Integración técnica GitHub ? Asana — Nexus33

Fecha de implementación: 29 de julio de 2026 **Autor de la implementación:** Hector García (vía sesión asistida con Claude Code) **Propósito:** documentar cada paso técnico para que cualquiera pueda revisar, ajustar, depurar, o extender esto a un repo/proyecto nuevo sin tener que re-derivarlo desde cero.

---

1. Arquitectura — qué existe realmente

Hay **dos integraciones distintas** entre Asana y GitHub. Solo una de las dos es la que realmente automatiza algo hoy:

	App oficial "GitHub" de Asana	Automatización propia (la que funciona)
Dónde se activó	Asana → Ajustes → Aplicaciones → GitHub → Conectar (por proyecto)	GitHub Actions, un workflow por repo
Qué hace	Permite pegar manualmente la URL de un PR en el campo "Aplicaciones" de una tarea	Comenta automáticamente en la tarea de Asana correcta, sin intervención humana
Requiere copiar/pegar	Sí (se descartó por esto)	No
Estado actual	Queda conectada pero no se usa activamente	**Esta es la que está en producción**

Todo lo que sigue en este documento describe la **automatización propia**.

---

2. Cómo funciona (flujo técnico)

1. Un desarrollador abre/aprueba/mergea un PR cuyo **título** contiene un ticket con el patrón `NEXDEV-[0-9]+` (ej. `NEXDEV-123`), sin importar mayúsculas/minúsculas.
2. Un GitHub Action (`.github/workflows/link-asana-task.yml`, presente en cada uno de los 12 repos) se dispara con el evento correspondiente.
3. El Action extrae el ticket del título con `grep -oiE 'NEXDEV-[0-9]+'`.
4. Llama a la API de Asana:

- `GET /projects/{project_gid}/custom_field_settings` → busca el campo personalizado llamado exactamente `NEXDEV` y obtiene su `gid`.
- `GET /projects/{project_gid}/tasks?opt_fields=name,custom_fields` (paginado) → recorre las tareas del proyecto "Desarrollo" buscando la que tenga ese campo con el mismo valor que el ticket extraído.
- `POST /tasks/{task_gid}/stories` → agrega un comentario en la tarea con el evento ocurrido.

5. El comentario en Asana varía según el evento (ver tabla abajo).

Nota importante: esta búsqueda NO usa el endpoint de búsqueda avanzada de Asana (`/workspaces/{gid}/tasks/search`), porque ese requiere plan premium y no es confiable en plan Starter. En su lugar, se listan las tareas del proyecto directamente y se filtra en el propio script — funciona en cualquier plan pago de Asana.

---

3. Datos de configuración (IDs reales, no cambian salvo que se reestructure Asana)

Dato	Valor
Workspace de Asana (nexus33.com)	`1208690669624843`
Proyecto "Desarrollo" (gid)	`1209109605438289`
Nombre del campo personalizado en Asana	`NEXDEV` (texto libre, formato `NEXDEV-###`)
Nombre del secreto en GitHub	`ASANA_PAT` (repository secret, uno por repo)
Nombre del check obligatorio en branch protection	`Validar ticket Asana`

Si en algún momento se reorganiza el proyecto de Asana (se borra y se crea de nuevo, por ejemplo), estos `gid` cambian y hay que actualizar `ASANA_PROJECT_GID` en el workflow de los 12 repos.

---

4. Los 3 workflows involucrados (qué hace cada uno)

4.1 `.github/workflows/check-asana-ticket.yml` — el candado obligatorio

- Se dispara con: `pull_request: [opened, edited, synchronize, reopened]`.
- Verifica que el título del PR tenga `NEXDEV-[0-9]+` (insensible a mayúsculas).
- Si falta, falla el check (`exit 1`) — y como está en `required_status_checks` de branch protection, **bloquea el botón de Merge**.
- No habla con Asana para nada, es puramente un candado del lado de GitHub.

