Lecture 4 – Salesforce Automation, Apex, LWC, Digital Experience Cloud

Introduction to Salesforce

Richard Whitney – Software Engineer

Jason Finnegan – Senior Software Engineer

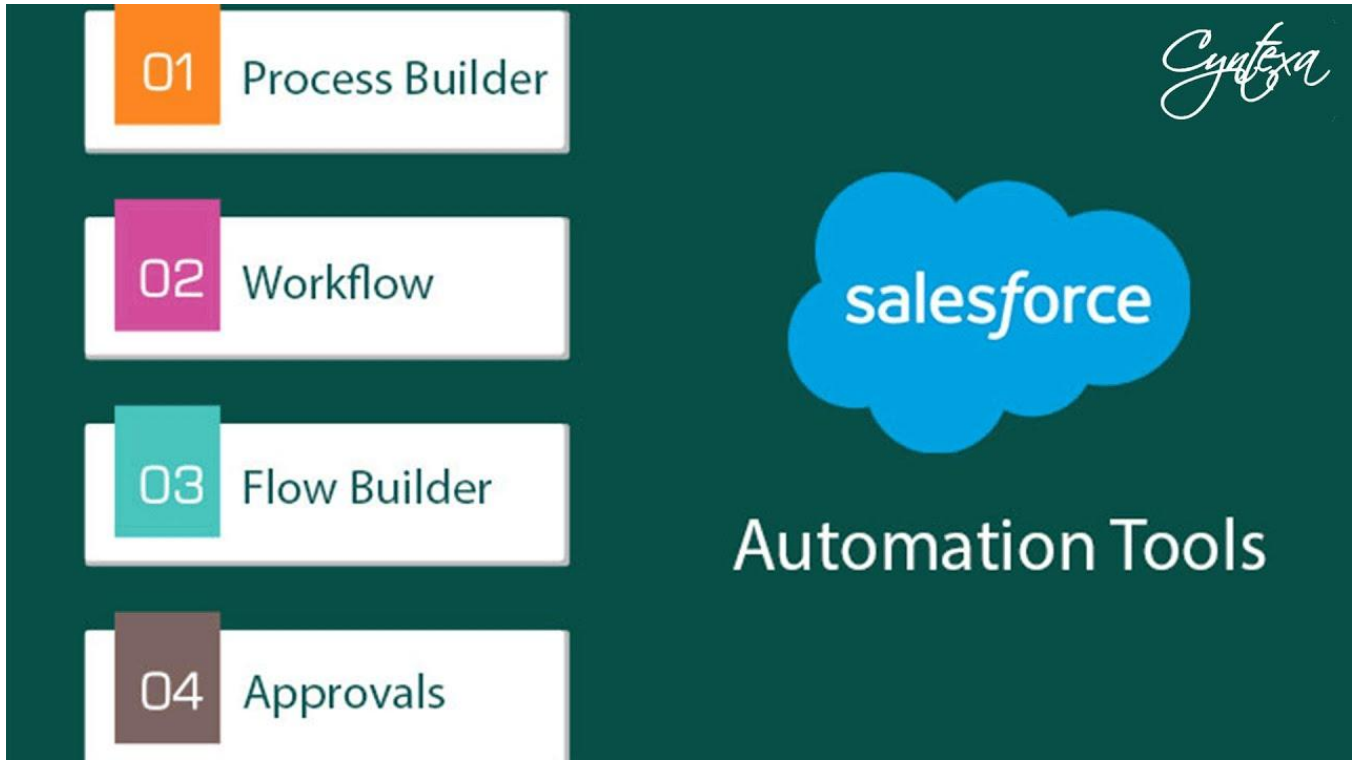
Introduction

- Salesforce Automation Tools
- Apex
- Lightning Web Components
- Digital Experience Cloud
- Data 360
- Salesforce AppExhange

- Lab Session:
  - Learn the basics of Flow and Build your own,
  - Apex and Apex Testing.
  - Creating your own Knowledge articles and Site on Experience Cloud

