Cloud Models

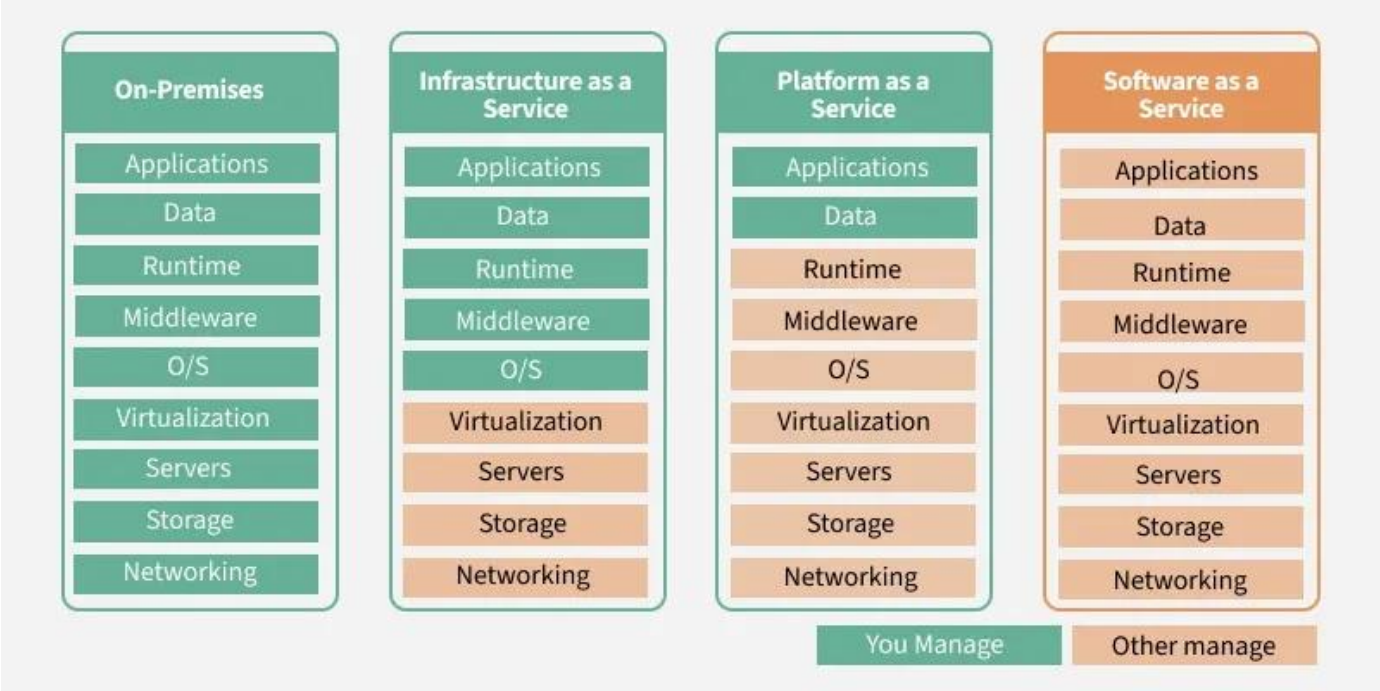
There is three Tyes of Cloud modes. They are IaaS, SaaS, PaaS.

Salesforce org (SaaS)

A typical Salesforce org is an example of SaaS because it's a complete, ready-to-use application that businesses subscribe to for tasks like customer relationship management

Force.com (PaaS)

Salesforce explains that Force.com, now part of its offerings, functions as a PaaS, providing developers with a platform to build and deploy their own applications on top of the Salesforce infrastructure. Salesforce provides both SaaS and PaaS solutions, offering a hybrid model to meet different customer needs.



Salesforce Environments

Salesforce environments are isolated instances used to build, test, and deploy applications.

Production org is used to deploy applications into the cloud.

Developer Environment is used to develop our own custom application as per client/user requirements.

Testing Environment is not possible directly we can use Sandboxes to test the application, In dev free edition sandbox is not available only in production deployment org only we can found sandboxes for testing our application before deploying into the cloud environment.

