Phase 8: Data Management & Deployment – Salesforce Project (Email Summarizer & Task Manager)

# Objective:

Manage Salesforce data efficiently, prevent duplicates, backup data, and deploy configurations from Sandbox to Production.

## 1️⃣ Data Import Wizard – Import Demo Records

Steps:

1. Setup → Data Import Wizard.

2. Select Custom Object (Email\_\_c / Task\_\_c).

3. Launch Wizard, upload CSV (50 demo Email records).

4. Map fields (Subject, Sender, Date, Status).

5. Start Import.

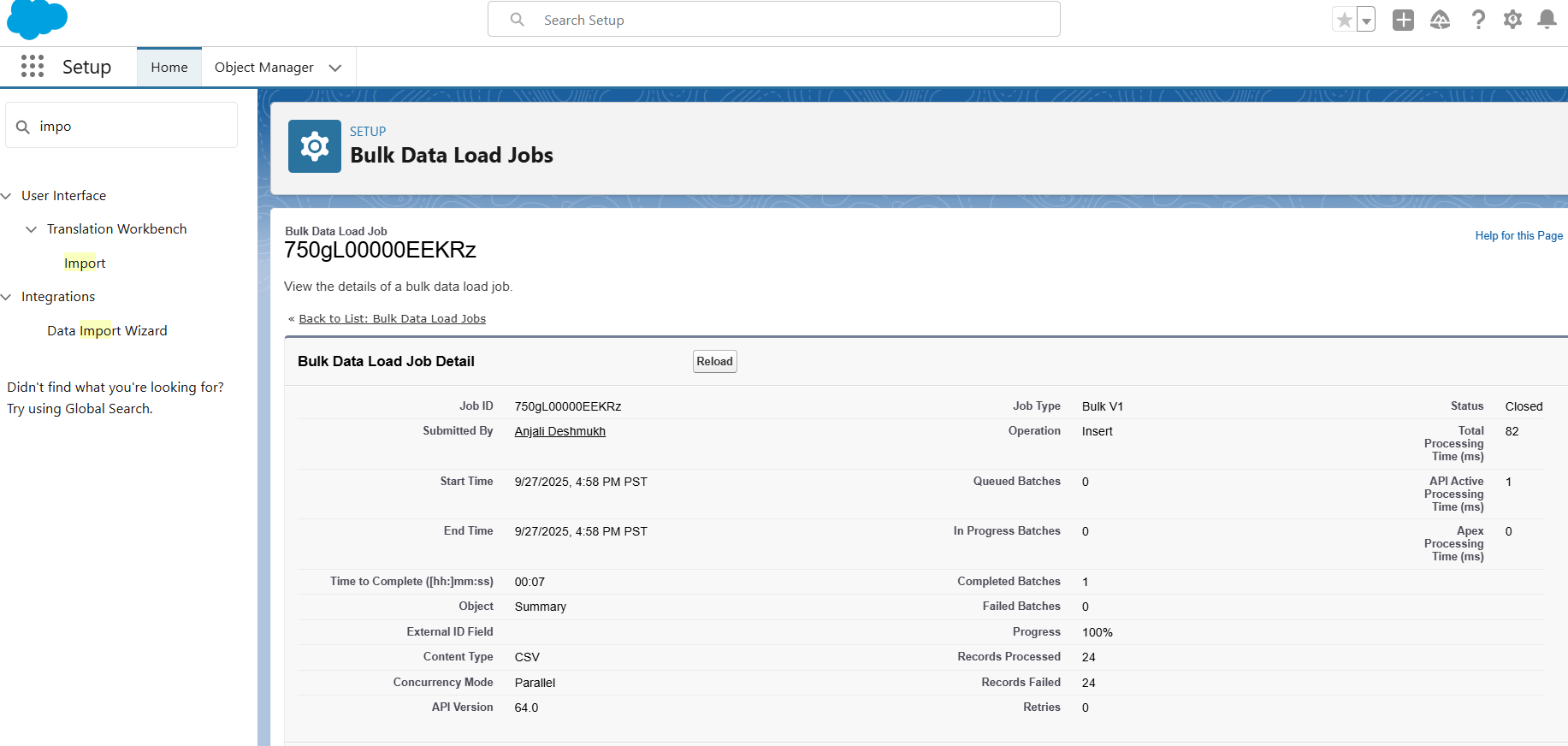
6. Verify records.

