Microsoft

Agentic automation adoption guide

Migrate and modernize automation with AI agents

Introduction

Agentic automation marks a transformative leap for modern enterprises, moving beyond traditional automation to deliver intuitive, AI-driven workflows. By integrating intelligent agents into conversational interfaces, organizations can empower users to ask, decide, and act within a single, secure environment.

This whitepaper provides IT leaders with a decision framework, migration strategies, cost modeling, and governance best practices to confidently adopt agentic automation, unlocking innovation and business value while maintaining Microsoft's standards for security and scalability.

What is agentic automation?

Agentic automation refers to using intelligent agents and AI-driven, conversational or autonomous agents as the front-end for executing automated workflows. In Microsoft's ecosystem, you can achieve this at scale with **Microsoft 365 Copilot and Microsoft Copilot Studio**. Agentic automation extends traditional automation by allowing users (or the AI itself) to interact with processes through natural language chat or triggers, with the AI agent orchestrating tasks in real time. Key components of agentic automation include:

1. **AI agents** are prebuilt or custom and can be conversational or autonomous, interacting with users via chat or running intelligently based on reason and triggers that make real-time decisions. The agent understands user requests and can combine language understanding with business logic.
2. **Agent flows** are automated workflows authored in Copilot Studio and triggered by the agent's reasoning. These run within the agent's context and return results instantly. An agent flow might, for example, retrieve data or perform an action in response to a user's question, and then the agent provides the answer or confirmation back to the user in chat.
3. **Computer use** is a tool in Copilot Studio that allows the AI agent to perform UI automation by controlling applications like a human would (i.e., simulated clicks, typing, navigation). This is effectively AI-driven RPA for web or desktop interfaces that lack APIs or where the UI changes frequently. Computer-using agents (CUA) handle brittle or complex UI by leveraging its vision and language understanding, complementing or even replacing traditional RPA in those cases.

In contrast, **traditional automation** refers to the established approach of running workflows. In the Microsoft portfolio, this includes:

1. **Cloud flows** in Power Automate are API-based workflows triggered by schedules, events, or manual invocations, ideal for predictable, repeatable processes that integrate systems and data via connectors. Cloud flows have mature governance (environments, solution management, sharing) and are typically used for back-end or background automation tasks.
2. **Desktop flows** in Power Automate are robotic process automation (RPA) bots that interact with UI on Windows for legacy or UI-only applications. Desktop flows can run unattended or attended, and are suited for stable, legacy system interactions (e.g. filling forms in an old application daily).

Licensing model

1. **Traditional automation** is generally licensed on a **fixed-cost basis** (e.g. per user or per bot with unlimited runs under that plan). This makes it cost-effective for high-volume scenarios but requires up-front planning of capacity.

2. **Agentic automation** is **consumption-based** where organizations pay only when the agent runs (metered in actions or messages). There is no separate Power Automate license needed for agent flows; even if an agent flow uses connectors, those calls are covered by the Copilot consumption cost. This usage-based model is flexible: it can be very economical for low or variable usage scenarios and pilot projects, and it naturally scales with demand.

When to use each: Traditional automation shines in **backend, predictable workloads** that run at scale, whereas agentic automation excels in **interactive, user-facing scenarios** and situations requiring AI-driven adaptability. In practice, most enterprises will use a **hybrid approach** – keeping traditional API automation or RPA for what they do best (high-volume, deterministic tasks) and adding AI agents on top for improved **user experience, ad-hoc queries, and intelligent decision-making** in context. The next sections explore how to choose the right approach for a given scenario and how to combine them effectively.