Salesforce Automation Tools

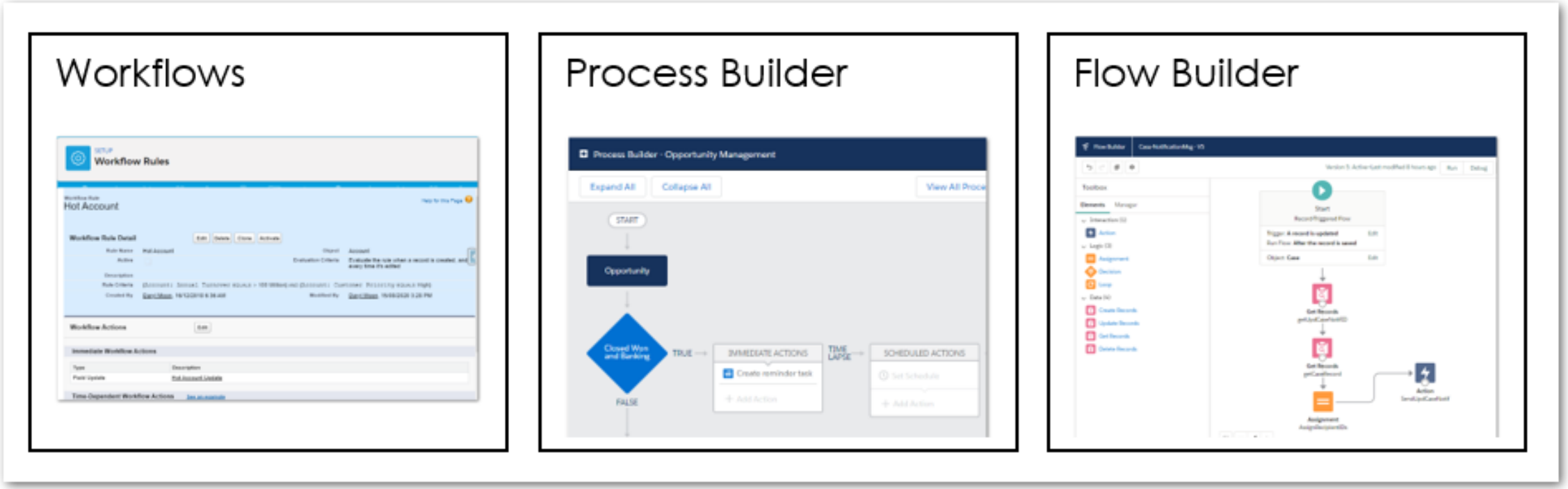
Salesforce Automation Tools



- Approval Processes
- Workflow Rules
- Process Builders
- Flows
  - What can Flows Automate
  - Flow Types

