

Conversación con Gemini

Exported: 12/6/2026 12:02:42

Jordi H.

Has dicho Hoy hemos Hoy hemos tenido un problema con github, hemos dejado de poder acceder a todo nuestro código fuente estamos ordenados en una Enterprise y varias corporaciones. La pregunta que quiero hacerte es cómo podemos cubrirnos de semejante problema, como debemos hacer backup o copias o clones de todo esto y con qué product

Gemini

Vaya susto. Que una organización estructurada en Enterprise y múltiples organizaciones (o corporaciones, como las llamas) se quede a ciegas sin acceso al código fuente es una pesadilla de infraestructura. Demuestra a que, por muy la nube que sea, confiar el 100% de la continuidad de negocio a un único SaaS sin un plan de contingencia local o multi-cloud es un riesgo crítico. Para cubrirnos las espaldas ante una caída total de GitHub o, peor aún, un problema con la cuenta Enterprise, la estrategia pasa por automatizar el respaldo tanto del **código (Git)** como de los **metadatos (Issues, PRs, wikis, acciones)**. Aquí tienes el enfoque de ingeniería para resolverlo, priorizando (como siempre) herramientas de código abierto y autoalojadas para mantener el control ab

1. La Estrategia: ¿Qué necesitamos respaldar?

No basta con hacer un git clone de los repositorios. Si GitHub se cae, necesitas recuperar la actividad. Un backup completo debe incluir

- El código: El árbol de Git, todas las ramas (branches) y los tags.
- Los metadatos: Pull Requests, Issues, comentarios, releases, proyectos y la configuración de usuarios/permisos (esto último es lo más difícil de replicar, pero se puede guardar en JSON/API)

2. Soluciones de Software Libre (Autoalojadas)

Si queremos mantener el espíritu de soberanía tecnológica y gestionar nosotros mismos el laboratorio de backups, estas son las mejores opciones del ecosistema open source

A. Forgejo / Gitea (Espejado Automático)

Una de las formas más eficientes de tener un "Plan de Desastre" activo es montar una instancia propia de **Forgejo** (un fork totalmente comunitario y libre) o **Gitea** en vuestra propia infraestructura o en otro proveedor de nube

- Cómo funciona: Ambas herramientas tienen una funcionalidad nativa de Mirroring (Espejado). Puedes configurar la instancia para que, mediante un token de GitHub, clone y sincronice automáticamente cada pocas horas (o mediante webhooks) todos los repositorios de tus organizaciones
- Ventaja: Si GitHub se cae, tienes una interfaz web idéntica a GitHub levantada y lista para que los desarrolladores cambien el git remote y sigan empujando código en minutos
- Qué cubre: Código, ramas, tags y, dependiendo de la configuración de migración, parte de los metadatos básicos

B. GitHub Backup (gh-backup)

Si lo que buscas es un backup puro (archivos listos para guardar en un almacenamiento en frío o S3 propio),

existen herramientas CLI excelentes

- python-github-backup o github-backup (de Jose Luis Rivas): Son herramientas de línea de comandos que interactúan con la API de GitHub
- Cómo funciona: Creas un contenedor (en Podman/Docker) que se ejecuta con un cron. Usando un token de la Enterprise, descarga de forma recursiva todas las organizaciones, sus repositorios (los clona con --mirror) y descarga en archivos .json todos los issues, pull requests, comentarios y wiki
- Almacenamiento: Esos directorios resultantes los puedes empaquetar y subir a un almacenamiento compatible con S3 (como MinIO si lo montas local, o Backblaze B2 / AWS S3 para salir de tu red)

3. Productos Comerciales (SaaS / Enterprise)

Si la directiva de la Enterprise prefiere una solución "llave en mano" con soporte y cumplimiento normativo estricto, el mercado de backups para SaaS de desarrollo está bastante maduro

- Rewind (antes BackHub): Es el estándar de facto para backups de GitHub. Se conecta a nivel Enterprise, automatiza copias diarias a su nube o a tu propio bucket de AWS S3, y permite restaurar metadatos y código con un par de clic
- GitProtect.io: Una solución corporativa muy potente. Permite definir políticas de backup estrictas, cifrado en tránsito y reposo, y almacenamiento multi-cloud (cumple con normativas ISO 27001 y SOC2)

4. Arquitectura Recomendada para vuestro caso (Enterprise)

Para una estructura compleja con varias corporaciones, lo ideal es **no depender de soluciones individuales por enterprise**.

