Career Guidance: How to Start a Career in Salesforce

Salesforce has become one of the most in-demand platforms in the world of CRM (Customer Relationship Management). With businesses adopting Salesforce globally, the demand for skilled professionals is skyrocketing. Whether you're a fresher, an IT professional looking for a switch, or someone from a non-technical background, Salesforce offers diverse career opportunities.

Why Choose Salesforce?

1. **High Demand**: Salesforce is the #1 CRM platform with millions of users worldwide.
2. **Attractive Salaries**: Salesforce roles are among the highest-paying in the IT industry.
3. **Diverse Roles**: Opportunities exist for Developers, Administrators, Business Analysts, Architects, and Consultants.
4. **Future-Proof Career**: Salesforce continues to grow rapidly with new innovations like AI (Einstein) and automation.
5. **No Strong Prerequisites**: Even non-technical graduates can build a career in Salesforce.

Career Pathways in Salesforce

1. Salesforce Administrator

- Focus: Managing users, customizing dashboards, automating processes.
- Best for: People with problem-solving skills, less coding.
- Certifications: Salesforce Administrator (ADM 201).

2. Salesforce Developer

- Focus: Coding with **Apex, Lightning Web Components (LWC), SOQL, and Triggers**.
- Best for: Those who enjoy programming.
- Certifications: Salesforce Platform Developer I & II.

3. Salesforce Consultant

- Focus: Implementing Salesforce solutions for clients.
- Best for: People with good business and communication skills.

4. Salesforce Architect

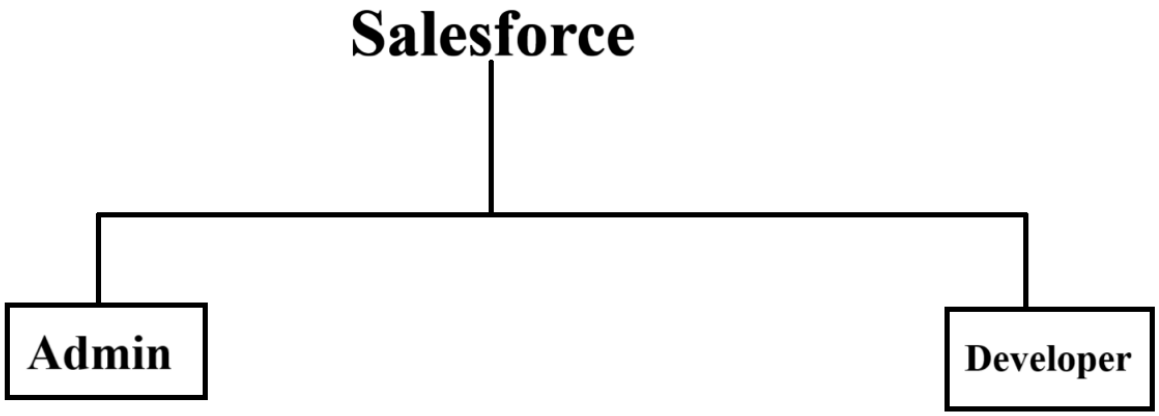
- Focus: Designing scalable solutions.
- Requires: Deep technical + business expertise.
- Certifications: Technical Architect, Application Architect, System Architect.

5. Salesforce Business Analyst

- Focus: Gathering requirements, bridging business & technical teams.
- Best for: Analytical thinkers with minimal coding.

6. Salesforce Marketing / Service / Commerce Specialist

- Focus: Niche roles in Marketing Cloud, Service Cloud, and Commerce Cloud.
- Best for: Professionals with domain-specific interests.



Salesforce Admin Topics (Complete List)

---

1. Salesforce Basics

- Salesforce Architecture (Multi-tenant, Metadata-driven, Cloud model)
- Editions, Licenses, and Features
- Company Profile (Fiscal Year, Business Hours, Holidays, Currencies)

---

2. User Setup & Security

- Creating & Managing Users

- Profiles & Permission Sets
- Permission Set Groups
- Roles & Role Hierarchy
- Organization-Wide Defaults (OWD)
- Sharing Rules (Criteria-based, Owner-based)
- Manual Sharing & Team Sharing (Account, Opportunity, Case Teams)
- Login IP Ranges, Login Hours
- Two-Factor Authentication (2FA)
- Session Security Settings
- Security Health Check

---

3. Data Modeling

- Objects (Standard vs Custom)
- Fields (Standard, Custom, Formula, Roll-Up Summary, Geolocation)
- Field Types (Lookup, Master-Detail, External Lookup, Indirect Lookup, Hierarchical)
- Relationships (Lookup, Master-Detail, Many-to-Many with Junction Objects, Self-Relationship)
- Field Dependencies (Controlling & Dependent Picklists)
- Schema Builder

---

4. Data Management

- Data Import Wizard
- Data Loader (Insert, Update, Upsert, Delete, Export)
- External IDs
- Deduplication (Duplicate Rules, Matching Rules)
- Data Skew Issues
- Large Data Volume Best Practices
- Mass Transfer Records, Mass Delete Records
- Data Archiving & Storage Management

---

5. Automation Tools

- Workflow Rules (legacy, deprecated)
- Process Builder (legacy, deprecated)
- **Flow Builder (current & most important):**
    - Record-Triggered Flow
    - Screen Flow
    - Autolaunched Flow
    - Schedule-Triggered Flow
    - Subflows