Automation Tools

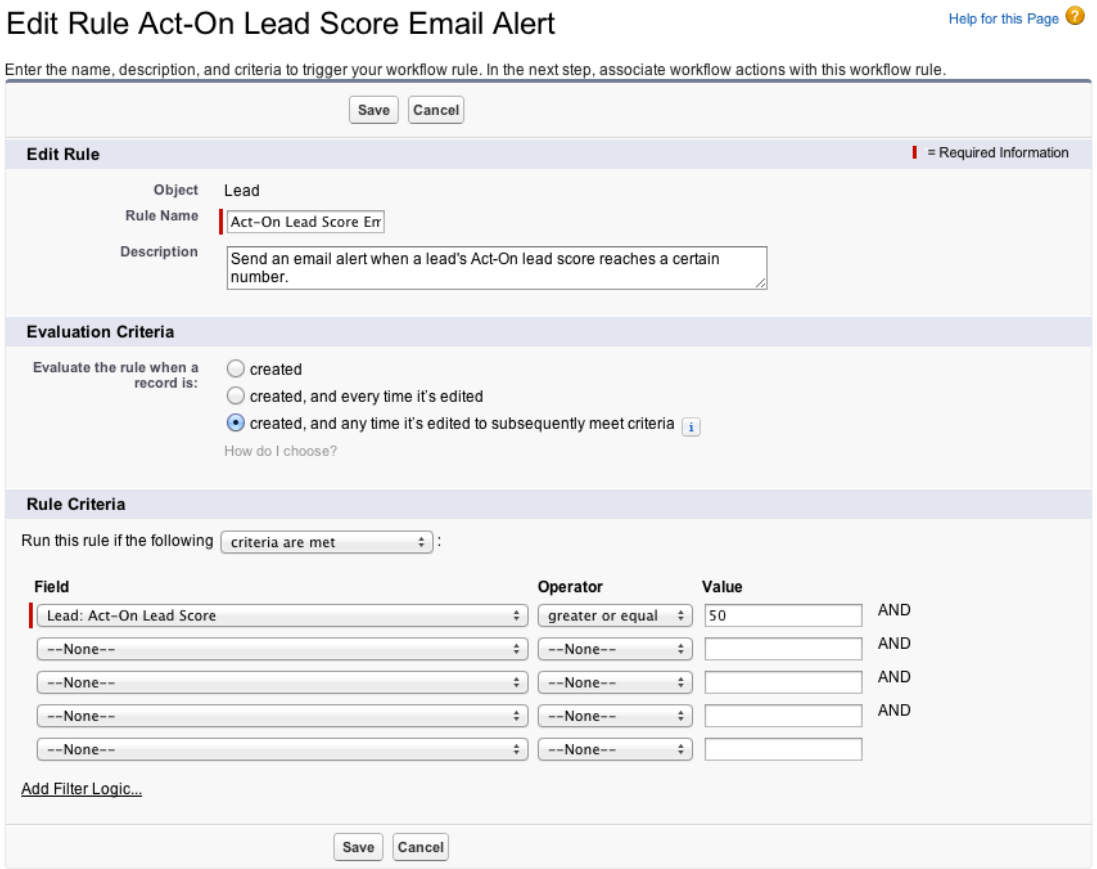
- Workflows & Process Builders were the predecessors to Flows & are retired
- From October 2022 creation of new workflow rules was disabled
- From May 2023 creation of new Process Builders was disabled



