Design & Security

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

DESIGN FROM BUSINESS NEEDS

Personas Tasks → **Platform** App type → **Data** Ownership → **Access** Least privilege

01 Design and secure business solutions

Match the tool to the work

- **Canvas apps** give makers control over screens, layout and formulas, with connections to multiple data sources. Choose them for focused tasks, mobile capture and tailored user experiences.
- **Model-driven apps** use Dataverse tables, relationships, forms, views and navigation to build a consistent data-centered experience. They suit related records, activities and guided business processes.
- **Power Automate** connects events to actions. **Power BI** models and visualizes data. **Power Virtual Agents** builds topic-based conversations; bots can call flows to retrieve or update business data.
- **Power Apps portals** provide external-facing Dataverse experiences. Design external identity and table permissions explicitly. Teams tabs expose apps or reports where employees collaborate; the underlying access requirements still apply.

Requirements and environments

- Identify **personas**, tasks, data owners, devices, connectivity and approval paths before choosing components. Separate data capture, business validation, workflow and reporting requirements.
- An **environment** contains apps, flows, connections and, when provisioned, a Dataverse database. Use the **Power Platform admin center** for environment administration and the **maker portal** for building solutions.
- Use separate **development**, **test** and **production** environments. Sandboxes support isolated testing; protect sensitive data and control access when copying production data into test environments.
- **Dataverse for Teams** supplies a team-centered data platform for Teams apps. Full **Dataverse** provides broader application, administration and security capabilities. Select the platform for the required features and scale.

Dataverse authorization

- **Security roles** grant table privileges such as Read, Create, Write and Delete. **User**, **Business Unit**, **Parent: Child Business Units** and **Organization** access levels determine the permitted row scope.
- **User or team-owned** tables support record ownership and scoped access. **Organization-owned** tables suit shared reference information. Choose ownership to match who should control individual records.

Design & Security

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

DESIGN FROM BUSINESS NEEDS

Personas Tasks → **Platform** App type → **Data** Ownership → **Access** Least privilege

- Privileges from assigned roles and teams are **cumulative**. A restrictive role cannot cancel a broader grant from another role. Inspect every relevant role when troubleshooting excess access.
- **Append** allows a row to be associated with another row; **Append To** allows other rows to be associated with it. Related-record operations can require privileges on both tables.
- **Column security** uses security profiles to grant Read, Create and Update access to supported columns. Enable security on the column and assign profiles to users or teams; some system columns cannot be secured.
- **Hierarchy security** can extend access through a manager or position hierarchy. Date-dependent editing restrictions require validation logic; column security alone does not evaluate a record age condition.

Sharing and least privilege

- Share a **canvas app** with individuals or supported security groups. **Users** run it; **co-owners** can edit and share it. Grant editing rights only to people responsible for maintaining the app.
- For a **model-driven app**, associate the intended security roles with the app and assign those roles to users or teams. Include privileges for every table and related operation the app requires.
- **App access**, **data-source access**, **connections** and **licensing** are separate requirements. Sharing an app does not automatically grant Dataverse or SharePoint data permissions.
- A gallery filter or hidden control changes the interface. Enforce confidential-record restrictions in **Dataverse roles**, source permissions or report security so users cannot bypass them through another client.
- **Environment Maker** permits resource creation but does not itself grant business-data access. **System Customizer** supports customization; **System Administrator** has broad environment authority. Use user access diagnostics to investigate unexplained access failures.

Governance and data boundaries

- **Data loss prevention policies** classify connectors as Business, Non-business or Blocked. Business and Non-business connectors cannot exchange data in the same app or flow; blocked connectors cannot be used.
- A flow that cannot be enabled may violate an applicable **DLP policy**. Review connector classification and policy scope with the responsible administrator; a maker role does not bypass policy.
- Plan **data residency**, sensitive-data handling and retention for each connected service. The **Service Trust Portal** provides Microsoft compliance evidence; a solution checker does not determine an organization's legal obligations.