- Approval Processes (Single/Multi-step)
- Assignment Rules (Lead & Case)
- Escalation Rules
- Email Alerts & Templates

---

6. UI Customization

- Page Layouts
- Record Types
- Business Processes (Sales, Support, Lead)
- Compact Layouts
- Path & Guidance for Success
- Lightning App Builder (Record Page, Home Page, App Page)
- Dynamic Forms & Dynamic Actions
- Global Actions vs Object-Specific Actions
- Buttons, Links, Quick Actions

---

7. Reports & Dashboards

- Report Types (Standard, Custom, Joined)
- Report Formats (Tabular, Summary, Matrix, Joined)
- Filters (Cross Filters, Field-to-Field Filters)
- Bucket Fields
- Row-Level Formulas
- Dashboards (Components, Filters, Dynamic Dashboards)
- Report & Dashboard Subscriptions

---

8. Sales Cloud Features (Admin Scope)

- Leads & Lead Conversion
- Campaigns & Campaign Influence
- Opportunities (Stages, Products, Quotes, Price Books)
- Forecasting (Collaborative Forecasts)
- Territory Management

---

9. Service Cloud Features (Admin Scope)

- Case Management
- Queues & Assignment Rules
- Case Escalation Rules

- Email-to-Case, Web-to-Case
- Service Console
- Knowledge Base (Articles, Data Categories)
- Entitlements & Milestones (SLA)

---

10. Data Security & Compliance

- Field-Level Security
- Record-Level Security (Role Hierarchy, Sharing Rules, Teams)
- Shield Platform Encryption (awareness)
- Data Masking
- Audit Trail & Setup Audit Trail
- Login History
- GDPR / Compliance Awareness

---

11. AppExchange & Integrations (Admin Level)

- AppExchange (Install, Manage, Uninstall Apps)
- Connected Apps & OAuth Basics
- Email Integration (Outlook, Gmail, Einstein Activity Capture)
- Salesforce Mobile App (Setup, Compact Layouts, Navigation)

---

12. Environment Management

- Sandboxes (Dev, Partial, Full, Scratch Org awareness)
- Change Sets (Inbound & Outbound)
- Deployment Basics (Metadata Migration)
- Salesforce Optimizer
- App Packaging (Basics)

---

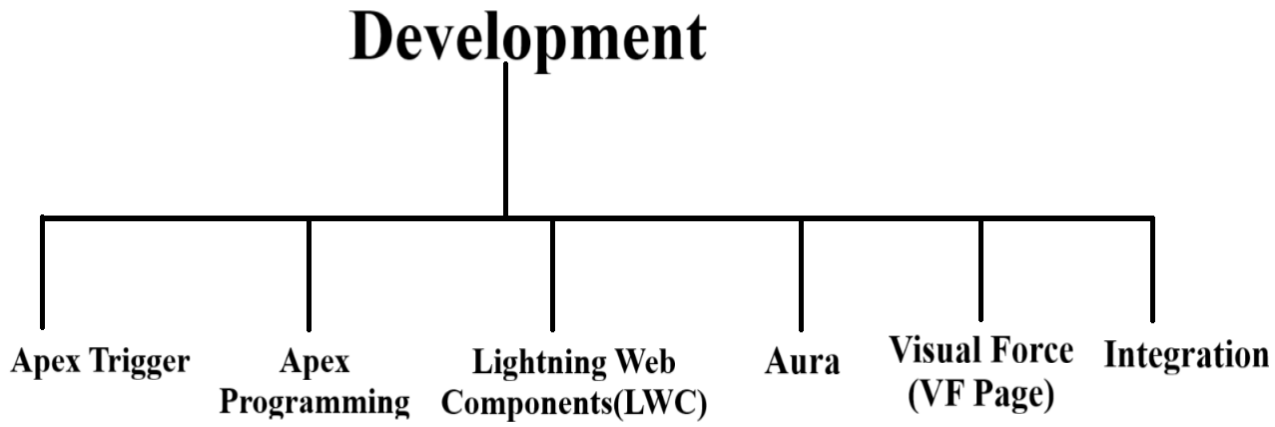
13. Collaboration

- Chatter (Groups, Feed Tracking, Publisher Actions)
- Slack Integration (awareness)

---

14. Monitoring & Troubleshooting

- Setup Audit Trail
- Debug Logs (for Flows, Automation)
- Login History
- Optimizer Reports
- System Status (trust.salesforce.com)



Salesforce Apex Trigger Topics (Complete List)

1. Basics & Syntax

- Introduction to Triggers
- Trigger Events (before/after insert, update, delete, undelete)
- Trigger Context & Structure
- Trigger.new, Trigger.old, Maps

2. Order of Execution

- Full Salesforce Order of Execution
- Difference between before & after triggers

3. Context Variables

- `isInsert`, `isUpdate`, `isDelete`, `isUndelete`
- `isBefore`, `isAfter`
- `Trigger.new`, `Trigger.old`
- `Trigger.newMap`, `Trigger.oldMap`
- `Trigger.operationType`

4. Bulkification

- Why bulkification is required
- SOQL/DML in loops
- Using Maps & Sets
- Handling collections in triggers

5. Recursion Handling

- Cause of recursion in triggers
- Static variables to prevent recursion
- Field-delta checks

