ChatGPT

Salesforce Automation Runbook

Overview

This project implements a **template-driven, reusable** automation system for Salesforce Leads. When a Lead is created (for example from a Web-to-Lead form), a record-triggered Flow sends a templated email to the new Lead with a PDF attachment and optionally notifies internal users. A PowerShell test script provides deterministic end-to-end validation and writes debug information to the Lead record so you can rapidly diagnose issues without manually inspecting logs.

The primary design goals are:

- **Reusability** – one Apex action and one Flow pattern can drive many campaigns by swapping template IDs and attachment IDs. You do **not** need to change Apex code when you add more email campaigns.
- **Maintainability** – email content lives in Salesforce Email Templates, flows are declarative, and all IDs and settings are passed into the Apex action. This keeps the codebase small and easy to audit.
- **Debuggability** – the system surfaces its progress directly on the Lead record (`Lead.Description`) with predictable status tokens (see Debugging & Test Workflow). A single test script exercises the entire pipeline and prints the debug field back to the console.

Use this runbook to understand, operate, and extend the system. It documents how the components fit together, how to configure your org, how to add new campaigns, and how to troubleshoot common problems.

Quick Start

These steps run the entire pipeline and verify that a Lead triggers the automation.

1. Open a terminal in the project root.
2. Navigate to your local Salesforce project directory (for example `E:\Salesforce`). The runbook assumes PowerShell on Windows but you can adapt the commands for other shells.

```
cd E:\Salesforce
```

1. Execute the test runner script. It will create a test Lead in your authenticated org, wait for the Flow and Apex action to run, and then query back the Lead to print its debug status.

```
.\test_lead_flow.ps1
```

1. Expected outcomes:

2. A new Lead record is created.

3. `Lead.Description` contains a status value (e.g. `EMAIL_SENT | LeadId=...`).
4. The Lead receives an email that uses the specified template and includes the attached PDF file.
5. If you enabled internal notification, the configured internal user receives an alert email with Lead details.

If any of these outcomes fail, use the debug status and the Troubleshooting section to isolate the issue.

Architecture Overview

Flow & Apex Interaction

When a Lead is inserted, a record-triggered Flow (`After Save`) runs. The Flow calls an Apex **Invocable Action** named `LeadResumeEmailAction`. This action accepts several inputs:

Input	Description
`leadId`	The Id of the Lead being processed.
`emailTemplateId`	Id of the Email Template to send (00X…).
`orgWideFromAddress`	Email address from your Org-Wide Email Addresses.
`contentDocumentId`	Id of the File to attach (069…).

The Apex action performs the following steps:

1. Query the latest `ContentVersion` for the supplied `ContentDocumentId`.
2. Convert the file's binary into a `Messaging.EmailFileAttachment`.
3. Send an email using `Messaging.sendEmail()` with the template and attachment.
4. Return control back to the Flow.

The Flow also updates `Lead.Description` at multiple points to expose a deterministic status. The status values, in order, are:

- `STARTED` – Flow has begun processing.
- `CALLING_EMAIL` – Apex action invocation is about to occur.
- `EMAIL_SENT` – Apex action successfully sent the email.
- `EMAIL_FAULT | ERROR=<details>` – An exception occurred during send. The error message is appended after `ERROR=`.

An optional **Send Email** node in the Flow can dispatch an internal notification. This sends an alert from an Org-Wide Email Address to a configured recipient (no personal addresses are stored in code; set this value in the Flow builder).

File & Template Reuse

The Apex action is deliberately generic. By passing in the Ids of the template, file, and sender, you can reuse the same Flow and Apex code for many campaigns. To add a new campaign, you create a new Email Template and update the Flow inputs (see Extending the System).

Repository Layout

The repository is organized so that all source of truth lives in the `force-app/main/default` directory. Only modify files in this tree and deploy them with Salesforce CLI.

Directory / File	Purpose
`force-app/main/default/classes/LeadResumeEmailAction.cls`	Apex invocable action used by the Flow.
`force-app/main/default/flows/Lead_Email_ResumeAttachment_OnCreate.flow-meta.xml`	Primary Flow that triggers on Lead creation.
`force-app/main/default/email/Leads/<Template>.email`	HTML body of your Email Template.
`force-app/main/default/email/Leads/<Template>.email-meta.xml`	Metadata for the corresponding template.
`test_lead_flow.ps1`	PowerShell script that creates a test Lead and prints debug output.

Files such as `mdapi_deploy_*` and `*.zip` are historical deployment artifacts and are **not** source of truth; ignore them.

Prerequisites & Setup

Before running the system, ensure the following:

1. **Salesforce CLI** (`sf`) is installed and available in your shell.
2. You have an authenticated org alias or username stored in a PowerShell variable (e.g. `$Org`). For example, run `sf org login web` or `sf org display` to confirm.

