


Phase 8: Data Management & Deployment

1. Data import Wizard

- **Definition:**
- Data Import Wizard ek Salesforce ka built-in tool hai.
- **Purpose:**
- Manual data entry ki jagah fast aur accurate import provide karna.
- Projects me testing aur data migration ke liye use hota hai.
- **Usage in Project:**
- Client data ko CSV se Salesforce me migrate karna.
- Marketing event ke leads bulk me import karna.



SETUP

Bulk Data Load Jobs

Job ID	750gL00000FJadR	Job Type	Bulk V1	Status	Closed
Submitted By	Urmil shrivas	Operation	Upsert	Total Processing Time (ms)	695
Start Time	10/11/2025, 1:11 AM PST	Queued Batches	0	API Active Processing Time (ms)	488
End Time	10/11/2025, 1:11 AM PST	In Progress Batches	0	Apex Processing Time (ms)	183
Time to Complete ([hh:mm:ss])	00:01	Completed Batches	1		
Object	Contact	Failed Batches	0		
External ID Field	Id	Progress	100%		
Content Type	CSV	Records Processed	20		
Concurrency Mode	Parallel	Records Failed	0		
API Version	65.0	Retries	0		

Reload

Batches

View Request	View Result	Batch ID	Start Time	End Time	Total Processing Time (ms)	API Active Processing Time (ms)	Apex Processing Time (ms)	Records Processed	Records Failed	Retry Count	State Message	Status
View Request	View Result	751gL00000Cc2ec	10/11/2025, 1:11 AM	10/11/2025, 1:11 AM	695	488	183	20	0	0		Completed

2. Data Loader

- Data Loader is a client application used for bulk data operations like insert, update, upsert, delete, and export.
- It efficiently handles large datasets and supports advanced field mapping and automation.
- Not implemented for this project since the dataset is small and all records were handled using the Data Import Wizard.

3. Duplicate rules

- The Standard Contact Duplicate Rule automatically prevents duplicate contacts in Salesforce.
- It checks matching criteria like Email, First Name, and Last Name to identify duplicates.

- In this project, it was tested to ensure duplicate contacts trigger an alert without blocking critical data entry.

SETUP

d

Duplicate Rules

Contact Duplicate Rule

Standard Contact Duplicate Rule

Help for this Page

Duplicate Rule Detail

EditDeleteCloneDeactivate

Rule Name

Standard Contact Duplicate Rule

Order

1 of 1

Reorder

Description

Identify contacts that duplicate other contacts and leads.

Object

Contact

Record-Level Security

Enforce sharing rules

Action On Create

Allow

Operations On Create

☒ Alert

☒ Report

Action On Edit

Allow

Operations On Edit

☐ Alert

☒ Report

Alert Text

Use one of these records?

Active

☒

Matching Rule

☒ Standard Contact Matching Rule

☒ Mapped

Matching Criteria

Matching rule for contact records. [More Info](#)

Matching Rule

☒ Standard Lead Matching Rule

☒ Mapped

Matching Criteria

Matching rule for lead records. [More Info](#)

Conditions

Created By

OrgFarm EPIC, 7/17/2025, 12:23 AM

Modified By

OrgFarm EPIC, 7/17/2025, 12:23 AM

EditDeleteCloneDeactivate

4. Data export and Backup

- Data Export was used to create a secure backup of all project records in CSV format.
- It ensured that Contacts, Leads, and Accounts are safely stored offline.
- This was done to prevent data loss and maintain data integrity for the project.

Monthly Export Service

[Help for this Page](#)

Data Export lets you prepare a copy of all your data in salesforce.com. From this page you can start the export process manually or schedule it to run automatically. When export is ready for download you will receive an email containing a link that allows you to download the file(s). The export files are also available on this page for 48 hours, after which time they are deleted.

Next scheduled export:

None

Export Now

Schedule Export

Scheduled By

Urmi shrivas

Schedule Date

10/11/2025

Export File Encoding

ISO-8859-1 (General US & Western European, ISO-LATIN-1)

Action

File Name

File Size

download

WE_00DgL000007QGpxUAG_1.ZIP

254.6K

5. Change sets

- Change Set is used to deploy metadata components like fields, objects, and rules from one Salesforce org to another.
- It allows smooth migration between sandbox and production environments without manual recreation.
- Not used in this project because all development was done in a single org, so deployment between orgs was not needed.

6. Unmanaged vs Managed Packages

- **Unmanaged Package:** A package where components (like objects, fields, and code) can be freely modified after installation in the target org.
- **Managed Package:** A controlled package where components are locked, updates are managed by the publisher, and it is mostly used for AppExchange apps.
- **In this project, packages were not required** as all development was done directly within a single org.

7. ANT Migration Tool

- The ANT Migration Tool is a command-line utility used to deploy and retrieve metadata between Salesforce orgs.
- It enables automated, bulk, and scripted migration, ideal for large or complex projects.

8. VS Code & SFDX

- VS Code with SF CLI was set up to manage project metadata and connect the local project to the Salesforce org.
- Commands like `sf org list`, `sf project status`, and `sf org open` were used to verify the connection.
- Some errors appeared at multiple points, so most development and testing were completed directly in the Salesforce org (browser).

SEARCH

DEBUGGER-PROJECT

force-app \ main \ default

lwc

accountDashboard

JS accountDashboard.js

accountDashboard.js-meta.xml

accountManager \ _tests_

accountManager.html

JS accountManager.js

accountManager.js-meta.xml

JS accountManager.test.js

objects

permissionsets

staticresources

tabs

triggers

LocalServiceCRM

.husky

.vscode

config

force-app

scripts

OUTLINE

TIMELINE

force-app > main > default > lwc > accountManager > _tests_ > JS accountManager.js > AccountManager

4 export default class AccountManager extends LightningElement {

10 handleInputChange(event) {

11 this[event.target.name] = event.target.value;

12 }

13

14 handleCreateAccount() {

15 // Imperative Apex Call

16 createAccount({ name: this.name, type: this.type, phone: this.phone })

17 .then(result => {

18 this.createdAccountId = result;

19 console.log('Account created with Id: ' + result);

20 // Optionally, dispatch an event to parent

21 const createdEvent = new CustomEvent('accountcreated', { detail: { id: result, name:

22 this.dispatchEvent(createdEvent);

23

24 // Reset inputs

25 this.name = '';

26 this.type = '';

27

28 })

29 .catch(error => {

30 console.error('Error creating account:', error);

31 });

32 }

33 }

34

See Real World Examples From GitHub

var console: Console