Dataverse & Model-driven Apps

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

FROM DATA TO EXPERIENCE

- **Tables** — Business objects → **Relationships** — Connections → **Forms & views** — Interaction → **Rules** — Validation

02 Build Dataverse and model-driven apps

Tables and column choices

- Reuse **standard tables** when their meaning fits: Account for organizations, Contact for people, and User for internal platform identities. Use **custom tables** for distinct business objects.
- **Virtual tables** represent external data at runtime without copying it into Dataverse. Provider capabilities determine supported operations; a virtual table is not a replacement for an offline data store.
- Use **text** for identifiers that may contain letters or leading zeros, **whole number** for counts, **decimal** for measurements, **currency** for money, and **date/time** for temporal values.
- **Choice** stores one controlled selection; **Choices** allows multiple selections. A **global choice** shares a centrally maintained option list across columns; a local choice belongs to one column.
- Use a **lookup** for a related business record with its own attributes, such as a warehouse address. A choice list suits stable categories rather than hundreds of independently maintained records.
- **Autonumber** generates formatted text identifiers with supported sequential and date segments. A display identifier serves a different purpose from the row's unique Dataverse ID; sequences can contain gaps.

Relationships and association data

- In **1:N**, one parent has many child rows. The child carries the **N:1 lookup** to the parent. A customer can have many orders; each order points to its customer.
- An **N:N relationship** associates many rows on both sides through an intersect table. Neither side becomes the single parent merely because it was selected first in the designer.
- Use an explicit **junction table** when the association needs attributes such as term, quantity or start date. Rental contracts can link an instrument and a renter while preserving rental history.
- Create relationships in the **table designer** or **solution explorer**. Design required lookups and cascading behaviors around valid ownership, deletion and reassignment of related records.

Calculated values and data quality

- **Simple columns** store supplied values. **Calculated columns** derive values from expressions. **Rollup columns** aggregate supported related rows, such as total sales for an account.
- **Rollups** are recalculated through platform processing and may not reflect a new transaction immediately. Use them for summaries; assess timing before using their values in an immediate decision.

Dataverse & Model-driven Apps

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

FROM DATA TO EXPERIENCE

Tables — Business objects → **Relationships** — Connections → **Forms & views** — Interaction → **Rules** — Validation

- Normalize **currency** before comparing regional totals. Define the conversion direction, applicable rate and rate date. Dataverse currency conversion and a custom exchange-rate table are distinct designs.
- Changing a **choice label** changes presentation; deleting an option can affect existing values. Plan explicit data migration when splitting a category rather than assuming historical rows can be classified automatically.

Import, transform and connect data

- **Power Query** cleans and shapes source data. **Dataflows** use queries to load and refresh Dataverse data; map source tables, columns and rows to the target schema.
- **Append** combines compatible rows from multiple sources; **merge** joins related columns. Set data types, remove duplicates and define keys before loading consolidated Excel data.
- An **on-premises data gateway** bridges supported cloud services to private sources such as local SQL Server. Configure a reachable gateway, authorized credentials and source permissions.
- Prefer an existing **connector** for a supported service. A **custom connector** exposes an API contract for a private or unsupported service. Connector capability and premium licensing classification describe different things.
- A **live connection** can retrieve refreshed source data; importing static data into an app creates a snapshot. Excel-backed apps require data in a supported table and a shared accessible storage location.

Forms, related data and navigation

- The **app designer** selects pages and components. The **site map** organizes navigation into areas, groups and subareas; give departments a clear navigation structure.
- A **main form** supports full record interaction and business process flows. **Quick create** streamlines record entry. A **quick view form** displays related-record information through a lookup.
- To show customer details on a repair form, build the **quick view form on Customer**, then place a quick view component on Repair and bind it to the customer lookup.
- A **subgrid** lists multiple rows on a form. Configure related rows, target table and default view; enable the view selector when users need alternate views.
- An **embedded canvas app** can provide a custom experience inside a model-driven form. Configure its form-column binding and use **ModelDrivenFormIntegration** to access the host record context.
- Save and **publish** form and navigation changes, include the intended components in the app, and verify the experience under the users' security roles.

Dataverse & Model-driven Apps / Canvas Apps

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

FROM DATA TO EXPERIENCE

- **Tables** — Business objects
- → **Relationships** — Connections
- → **Forms & views** — Interaction
- → **Rules** — Validation

Views, filters and search

- A **view** defines columns, ordering and filters for a table's row list. **Public views** provide shared definitions; users can create and share **personal views** for their own workflows.
- **Owner** identifies responsibility; **Created By** identifies who entered the row. For oldest-first assigned work, filter by the current owner and sort creation time ascending.
- **Sorting** changes order; **filtering** changes the rows shown. A view restriction improves focus but still depends on table security for enforceable access control.
- Configure **Quick Find** search columns and filters for the intended searches. Dataverse search can search across configured tables; grid column filters, views and cross-table search serve different needs.