Screenshot Placeholders:

* - Before import.
* - Field mapping.
* - Import success.

Tips:

* - Ensure CSV headers match API names.



## 2️⃣ Data Loader – Bulk Import

Steps:

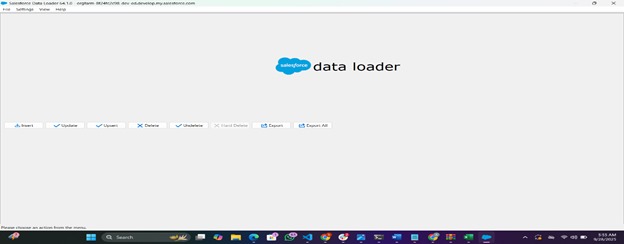
1. 1. Open Data Loader → Insert → Login.
2. 2. Select object (Task\_\_c).
3. 3. Upload CSV.
4. 4. Map fields.
5. 5. Finish insert.

Screenshot Placeholders:

* - Data Loader main screen.
* - Field mapping.
* - Insert success.

Tips:

* - Save success & error CSVs.
* - Use External ID if needed.

---

## 3️⃣ Duplicate Rules – Prevent Duplicates

Steps:

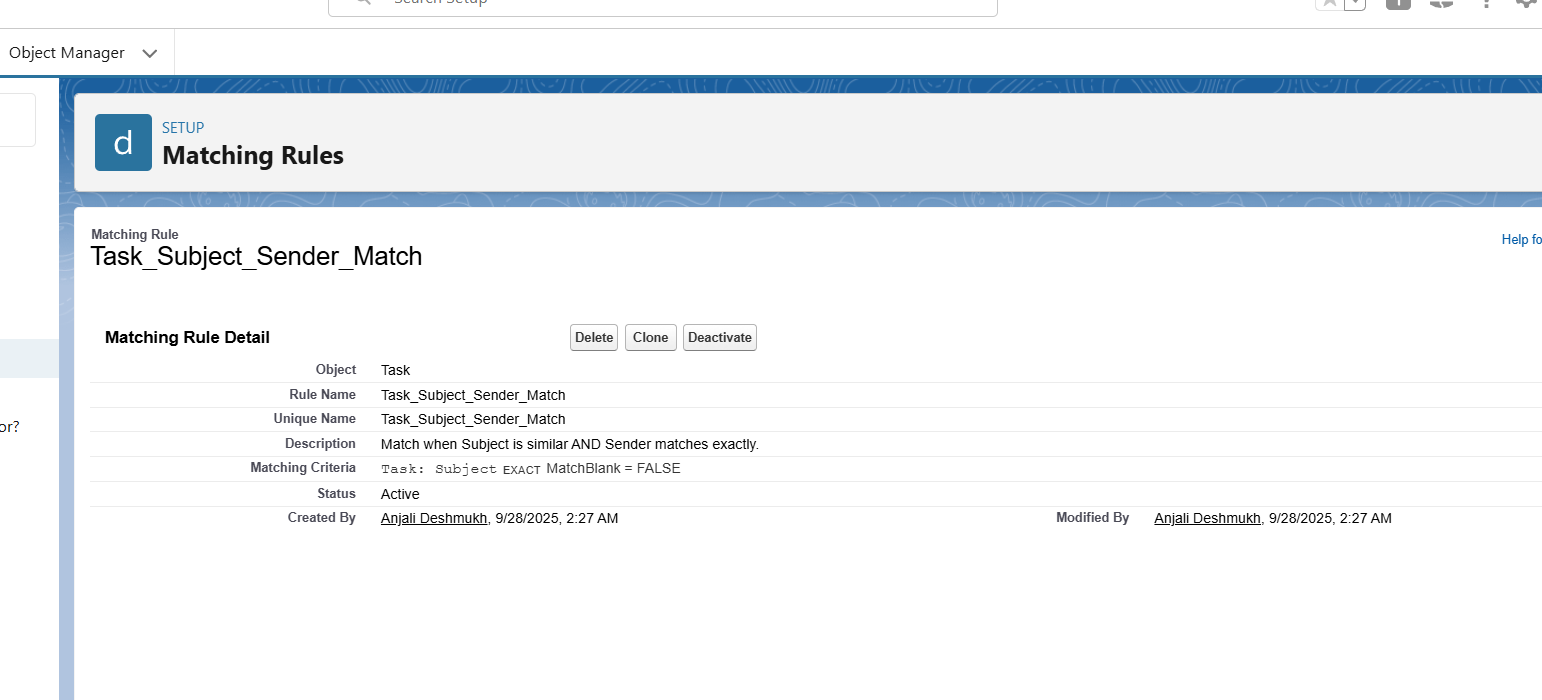
1. 1. Setup → Duplicate Management → Duplicate Rules.
2. 2. New Rule → Select object (Email\_\_c / Task\_\_c).
3. 3. Define Matching Rule (e.g., Subject + Sender).
4. 4. Set action (Alert/Block).
5. 5. Activate.