Types of Sandboxes:

Salesforce has four main types of sandboxes: **Developer**, **Developer Pro**, **Partial Copy**, and **Full Copy**, each designed for different testing and development needs. Developer sandboxes are for basic coding, Developer Pro offer more storage, Partial Copy includes sample production data, and Full Copy sandboxes provide a complete replica of the production environment for comprehensive testing and training

---

Salesforce UI

What is an APP?

App is simply we can say that combination of App Name, Logo and collection of Tabs (objects).

App Name: IMS App

Logo: we can add our own logo and choose the theme colors

Navigation Bar: Different types of Tabs grouped together built by Stand objects and Custom object tabs based on app requirements.

Types of Apps:

Standard Apps: By default, prebuilt by Salesforce will provide.
Example: Sales, Service, Marketing etc.

Custom Apps: Created by Admin/Developer based on specific business requirement needs.

Example: we can decide our own custom objects, tabs and dashboards

Our custom IMS app -> which is used to manage internet service customers and generate bills.

What is Tab?

Tab is a shortcut that is used to access Objects, web pages quickly from Navigation bar in salesforce.

Types of Tabs?
Object Tabs: Standard tabs and custom tabs are object type tabs.
Web Tabs: Where we can embed or access external web pages inside the Salesforce via URL link.

Visualforce Tabs: Used when we want to create custom Visualforce pages and want to access them via a tab.

Lightning Page Tabs: To expose a Lightning Page (App Page, Home Page, Record Page) as a tab.

Lightning Component Tabs: To expose a custom component using LWC as a Tab.

Types of Tabs Visibility

Three Types of Tabs visibility. They are **Default On, Default Off, Hidden**. We can control tab visibility with these types.

Default On

- Tab is **visible by default** in the App navigation bar.

Default Off

- Tab is **hidden by default**, but we can add it manually to our App.
- Example: A **Billing tab** that only some users might need to see.

Tab Hidden

- Users **cannot see or access** this tab at all.

Objects in Salesforce

Standard Objects

- These are predefined by Salesforce we can use these everywhere.
- These cover common business processes like Sales, Service, and Marketing.

- We cannot delete them, but we can customize them by adding fields, page layouts, validation rules, etc.

Eg: Account, Contacts, Leads, Opportunities, Case, Campaign, Product and Precook etc.

Custom Objects:

- These are created by us (Admin/Developer) to fit client/our company's unique business needs. We can see every custom object has a suffix '__c '.
- They store data that standard objects don't cover. We can add our own Custom fields, Relationships (lookup/master-detail), Page layouts, record types etc.

Record Name

When we create a custom object, Salesforce asks us to define the Record Name field.
This is the primary identifier of each record (like the "title" of that record).

- **Default Field** → Name/AutoNumber (cannot be deleted).
- There are of two types. We can create **record names like with text** or **AutoNumber** field to generate **unique number automatically** when a new record gets created by the user.

Track Field History:

- Helps us to track changes made to fields on standard or custom objects.
    - We can select up to 20 fields per object to track.
    - Salesforce automatically tracks the field's history old value, new value, changed by, and date. We can see this object's History in object record related lists.
    - If we want to track our custom object fields also, we need to enable track field history and we can select which fields should be tracks on our own.
    - We can track fields up to 600 on our object.

Deployment Models (Custom Objects)

When we tick Deployed while creating Custom object it is available for all users based on profiles. Another way if we select in development which means it is incomplete or useful for testing. It is not visible to everyone except admin.

Layouts in Salesforce:

Layouts control how data is shown and interacted with by users. There are different types of layouts. They are Page layout, Compact Layout, Mini Page layout, Search Layout, Record Types with Page Layouts.

- **Page Layout:** Full details form (different for HR/Interviewer).
- **Compact Layout:** Top summary of an object (Name, Status, Date).
- **Mini Layout:** Hover preview (Email, Phone).
- **Search Layout:** Controls what appears in search/lookups.
- **Record Type + Page Layout:** Different page layouts for Fresher vs Experienced.

Page Layout:

Controls how fields, sections, related lists, and buttons appear on a record detail page.

Eg: There are two different users like HR and Interviewer, different users see only what they need. For the

HR profile -> Page Layout shows **Applicant Name, Job Role, Resume Upload, Status**.
Interviewer profile -> Page Layout hides **Salary Info** and only shows **Candidate Skills** etc.