Decision framework: Choosing agentic vs. traditional automation

One of the first questions IT leaders face is: *"What automation approach is best for my scenario – an AI-driven agent or a traditional workflow?"* The answer depends on the nature of the process. Below is a **chooser table** with key scenarios and the recommended approach for each, to help you decide at a glance:

Scenario / Requirement	Best Approach	Rationale
Chat-first, conversational request – A user asks for something via Teams or an AI suggests an action in chat.	**Agent + agent flow** (Copilot Studio)	Use an AI **agent** to handle the request in natural language and trigger an **agent flow** to fulfill it. *Why:* Ideal for user-initiated or AI-assisted tasks embedded in conversation, where real-time response is needed.
Unstructured or changing UI automation – The process interacts with external websites or apps that lack stable APIs and their UI changes frequently.	**Agent + computer use** (Copilot Studio)	Let the AI agent perform UI automation using **computer use agents (CUA)**. *Why:* The agent can intelligently adapt to minor UI changes. Traditional RPA would be brittle here, while CUA offers resilience with AI vision.

Scheduled or event-driven process – Runs regularly on a timer or trigger, and needs to be shared or managed by many users.	**Cloud flow** (Power Automate)	Implement as a standard **cloud flow**. *Why:* Cloud flows have robust scheduling, broad connector support, and easy sharing/governance for team-wide or organization-wide processes. They excel at reliable, repeatable tasks.
Legacy application or stable desktop UI – Involves a consistent internal system with a GUI (e.g. mainframe app, on-premises system without API).	**Desktop flow** (Power Automate)	Use a deterministic **desktop flow**. *Why:* Power Automate's desktop flows are purpose-built for repetitive UI tasks on legacy systems. They can run in attended or unattended mode.

Mix and match: These recommendations are not silos. Often a **hybrid solution** yields the best outcome. For example, an AI agent might handle the Teams chat interaction with a user, then call a cloud flow in the background to perform a complex integration or update a database. Or an agent could accept a user's request and then trigger traditional automation to manipulate a legacy system's UI. Microsoft's platform allows agents and both traditional and agentic flows to **invoke each other**, so you can combine patterns as needed to meet a scenario's requirements.

When in doubt, start with the **user experience** and process requirements: if users will initiate it through chat or if AI context/decision-making is needed, lean towards an agentic approach; if it must run reliably on a schedule or be reused broadly, a traditional flow may be more appropriate. You can always integrate the two – for instance, an agent provides a conversational **front-end** while the heavy lifting is done by a backend workflow or RPA process.

Key questions to get started

To decide the best approach, ask these questions when evaluating a workflow or talking with business stakeholders:

- **How does the process start?** If someone initiates it via a chat (e.g. typing a request to a bot in Microsoft Teams), that strongly indicates an **agent + agent flow** solution for a chat-first experience. If it runs automatically on a **schedule or external event**, a **cloud flow** is likely more suitable.
- **What system or interface is involved?** If it involves a **stable internal application** or database with an existing UI, a **desktop flow** might suffice. If

it needs to work with **changing websites or UIs without APIs**, consider using the **computer use** tool in Copilot Studio.

- **How often or how intensively does it run?** For a **high-volume, frequent** process, a fixed-license cloud or desktop flow could be more cost-effective and easier to manage. For a **variable or infrequent** process (or a new pilot where usage is uncertain), using agentic automation with consumption-based pricing provides flexibility and lower upfront cost.
- **Do you already have an automation for this?** If **Power Automate flows exist** for parts of the process, you can **reuse them** – an agent flow can call those cloud flows (and through them, invoke desktop flows) rather than rebuilding from scratch. Conversely, if nothing exists yet, you have freedom to start with whichever approach fits best, but plan with best practices from the start.

By focusing on how the process starts, what systems it touches, and how predictable the workload is, you can map any scenario to the right approach. Keep initial discovery conversations high-level and in the language of the business user; their answers to these questions will naturally align with a technical recommendation (agentic vs. traditional, or a hybrid of both).

Migration playbook: Integrating agents without disruption

Adopting agentic automation **does not require throwing away** your existing Power Automate automations. On the contrary, the highest value often comes from a **phased integration**, where Copilot Studio agents are introduced gradually on top of proven automation. This way, you enhance user experiences without risking the reliability of core processes. Below is a five-step migration playbook to help IT teams plan a smooth transition:

Step 1: Inventory and assess current processes

Inventory all current flows – catalog your existing Power Automate cloud flows and desktop flows (you can use the Power Platform admin center for this). Note how each is triggered (manual, scheduled, etc.) and any pain points (e.g. brittle UI steps). Identifying what you already have sets the baseline.