6. Before vs After Triggers

- When to use before triggers
- When to use after triggers
- Use-case matrix

7. Special Scenarios

- Upsert behavior
- Merge operation
- Lead Conversion
- Delete vs Undelete

8. DML & SOQL in Triggers

- Governor limits in triggers
- Best practices with DML
- Handling large data volumes (LDV)

9. Error Handling

- Using `addError()`
- Field-level vs record-level errors

10. Cross-Object Logic

- Parent-to-child updates
- Child-to-parent updates
- Roll-up alternatives

11. Async Operations

- Queueable from trigger
- Future methods (legacy)
- Batch Apex invocation
- Platform Events from triggers

12. Callouts

- Why direct callouts aren't allowed
- Queueable with callouts
- Platform Event → Integration pattern

13. Security & Sharing

- Triggers run in system mode
- Enforcing CRUD/FLS manually
- `with sharing` vs `without sharing` in handler classes

14. Design Patterns

- One trigger per object principle
- Trigger Handler class pattern
- Switchboard pattern
- fflib (Apex Commons) trigger framework
- Metadata-driven trigger frameworks

15. Performance & Best Practices

- Idempotent trigger logic
- Avoiding unnecessary updates
- Logging & debug strategies

16. Testing Triggers

- Writing bulk tests
- Negative test cases (`addError`)
- Merge/Undelete test cases
- Using Test Data Factory
- Testing async jobs from triggers

17. Deployment & Lifecycle

- Deployment order (objects → fields → triggers → classes)
- Version control & CI/CD
- Feature toggles for triggers

18. Triggers vs Automation Tools

- Trigger vs Workflow vs Process Builder vs Flow
- Coexistence with Flows (hybrid approach)
- Migration from legacy automation to triggers

19. Real-Time Scenarios

- Preventing duplicates
- Auto-updating related objects
- Enforcing complex validations
- Ownership propagation
- Complex approval restrictions

20. Advanced Topics

- Record locking & `UNABLE_TO_LOCK_ROW`
- Handling multi-currency
- Multi-language error messages
- Trigger on ContentDocument, EmailMessage, Knowledge
- Change Data Capture (CDC) vs Triggers

Salesforce Apex Programming Topics (Complete List)

1. Apex Basics

- What is Apex?
- Apex execution context
- Syntax & data types (primitive, sObject, collections)
- Variables & constants
- Operators & expressions
- Control structures (`if-else`, `switch`, loops)

2. Object-Oriented Concepts in Apex

- Classes & Objects
- Constructors
- Methods (static vs instance)
- Access Modifiers (public, private, global, protected)
- Encapsulation, Inheritance, Polymorphism, Abstraction
- Interfaces
- Virtual, Abstract & Final classes

3. Data Types in Depth

- Primitive data types (Integer, String, Boolean, etc.)
- sObjects (Standard vs Custom)
- Collections: List, Set, Map
- Enum types
- Blob, Date, DateTime, Time

4. Collections & Advanced Usage

- Lists (add, remove, sort, search)
- Sets (uniqueness, membership tests)
- Maps (key-value, keySet, values, entries)
- Nested collections (List<Map<Id, sObject>>)
- Iterating collections efficiently

5. SOQL (Salesforce Object Query Language)

- Basic queries (SELECT … FROM … WHERE)
- Relationship queries (parent-to-child, child-to-parent)
- Aggregate functions (COUNT, SUM, AVG, MAX, MIN)
- Group By, Having clauses
- Dynamic SOQL (`Database.query()`)
- SOQL For Loops (to avoid heap limits)
- Governor limits on SOQL

6. SOSL (Salesforce Object Search Language)

- Basics of SOSL
- Returning multiple object types
- When to use SOQL vs SOSL
- Dynamic SOSL

7. DML Operations

- Insert, Update, Upsert, Delete, Undelete
- Bulk DML operations
- Database class methods (`Database.insert`, `Database.update`, etc.)
- Partial success & `Database.SaveResult`
- Handling DML exceptions
- Transaction management (Savepoints & Rollbacks)

8. Governor Limits

- SOQL/DML limits
- CPU time limits
- Heap size limits
- Best practices to avoid hitting limits
- Tools to monitor (Developer Console, Debug Logs)

9. Exception Handling

- Try, Catch, Finally
- Custom Exceptions
- Throwing exceptions (`throw`)
- `addError()` vs throwing an exception
- Checked vs Unchecked exceptions

10. Apex Triggers (in context of programming)

- Trigger events (before/after insert, update, delete, undelete)
- Context variables
- Bulkification in triggers
- Recursion handling
- Trigger frameworks & best practices

11. Asynchronous Apex

- Future methods (`@future`)
- Batch Apex
- Queueable Apex
- Scheduled Apex
- Continuations
- Best practices for async Apex

12. Apex Testing

- Test classes & methods
- `@isTest` annotation
- `Test.startTest()` and `Test.stopTest()`
- Test data creation vs `@testSetup`
- Asynchronous job testing
- Code coverage vs meaningful test coverage
- Negative test cases
- `SeeAllData=true/false`

13. Apex Integration

- REST callouts (GET, POST, PUT, DELETE)
- SOAP callouts (WSDL2Apex)
- HTTP classes (`HttpRequest`, `HttpResponse`)
- Named Credentials
- Callouts from Queueable Apex
- Handling JSON & XML (`JSON.deserialize`, `JSON.serialize`)
- Authentication (OAuth 2.0, JWT, Connected Apps)

