api0

GUIDE 02

Mettre ton produit dans Teams sans développer de bot

Tes clients vivent dans Microsoft Teams. Voici comment y mettre ton produit — en conversation, avec les droits de chaque utilisateur — sans écrire, héberger ni maintenir un bot.

Mis à jour le 7 novembre 2026 · Lire en ligne : api0.ai/fr/guides/produit-dans-teams-sans-bot

---

Quand un client B2B demande « vous avez quelque chose dans Teams ? », la réponse honnête est souvent non. Parce qu'un bot Teams, ce n'est pas un petit projet :

- une ressource Azure Bot, une inscription d'app, des secrets qui expirent ;
- un service web qui reçoit les messages du Bot Framework et y répond ;
- un moteur de conversation qui comprend les demandes et appelle ton API ;
- l'authentification de chaque utilisateur, pour qu'il n'agisse qu'avec ses droits ;
- un manifest d'app Teams, des icônes, une validation par l'admin du client ;
- et la maintenance de tout ça.

Avec api0, tu gardes la seule partie que Microsoft impose — la ressource Azure Bot — et le reste est fourni.

Comment ça marche

```
Utilisateur Teams ──► Azure Bot (le tien) ──► api0 ──► ton API
                                                 │
                                    outils, identité, journal, crédits
```

1. L'utilisateur écrit au bot dans une conversation privée Teams.
2. Microsoft transmet le message au **messaging endpoint** fourni par api0.
3. L'assistant d'api0 comprend la demande et appelle les **outils** créés à partir de ton API — au nom de cet utilisateur.
4. La réponse revient dans Teams.

Ce sont les mêmes outils que dans Claude ou WhatsApp : tu configures ton API une seule fois.

Ce guide présente l'approche. Le pas-à-pas complet, avec l'approbation par l'admin Teams et les erreurs courantes, est dans le guide Teams.

Avant de commencer

- Ton API importée dans api0 (onglet **Tools**) — voir le guide principal.
- L'accès réglé (onglet **Access**) : idéalement, l'identifiant de chaque personne, pour que ton application applique ses propres droits.
- Un accès au **portail Azure** avec le droit de créer une ressource *Azure Bot*.
- Côté client : un admin Teams qui autorise l'app, ou le chargement d'apps personnalisées autorisé.

Étape 1 — Créer la ressource Azure Bot

1. Portail Azure → *Créer une ressource* → **Azure Bot**.
2. Nom du bot, abonnement, groupe de ressources. Plan tarifaire : **F0**.
3. **Type d'app : Single Tenant.** Note l'**App ID** (un GUID) et l'**ID du tenant Microsoft**.
4. Dans la ressource → *Configuration* → *Manage Password* → **New client secret**. Copie la valeur immédiatement : Azure ne la montre qu'une fois.

Étape 2 — L'enregistrer dans api0

app.api0.ai → **Channels** → **Teams** :

- App ID, secret, ID du tenant Microsoft ;
- nom du bot (celui que verront tes utilisateurs) ;
- optionnel : un prompt système (ton, longueur des réponses, confirmations).

api0 demande immédiatement un jeton à Microsoft avec ces valeurs. Si l'App ID et le secret ne vont pas ensemble, tu le sais tout de suite, avec un message clair — pas trois jours plus tard en production.

En retour, api0 affiche le **messaging endpoint** de ton bot.

Étape 3 — Brancher Azure sur api0

Dans la ressource Azure Bot :

1. *Configuration* → **Messaging endpoint** → colle l'URL fournie par api0 → Appliquer.
2. *Channels* → **Microsoft Teams** → accepte les conditions → Appliquer.

Étape 4 — Le package d'app Teams

Un bot n'atteint personne tant qu'une app Teams qui le référence n'est pas installée. Cette app est un zip : un `manifest.json` et deux icônes. C'est souvent là que les projets s'arrêtent.

Dans Channels → Teams, api0 **génère le package** à partir de ce qu'il sait déjà : App ID, nom du bot, page « Get started » de ton tenant. Tu le télécharges, c'est tout.

L'app est en scope **personnel** : le bot répond en conversation privée, là où il a du sens.

Tu préfères le préparer toi-même (icônes à ta marque, textes) ? Un template de manifest est fourni.

Étape 5 — Installer et tester

- Pour toi : Teams → *Apps* → *Gérer vos apps* → **Charger une app**.
- Pour une organisation : l'admin Teams la publie depuis le *Teams admin center*.

Dans api0, le bouton **Test** vérifie la chaîne. Écris ensuite au bot : au premier message, il envoie à la personne un lien pour **relier son compte**. Une fois relié, chaque action est faite avec son identité.

Éditeur de logiciels : un bot par client

Si tu vends un logiciel à plusieurs entreprises, le modèle le plus propre est **un tenant api0 par client** :

- chaque client a ses outils, ses utilisateurs, ses crédits et son journal ;
- son bot vit dans **son** tenant Azure (Single Tenant), ce que ses équipes sécurité préfèrent ;
- tu factures l'usage client par client.

Tu peux faire la mise en place pour lui (avec un compte Azure de son tenant Microsoft) ou lui envoyer la procédure ci-dessus : c'est une demi-heure côté client.

Ce que ton client y gagne

- Ton produit dans l'outil qu'il ouvre déjà toute la journée.
- Des réponses et des actions en langage naturel, sans formation.
- Chaque action avec les droits de la personne, rien de plus.
- Un journal de chaque appel.

Ce que tu y gagnes

- Pas de bot à écrire, héberger ou maintenir.
- La même configuration sert Claude, WhatsApp et Telegram.
- Un argument de vente concret face aux demandes « IA » de tes clients.

Problèmes fréquents

Symptôme	Cause probable
« App ID invalide » dans api0	Le secret a été collé dans le champ App ID. L'App ID est un GUID.
Microsoft refuse le couple App ID / secret	Secret expiré, ou mauvais ID de tenant Microsoft.
Le bot ne répond pas	Messaging endpoint non collé dans Azure, ou canal Teams non activé.
L'app n'apparaît pas dans Teams	Chargement d'apps personnalisées désactivé : passer par l'admin Teams.