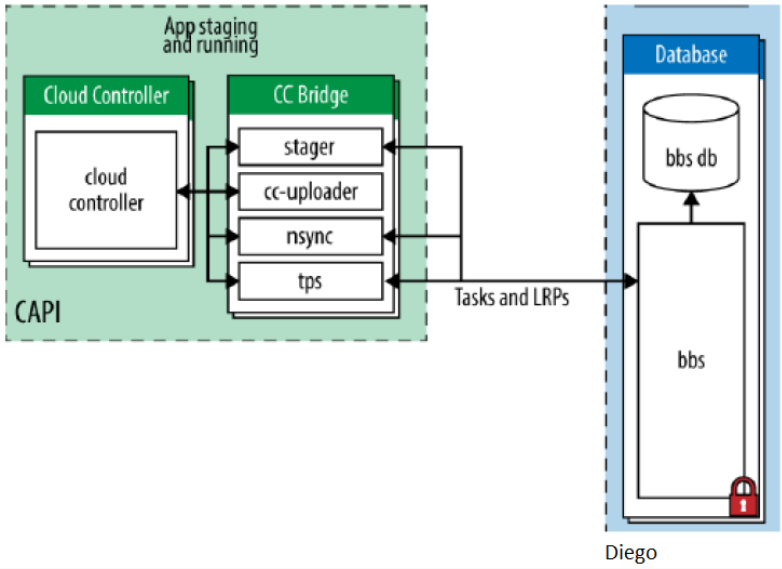


- EN LA BLOBSTORE TAMBIÉN SE PUEDEN ALMACENAR “BUILDPACKS”, ARCHIVOS DE GRAN TAMAÑO GENERADOS DURANTE LA PREPARACIÓN DE UNA APLICACIÓN, Y SE ALMACENAN TAMBIÉN PARA SU POSTERIOR REUTILIZACIÓN. MÁS RÁPIDAMENTE SE PONE EN MARCHA LA APLICACIÓN CUYO BUILDPACK HA SIDO CACHEADO CON ANTERIORIDAD.

CACHEO

3. CACHEO DE LRP’s EN LA BBS

- **LRP:** PROCESOS DE LARGA DURACIÓN.
- **BBS:** BULLETIN BOARD SYSTEM, COMPONENTE DEL SUBSISTEMA DIEGO. MANTIENE UNA CACHÉ ACTUALIZADA DEL ESTADO EN TIEMPO REAL DEL CLUSTER DIEGO, ASÍ COMO UNA REPRESENTACIÓN “AL MOMENTO” DE LAS INSTANCIAS “ACTUAL LRP’s” EN EJECUCIÓN Y LAS “DESIRED LRP’s”.
- SI **ACTUAL LRP’s < DESIRED LRP’s** -> **DEFICIENCIA** -> LLEVA A CABO UNA NUEVA SUBASTA DE TAREAS Y LRP’s.
- SI **ACTUAL LRP’s > DESIRED LRP’s** -> **EXCEDENCIA** -> BBS MATA ALGUNAS INSTANCIAS DE LAS ACTUAL LRP’s.
- DEBIDO A ESA **COMPARACIÓN CONSTANTE** ENTRE ACTUAL LRP’s Y DESIRED LRP’s QUE LLEVA A CABO LA BBS, **SE HACE USO DE UNA CACHÉ**.



CACHEO

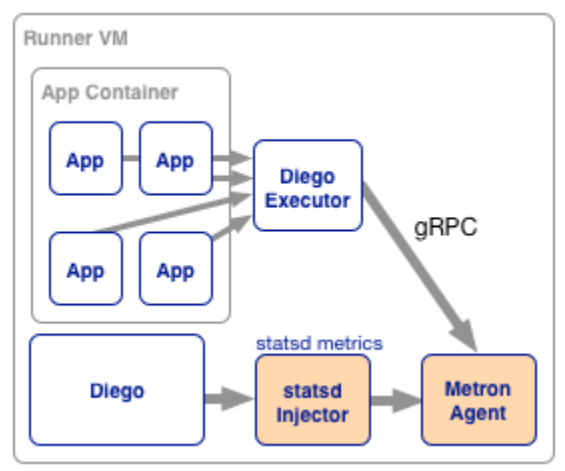
4. CACHEO DE LOG's Y MÉTRICAS EN LOGGREGATOR

- LOS LOGS SON REPORTES DE EVENTOS DETECTADOS,
  EVENTOS QUE REQUIEREN ALGUNA TOMA DE DECISIÓN,
  ERRORES O CUALQUIER OTRO MENSAJE
  QUE SE QUERÍA QUE SE GENERASE.

