

Contents

Understanding Microsoft 365 Environments 2

User Access Control..... 2

Licenses, Limitations & Workarounds 3

Using Microsoft 365 Custom Connectors 4

Power Automate Solutions 5

 Create and use a Solution..... 5

 Effective Design Patterns (recommended practices)..... 6

 Sample Practice Solutions..... 6

 Solution 1: Branch Cash Threshold Alerts (Excel-based)..... 6

 Solution 2: Customer Complaint Acknowledgement & SLA (Excel-based)..... 8

 Solution 3: Multi-File Ingestion Pipeline 10

MS365 Power Automate Collaboration 14

More Examples for Practice 14

 MS365 Power Automate Custom Connectors..... 14

 MS365 Power Automate Collaboration 16

 MS365 Power Automate AI Hub & RPA 17

Microsoft Fabric 18

 The Core Concepts 19

 Guide (The Workflow) 19

 Demonstration Scenarios..... 20

Understanding Microsoft 365 Environments

An Environment is a space to store, manage, and share your organization's business data, apps, and flows. It lives within your M365 Tenant.

The "Default" Environment: Every tenant has one. Every user is automatically a "Maker" here. **Warning:** Do not use this for critical production apps (too easily cluttered).

Types of Environments:

- **Sandbox:** For development and testing. Can be reset or copied.
- **Production:** For live work. Optimized for steady performance.
- **Developer:** For individual use (personally assigned) to test premium features.
- **Dataverse:** Explain that environments often house a "Dataverse" database (the backend for Power Apps/Automate).

Demos

- Go to the **Power Platform Admin Center** (admin.powerplatform.microsoft.com).
- Click on **Environments** in the left navigation pane.
- Point out the "Default" environment and any others listed.
- **Create a Sandbox:**
 - Click **New**.
 - **Name:** "Training Sandbox".
 - **Type:** Select **Sandbox**.
 - **Region:** Select your local region.
 - **Create Database:** Toggle "Yes" (Crucial for showing Dataverse capabilities later).
- Show the difference in URL/ID when switching between environments in the Power Automate portal (top right corner environment picker).

User Access Control

Level 1: Environment Security Roles (The "Gatekeepers"):

- **Environment Admin:** Can do everything (configure DB, add users).
- **Environment Maker:** Can build apps/flows but cannot necessarily see *other people's* data.

Level 2: Flow Sharing (The "Collaborators"):

- **Owners:** Can edit the flow, delete it, and view run history.
- **Run-Only Users:** (Specific to Instant/Button flows) Can *trigger* the flow but cannot edit it or see its history.
- **Service Accounts:** Using a generic account (e.g., "HR Bot") for production flows so they don't break when a specific employee leaves.

Demo

- **Step 1 (Environment Level):**
 - In Admin Center, select your "Training Sandbox".
 - Click **Settings > Users + permissions > Security roles**.
 - Show the list of roles (highlight *Environment Maker*).
- **Step 2 (Flow Level - Ownership):**
 - Go to Power Automate (make.powerautomate.com).
 - Create a simple "Instant" flow (Trigger: Manually trigger a flow). Save it.
 - On the Flow Details page, find the **"Owners"** box (bottom right).
 - Click **Edit** > Add a colleague. Explain that this person can now *break* the flow if they aren't careful.
- **Step 3 (Flow Level - Run-Only):**
 - On the same details page, look for the **"Run-only users"** box (only visible for Instant flows).
 - Click **Edit** > Add a user.
 - **Crucial Step:** Show the "Connections Used" dropdown. You can force the user to "Provide their own connection" (security) or "Use this connection" (convenience).

Licenses, Limitations & Workarounds

Use this comparison table to simplify the complex licensing model:

License Type	Best For	Connector Access	Limits (approx.)
Seeded (Free/M365)	Personal productivity (Email, Teams, SharePoint).	Standard Only (O365)	~6,000 API requests/day
Premium (Per User)	Power users needing SQL, HTTP, or custom connectors.	Premium + Standard	~40,000 API requests/day
Process (Per Flow)	Departmental flows used by <i>many</i> people.	Premium + Standard	~250,000 API requests/day

- **Throttling:** What happens when you hit the limit? The flow slows down, queues, or fails (429 Error).
- **Concurrency:** By default, "Apply to Each" loops run sequentially (slow). You can make them run in parallel.