3. An **Org-Wide Email Address** exists in your org for the sender address you intend to use (e.g. `noreply@yourdomain.com`).
4. You have created at least one Email Template and uploaded the PDF you intend to attach via Salesforce Files.

Configuration Values

Several Ids must be provided to the Flow and test script. Use the Salesforce CLI to find them. The sample queries below assume you have your org alias stored in `$Org`.

EmailTemplate Id (00X...)

Run the following to list your most recently modified Email Templates:

```
sf data query -o $Org -q "SELECT Id, Name, DeveloperName, FolderName FROM
EmailTemplate ORDER BY LastModifiedDate DESC LIMIT 20"
```

Choose the `Id` for the template you want to send. This value is passed to the Flow as `emailTemplateId`.

Resume ContentDocumentId (069...)

To find the file that will be attached, query Salesforce Files (ContentVersion):

```
sf data query -o $Org -q "SELECT Id, Title, FileType, ContentDocumentId,
VersionNumber, CreatedDate FROM ContentVersion WHERE Title LIKE '%Resume%' ORDER
BY CreatedDate DESC LIMIT 10"
```

Use the `ContentDocumentId` value (the field starting with `069`), not the `ContentVersion` Id (`068...`). This Id is passed to the Flow as `contentDocumentId`.

Org-Wide Email Address

List your Org-Wide Email Addresses:

```
sf data query -o $Org -q "SELECT Id, Address, DisplayName, IsAllowAllProfiles
FROM OrgWideEmailAddress ORDER BY Address"
```

Use the `Address` field (e.g. `noreply@yourdomain.com`). This string becomes the `orgWideFromAddress` input to the Apex action.

Changing Email Content

Email content lives in Salesforce Email Templates. You have two ways to modify it:

Option A – Salesforce Setup UI

1. Navigate in Salesforce to *Setup* → *Email Templates*.
2. Open your template and edit the Subject and HTML Body fields.
3. Save your changes. Deploying from this repository is not necessary for UI edits.

Option B – Repository (recommended for version control)

1. Edit the corresponding file under `force-app/main/default/email/Leads/<Template>.email` and its metadata file. Use HTML for the body.
2. Deploy the template with Salesforce CLI:

```
sf project deploy start -o $Org -m "EmailTemplate:Leads/<TemplateDeveloperName>"
```

Version control of your templates ensures reproducible deployments and easy code review.

Extending the System

There are two common patterns for adding new campaigns or variants.

Pattern A – Single Flow, Template Switch (Recommended)

Create one “router” Flow that inspects Lead fields such as `LeadSource`, `Company`, `Campaign` flags, or any custom metadata. Based on these values, the Flow chooses which `emailTemplateId` and `contentDocumentId` to pass to the Apex action. Pros:

- Fewer flows to maintain; all logic lives in one place.
- Centralized debug surface.

Cons:

- The Flow becomes a dispatcher and may grow complex if many rules exist.

Pattern B – One Flow per Campaign

Clone the base Flow for each campaign and hard-code different template and attachment IDs inside each copy. Pros:

- Simple mental model; each Flow maps 1:1 to a campaign.
- Isolated deployments; one change doesn’t affect others.

Cons:

- Many flows clutter the org if you create dozens of variants.

Adding a New Template + Flow (Step-by-Step)

1. **Create a Template** – add a new file under `force-app/main/default/email/Leads/` with the desired content. Deploy it using the CLI command above.
2. **Clone the Flow** – copy `Lead_Email_ResumeAttachment_OnCreate.flow-meta.xml` to a new file (rename both the filename and internal `<label>` and API name). Update the Apex action's inputs (`emailTemplateId` , `contentDocumentId` , and optionally `orgWideFromAddress`).
3. **Deploy the Flow** – deploy your cloned flow directory using:

```
sf project deploy start -o $Org -d "force-app/main/default/flows"
```

1. **Test** – run the test script and confirm that the new template is delivered with the expected attachment and debug status.

Swapping the Attached Resume

To change which file is attached, upload the new PDF to Salesforce Files and update the `contentDocumentId` value passed to the Apex action. Verify the file's latest version before deployment:

```
$Doc="069xxxxxxxxxxxxxxx"
sf data query -o $Org -q "SELECT Id, Title, ContentSize, FileType,
VersionNumber, CreatedDate FROM ContentVersion WHERE ContentDocumentId = '$Doc'
ORDER BY VersionNumber DESC LIMIT 1"
```

The query confirms the latest version number; the Apex action always picks the highest `VersionNumber` for the specified `ContentDocumentId`.