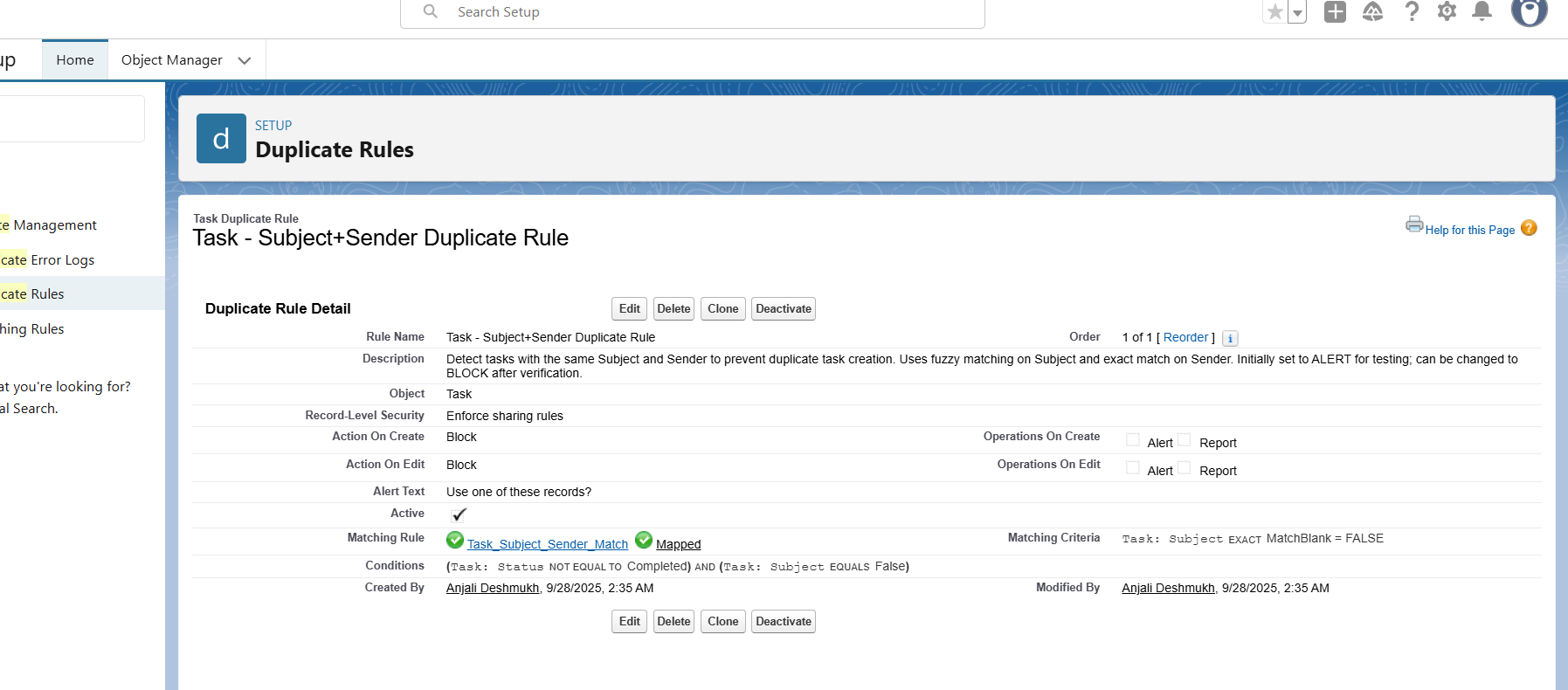
Screenshot Placeholders:

* - Duplicate rule page.
* - Matching criteria.