14. Apex & Platform Events

- Publishing platform events from Apex
- Subscribing to platform events with triggers
- ReplayId handling

- Event-driven architectures with Apex

15. Advanced Apex Concepts

- Dynamic Apex (`Schema` namespace)
- Dynamic SOQL & SOSL
- Dynamic DML
- Describe calls (field describe, object describe)
- Custom Metadata / Custom Settings access via Apex
- sObject tokenization (`Account.SObjectType`)
- Dependency Injection (using interfaces & factories)
- Apex Design Patterns (Singleton, Factory, Strategy, Unit of Work)

16. Security in Apex

- Apex runs in system mode vs user mode
- Enforcing CRUD & FLS in Apex (`Schema.sObjectType.Account.fields.Name.isAccessible()`)
- `with sharing` vs `without sharing`
- Inherited sharing
- Secure coding practices

17. Apex & Lightning

- Using Apex in Lightning Web Components (LWC)
- `@AuraEnabled` methods
- Cacheable Apex methods (`@AuraEnabled(cacheable=true)`)
- Calling Apex imperatively vs wire service
- Apex methods with parameters in LWC

18. Apex Performance Optimization

- Avoiding unnecessary queries/DML
- Using Maps & Sets for performance
- Selective SOQL queries
- Avoiding nested loops
- Best practices for large data volumes (LDV)
- Query & transaction optimization

19. Deployment & Packaging

- Deploying Apex code with Change Sets
- Source-driven development (SFDX, Git)
- CI/CD with Apex code
- Namespaces in managed packages
- Versioning Apex classes

20. Real-Time Scenarios & Use Cases

- Trigger vs Flow vs Process Builder vs Apex decision-making
- Complex validation rules using Apex
- Rollup summary replacement logic
- Ownership propagation
- Duplicate prevention logic
- Integration with external APIs
- Email services in Apex

Salesforce Lightning Web Component Topics (Complete List)

1. Introduction to LWC

- What is Lightning Web Component?
- Difference between **Aura vs LWC**
- Benefits of LWC (performance, modern JS, reusable)
- LWC Architecture (shadow DOM, base lightning components, services)

---

2. Setup & Basics

- Setting up LWC in Salesforce Org / VS Code
- Project structure (HTML, JS, XML files)
- Component lifecycle in LWC
- Naming conventions & best practices

---

3. HTML & Templates

- HTML structure in LWC
- Conditional rendering (`if:true`, `if:false`)
- Iterators (`for:each`, `iterator`)
- Data binding (one-way vs two-way)
- Template expressions (`{property}`)

---

4. JavaScript in LWC

- ES6+ Features (let, const, arrow functions, template literals, spread/rest)
- Variables & Constants

- Functions & Classes in JS
- Event handling in JS
- Getter & Setter methods in LWC
- Modules & Imports

---

5. LWC Decorators

- `@api` → Public properties/methods
- `@track` → Reactive private properties (legacy)
- `@wire` → Reactive service call to Salesforce data
- `@readonly` → Immutable properties (conceptual)
- `@AuraEnabled` → When exposing Apex methods to LWC

---

6. Data Binding & Events

- One-way & two-way binding
- Handling input fields (`onchange`, `onclick`, etc.)
- Event propagation: `bubbles`, `composed`
- Custom events (`CustomEvent`)
- Parent-to-child communication
- Child-to-parent communication
- Pub-Sub model (for unrelated components)
- Lightning Message Service (LMS)

---

7. LWC & Salesforce Data

- Using **@wire with Apex**
- Cacheable Apex (`@AuraEnabled(cacheable=true)`)
- Imperative Apex calls (`import method from '@salesforce/apex/...'`)
- SOQL & filtering in Apex for LWC
- Reactive parameters in @wire
- LDS (Lightning Data Service): `getRecord`, `getFieldValue`, `createRecord`, `updateRecord`, `deleteRecord`

---

8. Lightning Base Components

- `<lightning-input>`, `<lightning-button>`, `<lightning-card>`
- `<lightning-datatable>` (key interview topic!)
- `<lightning-record-form>`
- `<lightning-record-view-form>`
- `<lightning-record-edit-form>`

- `<lightning-layout>`, `<lightning-tabset>`, `<lightning-combobox>`

---

9. Styling in LWC

- CSS in LWC (component-specific, shadow DOM)
- SLDS (Salesforce Lightning Design System)
- Styling hooks
- Shared CSS across components
- Dynamic styling (conditional classes)

---

10. Lifecycle Hooks

- `constructor()`
- `connectedCallback()`
- `renderedCallback()`
- `disconnectedCallback()`
- `errorCallback(error, stack)`

---

11. Navigation & Utility Services

- NavigationMixin (navigate to record, list view, URL, etc.)
- ShowToastEvent for notifications
- RefreshApex for refreshing wired data
- Wire adapters (e.g., `getRecord`, `getObjectInfo`)

---

12. Integration with Apex

- Calling Apex imperatively
- Calling Apex with @wire
- Apex with parameters (dynamic filters)
- Handling errors in Apex calls
- Apex best practices for LWC

---

13. Lightning Message Service (LMS)

- What is LMS?
- Publishing a message
- Subscribing to a message
- Communication between LWC ↔ Aura ↔ VF using LMS

