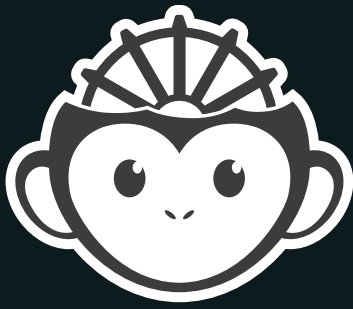




Documentation

Stackmill 1.2.1

Édition du 2026-10-04

Sommaire

Stackmill	5
Votre écran expliqué	6
Démarrer la cellule	8
Choisir une recette et un objectif	10
Les postes de palette	11
Quand elle se met en pause	13
Quand elle est en défaut	14
Arrêts d'urgence et de sécurité	16
Déplacer le robot à la main	17
L'étalonnage	19
Vitesse et charge	20
Fiche opérateur	21
Ce qu'il vous faut	23
Installer sur un poste	24
Installer un serveur Stackmill	25
Ajouter Stackmill à un serveur existant	28
Configurer le routeur 1.2.1	31
Activer une licence	33
Premier lancement	34
Mises à jour	35
Units and frames	37
Robot	38
Tool	39
Position	40

Reference	41
Volume	42
Work zone	43
Container	44
Imported object	45
Programs	46
Recipes	48
Variables	49
Cells, devices and flows	50
Licences	51
The 3D view	52
Objects panel	53
Programs panel	54
Variables panel	55
Code view	56
Menus	57
Settings	58
Live mode	59
Building an operator HMI	60
A full palletisation	61
Depalletising	63
Bundles and interlayers	64
Two stations	65
A calibration program	66
A standard program	67
FANUC controller setup	68

Exporting to the robot	70
Backup and restore	71
FAQ	72
Glossaire	74
Dépannage	76
Sécurité	78
L'assistance à distance	80
Nouveautés	81
Feuille de route	82
Notes de version	84

Stackmill

Créer, simuler et piloter une cellule de palettisation — sans écrire de code robot.



Je conduis une cellule

La démarrer, choisir une recette, et quoi faire quand elle s'arrête.

[Conduire une cellule](#)



Je crée des programmes

Objets, programmes et recettes, puis une palettisation complète de bout en bout.

[Concepts](#) · [Guides](#)



J'installe Stackmill

Sur un poste, sur un serveur, et comment fonctionnent les licences.

[Démarrer](#) · [Les licences](#)

Obtenir le logiciel

Téléchargez la démo sur apemill.com pour l'essayer : elle édite, enregistre et simule, mais n'exporte pas de programme. Une université, un enseignant ou un étudiant est en général éligible à une licence éducation gratuite.

Nous écrire

Une question, un besoin de formation, une fonctionnalité qui vous manque — contact@apemill.com.

Votre écran expliqué

L'écran opérateur ne change jamais de forme. Les quatre mêmes zones sont toujours au même endroit, quoi que fasse la cellule.

La barre d'état

En haut, un bandeau coloré donne l'état de la cellule d'un seul coup d'œil.

Couleur	État	Signification
Gris	Hors ligne	Aucun lien avec le robot. Rien ne bougera.
Bleu clignotant	Initialisation	Démarrage en cours. Attendre la fin.
Vert	Prêt	Armé, en attente de Démarrer.
Turquoise	En cours	En production.
Ambre	En pause	Arrêt volontaire. La reprise est possible.
Rouge	Défaut	Arrêté par un problème.
Orange	Annulé	Le cycle a été abandonné. Il faut réarmer.

À côté de l'état figurent la recette en cours, la phase — prise, transfert, dépose, retour au HOME — et la vitesse d'exécution.

La vue palette

Le centre de l'écran, c'est la cellule vue de dessus : chaque poste, ce qui s'y trouve, et où en est sa recette. Chaque poste affiche son état et son comptage, ce qui permet de voir tout de suite lequel se terminera en premier. Le détail est dans [Les postes de palette](#).

Les commandes

Les boutons sont à droite. **Seuls les boutons utilisables à cet instant sont actifs** — si Démarrer est grisé, la cellule vous dit qu'elle n'est pas prête, et la zone de message dit pourquoi. C'est voulu : un opérateur ne devrait jamais avoir à retenir quel bouton est autorisé dans quel état.

Les messages

Sous les commandes, une ligne indique ce que la cellule attend ou ce qui ne va pas, et quoi faire. C'est le texte le plus important de l'écran. Tous les messages qu'un opérateur peut voir sont listés dans [Quand elle est en défaut](#) et [Les postes de palette](#), avec l'action attendue.

Démarrer la cellule

Avant que quoi que ce soit ne bouge

Vérifier que personne n'est à l'intérieur de la barrière et que toutes les protections sont fermées. Le logiciel ne voit pas les personnes. Démarrer une cellule est une décision que vous prenez au sujet d'une machine qui va bouger seule.

Avant de démarrer

- La barrière est fermée et personne n'est à l'intérieur.
- Une palette est présente sur au moins un poste, et c'est la bonne.
- La baie du robot est en **AUTO** avec le mode distant actif. Sinon l'écran affiche Robot pas en mode distant et Démarrer reste inactif.

L'initialisation

Une cellule qui vient d'être mise sous tension, ou qui a été annulée, doit être initialisée avant de pouvoir tourner. L'écran vous guide étape par étape ; aucune étape ne peut être sautée ni prise dans le désordre.

Confirmer la charge

L'initialisation pose une question à laquelle il faut répondre honnêtement : **l'outil est-il vide, ou tient-il une pièce ?**

Le robot n'a aucun moyen de le savoir. Votre réponse détermine sa façon de bouger — un robot qui se croit vide alors qu'il tient un carton bougera comme s'il était léger, et un robot qui se croit chargé alors qu'il est vide ralentira pour rien. Dans le doute, regardez l'outil.

Aller au HOME

Une fois la charge confirmée, le robot retourne à sa position HOME. Il se déplace lentement et sur un trajet vérifié. S'il y est déjà, l'étape passe immédiatement.

Si le robot ne peut pas atteindre le HOME — il est contre quelque chose, ou dans une position que la trajectoire ne couvre pas — l'écran propose les options manuelles décrites dans [Déplacer le robot à la main](#).

Lancer la production

La cellule étant **Prête**, choisir la recette (voir [Choisir une recette et un objectif](#)) et appuyer sur **Démarrer**. La cellule reprend là où la recette l'indique, pas forcément à la première couche : si une palette était à moitié pleine à l'arrêt, elle continue de la remplir.

Choisir une recette et un objectif

Une **recette**, c'est un produit sur un type de palette : la taille du carton, le nombre par couche, la façon dont les couches s'empilent. Changer de produit, c'est changer de recette.

Choisir la recette

Sélectionnez-la dans la liste avant de démarrer. La cellule ne démarrera pas sans, et la zone de message indique la recette n'a jamais été démarrée si vous essayez.

La recette ne peut pas être changée en cours de production. Arrêtez d'abord la cellule, et laissez-la terminer le carton qu'elle tient.

Palettiser ou dépalettiser

La même recette fonctionne dans les deux sens. **Palettiser** construit la pile ; **dépalettiser** la défait, dans l'ordre inverse. Le sens se choisit à côté de la recette.

Définir un objectif

Par défaut la cellule remplit la palette puis s'arrête. Vous pouvez aussi lui demander de s'arrêter plus tôt :

- **Par couche** — s'arrêter quand un nombre de couches est terminé. Pratique pour finir une palette partielle en fin de poste.
- **Par pièce** — s'arrêter après un nombre de cartons.
- **La palette entière** — le comportement par défaut.

L'objectif compte ce qui est posé à partir de maintenant, et l'écran affiche à la fois l'objectif et l'avancement.

Le changer en cours de production

L'objectif peut être changé pendant que la cellule tourne ; la recette non. Un nouvel objectif prend effet à la fin du carton en cours — rien n'est lâché ni abandonné pour lui obéir.

Les postes de palette

Un poste, c'est un emplacement où se trouve une palette. Une cellule en compte un ou deux.

Lire un poste

Ce qu'indique le poste	Ce que ça veut dire	Quoi faire
Aucune palette	Rien n'est posé dessus	Poser une palette vide
À valider	Une palette est là mais personne ne l'a confirmée	La contrôler, puis la valider
Terminée	La recette a fini cette palette	L'évacuer, en poser une vide
Retirée en cours de cycle	Une palette a été enlevée pendant que la cellule s'en servait	En remettre une, puis appuyer sur Reset

Valider une nouvelle palette

Valider, c'est vous qui dites à la cellule cette palette est vide, d'équerre, et du bon type. La cellule ne posera rien sur un poste non validé : le message Palette ... en attente de validation s'affiche et elle attend.

Regardez avant de valider. Une palette cassée, mal placée ou déjà chargée produira une pile qui penche — et la cellule n'a aucun moyen de le voir.

Retirer une palette pleine

Quand une palette est **Terminée**, évacuez-la et posez-en une vide. Le poste repasse à À valider, et la cellule repart dès que vous validez.

Travailler avec deux postes

Avec deux postes, la cellule remplit l'un pendant que vous videz l'autre, et la production ne s'arrête jamais pour un changement de palette. La vue palette montre lequel se remplit et lequel attend. Si les deux sont indisponibles, la cellule attend et affiche En attente d'une palette — elle ne se met pas en défaut.

Tare des pièces 1.2.1

Quand le robot sait peser ce qu'il porte, le menu engrenage d'un poste propose une entrée de plus : **Tare à la prochaine pièce** repèse la prochaine pièce. Un changement de recette le fait tout seul ; utilisez l'entrée quand le produit change sans changer de recette.

Quand elle se met en pause

Une pause n'est pas un défaut. La cellule s'est arrêtée proprement, volontairement, et tout ce qu'elle sait reste vrai.

Pourquoi la pause

Soit quelqu'un a appuyé sur **Pause**, soit la cellule a atteint un point où il lui manque quelque chose — une palette, une pièce, une validation. La zone de message dit laquelle.

Reprendre

Appuyer sur **Reprendre**. Le robot repart d'où il s'est arrêté, dans le même cycle, avec le même carton. Rien n'est refait, rien n'est sauté.