Tips:

* - Test by creating duplicate record.





## 4️⃣ Data Export & Backup

Steps:

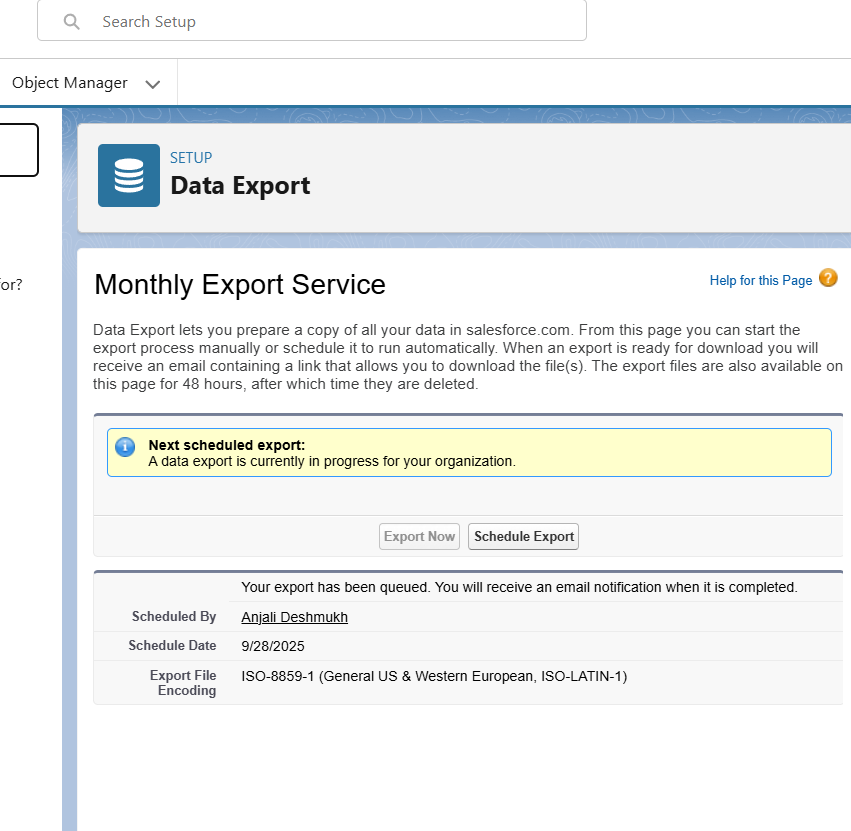
1. 1. Setup → Data → Data Export.
2. 2. Choose Export Now or Schedule (weekly).
3. 3. Select objects (Email\_\_c, Task\_\_c, User).
4. 4. Start Export.
5. 5. Download ZIP.

