Integracion Github - Asana

- Flujo de desarrollo GitHub - Asana
- Integración técnica GitHub ↔ Asana — Nexus33
- Estandares para crear un rama en GIT HUB

Flujo de desarrollo GitHub - Asana

Vigente desde: 29 de julio de 2026 **Aplica a:** los 12 repositorios activos bajo `DevelopmentNexus33`
Motivo: cumplimiento SOC2 — trazabilidad de cambios a tickets autorizados + segregación de funciones (nadie aprueba su propio código)

---

1. Repositorios cubiertos

Repo	Rama principal	Rama de desarrollo
nexus33-microservice-nxs	`master`	`development`
nexus33-config-file	`master`	`development`
nexus33-frontend	`master`	`development`
nexus33-app-verix	`main`	`development`
nexus33-web-site-page	`main`	`development`
nexus33-microservice-security	`master`	`development`
nexus33-api-gateway	`master`	`development`
nexus33-authentication-server	`master`	`development`
nexus33-common	`master`	`development`
nexus33-config-server	`master`	`development`
nexus33-eureka-server	`master`	`development`
compax-frontend	`main`	`development`

No incluidos en este flujo (quedan intactos, fuera de alcance por ahora): `nexus33-microservice-nexuspoint`, `nexus33-zipkin-server`, `nexus33-config-file-cityofhighpoint`, `nexus33-config-file-safesky`.

---

2. Qué cambió

A partir de hoy, en **todos** los repos de la tabla:

1. **No hay push directo a `master`/`main` ni a `development`.** Todo cambio entra por Pull Request, sin excepción — ni siquiera el dueño de la cuenta puede saltárselo (`enforce_admins` activo).
2. **Se necesita mínimo 1 aprobación** de otra persona para poder mergear. Nadie puede aprobar su propio PR.
3. **La aprobación tiene que venir de un Code Owner** (archivo `.github/CODEOWNERS` en cada repo — hoy incluye a todos los colaboradores del repo, revisar/ajustar si en el futuro se quiere dividir por área).
4. **Todo PR debe incluir el ticket de Asana en el título**, formato `NEXDEV-123` (número real del campo personalizado de la tarea). Un chequeo automático bloquea el botón de Merge si falta.
5. **No se permite force-push ni borrar** `master`/`main`/`development`.
6. Si se suben commits nuevos después de una aprobación, esa aprobación se descarta automáticamente y hay que volver a aprobar.

---

3. Los 2 campos que importan en un PR

Campo	¿Qué es?	¿Quién lo pone?	¿Bloquea el merge?
Asignado (*Assignee*)	Quién está trabajando en el cambio. Puramente organizativo, no tiene efecto técnico.	Cualquiera, manual (normalmente el autor se auto-asigna)	No
Revisor de código (*Reviewer*)	Quién revisa **y aprueba** el PR. GitHub lo asigna automáticamente según el archivo `.github/CODEOWNERS` de cada repo (hoy: cualquier colaborador del repo). Cuando ese revisor le da clic a "Approve", esa aprobación es la que se exige para poder mergear.	Automático (CODEOWNERS), o manual si hace falta agregar a alguien más	**Sí** — se necesita mínimo 1 aprobación de un revisor

No hay más roles que estos 2 dentro de un PR. (Aparte existe el nivel de acceso de cada persona al repo — Read/Write/Admin, configurado una sola vez en `Settings → Collaborators` — pero eso no cambia de un PR a otro.)

---

4. Cómo referenciar el ticket de Asana

Va en el **título del PR** (la caja de una sola línea arriba de la descripción, no en el cuerpo), en cualquier parte del texto:

```
Recuperar contraseña NEXDEV-123
```

```
NEXDEV-123: recuperar contraseña
```

```
Fix login bug (nexdev-123)
```

El chequeo no distingue mayúsculas/minúsculas. Si falta, el PR se puede crear igual, pero el botón de **Merge queda bloqueado** — se soluciona editando el título del PR ya creado (el check se vuelve a correr solo, no hace falta abrir uno nuevo).

Adicional (recomendado, no forzado por el sistema): pegar también el link de la tarea de Asana en la descripción del PR, para que la integración Asana↔GitHub la muestre en ambos lados.

---

5. Flujo completo — ejemplo real

Caso: ticket `NEXDEV-123`, "Recuperar contraseña", toca `nexus33-authentication-server` y `nexus33-frontend`.