- LAS MÉTRICAS SON MEDICIONES QUE SE HACEN SOBRE
  VM's CONTENEDORAS DE APLICACIONES,
  ASÍ COMO EL ESTADO DE LOS COMPONENTES.

- EN LOGGREGATOR SE RECOPILAN LOGS DE APLICACIONES
  DESPLEGADAS EN CLOUD FOUNDRY ASÍ COMO LOGS
  DEL FUNCIONAMIENTO DE COMPONENTES QUE CONFORMAN LA NUBE.

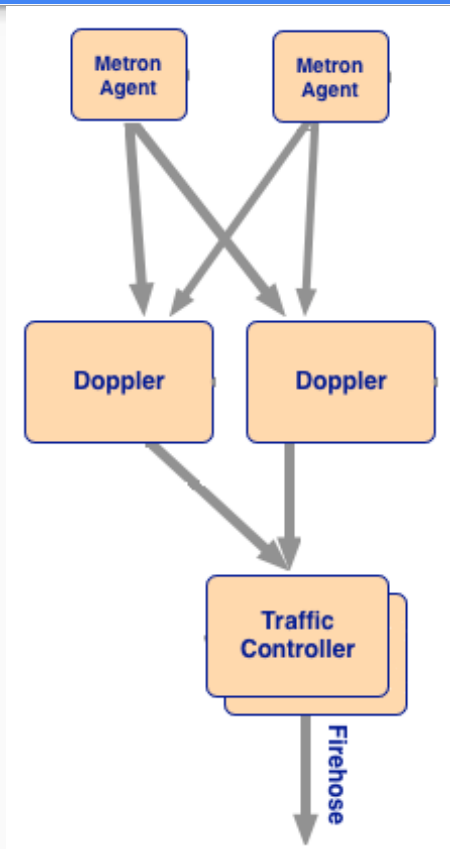
- LOS LOGS SON GENERADOS POR LOS "METRON AGENTS"
  COLOCADOS EN LAS VM's DONDE SE EJECUTAN LOS CONTENEDORES
  DE LAS APLICACIONES.



CACHEO

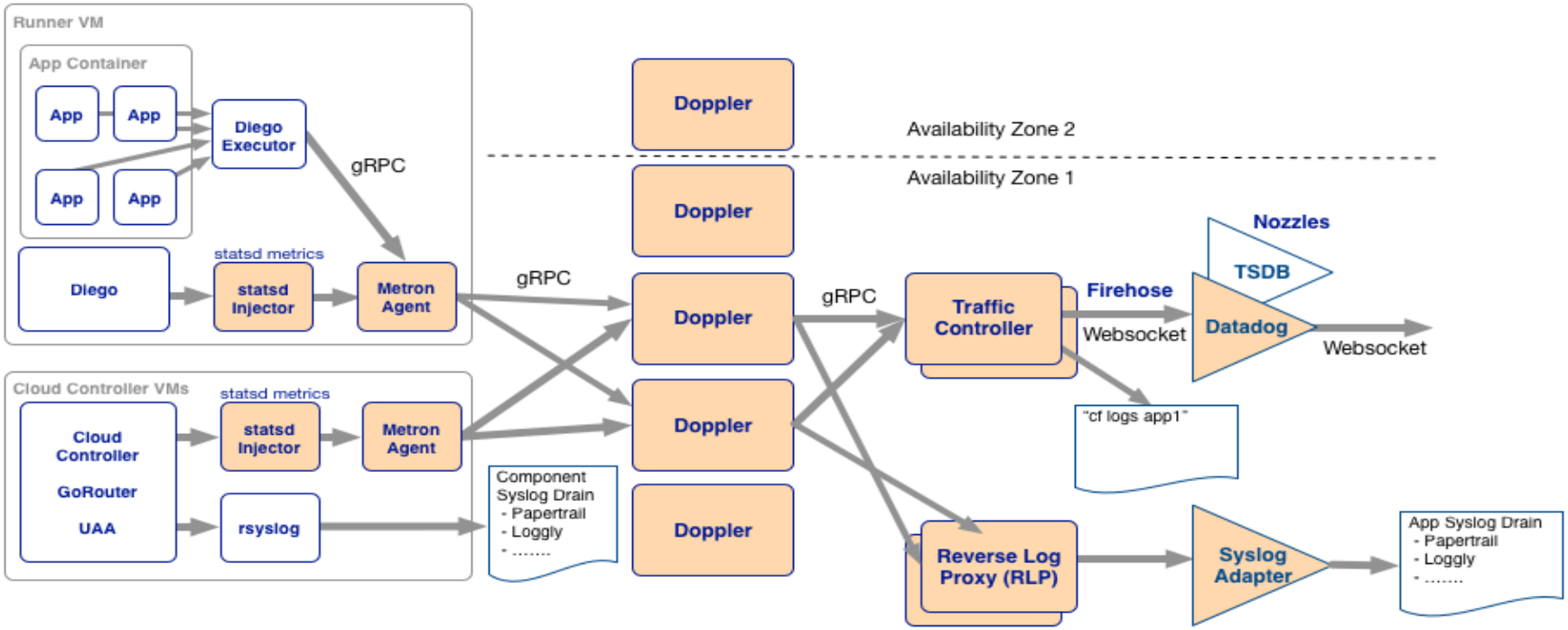
4. CACHEO DE LOG's Y MÉTRICAS EN LOGGREGATOR