Internal Notification

You can send an internal alert each time a new Lead triggers the Flow. Add a **Send Email** element to the same Flow and configure these inputs:

- **Recipient Addresses** – the internal email(s) to notify, separated by semicolons.
- **Sender Type** – `OrgWideEmailAddress`.
- **Sender Email Address** – select the Org-Wide address used for outbound mail.
- **Subject** – for example:

```
New Lead Created: {!$Record.FirstName} {!$Record.LastName} ({!$Record.Email})
```

- **Body** – include Lead details such as name, email, company, lead source, created date, and record Id using merge fields. Example:

```
Name: {!$Record.FirstName} {!$Record.LastName}
Email: {!$Record.Email}
Company: {!$Record.Company}
Lead Source: {!$Record.LeadSource}
Created: {!$Record.CreatedDate}
Lead Id: {!$Record.Id}
```

Place the notification either **before** or **after** the Apex send step depending on whether you want alerts for all Leads or only for successful sends.

Debugging & Test Workflow

Test Script Usage

Always use the provided PowerShell script for deterministic testing:

```
.\test_lead_flow.ps1
```

The script creates a Lead, waits for the Flow and Apex to complete, and queries back the `Lead.Description` field. It prints the debug status to the console so you can see exactly where the process stopped.

Reading the Debug Status

The Flow writes a status token into `Lead.Description` at each stage:

Status	Meaning
`STARTED`	The Flow has begun running.
`CALLING_EMAIL`	The Flow is about to call the Apex action.
`EMAIL_SENT`	The Apex action completed and the email was delivered.
`EMAIL_FAULT | ERROR=<details>`	An error occurred during send; the error message is included.

If you see `EMAIL_FAULT`, read the error message to diagnose the problem (for example, invalid template Id or missing attachment). Use the Troubleshooting section to guide your investigation.

Troubleshooting

Below are common issues and how to resolve them. Use the debug status and CLI queries to narrow down the cause.

"Org-Wide Email provided is not valid"

- The `orgWideFromAddress` input must match an Address that exists in `OrgWideEmailAddress`. Run the query in the Configuration Values section and verify the address string matches exactly (case matters).

Lead Created but No Attachment

- Ensure that you pass a **ContentDocumentId** (starts with `069`), not a **ContentVersionId** (starts with `068`).
- Use the ContentVersion query above to confirm that there is at least one version for the document Id you specified.

Email Sends but Internal Notification Missing

- Check that the Send Email element in the Flow is active and that the recipient address field is populated.
- Confirm that daily email limits in Salesforce have not been exceeded.

General Debugging Tips

- Always test using the script and read the debug status. It is your first indicator of where the Flow stopped.
- If a new template or flow is added, deploy the template and flow, then run the test script. Confirm that the email arrives with the attachment and that any internal notification arrives if enabled.
- If you need to inspect raw Apex errors, temporarily set debug logging in Salesforce Setup and monitor the debug logs for the running user.

Glossary

Term	Definition
Flow (Record-Triggered)	A declarative automation that runs when a record is created or updated. In this system, the Flow triggers after a Lead is created.
Email Template	A Salesforce template containing subject and HTML body with merge fields. The Apex action uses the template to send consistent emails.
Org-Wide Email Address	A verified sender address that can be selected by Flows, Process Builders, or Apex. It ensures emails are sent from a trusted domain.

Term	Definition
ContentDocumentId (069...)	The Id for a file container in Salesforce Files. Multiple versions of a file share the same ContentDocumentId.
ContentVersionId (068...)	The Id of a specific file version. Do not use this Id in the Apex action; use the ContentDocumentId instead.
Apex Invocable Action	A public Apex class and method that can be called from Flow. Here, `LeadResumeEmailAction` sends the email and attachment.

Deployment Commands

Use the following CLI commands to deploy components from this repository. All commands assume you have your org alias in `$Org`.

Operation	Command
Deploy Apex classes	`sf project deploy start -o $Org -d "force-app/main/default/classes"`
Deploy Flows	`sf project deploy start -o $Org -d "force-app/main/default/flows"`
Deploy Email Templates	`sf project deploy start -o $Org -d "force-app/main/default/email"`
Run end-to-end test	`.\test_lead_flow.ps1`

Repo Hygiene

To keep your version control history clean:

- Do **not** commit local backup or temporary files. Exclude patterns such as `*.bak`, `*.localbak_*`, `_archive*`, `_disabled_*`, temporary zipped deployment folders, and Apex debug logs.
- Add these patterns to `.gitignore` so that accidental files are not staged.

Final Notes

This runbook provides a structured reference for operating and extending the Salesforce Lead email automation. By following the patterns described here—template-driven configuration, a reusable Apex action, clear debug signals, and CLI-driven testing—you can confidently add campaigns, swap attachments, and troubleshoot issues with minimal risk. Keep this document updated as your automation evolves.