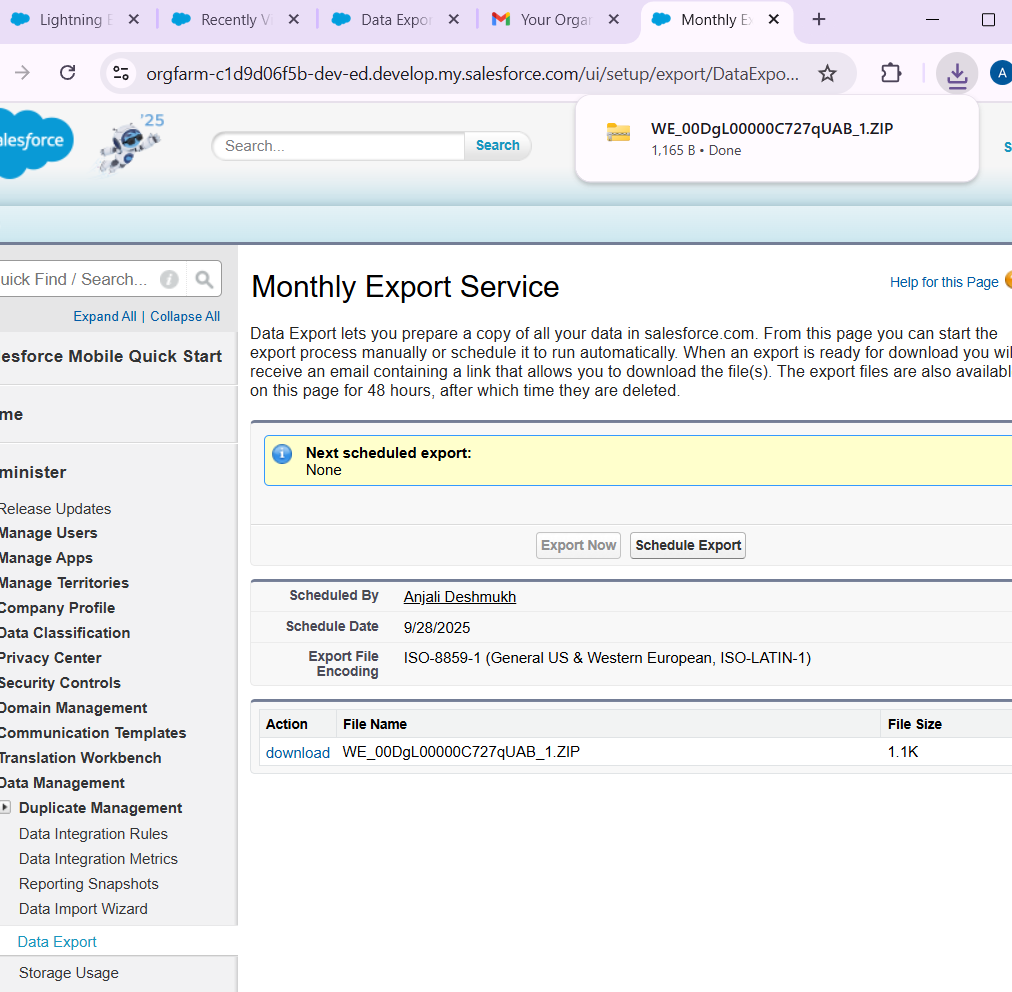
Screenshot Placeholders:

* - Export screen.
* - Downloaded ZIP.

Tips:

* - Schedule automated backups.
* - Keep multiple backup versions.





## 5️⃣ Change Sets – Deploy Configurations

Steps:

1. 1. Setup → Outbound Change Sets → New.
2. 2. Name: Phase8\_Deployment.
3. 3. Add Components (Objects, Fields, Workflows, Apex Classes, Lightning Pages).
4. 4. Upload to Production.
5. 5. Inbound Change Sets → Deploy.

Screenshot Placeholders:

* - Change set creation.
* - Components added.
* - Deployment success.

Tips:

* - Test in Sandbox first.
* - Include Apex tests if required.

**1. Use Case of Change Sets**

Change Sets are Salesforce’s **point-and-click deployment tool** to move **metadata (configuration & code)** from one Salesforce org to another.

👉 **Use case in your Phase 8 (Email Summarizer & Task Manager project):**

* You’ve created custom objects (e.g., **Email Summary**, **Task**), fields, Apex classes, Flows, and maybe Lightning Pages in your **development org**.
* Before your app can be used in a **real environment (Production or another org)**, you must migrate these configurations.
* Instead of re-building everything manually, you package them in a **Change Set** and deploy.

**Example for your project:**

* Components:
  + Custom Object: Summary\_\_c (Email Summary).
  + Fields: Has\_Action\_Items\_\_c, Key\_Action\_Items\_\_c, etc.
  + Apex Classes: summarizer logic, task assignment service.
  + Flow: automated creation of Task when Email is summarized.
  + Lightning Page: dashboard for users.

➡️ You create an **Outbound Change Set** in Sandbox (or Dev org → if connected to Prod).  
➡️ Upload it → Inbound Change Set in Production.  
➡️ Deploy, run tests, confirm your app is now in Production.