Business rules and process guidance

- A **business rule** combines conditions with supported actions such as setting values or validating data. Define its scope, validate it, save it and activate it.
- **Table scope** runs supported logic on the server and is required for canvas use. **All Forms** or a specific form scopes model-driven form logic; show/hide and enable/disable actions do not control canvas controls.
- Use canvas **Visible**, **DisplayMode** and validation formulas for immediate canvas presentation. Server validation protects data across clients; an asynchronous cloud flow may run after the original save.
- A **business process flow** guides users through stages and steps on model-driven forms, potentially across tables. Required steps support process consistency; use cloud flows for emails and cross-service actions.

03 Build accessible canvas apps

Choose controls by interaction

- A **gallery** repeats a customizable layout for multiple records, including text and photos. A **data table** presents rows in columns. A **display form** shows one record; an **edit form** supports entry and updates.
- **Text input** captures text; a **slider** selects a numeric range; **radio** selects one visible option. A **list box** can expose multiple choices; a **combo box** adds searchable single or multiple selection.
- **Barcode reader** scans supported barcodes or QR codes. Use the decoded identifier to look up the matching employee or product; scanning a badge does not automatically supply its photo.
- **Image** displays a picture. **Camera** captures a device photo; **Add picture** accepts a photo or uploaded image. **Pen input** captures drawings or signatures.

Canvas Apps

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

CANVAS APP BUILDING BLOCKS

- **Controls** — Input & display
- **Power Fx** — Properties & actions
- **State** — Variables & collections
- **Offline** — Queue & sync

- A **Timer** uses **Duration** in milliseconds and events such as **OnTimerEnd**. Convert text input with **Value**. Persist timestamps for long-running elapsed time rather than assuming a screen timer survives app suspension.
- Screen **BackgroundImage** supplies a background; **LoadingSpinner** controls the built-in loading indicator. Keep the indicator distinct from a success or failure message.

Properties and declarative formulas

- Bind a control's **Text**, **Fill** or **Visible** property to a formula so it recalculates when inputs change. Use **OnSelect** and **OnChange** for actions such as saving or setting variables.
- **If** selects a result from a condition; **Switch** handles several cases. For a risk label, use **If(SliderA.Value + SliderB.Value >= 150, "HIGH RISK", "")** in the label's Text property.
- Use **Fill** for conditional color, for example **If(Risk.Value > 75, Color.Red, Color.Green)**. Include text or an icon so color is not the only indication of state.
- **Value** converts numeric text; **Text** formats values. **IsBlank** tests a missing scalar value; **IsEmpty** tests an empty table. Zero remains a valid number rather than a missing value.
- **RoundUp(quantity, 0)** rounds a positive requirement to whole units. Validate measurements and guard division against blank or zero denominators before calculating quantities.

State, collections and navigation

- **Set(total, 5)** creates or updates app-wide global state. **UpdateContext({total: 5})** updates screen-local state. Both can store records or tables, as well as simple values.
- **Navigate(DetailScreen, ScreenTransition.None, {selectedRow: Gallery1.Selected})** opens a screen and passes context. **Back()** returns to the previous screen.
- A **collection** is an in-memory table shared across screens. **Collect** adds rows, **Clear** removes rows, and **ClearCollect** replaces the collection with new content.
- Use **App.OnStart** for suitable initialization and **Screen.OnVisible** for screen-entry actions. Repeated Collect calls append rows again; initialize deliberately to avoid duplicated cached data.
- **Environment variables** hold deployment configuration, such as a service URL. They are not a substitute for per-user session variables or local collections.

Canvas Apps

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

CANVAS APP BUILDING BLOCKS

- **Controls** — Input & display
- **Power Fx** — Properties & actions
- **State** — Variables & collections
- **Offline** — Queue & sync

Forms and data operations

- Set a form's **DataSource** to the table and **Item** to the selected record, such as **Gallery1.Selected**. **ViewForm**, **EditForm** and **NewForm** change how the form is used.
- **SubmitForm** validates and saves the edit form. Use **OnSuccess** to navigate or reset after a successful save and **OnFailure** to explain the error. **ResetForm** discards pending control edits.
- **Patch** creates or updates selected columns without a full form. **Patch(Orders, Defaults(Orders), {Title: "New order"})** creates a row; an existing base record targets an update.
- **UpdateIf** changes matching records; **Remove** and **RemoveIf** delete records. **Refresh** retrieves a fresh source copy. Check connector delegation and operation support before using bulk formulas.
- Form **data cards** connect columns to input controls. Their Default and Update properties determine displayed and submitted values; use **DisplayMode** to communicate editability without replacing source authorization.