Demos

- **Step 1 (Check License):**
 - In Power Automate, click the **Gear Icon** (top right) > **View My Licenses**. Show them what they have.
- **Step 2 (Concurrency Workaround):**
 - Open a flow with an **"Apply to Each"** loop.
 - Click the **"..."** on the loop action > **Settings**.
 - Toggle **Concurrency Control** to **On**.
 - Move the slider to 20-50. Explain how this drastically speeds up processing large lists.
- **Step 3 (Architecture Workaround - Child Flows):**
 - *Concept only (unless time permits):* Explain that if a flow is too huge, they should break it into "Child Flows" to distribute the API calls and make it manageable.

- **Step 4 (Logic Apps Workaround):**
 - Show the **Azure Portal**. Explain that "Logic Apps" is the "pay-as-you-go" version of Power Automate.
 - *Scenario:* "If you have a flow that runs once a month but uses a Premium connector, don't pay \$15/month for a license. Use Logic Apps and pay pennies for that one run."

Using Microsoft 365 Custom Connectors

If Power Automate is the factory, a "Connector" is the truck bringing supplies in. If Microsoft doesn't have a truck for your specific supplier (e.g., a niche currency converter), you build a "Custom Connector." It wraps an **API** (Application Programming Interface) in a user-friendly box.

The 3 Main Components:

- **General:** Where does the data live? (Host/URL).
- **Security:** How do we login? (API Key, OAuth, or "No Auth").
- **Definition:** What specific actions can we do? (Get data, Post data).

Prerequisites: You usually need a Premium License (Per User or Per Flow) to use custom connectors.

Demo

For this demo, we will use a free, public API (like reqres.in or jsonplaceholder.typicode.com) so you don't need real API keys.

- **Step 1 (Start the Wizard):**
 - Go to **Data > Custom Connectors** in the left menu.
 - Click **New custom connector > Create from blank**.
 - **Name:** "Demo User Fetcher".
- **Step 2 (General):**
 - **Scheme:** HTTPS.
 - **Host:** reqres.in (Do not include https://).
 - **Base URL:** /
- **Step 3 (Security):**
 - Select **No authentication** (Keep it simple for the first demo).
- **Step 4 (Definition - The "Magic" Part):**
 - Click **Definition > New Action**.
 - **Summary:** "Get User Info".
 - **Operation ID:** GetUser (This is what appears in the flow later).
 - **Request:** Click **Import from sample**.
 - **Verb:** GET.
 - **URL:** https://reqres.in/api/users?page=2
 - Click **Import**.
- **Step 5 (Test):**
 - Click **Create Connector** (top right).
 - Go to the **Test** tab > **New Connection > Create**.
 - Back in the Test tab, click **Test operation**.
 - *Outcome:* Show the students the "200 OK" status and the JSON data appearing in the Body.

Power Automate Solutions

A **Solution** is a **folder with rules** inside the Power Platform that can contain:

- Power Automate **flows**
- Power Apps **apps**
- **Dataverse** tables
- **Environment variables**
- **Connection references**
- Custom connectors

Solutions make it possible to:

- Move automations between **Dev → Test → Prod**
- Avoid breaking flows when credentials change
- Work cleanly in **teams**
- Follow **ALM (Application Lifecycle Management)** best practices

Note: You **cannot convert** an existing “My flow” into a solution-aware flow. You must recreate it inside a Solution.

Create and use a Solution

Step 1: Create a Solution

1. Go to **Power Automate**
2. Left menu → **Solutions**
3. Click **+ New solution**
4. Fill in:
 - Name: Invoice Automation
 - Publisher: default or custom
 - Version: 1.0.0.0

You now have a solution container

Step 2: Create a flow inside the Solution

1. Open the solution
2. **+ New → Automation → Cloud flow**
3. Choose trigger (e.g. *When a file is created in SharePoint*)

Creating the flow **inside** the solution is critical

Step 3: Add an Environment Variable

1. Inside the solution → **+ New → Environment variable**
2. Example:
 - Name: FinanceEmail
 - Type: Text
 - Default: finance@company.com

3. In your flow:
 - Use **Environment Variable** instead of hardcoded email

Step 4: Use Connection References

When you add connectors (Outlook, SharePoint, Teams):

- Power Automate automatically creates **Connection References**
- On import to another environment, users are prompted to rebind

Effective Design Patterns (recommended practices)

Pattern 1: Configurable Flow

Bad

Send email to: finance@company.com

Good

Send email to: Environment Variable → FinanceEmail

Pattern 2: One Solution = One Business Capability

Examples:

- HR Onboarding Automation
- IT Service Desk Automation
- Invoice Processing

Avoid “mega-solutions” with unrelated flows.