Abandonner

Annuler abandonne le cycle au lieu de le poursuivre. À utiliser quand reprendre n'est pas ce que vous voulez — le mauvais produit est en place, ou quelque chose doit être dégagé dans la cellule.

Après une annulation la cellule est **Annulée** et doit être réarmée avant de pouvoir tourner, donc repasser par [l'initialisation](#). **Une pièce tenue par l'outil y reste** — le réarmement vous posera la question.

Ce qui est conservé

Une pause ne change rien sur la palette. Le comptage, la couche en cours et l'objectif y survivent, et survivent aussi à une annulation. Reprendre ou redémarrer continue la même palette au lieu d'en commencer une nouvelle.

Quand elle est en défaut

Un défaut signifie que la cellule s'est arrêtée parce que quelque chose n'allait pas. Elle reste arrêtée tant que vous ne traitez pas la cause — elle ne repartira jamais toute seule.

Lire le message

L'écran donne le défaut en deux parties : **ce qui s'est passé**, et **quoi faire**. Lisez la seconde d'abord ; elle est écrite pour la situation dans laquelle vous êtes réellement.

Les défauts avec reprise

Certains défauts arrêtent le cycle sans le perdre. Le message indique cycle reprenable, et une fois la cause disparue vous pouvez **Acquitter et reprendre** au même carton.

Message	Quoi faire
Défaut robot — cycle reprenable	Corriger la cause s'il y en a une, puis Acquitter et reprendre.
Robot pas en mode distant	Remettre la baie en AUTO avec le mode distant actif, puis Reset.
En attente d'une palette, d'une pièce ou d'une validation	Rien n'est cassé. Lui donner ce qu'elle demande.

Les défauts sans reprise

D'autres mettent fin au cycle. Corriger la cause, appuyer sur **Reset**, et la cellule revient dans un état à partir duquel on peut réarmer.

Message	Quoi faire
Défaut robot	Lire le détail à l'écran, corriger la cause, puis Reset.
Programme arrêté : ...	La raison est nommée dans le message. La corriger, puis Reset. Le code reste affiché tant que la cellule n'est pas réarmée.
Arrêt contact — le robot a touché quelque chose	Libérer le contact, puis Reset. Voir Arrêts d'urgence et de sécurité .
Charge outil différente de celle déclarée	Regarder ce que tient l'outil, puis corriger la charge à l'initialisation.

Les étapes de récupération

Quand un défaut laisse le robot là où il ne devrait pas être, l'écran vous en fait sortir au lieu de vous laisser deviner : il vous dira si l'outil tient quelque chose, proposera de le libérer, et ramènera le robot au HOME sur un trajet vérifié. Suivez les étapes dans l'ordre.

Demander de l'aide

Arrêtez-vous et appelez quand le message n'est pas dans cette liste, quand le même défaut revient après un reset, ou quand la cellule demande de déplacer le robot à la main et que vous n'y avez pas été formé. **N'ouvrez pas la barrière pour aller voir.**

Arrêts d'urgence et de sécurité

 bouton rouge n'est pas sur cet écran

L'arrêt d'urgence, c'est le bouton physique de la cellule. Utilisez-le dès que quelque chose ne va pas. Rien dans le logiciel ne le remplace, et rien dans le logiciel ne peut passer outre.

Arrêt d'urgence

Tant que l'arrêt d'urgence est enclenché, le robot n'a pas la puissance de bouger et l'écran affiche Arrêt d'urgence actif. Déverrouiller le bouton, puis **réarmer** la cellule — le déverrouiller ne suffit pas à la remettre en service, et c'est délibéré.

Arrêt collaboratif

Un robot collaboratif s'arrête de lui-même quand il touche quelque chose. L'écran affiche Arrêt contact — le robot a touché quelque chose.

Libérer d'abord le contact : dégager ce contre quoi le robot appuie, et s'assurer qu'il ne repose ni sur une personne ni sur une pièce. Puis appuyer sur **Reset**. Si le robot est coincé et ne peut pas être dégagé à la main, utiliser l'option de sortie proposée à l'écran plutôt que de forcer.

Arrêt logiciel

L'écran propose aussi un **Arrêt logiciel**, qui stoppe le mouvement par le logiciel. Il est pratique et immédiat, mais ce **n'est pas** un organe de sécurité : il dépend du logiciel et du réseau. Pour tout ce qui compte, utilisez le bouton rouge.

Réarmer

Après un arrêt d'urgence, un arrêt collaboratif ou une annulation, la cellule doit être réarmée. Le réarmement relance l'[initialisation](#), au cours de laquelle la cellule demande si l'outil tient une pièce — répondez d'après ce que vous voyez, pas d'après votre souvenir.

Déplacer le robot à la main

 Personnel formé uniquement

Tout ce qui suit déplace un robot industriel sous votre commande, avec les protections dans un état qui lui permet de bouger. Si vous n'y avez pas été formé, arrêtez-vous ici et appelez quelqu'un qui l'est.

Qui peut le faire

Le mouvement manuel sert à la mise au point, au dégagement d'un bourrage et à la récupération après un défaut. Il ne fait pas partie de la conduite en production, et les boutons sont masqués dans une session opérateur normale.

Le jog

Le jog déplace un axe, ou l'outil, petit à petit tant que vous maintenez la commande. Deux modes :

- **Articulaire** — un axe à la fois. Prévisible, et le moyen le plus sûr de sortir d'une position délicate.
- **Linéaire** — l'outil se déplace en ligne droite. Plus facile à viser, mais plusieurs axes bougent en même temps.

La vitesse de jog est limitée, et le mouvement s'arrête dès que vous relâchez.

Le mouvement libre

Le mouvement libre desserre les freins pour que le bras puisse être déplacé à la main. L'écran demande confirmation avant — **le bras peut tomber sous son propre poids au desserrage** s'il est déployé ou chargé. Soutenez-le avant de confirmer, et retirez la charge de l'outil si vous le pouvez.

Sortir d'une collision

Si le robot est contre quelque chose et que le mouvement normal est refusé, l'écran propose une sortie dédiée : un déplacement court et lent qui l'écarte du contact. À utiliser plutôt que de jogger à l'aveugle, et plutôt que de forcer le bras.

Revenir au HOME

Une fois le robot dégagé, renvoyez-le au HOME. La cellule vérifie le trajet avant de bouger. Une fois au HOME, la cellule peut être réarmée et la production relancée depuis [Démarrer la cellule](#).

L'étalonnage

L'étalonnage, c'est la façon dont la cellule apprend où se trouve exactement un poste de palette. Une cellule dont les postes ont bougé posera les cartons légèrement à côté, et l'erreur grandit à chaque couche.

Quand réétalonner

- Après que quoi que ce soit ait heurté ou déplacé un poste.
- Après un déplacement du robot ou de son socle.
- Quand la pile penche visiblement, ou que les cartons tombent un peu plus loin à chaque couche.

Dans le doute, étalonnez. Cela coûte quelques minutes et n'a rien de destructif.

Semi-automatique

Vous amenez vous-même le robot sur chaque point de mesure et vous confirmez. Plus lent, mais cela fonctionne quand le poste est loin de là où la cellule l'attend — exactement le cas après un choc.

Automatique

Le robot trouve les points lui-même en touchant le poste. Plus rapide, et le choix normal pour un contrôle de routine. Il faut que le poste soit à peu près là où il était, sinon le palpeur ne trouve rien.

Vérifier le résultat

La cellule affiche la correction mesurée avant de l'appliquer. Quelques millimètres, c'est normal. Une correction nettement plus grande signifie que quelque chose a réellement bougé — allez voir avant d'accepter, car accepter une mauvaise mesure apprend à la cellule une mauvaise position.

Si le palpeur ne trouve rien sur toute sa course, ou si la correction sort de la plage acceptée, la cellule refuse le résultat et le dit. C'est le système qui vous protège : réparez le poste, puis recommencez.

Vitesse et charge

Changer la vitesse

Le réglage de vitesse règle l'allure de la cellule, du pas à pas au régime nominal. Il ne change que la vitesse : **le robot suit exactement le même trajet**. Ralentir est donc toujours sans risque, et c'est le bon réflexe quand on observe une nouvelle recette pour la première fois.

La vitesse n'est volontairement pas conservée d'une production à l'autre. Une cellule ralentie hier pour une bonne raison ne doit pas rester lente aujourd'hui sans que personne ne le sache.

Ce que veut dire la charge

La charge, c'est ce que porte l'outil. Le robot s'en sert pour décider de son accélération et de la façon de tenir une position — un carton lourd ne se déplace pas comme une pince vide.

La cellule vous demande de confirmer la charge à [l'initialisation](#), parce qu'aucun capteur ne peut le lui dire.

Charge incohérente

Charge outil différente de celle déclarée signifie que ce que la cellule croit porter ne correspond pas à ce qu'elle mesure.

Arrêtez-vous et regardez l'outil. Soit il tient quelque chose qu'il ne devrait pas — un carton jamais relâché — soit il est vide alors que la cellule le croit chargé. Corrigez la charge à la prochaine initialisation plutôt que de continuer : un robot qui bouge avec la mauvaise charge est un robot qui bouge avec les mauvais efforts.

Fiche opérateur

Barre d'état

	État	Ce que ça veut dire
	Hors ligne	Aucun lien avec le robot. Rien ne bougera.
	Initialisation	Démarrage en cours. Attendre.
	Prêt	Armé, en attente de Démarrer.
	En cours	En production.
	En pause	Arrêt volontaire, reprise possible.
	Défaut	Arrêté par un problème. Voir ci-dessous.
	Annulé	Cycle abandonné. Réarmer pour repartir.