Workflows

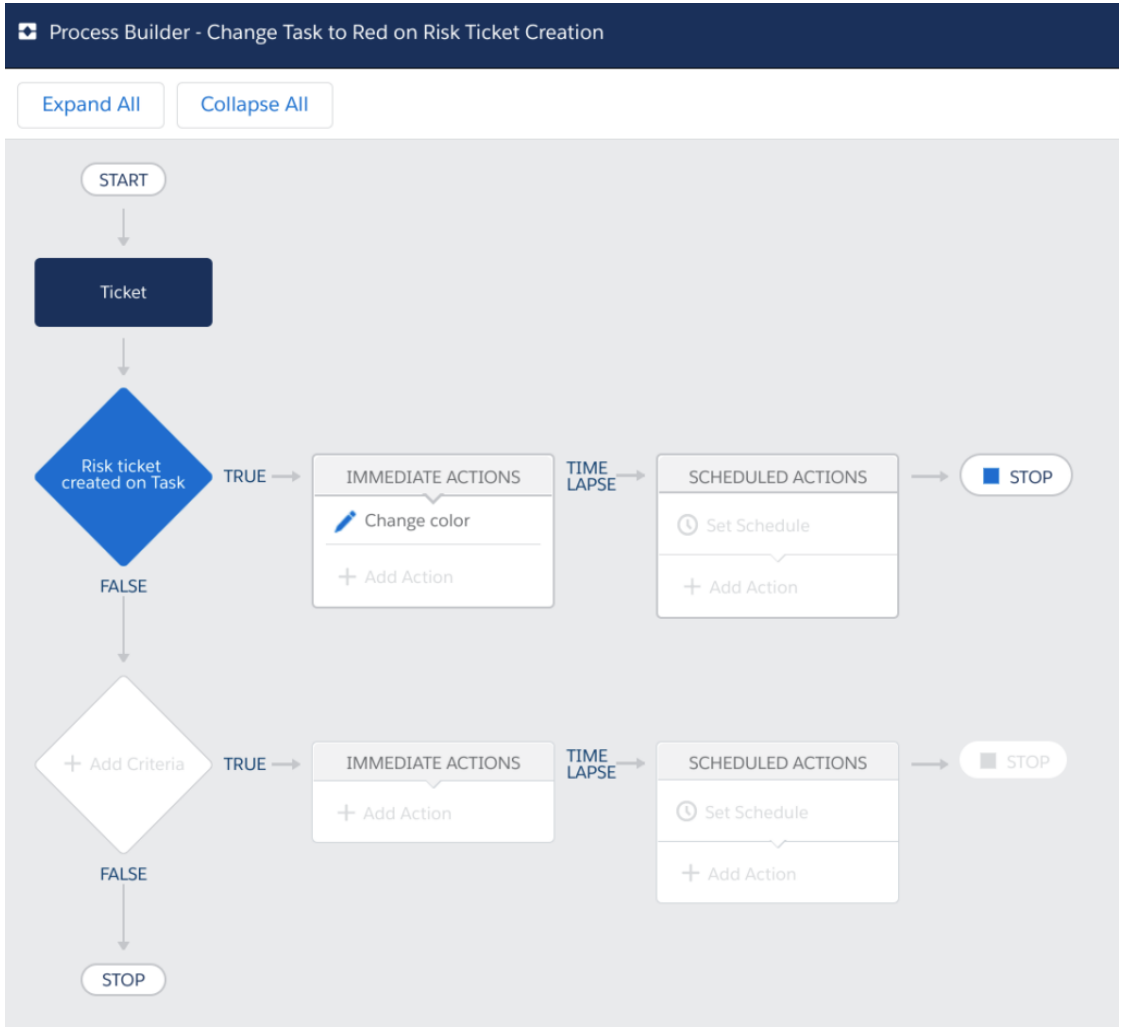
- Main container for a set of workflow instructions. These instructions can always be summed up in an if/then statement.

- Two main components.
  - Criteria: the “if” part of the “if/then” statement.
    - Must be true of the record for the workflow rule to execute the associated actions.
  - Actions: the “then” part of the “if/then” statement.
    - What to do when the record meets the criteria.



Process Builder

- Supports three types that determines what triggers the process.
  - A record change process starts when a record is created or updated.
  - An event process starts when a platform event message is received.
  - An invocable process starts when something else, like another process, invokes it.
- Each process consists of:
  - Criteria that determine when to execute an action group.
  - Action groups, which consist of immediate or scheduled actions. Only record change processes support scheduled actions.