So, the **use case is safe migration of your app from “where you build” → “where people use it.”**

**🔹 2. Developer Edition vs Sandbox**

This is where confusion often happens. Let me break it down:

| **Feature** | **Developer Edition (DE)** | **Sandbox** |
| --- | --- | --- |
| **What it is** | A free, standalone Salesforce org (like your personal playground). | A copy of your Production org, tied directly to it. |
| **Purpose** | For building, testing, and training (great for projects, learning, prototypes). | For testing deployments safely before pushing changes to Production. |
| **Data** | Comes empty with sample data only. | Mirrors Production data (depending on Sandbox type: Full, Partial, Developer). |
| **Users** | Independent user set (only you unless you add others). | Inherits users and roles from Production. |
| **Deployment** | Cannot directly use Change Sets to Production unless you have a “deployment connection.” Usually DE → Prod is not supported. | Fully supports Outbound → Inbound Change Sets with Production (they are connected). |
| **Backup/Testing** | No automatic backup of Prod data. | Safer testing with real Production metadata and data. |
| **Best Use Case** | Learning, prototyping, small apps, AppExchange publishing. | Testing upgrades, running full UAT, validating before go-live. |

**🔹 3. How it applies to your project**

Since you **built in a Developer Edition (not a Sandbox)**:

* You **cannot deploy via Change Sets to Production**, because DEs are not tied to a Production org.
* But you have **other options** for deployment:
  1. **Packages**
     + You can create an **Unmanaged Package** in your Developer Edition containing your app.
     + Install that package in another org (Production, another DE, or Sandbox).
     + This is the common way for student projects / small apps.
  2. **VS Code + SFDX (Salesforce CLI)**
     + Connect both orgs and use source-based deployments.
     + More dev-friendly and flexible, but needs setup.
  3. **ANT Migration Tool**
     + Script-based deployment (less common for small student projects).

👉 **So in your case (since this is a project, not a live company org)**:

* You should **use an Unmanaged Package** instead of Change Sets.
* Change Sets are mentioned in Phase 8 because that’s the **real-world practice** when working with Sandbox + Production.

## 6️⃣ Unmanaged vs Managed Packages

Steps:

1. 1. Setup → Packaging → Packages → New.
2. 2. Add components.
3. 3. Select Managed if publishing to AppExchange.

Screenshot Placeholders:

* - Package creation.
* - Components added.

Tips:

**1 — Theory: Unmanaged vs Managed Packages**

**Unmanaged Package (what it is)**

* A simple packaging mechanism that groups metadata (objects, fields, classes, pages, flows) so you can install them into another org.
* **Code and metadata are fully visible and editable** after installation.
* **No upgrade path**: installing a new version does not automatically update previously installed components (you’d have to manually manage changes).
* Best for: distributions where source should remain editable (internal rollouts, training, student projects, examples).

**Managed Package (what it is)**

* A versioned, namespace-scoped package intended for distribution (AppExchange).
* Supports **versioning**, **upgrades**, **license management**, and **component protection** (some parts can be hidden/protected).
* Typically created in a Developer Edition org (packaging org) where you register a **namespace**. Modern packaging also supports 2GP (SFDX-based) with Dev Hub.
* Best for: commercial apps, ISV distribution, when you want automatic upgrades and IP protection.

**Key differences (table)**

| **Feature** | **Unmanaged** | **Managed (1GP / 2GP)** |
| --- | --- | --- |
| Editable after install | Yes (fully) | Typically no — protected components hidden |
| Versioned upgrades | No | Yes — install upgrades automatically |
| Namespace | No (unless you add) | Yes (namespace assigned) |
| AppExchange-ready | Yes (technically) but not ideal | Designed for AppExchange |
| License/Monetization | No | Yes (licensing & usage controls) |
| Best for | Internal, one-time installs, teaching | Commercial apps, repeatable upgrades |

**2 — Use-cases (practical examples for your project)**

**Use-cases for Unmanaged (for you now)**

* You want to move the Email Summarizer & Task Manager from your Developer Edition into a colleague’s org, a client org, or a sandbox for demonstration.
* You want recipients to be able to edit Apex, flows and pages (e.g., teammates will customize layouts).
* Quick share without formal release/versioning.