14. Composition Patterns

- Parent-to-child communication (`@api` methods/properties)
- Child-to-parent communication (custom events)
- Sibling communication (pub-sub / LMS)
- Slots (`<slot>`)
- Dynamic component creation

---

15. Advanced LWC Topics

- Reactive variables & reactivity system
- Working with large data (pagination, infinite scrolling)
- Using third-party libraries (Chart.js, D3.js, etc.)
- Using static resources in LWC
- Calling external APIs from LWC (via Apex callouts)
- Platform events with LWC
- Change Data Capture (CDC) with LWC
- File upload/download in LWC
- Error handling & debugging in LWC

---

16. Testing in LWC

- Jest Testing Framework for LWC
- Writing unit tests for components
- Mocking Apex in tests
- Code coverage in LWC tests

---

17. Deployment & Packaging

- Deploying LWC using Change Sets
- Deploying LWC using SFDX
- LWC in Managed Packages
- Source-driven development (CI/CD with GitHub Actions, Azure DevOps, Jenkins)

---

18. Performance & Best Practices

- Avoiding unoptimized loops & rerenders
- Using cacheable methods
- Debouncing input handlers

- Lazy loading components
- Optimizing DOM updates
- Best practices for large data tables

---

19. Security in LWC

- Locker Service
- Lightning Web Security (LWS)
- XSS prevention in templates
- CRUD & FLS enforcement via Apex
- Security review readiness

---

20. Real-Time Scenarios

- LWC for record forms (edit, create, view)
- LWC with datatable + inline editing
- Search & filter components
- Reusable modal components
- Multi-step wizard with LWC
- File upload preview with LWC
- Dynamic picklist using metadata
- Pagination component
- Integration with external REST API

Salesforce Aura Component Topics (Complete List)

1. Introduction to Aura

- What is Aura Framework?
- Difference between **Aura Components vs LWC**
- Aura component architecture (Component, Controller, Helper, Renderer)
- Component Bundle files (.cmp, .js, .css, .xml, etc.)

---

2. Aura Component Basics

- Creating a component (`.cmp`)
- Component markup structure
- `<aura:component>` tag basics
- Component attributes (`<aura:attribute>`)
- Data binding (`{! }`)
- Expressions (`value`, `v.`, `c.`)

---

3. Attributes & Value Providers

- Defining attributes (`aura:attribute`)
- Default values
- Access modifiers (public, private)
- Value providers: `v`, `c`, `this`
- Passing data between components

---

4. Event Handling

- Component Events
- Application Events
- Event phases (capture vs bubble)
- Registering an event (`aura:registerEvent`)
- Firing an event (`cmp.getEvent().fire()`)
- Handling an event (`aura:handler`)
- Event propagation (component → parent → application)

---

5. Controllers & Helpers

- Client-side Controller (JavaScript)
- Helper methods (logic reusability)
- Server-side Controller (Apex Controller)
- Invoking Apex from Aura (`@AuraEnabled`)
- Action parameters & return types
- Handling promises/callbacks (`setCallback`)

---

6. Component Communication

- Parent-to-child communication (attributes, aura:id)
- Child-to-parent communication (events)
- Sibling communication (via events / application event)
- Aura vs LWC communication patterns

---

7. Expressions in Aura

- Expression syntax (`{! }`)
- One-way vs two-way binding
- `aura:valueChange` event
- Formula expressions in attributes

---

8. Iteration & Conditional Rendering

- `aura:iteration` (looping over lists)
- `aura:if` (conditional rendering)
- Handling empty states

---

9. Lightning Base Components in Aura

- `<lightning:input>`
- `<lightning:button>`
- `<lightning:card>`
- `<lightning:datatable>`
- `<lightning:recordForm>`
- `<lightning:recordEditForm>`
- `<lightning:recordViewForm>`

---

10. Styling Aura Components

- Component-specific CSS (`.css` file)
- SLDS (Salesforce Lightning Design System)
- `aura:html` tag for HTML injection
- Applying conditional styles
- Shared CSS across components

---

11. Aura Component Lifecycle Hooks

- `init` (doInit handler)
- `render`
- `afterRender`
- `rerender`
- `unrender`
- `destroy`

12. Lightning Data Service (LDS)

- `force:recordData`
- Creating, updating, deleting records without Apex
- `getNewRecord` and `reloadRecord` methods
- Record form components (`lightning:recordForm`)

---

13. Navigation & Utility in Aura

- `lightning:navigation` service
- Toast messages (`force:showToast`)
- Modal dialogs (`lightning:overlayLibrary`)
- Spinner handling (`lightning:spinner`)

---

14. Security in Aura

- Locker Service basics
- Lightning Data Service & security
- CRUD & FLS checks in Apex
- Preventing XSS in Aura
- CSP (Content Security Policy) restrictions

---

15. Integration with Apex

- `@AuraEnabled` methods in Apex
- Calling Apex imperatively in Aura
- Passing parameters to Apex methods
- Handling errors & exceptions from Apex
- Bulk data handling via Aura + Apex

---

16. Performance Best Practices

- Avoiding unnecessary rerenders
- Using LDS instead of Apex where possible
- Bulkification with Aura + Apex
- Caching data (storable actions)
- Pagination handling in Aura datatables

