Skillstinget Deployment Guide

This guide covers deploying the two optional components that run in **your own Azure subscription**: the AI proxy (drafting that never leaves your boundary) and the shadow-skill scanner (Discovery). The Skillstinget app itself is SaaS and needs no deployment; sovereign app hosting is covered in the Admin Guide. Everything here runs under your identity, in your tenant - nothing is hosted by TeamKRM.

Prerequisites

Where things live

The **SkillstingetConfig** list always sits on the **tenant root site** - it is the bootstrap anchor the app reads before any configuration exists. The **catalog site** you choose (the tenant root works, but a dedicated site collection is fully supported and often preferable) hosts the content lists: SkillstingetSkills, SkillstingetPublishLog and SkillstingetDiscoveredSkills. Point the scanner's catalogSiteUrl and the Protect script at the right targets accordingly: the scanner's Sites.Selected grant goes on the **catalog** site; Protect-SkillstingetConfigList runs against the **root** site.

Root-site rights for setup: the admin performing setup needs edit rights on the tenant **root** site (the config anchor). Global Administrator alone is not enough — SharePoint rights are per site. SharePoint admin center → Active sites → root site → Membership → add yourself as Site admin, or Set-SPOUser -IsSiteCollectionAdmin. Regular users never write the config; they only need default read.

Dedicated-site setup: create a Communication site, add "Everyone except external users" to the site **Members** group (the app acts as each signed-in user: readers browse and activate, submitters create items), then set it as the Catalog site in the wizard or Settings — the content lists are created there automatically on first use. Approval rights remain governed by the Entra approver group regardless of site permissions. Note: switching an existing tenant to a new site does not move existing catalog data; choose the site before seeding, or contact us about migration.

Roles: Azure subscription Contributor (or Owner) for resources; Global Administrator for the scanner identity (app registration, admin consent, site grant). **Tools by path:** the portal path needs only a browser; the script path needs Azure CLI (and, as a fallback for the site grant, PnP.PowerShell or Graph Explorer). Every component offers both paths - pick per component.

Component 1: Local AI (Azure OpenAI + proxy)

One Function App plus an Azure OpenAI account in your subscription. Users' drafting prompts go browser → your Function → your Azure OpenAI; the AI key exists only in the Function's settings.

Path A - script (recommended for CLI users)

Download the trio (script, Bicep, code zip) from Settings → Admin tools → "AI in your own Azure" into one folder, then:

```
Unblock-File .\Deploy-SkillstingetAiProxy.ps1
.\Deploy-SkillstingetAiProxy.ps1 -ResourceGroup rg-skillstinget-ai
```

The script discovers a current model with quota in your region, deploys everything, and prints the endpoint URL. Paste it into Settings → AI drafting → AI draft endpoint and click **Test**.

Path B - Deploy to Azure button (no PowerShell)

Click **Deploy to Azure** (Admin tools or the setup wizard). The portal form shows pinned defaults (model, version, capacity, SKU). Unlike the script, the button cannot discover models - if deployment fails on a retired model or missing quota, adjust Model Name/Version (see troubleshooting) or lower Capacity / switch SKU to Standard. After deployment: the Function → runDraft (or the deployment outputs) → Get function URL → paste into the AI draft endpoint → Test.

Allowed Origin: the URL your users open Skillstinget from. The product's own hosts are pre-allowed; sovereign deployments on your own domain must set it (or add later: az functionapp cors add).

Component 2: Shadow-skill scanner (Discovery)

An Azure Function that enumerates enabled users, probes each OneDrive's Documents/Cowork/skills folder via batched Graph calls, and mirrors findings into the SkillstingetDiscoveredSkills list on your catalog site. Weekly by default, plus on-demand. Deployment is **two acts**: identity (Entra) and infrastructure (Azure). Scripts can do both; the button does only Act 2.

Act 1 - identity

Application permissions (all three, application type, admin-consented): Files.Read.All, User.Read.All, Sites.Selected. Then a **site grant** giving the app **manage** access to the catalog site only - manage, not write: creating the findings list requires it.