Step 2: Qualify opportunities

Review which processes are good candidates for agentic automation. Look for flows that are user-driven or ad-hoc, processes with variable or unpredictable usage, or

those that struggle with UI changes. Also consider where adding a conversational interface or AI decision-making could significantly improve speed or user satisfaction.

Step 3: Choose a migration path

Decide how to integrate or convert each candidate:

- Convert a cloud flow to an agent flow: If a cloud flow largely meets needs but you want it in Copilot Studio, use the "Convert to agent flow" feature to bring it into the agentic environment.
- Wrap with an agent: Keep the cloud or desktop flow as-is but create an agent that triggers it. The agent can call the flow and then report results back to the user in chat. This wrap approach is low effort and preserves the existing logic.
- Replace RPA with CUA: If a desktop flow automates an external website or highly dynamic UI, consider reimplementing those steps with the computer use tool in an agent, leveraging AI for resilience.
- Hybrid orchestration: Use an agent for the conversational front-end or autonomous orchestration, while leaving heavy processing in a cloud or desktop flow. The agent handles user interaction and calls traditional flows for backend work – making the most of fixed and consumption licensing.

Step 4: Pilot and measure

Implement a pilot project with one or two processes. Deploy an agent for a controlled user group or scenario. Monitor the agent's usage, success rates, and performance. Crucially, measure consumption costs vs. expectations using the Copilot Studio usage telemetry or estimator tool. Gather user feedback on the new experience.

Step 5: Scale and govern

Based on pilot success, scale up gradually. Roll out the agentic solution to more users or additional scenarios but do so with the governance in place. Establish guardrails like managed environments and DLP policies before a broad rollout. Continue to run mission-critical workflows on traditional automation in parallel until the new approach is proven. Over time, optimize the mix of agentic and traditional automation for reliability and innovation

Migration checklist: Below is a summary checklist to ensure a smooth transition (✓ items you should cover):

- ✓ **Inventory current automations** – List all cloud flows and desktop flows (RPA), along with their triggers, owners, and any issues.
- ✓ **Identify agentic candidates** – Flag processes that are good fits for AI agents (chat-driven, highly variable, or UI-fragile scenarios).
- ✓ **Assess AI value-add** – For each candidate, confirm that a conversational interface or AI decision-making would improve the process (e.g. faster request handling, better user experience).
- ✓ **Choose integration path** – Decide for each whether to convert, wrap, replace, or run hybrid. Map out how existing flows will connect to new agent flows.
- ✓ **Run a pilot** – Implement the agentic automation on a small scale first. Measure usage and collect feedback.
- ✓ **Monitor costs** – Use Microsoft's Copilot Studio estimator and actual telemetry to understand consumption costs and compare them to equivalent licensing costs. Adjust your plan if needed (for example, if consumption is higher than expected, you might keep some parts under fixed licensing).
- ✓ **Enable governance** – Before wider rollout, set up managed environments and policies to control who can create agents, which connectors can be used, and spending limits (details in Governance section).
- ✓ **Gradually scale up** – Expand adoption to more users or additional processes once confidence is gained. Maintain a mix of agentic and traditional flows as appropriate; not everything must become an agent. The end state will likely have both coexisting for different needs.

By following this playbook, IT teams can introduce Copilot-based automation **incrementally**, preserving business continuity. You leverage what's already working (existing Power Automate flows) and improve it by layering AI-driven capabilities where they add the most value. The result is a modernized automation portfolio with minimal disruption, allowing you to scale responsibly.