1. Auditoría mediante la API de GitHub: Desarrollar un pequeño script en Python (vuestro lenguaje vehicular) que use la API de GitHub Enterprise para listar dinámicamente todas las organizaciones y repositorios nuevos. Los entornos vivos cambian cada día y no puedes añadir repositorios a mano al backu
 2. El Nodo de Backup (Laboratorio/Servidor local): Un contenedor dedicado que corra una tarea programada. Este nodo descarga los clones --mirror (que incluyen todas las referencias Git) y los metadatos vía API
 3. Regla 3-2-1: * 3 copias: El GitHub original, la copia local en vuestro servidor de backups (los archivos sueltos/espejo)
- 2 soportes diferentes: Disco local / cabina de almacenamiento y almacenamiento en bloques.
 - 1 copia fuera de la oficina: Un bucket S3 cifrado en otro proveedor de infraestructura.

¿Tuvisteis problemas con la caída global de servicios de GitHub o fue un bloqueo/problema de credenciales de vuestra propia organización Enterprise? Si me das más detalles sobre cómo tenéis estructurada la infraestructura donde alojaríais el backup, podemos diseñar el script o contenedor específico para dejarlo automatiza

Jordi H.

Has dicho Nos vamos a centrar solo en soluciones para la organización, no para toda la enterprise. Además, vamos a usar la primera propuesta que nos has hecho

Gemini

Entendido, nos enfocamos en una sola organización y utilizaremos Forgejo o Gitea para el espejado automático. Para poner en marcha esta solución, lo primero sería desplegar una instancia de Forgejo o Gitea en un servidor accesible para vosotros. Una vez configurado, podréis utilizar la funcionalidad de sincronización o mirroring para conectaros a vuestra organización de GitHub mediante un token. Esto permitirá que todos los repositorios se clonen y mantengan actualizados de forma automática según lo configuréis. Al centralizarlo en una sola organización, la gestión de los permisos y el control del acceso son mucho más sencillos para vuestro equipo. ¿Os gustaría que profundizáramos en algún aspecto técnico del despliegue de Forgejo o

en la configuración del mir

Jordi H.

Has dicho Otra cosa más. Nos comenta el director de IT que ha habido un problema con la rotación de unos

Gemini

Dime, ¿qué problema ha habido con la rotación de qué? Estoy aquí atento para seguir ayudándoles con este asunto

Jordi H.

Has dicho Un problema con la rotación de los certificados. No sabemos mucho más, pero es probable que podamos recuperar el acceso en unas horas o días

Gemini

Ese problema con la rotación de certificados puede afectar temporalmente a las conexiones seguras, interrumpiendo la comunicación entre vuestros sistemas y GitHub, incluyendo los procesos de sincronización automática que queráis configurar con Forgejo. Cuando se restaure el acceso, será muy importante verificar que los certificados estén correctamente emitidos y actualizados para evitar que se repita el fallo. Además, esto reafirma la importancia de tener vuestro propio sistema de respaldo, como el que estáis planeando, para que en situaciones como esta no dependáis exclusivamente de los servicios de terceros y podáis mantener la continuidad de vuestras operaciones. ¿Hay algo más que os gustaría revisar sobre la configuración de Forgejo para este caso?

Jordi H.

Has dicho La manera de reaccionar que hemos tenido ha sido mantener las sesiones abiertas sin cerrarlas porque no podíamos hacer nuevos logins. El problema parecía que estaba en el single sign on y con las sesiones abiertas lo que hemos hecho han sido bajadas en zip de todos los repositorios y además clonaciones con el GitHub desktop

Gemini

Fue una reacción muy acertada mantener las sesiones abiertas para poder asegurar el acceso y salvar el código. Descargar los ZIPs y clonar con GitHub Desktop es una manera efectiva de actuar rápido en una emergencia, pero dependieron de la intervención manual de vuestro equipo. Por eso, planificar la automatización mediante soluciones como Forgejo, tal como comentamos, es tan importante, ya que eliminaría la necesidad de realizar estas descargas manuales y garantizaría que siempre tenéis una copia exacta y sincronizada de todos vuestros repositorios disponible sin intervención de los usuarios. ¿Os habéis planteado ya cómo sería el proceso de migración de los repositorios actuales a esa nueva infraestructura?

Jordi H.

Has dicho Otra pregunta, en el caso de utilizar el forjejo, ¿las copias se realizarían a a tiempos concretos o cada vez que alguien subiera nuevo código

Gemini

Con Forgejo, puedes configurar el espejado de diferentes maneras. Una opción es programar sincronizaciones periódicas cada cierto tiempo, como cada hora o cada día, para que el sistema busque cambios en GitHub. L a otra opción, que es más en tiempo real, es utilizar webhooks para que, en cuanto alguien suba nuevo código a GitHub, este notifique a Forgejo y se realice la copia inmediatamente. ¿Preferís un enfoque periódico o en tiempo real para vuestra organizac

Jordi H.

Has dicho Muéstrame las diferencias entre forjejo y jitea y los puntos fuertes también.