Script path: Deploy-SkillstingetScanner.ps1 creates the registration, consents, attempts the grant, and prints exact fallback commands if the CLI's Graph token cannot grant (common). **Portal path:** Entra → App registrations → New ("Skillstinget Scanner", single tenant) → copy the Application (client) ID → Certificates & secrets → new secret (copy the *Value* immediately) → API permissions → add the three → Grant admin consent. Check first that no "Skillstinget Scanner" registration already exists - duplicates cause identity confusion.

The site grant (portal path) - Graph Explorer as a Global Administrator (consent Sites.FullControl.All under Modify permissions):

```
GET  https://graph.microsoft.com/v1.0/sites/{hostname}          ← root site
GET  https://graph.microsoft.com/v1.0/sites/{hostname}:/sites/{site}  ← non-root
→ copy the full three-part id, then:
POST https://graph.microsoft.com/v1.0/sites/{site-id}/permissions
{ "roles": ["manage"], "grantedToIdentities": [ { "application": {
  "id": "<scanner client id>", "displayName": "Skillstinget Scanner" } } ] }
```

If a grant already exists at the wrong level, PATCH .../permissions/{id} with { "roles": ["manage"] } instead.

Act 2 - infrastructure

Script path: the same script deploys Bicep + code. **Button path:** Deploy to Azure → paste Client Id and Secret from Act 1, catalog site URL, pick the **Privacy Mode** (see next chapter), deploy. The Function may live in **any** subscription — even one homed in another directory (common with partner subscriptions): set the **Tenant Id** parameter to the directory id of the tenant that owns the site; the credentials decide what gets scanned, not where the compute runs. Then: Function → runScan → Get function URL → paste into Settings → Organization → Scanner scan URL → **Test scanner**. The test runs a five-point preflight (auth, users, site, OneDrive, list rights) and prints prefilled fixes for anything missing. When all green: **Scan now** in Discovery.

Privacy modes

Mode	What is stored	Posture
anonymous (default)	One row per distinct skill: name, description, user COUNT, latest change. No identities, ever.	Aggregates of this kind generally fall outside GDPR scope. Reach owners by broadcast - announce the governed version; owners self-recognize.
pseudonymous	Per-instance rows; identity = salted HMAC token (anon:...). Stable for statistics, unreadable in the list.	Still personal data under GDPR (re-identification possible for salt holders). The product ships no reversal; reversal requires deliberate, Activity-Log-visible access to the Function's settings. Restrict that RBAC group; define a reversal policy.
full	Real names and UPNs per instance.	For dev tenants and environments whose privacy assessment permits direct identification.

Precedence: PRIVACY_MODE app setting on the Function (infrastructure lock: Azure RBAC + Activity Log) → the Discovery privacy dropdown in Settings → Organization (approver-controlled; changes are stamped with who/when in-app, and backed by SharePoint item version history and the Microsoft 365 audit log) → anonymous. **Mode changes self-clean:** the next scan removes data belonging to other modes. **Erasure is emergent:** users who delete skills or leave the organization drop out of findings on the next scan.

Operations

Updating: script-deployed Functions: re-run config-zip with the new code zip. Button-deployed Functions mount a versioned public zip (WEBSITE_RUN_FROM_PACKAGE); when a new version ships, edit that one app setting to the new URL and restart. **Cost:** both Functions run on Consumption (typically under a euro/month at rest); Azure OpenAI bills per token (gpt-5-mini drafting: fractions of a cent per draft); the scanner's Graph calls are free. **Schedule:** SCAN_SCHEDULE app setting, NCRONTAB, default Monday 03:00 UTC.

Scale notes

Discovery groups by distinct skill and renders the 300 most widespread with search - tested comfortably at 40,000 findings. The scanner batches all Graph traffic (20 requests per call, throttle-aware). Very large first scans (10,000+ users) can approach the Consumption plan's 10-minute ceiling: set functionTimeout in host.json to 00:10:00, prefer the weekly timer for first runs (timers are immune to the 230-second HTTP response limit), or use a Premium plan. Steady-state weekly scans are far lighter.