Si ça s'arrête

Message à l'écran	Quoi faire
Robot déconnecté	Vérifier l'alimentation et le réseau du robot ; la reconnexion est automatique.
Arrêt d'urgence actif	Déverrouiller l'arrêt d'urgence, puis réarmer la cellule.
Arrêt contact — le robot a touché quelque chose	Libérer le contact (dégager le robot), puis appuyer sur Reset .
Défaut robot	Lire le détail à l'écran, corriger la cause, puis appuyer sur Reset .
Défaut robot — cycle reprenable	Corriger la cause si besoin, puis Acquitter et reprendre pour continuer où il s'est arrêté.
Programme arrêté : ...	Corriger la cause, puis Reset . Le code reste tant que la cellule n'est pas réarmée.
Robot pas en mode distant	Passer le contrôleur en AUTO avec le mode distant actif, ou appuyer sur Reset .
Charge outil différente de celle déclarée	Vérifier ce que tient l'outil, puis corriger la charge pendant l'initialisation.

Si ça attend

Message à l'écran	Quoi faire
Palette ... en attente de validation	Contrôler la palette, puis appuyer sur son bouton de validation.
En attente d'une palette	Placer une palette vide sur une station libre.
En attente d'une pièce sur le convoyeur	Rien. Le cycle reprend dès qu'une pièce arrive au point de prise.
En attente de la prochaine opération...	Rien. Normal.
Arrêt demandé — fin de l'étape en cours	Rien. Il s'arrête à la fin de l'étape.

Les boutons

Bouton	Fait quoi
Démarrer	Lance la production avec la recette sélectionnée.
Pause / Reprendre	Arrête en fin de mouvement / repart.
Stop	Termine l'étape en cours, puis s'arrête.
Annuler	Abandonne le cycle. La pièce dans l'outil y reste.
Arrêt logiciel	Arrêt d'urgence logiciel. Ne remplace pas le bouton rouge.
Reset	Acquitte un défaut une fois sa cause disparue.
Réarmer	Remet en service une cellule annulée ou en arrêt d'urgence.
Home	Envoie le robot à sa position HOME.
Vitesse	Change la vitesse d'exécution. Ne change pas la trajectoire.

Les postes de palette

Aucune palette — en poser une · **À valider** — la contrôler, puis valider · **Terminée** — l'évacuer · **Retirée en cours de cycle** — une palette a été enlevée pendant le cycle ; appuyer sur Reset après en avoir remis une.

Ne pas ouvrir la barrière. Si le message n'est pas dans cette liste, ou si le même défaut revient : arrêter la cellule et appeler
Nom Téléphone

Ce qu'il vous faut

L'ordinateur

Stackmill fonctionne sous Windows et sous Linux, sur du matériel ordinaire. Il dessine la cellule en 3D et calcule des trajectoires : il lui faut une vraie carte graphique plutôt qu'un circuit intégré, et davantage de mémoire que de cœurs.

Une installation serveur tourne dans Docker et ne demande rien à la machine hôte au-delà de Docker — voir [Ajouter à un serveur existant](#).

Le module de sécurité

La présence d'un **TPM 2.0** détermine le comportement de la licence, et cela vaut la peine d'être établi avant l'installation plutôt qu'après :

- **Avec un TPM** — la licence se lie à la machine puis fonctionne sans aucun accès internet. C'est la seule configuration dans laquelle le logiciel pilote un vrai robot.
- **Sans** — la licence reste en ligne et le logiciel se signale régulièrement. Une telle machine ne fait que de la simulation.

La plupart des PC industriels et des portables professionnels ont un TPM. [Les licences](#) explique ce qui en découle.

Le robot

Stackmill dialogue avec les baies **FANUC**. La baie doit être joignable sur le réseau et réglée pour accepter les commandes distantes ; les options et la mise en route sont dans [Configurer le contrôleur FANUC](#).

D'autres marques sont à la feuille de route — le driver, le solveur et l'exportateur sont déjà écrits sans dépendance à une marque.

Le réseau

- Entre Stackmill et le robot : un lien Ethernet ordinaire. Les deux doivent pouvoir se joindre ; rien ne transite par nous.
- Entre Stackmill et internet : nécessaire **une fois**, pour activer la licence. Ensuite une machine avec TPM n'en a plus jamais besoin, mais sans accès le logiciel ne peut pas vous signaler qu'une nouvelle version existe.

Installer sur un poste

Téléchargez l'installateur sur apemill.com. C'est le même fichier que la démo : sans licence, il édite, enregistre et simule, et n'exporte pas de programme.

Windows **Linux**

Lancez l'installateur et laissez-vous guider. Il place Stackmill à l'endroit habituel, ajoute un raccourci, et ne demande aucune configuration.

Téléchargez le paquet correspondant à votre distribution et installez-le normalement. Stackmill a besoin d'un environnement graphique — il dessine une vue 3D.

Ce qui est installé

Une application, et un dossier pour votre travail. **Les projets, les réglages et la licence sont en dehors de l'application** : c'est pourquoi une [mise à jour](#) n'y touche jamais, et pourquoi une désinstallation n'emporte pas vos programmes.

Ensuite

[Activer une licence](#), puis [le premier lancement](#).

Installer un serveur Stackmill

Cette procédure transforme un mini PC neuf en serveur Stackmill. Elle prend une vingtaine de minutes, essentiellement d'attente, et ne demande aucune connaissance de Linux : l'installation ne pose aucune question.

Le disque est entièrement effacé. Utilisez une machine qui ne contient rien.

Ce qu'il faut

- Un mini PC x86-64 avec au moins 8 Go de mémoire et un disque de 64 Go, avec le **TPM 2.0** et le **Secure Boot** activés dans le BIOS. Le TPM 2.0 s'appelle souvent Intel PTT ou AMD fTPM.
- Une clé USB de 8 Go ou plus.
- Un câble Ethernet vers un réseau qui distribue les adresses automatiquement, avec un accès internet pour l'activation. Ensuite la machine peut rester hors ligne définitivement.
- Un ordinateur portable sur le même réseau.
- La clé de licence serveur du client, et un accès au portail Apemill.

1. Télécharger l'image

Dans le portail, ouvrez **Serveur** et téléchargez l'ISO. Une longue empreinte est affichée à côté ; gardez la page ouverte, vous pourrez la comparer après l'écriture de la clé.

2. Écrire la clé USB

Utilisez [balenaEtcher](#) (Windows, macOS, Linux) ou Rufus sous Windows. Choisissez l'ISO, choisissez la clé, écrivez. Rien d'autre à régler.

3. Installer

Branchez la clé et le câble Ethernet sur le mini PC et allumez-le. Ouvrez le menu de démarrage — F12, F11, F8 ou Échap selon la marque — et choisissez la clé USB.

Une seule entrée apparaît : **Install Stackmill server — ERASES THE WHOLE DISK**. Elle démarre seule au bout de dix secondes.

L'installation se déroule sans intervention pendant une dizaine de minutes, puis la machine redémarre. Retirez la clé à ce moment-là.

4. Premier démarrage

La machine se configure : elle chiffre le disque, lie la clé à sa propre puce TPM et démarre Stackmill. Laissez-lui deux ou trois minutes.

L'écran affiche ensuite la version, l'adresse à ouvrir, et non activé.

5. Activer

Depuis le portable, ouvrez **http://stackmill.local**. Si ce nom ne répond pas, utilisez l'adresse affichée sur l'écran du mini PC.

Stackmill s'ouvre sur le panneau de licence. Collez la clé de licence serveur et validez. La machine se lie à sa puce TPM, et l'écran du mini PC indique maintenant activé.

L'installation est terminée. Internet n'est plus nécessaire : la licence fonctionne hors ligne définitivement. Seules les mises à jour utilisent à nouveau le réseau.

En cas de problème

Ce que vous voyez	Ce que cela signifie
La licence est refusée faute de TPM	Le TPM 2.0 est désactivé dans le BIOS. Activez-le, redémarrez, réactivez. Rien n'est perdu.
Aucune adresse à l'écran	Le réseau n'en a distribué aucune. Vérifiez le câble, puis le serveur d'adresses du réseau.
<code>stackmill.local</code> ne s'ouvre pas	Certaines machines Windows ne résolvent pas ce nom. Utilisez l'adresse affichée à l'écran.
Deux serveurs Stackmill sur un réseau	Le second répond à <code>stackmill-2.local</code> . Utilisez plutôt les adresses.
La licence est refusée pour machine virtuelle	Stackmill ne fonctionne que sur du matériel réel.

Notes pour l'informatique du site

- Le serveur utilise le port **80** pour l'interface et les ports **22222**, **29750**, **33500**, **38890**, **41800**, **42400** pour la liaison entre l'interface et la cellule robotisée. Tous sur le réseau local uniquement.

- Donnez une adresse fixe à la machine en la réservant dans le serveur d'adresses du réseau. Le serveur n'a rien à reconfigurer.
- Rien n'accepte de connexion venant de l'extérieur du réseau local. L'assistance à distance s'ouvre avec l'accord du client, depuis la machine vers l'extérieur, et le client y met fin quand il le souhaite.
- Les mises à jour ne sont jamais automatiques. Quelqu'un décide, depuis l'interface ou par une commande.

Ajouter Stackmill à un serveur existant

Pour un serveur Linux que le client administre lui-même. Si la machine est dédiée à Stackmill, l'[installation du serveur](#) est plus simple et s'occupe aussi du système.

Ce qu'il faut

- Ubuntu 22.04, Ubuntu 24.04 ou Debian 12 en x86-64, avec un accès root.
- Un **TPM 2.0** (`/dev/tpmrm0`). Sans lui, la licence exige un internet permanent et la machine ne peut que simuler, jamais piloter un vrai robot.
- Une machine physique. Une machine virtuelle est refusée : sa puce de sécurité se copie avec elle.
- Internet pendant l'installation.
- L'identifiant d'organisation et le jeton de registre, sur la page **Serveur** du portail.

Installer

Le portail affiche la commande, déjà remplie pour l'organisation :

```
curl -fsSL https://dist.apemill.com/install.sh | sudo bash -s -- --org <id> --token <jeton>
```

Pour la lire d'abord, téléchargez `install.sh` et son empreinte depuis la même page, vérifiez-la, puis exécutez le script avec les mêmes options.

Il installe Docker si la machine n'en a pas, télécharge l'image Stackmill, vérifie la signature d'Apemill dessus et démarre le service. L'exécuter deux fois ne change rien et ne touche jamais aux données.