Gemini

Aunque Forgejo y Gitea comparten sus raíces, sus filosofías difieren en la gobernanza. Gitea es un proyecto que ha tomado un enfoque más comercial, mientras que Forgejo nació como un fork comunitario, gestionad o por la asociación Codeberg y centrado en la soberanía y la libertad del software. El punto fuerte de Gitea radica en su ecosistema más amplio y su mayor integración con herramientas de terceros. Por su parte, las fortalezas de Forgejo incluyen su enfoque estricto en la privacidad, la ausencia de funciones privativas y la garantía de que el proyecto siempre pertenecerá a la comunidad. Para un sistema de backup, la elección dependerá de si valoráis más el control comunitario y la independencia o el soporte a un ecosistema comercial más m

Jordi H.

Has dicho vamos a instsalar forgejo

Gemini

¡Excelente decisión! Elegir **Forgejo** os garantiza un control total del software, alineado con la soberanía tecnológica y el código libre real, sin sorpresas comerciales en el futuro. Dado que vuestro objetivo inmediato es **usarlo como un nodo de respaldo (espejo)** para vuestra organización de GitHub y mitigar el riesgo de otra caída, lo ideal es desplegarlo mediante **contenedores**. Esto os facilitará enormemente la gestión, las actualizaciones y el aislamiento en vuestra infraestructura (sea en vuestro entorno local, home lab o laboratorio). A continuación, os presento una guía de despliegue rápido utilizando **Docker Compose / P** es el estándar de la industria.

1. El Archivo de Configuración (`docker-compose.yml`)

Cread un directorio exclusivo para Forgejo y guardad este archivo. Está configurado con una base de datos **PostgreSQL** (más robusta para entornos corporativos que la base de datos SQLite por defecto) y persistencia de datos en volúmenes. YA

CODE

```
version: '3.8'

services:
  server:
    image: codeberg.org/forgejo/forgejo:9
    container_name: forgejo
    environment:
      - USER_UID=1000
      - USER_GID=1000
      - FORGEJO__database__DB_TYPE=postgres
      - FORGEJO__database__HOST=db:5432
      - FORGEJO__database__NAME=forgejo
      - FORGEJO__database__USER=forgejo
      - FORGEJO__database__PASSWD=Un_Password_Muy_Seguro_Aqui
    restart: always
    volumes:
      - ./forgejo-data:/data
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "3000:3000"
      - "2222:22"
    depends_on:
      - db

  db:
    image: postgres:15-alpine
    container_name: forgejo_db
    restart: always
    environment:
      - POSTGRES_USER=forgejo
      - POSTGRES_PASSWORD=Un_Password_Muy_Seguro_Aqui
      - POSTGRES_DB=forgejo
    volumes:
      - ./postgres-data:/var/lib/postgresql/data
```

⚠ **Nota:** Cambiad Un_Password_Muy_Seguro_Aqui por una contraseña fuerte antes de levantar el servicio.

2. Lanzamiento del Servicio

Para arrancar el servidor, ejecutad en vuestra terminal: Bash

CODE

```
docker-compose up -d
# O si utilizáis Podman:
podman-compose up -d
```

Una vez levantado, podréis acceder a la interfaz web de vuestra réplica a través de `http://<IP_DEL_SERVIDOR>:3000`

3. Configuración Inicial Clave para Backups

Cuando entréis por primera vez a la web, os aparecerá el asistente de instalación. La mayoría de los datos ya estarán rellenos gracias al archivo YAML, pero prestad atención a lo siguiente

- URL de la instalación (ROOT_URL): Configurad aquí el dominio o la IP exacta con la que vuestro equipo accederá a Forgejo
- Cuenta de Administrador: Creará el primer usuario. Este será el administrador global que gestionará la organización de respaldo
- Deshabilitar el registro público (Recomendado): En la sección de "Configuración opcional", desactivad el

registro automático de usuarios. Al ser un entorno de backup interno de vuestra organización, no queréis que cualquiera pueda crearse una cuenta

4. Próximo Paso: Conectar con GitHub

Una vez tengáis el panel de Forgejo operativo, los pasos para automatizar el respaldo de la organización serán:

1. Crear la Organización en Forgejo: Cread una organización con el mismo nombre que tenéis en GitHub para mantener la estructura espejo
2. Generar un Token en GitHub: Necesitaréis un Personal Access Token (PAT) en GitHub con permisos de lectura (repo y read:org) para que Forgejo pueda clonar el código a pesar del Single Sign-On (SSO)
3. Configurar el Espejo de Repositorios: Forgejo permite migrar repositorios seleccionando la opción "Migración -> Espejo (Mirror)". Introduciendo la URL de GitHub y el token, empezará a sincronizar vuestro código automáticamente según el intervalo que defináis (por ejemplo, cada pocas horas)

¿Tenéis ya pensado dónde vais a desplegar este contenedor (en local, en la nube...) para que os eche una mano con la configuración de red o el almacenamiento de los volúmenes

Jordi H.