- LOG's Y MÉTRICAS SE ENVÍAN A LOS SERVERS "DROPPLERS"
  QUE CONTIENEN UNA CACHÉ.
  SE CACHEAN Y SE ENVÍAN AL "TRAFFIC CONTROLLER".
- EL "TRAFFIC CONTROLLER" LOS RECIBE Y MANEJA SOLICITUDES
  DE LOGS POR PARTE DEL CLIENTE INTERESADO,
  PROPORCIONANDO UNA API EXTERNA.
- LOS LOGS PASAN POR EL "FIREHOSE" (MANGUERA) QUE STREAMEA (FLUJO)
  ESOS DATOS PROVENIENTES DE "TRAFFIC CONTROLLER".
- LOS DATOS PASAN POR EL "FIREHOSE" Y SE BORRAN, NO SON CONSTANTES,
  DE AHÍ LA IMPORTANCIA DE CACHEARLOS.
- LA CACHÉ EN LOS "DROPPLERS" PERMITE CONSULTAR ESOS DATOS DESDE
  LA "FIREHOSE" DURANTE UN PERIODO DE TIEMPO, LUEGO SE BORRAN.
- LOS DATOS SE RECUPERAN A TRAVÉS DE UNA INTERFAZ RESTFUL DE LA CACHÉ.
- MÁS SERVERS "DOPPLERS" -> MÁS CACHÉ's -> MÁS SE ACELERA LA RECUPERACIÓN DE DATOS.



CACHEO

4. CACHEO DE LOG's Y MÉTRICAS EN LOGGREGATOR



CACHEO

5. ESCALADO DE CACHÉ DE LOGS

- SE CONSIGUE UNA ACELERACIÓN EN LA RECUPERACIÓN DE LOG’s, SOBRE TODO EN AQUELLOS DESPLIEGUES CLOUD FOUNDRY QUE CUENTAN CON MÚLTIPLES MÁQUINAS VIRTUALES “DOPPLERS”.

- ESCALAR LA CACHÉ DE LOGS CONSISTE EN LEVANTAR UN MAYOR NÚMERO DE SERVERS “DOPPLERS” DE MANERA QUE SE ACELERA LA RECUPERACIÓN DE LOGS.

CACHEO

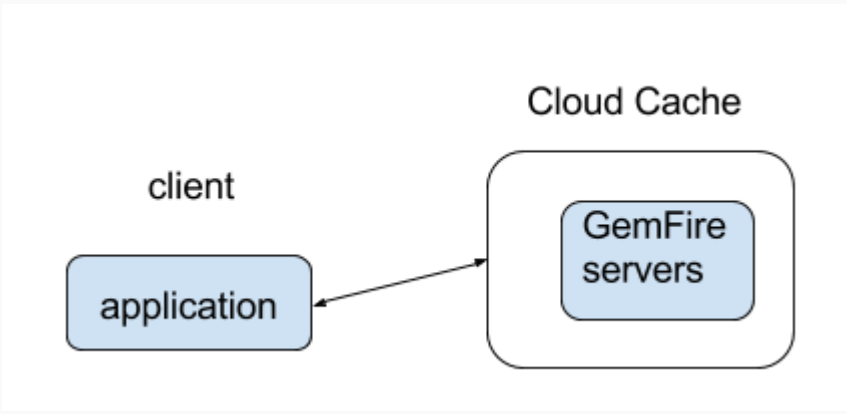
6. CACHEO DE MICROSERVICIOS CON PIVOTAL CLOUD CACHE