Licensing and cost considerations

Adopting agentic automation involves understanding **consumption-based pricing model** and how it compares with **traditional fixed licensing**. Enterprise IT leaders should plan for costs and licensing as part of an automation strategy. Here's an overview of the two models and guidance on optimizing costs:

- **Consumption-based:** Copilot Studio uses a **pay-as-you-go** pricing model, where you are billed per use (typically per action or message processed by the agent). There is no separate Power Automate license required to run

agent flows. In fact, even if an agent flow uses connectors to systems (the same connectors that Power Automate uses), those calls are covered by the Copilot consumption charges. Organizations can either pay per use (metered billing, e.g. on the order of cents per action or message) or purchase **prepaid capacity** for Copilot Studio if they expect steady usage. This model is **highly cost-efficient for low-volume or variable workloads**: you only pay when the automation actually runs, which means pilots or occasional processes might cost just a few dollars.

- **Fixed licensing:** Power Automate cloud flows and desktop flows are typically licensed on a **per-user or per-bot** basis, with monthly or annual fees. For instance, an organization might pay a set fee and then run unlimited flows under that license (subject to fair usage limits). This **fixed-cost model** is **cost-effective for high-volume, predictable scenarios.**

No double-paying: A common question is whether adding Copilot Studio means paying twice – for Copilot and Power Automate. The answer is that **you do not pay twice for the same workflow**. If you move a workflow entirely into an agent flow, you don't need a Power Automate license for it; you'll just pay the Copilot consumption. If you have an agent that calls a cloud flow, then **the cloud flow uses its existing Power Automate license,** but the agent itself only incurs a small consumption cost for the orchestration. In practice, you will allocate workloads to avoid overlap: keep the high-volume tasks on fixed licenses (cloud or desktop flows) for efficiency, and use consumption for new, user-facing, or lower-volume tasks. Also note that when using agent flows, **premium connectors are included** in the consumption cost. This means moving some processes to Copilot Studio could potentially save on the need for certain Power Platform add-ons.

Licensing summary: In effect, **agentic automation introduces a cloud-like consumption billing in the realm of automation**, whereas traditional automation remains like an owned resource. Many organizations will use **both models side by side**: consumption-based Copilot Studio for flexibility and innovation, and fixed licensing for mission-critical heavy workloads. The choice comes down to balancing **cost predictability vs. utilization**:

- If you value *predictability and unlimited usage*, ensure you have adequate Power Automate licenses for those core processes.
- If you value *flexibility and fine-grained cost control*, leverage Copilot's metered model for workflows that don't run constantly or where you want to start small and scale usage as needed.

Estimating and controlling costs: We recommend modeling a few scenarios with expected volumes to find your cost "sweet spot". If a workflow executes thousands of times per day, calculate the estimated Copilot consumption and see if it approaches the monthly cost of a license – if so, that might remain on Power Automate. For moderate usage, you may find consumption is well under a license cost. Also consider the *qualitative* benefits: an agentic process might accomplish more per run (thanks to

AI) and provide better user outcomes, which can justify its cost beyond pure transaction counts.

To keep costs predictable, Microsoft enables setting up **budgets, alerts, and caps** on Copilot consumption. Admins can configure an environment to cap Copilot Studio usage at a certain amount or send alerts when certain thresholds are reached. In addition, **Copilot credits** can be purchased to cover a bulk of usage at a discount, which is useful if you foresee steady load. Many organizations might use a hybrid cost approach: e.g. buy credits for baseline usage but also allow pay-as-you-go for occasional spikes, with caps to prevent unexpected overruns.