Has dicho tengo funcionando un servidor mysql en el mismo sistema para cosas variadas. lo puedo usar? hay inconvenientes

Gemini

Sí, **puedes usarlo perfectamente**. Forgejo (y el ecosistema Gitea del que proviene) tiene un soporte nativo excelente para MySQL y MariaDB, además de PostgreSQL. Sin embargo, dado que estás consolidando servicios en un mismo servidor, hay varios factores técnicos, de rendimiento y arquitectónicos que debes poner en la balanza antes de proceder

El Enfoque de Configuración (Docker/Podman + Host)

Al tener MySQL corriendo directamente en el sistema host (o en otro contenedor independiente), la configuración de tu docker-compose.yml cambia. Ya no necesitas el servicio db interno, sino redirigir Forgejo hacia el puerto de tu MySQL actual (por defecto 3306). Tu archivo quedaría simplificado así: YA

CODE

```
version: '3.8'

services:
  server:
    image: codeberg.org/forgejo/forgejo:9
    container_name: forgejo
    environment:
      - USER_UID=1000
      - USER_GID=1000
      - FORGEJO__database__DB_TYPE=mysql
      - FORGEJO__database__HOST=172.17.0.1:3306 # IP de la pasarela Docker o la IP de tu red local
      - FORGEJO__database__NAME=forgejo_db
      - FORGEJO__database__USER=forgejo_user
      - FORGEJO__database__PASSWORD=Tu_Password_De_MySQL
    restart: always
  volumes:
    - ./forgejo-data:/data
    - /etc/timezone:/etc/timezone:ro
    - /etc/localtime:/etc/localtime:ro
  ports:
```

- "3000:3000"
- "2222:22"

💡 **Nota de red:** Si MySQL está en el host y Forgejo en un contenedor, localhost:3306 dentro del contenedor no funcionará (apuntaría al propio contenedor). Debes usar la IP de la interfaz puente de Docker (habitualmente 172.17.0.1) o la IP privada del servidor, y asegurarte de que MySQL acepte conexiones en esa interfaz

Ventajas de reutilizar tu MySQL

- Menor consumo de recursos: Evitas levantar otro proceso de base de datos (postgres) en memoria RAM. Para laboratorios o servidores compactos, esto optimiza mucho el hardware
- Centralización del mantenimiento: Tus scripts actuales de mantenimiento, optimización y, sobre todo, vuestras estrategias de backup de bases de datos (como mysqldump o copias de volumen) se aplicarían también a Forgejo de forma unificada

Inconvenientes y riesgos a considerar

A pesar de ser viable, en entornos corporativos o de infraestructura crítica conviene analizar estos contras:

1. El "Efecto Vecino Ruidoso" (Disputa de recursos)

Al compartir el motor de base de datos con "cosas variadas", si uno de tus otros servicios lanza una consulta masiva, mal indexada, o sufre un pico de carga, **degradará el rendimiento de Forgejo** (y viceversa). Si Forgejo está haciendo un espejado masivo de decenas de repositorios a la vez, el uso de I/O de disco y CPU de MySQL se resentirá

2. Aislamiento y Seguridad

Si un servicio vulnerable de esas "cosas variadas" llega a comprometer el acceso administrador de MySQL, la base de datos de Forgejo (que contiene metadatos de código, tokens de acceso y hashes de usuarios) quedará expuesta. El aislamiento total (un contenedor por servicio con su propia BD interna) mitiga este riesgo de impacto

3. Requisito de configuración en MySQL (¡Crucial!)

Forgejo requiere que MySQL esté configurado específicamente para manejar correctamente índices largos y UTF-8 moderno. Asegúrate de que tu servidor MySQL tenga activados estos parámetros en su archivo de configuración (my.cnf): InnoDB, TO

CODE

```
[mysqld]
innodb_file_format = Barracuda # (Si usas versiones antiguas de MySQL)
innodb_large_prefix = 1
character-set-server = utf8mb4
collation-server = utf8mb4_general_ci
```

Si tu MySQL ya da servicio a aplicaciones antiguas que requieren codificaciones heredadas, cambiar esto de forma global podría romper algo

Recomendación

Si el servidor va holgado de recursos (RAM/CPU), **la arquitectura limpia de mantener la base de datos** a largo plazo: evita conflictos de red, no toca tu instalación actual de MySQL y se levanta y destruye en un solo bloque. Si prefieres aprovechar tu infraestructura MySQL por consistencia en las copias de seguridad, adelante, solo recuerda crear un usuario específico para Forgejo limitado estrictamente a su nueva base de datos (forgejo_db). ¿Qué versión de MySQL tienes corriendo actualmente en el sistema?