4.2 `.github/workflows/link-asana-task.yml` — el que comenta en Asana

- Se dispara con: `pull_request: [opened, closed]` y `pull_request_review: [submitted]`.
- Busca la tarea en Asana y comenta según el evento:

Evento	Condición	Comentario en Asana
`pull_request.opened`	—	`PR abierto por <usuario>`
`pull_request_review.submitted`	`review.state == approved`	`Aprobado por <usuario>`
`pull_request_review.submitted`	`review.state == changes_requested`	`Cambios solicitados por <usuario>`
`pull_request.closed`	`merged == true`	`Mergeado a <rama base> por <usuario que mergeo>`
`pull_request.closed`	`merged == false`	`Cerrado sin mergear`

- **No es un check obligatorio** — si falla (ej. Asana está caído, o no encuentra la tarea), no bloquea el merge, solo se salta el paso con un `::warning::`.
- Si el secreto `ASANA_PAT` no está configurado en el repo, se salta silenciosamente (`exit 0`) sin marcar error — así no rompe repos donde todavía no se configuró.

4.3 `.github/workflows/auto-assign-pr.yml` — asignación automática

- Se dispara con: `pull_request: [opened]`.
- Asigna el PR (campo "Assignee", no "Reviewer") a quien lo abrió, usando el token automático `secrets.GITHUB_TOKEN` (no necesita `ASANA_PAT` ni ningún secreto configurado a mano).
- Es editable manualmente después, esto solo pone el valor por defecto.
- No tiene relación con Asana, es puramente organizativo del lado de GitHub.

---

5. El secreto `ASANA_PAT` — cómo se generó y se instaló

1. En Asana: **perfil → Ajustes → Aplicaciones → "Ver la consola del desarrollador" → Create new token.**
2. El token se pegó manualmente (nunca por chat/API) en cada uno de los 12 repos: `Settings → Secrets and variables → Actions → New repository secret`, nombre exacto `ASANA_PAT`.
3. *(Detalle técnico de cómo se automatizó la carga la primera vez, por si se repite):* GitHub exige que el valor de un secreto se suba cifrado con la llave pública del repo (`crypto_box_seal` de libsodium) — se hizo con un script de Node.js usando el paquete `libsodium-wrappers`:
   - `GET /repos/{owner}/{repo}/actions/secrets/public-key` → devuelve `key` (base64) y `key_id`.
   - Cifrar el valor del token con esa llave pública (`sodium.crypto_box_seal`).
   - `PUT /repos/{owner}/{repo}/actions/secrets/ASANA_PAT` con `{ encrypted_value, key_id }`.
   - El token nunca se imprimió en ningún log ni salida de consola durante este proceso.

---

6. Cómo agregar un repo NUEVO a esta integración (runbook)

1. **Confirmar que GitHub Actions está habilitado en el repo.** Ver sección 8 — este fue el bug más grande que encontramos: viene deshabilitado por defecto en cuentas personales. Revisar en `Settings → Actions → General`, o vía API: `GET /repos/{owner}/{repo}/actions/permissions` debe devolver `"enabled": true`.
2. **Copiar los 3 archivos de workflow** de cualquier repo existente (ej. `nexus33-frontend`) a `.github/workflows/` del repo nuevo:
   - `check-asana-ticket.yml`
   - `link-asana-task.yml`
   - `auto-assign-pr.yml`
3. **Agregar el secreto** `ASANA_PAT` en el repo nuevo (mismo token de Asana que ya existe, no hace falta generar uno nuevo — es el mismo workspace).
4. **Confirmar los nombres de rama** del repo nuevo (`master` vs `main`, `development` vs `develop`) y ajustar branch protection en consecuencia.
5. **Configurar branch protection** en las ramas principal y de desarrollo:
   - `required_pull_request_reviews.required_approving_review_count = 1`
   - `required_pull_request_reviews.dismiss_stale_reviews = true`
   - `required_pull_request_reviews.require_code_owner_reviews = false`

- `enforce_admins = true`
- `required_status_checks.contexts = ["Validar ticket Asana"]`, `strict = false`
- `allow_force_pushes = false`, `allow_deletions = false`

6. **Ojo con el problema de arranque:** el PR que introduce estos workflows por primera vez **no puede pasar su propio check** (GitHub no ejecuta un workflow nuevo en el mismo PR que lo agrega). Hay que desactivar temporalmente `required_status_checks` (ponerlo en `null`), aprobar y mergear ese PR inicial, y **recién ahí** volver a activar el required check. Ver sección 8 para más detalle de por qué pasa esto.
7. Si el ticket de este repo nuevo corresponde a otro proyecto de Asana (no "Desarrollo"), hay que ajustar `ASANA_PROJECT_GID` en `link-asana-task.yml` para ese repo — hoy el valor está fijo, no es dinámico por PR.