Hybrid scenarios and best practices

In real-world deployments, **agentic and traditional automation often work in tandem**. This hybrid approach allows each technology to play to its strengths: the Copilot agent provides a user-friendly, intelligent interface, and Power Automate flows handle stable backend operations. Below are a couple of illustrative scenarios and best practices that demonstrate combining agentic and traditional tools:

Example 1: IT Helpdesk – Password Reset Chatbot
Scenario: Employees frequently forget passwords and call the helpdesk. This process can be automated.
Solution: Deploy a **Copilot-based IT Helpdesk agent** in Microsoft Teams that handles password reset requests via chat. When a user types "I forgot my password" to the bot, the **agent** (Copilot) uses an agent flow to walk the user through verification steps (e.g., confirm identity with a text or security questions), then calls an existing **Power Automate cloud flow** that interfaces with the directory system (e.g., Azure AD) to actually reset the password and log the event. The agent then reports success back to the user ("Your password has been reset and emailed to you").
Why hybrid: Password resets involve user interaction (best handled by an AI agent conversing in natural language) and a transactional system update (best handled by a reliable, pre-built cloud flow). The combination improves user experience (instant, 24/7 support in chat) while leveraging IT's existing automation for the actual reset procedure. **Cost-wise**, this is efficient: the chat bot only incurs consumption cost when used (maybe a few cents per request), and the heavy lifting is done by a licensed flow that can handle as many resets as needed. This frees up helpdesk staff and provides quicker resolution for employees.

Example 2: HR Onboarding – New Hire Copilot
Scenario: HR wants to streamline onboarding. New employees must submit documents and get accounts set up in multiple systems.
Solution: Create an **Onboarding Copilot agent** that greets new hires in Teams and converses with them to collect required information and documents (passport, tax forms, etc.). The agent uses an **agent flow** to gather and verify inputs from the new hire in a friendly Q&A style. Once all data is collected, the agent either directly uses

connectors (via an agent flow) to create accounts (if simple), or more likely, triggers a set of **Power Automate cloud flows** that HR IT already maintains for provisioning accounts, assigning equipment, and scheduling orientation sessions. The agent can then confirm to the new hire that "Everything is set!" and even answer onboarding FAQs using its knowledge.
Why hybrid: The conversational front-end makes the onboarding experience more engaging and clearer for the new hire (no confusing forms or emails). Meanwhile, IT can reuse its robust backend flows for provisioning in AD, payroll, etc., ensuring nothing is missed. This **reduces manual HR work** and speeds up readiness. From a cost perspective, onboarding tends to come in batches (new hire waves) which are ideal for a consumption model – you don't need to license every possible new hire in advance, you just pay for the agent interactions when they happen. Best of all, this approach scales: whether you onboard 5 people or 500 people in a month, each will get a consistent, automated experience without overwhelming HR staff.

Other use cases: Many other scenarios across departments can benefit similarly:

- **Finance:** An AI agent that handles expense report inquiries – employees ask in chat if an expense is approved or to initiate an approval. The agent can do initial checks (policy via AI), then call an approval flow.
- **Sales:** A Copilot that assists in quote generation by gathering deal info via chat, using GPT to suggest optimized discounts, and then invoking a cloud flow to generate a quote document and CRM entry.
- **Customer Service:** A support chatbot that answers FAQs and when needed triggers a workflow (cloud flow) to create a support ticket or schedule a service call, blending AI-driven answers with traditional process execution.

Best practices for hybrid automation:

- **Use agents for what they do best – conversation and reasoning.** Design the agent to handle the **user interaction, data gathering, and dynamic decision-making**. For instance, let the agent ask clarifying questions, apply AI to parse unstructured input, or even use GPT-based actions for summarization or extraction if needed.
- **Offload heavy or lengthy tasks to flows.** If a task will take a long time (say more than a minute) or requires complex transactions, it's often better done in a cloud or desktop flow to avoid keeping the user waiting in chat. The agent can always respond with "Got it, I'm working on that and will update you when done." and then update the user once the flow completes. This pattern ensures a snappy user experience.
- **Keep it seamless:** Use the integration capabilities – e.g. Power Automate has a connector to call an agent (for cases where a cloud flow might proactively message a user via an agent), and Copilot agents can call flows – to make the boundaries invisible. To the end user or stakeholder, it should feel like one solution. The Microsoft platform supports **passing context between agents and flows** so that, for example, data collected by the agent can be handed

off to the flow, and results from the flow come back to the agent to present to the user.