Compact Layout:

Decides which fields appear in the **record highlights panel** (top of the record) and in the Salesforce mobile app. When user opens the record on mobile, they quickly see key info without opening the full page.

Mini Page Layout

Appears in **hover details** when you hover over a lookup field. Saves time by showing quick info without opening the full record.

Search Layout

Decides what fields appear in search results, lookup dialogs, and list views.

- In Global Search → Results show **Applicant Name, Job Role, Status**.
- In Lookup Search → When another object looks up Job Application, users see **Applicant Name + Applied Date** instead of irrelevant default fields.

Record Types with Page Layouts

Now we will see different **page layouts and picklist values** depending on the business scenario.

Eg:

- **Record Type 1:**
  Intern Application -> Page Layout shows University, Graduation Year, Internship Duration.
- **Record Type 2:**
  Full-time Application -> Page Layout shows Previous Employer, Salary Expectation, Notice Period.
- HR can pick record type while creating a record, and fields/layout change accordingly.

Relationships in Salesforce

A **relationship** in Salesforce means **linking two objects together** so that data is connected.

- It's like a **foreign key** in databases.
- Example: A **Contact** belongs to an **Account**.

Types of Relationships:

Lookup Relationship

- A loose relationship between two objects.
- Child record can exist **without a parent**.
- You can link one record to another, but they are independent.
  Eg: Worker__c (Custom Object) → Lookup to **Account** (Contractor).
  A worker may or may not be linked to a contractor.

Master-Detail Relationship

- A strong relationship where the **child record cannot exist without the parent**.
- If parent is deleted → child is also deleted (cascade delete).
- Child record **inherits sharing and security** from the parent.

Eg: **Payment__c (Child)** → Master-Detail to **Worker__c (Parent)**.
If Worker is deleted → all related Payments are deleted.

Many to Many Relationship (Junction Point)

- Allows a record of one object to be related to multiple records of another object.
- Achieved by creating a **Junction Object** with two master-detail relationships.

Note: Which side we choose first Master-Detail relationship that should be owner means primary relationship and other one will be considered as secondary relationship.

Eg: **Worker__c ↔ Project__c**

- One worker can work on multiple projects.
- One project can have multiple workers.

Hierarchal Relationship (only for user object)

It is a self-relationship in the salesforce i.e. a lookup relationship to the same object, creating a link between records of the same object type. This enables you to establish hierarchical structures, like a manager-employee hierarchy, or to relate records within the same object

- Special relationships are available only for the **User** object, but we can create on our own also based on our requirements.
- we define one user reporting to another user.
  Eg: **User A (Developer)** → Reports to **User B (Project Manager)**.

Summary:
Real-life IT based Example:

- **Lookup** = Employee ↔ Laptop (Employee may or may not have a laptop).
- **Master-Detail** = Bug ↔ Project (A Bug must belong to a Project).
- **Many-to-Many** = Developer ↔ Project (A developer can work on many projects, and a project can have many developers).
- **Hierarchical** = QA Tester reports to QA Lead.

Note:

- Relationships should select from many sides object always
- We can create Cross-Object fields and Lookup filters on both relationship sides. But **Rollup summary** fields are possible only in **Master detail relationships**.

- For master-detail relationship we cannot create global actions on childside. But we can create object specific actions.

Record Types:

Note: We have seen previously that every profile can access only a single page layout at once. To overcome this problem, we can use Record Types.

Record Type mainly controls the Page layout & Picklist values while creating a particular record type.

Steps: First, we need to make objects slicing means we need to create different page layouts for a specific object with different fields and picklist values based on requirement for specific record creation.

Eg: In a Training Institute create records based on the type of student. Like fresher, graduated, experienced. Based on selected category type has different fields and some common fields, and we can control picklist value for each specific record type of page layout.

Problem: For one profile has only one Page layout but we need multiple page layouts for created diff categories and, we know that one user has only one profile.

Solution: With different Record Types we can create multiple Page layout on a single profile, but one profile has many users.

Record types will have multiple page layouts based on multiple profiles. We can provide restrict access to created record of certain record type.
Note: Record Type will only allow control the creation. Even if the profile has not accessed the user of the record type will be able to see, modify other records created using that record type.