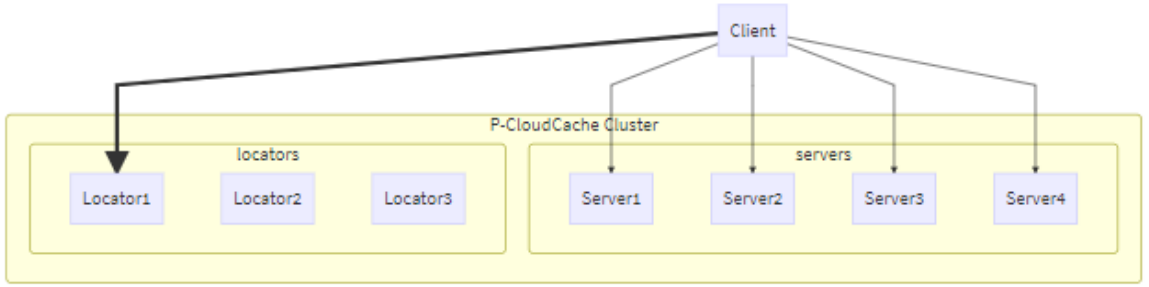
- SOLUCIÓN DE CACHING PARA NUESTRAS APLICACIONES, OFRECIDA POR PIVOTAL SOFTWARE.
- BASADA EN CLUSTERING DE SERVERS “GEMFIRE”.
- CADA “GEMFIRE” ES UN ALMACÉN DE DATOS CONSISTENTE, DE BAJA LATENCIA Y TOLERANCIA A FALLOS.
- SERVERS “LOCATORS” QUE LOCALIZAN A LOS “GEMFIRE”.
- PROPORCIONA ALTA DISPONIBILIDAD, CONSISTENCIA Y GARANTÍA DE REPLICACIÓN DE DATOS.

CACHEO

6. CACHEO DE MICROSERVICIOS CON PIVOTAL CLOUD CACHE

- EL “GEMFIRE” MANTIENE LOS DATOS EN PARES “CLAVE/VALOR”.
- CADA PAR ES UNA “ENTRADA”.
- LAS ENTRADAS SE AGRUPAN LÓGICAMENTE CONJUNTOS LLAMADOS “REGIONES”.
- UNA REGIÓN ES UNA ESTRUCTURA DE DATOS DE TIPO “MAPA” O “DICCIONARIO”.