1. **Ticket en Asana** — ya existe con su número `NEXDEV-123`.
2. **Crear rama** desde `development`:
   ```
   git checkout development && git pull
   git checkout -b feature/recuperar-password
   ```
3. **Desarrollar y hacer push** de la rama (esto sí está permitido, no es la rama protegida):
   ```
   git push origin feature/recuperar-password
   ```
4. **Abrir PR** hacia `development`, título: `Recuperar contraseña NEXDEV-123`. Pegar el link de la tarea de Asana en la descripción.
5. **Revisor aprueba** el PR (Approve). Con eso se cumple el mínimo de 1 aprobación + code owner.
6. **Check de ticket pasa** automáticamente (ya tiene `NEXDEV-123` en el título).
7. **Merge a `development`.** Se prueba en el ambiente correspondiente (`nexus33-config-file` tiene configs `test`/`prod` separadas por cliente: `bistatedev`, `cityofhighpoint`, `sempra`, `verix`, `demo`).
8. **Cuando está validado:** PR de release, `development` → `master`, mismas reglas.
9. **Merge a `master`.**
10. **Build/deploy a producción** — disparado por un webhook externo a GitHub *(pendiente documentar el detalle exacto de este último tramo)*.

6. Integración Asana ? GitHub

- Conectada desde Asana: **perfil → Ajustes → Aplicaciones → Ver todas las aplicaciones → GitHub → Conectar**, autorizado contra la cuenta `DevelopmentNexus33`.
- Se conecta proyecto por proyecto en Asana (no hay gestión centralizada de apps en el plan Starter — eso es exclusivo de Enterprise+).
- Requiere ser Admin del workspace de Asana + Admin/dueño de la organización en GitHub.
- Con la tarea vinculada (link pegado en la descripción del PR), Asana refleja el estado del PR como comentario/adjunto en la tarea.

---

7. Preguntas frecuentes

¿Qué pasa si necesito hacer un cambio urgente y no tengo ticket? Crea el ticket en Asana primero (toma segundos) y referencia su número. No hay excepción técnica para saltarse esto — es intencional.

¿El chequeo revisa cada commit? No. Revisa únicamente el **título del PR**, una sola vez por PR (se re-evalúa si editas el título). No es necesario poner el ticket en cada commit individual.

¿Puedo aprobar mi propio PR? No, GitHub no lo permite estructuralmente.

¿Qué pasa si edito el título después de que el check falló? Se vuelve a correr automáticamente. No hace falta cerrar ni recrear el PR.

---

8. Pendiente / roadmap

☐ Documentar qué dispara exactamente el build/deploy a producción tras el merge a `master`.

☐ `generate_changelog.js` actualizado para traer evidencia real de PR (aprobador, fecha, ticket) vía API de GitHub, como reporte de evidencia SOC2.

☐ Evaluar dividir `CODEOWNERS` por carpeta/servicio en vez de "todos aprueban todo".

☐ Secret scanning y 2FA obligatorio — no disponibles en el plan actual (GitHub Pro personal); requeriría migrar a organización con GitHub Advanced Security o Enterprise.

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.

Estandares para crear un rama en GIT HUB

Las Ramas Principales (El Núcleo)

Estas ramas son intocables de forma directa; nadie debería programar sobre ellas, solo recibir *Pull Requests* (PRs).

- **`main` (o `master`)**: Es el código sagrado. Lo que está aquí es la versión estable que está desplegada en el servidor de producción.
- **`develop` (o `dev`)**: Es la rama de integración. Aquí se une todo el código nuevo de los desarrolladores para ser probado en el ambiente de pruebas antes de salir a producción.

Convenciones para Sub-ramas (Trabajo Diario)

Para las ramas donde tú y tu equipo van a tirar código (las que nacen de `develop`), usamos un estándar de **prefijos** seguidos de un slash (`/`) y un nombre muy descriptivo.

- **`feat/` (Nuevas Funcionalidades):** Se usa cuando vas a crear un módulo, una pantalla o un servicio completamente nuevo.
  - *Ejemplo:* `feat/dashboard-pending-tasks`
- **`fix/` o `bugfix/` (Solución de errores):** Se usa para arreglar un comportamiento incorrecto que se detectó durante el desarrollo o las pruebas.
  - *Ejemplo:* `fix/smart-table-rendering-error`
- **`hotfix/` (Urgencias en Producción):** Son parches críticos. Estas ramas nacen directamente de `main` porque hay un error grave en producción que no puede esperar al ciclo normal de desarrollo.
  - *Ejemplo:* `hotfix/login-crash-500`
- **`enhancement/` (Mejoras):** Se usa cuando la funcionalidad ya existe y no tiene errores, pero la vas a optimizar (como hacer un query de SQL más rápido o mejorar el UI/UX).
  - *Ejemplo:* `enhancement/query-performance-form-person`
- **`chore/` (Mantenimiento):** Tareas técnicas que no afectan directamente al usuario final, como actualizar dependencias, cambiar el `pom.xml` o configurar Docker.
  - *Ejemplo:* `chore/update-angular-v11`

Buenas Prácticas (Senior Tips)

- **Todo en minúsculas y separado por guiones medios (`-`):** Nada de usar espacios, ni mayúsculas, ni CamelCase.
- **Sé descriptivo pero conciso:** Ni muy corto (`fix/error`), ni un testamento gigante (`feat/boton-que-hace-click-en-el-dashboard`).
- **Idioma unificado**