- **Monitor both sides:** Ensure you have monitoring in place for the agent (e.g. track the number of conversations, success/failure of agent flows) and for the underlying flows (runs, errors). A failure in either should alert the appropriate support team. Often the cloud flows are already monitored; adding monitoring/telemetry for the agent side is important too, so you know how users are engaging and if the AI needs additional training or prompt tuning.

By combining conversational agents with traditional automation, organizations can achieve **new capabilities** that neither approach could deliver alone. Users get the benefit of an AI assistant that can fulfill requests end-to-end, while IT maintains the robustness and governance of the proven automation running behind the scenes. Early adopters have reported higher user satisfaction, reduction in training (since the chat interface guides users), and more agility in automating ad-hoc tasks that previously weren't worth automating via traditional means. Hybrid is not just normal – it's often the optimal strategy.

Governance and security

As enterprise IT introduces Copilot agents alongside Power Automate flows, **governance and security** must remain a top priority. Fortunately, Microsoft's automation ecosystem extends its mature governance capabilities to Copilot Studio, ensuring that organizations can enforce compliance, security, and control at every step. Here are key governance and security considerations and best practices:

- **Managed Environments:** Microsoft Power Platform offers *Managed Environments* which allow administrators to put guardrails around automation assets. This includes who can create agents or flows, how solutions are shared, and even capacity allocation. By using a managed environment for Copilot Studio agents and flows, you can **prevent "agent sprawl"** by requiring proper approval and providing oversight.
- **Data loss prevention (DLP) policies:** DLP policies let you control which connectors and data sources can be used together, to protect sensitive data. These same policies **apply to agent flows** as they do to cloud flows. For example, you might block an agent from combining an internal HR system connector with an external social media connector, just as you would for any flow. Review and update your DLP rules to encompass new connectors or scenarios introduced by Copilot Studio. This ensures that even as users build powerful agents, they cannot inadvertently pipe confidential data to unapproved services.
- **User permissions and access control:** Agents operate with defined permissions – typically either running in the context of the user who invokes

them or under a specific service account. **Agents respect the same Azure AD and role-based access controls** as other services. For instance, if an agent tries to retrieve data from a system and the user doesn't have access to that data, the agent will be blocked. Use **least-privilege principles**: if an agent uses a service account to perform certain actions (to avoid requiring every end-user to have certain permissions), ensure that account has only the minimal rights needed. All actions taken by agents should be traceable to either the requesting user or a managed service identity, for auditing.

- **Auditing and monitoring:** Just as you monitor flow runs, you should monitor agent interactions. The Power Platform Admin Center (PPAC) provides analytics on flow runs, and Copilot Studio will provide logs of agent conversations and actions. Establish routine audits of agent activity – what triggers are used, which flows are they calling, how many messages are they sending – to spot any anomalies or misuse. Additionally, integrate Copilot/Power Automate logs with your SIEM (Security Information and Event Management) system if you have one, to get unified security insights.
- **Cost Governance:** Governance is not only about security but also controlling consumption. Use the admin settings to set **consumption limits or alerts** for Copilot Studio in each environment. For example, you might set a monthly usage cap and receive notifications at 50%, 75%, and 90% of the budget. This prevents accidental overuse (for instance, if someone creates an agent that loops or gets widely adopted overnight). Coupled with Managed Environments, you can even require that new agent creations default to a certain environment that has these controls, rather than personal or uncontrolled environments.
- **Lifecycle Management:** Treat agents as you would any important application or bot. This means having a process for **change management** (testing updates to agent logic in a test environment before production), **documentation** (what the agent does, who to contact if issues), and **ownership** (assign an owner or team responsible for each critical agent).
- **Security of AI Components:** With AI in the mix, consider additional aspects like the content of prompts or knowledge sources the agent uses. Ensure that any enterprise data the agent has access to (through its grounding or plugins) is governed by the same permission checks. If the agent uses external AI services, review those integrations for compliance with your organization's security and privacy standards.