Pattern 3: Environment Strategy

Recommended:

- **Dev** – build & test
- **Test/UAT** – user validation
- **Prod** – live automation

Export solution from Dev → Import to Prod.

Sample Practice Solutions

Solution 1: Branch Cash Threshold Alerts (Excel-based)

Business Use-Case (Simple)

Each branch submits **end-of-day cash** into an Excel table.

If cash is **too low or too high**, Treasury is alerted automatically.

Excel file

- Location: SharePoint or OneDrive for Business
- File name: BranchCashReports.xlsx
- Table name: CashReports

Table columns

Column Name Type

BranchCode Text
BusinessDate Date
CashAmount Number
SubmittedBy Text
SubmittedOn DateTime

Solution name: Branch Cash Alerts (Excel)

Components inside the Solution

- Cloud flow: CF_BranchCash_AlertOnSubmit
- (Optional) Child flow: CF_SendAlert
- **Environment Variables**
- **Connection References**

Create the Solution

1. Power Automate → **Solutions**
2. **+ New solution**
3. Name: Branch Cash Alerts (Excel)
4. Version: 1.0.0.0

Create Connection References (Solution-only)

Inside the solution → **New** → **Connection reference**

Create:

- CR_Excel → **Excel Online (Business)**
- CR_Outlook → **Microsoft 365 Outlook**
- CR_Teams → **Microsoft Teams**

These prevent the flow from being tied to *your* account.

Create Environment Variables (Solution-only)

Inside solution → **New** → **Environment variable**

Name	Type	Example Default
EV_MinCashThreshold	Number	500000
EV_MaxCashThreshold	Number	5000000
EV_AlertEmail	Text	treasury@bank.com
EV_TeamsChannelId	Text	<channel-id>
EV_ExcelFilePath	Text	/Shared Documents/BranchCashReports.xlsx
EV_ExcelTableName	Text	CashReports

In **Prod**, only the *Current Value* changes — no flow edits.

Build Cloud Flow: Cash Alert Flow

Create Flow

Inside solution → **New** → **Cloud flow** → **Automated**

- Name: CF_BranchCash_AlertOnSubmit
- Trigger: **Excel Online (Business)**

When a row is added to a table

- File: EV_ExcelFilePath
- Table: EV_ExcelTableName
- Connection reference: CR_Excel

Flow Logic (step-by-step)

Condition – Check Cash

If CashAmount < EV_MinCashThreshold OR CashAmount > EV_MaxCashThreshold
--

If YES → Send Alerts

- Send email (CR_Outlook)
 - To: EV_AlertEmail
 - Subject: Cash Alert – @{BranchCode}
 - Body: Cash details
- Post message to Teams (CR_Teams)
 - Channel: EV_TeamsChannelId

If NO

- Do nothing

Solution 2: Customer Complaint Acknowledgement & SLA (Excel-based)

Business Use-Case (Simple)

Complaints are logged in Excel.

System:

1. Acknowledges customer automatically
2. Notifies Compliance
3. Sends SLA reminders if overdue

Excel file

- File name: CustomerComplaints.xlsx
- Table name: Complaints

Table columns

Column Name	Type
ComplaintID	Number
CustomerName	Text
CustomerEmail	Text
ComplaintText	Text
CreatedOn	DateTime
Status	Text
AcknowledgedOn	DateTime
LastUpdated	DateTime

Solution name: Complaint Intake & SLA (Excel)

Components

- Cloud flow: CF_Complaint_Acknowledge
- Cloud flow: CF_Complaint_SLA_Reminder
- Environment variables
- Connection references

Environment Variables

Name	Type	Default
EV_ExcelComplaintFile	Text	/Shared Documents/CustomerComplaints.xlsx
EV_ExcelComplaintTable	Text	Complaints
EV_ComplianceEmail	Text	compliance@bank.com
EV_SLAHours	Number	48
EV_AckEmailBody	Text	Thank you for contacting the bank...