---

17. Advanced Aura Concepts

- Component inheritance (`extends`)
- Interfaces (`implements`) → `force:appHostable`, `flexipage:availableForAllPageTypes`, etc.
- Aura services (`$A.get`, `$A.enqueueAction`)
- Dynamic component creation (`$A.createComponent`)
- Composition patterns (nested components)

---

18. Testing & Debugging Aura

- Using Chrome DevTools for Aura debugging
- Lightning Inspector extension
- Debug logs for server-side errors
- Jasmine/Karma test setups for Aura (rare, mostly manual)

---

19. Deployment & Packaging

- Deploying Aura components (Change Sets, SFDX)
- Namespaces in managed packages
- Source-driven development with Aura
- CI/CD for Aura bundles

---

20. Real-Time Scenarios

- Creating custom lookup component
- Aura datatable with inline edit + Apex
- Search component with SOSL
- Custom file upload component
- Pagination with Aura iteration
- Parent-child modal component
- Event-driven dashboards

Salesforce Visual Force Topics (Complete List)

1. Introduction to Visualforce

- What is Visualforce (VF)?
- When to use Visualforce vs LWC/Aura
- Visualforce page structure (`<apex:page>`)
- Page Controller (Standard Controller, Custom Controller, Extension)

---

2. Visualforce Basics

- VF markup syntax
- Standard Controllers (`<apex:page standardController="Account">`)
- Standard List Controllers
- Controller Extensions
- Custom Controllers

---

3. Visualforce Page Structure

- `<apex:page>` tag & attributes
- `<apex:form>`
- `<apex:pageBlock>` & `<apex:pageBlockSection>`
- `<apex:pageBlockTable>`
- `<apex:pageBlockButtons>`
- `<apex:facet>`

---

4. Visualforce Components

- Input Components (`<apex:inputText>`, `<apex:inputField>`, `<apex:inputCheckbox>`)
- Output Components (`<apex:outputText>`, `<apex:outputField>`, `<apex:outputPanel>`)
- Data Tables (`<apex:dataTable>`, `<apex:pageBlockTable>`)
- Command Components (`<apex:commandButton>`, `<apex:commandLink>`)
- Message Components (`<apex:messages>`, `<apex:pageMessages>`)
- Tabs & Layout (`<apex:tabPanel>`, `<apex:tab>`)
- Reusable Components (`<apex:component>`, `<apex:composition>`)

---

5. Data Binding & Controllers

- Binding to Standard Controllers
- Binding to Custom Controllers
- Using Controller Extensions

- Getter & Setter methods in controllers
- Passing parameters between pages (`<apex:param>`, `ApexPages.currentPage().getParameters()`)

---

6. Visualforce Expressions

- `{! }` expression syntax
- Global variables (e.g., `$User`, `$Profile`, `$Organization`)
- Formula expressions in VF pages

---

7. Action Components

- `<apex:actionFunction>`
- `<apex:actionSupport>`
- `<apex:actionRegion>`
- `<apex:actionStatus>`
- `<apex:actionPoller>`
- AJAX behavior in VF

---

8. Page Navigation

- Standard navigation with buttons & links
- PageReference in Apex (`new PageReference('/apex/MyPage')`)
- Redirects & URL parameters
- Passing parameters between VF pages

---

9. Styling in Visualforce

- CSS in VF pages (`<apex:stylesheet>`)
- Inline CSS
- Salesforce Lightning Design System (SLDS) in VF
- Using static resources for CSS/JS

---

10. JavaScript & VF

- Embedding JavaScript in VF (`<script>` tag, `<apex:includeScript>`)
- Using jQuery in VF
- Handling events with JS
- Calling Apex methods from JavaScript (Remote Objects, JS Remoting)

---

11. Visualforce Charts & Dynamic UI

- `<apex:chart>` components
- `<apex:repeat>` for dynamic rendering
- `<apex:dynamicComponent>`
- Conditional rendering (`rendered` attribute)

---

12. Advanced VF Features

- Custom Components (`<apex:component>`)
- Component attributes (`<apex:attribute>`)
- `<apex:composition>` for templates
- `<apex:define>` and `<apex:insert>` (templating)
- Visualforce Templates for reusability

---

13. Integration & Remote Calls

- JavaScript Remoting (`@RemoteAction`)
- Visualforce Remote Objects
- Calling REST/SOAP APIs from VF (via Apex)
- Using `<apex:iframe>` for embedding external apps

---

14. Visualforce & Salesforce1 / Mobile

- `<apex:page lightningStylesheets="true">` for Lightning UI
- VF Pages in Salesforce Mobile App
- Responsive design in VF

---

15. Security in Visualforce

- CRUD & FLS checks in controllers
- Security best practices (no hardcoded Ids)
- XSS prevention in VF (`<apex:outputText escape="true">`)
- CSRF & clickjacking protections
- Locker Service considerations for VF + LWC/Aura hybrid

---

16. Performance Optimization

- View State in VF (impact & size reduction)
- Reducing View State (transient variables, minimizing components)
- Using StandardSetController for pagination
- Lazy loading in VF
- Optimizing SOQL/DML in controllers

---

17. Testing VF Pages

- Writing Apex tests for VF controllers
- Testing Standard Controllers
- Testing Controller Extensions
- Using `PageReference` in tests
- Asserting Page parameters & redirects