Query correctly at scale

- **Filter** returns matching rows; **LookUp** returns the first matching row or a selected value. Equality tests exact values; **Search** finds text within supported columns and **StartsWith** matches a prefix.
- **SortByColumns** controls row order. **Distinct** returns unique values for a selection list; for large sources, confirm that the operation can retrieve every required value.
- **Delegation** sends supported query work to the data source. Support depends on the connector, column types and complete expression. Resolve delegation warnings before trusting large-table results.
- A **nondelegable** query can evaluate only a limited local subset and silently miss matching rows. Caching a large source with ClearCollect does not make an incomplete retrieval complete.
- Retrieve only needed rows and columns, reuse appropriate cached reference data, and refresh when freshness matters. Balance reduced network traffic against stale data and device memory.

Canvas Apps

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

CANVAS APP BUILDING BLOCKS

- **Controls** — Input & display
- **Power Fx** — Properties & actions
- **State** — Variables & collections
- **Offline** — Queue & sync

Offline capture and synchronization

- **Connection.Connected** indicates network connectivity, not guaranteed service availability. Handle failed writes even when it is true and show a readable online/offline indicator.
- **SaveData** persists a local collection; **LoadData** restores it on supported clients. Queue pending edits locally, retry when connectivity returns, and remove items only after confirmed server writes.
- Local collections do not automatically synchronize. Design duplicate prevention, conflict handling and attachment storage; **SubmitForm** alone does not create a durable offline queue.
- For **model-driven mobile offline**, configure the app's offline profile, required tables, related rows and supported attachments. Test loss of connectivity and synchronization on the actual mobile client.

Reuse and responsive layout

- A **canvas component** bundles reusable controls and behavior. Define **input properties** for supplied data and **output properties** for exposed results; a reusable gallery can accept the current record as input.
- A **component library** shares components across apps. Publish library changes, then review and accept the update in consuming apps and publish those apps to deliver the change.
- **Power Apps component framework (PCF)** supplies code components. Canvas components and component libraries provide maker-authored reuse; choose the mechanism that matches the customization requirement.
- Use **horizontal and vertical auto-layout containers**, flexible dimensions and parent-relative sizing for responsiveness. Test phone, tablet and browser orientations; a fixed tablet design alone does not make an app responsive.
- For responsive canvas layouts, review **Scale to fit**, aspect-ratio locking and orientation settings. A **manual container** preserves explicit positioning; an **auto-layout container** arranges children by its layout rules.
- Centralize shared values such as button color in an app-wide variable or reusable component. Bind each **Fill** property to that value so one change updates the intended controls.

Accessibility and interaction

- Set **AccessibleLabel** on meaningful images, icons and controls. Use descriptive text, logical reading order and keyboard operation; provide accessible text equivalents for critical visual content.
- Use **TabIndex** and layout order deliberately. Give interactive controls visible focus with **FocusedBorderThickness** and contrasting focus color; avoid unnecessary keyboard stops on decorative controls.

Canvas Apps / Automation & AI

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

EVENT TO BUSINESS RESULT

Trigger Event or request → **Evaluate** Data & conditions → **Act** Update or approve → **Observe** Runs & outcomes

- Use sufficient **contrast**, readable text and clear error messages. Transparent styling alone does not define accessible visibility; verify what assistive technology announces.
- Run **App checker > Accessibility**, fix reported issues and re-check. Then test keyboard navigation and a screen reader; automated checks cannot prove the full experience is accessible.

04 Automate processes and conversations

Flow types and trigger selection

- **Automated cloud flows** respond to events, such as a newly created list item or uploaded file. **Instant flows** start on demand; **scheduled flows** use recurrence for timed work.
- Choose a trigger for the exact event: **item created** differs from **item created or modified**. SharePoint list items, library files and file content are related but distinct objects.
- A canvas app calls a flow through a **Power Apps trigger** and passes inputs. Use manual triggers for deliberate user actions and recurrence for regular reminders, including the required time zone.
- **Desktop flows** automate supported desktop and browser interactions, useful for legacy software without an API. Prefer a cloud connector when the service exposes the needed operation.
- **Delay** waits for an interval; **Delay until** waits for a timestamp. A manually started flow can wait until a processing window without becoming a daily scheduled flow.