Flow 1: Complaint Acknowledgement

Trigger

Excel Online (Business) → *When a row is added to a table*

- File: EV_ExcelComplaintFile
- Table: EV_ExcelComplaintTable
- Connection: CR_Excel

Actions

Send acknowledgement email

- To: CustomerEmail
- Body: EV_AckEmailBody
- Connector: CR_Outlook

Notify Compliance

- To: EV_ComplianceEmail
- Subject: New Complaint Logged – ID @{ComplaintID}

Update Excel Row

- Set:
 - Status = Open
 - AcknowledgedOn = utcNow()
 - LastUpdated = utcNow()

Flow 2: SLA Reminder (Scheduled)

Trigger

Scheduled cloud flow

- Every 1 hour (or daily)

Actions

List rows present in table

- File: EV_ExcelComplaintFile
- Table: EV_ExcelComplaintTable

Apply to each

- Condition:

Status = Open

AND

CreatedOn + EV_SLAHours < utcNow()

If overdue

- Send reminder email to EV_ComplianceEmail
- Optionally update LastUpdated

Solution 3: Multi-File Ingestion Pipeline

Automated pipeline that captures 3 file uploads via MS Forms, logs metadata to Excel, and triggers an approval process. Approved files are aggregated and emailed to the target recipient; rejected submissions trigger a notification only. Includes full audit logging.

Phase 1: The Prerequisites

Before opening Power Automate, set up your data sources.

1. Excel Setup:

- Create a new Excel file in OneDrive or SharePoint (e.g., ProjectUploads.xlsx).
- Create a table with these exact columns: ResponseID, Username, UserEmail, SubmissionTime, FileNames, Status.
- **Important:** Select the headers and row below, press Ctrl+T, check "My table has headers," and name the table (e.g., UploadLog) in the **Table Design** tab.

2. Microsoft Forms Setup:

- Create a Form with 3 separate **File Upload** questions.
- Set all three as **Required** (this simplifies the flow logic).

Phase 2: Create the Solution

1. Go to **Power Automate** > **Solutions** (left sidebar).
2. Click **+ New Solution**, name it "File Approval System", and click **Create**.
3. Open the new Solution, click **+ New > Automation > Cloud flow > Automated**.
4. Name it "Approval Workflow" and select the trigger: **When a new response is submitted** (Microsoft Forms).

Phase 3: The Trigger & Data Gathering

1. **Trigger:** Select your Form in the "Form Id" dropdown.
2. **Action:** Add **Get response details**.
 - **Form Id:** Select your form.
 - **Response Id:** Select Response Id from the Dynamic Content.
3. **Action:** Add **Initialize variable** (for Attachments).
 - **Name:** varAttachments
 - **Type:** Array
 - **Value:** (Leave blank)
4. **Action:** Add **Initialize variable** (for File Names).
 - **Name:** varFileNameString
 - **Type:** String
 - **Value:** (Leave blank) - *We will use this to list filenames in the email body/Excel.*

Phase 4: The Merge & Loop

Since you have 3 separate questions, we must merge them into one list to process them efficiently.

1. **Action:** Add a **Compose** action.
 - Click inside **Inputs**, select the **Expression** tab (fx), and paste this exactly:

```
union(json(outputs('Get_response_details')?['body/question1_id']),  
json(outputs('Get_response_details')?['body/question2_id']),  
json(outputs('Get_response_details')?['body/question3_id']))
```

- *Note: You must replace question1_id etc., with the actual dynamic content from your form.*
 - **Easier Method:**
 1. Click Inputs -> Expression. Type union(json(.
 2. Switch to Dynamic Content -> Select **File Question 1**.
 3. Type), json(.
 4. Select **File Question 2**.
 5. Type), json(.
 6. Select **File Question 3**.
 7. Type). Click OK.
2. **Control:** Add **Apply to each**.
 - **Select an output:** Select the **Outputs** of the **Compose** step above.

INSIDE THE LOOP:

A. Parse JSON:

- **Content:** Select Current item.
- **Schema:**

```
{
  "type": "object",
  "properties": {
    "name": {"type": "string"},
    "id": {"type": "string"}
  }
}
```

B. Get file content:

- **File:** Select id (from Parse JSON).

C. Append to array variable (Prepare Email Attachments):

- **Name:** varAttachments
- **Value:**

```
{
  "Name": "@{body('Parse_JSON')['name']}",
  "ContentBytes": "@{body('Get_file_content')['$content']}"
}
```

D. Append to string variable (Prepare Text List):

- **Name:** varFileNameString
- **Value:** @{body('Parse_JSON')['name']}

- (Note: The
 makes it go to a new line in the HTML email later).

Phase 5: Initial Logging & Approval

Outside the loop now.

1. Action: Add a row into a table (Excel Online Business).

- **Table:** UploadLog.
- **ResponseID:** Response Id.
- **Username:** Responders' Email (or Name).
- **FileNames:** varFileNameString.
- **Status:** Type "Pending".