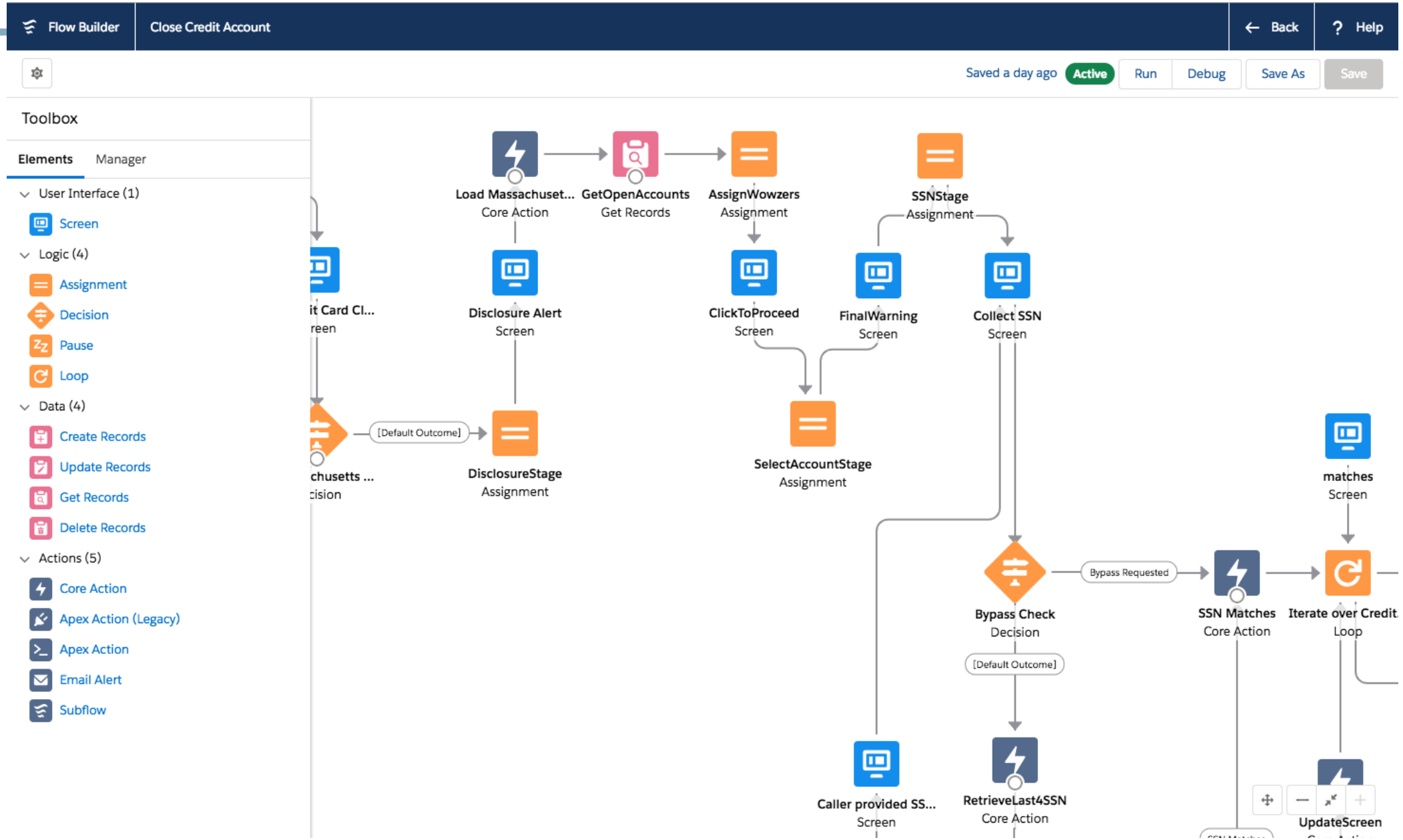


Flows

- Flow is more complex and powerful, allowing for more advanced and customized automation processes using flowchart-like logic