Summary:

- **What?** Record type we can control (Page layouts and Picklist values)
- We can assign diff page layouts on the same record type.
- Assign same page layout on diff record type
- Providing restricts access to control only on creation.
- **Why?** Single profile cannot have multiple page layouts. We can do this with record type.

Data Management (Import + Export)

In Our salesforce org Data management simply means manage data by Import and Exporting data.

Data Import:

Operations:

1. Insert – Inserting a new record into salesforce object
2. Update – Updating an existing record by using Record ID or External ID
3. Upsert – Both (Insert + update) at a time.

This is possible for with file format ' .CSV ' (Comma Separated Value).

It is similar to mapping in Java <key: value> pair mapping. We can import data by performing Column Mapping while importing our external data into salesforce org.

There are 2 standard tools for importing data into salesforce org. They are:

a. **Data import Wizard** – with this we can import up to 50,000 records only. We can perform above all operations with this standard tool.
b. **Data Loader** – It is a client command line application. With this we can import up to 5 million records at a time.

Data Import Wizard:

It S**upports for only few** standard objects like Accounts, contacts, leads, solution, campaign members and for custom objects too.

Data Loader: (Import + Export & Delete records)

First, we need to download and use this application in our system. This is a client command line type application.

- We can import up to 5 million records at a time.
- Suports all the objects
- We can perform operations like import, export and delete records.
- We need to do mapping to perform these operations.

- It works on command line.

Note: the client-based Salesforce Data Loader is not included in the Professional Edition because it requires API access, which is limited in that edition.

Eg: If we want to insert 2000 records in Products standard object it is not possible with Data import wizard. Because only a few standards are allowing to import with data import wizard. In such case we can use Data Loader to import records into salesforce org.

Important Points:

While importing few things should remember. They are

1. Picklists are based on checkbox (restrict access value from the value set) while create a data type field. If it is checked, we can import otherwise data import fails.
2. Checkbox (true/false) - some time fild dependcy will depend on checbox type.
3. Multiselect Pick list (we should check that every value is separated by semicolon(;) after each value.
4. Formula Fields & Rollup Summary fields are read-only. So, no need to map or import)
5. Must follow validation rules of that object fields
6. Required fields
    a. Global level Required filed (while creating data type field that time only we need to check)
    b. Page layout Required option not mandatory.
7. Order (Lookup/master)
    a. Eg: Student should exist to import Ratings

Data Export:

- We can export all object recors data (Full data with all field records)
- We can use it weekly once to export from org
- So, we can use some other external tools for export if required like Zitterbit, Dataloader.io etc.

Data Loader:

Export: Using data loader we can export full data (or) Specific only few limited records from an org.

Export All:

The main difference between export and export all is with export we can export available records on object but with export all we can export all the available records on objects and Deleted (Bin) records will also export with the specific object records.

There are two ID's we can use to import the data. They are unique and safe way to import data and we can perform all operations without any loss of data while importing. They are:

a. Record ID – 15-digit Unique Case sensitive ID
b. External ID – 18-digit Unique case insensitive ID

Record ID:

- It is a 15–digit unique ID – case sensitive generated inside salesforce org automatically for each new record.
- Mostly user in API for development persepective

Note: First 3 char defines Object id. Eg: Account – 001, contact – 003 etc.

External ID:

This is case insensitive id. It has 18-digits. The first 15-digits are same as Record ID, but last 3 digits is the checksum of the capitalization of first 15-digit of record ID.

- It is Custom field
- We can use it identity record uniquely in the external system while we are importing in the salesforce org.
- Without record ID we can use external ID field to UPSERT or UPDATE record by importing
- Salesforce supports a maximum of 25 External ID fields per object, which are custom text, number, or email fields used to integrate with external systems, although this limit can be increased by contacting Salesforce Support.

Steps to Convert 16-dgit to 18-digit:

1. First, we need to divide 15 digits into 3 equal parts.
2. Reverse those divided digits order
3. If there is any (A-Z) capitalize letter found replace with 1 or else replace with 0
4. After that we need to convert the digits into Binary. We will get three Characters that 3 should be added at the end of the 15-digit and make 18-digit case insensitive

Note:

- We can use Functional Method to convert 15-digit to 18-digit by using CASESAAFEID(ID). Inside that method we need to pass a 15-digit record id it will convert to an 18-digit id.
- Only for these fields only external fields valid
    - Text
    - Number
    - Email
    - AutoNumber
    - Date is based on Profile time zone settings.