2. Action: Start and wait for an approval.

- **Approval Type:** Approve/Reject - First to respond.
- **Title:** "Review Files from @{outputs('Get_response_details')['body/responderEmail']}".
- **Assigned to:** Enter the approver's email.
- **Item Link:** (Optional) You can link to the Excel file here.

Phase 6: The Condition (The Fork)

1. Control: Add a Condition.

- Logic: Outcome (from Approval step) **is equal to** Approve.

IF YES (Approved):

1. Action: Update a row (Excel).

- Key Column: ResponseID.
- Key Value: Response Id (from Trigger).
- Status: "Approved".

2. Action: Send an email (V2).

- To: Recipient Email.
- Subject: "Files Approved".
- Body (Click "Code View" button):

```
<h3>Submission Approved</h3>
<table border="1" style="border-collapse: collapse;">
<tr><td><b>User:</b></td><td>@{outputs('Get_response_details')['body/responderEmail']}</td></tr>
<tr><td><b>Time:</b></td><td>@{outputs('Get_response_details')['body/submitDate']}</td></tr>
<tr><td><b>Files:</b></td><td>@{variables('varFileNameString')}</td></tr>
</table>
```

- Attachments: Click the little "T"/Array icon and select varAttachments.

IF NO (Denied):

1. Action: Update a row (Excel).

- Key Column: ResponseID.
- Key Value: Response Id.
- Status: "Denied".

2. Action: Send an email (V2).

- To: Recipient Email.
- Subject: "Submission Denied".
- Body:

```
<h3>Submission Denied</h3>
<p>The files submitted by @{outputs('Get_response_details')['body/responderEmail']} were
rejected.</p>
<p><b>Files submitted:</b><br>@{variables('varFileNameString')}</p>
```

- Attachments: (Leave empty).

MS365 Power Automate Collaboration

Co-Authoring (New Standard): Just like in Word or Excel, multiple people can now edit a flow at the same time. You can see their avatar and cursor in real-time.

Sharing vs. Send a Copy:

- **Add Owner (Share):** You are giving them the keys to the house. They can edit, delete, and break your flow. Use this for *teammates*.
- **Send a Copy:** You are giving them a *blueprint*. They get their own separate version to build. Use this for *templates*.

Comments: You can leave "sticky notes" on specific actions to explain complex logic to your future self or colleagues.

Demo

- **Step 1 (The "Share" Button):**
 - Open any existing flow.
 - Click **Share** in the top menu.
 - **Action:** Type a student's name (or a dummy account).
 - **Highlight:** Show how they appear under "Owners" and explain that they now have full edit rights.
- **Step 2 (Send a Copy):**
 - Go back to the Flow Details page.
 - Click **Send a copy** (Top ribbon).
 - **Scenario:** "I built a great 'Out of Office' tool. I want you to have it, but I don't want you messing with *my* specific dates."
 - Show how it generates a link/email for them to create their *own* independent instance.
- **Step 3 (Comments & Presence):**
 - Edit a flow.
 - Right-click any action step (e.g., "Send an email") > **New Comment**.
 - Type: "*Check this logic before enabling in Production.*"
 - Show the little "comment bubble" icon that appears.

More Examples for Practice

MS365 Power Automate Custom Connectors

Demo 1: The "No-Auth" Connector (Beginner)

Show the basic mechanics without getting stuck on security.

Target API: <https://catfact.ninja/fact> (Public, no key required).

1. **Create:** Go to **Custom Connectors > New > Create from blank**. Name: "CatFacts".
2. **General:** Host: catfact.ninja. Base URL: /. Scheme: HTTPS.
3. **Security:** Select **No authentication**.
4. **Definition:**
 - New Action. ID: GetFact. Summary: "Get a Cat Fact".

- **Import from sample** > Verb: GET > URL: <https://catfact.ninja/fact> > Import.
- 5. **Test:** Create Connector > Test Tab > New Connection > **Test Operation**.
 - *Result:* Show the JSON response in the body with a random fact.

Demo 2: The "API Key" Connector (Intermediate)

Teach the most common authentication standard.

Target API: <https://api.openweathermap.org> (Requires free key).

1. **Create:** Create from blank. Name: "WeatherBot".
2. **Security:**
 - Type: **API Key**.
 - Parameter Label: "API Key".
 - Parameter Name: appid.
 - Parameter Location: **Query** (Important: Explain this puts the key in the URL).
3. **Definition:**
 - New Action. ID: GetWeather.
 - **Import from sample** > Verb: GET.
 - URL: <https://api.openweathermap.org/data/2.5/weather?q=London> (Do not include the key here, the security tab handles it).
 - Click Import.
4. **Test:** Provide your real API Key in the Connection step and test.