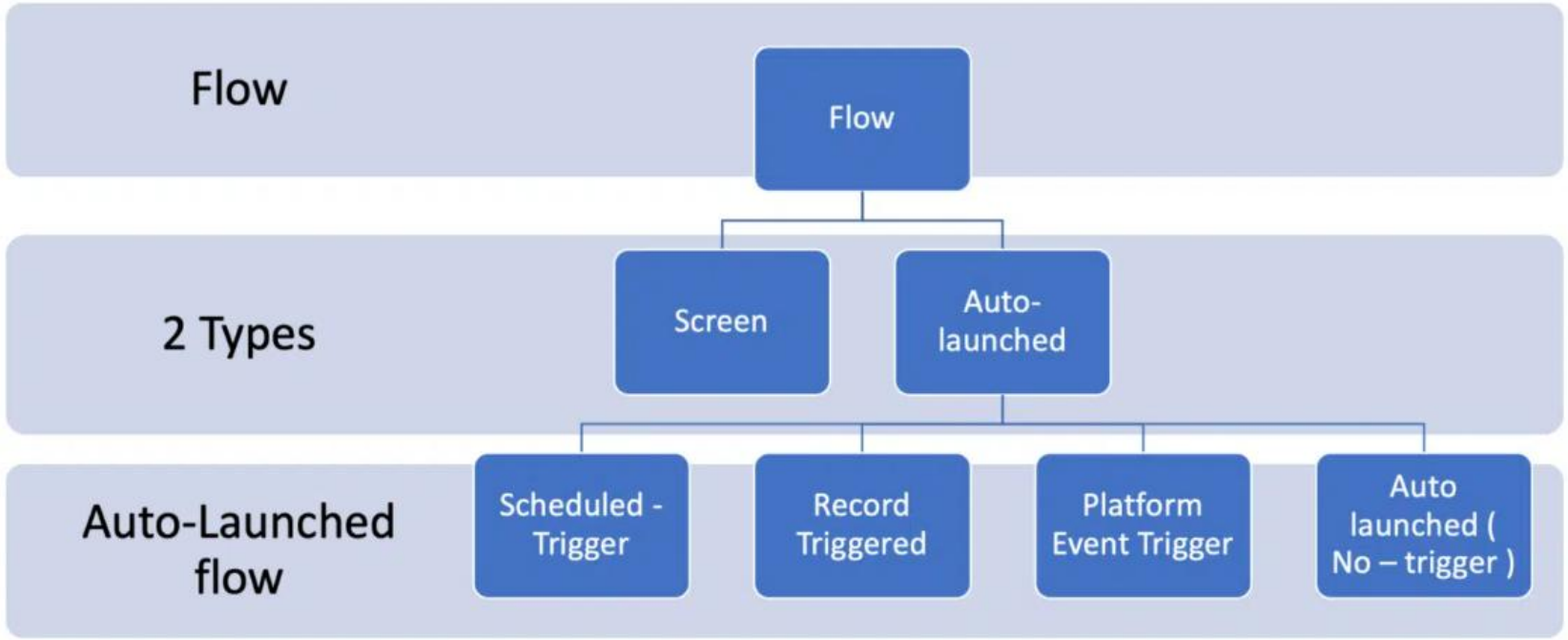
- Flows can:

  - Loop over records
  - Display screens for user input
  - Evaluate conditions
  - Create, Update & Delete records
  - Send an email
  - Collect input from Internal & external users
  - Send a custom notification
  - Send a survey
  - Submit a record for approval
  - Run another flow in the context of the current flow
  - Send outbound messages


Flow Builder
Close Credit Account
Back
Help
Saved a day ago
Active
Run
Debug
Save As
Save
Toolbox
Elements
Manager
User Interface (1)
Screen
Logic (4)
Assignment
Decision
Pause
Loop
Data (4)
Create Records
Update Records
Get Records
Delete Records
Actions (5)
Core Action
Apex Action (Legacy)
Apex Action
Email Alert
Subflow
Load Massachuset...
Core Action
GetOpenAccounts
Get Records
AssignWowzers
Assignment
SSNStage
Assignment
Disclosure Alert
Screen
ClickToProceed
Screen
FinalWarning
Screen
Collect SSN
Screen
[Default Outcome]
DisclosureStage
Assignment
SelectAccountStage
Assignment
matches
Screen
Bypass Requested
Bypass Check
Decision
SSN Matches
Core Action
Iterate over Credit
Loop
[Default Outcome]
Caller provided SS...
Screen
RetrieveLast4SSN
Core Action
UpdateScreen

Flow Types

- Auto launched Flow with a Record Trigger
- Auto launched Flow with a Schedule Trigger
- Auto launched Flow with a Platform Event Trigger
- Auto launched Flow with No Trigger – executed from other flows
- Screen Flow



Flow Types

Auto launched Flow No Trigger

- Doesn't require user interaction.
- Doesn't support screens, local actions or choices
  - e.g. user clicks button to start background automations, such as updating related records or sending emails

Screen Flow

- Requires user interaction because it includes screens, local actions, steps, choices, or dynamic choices.
- Good for wizard like flows where we need to collect information from the user and take actions based on the input.