Now we will see some important topics -> Salesforce Administration point of view

Basic Example:

- **License** = Which floor of the building you're allowed in.
- **Profile** = Which rooms you can enter on that floor.
- **Role** = Which people's files you're allowed to see.
- **Permission Set** = A temporary visitor passes for extra access.

Salesforce Point-of-view:

- **User** -> **Individual** login account.
- **User License** -> Decides **what base Salesforce features** the user can access.
- **Profile** -> Decides **what user can do** with objects, fields, apps.
- **Standard Profiles** -> Predefined sets of permissions (System Admin, Standard User, Read-Only, Marketing User, etc.).
- **Permission Set = Extra access** you give to a user **on top of their Profile** (without cloning or changing the profile).

User

- A **User** - Anyone who logs into Salesforce.
- Each user has:
    - **Username** (unique, looks like an email)
    - **Email & Profile**
    - **Role** (optional, used for data visibility in hierarchy)
    - **User License** (decides what kind of Salesforce access they get)
    - **Profile** (decides what they can do inside Salesforce)

Eg: Rakesh is a Salesforce User with a Sales Cloud License.

User Licenses

The **User License** tells which **features and objects** the user can access. It's like a **base permission**.

- Profiles are built on top of licenses.

Common Types:

1. **Salesforce** (full CRM license is used to access to standard + custom apps)

2. **Salesforce Platform** (cheaper, but no access to CRM objects like Leads, Opportunities; can use custom apps/objects)
3. **Chatter Free / External** (only collaboration, no records access)
4. **Community/Experience Cloud** (for external customers/partners) etc.

Profile

- A **Profile** = Defines what a user can **see** and **do** in Salesforce.

Controls:

- Object permissions (Create, Read, Edit, Delete)
- Field-level permissions
- Tab visibility
- Record types & page layouts access
- App access

Standard Profiles

Salesforce gives some **prebuilt (standard) profiles**. These can't be deleted, but we can clone them to make custom profiles.

1. **System Administrator**
    a. Full access to everything.
    b. Can create/edit users, apps, fields, objects.
2. **Standard User -** Can create/edit their own records, run reports, use most apps.
3. **Read Only -** Can only view records. No edits.
4. **Marketing User -** Special permission for managing campaigns and leads.
5. **Contract Manager -** Can manage contracts but limited access elsewhere.
6. **Solution Manager -** Can manage Salesforce Knowledge articles/solutions.
7. **Self-Service Portal User / Customer Community User -** For external users (customers/partners).

Eg:
In an IT company builds an **App in Salesforce**.

- **Users**:
    - Developers, Testers, Project Managers, Clients

- **User Licenses**:
  - Developers → Salesforce Platform License
  - Managers → Salesforce License (need reports, dashboards, opportunities)
  - Clients → Community License
- **Profiles**:
  - Developers → Custom Profile cloned from **Standard User** (CRUD on Project__c, Bug__c)
  - Testers → Custom Profile (CRUD on Bug__c only)
  - Project Manager → System Administrator (full access)
  - Clients → Read-Only Profile (can see their project status but cannot edit)

Roles (Data Visibility)

- **Role = Who sees what data** in the org (controls record-level sharing via **Role Hierarchy**).
- **Higher** roles **see** the data of roles **below** them.
- Think of it as an **organization chart for data visibility**.

Example (IT Company Role Hierarchy):

- CEO
  - Project Manager (Anil)
    - Developers (Ramesh)
    - Testers (Priya)

Hierarchy Data visibility Flow:

- Ramesh (Developer) -> sees only his Project records.
- Priya (Tester) -> sees only her Bug records.
- Anil (Manager) -> sees all records of Ramesh + Priya (Because he is higher in role hierarchy).

Permission Sets

- **Permission Set = Extra access** you give to a user **on top of their Profile** (without cloning or changing the profile).
- Flexible way to assign special permissions to individuals or groups.