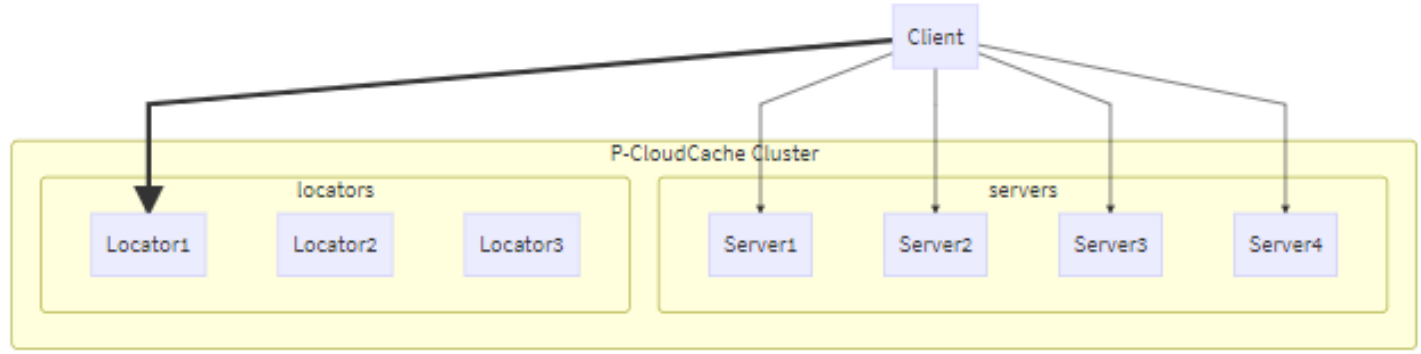




CACHEO

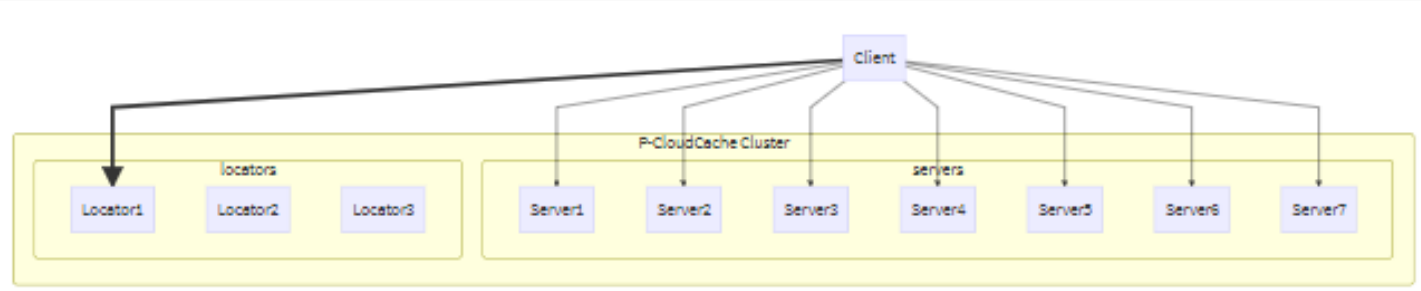
6. CACHEO DE MICROSERVICIOS CON PIVOTAL CLOUD CACHE

ESCALADO



MÁS SERVERS “GEMFIRE” -> MÁS CAPACIDAD CACHÉ

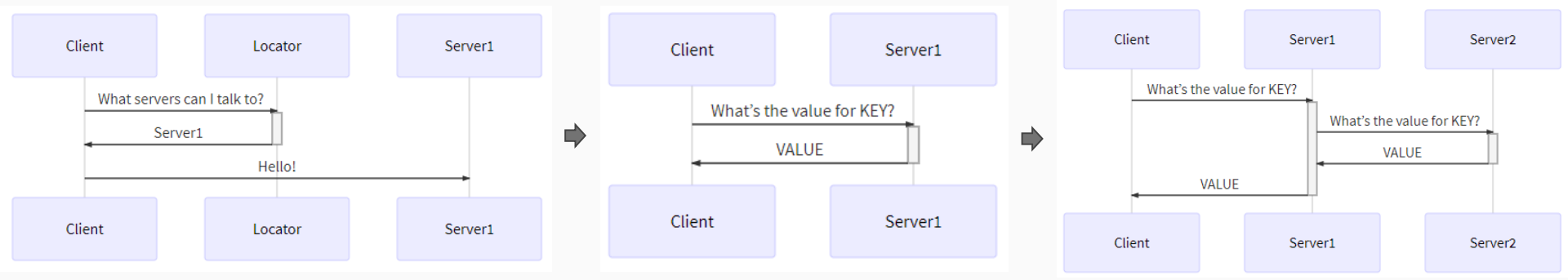
SERVIDORES “LOCATOR” SIEMPRE HAY TRES



CACHEO

6. CACHEO DE MICROSERVICIOS CON PIVOTAL CLOUD CACHE

- RECIBE EL NOMBRE DE “LOCATOR” YA QUE SU FUNCIÓN ES LOCALIZAR UNO DE LOS SERVERS “GEMFIRE” DISPONIBLE, A PETICIÓN DE LA APP QUE QUIERE PEDIRLE UNOS DETERMINADOS DATOS.



1. LA APP CLIENT PREGUNTA AL LOCATOR POR UN “GEMFIRE” CON EL QUE PODER COMUNICARSE.
2. LA APP CLIENT LE PIDE AL SERVER GEMFIRE EL VALOR ASOCIADO A LA CLAVE “KEY”.
3. SI EL SERVER1 NO TUVIESE EL DATO, ÉSTE LE PREGUNTA A SERVER2.

Replicación 

REPLICACIÓN

IMPORTANCIA DE LA REPLICACIÓN:

- Prevenir o minimizar pérdidas en caso de desastre
- Protección ante ataques
- Asegurar disponibilidad durante actualizaciones

REPLICACIÓN

¿CÓMO SE LLEVA A CABO?:

- BOSH crea y gestiona VM clonadas
- Duplicación de instancias
- Cloud Controller gestiona la demanda y lanza las aplicaciones
- El cliente debe replicar routers para garantizar disponibilidad de la red

Disponibilidad

- La disponibilidad de la plataforma es fundamental para la continuidad del servicio.

- Solución de fallos:
  - Diseño para mayor resistencia y alta disponibilidad.
  - Emplear mecanismos de respaldo y restauración.
  - Ejecutar pruebas de verificación de la plataforma de manera constante.