Demo 3: The "Swagger Import" (Efficiency)

Goal: Show how to avoid manual typing by importing a definition file.

Target: The classic "PetStore" example.

1. **Find Swagger:** Google "Petstore Swagger JSON" or use <https://petstore.swagger.io/v2/swagger.json>.
2. **Create:** Go to Custom Connectors > **New** > **Import an OpenAPI from URL**.
3. **Setup:**
 - Paste the URL above.
 - Name: "PetStore Import".
 - Click **Import**.
4. **Review:** Go to the **Definition** tab.
 - Check that *all* the actions (Get Pet, Add Pet, Delete Pet) were created automatically. You didn't have to type anything.

Demo 4: The "Webhook" Debugger (Advanced)

Teach how to debug what Power Automate is sending out.

Tool: Webhook.site (A free site that shows incoming requests).

1. **Setup:** Go to webhook.site and copy your unique URL.
2. **Create:** New Custom Connector > "DebugTool".
3. **General:** Paste the *Host* from your webhook URL (e.g., webhook.site).
4. **Definition:**
 - New Action: SendDebug.
 - Import from sample > Verb: POST > URL: (Paste full webhook URL).
 - **Body:** Type { "test": "hello" } to define a JSON payload.

5. **Test:** Run the test operation.
 - *Result:* Flip tabs to the Webhook.site window. You will see the request appear instantly. This proves the connector works.

Demo 5: The "Polite" Connector (Validation)

Define "Response" schemas so Dynamic Content works nicely.

Context: Use the "CatFact" connector from Demo 1.

1. **Edit:** Open the "CatFacts" connector > **Definition** tab.
2. **Response Section:** Scroll down to the "Response" area (default block).
3. **Add Response:** Click **Add default response**.
4. **Paste JSON:** Paste a sample response: `{"fact": "Cats are cool", "length": 14}`.
5. **Save & Use:**
 - Go to a standard Cloud Flow.
 - Add the "Get Cat Fact" action.
 - Add a "Compose" action.
 - *Result:* Show that "Fact" and "Length" now appear as **green chips** in the Dynamic Content list. (Without this step, users have to write formulas).

MS365 Power Automate Collaboration

Demo 6: Co-Authoring (Real-Time)

Demonstrate multi-user editing.

1. **User A:** Create a flow. Add a "Manual Trigger". Save.
2. **Share:** Click **Share** > Add User B as "Owner".
3. **Simultaneity:**
 - User A: Stay in the designer.
 - User B: Open the flow on a separate screen.
 - *Action:* User B adds an "Initialize Variable" action.
 - *Visual:* User A sees the action appear instantly with User B's initials next to it.

Demo 7: Run-Only Users (Delegation)

Share a tool without risking the code.

1. **Create:** Build a flow with a **Manual Trigger** (Instant Cloud Flow). Action: "Send an email".
2. **Details Page:** Exit the edit mode. Look at the bottom right for **"Run-only users"**.
3. **Edit:** Click **Edit** > Add a colleague.
4. **Connection Strategy:**
 - Change the "Connections Used" dropdown to **"Provided by run-only user"**.
 - When the user clicks the button, it will send the email from *their* Outlook, not yours.

Demo 8: Send a Copy (Templating)

Give a copy to someone so they can't break yours.

1. **Navigate:** Go to the "My Flows" list.
2. **Select:** Click the "..." (Context menu) on a flow.
3. **Action:** Select **Send a copy**.
4. **Input:** Enter the student/colleague's email.

5. Result:

- The recipient receives an email.
- When they click it, it creates a *brand new* flow in their environment.
- You can delete your flow, and theirs will still work. They are disconnected.

Demo 9: Solutions (Packaging)

Enterprise-grade organization.

1. **Create Solution:** Go to **Solutions** (left nav) > **New Solution**. Name: "HR Tools".
2. **Add Components:**
 - Open "HR Tools".
 - Click **New > Automation > Cloud Flow**.
 - Create a simple flow and save it.
3. **Add Variable:** Click **New > More > Environment Variable**.
 - Name: "SiteURL". Value: "http://sharepoint..."
 - Using Variables inside Solutions allows you to change the SharePoint site easily when moving from Dev to Production.

Demo 10: Export/Import (ALM)

Moving logic between environments.