Example:

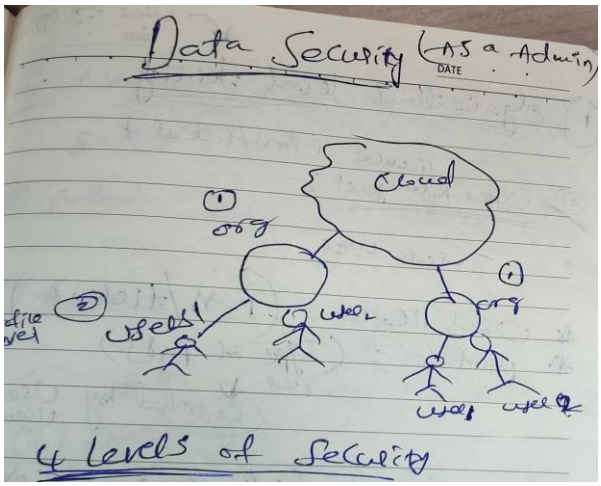
- Ramesh (Developer Profile) -> normally cannot delete Records, Bugs.
- But for certain period time, he needs **Delete permission** -> Assign a **Permission Set: "Bug Delete"**.
- Priya (Tester) -> normally can't export reports -> Assign a **Permission Set: "Report Exporter"**. Giving extra access on top of their profile.

Real Life Example based on IT company based:

- **User**: Ramesh (Developer)
  - **License**: Salesforce Platform
  - **Profile**: Developer Profile (CRUD on Projects, Bugs)
  - **Role**: Developer Role (sees only his records)
  - **Permission Set**: "Bug Delete" (temporary extra permission)
- **User**: Priya (Tester)
  - **License**: Salesforce Platform
  - **Profile**: Tester Profile (CRUD on Bugs only)
  - **Role**: Tester Role (sees only her Bugs)
  - **Permission Set**: "Report Exporter"
- **User**: Anil (Manager)
  - **License**: Salesforce
  - **Profile**: System Admin (full access)
  - **Role**: Manager Role (sees all records below him: Developers + Testers)
  - **Permission Set**: Not needed (already full access)

Flow: User logs in -> License defines features -> **Profile** defines permissions -> **Role** defines data visibility -> **Permission Set** adds extra rights.

Data Security



Data security is important because we need to control what a user or group of users can see in the org or app. We can easily assign different sets to different groups of users. We can control access by the data security levels to our whole org and some data level security.

Salesforce provides a comprehensive and flexible data security model to secure data at different levels. Salesforce also provides sharing tools to open and allow secure access to data based on business needs.

There are four levels of security in the sales force. They are:

1. Organizational Level Security
2. Object Level Security (CRUD)
3. Field Level Security
4. Record Level Security

Organizational Level Security is used to secure the Org before securing the users data inside the org. The remaining three keys related to data in Salesforce: **objects, fields,** and **records. Objects** are similar to tables in databases. **Fields** are like columns of the table. **Records** are similar to rows of data inside the table. Salesforce uses object-level, field-level, and record-level security to secure access to object, field, and individual records.

1. Organization Level Security

To secure org we must cover some security steps:

a. User Management (License) - pass / ticket to enter to an event
b. Managing Password Policies
c. Restrict Access by IP Address or Location
    i. Static IP – Static IP restrictions are typically set for trusted office networks or specific locations with fixed IP addresses.
    ii. Dynamic IP - For dynamic environments, advanced solutions may use geolocation APIs, custom logic, or third-party tools to determine location or country, require multi-factor authentication (MFA) for unknown networks.
    - Org Level – can login outside with challenge like OTP.
    - Profile Level – cannot login outside org range.
d. Restrict Login Access by Time
    a. Eg: Stock Market
e. Health Check for Security Status

a. User Management:

- Based on type of User License (Pass/ticket)
- Profile – controlled by user licensee (type of pass)

Note: We cannot delete once the user account is created in the org. We can freeze the user, why? Because the user did some work that data should be dependent on the user account should not change for future safety. That's why the salesforce does not provide the option to delete users from org.