Data and control inside flows

- **Dynamic content** refers to trigger or action outputs. **Compose** stores an expression result; initialize and set **variables** when a value must change during execution.
- **List rows** retrieves Dataverse records; use a source filter and sort to limit work. **Apply to each** processes an array; preserve sequential execution when order or shared variables matter.
- **Condition** branches on true/false; **Switch** selects among values. Combine **and**, **or**, **equals**, **greater** and **contains** to express business predicates.
- **Filter array** selects existing array items; **Select** reshapes them; **Join** converts an array to text. **substring(text, 0, 1)** extracts a first character when the string is nonempty.

Automation & AI

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

EVENT TO BUSINESS RESULT

Trigger Event or request → **Evaluate** Data & conditions → **Act** Update or approve → **Observe** Runs & outcomes

- Guard **null**, empty strings and zero separately before math or comparisons. **coalesce** provides a fallback for null; **empty** tests supported empty values. A numeric zero must not disappear accidentally.
- Use **Scopes** to group related actions and **Configure run after** for failure handling. Review run history, inputs and outputs to distinguish a trigger problem from a failed connector action.

Approvals, notifications and ownership

- **Start and wait for an approval** creates a request and waits for an outcome. Branch on the outcome, persist the decision and notify the next participant.
- **Everyone must approve** differs from **First to respond**. Ordered approvals require successive stages; a multi-person request does not automatically enforce a sequence.
- For manager approval, retrieve the relevant user's manager with **Get manager**, then address the approval to that identity. Upload triggers may require separate file-content retrieval before attaching the document to an email.
- **Adaptive cards** present structured information and collect supported responses in Teams. Choose a wait-for-response action when later steps depend on the response; a simple posted card need not pause the flow.
- The **Teams Approvals app** provides a central approval experience. A flow **co-owner** can maintain the flow; **run-only users** can invoke an eligible instant flow without receiving editing rights.

Chatbot topics, entities and actions

- A **topic** defines a conversation path. **Trigger phrases** identify user intent; question, message and condition nodes collect information, respond and branch.
- **Entities** recognize structured values in user responses. A **closed-list entity** can map synonyms such as jeans and trousers to one category without creating a topic for every synonym.
- **Variables** retain answers and action results. Pass inputs to a **Power Automate action** to query Dataverse, then use its returned values in the conversation.
- **Publish** makes tested changes available to users. Configure the Teams channel and intended audience; distribute one suitable organizational bot when departments can be represented by topics.
- Separate **bot users** from authors and owners. Use the test conversation pane, analytics and available session transcripts to diagnose routing or action failures with appropriate access.

Automation & AI / Analytics & Delivery

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

EVENT TO BUSINESS RESULT

Trigger Event or request → **Evaluate** Data & conditions → **Act** Update or approve → **Observe** Runs & outcomes

AI Builder model selection

- **Text recognition** extracts text from images. **Document processing** extracts structured fields and tables from documents. **Business card reader** extracts contact details from business cards.
- **Object detection** locates recognized objects in images. A custom model needs representative labeled training images; **prediction** instead learns patterns in historical tabular data to estimate an outcome.
- **Category classification** assigns text to categories. **Sentiment analysis** estimates expressed sentiment. **Key phrase extraction** identifies main phrases; **entity extraction** identifies values such as people, places or organizations.
- **Language detection** identifies the predominant language of text. Select the model for the required output rather than using any text model interchangeably.
- For a custom model, prepare representative data, **train**, evaluate and **publish**, then use it in an app or flow. Prebuilt models are ready to invoke; review confidence and route uncertain results for human validation.