1. **Export:** Select the "HR Tools" solution from Demo 9.
2. **Package:** Click **Export solution > Next > Managed** (Locked) or **Unmanaged** (Editable). Click Export.
3. **Import:**
 - Switch to a different Environment (top right corner).
 - Go to Solutions > **Import solution**.
 - Upload the zip file you just downloaded.
 - *Result:* Your flow is now deployed to the new environment.

MS365 Power Automate AI Hub & RPA

Demo 11: Sentiment Analysis (AI Cloud)

Analyze text sentiment.

1. **Trigger:** "Manually trigger a flow" > Add text input "MyFeedback".
2. **AI Action:** Add action "**Analyze positive or negative sentiment in text**" (AI Builder).
3. **Map:** Set "Language" to English. Set "Text" to the "MyFeedback" dynamic content.
4. **Logic:** Add **Condition**. Sentiment is equal to negative.
5. **Run:** Test with the phrase "I am so unhappy with this product."
6. **Verify:** Check the run history to see the flow went down the "True" path.

Demo 12: Invoice Processing (AI Cloud)

Extract data from a PDF.

1. **Trigger:** "When a file is created (OneDrive)".
2. **AI Action:** Add action "**Extract information from invoices**".
3. **Map:** Set "Invoice File" to "File Content" from the OneDrive trigger.
4. **Output:** Add a "Compose" action.

5. **Dynamic Content:** Search for "Total Amount". You will see the AI model has automatically found the total field. Select it.
6. **Run:** Drop a sample invoice PDF into the OneDrive folder and watch the flow extract the price.

Demo 13: Language Detection (AI Cloud)

Routing support tickets.

1. **Trigger:** Manual trigger > Text input "Message".
2. **AI Action:** Add "**Detect the language being used in text**".
3. **Map:** Map the input text.
4. **Switch Case:** Add a **Switch** control. Map it to "Language Code".
5. **Cases:**
 - o Case 1: en (English) -> Action: "Assign to John".
 - o Case 2: es (Spanish) -> Action: "Assign to Maria".
 - o Case 3: fr (French) -> Action: "Assign to Pierre".

Demo 14: Web Scraping (RPA / Desktop Flow)

Getting data from a website that has no API. Pre-req: Power Automate Desktop installed.

1. **Launch:** Open Power Automate Desktop > New Flow.
2. **Browser:** Drag "**Launch new Chrome**". URL: www.bing.com.
3. **Record:** Click the **Recorder** button (looks like a red dot).
4. **Action:**
 - o Type "Current Temperature" in the search bar.
 - o Click Search.
 - o Hover over the temperature number (e.g., "72°"). Right-click > **Extract element value** > **Text**.
5. **Finish:** Click "Done". The steps will appear in the editor.
6. **Run:** Press Play. Watch the "Ghost hands" open the browser and grab the text.

Demo 15: Excel Automation (RPA / Desktop Flow)

Automating legacy Excel reporting.

1. **Launch:** Open Power Automate Desktop > New Flow.
2. **Start Excel:** Drag action "**Launch Excel**". Select "Launch a blank document".
3. **Write:** Drag action "**Write to Excel worksheet**".
 - o Value to write: "Report Date: %Now%".
 - o Column: 1, Row: 1.
4. **Save:** Drag action "**Save Excel**". Save as "DailyReport.xlsx".
5. **Close:** Drag action "**Close Excel**".
6. **Run:** Press Play. Watch Excel pop open, data appear, and the file close automatically.

Microsoft Fabric

The "Hook" for Power Platform Users

*"Imagine if **Power BI** (Reporting), **Azure Synapse** (Data Engineering), and **Azure Data Factory** (ETL) all moved into the same house, shared the same kitchen, and stopped charging you for moving furniture between rooms. That house is **Microsoft Fabric**."*

The Core Concepts

Use this table to translate their existing skills into Fabric terms.

Power Platform Concept	Fabric Equivalent	Why it's different
OneDrive	OneLake	The "OneDrive for Data." You store data <i>once</i> , and all apps (SQL, Power BI, Python) read it from there. No copies.
Power Query	Dataflow Gen2	It's Power Query, but it writes data into a Lakehouse/Warehouse (not just a dataset).
Power Automate	Data Pipelines	Like Cloud Flows, but designed for moving <i>massive</i> data (GBs/TBs) rather than individual items.
Import Mode	Direct Lake	A new mode that is as fast as Import but real-time like DirectQuery.
Data Trigger	Real-Time Intelligence	(Formerly Data Activator). Triggers actions based on data patterns, not just simple events.

Guide (The Workflow)

Teach Fabric in this 4-step lifecycle:
Ingest → Store → Process → Serve.