In summary, **all the safeguards that IT expects – administration, governance, security, compliance – are available for agentic automation as for traditional automation**. Use them proactively. Key actions include enabling Managed Environments, applying DLP to agents, monitoring usage, and enforcing least-privilege access. By doing so, you can confidently deploy AI-driven automation while protecting your enterprise's data and maintaining control from day one.

Conclusion and next steps

Agentic automation represents a significant step forward in how enterprises can streamline operations and empower users through AI. By introducing conversational or autonomous agents that work together with traditional automation, organizations can achieve **greater efficiency, improved user satisfaction, and new capabilities** that were previously hard to implement (like on-demand process execution via chat, or AI-driven handling of unstructured tasks). Importantly, this can be done while **leveraging existing investments** – your Power Automate flows, RPA bots, and governance mechanisms are not only still useful, but they are also the foundation on which Copilot builds.

For IT decision-makers evaluating agentic automation, here are some **actionable next steps**:

- **Identify 1–2 pilot use cases:** Look for one or two high-impact scenarios in your organization that fit the agentic model – for example, a helpdesk FAQ bot that could execute actions (like the password reset scenario), or a departmental process with frequent requests that could be handled in chat. Starting with a contained pilot will let you demonstrate value quickly.
- **Engage with Microsoft Copilot Studio:** If you have access to Microsoft 365 Copilot and Copilot Studio, set up a development environment for your team to start building an agent. Microsoft and its partners can provide guidance or even prototypes. Even if Copilot Studio is in early stages, experimenting with it will illuminate how it can integrate with your current systems.
- **Leverage existing flows:** Inventory your current Power Automate automations and think about how an AI agent could front-end them. Plan to **reuse** rather than rebuild – for instance, connect an agent to a flow that already sends weekly reports, so users can ask "Hey Copilot, send me the latest weekly report now" in Teams and trigger it on-demand. This approach minimizes development effort and proves quick wins.
- **Model and monitor costs:** Before broad rollout, use the Copilot Studio estimator or your pilot data to project costs. Identify which workloads will remain on fixed licensing and which on consumption. Set up budgets or purchase the appropriate capacity (e.g., prepaid credits) if needed. Establish cost monitoring from day one so there are no surprises.
- **Implement governance controls early:** Turn on Managed Environments and data loss prevention (DLP) policies for your Copilot Studio environment as you begin the pilot. By baking in security and compliance at the pilot phase, you ensure any issues are caught when the scope is small. It also helps build trust with stakeholders (security officers, compliance, etc.) that this new technology is under control.

- **Educate and train IT teams:** Ensure your IT administrators and relevant developers are familiar with Copilot Studio and how it differs from (and connects to) Power Automate. Microsoft offers documentation and learning paths for Copilot Studio. Understanding its capabilities (and limitations) will help in designing the right solutions and avoiding pitfalls (like trying to use an agent where a simple flow would do, or vice versa).
- **Plan the rollout to users:** When moving from pilot to production, have a rollout plan. This might include user communications about the new "AI assistant" available to help them, collecting feedback channels (so users can report if the Copilot gave a wrong answer or if they have ideas for new skills), and metrics for success (e.g., reduction in support tickets, faster cycle times, user satisfaction scores). A positive introduction can drive adoption, which is key to realizing value from any new technology.

In conclusion, agentic automation with Microsoft is an **evolutionary step** – not a rip-and-replace. It allows enterprises to **augment existing automation with intelligence and natural interfaces**, creating a more adaptable and user-friendly automation landscape. CIOs and IT leaders can dramatically improve how automation serves the business: making it more accessible (through conversational or autonomous AI), more flexible (through AI-driven logic and pay-per-use scaling), and more aligned with modern work (meeting users where they collaborate, like in chat). By following the guidance in this paper – using the right tool for the right job, migrating methodically, minding licensing, combining approaches, and governing effectively – organizations can confidently embrace agentic automation.

To learn more and get started, visit aka.ms/AgenticAutomation/Docs.