Options utiles :

Option	Pour
<code>--http-port 8080</code>	quand quelque chose répond déjà sur le port 80
<code>--data-dir /srv/stackmill</code>	placer les données sur un autre disque
<code>--bundle stackmill-server- <version>.tar</code>	une machine sans internet : téléchargez l'archive depuis le portail et copiez-la
<code>--version 1.4.0</code>	une version précise plutôt que la plus récente
<code>--uninstall</code>	retirer Stackmill en conservant les données

Activer

Ouvrez `http://<adresse du serveur>`, collez la clé de licence serveur dans le panneau de licence et validez. L'activation enregistre aussi l'identifiant dont la machine a besoin pour ses mises à jour : plus rien à saisir ensuite.

Exploiter

```
stackmillctl status      # version, licence, cellule en marche ou non
stackmillctl check      # existe-t-il une version plus récente
sudo stackmillctl update # l'installer : refuse si une cellule tourne, revient en
arrière en cas d'échec
sudo stackmillctl rollback # revenir à la version précédente
stackmillctl logs
sudo stackmillctl backup  # archiver le dossier de données
```

L'interface propose aussi un bouton **Mettre à jour**, qui exécute exactement la même chose.

Sans l'installateur

Stackmill fonctionne avec un simple `docker compose` sur n'importe quel hôte Docker : connectez-vous à `docker.apemill.com` avec l'identifiant d'organisation et le jeton, fixez une version dans le fichier `compose`, démarrez. Il n'y a alors ni `stackmillctl`, ni bouton **Mettre à jour**, ni assistance à distance — la fenêtre de mise à jour affiche les deux commandes à lancer. Demandez le fichier `compose` à Apemill.

Ce qu'il faut sauvegarder

Uniquement le dossier de données, `/opt/stackmill/data` par défaut. Il contient les projets, la licence et les réglages ; tout le reste se réinstalle depuis le portail en quelques minutes.

Le restaurer sur une machine **sans** TPM impose de réactiver la licence : le serveur de licences ne peut pas distinguer une restauration d'une machine dupliquée. Avec un TPM, la restauration fonctionne telle quelle.

Configurer le routeur 1.2.1

Chaque machine Stackmill a son propre routeur. Il relie le robot, le serveur et la tablette, et donne son WiFi à la tablette. Stackmill le configure et modifie son WiFi lui-même : rien d'autre n'est nécessaire.

Routeurs pris en charge

Stackmill configure le **MikroTik hAP ax S**, le routeur livré avec la machine. C'est le seul modèle testé, et Stackmill refuse tout autre modèle : un routeur qui n'est pas un hAP ax S n'est jamais modifié.

Une machine sur un réseau que vous gérez vous-même, ou une machine virtuelle, n'a rien à configurer. Pour utiliser un autre routeur, contactez Apemill : nous ajoutons du nouveau matériel sur demande.

Ce qu'il faut

- Le routeur, neuf ou déjà configuré, relié au serveur Stackmill par un câble réseau, sur l'un de ses ports **2 à 5**.
- Le mot de passe imprimé sur l'étiquette du routeur.
- Une session administrateur dans Stackmill.

Un ordinateur avec Stackmill peut aussi configurer le routeur, branché sur les mêmes ports. La fenêtre affiche alors l'adresse du routeur : **192.168.88.1** pour un routeur neuf.

Configurer

1. Ouvrez **Fichier ► Routeur....** Stackmill trouve le routeur.
2. Saisissez le **mot de passe admin** imprimé sur l'étiquette du routeur.
3. Vérifiez le **WiFi machine** : son nom et son mot de passe.
4. Vérifiez le **pays**, puis **Configurer**.

Le WiFi machine apparaît environ une minute plus tard. Un routeur qu'une autre machine de votre organisation a déjà configuré n'a besoin de rien : Stackmill s'y connecte seul.

Uniquement le routeur de la machine

Ne configurez que le routeur livré avec la machine : Stackmill réécrit son WiFi et ses identifiants.

Plusieurs machines dans un même local

Chaque machine a son propre routeur et son propre WiFi, pour que chaque tablette rejoigne sa machine. Stackmill propose un nom qu'aucune autre machine de votre organisation n'utilise, comme `Stackmill-2`. Choisissez un nom qui désigne la machine, par exemple `Stackmill-Ligne2`.

Modifier le WiFi machine

Dans **Fichier ► Routeur...**, à côté du WiFi machine, cliquez sur **Modifier**, saisissez le nouveau nom et le nouveau mot de passe, puis **Modifier le WiFi**. Le mot de passe de l'étiquette n'est pas nécessaire.

Chaque tablette sur ce WiFi est déconnectée aussitôt. Rejoignez le nouveau WiFi depuis chaque tablette, puis rouvrez Stackmill.

Le portail licence

Stackmill enregistre chaque routeur sur le portail licence, page **Serveur**, carte **Routeur**. Tout le monde dans votre organisation y lit le WiFi machine de chaque routeur ; les administrateurs de votre organisation y trouvent aussi son mot de passe admin. Une modification faite depuis Stackmill y apparaît aussi. Sans internet, la fenêtre affiche **En attente d'internet** et enregistre le routeur dès que la machine est en ligne.

Connecter la tablette

Sur la tablette, rejoignez le WiFi machine avec son mot de passe, puis ouvrez Stackmill. Le WiFi machine est en 5 GHz : la tablette doit le prendre en charge.

Donner internet à la machine

Au choix :

- **Un câble** : branchez le réseau du site sur le port **1** du routeur. Rien d'autre à faire.
- **Le WiFi du site** : dans **Fichier ► Routeur...**, **Rechercher**, choisissez le réseau ou tapez son nom, tapez son mot de passe puis **Connecter**.

Le câble l'emporte quand les deux sont présents. Un réseau qui ne se connecte pas en 45 secondes est abandonné, et la machine revient à celui sur lequel elle était.

Activer une licence

Une clé de licence ressemble à APE-XXXX-XXXX-XXXX . Elle se saisit une fois, et appartient ensuite à cette machine.

Saisir la clé

Ouvrez le panneau de licence, tapez la clé, validez. **La machine doit avoir internet à ce moment-là** — c'est le seul point où elle doit nous joindre.

La suite dépend de la machine, et [Les licences](#) explique pourquoi : avec un module de sécurité la licence se lie à la machine et fonctionne ensuite hors ligne ; sans, elle reste en ligne et le logiciel se signale régulièrement.

Une machine sans internet

Une cellule en atelier n'a souvent aucune route vers l'extérieur. Le panneau de licence produit alors un **fichier de demande**. Portez-le sur n'importe quelle machine connectée, déposez-le sur le portail de licence, et rapportez la réponse. Rien d'autre ne change sur la machine.

Vérifier la licence

La pastille de licence se trouve en haut de la fenêtre. Elle donne l'état, et l'ouvrir donne le détail — ce que la licence autorise et, pour une licence en ligne, sa dernière vérification.

Changer de machine

Une licence suit la machine à laquelle elle est liée : une nouvelle carte, un TPM effacé ou un nouvel ordinateur demandent une nouvelle liaison. Faites-en la demande depuis le panneau de licence : nous l'approuvons en ligne, ou vous transportez un fichier si la machine est hors ligne.

Faites-le **avant** de mettre l'ancienne machine au rebut quand c'est possible. Il est plus rapide de libérer une licence encore en service que de prouver qu'une machine a disparu.

Une licence flottante se déplace sans demande : **Libérer la place** sur l'ancienne machine, puis activez la clé sur la nouvelle.

Premier lancement

L'écran d'accueil

Stackmill s'ouvre sur un écran d'accueil plutôt que sur une vue 3D vide : les projets récents, le projet de démonstration, et de quoi en créer un nouveau. Rien n'est chargé avant votre choix.

Ouvrir le projet de démonstration

Commencez par là, même si vous avez une cellule à construire. Le projet de démonstration est une cellule de palettisation complète et fonctionnelle — robot, outil, palette, recette, programme. Tous les guides de cette documentation se suivent dedans, et toutes les captures y sont prises : ce que vous voyez à l'écran est ce que vous voyez sur la page.

Il est aussi sans risque : c'est une simulation, et rien dedans ne peut faire bouger un vrai robot.

Se connecter à un robot

Un projet ne pilote un robot qu'une fois que vous lui avez dit lequel et où. C'est une [cellule](#), et cela se règle par projet. D'ici là, tout ce que vous faites est simulé — ce qui est la bonne façon de construire un programme.

Où sont rangés les projets

Dans vos documents, pas dans l'application. Un projet est un fichier unique : copiez-le, envoyez-le, mettez-le sous gestion de version. Sur une installation serveur, les projets vivent dans le dossier de données du serveur, pour que tous ceux qui s'y connectent voient la même liste.

Mises à jour

Stackmill cherche une nouvelle version une minute après son démarrage, puis une fois par jour. Rien n'est jamais installé tout seul : une mise à jour est une décision, et sur une cellule en production c'est une décision qui a une date.

La vérification ne bloque rien. Une machine sans internet continue de fonctionner exactement comme avant ; elle n'annonce simplement jamais de nouvelle version.

Où elle apparaît

À côté du badge de licence, et dans [Aide ▶ Stackmill ©](#). Les deux ne sont visibles qu'en session administrateur ou développeur — on ne propose pas de mise à jour à un opérateur qui fait tourner une cellule. Le badge ouvre le détail : version, date et notes de version.

[Aide ▶ Mettre à jour...](#) ouvre la même fenêtre à tout moment.

Installer sur l'application de bureau

La fenêtre installe la mise à jour : le téléchargement est vérifié avec la signature d'Apemill, puis l'application redémarre sur la nouvelle version. Le bouton est désactivé tant qu'une cellule tourne — arrêtez-la d'abord.

Installer sur un serveur

- **Un serveur avec le service hôte** (installé depuis l'ISO ou avec `install.sh`) — cliquez **Mettre à jour**. Le serveur redémarre et tout le monde est déconnecté pendant environ une minute ; la fenêtre affiche la progression, la page se reconnecte, et le résultat apparaît en notification. Elle refuse tant qu'une cellule tourne, vérifie la signature de l'image, et revient d'elle-même à la version précédente si la nouvelle ne démarre pas.
- **Un serveur sans ce service** — la fenêtre affiche la commande à lancer sur la machine, avec les mêmes garde-fous :

```
sudo stackmillctl update
```

Sans `stackmillctl`, mettez le nouveau tag dans `docker-compose.yml` puis lancez `docker compose pull && docker compose up -d`.