05 Analyze data and manage delivery

Power BI reports and distribution

- **Power BI Desktop** connects, transforms, models and authors reports. The **Power BI service** supports publishing, sharing, dashboards and refresh; it also provides report authoring capabilities.
- A **semantic model** holds tables, relationships and calculations. A **report** contains one or more visual pages; a service **dashboard** is a single canvas of pinned tiles.
- **Power Query** changes data types, renames columns and removes unwanted columns before load. **DAX measures** calculate in filter context; calculated columns produce row-level model values.
- Configure **refresh**, credentials and an on-premises gateway where required. An imported model is a snapshot until refreshed; choose connectivity for the needed freshness and source capabilities.
- **Row-level security (RLS)** filters model rows by role. Define and test rules, then assign service role membership. Hiding visuals or sharing a dashboard alone does not isolate departmental data.

Analytics & Delivery

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

DELIVER AND IMPROVE

Build Apps & reports → **Validate** Checks & tests → **Deploy** Solutions → **Monitor** Usage & failures

Visualizations, dashboards and embedding

- Use **line charts** for trends, **column or bar charts** for comparisons, **pie charts** for a small part-to-whole breakdown, and **KPIs** for progress against a target. Canvas apps include native charts without requiring Power BI.
- **System dashboards and charts** are shared app components; **personal dashboards and charts** belong to users and can be shared. Verify dashboard role access, app inclusion and publishing when a dashboard is missing.
- **Interactive dashboards** offer visual and global filters. **Single-stream** uses one stream; **multi-stream** supports several. Configure the filter table, date column and time frame to match the monitoring task.
- Embedded **Power BI** requires access in both the host app and Power BI. Enable applicable embedding settings and share the report or dashboard with the intended audience.
- Pass selected-record context through supported **Power BI filters** or embedding parameters. A **Power Apps visual in Power BI** embeds an app in a report, which is the opposite hosting direction.
- A **Word template** can generate a document from a record and selected related data. **Excel exports and PivotTables** support further analysis; copying a Power BI visual as an image provides a static snapshot rather than a live report.

Solutions and lifecycle

- **Unmanaged solutions** are the development source. Export a **managed solution** as a deployment artifact for downstream environments; managed properties control supported customization.
- A **publisher prefix** avoids schema-name collisions. Choose it before creating components; changing it does not rename existing metadata. Use a named custom solution rather than the nonexportable Default Solution.
- **Update** changes a managed solution without removing omitted components. A **patch** delivers limited changes. An **upgrade** can remove components no longer present in the replacement solution.
- **Solution history** shows operations and results; **solution layers** explain contributing customizations; **dependencies** explain required components. An unmanaged layer can affect effective behavior.
- Deleting an **unmanaged solution** removes its container, leaving customizations. Uninstalling a **managed solution** can remove components and associated data. Do not treat uninstalling as routine rollback.

Analytics & Delivery

Microsoft Power Platform App Maker

KNOWLEDGE CHEATSHEET

DELIVER AND IMPROVE

Build Apps & reports → **Validate** Checks & tests → **Deploy** Solutions → **Monitor** Usage & failures

Package complete, repeatable releases

- Package apps, **solution-aware flows**, custom tables, required columns, forms, views, roles and other dependencies in a coherent solution. Solutions move configuration; business rows need a separate migration plan.
- **Connection references** bind solution components to target connections. **Environment variables** hold target-specific configuration such as URLs. Supply target values and connections during import or deployment.
- Cross-environment or **cross-tenant migration** requires target identities, permissions, credentials and connector configuration. App sharing does not copy the application and its dependencies into another tenant.
- Publish customizations, run relevant checks, export and test the import in a separate environment. An **app package** has narrower scope than a full solution; confirm supported resources before choosing it.
- Use **source control** and an ALM pipeline, such as Azure DevOps with Power Platform build tools, to retain artifacts, document changes and repeat deployment. Version solutions with **major.minor.build.revision**.

Versions, validation and diagnostics

- **Save** records a canvas app version; **Publish** makes a saved version live for users. Restoring an older app creates a new version, which must be published to replace the live version.
- Use **two separately shared apps** when groups must test different experiences simultaneously. One app's saved versions are not independent live releases for separate audiences.
- **App checker** finds canvas formula and accessibility issues. **Flow checker** detects flow configuration issues. **Solution checker** applies static analysis rules to supported solution components.
- **Test Studio** organizes an app's tests into **test suites**, **test cases** and **test steps**, with assertions to verify outcomes. Record interactions, add assertions and use representative inputs and roles.
- **Monitor** traces canvas app events and connector calls during authoring or a published session. Inspect failures, latency, delegation warnings and throttling responses; test accessibility and offline behavior on real target clients.