---

18. Deployment & Packaging

- Deploying VF pages (Change Sets, SFDX)
- Namespaces in managed packages
- VF in Salesforce managed apps
- Versioning VF pages

---

19. VF with Lightning (Hybrid)

- Using VF pages inside Lightning Experience
- VF pages inside Lightning tabs
- Embedding LWC/Aura inside VF (and vice versa via `<lightning:out>`)
- Lightning Stylesheets in VF pages

---

20. Real-Time Scenarios

- Custom search page with VF
- VF datatable with pagination & inline editing
- VF wizard (multi-step forms)
- VF for generating custom reports
- File upload & preview in VF
- VF for custom approval screens
- Embedding external app via `<apex:iframe>`

Salesforce Integration Topics (Complete List)

1. Integration Basics

- What is Integration?
- Types of Integration: Data Integration vs Process Integration vs UI Integration
- Real-time vs Batch Integration
- Synchronous vs Asynchronous Integration
- Point-to-Point vs Middleware Integration

---

2. Salesforce APIs

- **REST API** (CRUD, Composite, Batch, Query, Search APIs)
- **SOAP API** (Enterprise WSDL, Partner WSDL)
- **Bulk API v1 & v2** (large data volumes)
- **Streaming API** (Push Topics, Generic Streaming)
- **Platform Events**
- **Change Data Capture (CDC)**
- **Metadata API**
- **Tooling API**
- **Apex REST API** (Custom REST services)
- **Apex SOAP API** (Custom SOAP services)
- **GraphQL API** (new, efficient querying)

---

3. Integration Patterns

- Remote Process Invocation — Request & Reply (Synchronous)
- Remote Process Invocation — Fire & Forget (Asynchronous)
- Batch Data Synchronization (Scheduled/Bulk)
- Remote Call-In (External System → Salesforce API)
- Data Virtualization (Access without storing)
- Publish/Subscribe (Event-Driven with Platform Events & CDC)
- Orchestration with Middleware

---

4. Authentication & Security

- OAuth 2.0 Flows (Web Server, User-Agent, Username-Password, JWT Bearer, Refresh Token)
- Connected Apps & Auth Providers
- Named Credentials (with/without password, OAuth)
- Certificates & Keys (Self-signed, CA-signed)
- Two-Way SSL (Mutual Authentication)
- Single Sign-On (SSO) with SAML / OpenID Connect
- External Identity Provider (IdP) vs Salesforce as IdP
- Multi-Factor Authentication (MFA) in integrations

---

5. Callouts from Salesforce

- HTTP Callouts using `HttpRequest` & `HttpResponse`
- REST Callouts with JSON/XML parsing
- SOAP Callouts with WSDL2Apex
- Named Credentials for simplified authentication
- Handling callout exceptions (timeouts, 401, 500 errors)
- Callouts from Future/Queueable/Batch Apex (async)
- Continuation (for long-running callouts)

---

6. Inbound Integrations to Salesforce

- REST API call-ins (external app calling Salesforce REST endpoints)
- SOAP API call-ins (Enterprise/Partner WSDL)
- Apex REST services (`@RestResource`)
- Apex SOAP services (`webservice` keyword)
- Web-to-Lead & Web-to-Case forms
- Email-to-Case & Email Services
- Salesforce Sites & Experience Cloud (external forms)

---

7. Middleware & External Systems

- Common middleware: MuleSoft (native), Dell Boomi, Informatica, Jitterbit, Workato
- ETL Tools vs ESB
- Integration via Kafka/Event Bus
- Salesforce Connect (OData adapters, External Objects)
- External Services (OpenAPI/Swagger to Apex actions)
- Data Loader / Data Import Wizard (simple batch)

---

8. Event-Driven Integrations

- Platform Events (publish/subscribe model)
- Change Data Capture (CDC) events
- PushTopic events
- Generic Streaming events
- Pub/Sub API (new unified API for event streaming)
- Event Bus Replay & durability
- Best practices for event-driven integrations

---

9. Large Data Volume (LDV) Integrations

- Bulk API (v1 vs v2) for millions of records
- Chunking strategies for data imports
- Optimizing queries for LDV integrations
- Asynchronous processing (Batch/Queueable/Platform Events)
- Data skew & locking considerations

---

10. Integration Error Handling & Monitoring

- Retry logic & Idempotency
- Dead-letter queue handling (for failed messages)
- Integration user strategy (dedicated user, least privileges)
- Named Credential rotation & monitoring
- Event Monitoring & Shield for tracking integrations
- Debug logs, Event logs, API usage monitoring
- Tools: Salesforce Shield, Splunk, NewRelic, MuleSoft Monitoring

---

11. Real-Time Integration Scenarios

- Salesforce ↔ ERP (SAP, Oracle, MS Dynamics)
- Salesforce ↔ Payment Gateways (Stripe, PayPal)
- Salesforce ↔ External Databases (Postgres, MySQL, SQL Server)
- Salesforce ↔ AWS/Azure/GCP (S3, Lambda, EventBridge)
- Salesforce ↔ Mobile Apps (OAuth + REST APIs)
- Salesforce ↔ Marketing Tools (HubSpot, Mailchimp, Marketo)
- Salesforce ↔ Slack (native integrations + APIs)

---

12. Integration Best Practices

- Choose right API (REST vs SOAP vs Bulk vs Streaming)
- Minimize API calls (Composite API, Bulk API)