**Use-cases for Managed**

* You plan to sell or publish your Email Summarizer on **AppExchange**.
* You want to issue small updates and let customers upgrade in-place.
* You want to protect core summarizer intellectual property (hide implementation).
* You want license controls and usage tracking.

**3 — Packaging Practicalities & Steps (short)**

**Create Unmanaged package (quick flow):**

1. Setup → **Packaging** → **Packages** → **New**.
2. Name: EmailSummarizer\_Unmanaged → Save.
3. Click **Add** → choose component types → add your objects, fields, Apex classes, Lightning pages, Flows, Permission Sets (as needed).
4. Click **Upload** (for unmanaged it will create an install link rather than AppExchange listing).
5. Use the install link to install into target org.

**Create Managed package (high level):**

1. Use a Developer Edition org (packaging org). Register a **namespace** (Setup → Packages → Register Namespace).
2. Create package, add components, **Upload** a managed package version (this is the "release").
3. Publish to AppExchange or provide install links.
4. Use packaging tools (1GP UI or 2GP via SFDX) to manage versions.

## 7️⃣ ANT Migration Tool – Command-Line Deployment

Steps:

1. 1. Download ANT Migration Tool.
2. 2. Configure build.xml & build.properties.
3. 3. Run: `ant deployCode`.
4. 4. Check logs.

Tips:

* - Useful for large deployments.
* - Requires metadata API knowledge.

---**4 — Theory: ANT Migration Tool (what it is and how it works)**

**What the ANT tool is**

* A Java/Ant-based client that uses the **Salesforce Metadata API** to **retrieve** and **deploy** metadata (objects, classes, pages, layouts, etc.).
* You build a package.xml that lists the metadata types and members you want, and then run Ant targets like retrieve or deployCode.
* The tool is scriptable and therefore useful for automation/CI or for moving metadata when Change Sets aren’t possible.

**How it compares to other options**

* **Change Sets** — point-and-click, only between connected Sandboxes and Production; manual and UI-driven.
* **ANT** — script-driven, no UI, great for automation and when orgs aren’t connected.
* **SFDX (Salesforce CLI)** — the modern, source-driven replacement; better for source control/2GP and developer workflows. SFDX is recommended for modern CI/CD, but ANT still useful in many teams.

**Typical ANT workflow**

1. Set up Java + Apache Ant on your machine.
2. Configure build.properties with org credentials (username/password+security token or OAuth).
3. Make or edit package.xml to specify metadata to move.
4. Run Ant targets:
   * ant retrieveCode (pull from an org to local)
   * ant deployCode (push local metadata to an org)
5. Check console logs and Ant-generated result files for success/errors.

**5 — Example files (minimal) — for reference**

**sample build.properties**

# For Production

sf.username=your.username@org.com

sf.password=yourPassword+securityToken

sf.serverurl=https://login.salesforce.com

# For Sandbox (use test.salesforce.com)

# sf.serverurl=https://test.salesforce.com

**sample package.xml** (include the custom object and an Apex class)

<?xml version="1.0" encoding="UTF-8"?>

<Package xmlns="http://soap.sforce.com/2006/04/metadata">

<types>

<members>Summary\_\_c</members>

<name>CustomObject</name>

</types>

<types>

<members>SummaryService</members>

<name>ApexClass</name>

</types>

<version>57.0</version>

</Package>

**sample Ant command (in your terminal)**

ant deployCode

(The Ant build.xml included with the Salesforce ANT package defines the deployCode target; it uses your build.properties.)

Security note: don’t commit build.properties with plaintext credentials to source control. Use CI secrets or OAuth.

**6 — Use-cases for ANT in your Email Summarizer & Task Manager project**

* You built in a **Developer Edition** and you want to push metadata to a **Production org** or a client org where Change Sets are not available or you don’t have a deployment connection → use ANT (or SFDX).
* You want to automate deployments (scripts that run nightly or as part of CI) or keep a repeatable process to deploy named metadata subsets.
* You want to include package.xml versioning so you can deploy incremental updates (e.g., new Apex class, changed Lightning page).
* You’re preparing for AppExchange (used in older workflows to retrieve metadata from packaging orgs).