- **Dans un navigateur, sur une machine de bureau** — la fenêtre donne le lien de téléchargement ; installez-le sur cette machine.

Installer une version précise

La fenêtre accepte un numéro de version, bêtas comprises. Sur un serveur, **Mettre à jour vers** installe exactement celle-là, avec les mêmes garde-fous ; sans service hôte : `sudo stackmillctl update --to <version>`. Sur un poste, elle télécharge l'installateur de cette version. Une version antérieure est signalée : un fichier enregistré par une version plus récente n'est pas garanti de fonctionner avec elle.

Sauter plusieurs versions

Certaines versions ne s'atteignent pas d'un seul coup, parce que le format des fichiers projet a changé entre-temps. Stackmill indique alors la version par laquelle passer. Installez celle-là, puis la plus récente.

Les projets, la licence et les réglages ne sont jamais touchés par une mise à jour. Sur un serveur ils vivent dans `/data`, que la mise à jour ne remplace pas.

Units and frames

Units

Millimetres and degrees, everywhere, with no exceptions and no unit selector. Masses are kilograms. A field never means anything else than what this paragraph says.

Coordinate frames

A position is meaningless on its own; it means something in a frame.

- **World** — the fixed origin of the scene. Everything can ultimately be expressed in it.
- **User frames** — origins you create, usually one per pallet station or per machine. A [reference](#) object is how you make one.
- **Tool** — carried by the robot, at the [TCP](#). Moving in the tool frame means moving relative to the gripper, whichever way it is pointing.

Working in a user frame is what makes a cell maintainable: when the pallet station is moved by 30 mm, you correct the frame and every position on it follows.

Positions and orientation

A position is a point **and** an orientation — where the tool is and which way it faces. Two positions with the same point and different orientations are different positions, and the robot reaches them differently.

Robot configuration

An industrial arm can often reach the same position in more than one posture: shoulder front or rear, elbow up or down, wrist flipped. These are its **configurations**.

They matter because the robot cannot swap between them while moving. A trajectory that would need to change configuration mid-path is not a trajectory, and the software will tell you so rather than trying.

Robot

The arm. A project usually has one.

Adding a robot

Choose the model from the list. The model carries its geometry, its axis limits and its reach, so choosing correctly matters more than anything you will set afterwards.

Place it where it really is. Everything else in the project is measured from it.

Reach and configuration

A position is reachable if the arm can hold it in a valid [configuration](#) without exceeding an axis limit. The software can preview this: select a position and it will show the robot standing in it, or tell you it cannot.

Use it while designing. A pallet corner that is reachable only in a different configuration from the rest is a cell that will fail on the last box of a layer, not on the first.

The HOME position

HOME is where the robot waits and where recovery brings it back. Choose one that is clear of everything, reachable from anywhere the robot works, and comfortable to look at — an operator will see it more than any other.

Axes and limits

Each axis has a range, and the software will not plan through it. Where a cell is tight, that limit is what a trajectory fails on: the position is reachable, and the route to it is not.

Tool

What the robot grips with, mounted on its flange.

Jaws and opening

Describe the tool as it is: its shape, and how it grips. The opening is what the robot commands to take and release a part, and the software uses it to know when a part is held.

Today the shipped tools are suction; mechanical jaws and cylinder grip are on the [roadmap](#).

The TCP

The **Tool Centre Point** is the exact point the robot positions — usually the middle of the gripping face. It is the single most important measurement in the project: every position is relative to it, so a TCP that is 5 mm out puts every box 5 mm out.

Measure it, do not estimate it. If the stack is consistently offset in one direction, suspect the TCP before anything else.

Payload

The mass the tool carries, and where that mass sits. The robot uses it to decide how hard it can accelerate. Too low and the robot moves aggressively with a heavy box; too high and it crawls for nothing.

The operator confirms the payload at [start-up](#), because no sensor can tell the software whether the tool is currently holding something.

Gripping several parts

A tool can be described as taking more than one part at a time. The recipe then places them as a group, which is what makes a fast cell fast — but every part in the group must be reachable in one motion, so it constrains the pattern.

Position

A place. Positions are the vocabulary a program is written in.

Waypoints

A plain position the robot passes through or stops at: an approach, a clearance point, a place to wait. Waypoints exist to shape a movement — around an obstacle, or into a rack from the right direction.

Object positions

Where a part is picked from. The recipe needs to know exactly where a box arrives, in what orientation, and how the tool should approach it.

Container positions

Where a [container](#) sits — the pallet station. The recipe computes every drop from it, so this is the position that [calibration](#) measures and corrects.

Editing a position

A position can be typed in, or taught by moving the robot there and capturing it. Typed is exact and repeatable; taught is honest about where things really are. Most cells use both: typed for what was designed, taught for what was built.

Positions belong to a [frame](#). Put them in the frame of the thing they describe, and a station that moves takes its positions with it.

Reference

An origin. A reference carries a position and an orientation, and other objects hang off it.

What it is for

To give a group of things one place to be moved from. A pallet station is the usual example: the station's reference holds the container, its approach positions and its drop positions, so correcting the reference corrects the lot.

A cell built on references survives the real world. A cell built with everything in world coordinates has to be re-taught every time something is bumped.

Creating one

Add it, place it, then drag onto it whatever belongs to it. Its helper shows the three axes, which is worth leaving on while you build: an origin rotated 90° from what you assumed is very hard to see and very easy to check.

Moving a whole branch

Move the reference and everything under it moves. That is the whole point, and it is also the warning: check what is under a reference before you move it.

Volume

A box. The simplest way to tell the scene that something is there.

Size and placement

Three dimensions and a position. Use volumes for the things that occupy space without needing to be modelled in detail: a fence, a conveyor, a control cabinet, a machine the robot must reach around.

Collisions

A volume is most useful with collision on: it becomes an obstacle the planner must route around. That is what turns "the robot might hit the cabinet" into "the software will not plan through the cabinet".

Using it as an obstacle

Model obstacles slightly generously. A volume 20 mm larger than the real cabinet costs nothing and buys margin; a volume 20 mm smaller is a collision the software promised would not happen.

For anything shaped where a box is too crude, [import the geometry](#) instead.

Work zone

A volume with a rule attached: the robot must stay inside it, or must stay out of it.

What it protects

Work zones express the limits of a cell that are not physical objects — the reach you are willing to give the robot, the corridor a person walks through, the space above a conveyor.

It is not a safety function

A work zone is checked by the software when it plans. It is not a safety device and it does not replace the robot controller's own zone monitoring. See [Safety](#).

Defining one

Give it a size and a position, and say which side of it the robot belongs on. Like every other object it can hang off a [reference](#), which is how a zone follows the station it belongs to.

What happens at the border

A trajectory that would breach the zone is not produced: the software reports it instead of planning around the rule. That is deliberate — a zone quietly ignored is worse than no zone.

Container

What parts are stacked on, or into.

Pallets

The usual container: a footprint, a height, and a position in the cell. The [recipe](#) computes every drop position from the container, so its dimensions have to be the real ones.

A cell normally has one or two container positions — the [stations](#) an operator loads and unloads.

Boxes

Palletising into a box rather than onto a pallet is on the [roadmap](#).

Interlayers

A sheet placed between layers. Where the recipe uses them, it needs to know their thickness: an interlayer that is not declared makes every layer above it land slightly low, and the error accumulates up the stack.

Imported object

Geometry from somewhere else, when a [volume](#) is too crude to be useful.

Supported formats

Mesh geometry — the shapes a 3D file describes. STEP and other CAD assembly formats are on the [roadmap](#); today, convert to a mesh first.

Scale and origin

Two things go wrong with imported geometry, and both are worth checking immediately:

- **Scale** — a file in metres imported as millimetres is a thousand times too big, and it is obvious. A file in inches is 25.4 times out, and it is not.
- **Origin** — CAD files are often exported around an origin that meant something to the designer and nothing to you. Place the object by its origin, then check it against something you know.

Collision meshes

Detailed geometry is expensive to check for collisions. Where an imported object is only ever an obstacle, a simpler shape around it is faster and just as safe — usually a [volume](#).

Programs

The four kinds

Kind	What it is
Palletising	Generated from a recipe . This is what most cells run.
Standard	Written line by line, for logic a recipe does not cover.
Calibration	Measures where something really is. See Calibration .
External	A program that lives on the robot, called from Stackmill.

Programs live in folders and can call one another.

Palletising programs

You do not write these. You describe the product, the pallet and the pattern in a [recipe](#), and the program is generated — every pick, every drop, every approach, in the right order, with the trajectories planned.

That is the point of Stackmill: the part of the job that is arithmetic and repetition is done for you, and the part that is judgement is left to you.

Standard programs

For everything a recipe does not describe: waiting on a signal, driving a conveyor, choosing between two behaviours. Written as a list of instructions.

The instruction set

Instruction	Does
MOVE	Go to a position
SET	Give a variable a value
WAIT	Wait for a condition or a delay
IF	Do something only when a condition holds
FOR / WHILE	Repeat
CALL	Run another program
LBL	A named point in the program
PRINT / COM	A message, and a comment

Conditions are built from [variables](#). There is no text syntax to learn: an instruction is chosen and filled in.

External programs

Where a robot already has a program you want to keep, Stackmill can call it rather than replace it. Useful when commissioning a cell alongside something that already works.

Recipes

A recipe is one product on one kind of pallet. Changing product means changing recipe, and it is the only thing an operator picks before starting.

What a recipe holds

- The part: its dimensions and its mass.
- The container: which [pallet](#) it stacks on.
- The pattern: how parts sit in a layer, and how layers stack.
- The trajectories: how the robot gets from the pick to each drop.

Dispositions

The **disposition** is the arrangement of parts in a layer. Stackmill can generate it: give it the part and the pallet and it produces the arrangements that fit, ordered by how well they use the surface.

Generated dispositions are a starting point, not a verdict. The software optimises area; it does not know that your boxes have a printed face that must point outwards, or that the film wrapper needs a margin.