Salesforce given alternate options like **inactive or freeze. Inactive** means we can assign his user license to others whereas **freeze** is alternate for delete we cannot give his access to other users like inactive user type.

b. Managing Password Policies

Salesforce give access to Password policies we can set our own different type of password policies from the given options and make org very secure with high standard security which will apply to

the entire org. We can also give profile level password policies which means all users inside the profiles applied to that password level policies for high security at profile level.

c. Restrict Access by IP Address or Location

- We can restrict the org outside of IP range. Static IP restrictions are typically set for trusted office networks or specific locations with fixed IP addresses.
- Salesforce allows administrators to configure trusted IP ranges at both the **organization and profile levels.**
- At the organization level, trusted IP ranges let users log in without identity verification (like email/mobile OTP); outside these ranges, identity verification is required but login is still possible.
- At the profile level, access is strictly denied for users outside the set IP range—no verification or OTP; the login fails outright.
- This method is effective for organizations with fixed office locations and consistent network addresses.

d. Restrict Login Access by Time

- User can login based on Login hour range set by the profile
- It is possible only in Profile Level
- Eg: Stock Market

e. Health Check

Inside the Org if we go to setup and search Health Check and check our org level security status and we can see the percentage of Security Level.

2. Object/Profile Level Security

Defn: Simply defined as collection of settings and Permission which defines what the user will be able to see or do.

In the above our org is secured but still inside the org so many users don't have restrictions on data to access or modify, to secure data on object level we can restrict data access on object level using object level security.

- We can control object level permission for both standard and custom objects.
- We can set permissions for a particular object.
- We can give permission to view, edit and delete any records of that object.

- We can control object permissions using profiles and permission sets.

a. Profiles

Profile level security is simply defined as collection of settings and Permission which defines what the user will be able to see or do.

- Object Level access
- Feild Level security (preform actions like see, edit)
- Record Type security
- Tab access

A. Standard Profiles
B. Custom Profiles

A. **Standard profiles:** In an org salesforce by default few standard profiles are provided.
   Eg: Stander User – some limited base level access
   Solution Manager – Read Only user
   Marketing Manager – to create campaigns and manage leads
   Contact Manger – manage contracts of an org
   System Admin – Full access to the entire org.

Note: We cannot change the settings and permissions of standard profiles. Salesforce doesn't allow us to do that. We cannot delete the standard profiles.

B. Custom Profiles
   Standard profiles all permissions from standard user along with some additional features. We can change the settings of custom profiles.

b. Permission Sets (additional access like delete or edit)

Earlier, each user could have only one profile, which defined all of their permissions. If a user needed additional access beyond their profile, we had to create another profile tied to the same license. However, a user can only have one profile at a time, so switching to a new profile meant losing access from the previous one. This required manual changes, frequent profile creation, and became difficult to manage.

To overcome this limitation, Salesforce introduced **Permission Sets**, which provide additional access on top of a user's current profile. Instead of creating multiple profiles, we can simply assign Permission Sets to users as needed, making access management much more flexible and easier to maintain.

Permission set cannot restrict the access of provided by profiles. On top of that by using a permission set we can give additional access to users.
When ever need to do multiple users works on the same profile as taking another extra work or extra access like edit or delete.

Permission sets as the primary way to assign object and field permissions. Beacause, they are more flexible and easier to manage.

Note: Profile ----<-Users ->---<-- Permission sets (1 to many and Many to Many)
For single profile multiple users possible for single user we can gave many permissions sets similarly single permission can assigned to multiple users allowed.

A many-to-many relationship exists between Users and Permission Sets. This relationship is managed through a junction object called **Permission Set Assignment**, which defines which Permission Sets are assigned to which Users.

- **Users >--< Permission Sets** => Many-to-Many
    - One User can have multiple Permission Sets.
    - One Permission Set can be assigned to multiple Users.
- **Junction Object** = **Permission Set Assignment**
    - It links User and Permission Set.

Note: On same profile but different works assigned with additional access for some temporary duration using permission sets.

Difference between Profile and Permission set

Profile	Permission Set
Collection of settings & Permissions what to see or edit by the user	Same as Profile and top of that we get additional access
Mandatory	**Not** Mandatory
Restrictive Access	Additional Access
1 User has 1 profile	1 user has Mutiple permission sets

3. Feild Level Security

Now we have object level security access with some restrictions but still the user can be able to see all fields and can modify fields of those records of those objects.