---

7. Cómo agregar un proyecto de Asana NUEVO (si algún día se necesita)

Hoy la automatización busca únicamente en el proyecto "Desarrollo" (`1209109605438289`). Si se necesita que otro proyecto de Asana también participe:

- **Opción simple:** duplicar la lógica de búsqueda en el script para recorrer una lista de `project_gid` en vez de uno solo, y detenerse en el primero donde encuentre coincidencia.
- Cada proyecto nuevo necesita su propio campo personalizado equivalente a `NEXDEV` (puede tener otro nombre/prefijo, ej. `COMPAX-123` para `compax-frontend`), y el script tendría que aceptar qué prefijo buscar según el repo.
- Esto no está construido todavía — es una extensión pendiente, no algo que ya soporte el código actual.

---

8. Troubleshooting — problemas reales que ya nos pasaron

8.1 "El PR está aprobado pero no me deja mergear" (`mergeable_state: blocked`)

Causa más común: GitHub Actions está deshabilitado a nivel de repo. Nos pasó en 9 de los 12 repos el primer día. **Diagnóstico:** `GET /repos/{owner}/{repo}/actions/permissions` → si `enabled: false`, ese es el problema. El check obligatorio nunca corre, entonces nunca reporta éxito, entonces GitHub deja el PR eternamente "esperando". **Arreglo:** `PUT`

`/repos/{owner}/{repo}/actions/permissions` con `{"enabled": true}`. Para los PRs que ya estaban abiertos y atascados, hace falta generar un evento nuevo para que el workflow corra por primera vez — la forma más simple es editar la descripción del PR (dispara el evento `edited`), no hace falta cerrar y reabrir.

8.2 "Agregué un workflow nuevo pero el check nunca corre en ese mismo PR"

Causa: comportamiento esperado de GitHub Actions — un workflow que se agrega en un PR no se activa para ESE PR, porque todavía no existe en la rama base al momento de evaluar el evento. Sí va a funcionar para cualquier PR que se abra DESPUÉS de que este se mergee. **Arreglo:** desactivar temporalmente `required_status_checks` (ponerlo `null`) antes de mergear el PR que introduce el workflow, y reactivarlo después. Es 100% reversible, no rompe nada — el PR simplemente no puede autoevaluarse a sí mismo la primera vez.

8.3 "Configuré `required_status_checks` con el nombre del check y ahora nada pasa"

Causa: el nombre en `contexts` tiene que coincidir exactamente con el nombre del check tal como lo reporta GitHub Actions (el campo `name:` del job en el YAML). Si no coincide ni una coma, el check nunca se satisface. **Arreglo:** revisar en la pestaña "Checks" de cualquier PR cuál es el nombre exacto que aparece, y usar ese mismo texto literal en `required_status_checks.contexts`. Se puede corregir en cualquier momento desde branch protection sin perder nada — no es destructivo.

8.4 El comentario en Asana no aparece

Posibles causas, en orden de probabilidad:

1. El repo no tiene el secreto `ASANA_PAT` configurado todavía (el workflow se salta el paso silenciosamente, revisar el log del run en la pestaña Actions).
2. El título del PR no tiene el patrón `NEXDEV-###` reconocible.
3. No existe ninguna tarea en el proyecto "Desarrollo" con ese valor exacto en el campo personalizado `NEXDEV`.
4. GitHub Actions deshabilitado en ese repo (ver 8.1).

---

9. Seguridad

- El token `ASANA_PAT` vive únicamente como secreto cifrado de GitHub Actions — nunca se guardó en texto plano en ningún archivo del repo, chat, ni commit.

- El workflow que lo usa (`link-asana-task.yml`) no es un check obligatorio, así que un fallo de autenticación con Asana nunca bloquea el trabajo de desarrollo.
- Si se necesita rotar el token (ej. alguien deja la empresa y era el dueño del token), hay que generarlo de nuevo desde la cuenta de Asana correspondiente y volver a subirlo como secreto en los 12 repos — no hay un paso intermedio que dependa de una sola persona salvo quien administre Asana.

Revision #1
Created 2026-07-29 19:42:08 UTC by Admin
Updated 2026-07-29 19:42:30 UTC by Admin