Different shapes are handled differently — rectangular parts, cylinders, bottles and bundles each have their own way of tiling a layer.

Layers and patterns

A stack is layers, and consecutive layers usually differ: rotating alternate layers is what keeps a pallet stable. The recipe holds the whole sequence, so a pallet is not simply the same layer repeated.

Trajectories

For every drop the robot needs a route: an approach, the placement, a retreat. These are generated with the disposition and checked against the scene, so a pattern that cannot be built without hitting the stack is reported rather than exported.

Variables

Values a program reads and writes, and the way Stackmill and the robot say things to each other.

Types

Booleans, integers, floating-point numbers, text, positions and joint positions — each also available as an array or as a fixed set of choices.

A variable can be marked read-only where it reports something rather than controls it.

User and generated variables

- **User variables** are yours: a counter, a mode, a threshold.
- **Generated variables** are produced by Stackmill from the project — the list of tools, the list of references, the recipes, the programs. They exist so a program can refer to "the current recipe" without hard-coding a number that changes the next time you edit the project.

Prefer a generated variable wherever one exists. A program written against them survives edits that a program written against numbers does not.

FANUC registers

On the robot, values live in registers: numeric registers, position registers, digital inputs and outputs. Stackmill maps what it needs onto them and writes the map when you [export](#).

You rarely need to care which register is which — but when reading a program on the pendant, or when a controller reports an error against a register number, this is the map that explains it.

Cells, devices and flows

A project describes an installation. A **cell** is that project connected to real hardware and running.

What a cell is

A cell names what is really there and how to reach it. Until a project has one, everything it does is simulated — which is the right way to build a program, and the reason the demo project is safe to open on any machine.

Devices

A device is one thing the cell talks to: a robot, or a person at a screen. A robot device carries its address and the programs it can run. Adding a device is how the cell learns that this project drives that robot.

Flows

A flow is logic that runs in the cell rather than on the robot: watch a value, react to it, tell a device to do something. Flows are built from nodes rather than written as code — log a message, set a value, wait for a value to reach a target, run a program on a device.

Use them for the coordination between things. The robot's own motion belongs in a [program](#).

The operator HMI

The screens an operator uses are part of the cell, not of the robot. They are built from a small set of components — a button, a label, a selector — each bound to a variable, so pressing a button on a tablet sets the value a flow or a program is watching.

That is why the same screens open on a laptop, a tablet or the FANUC pendant: they are clients of the cell. What an operator does with them is [Operating a cell](#).

Licences

Une clé de licence (APE-XXXX-XXXX-XXXX) est saisie une fois dans le panneau de licence, machine connectée à internet. La clé est alors liée à cette machine, sauf licence flottante.

Avec un module de sécurité (TPM 2.0)

La licence est liée au TPM de la machine et fonctionne ensuite sans internet. C'est le cas normal d'une cellule de production : le pilotage d'un robot réel n'est possible que sur une telle machine.

Sans module de sécurité

La licence reste en ligne : le logiciel se signale au serveur de licence toutes les heures et ne démarre pas sans lui. Une telle machine ne fait que de la simulation.

Licence flottante

Une licence flottante n'appartient à aucune machine : chaque machine qui l'active prend une place, dans la limite du nombre acheté. Elle reste en ligne, et pilote un robot réel si l'offre le permet. **Libérer la place** ou **Oublier cette licence** dans le panneau de licence libère la place aussitôt ; le portail en libère aussi, et une machine muette pendant 14 jours perd la sienne.

Démo

Sans licence, Stackmill édite, enregistre et simule. L'export est désactivé, et les fichiers enregistrés en démo ne s'ouvrent que sur la machine qui les a créés.

Éducation

Une licence éducation est une licence normale dont les programmes s'arrêtent après une durée fixe. L'opérateur les relance ; le logiciel est complet pour l'enseignement et inutilisable en production.

Changement de matériel

Après un changement de carte ou une remise à zéro du TPM, le panneau de licence émet une demande qu'Apemill valide en ligne, ou un fichier à transmettre au portail de licence si la machine est hors ligne.

The 3D view

The scene as it is. Everything in the project can be seen, selected and moved here.

Navigating

Orbit, pan and zoom with the mouse. The view is a camera, not a state: moving it changes nothing in the project, so it is always safe to look from somewhere else.

Selecting

Click an object to select it; its details appear in the object panel and its handles appear in the view. Selection is shared across panels — selecting in the tree selects in the view.

The gizmos

A selected object gets handles to move and rotate it. Drag them for a quick placement; type the numbers in the object panel when the value matters. Most cells are built by dragging roughly and then typing exactly.

Display options

Objects can show helpers — frames, jog directions, gripper opening — and collision shapes can be shown as well as solid geometry. All of it is display only: nothing here changes what the robot will do.

Objects panel

The tree

Every object in the scene, arranged as a tree. Children move with their parents — see [Objects](#) for why that matters.

Drag an object onto another to change its parent. Expand and collapse to keep a large cell readable.

Adding an object

The buttons at the top add each type: robot, tool, reference, volume, position, work zone, and import. A new object arrives at the tree root or under the current selection; put it where it belongs before you place it, and its placement will follow the parent.

Object info

The panel beside the tree shows the selected object: its name, type, position, size, and the settings specific to its type. It is also where the switches live — collision on or off, helper shown or hidden, locked to its parent or not.

Locking

Locking an object to its parent stops it being dragged out by accident. Once a cell is correct, lock the things that should never move again.

Programs panel

The list

Every program in the project, in folders. The panel has tabs for programs, variables and configuration.

Folders

Folders are for you, not for the robot. Group by station, by product, by whatever makes the list readable — the export does not care.

Creating a program

Choose the kind when you create it: palletising, standard, calibration or external. The kind cannot be changed afterwards, because each is a different thing — see [Programs](#).

Recipes

A palletising program holds its [recipes](#). Selecting one shows what it produces: the layer, the stack, the count, and the trajectories.

Variables panel

The list

Every variable, with its type, its default and its current value. Generated variables are shown alongside yours but cannot be edited — they come from the project.

Adding a variable

Give it a name and a type. Names are what programs read, so name for what the value means rather than where it is used.

Linking to an object

A variable can point at an object in the scene — a position, a tool, a reference. This is what lets a program say "go to the pick position of the current recipe" and stay correct after the scene is edited. See [Variables](#).

Code view

Where a standard program is written and read.

Reading a line

Each line is one [instruction](#) with its arguments filled in. Lines are numbered, and blocks are indented, so the shape of the program is visible without reading it.

Adding an instruction

Insert a line and choose the instruction; the arguments it needs appear as fields. There is no syntax to get wrong — you cannot write a line that does not parse.

Blocks

`IF`, `FOR` and `WHILE` contain other lines. Lines inside them move with them, which is what makes restructuring a program safe.

Errors

A line that cannot work — a variable that no longer exists, a position that was deleted — is marked. A program with a marked line will not export.

Menus

Menu	Holds
File	New, open, save, and export to the robot
Edit	Undo, redo, and project-wide edits
View	What is shown in the 3D view, and the panel layout
Help	This documentation, the contact form, and Stackmill © with the version and the update check
Admin	Present only in an admin session: the cell's own settings and diagnostics

The licence badge and, in an admin or developer session, the update badge sit beside the menus rather than inside them — they report a state rather than perform an action.

Settings

Group	What it covers
Global	Language, units display, and the defaults a new project starts from
Startup	What happens when Stackmill opens, and which project it opens
Access	Who may do what: an operator session hides everything that edits or moves
Snapshots	How often the project is captured behind you, and how many are kept
Planning	How hard the trajectory planner works before giving up
Live	How the cell behaves when connected: what starts automatically, what needs confirming

Access is the one to set before a cell goes into production. An operator should be given a session that can start, stop, choose a recipe and recover — and nothing that can edit the project or move the robot by hand.

Live mode

The difference between a project you are editing and a cell that is running.

Entering live mode

A project with a [cell](#) can be taken live: Stackmill connects to the devices and the operator panels become real. Until then everything is simulated.

The live panels

Live mode replaces the editing panels with the operating ones — state, pallet view, controls, messages. They are documented from the operator's side in [Operating a cell](#), which is the right page to hand to whoever will use them.

Test mode

A cell can be exercised without the hardware: the panels behave as they would, driven by simulated state. Use it to check an operator screen before commissioning, and to train someone without occupying the cell.

Leaving live mode

Leaving live disconnects the devices and returns to editing. **It does not stop a running cell** — stop the cell first, and check that it has, before you go back to editing a project whose robot is still moving.

Building an operator HMI

The screens an operator sees are built in the project, not written in code.

The editor

A canvas holding the components of one screen. What you place is what an operator gets, on whatever device opens the cell.

The components

A small set on purpose: a **button**, a **label**, a **selector**, and a **separator**. A screen an operator can use under pressure is a screen with few things on it.

Binding to variables

Every component is bound to a [variable](#): a button sets one, a label shows one, a selector chooses among values. Nothing is wired directly to the robot — the variable is the contract, and a [flow](#) or a program does the rest.

Publishing to the panel

The screen is part of the project, so it reaches the panel when the cell does. Design it at the size of the real device — the FANUC tablet pendant, not your monitor — and test it in [test mode](#) before anyone depends on it.

A full palletisation

The whole job, once: from an empty project to a program a robot runs.

What we are building

One product, one pallet, one robot. A cell that picks boxes from a fixed position and stacks them until the pallet is full.

The scene

Add the [robot](#) and place it where it really is. Add a [reference](#) for the pallet station and place that. Everything else hangs off those two.

Model what the robot could hit — the fence, the conveyor, the cabinet — as [volumes](#) with collision on. It costs minutes now and saves a crash later.

The tool

Add the [tool](#) and mount it on the robot. Measure the [TCP](#) rather than estimating it, and enter the payload.

The container

Add the [container](#) under the station's reference, with the pallet's real dimensions. Add the pick position where boxes arrive.

The recipe

Create a palletising program and a [recipe](#) in it. Enter the part: its dimensions and its mass.