Field-Level Security controls whether a user can **see, edit, or not access a specific field** on an object.

Even if a user has access to an object (like **Account**), you may not want them to see or edit all fields.

We can set Field-Level Security in:

1. **Profiles** – controls field access for a group of users.
2. **Permission Sets** – grant additional field access to specific users.

We can set security like which can be access see, edit or Hide

- **Visible** -> User can see the field.
- **Read Only** -. Users can see but cannot edit the field.
- **Hidden (not visible)** -> Field is completely invisible to the user.

We can hide unnecessary fields that are irrelevant to the particular profile before assign or giving additional access with permission sets.

Access: Setup ----> Field Accessibility

4. Record Level Security

We can restrict access to records for users, even if the user has object level access.

Inside the objects `n` no.of Records we can see. After getting access to an object even though there is some security level provided on object-level and field level on top of profiles. But still users have access to each record that is present inside that object.

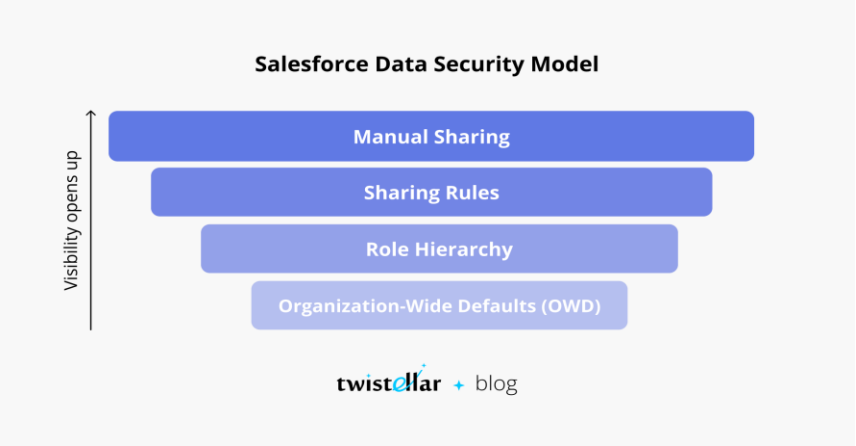
We can restrict accessing CRUD on the record. We can control access in 3 ways.

- Public Read Only
- Public Read/Write
- Private

Eg: We can see our own records but not others.

We can manage Record level access in the following ways. They are:

a. OWD (Organization wide Default) - default set an organization will be applicable to entire org
b. Role Hierarchy – top level users can see below of the user data and manage but not vice versa
c. Sharing Rule – Sharing automatic exception
d. Manual Sharing – Sharing Manually



As an admin he/she can start by configuring org-wide defaults, to lock down your data to the most restrictive level as Private. Then you use the other record-level security tools to grant additional access to selected users when needed.

a. **OWD:**
   - It specifies the default level of access of records.
   - org-wide defaults, to lock down your data to the most restrictive level as Private.
   - We have 3 access levels. They are:
     1. **Private** – no one can see except the admin (More Priority). Users can only see their own records.
     2. **Public Read/Write** - all users can see record and modify records (Least Priority)
     3. **Public Read Only** – can only see but don't have access to edit

   Note: Access depends on parent object.

b. Role Hierarchy

- Users higher in the hierarchy automatically inherit access to records owned by users below them.
- Eg: A Manager can see all records of their team members.

In an organization, different job roles have different record access requirements. Normally **job roles** are sorted in a **hierarchical way**: users with a **higher role** should have **access** to records to which users in **lower roles** have access. Note that a role hierarchy rarely maps to an org chart. It really maps hierarchies of data access. **Salesforce provides an easy way to share records with managers and represent a role hierarchy**. To use this sharing rule, an admin must first add the user to a role and grant access.

c. Sharing Rule

- You can share records with:
    - Roles
    - Public Groups
    - Specific users
- Eg: Share **all Projects in Bangalore** with the **Bangalore Sales Team** role.
  Note: Criteria-based sharing rules let users access records based on the value of a field in a record, irrespective of who owns the record.

d. Manual Sharing:

Record manually shares a record with a specific user.

Eg: Sharing a record or code with another developer manually to collaborate and discuss.

Record-Level Security will Controls which records a user can **see, edit, or delete** in Salesforce.