- Enforce security (CRUD/FLS, OAuth, Named Credentials)
- Governor limits awareness
- Idempotent design (prevent duplicates)
- Resilient retry logic for async integrations
- Use middleware for complex orchestrations
- Document integration flows

---

13. Advanced Integration Topics

- Salesforce as **System of Record vs System of Engagement**
- Multi-org integration strategies
- Master Data Management (MDM) with Salesforce
- Data synchronization challenges (conflict resolution)
- Integration patterns with **Event Bus + Middleware**
- Real-time API performance optimization (cache, throttling)
- Handling rate limits & API quotas (REST 429, Bulk retries)
- Hybrid integrations (mix of APIs, events, middleware)

---

14. Testing & Deployment

- Mocking callouts in Apex tests (`Test.setMock()`)
- Testing Named Credentials & Auth flows
- Test coverage for integration classes
- CI/CD for integration deployments
- Sandbox testing with external systems
- Post-deployment validation & smoke testing

---

15. Interview / Real-World Scenarios

- When to use REST vs SOAP vs Bulk API
- When to choose Platform Events vs CDC vs Streaming API
- Designing an integration for millions of records (LDV)
- Handling retries & duplicate prevention
- Best way to call an external API from Salesforce
- Best way to expose Salesforce data to external systems
- Middleware vs Direct API — when to use what?

Step-by-Step Guide to Start a Career in Salesforce

Step 1: Learn the Basics

- Start with **Salesforce Trailhead** (free learning platform).
- Cover fundamentals: CRM basics, Salesforce navigation, data models.

Step 2: Choose Your Role

- If you like **coding → Developer path**.
- If you prefer **managing systems → Administrator path**.
- If you're good at **communication & business processes → Consultant/Analyst path**.

Step 3: Get Hands-On Practice

- Use **Trailhead Playgrounds** to practice real scenarios.
- Work on mini-projects: building apps, automation flows, and reports.

Step 4: Get Certified

- Start with **Salesforce Administrator Certification (ADM 201)**.
- Later pursue **Developer (PD1)**, **App Builder**, or Consultant certifications.

Step 5: Build a Portfolio

- Contribute to Salesforce **GitHub projects**.
- Create your own Salesforce apps and showcase them.

Step 6: Apply for Jobs & Internships

- Target roles like **Salesforce Admin Intern, Junior Developer, or Support Specialist**.
- Use platforms like **LinkedIn, Indeed, and Salesforce Job Board**.

Step 7: Network with the Salesforce Community

- Join **Salesforce Trailblazer Community**.
- Attend local **Salesforce Saturday meetups** and **Dreamforce events**.

Essential Skills to Learn

- **For Admins**: Flows, Security, Reports, Data Management.
- **For Developers**: Apex, LWC, SOQL, Triggers, Integration.
- **For Consultants/Analysts**: Business Processes, Requirement Gathering.
- **Soft Skills**: Problem-solving, Communication, Teamwork.

Certifications Roadmap

1. Salesforce Certified Administrator
2. Salesforce Platform App Builder
3. Salesforce Platform Developer I & II
4. Salesforce Consultant Certifications (Sales Cloud, Service Cloud, etc.)
5. Salesforce Architect Certifications (for advanced professionals)

Career Growth Opportunities

- **Entry Level**: Salesforce Admin / Developer → ₹4-8 LPA
- **Mid-Level**: Salesforce Consultant / Senior Developer → ₹8-18 LPA
- **Advanced**: Salesforce Architect / Program Manager → ₹20-40+ LPA

Tips for Success

- Be active on **Trailhead** (gamified learning).
- Contribute to Salesforce forums.
- Gain **real-world project experience**.
- Stay updated with **new Salesforce releases (3 per year)**.
- Build a strong **LinkedIn profile** showcasing certifications.

Final Thoughts

A career in Salesforce is **future-proof, rewarding, and diverse**. Whether you're a fresher or an experienced professional, you can enter this ecosystem with the right learning path, certifications, and community involvement.

Remember: Start small, keep learning, get certified, and network with the Salesforce community. Your Salesforce career journey will grow step by step!

Best of Luck

Trusted by 2000+ learners to crack interviews at TCS, Infosys, Wipro, EY, and more.

Want more Real Salesforce Interview Q&As?

- ✓ For Beginners (1–4 Yrs Experience): ☞ https://lnkd.in/gvRbaChu
- ✓ For Intermediate Developers (4–8 Yrs Experience): ☞ https://lnkd.in/gHpeRAqa

For All Job Seekers – 500+ Questions from Top Tech Companies: ☞ https://lnkd.in/gsATfUk3

- ✓ Student Journey – 34 Days to Crack Salesforce Interview: ☞ https://lnkd.in/gJMgzXVY

Mega Interview Packs:

- ✓ 600 Real Q&A (Recruiter Calls) → https://lnkd.in/gFs-CkxT
- ✓ 100 Real-Time Scenarios (Admin + Apex + LWC + Integration) → https://lnkd.in/gYNxZMh4

Career Boosters:

- ✓ Salesforce Project (Sales Cloud) → https://lnkd.in/gmBdJeD9
- ✓ Resume Templates (ATS-Friendly) → https://lnkd.in/gRtHP_Dn

Visit us On ☞

www.trailheadtitanshub.com