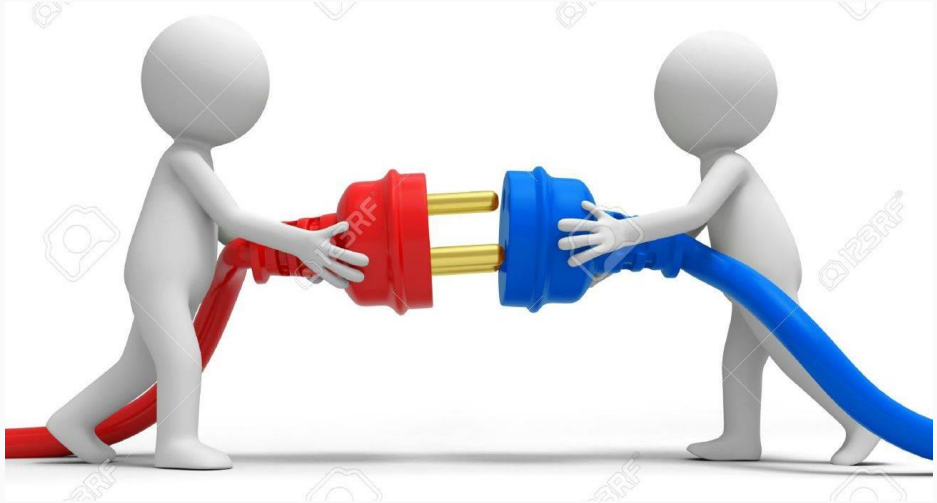
➔ Alta disponibilidad: grado en que un componente está disponible para un usuario cuando éste lo necesite.

➔ Configuración redundante de cada uno de los componentes que haya en la infraestructura.

- Ejemplo: una máquina tiene una probabilidad alta de tener un fallo ⇒ réplica.

➔ Cloud Foundry, junto con BOSH, incorpora capacidad de recuperación automática:

- Reiniciar procesos fallidos del sistema.
- Recrear máquinas virtuales no disponibles.
- Implementación dinámica de nuevas instancias de aplicaciones si una aplicación deja de responder.
- Distribución de las aplicaciones para forzar la separación de la infraestructura subyacente.

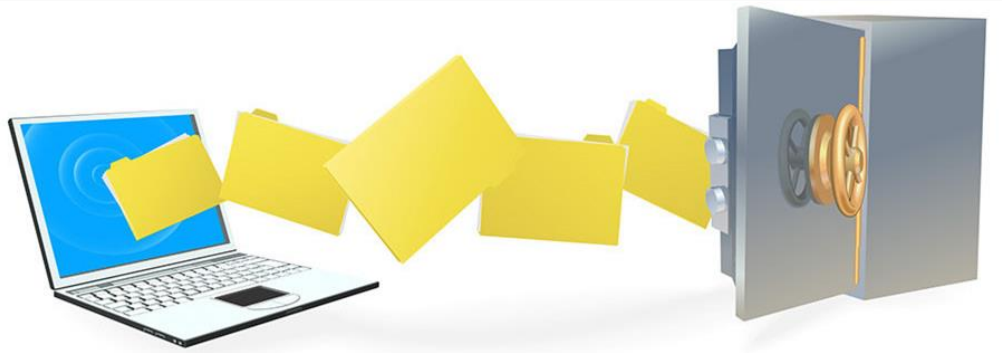


- ➔ Los servicios de respaldo de datos deben mantener la coherencia entre los diferentes centros de datos.
- ➔ Modelo de escritura simultánea.
  - ◆ La experiencia del usuario final lenta.
- ➔ Modelo de escritura diferida.
  - ◆ Riesgo: datos no sincronizados.

- ➔ Posible solución: usar Cassandra.
  - ◆ Permite conservar los cambios localmente.
  - ◆ Algoritmos de resolución de conflictos.

- Situaciones que provocan restaurar completamente su entorno:
  - Creación de una copia de una implementación existente para crear un entorno nuevo.
  - Motivos de mantenimiento.
  - Recuperación de un ataque malicioso.