**7 — Difference between Developer Edition and Sandboxes (focused on packaging & ANT)**

**Developer Edition (DE)**

* **Independent org** — not connected to a Production org.
* **Can be used as a packaging/org**: you can register a **namespace** in a DE and create managed/unmanaged packages there. Many ISVs use special DE packaging orgs.
* For **Unmanaged packages**: DE is fine — create package and install into other orgs.
* For **Managed packages**: you typically use a DE (or packaging org) to register namespace and upload managed package versions (you can’t register a namespace in a Sandbox).
* **ANT**: works fine — you can run ANT to push from DE to any other org (with credentials) because ANT uses Metadata API and doesn’t require org connections.

**Sandbox**

* **A copy of Production metadata/data** (Developer, Partial, Full sandbox variants).
* **Connected to Production** and supports **Change Sets** (outbound from Sandbox to Production). That is the standard Salesforce UI deployment path.
* **Packaging**: Sandboxes are not packaging orgs — they cannot be used to register a namespace or upload managed packages for AppExchange. Usually you develop in sandbox and package from a packaging org or DE if needed.
* **ANT**: also works with Sandboxes — connect using the sandbox server URL (https://test.salesforce.com) and credentials. Ideal for testing a deploy before production.

**Practical implications for your project**

* Because you built in **Developer Edition**, you **cannot** use Change Sets (they require a sandbox connected to production). So:
  + For quick installs → **create an Unmanaged Package** in your DE and install into the target org.
  + For scripted/repeatable deployments or to push to Production → use **ANT** or **SFDX** (recommended SFDX).
  + For AppExchange → move your packaging work into a Developer Edition packaging org, register a namespace, and create a managed package.

**8 — Recommended path for *your* Email Summarizer & Task Manager project**

1. **Short term / demo / handing in project**
   * Create an **Unmanaged Package** in DE, add components (Summary\_\_c, Task triggers, Apex classes, Flows, Lightning pages), upload and install into the target org (client or instructor’s org or a sandbox you control).
   * Use Data Loader to import sample records (data not included in package).
2. **If you want repeatable deployments / CI**
   * Move source to **version control (Git)**.
   * Start using **SFDX** (preferred) or **ANT** to deploy from your repo to target orgs. SFDX is recommended for modern workflows; ANT is fine if you’re more comfortable with XML-based package.xml scripting.
3. **If you plan to publish to AppExchange**
   * Create/choose a **packaging Developer Edition org**, register a **namespace**, convert the project to a managed package (1GP or 2GP via SFDX), and follow AppExchange packaging and security review processes.

**9 — Common pitfalls & tips**

* **Missing dependencies**: packages or ANT deploys fail if dependent components (fields, layouts, labels) are not included. Always run a retrieve to discover dependencies or use SFDX to identify them.
* **Profiles & Permissions**: packages won’t necessarily carry all profile permissions; include Permission Sets if you want to ship permissions.
* **Data vs Metadata**: packages and ANT move metadata only — use Data Loader / Import Wizard for data.
* **Apex tests & coverage**: Production deployments that include Apex require passing tests and org-wide coverage ≥ 75%. Include test classes.
* **Naming / Namespace**: managed package requires namespace; once set it can’t be changed. Plan carefully.
* **Use version control** before using ANT/SFDX — it’s easier to track and revert changes.

## 8️⃣ VS Code & SFDX – Dev-Friendly Deployment

Steps:

1. 1. Install VS Code + Salesforce Extension Pack.
2. 2. Authorize Org: `sfdx force:auth:web:login -a Sandbox`.
3. 3. Pull/Push metadata: `sfdx force:source:pull` / `sfdx force:source:push`.
4. 4. Run Apex tests.

Screenshot Placeholders:

* - VS Code authorization.
* - Source push success.
* - Apex test run.

Tips:

* - Best for version control & team collaboration.

---

# Summary / Key Learnings

- Efficient data import & bulk operations.

- Preventing duplicates ensures data integrity.

- Regular backups protect data.

- Change Sets & VS Code deployments streamline migration.

- Understanding managed/unmanaged packages is essential for AppExchange publishing.

---

End of Phase 8 Documentation