Step 1: Ingest (Getting Data In)

- **Action:** Create a **Lakehouse**.
- **Tool:** Data Pipelines or Dataflow Gen2.
- **Teaching Point:** Show them they can bring data from *anywhere* (AWS S3, Google, On-Prem) without writing code.

Step 2: Store (OneLake)

- **Action:** The data lands in the "Files" section of the Lakehouse.
- **Teaching Point:** Explain **Delta Parquet**. It's just a file format (like CSV but smarter) that everyone can read.

Step 3: Process (Cleaning & Modeling)

- **Action:** Use a **Notebook** (Python) or **SQL Endpoint** (T-SQL) to clean the data.
- **Teaching Point:** "Bilingual Data." The Python coder and the SQL analyst can work on the *exact same table* at the same time.

Step 4: Serve (Reporting)

- **Action:** Create a Power BI Report using **Direct Lake**.
- **Teaching Point:** No refresh schedule! If data updates in the Lake, the report updates instantly.

Demonstration Scenarios

Demo 1: The "No-Copy" Shortcut (The Wow Factor)

- **Goal:** Show how OneLake eliminates data duplication.
- **Steps:**
 1. Create a **Lakehouse A** and upload a CSV file (e.g., "Sales_Data.csv").
 2. Create a **Lakehouse B** (imagine this is a different department).
 3. In Lakehouse B, right-click "Files" → **New Shortcut**.
 4. Select "Microsoft OneLake" → Browse to Lakehouse A → Select "Sales_Data.csv".
 5. **Reveal:** The file appears instantly in Lakehouse B.
 6. **Twist:** Delete the file in Lakehouse A (or rename it) to show it's a *link*, not a copy.

Demo 2: Dataflow Gen2 (The Comfort Zone)

- **Goal:** Show them Power Query writing to a database.
- **Steps:**
 1. Open **Data Factory** experience → New **Dataflow Gen2**.
 2. Import an Excel file (just like PBI Desktop).
 3. Perform a transformation (e.g., "Remove Top Rows" or "Split Column").
 4. **Crucial Step:** Click **Add Data Destination** (bottom right) → Select your Lakehouse.
 5. Publish. Show them the table appearing in the Lakehouse automatically.

Demo 3: The "Bilingual" Data (SQL vs. Python)

- **Goal:** Show flexibility.
- **Steps:**
 1. Open your Lakehouse.
 2. **Persona 1 (The Engineer):** Click **Open Notebook**. Write simple PySpark code to read a table:

Python

```
df = spark.sql("SELECT * FROM Sales LIMIT 10")
display(df)
```

3. **Persona 2 (The Analyst):** Switch the top-right dropdown from "Lakehouse" to **"SQL Analytics Endpoint"**.
4. Write a SQL query on the *same* table:

SQL

```
SELECT TOP 10 * FROM Sales
```

5. Explain that no data was moved; just viewed through different lenses.

Demo 4: Direct Lake Mode (The "Killer Feature")

- **Goal:** Demonstrate speed without "Refresh".
- **Steps:**
 1. In the Lakehouse, click **New Semantic Model**.
 2. Select your tables and click **Confirm**.
 3. Point out the Storage Mode is **"Direct Lake"**.
 4. Create a quick report.
 5. **The Test:** Go back to your Notebook/SQL and insert a new row of data.

6. Immediately click "Refresh Visuals" on the report. The data appears *instantly*. No "Scheduled Refresh" needed.

Demo 5: Real-Time Intelligence (The "Power Automate" trigger)

- **Goal:** Trigger an action when data changes (Reflex).
- **Steps:**
 1. Go to a Power BI report visual (e.g., a card showing "Total Defects").
 2. Click the three dots (...) → **Set Alert (Data Activator)**.
 3. This opens the **Real-Time Intelligence** interface.
 4. Create a rule: "When 'Defects' > 50".
 5. **Action:** Choose "**Send Email**" or "**Run a Power Automate Flow**".
 6. Explain this is better than standard PBI alerts because it runs on the data stream, not just the report view.

Demo 6: Data Pipelines (The Orchestrator)

- **Goal:** Orchestrate a complex task.
- **Steps:**
 1. Create a **Data Pipeline**.
 2. Add a "**Copy Data**" activity (Move data from generic Blob Storage to Lakehouse).
 3. Add a "**Notebook**" activity (Clean the data).
 4. Connect them with an arrow (On Success).
 5. Run it. Show the beautiful "Gantt chart" style monitoring view that Power Automate users usually envy.