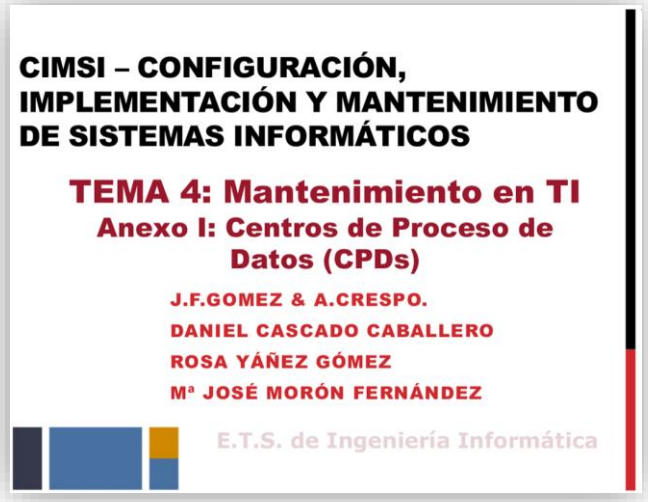
- Hay varios proyectos que existen para respaldar y restaurar Cloud Foundry, como por ejemplo:
  - cf-converger (Engineer Better).
  - cfops (Pivotal Services).



Mantenimiento

Mantenimiento general:

- Datacenter
  - Alimentación eléctrica
  - Extinción de incendios
  - Climatización
  - ...
- Hardware y software
  - Quitar polvo
  - Reemplazar hardware defectuoso
  - Actualizaciones de paquetes, dependencias, kernels
  - ...



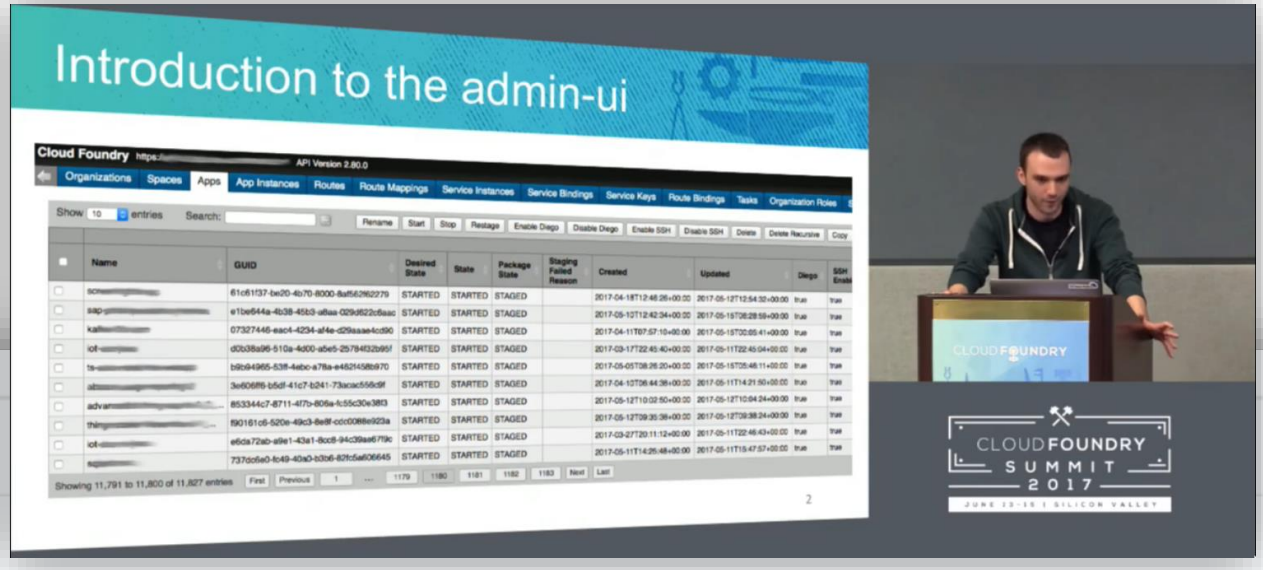
Mantenimiento específico:

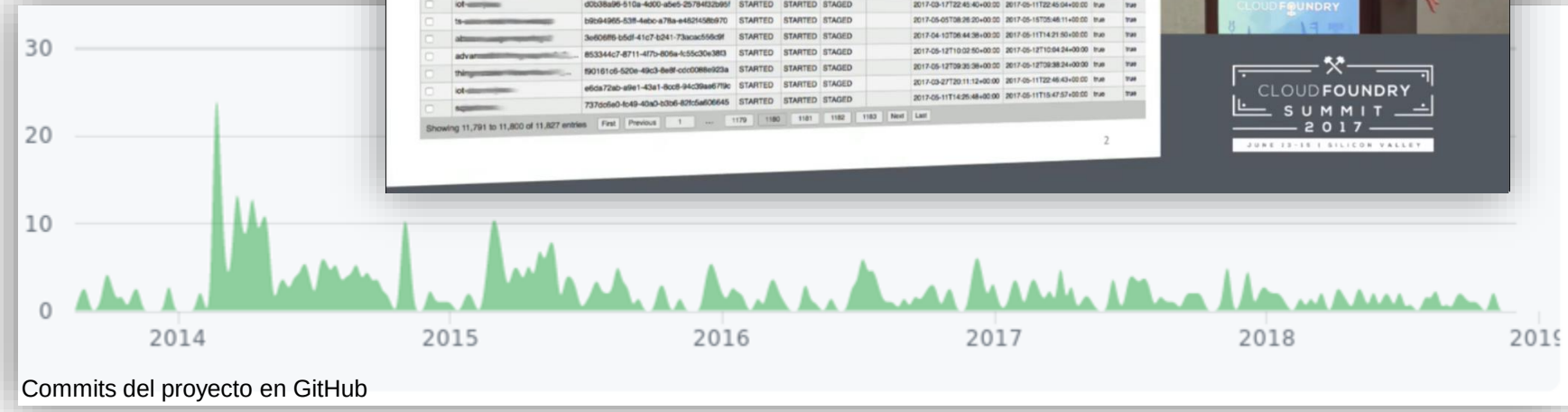
- Inigo y CATs(CF Acceptance Tests)
  - Suite de tests que se le pasan a los componentes Diego para asegurarnos de que funcionan correctamente
  - Ejecutados con cfdot CLI y peticiones curl

- cfdot CLI(Cloud foundry diego operator toolkit)
  - CLI que permite lanzar comandos que cambien el comportamiento de los LRP

Mantenimiento específico:

- admin-ui
  - Proyecto de la comunidad para crear una interfaz gráfica con la que controlar CF





Commits del proyecto en GitHub

Uso en el mundo real

Definición:

El uso en el mundo real hace referencia a quien y a como se hace uso del servicio, en concreto Cloud Foundry, en el ámbito laboral en base a lo que ofrece ese servicio y a sus posibilidades.

Uso por parte de usuarios individuales:

- Código abierto y gratuito.
- Servicio tanto local como en la nube.
- Modelo de servicio PaaS.
- Espacio de memoria individual para cada usuario.

Uso por parte de empresas:

- Contenedores para optimizar la ejecución y gestión de aplicaciones.
- Variante orientada específicamente para empresas.
- Soporte de expertos para el desarrollo de aplicaciones.
- Migración de local a la nube.
- Dispone de consultas tecnológicas.
- Gestor de cuentas para cada cliente por separado.

Referencias

Cloud Foundry Foundation 2018,
Cloud Foundry Components https://docs.cloudfoundry.org/concepts/architecture/index.html - Documentación web de Cloud Foundry}

Winn, D. (2016).
Cloud foundry: the cloud-native platform. (First edition.). Beijing, China: O'Reilly.

Winn, D. (2017).
Cloud Foundry: the definitive guide: develop, deploy, and scale. First Edition. Beijing, China: O'Reilly.

Farmer, R., Jain, R., y Wu, D. (2017).
Cloud Foundry for developers: deploy, manage, and orchestrate cloud-native applications with ease. Birmingham, England: Packt.s.

Michael Grifalconi.
Deploy the admin-ui as a CloudFoundry applications. https://www.youtube.com/watch?v=9Np7swZN-4g - Cloud Foundry Youtube channel.

Cloud Foundry Foundation 2018.
Running Cloud Foundry https://docs.cloudfoundry.org/running/index.htm - Documentación web de Cloud Foundry

Cloud Foundry Foundation 2018.
Administering and Operating Cloud Foundry https://docs.cloudfoundry.org/adminguide/ - Documentación web de Cloud Foundry

F. Gómez, A. Crespo, Daniel Cascado Caballero, Rosa Yánez Gómez, Mª José Morón Fernández.
TEMA 4: Mantenimiento en TI - Temario de la asignatura CIMSI.

Marie Doleželová, Marc Muehlfeld, Stephen Wadeley, Tomáš Čapek, Jaromír Hradílek, Douglas Silas, Jana Heves, Petr Kovář, Peter Ondrejka, Petr Bokoč, Martin Prpič, Adam Kvítek, Eliška Slobodová, Eva Kopalová, Miroslav Svoboda, David O'Brien, Michael Hideo, Don Domingo, John Ha.
System Administrator's Guide https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html-single/System_Administrators_Guide/ - Documentación web de RedHat

Juan F. Gómez Fernández, Adolfo Crespo Márquez. ***Maintenance Management in Network Utilities, Framework and Practical Implementation*** 2012 Springer Verlag UK. London.

Q&A