The disposition

Let Stackmill generate the layer arrangements, then choose. It orders them by how well they use the pallet; you choose on everything it cannot know — which face must point out, what the wrapper needs, how the boxes actually behave.

Set how layers alternate. Rotating alternate layers is what keeps a stack standing.

The trajectory

Generate the trajectories. Each drop gets an approach, a placement and a retreat, checked against the scene. Anything that cannot be reached, or cannot be reached without a collision, is reported here — before the robot ever moves.

Simulating

Run it. Watch the whole pallet build, and watch the last box of a layer as closely as the first: the corners are where reach and collision problems live.

Exporting

[Export to the robot](#). Stackmill writes the program and the register map.

Running it

Set up the [cell](#), take it live, and run the first pallet **slowly** — see [Speed](#). Then hand [Operating a cell](#) to whoever will run it.

Depalletising

What changes

Almost nothing. The same recipe describes the stack; running it in the depalletising direction takes it apart in reverse order instead of building it.

What does change is the world: a pallet arriving full is a pallet whose contents you did not place, so it may not be exactly where the recipe thinks.

Setting it up

Build the recipe as if you were palletising — see [A full palletisation](#) — then choose the direction. Check the first pick of the top layer and the last pick off the bottom: those are the two the geometry is most likely to get wrong.

Running it

The operator chooses the direction beside the recipe; see [Choosing a recipe and a target](#). A target by layer is more useful here than when building — it is how you take two layers off and stop.

Bundles and interlayers

Bundles

A bundle is several parts handled as one — a shrink-wrapped pack, or several boxes taken in a single grip. The recipe treats the bundle as the unit it places, so the pattern is computed from the bundle's dimensions, not the individual part's.

The tool has to be able to take the whole bundle in one motion. This is where a [multi-part grip](#) is described.

Interlayers

A sheet between layers. Declare its thickness in the recipe: an interlayer the recipe does not know about makes every layer above it land low, and the error adds up over the stack.

Stickers

Where a face must be labelled, the recipe needs to know which face and which way it points — which constrains the disposition. Decide this before choosing a layer pattern, not after: a pattern that puts half the boxes label-inwards is a pattern you cannot use.

Two stations

When two stations help

Two pallet positions let the cell fill one while an operator clears the other, so production does not stop for a pallet change. On a cell that fills a pallet in a few minutes, this is the difference between running and waiting.

Setting them up

Give each station its own [reference](#), and hang its container and positions off it. Two stations means two frames, and correcting one must not touch the other — which is exactly what references give you.

Each station is [calibrated](#) separately.

What the operator sees

Both stations appear in the pallet view, each with its own state and count. The cell chooses which to fill; the operator validates and clears. It is all in [Pallet stations](#), which is the page to hand over.

A calibration program

Calibration measures where something really is, and corrects the project to match.

What it measures

The position of a [container](#) — a pallet station — relative to the robot. That is the measurement everything else on the station depends on.

The probes

A probe is one touch: the robot approaches until it makes contact, and records where. Several probes on known faces are enough to place the station in space.

Probes need somewhere solid and repeatable to touch. Choose faces that are part of the station, not part of what happens to be on it.

Semi-automatic and automatic

- **Semi-automatic** — you bring the robot to each point. Works when the station has moved further than the automatic search will look.
- **Automatic** — the robot finds the points itself. Faster, and the routine choice.

Both are run from the operator screen; see [Calibration](#).

Applying the result

The measured correction is shown before it is applied, and refused outright if the probe found nothing or the correction is outside the accepted band. Look at a large correction before accepting it — accepting a bad measurement teaches the cell a wrong position, and every drop after it inherits the error.

A standard program

For the logic a recipe does not describe.

Creating it

Create a program of kind **standard**. It starts empty, as a list of lines.

The instructions

Add lines and fill in their arguments — the set is in [Programs](#). There is no syntax to learn and no way to write a line that does not parse.

Variables and conditions

Conditions are built from [variables](#). Where you need to refer to something in the project — a tool, a recipe, a position — use the generated variable rather than a number: it survives edits that a number does not.

Calling it from a recipe

A palletising program can call a standard program at defined moments — before a pick, after a drop, at the end of a layer. That is how a cell does something extra without you writing the palletising itself.

FANUC controller setup

What the robot needs before Stackmill can drive it. Do this once, when commissioning.

Controller options

Some of what follows depends on options being present on the controller. Check them before quoting a cell — an option that is missing is a purchase, not a setting. Ask us if you are not sure which your installation needs.

Controller options

Stackmill needs the controller to accept commands over the network and to run the programs it generates. Which option codes that requires depends on the controller generation.

Network

Give the controller a fixed address on the same network as the machine running Stackmill, and check that the two can reach each other before anything else. Most "the robot does not respond" problems are this.

REMOTE mode

The controller must be in **AUTO** with remote enabled, or it will accept nothing. Stackmill reports Robot not in remote mode and refuses to start rather than pretending.

Registers and I/O

Stackmill uses numeric registers, position registers and digital I/O to exchange state with the robot. The map is written when you [export](#). Avoid using the same registers for anything else — a register written by both sides is a fault that will look like something else.

Payload compensation 1.2.1

On a collaborative robot (CRX), Stackmill weighs the part the robot lifts (tare). It needs the **Dynamic payload** setting, first in the robot's **Configuration** tab (on by default), and this controller setup:

1. Set **both payload schedules 1 and 2 to the empty tool**: mass and centre of gravity of the tool alone. Schedule 1 is the empty tool, schedule 2 the loaded one; the robot adds the part to schedule 2 when it weighs it.

2. In the DCS collaborative robot setup, enable **Payload setting compensation** and set **Payload number to compensate** to 2 (see the FANUC collaborative robot manual, How to Estimate Payload Setting Compensation).
3. Apply the DCS changes and cold-start the controller.

Without it, turn **Dynamic payload** off: the payload schedules of the tools and the recipes are used as before, and there is no tare.

Program memory 1.2.1

A cell with many recipes can fill the controller's program memory: a program then fails to load. With FANUC's **TP DRAM/FILE Storage** option installed, tick **Shadow memory** in the robot's **Configuration** tab and export again. The generated programs are then stored outside the program memory and loaded when first run; the programs you edit stay where they are.

Without the option, leave it off: the controller ignores or refuses the setting.

Loading the Karel programs

Some functions run on the controller itself and are loaded once. After loading, cold-start the controller so it picks them up.

Exporting to the robot

What is exported

The robot's own program — the motions, the logic and the positions — plus the register map Stackmill uses to talk to it. Everything the robot needs to run the cell, and nothing that only matters inside Stackmill.

A program with an error will not export. That is deliberate: the last place to discover a broken reference is on the pendant.

Transferring

Send the export to the controller over the network, or carry it. Either way the controller ends up holding the program Stackmill generated.

Checking on the pendant

Look at the program on the pendant before running it — not to edit it, but to confirm that what arrived is what you exported and that the register numbers are the ones you expect.

Do not edit it there. The next export overwrites it, and a change that exists only on the pendant is a change nobody will find. Change it in Stackmill.

Re-exporting

Any change to the scene, the recipe or the program means exporting again. Names matter here: a renamed tool or program can change what the export is called, so check what you are overwriting on the controller.

Backup and restore

What to back up

- **The project file.** It holds the scene, the programs and the recipes — everything you built.
- **The robot's own program**, if it was edited outside Stackmill.
- On a server, the **data folder**, which holds the projects, the settings and the licence.

A project

One file. Copy it, or keep it under version control — a project is worth a history, and "yesterday's version worked" is a question you will eventually ask.

A server

Everything the server keeps lives in its data folder, which an [update](#) never touches. Back that folder up and you can rebuild the machine from nothing.

Restoring

A project opens on any machine with a licence that allows it — with one exception: a file saved in **demo mode** only opens on the machine that saved it.

Restoring a server means installing it again and putting the data folder back. Test that before you need it, not after.

FAQ

Les licences

Stackmill a-t-il besoin d'internet ?

Une fois, pour activer la licence. Ensuite une machine avec module de sécurité n'en a plus jamais besoin. Une machine sans se signale régulièrement et ne démarrera pas si elle n'y arrive pas.

Que se passe-t-il si votre serveur de licence tombe ?

Une machine avec module de sécurité n'est pas affectée. Une machine sans ne peut pas démarrer — c'est une des raisons pour lesquelles une cellule de production devrait avoir un TPM.

Puis-je déplacer ma licence sur une autre machine ?

Oui, en demandant une nouvelle liaison ; une licence flottante n'a besoin que de libérer sa place sur l'ancienne machine. Voir [Activer une licence](#).

Que peut faire la démo ?

Tout sauf exporter. Elle édite, enregistre et simule, et les fichiers qu'elle enregistre ne s'ouvrent que sur la machine qui les a produits.

L'installation

Windows ou Linux ?

Les deux, pour l'application de bureau. Une installation serveur tourne dans Docker.

Ai-je besoin d'un serveur ?

Seulement si vous voulez des écrans opérateur sur tablettes, ou plusieurs personnes dans la même cellule. Une personne qui programme à son bureau n'en a pas besoin.

Les programmes

Dois-je écrire du code robot ?

Non. Stackmill génère le programme propre au robot et vous l'exportez. Ce que contient le programme exporté est dans [Exporter vers le robot](#).

Puis-je modifier le programme généré sur le pupitre ?

Vous pouvez, et il sera écrasé au prochain export. Modifiez-le dans Stackmill.

Le robot

Quels robots sont pris en charge ?

FANUC aujourd'hui. D'autres marques sont à la [feuille de route](#).

Quelles options de baie me faut-il ?

Voir [Configurer le contrôleur FANUC](#).

La cellule

Deux robots peuvent-ils partager une palette ?

Pas aujourd'hui. C'est à la [feuille de route](#).

Combien de palettes une cellule peut-elle avoir ?

Un ou deux postes est le cas normal, et ce autour de quoi les écrans opérateur sont conçus.

Glossaire

Les mots employés par cette documentation et par le logiciel, dans le sens où ils les emploient.

Cellule

Un robot, son outil, les palettes autour de lui et les programmes qui le pilotent — tout ce qui compose une installation en état de marche. C'est aussi ce qu'on démarre et qu'on arrête.