Troubleshooting

Symptom	Cause	Fix
Grant fails: "Invalid hostname for this tenancy"	az tokens follow the SUBSCRIPTION's home directory - partner/MCPP subscriptions are often homed outside your login domain's tenant	Verify with: az rest GET /v1.0/sites/root --query webUrl. Then either sign in to the site's tenant with a subscription homed there, or split: identity via the portal path in the site's tenant, Function anywhere via the button with Tenant Id set to the site's directory
Deployment: "MissingSubscriptionRegistration ... Microsoft.Web"	Fresh subscription has never hosted this resource type - the portal auto-registers providers, CLI deployments do not	az provider register --namespace Microsoft.Web --wait (and Microsoft.Storage); current scripts do this automatically
"Invalid resource group location" on deploy	A resource group with that name already exists in a different region	Delete it, or re-run with -ResourceGroup <new-name> or -Location <its region>; current scripts abort with this hint instead of cascading
Script blocked: "not digitally signed"	Windows execution policy on downloaded files	Unblock-File .\<script>.ps1 once, then run
Script dies on a yellow az warning	PowerShell strict mode turns az stderr into errors (older guide versions)	Use the current scripts - they check exit codes instead
az: "Static site / resource not found"	CLI pointed at another subscription	az account set --subscription <the right one>; verify with az account show
Bad Request granting site permission via az rest	PowerShell strips quotes from inline JSON	Current script writes a temp file and uses --body "@file"
403 accessDenied granting via az rest, even as GA	Azure CLI's Graph token lacks Sites.FullControl.All	Use Graph Explorer (consent the scope) or PnP: Grant-PnPAzureADAppSitePermission -Permissions Manage
PnP Connect fails wanting a ClientId	PnP.PowerShell 2.x requires your own app registration	Register-PnPEntraIDAppForInteractiveLogin once per tenant, then -ClientId <id>
Scan: "Creating SkillstingetDiscoveredSkills failed (403)"	Site grant missing or write-level; list creation needs manage	The error and the Test scanner button print the exact prefilled request - roles: ["manage"]
Scan works but reads succeeded with NO grant	Files.Read.All alone can resolve sites/lists; creation still fails	Expected - create the manage grant; verify permissions are only the intended three
Two identical lists after upgrading	Historic lookup bug (fixed in 2.26.1): Graph filtered on internal names	Upgrade; app auto-uses the oldest list; delete the empty twins (recycle bin holds 93 days)
Button deployment fails: model deprecated / no quota	Templates pin defaults; Azure retires models	Set current Model Name/Version in the form (script path discovers automatically); lower Capacity or SKU Standard for quota

Symptom	Cause	Fix
Scan now / Draft fails only in the browser	Function CORS list lacks the app's origin - or a server 500 wearing a CORS costume	Test scanner button disambiguates; direct curl test bypasses CORS; az functionapp cors add --allowed-origins <origin>
Portal: "error downloading the template"	Template host must send CORS headers (app ≥ 2.32.4), or the domain is blocked on your network	Use the current buttons (canonical host + CORS); fallback: Build your own template → paste the JSON
"Update available" nag that never clears	Catalog markdown carries no version (fixed in 2.31.4)	Upgrade; the banner now appears only when updating actually changes something
Two "Skillstinget Scanner" app registrations	Repeated Act 1 attempts	Keep the one whose Client ID matches the Function's CLIENT_ID setting; delete the twin
New secret doesn't work	The secret's ID column was copied instead of its Value	Create a new secret; copy the Value immediately (shown once)

Uninstalling

Remove-Skillstinget.ps1 (Admin tools) removes everything step-by-step with individual confirmation: Azure resource groups, the scanner registration, and the enterprise app (revoking all consents). Your four lists are deliberately left as your data - the script prints export guidance and the deletion commands; deleted lists sit in the site recycle bin for 93 days. Skills users activated into their own OneDrives are never touched: they keep working in Copilot Cowork without Skillstinget.