Contenant

Ce sur quoi ou dans quoi les pièces sont empilées : une palette, un carton. Voir [Le contenant](#).

E/S (DI / DO)

Entrées et sorties tout-ou-rien — les signaux que la baie robot échange avec le reste de la machine.

Repère

Une référence pour les positions. Une position ne veut rien dire seule ; elle veut dire quelque chose dans un repère. Voir [Unités et repères](#).

HOME

La position où le robot revient entre deux tâches et après une récupération. Choisie pour qu'il soit dégagé de tout.

Jog

Déplacer le robot petit à petit, à la main, en maintenant une commande. Voir [Déplacer le robot à la main](#).

Charge

Ce que porte l'outil. Le robot en a besoin pour savoir comment accélérer. Voir [Vitesse et charge](#).

Pupitre

L'écran portatif ou la tablette rattachée à la baie robot. Les écrans opérateur Stackmill y tournent.

Recette

Un produit sur un type de palette : le carton, le motif, la pile. Voir [Les recettes](#).

Registre

Un emplacement numéroté dans la baie robot contenant une valeur ou une position. Stackmill écrit ceux dont ses programmes ont besoin.

REMOTE

Le mode de la baie dans lequel le robot accepte des commandes extérieures. Sans lui, Stackmill peut parler au robot mais pas le faire bouger.

Poste

Un emplacement de palette dans une cellule. Voir [Les postes de palette](#).

TCP

Tool Centre Point — le point exact de l'outil que le robot positionne. S'il est faux, toutes les positions du projet sont fausses de la même quantité.

TPM

Un module de sécurité dans la machine. Avec lui, une licence fonctionne hors ligne ; sans lui, non. Voir [Les licences](#).

Zone de travail

Un volume dans lequel le robot doit rester, ou dont il doit sortir. Voir [La zone de travail](#).

Dépannage

Les problèmes que rencontre un opérateur sont dans [Quand elle est en défaut](#). Cette page traite de ceux qui vous arrêtent avant même que la cellule tourne.

L'application

Elle ne démarre pas, ou se ferme aussitôt.

Relancez-la une fois. Si cela échoue encore, envoyez-nous ce qu'affichait l'écran et la version.

La vue 3D est noire, ou très lente.

La machine dessine avec le mauvais périphérique graphique. Vérifiez que Stackmill utilise la vraie carte graphique plutôt que le circuit intégré.

La licence

« Serveur de licence injoignable ».

La machine n'arrive pas à nous joindre. Cela ne compte qu'à l'activation et, sur une machine sans module de sécurité, à chaque vérification. Voir [Activer une licence](#).

La licence ne fonctionne plus après un changement matériel.

Une nouvelle carte ou un module de sécurité effacé rompt la liaison. Demandez une nouvelle liaison depuis le panneau de licence.

L'export est désactivé.

Le logiciel tourne en démo. Un fichier enregistré en démo ne s'ouvre d'ailleurs que sur la machine qui l'a enregistré.

La connexion au robot

Stackmill ne voit pas le robot.

Vérifiez d'abord l'alimentation et le réseau — la barre d'état indique Hors ligne et la reconnexion se fait seule dès que le robot répond.

Le robot est joignable mais rien ne bouge.

Presque toujours, la baie n'est pas en mode distant. L'écran affiche Robot pas en mode distant. Voir [Configurer le contrôleur FANUC](#).

La cellule

La pile penche, ou les cartons tombent plus loin à chaque couche.

L'idée que la cellule se fait de la position de la palette a dérivé. [Étalonnez](#).

Le même défaut revient après chaque reset.

Arrêtez-vous et appelez-nous plutôt que de réinitialiser encore. Un défaut qui se répète vous dit quelque chose que le reset ne corrige pas.

Sécurité

Stackmill fait bouger un robot industriel

Tout ce que décrit ce site est un logiciel qui commande une machine capable de blesser gravement. Rien ici ne remplace les équipements de sécurité de votre cellule, la formation des personnes qui l'utilisent, ni votre propre analyse de risques.

Avant toute chose

- Personne à l'intérieur de la barrière tant que la cellule peut bouger.
- L'arrêt d'urgence physique est l'arrêt d'urgence. L'[arrêt logiciel](#) est un confort et dépend d'un logiciel et d'un réseau en état de marche.
- Quiconque peut déplacer le robot à la main doit y avoir été formé. La page correspondante est écrite pour du personnel formé et le dit.

Ce que Stackmill ne fait pas

Ce n'est pas un système de sécurité, et il n'est pas certifié comme tel. Il ne :

- détecte pas les personnes, et ne sait pas si quelqu'un est dans la cellule ;
- remplace pas les protections, les interverrouillages, les barrières immatérielles ni l'automate de sécurité ;
- passe pas outre les fonctions de sécurité de la baie robot, qui restent maîtresses.

La détection de collision, les zones de travail et la validation des trajectoires servent à protéger vos pièces et votre cellule des erreurs de programmation. C'est une aide à la conception, pas un dispositif de protection.

Une simulation n'est pas une preuve

Une simulation est exactement aussi juste que ce qu'on lui a dit. Un robot posé à quelques centimètres de là où le projet le croit, un outil dont les cotes ont été mal saisies, une palette qui n'est pas où la cellule l'imagine : chacun produit une simulation parfaite et une cellule qui ne l'est pas.

Faites tourner un nouveau programme lentement la première fois, et regardez-le. La [vitesse](#) ne change que la vitesse, jamais le trajet : un premier passage lent vous dit donc la vérité sur le trajet.

Vos responsabilités

Mettre en service une cellule industrielle, en évaluer les risques, la protéger et déclarer sa conformité relèvent de celui qui la met en service. Si vous n'êtes pas sûr de ce que cela implique pour votre installation, parlez-nous-en avant la mise en service — cette conversation fait partie de ce que nous vendons.

L'assistance à distance

Parfois le plus rapide est que nous regardions la machine pendant que vous êtes devant.

Demander une session

Écrivez à contact@apemill.com avec la version et ce que vous observez. Si une session est la bonne réponse, nous conviendrons d'un créneau — l'assistance à distance se fait **avec quelqu'un présent devant la machine**, jamais sur une cellule que personne ne regarde.

Ce que nous voyons

Une session nous donne accès à cette machine uniquement, et pour sa durée. Nous l'utilisons pour lire les journaux, regarder le projet et reproduire ce que vous voyez.

Nous n'ouvrons pas de session de notre propre initiative, et une machine sans route vers internet ne peut pas en avoir.

Terminer une session

La session se termine quand vous y mettez fin, ou quand nous le faisons. Si vous voulez qu'elle s'arrête, arrêtez-la : la cellule n'en dépend pas, et rien de ce que nous faisons ne vaut plus que votre certitude sur qui est connecté à votre machine.

Nouveautés

Toutes les fonctionnalités ajoutées depuis Stackmill 1.0, de la plus récente à la plus ancienne. Si vous utilisez une version plus ancienne, voici ce qu'une mise à jour vous apporterait — [Les mises à jour](#) explique comment en prendre une.

Version	Ce que ça apporte	Où
1.2.1	Configurer le routeur	lire
1.2.1	Tare des pièces	lire
1.2.1	Running a calibration program	lire
1.2.1	Program memory	lire

Feuille de route

Nous ne publions plus de dates. Une date sur une feuille de route est une promesse sur un trimestre, et ça n'a jamais été ce qu'on avait besoin de savoir. Voici ce que nous construisons et ce que nous comptons construire, du plus proche au plus lointain.

Ce qui est déjà livré se trouve dans les [notes de version](#) ; ce qui a changé depuis la version que vous utilisez est dans [les nouveautés](#).

En cours

Les travaux engagés.

- **La supervision de production** — pièces comptées, temps de cycle, arrêts et alarmes, conservés dans le temps et exportables en CSV ou JSON.
- **Un tableau de bord de production** — les mêmes chiffres en direct, sur le pupitre opérateur et dans le navigateur.

Prévu

Décidé, pas encore commencé.

- La prise en charge d'autres marques de robots à côté de FANUC.
- L'allemand, dans le logiciel et dans cette documentation.
- Les licences flottantes, pour une équipe qui partage ses machines.
- Une frise de trajectoire, avec le temps de cycle estimé avant que la cellule tourne.
- Les cellules à plusieurs robots travaillant ensemble.
- Le contrôle de stabilité d'une palette terminée.
- Les préhenseurs mécaniques — mors et prise de cylindre. Aujourd'hui l'outil est à ventouses uniquement.
- Un axe linéaire (7^e axe) piloté par le robot.
- L'import d'assemblages CAO (STEP).
- L'import d'une partie d'un autre projet dans celui en cours.
- Plusieurs recettes actives sur différentes zones d'une même cellule.
- Le changement des dimensions d'une pièce sans arrêter la production.
- La maintenance prédictive, à partir des diagnostics du robot.

Sur demande

Des demandes réelles de clients, que nous réalisons quand l'un d'eux en a besoin. Si l'une d'elles est ce qu'il vous faut, [dites-le nous](#) — c'est ce qui la fait remonter.

- La palettisation dans des cartons.
- L'intégration à un système de gestion d'entrepôt (WMS).
- Les convoyeurs et les véhicules à guidage automatique.
- L'import et l'export de plans de palettisation par une API.
- Le chargement de palettes dans un camion.
- L'optimisation d'un plan de stockage.
- La vision 2D et 3D — mesurer les pièces à l'arrivée, corriger la prise.
- Des trajectoires optimisées pour l'énergie plutôt que pour la vitesse.
- Plusieurs robots palettisant sur la même palette.

Notes de version

Ce qui a changé, version par version, de la plus récente à la plus ancienne. Ce qui arrive est sur la [feuille de route](#).

1.2.0

La version actuelle.

- Le pilotage en direct : cellules, équipements et flux, avec une IHM opérateur qui tourne sur tablette ou sur le pupitre FANUC.
- L'installation serveur : une ISO qui installe une machine prête, et un installateur pour un serveur existant.
- Les mises à jour dans l'application, avec une vérification qui ne bloque jamais et une installation qui n'a jamais lieu toute seule.
- Cette documentation.