Auto launched Flow with a Platform Trigger (Platform-Trigger Flow)

- Process that starts when a particular platform event message is received
- Good for decoupling functionality.

Flow Types

Auto launched Flow with a Record Trigger (Record-Trigger Flow)

- Replacement / No Code version of Triggers
- Only these elements are supported: Assignment, Decision, Get Records, and Loop.
- Can trigger when
  - Record is created, updated, delete (Before or After DML Operation)
  - After a certain amount of time (Add a scheduled path)
    - e.g. x time after a record is created do something

Auto launched Flow with a Schedule Trigger (Schedule-Triggered Flow)

- Replacement/No Code version of Scheduled Apex
- Runs only from a schedule.
  - e.g., Every Monday send an email to x people

- This flow type doesn't support user interaction, screens, local actions, choices, or choice sets.

Apex code

Apex

- What is Apex?
- When Should I Use Apex?
- How Does Apex Work?
- Apex Class Definition
- Invoking Apex
- Unit Tests

What is Apex?

- Apex is a **strongly typed**, **object-oriented** programming language that allows developers to execute flow and **transaction control** statements on Salesforce servers in conjunction with calls to the API.
- Using syntax that looks like Java and acts like database stored procedures, Apex enables developers to add business logic to most system events, including button clicks, related record updates, and Visualforce pages.
- Apex code can be initiated by Web service requests and from triggers on objects.

What is Apex?

- **Integrated -** Built-in support for common Lightning Platform idioms, including:
  - Data manipulation language (DML) calls, such as INSERT, UPDATE, and DELETE, that include built-in DmlException handling
  - Inline Salesforce Object Query Language (SOQL) and Salesforce Object Search Language (SOSL) queries that return lists of sObject records
  - Looping that allows for bulk processing of multiple records at a time
  - Locking syntax that prevents record update conflicts
  - Custom public API calls that can be built from stored Apex methods
  - Warnings and errors issued when a user tries to edit or delete a custom object or field that is referenced by Apex

- **Easy to use**
  - Apex is based on familiar Java idioms, such as variable and expression syntax, block and conditional statement syntax, loop syntax, object and array notation. Where Apex introduces new elements, it uses syntax and semantics that are easy to understand and encourage efficient use of the Lightning Platform. Therefore, Apex produces code that is both succinct and easy to write.

- **Data focused**
  - Apex is designed to thread together multiple query and DML statements into a single unit of work on the Salesforce server. Developers use database stored procedures to thread together multiple transaction statements on a database server in a similar way. Like other database stored procedures, Apex does not attempt to provide general support for rendering elements in the user interface.

What is Apex?

- **Rigorous**
  - Apex is a strongly typed language that uses direct references to schema objects such as object and field names. It fails quickly at compile time if any references are invalid. It stores all custom field, object, and class dependencies in metadata to ensure that they are not deleted while required by active Apex code.
- **Hosted**
  - Apex is interpreted, executed, and controlled entirely by the Lightning Platform.
  - Multi-tenant aware
  - Like the rest of the Lightning Platform, Apex runs in a multi-tenant environment. So, the Apex runtime engine is designed to guard closely against runaway code, preventing it from monopolizing shared resources. Any code that violates limits fails with easy-to-understand error messages.
- **Easy to test**
  - Apex provides built-in support for unit test creation and execution. It includes test results that indicate how much code is covered, and which parts of your code could be more efficient. Salesforce ensures that all custom Apex code works as expected by executing all unit tests prior to any platform upgrades.
- **Versioned**
  - You can save your Apex code against different versions of the API. This enables you to maintain behavior.

When Should I Use Apex?

- Create Web services.
- Perform complex validation over multiple objects.
- Create complex business processes that aren't supported by Flow Builder.
- Create custom transactional logic (logic that occurs over the entire transaction, not just with a single record or object).
- Attach custom logic to another operation, such as saving a record, so that it occurs whenever the operation is executed, regardless of whether it originates in the user interface, a Visualforce page, or from SOAP API.

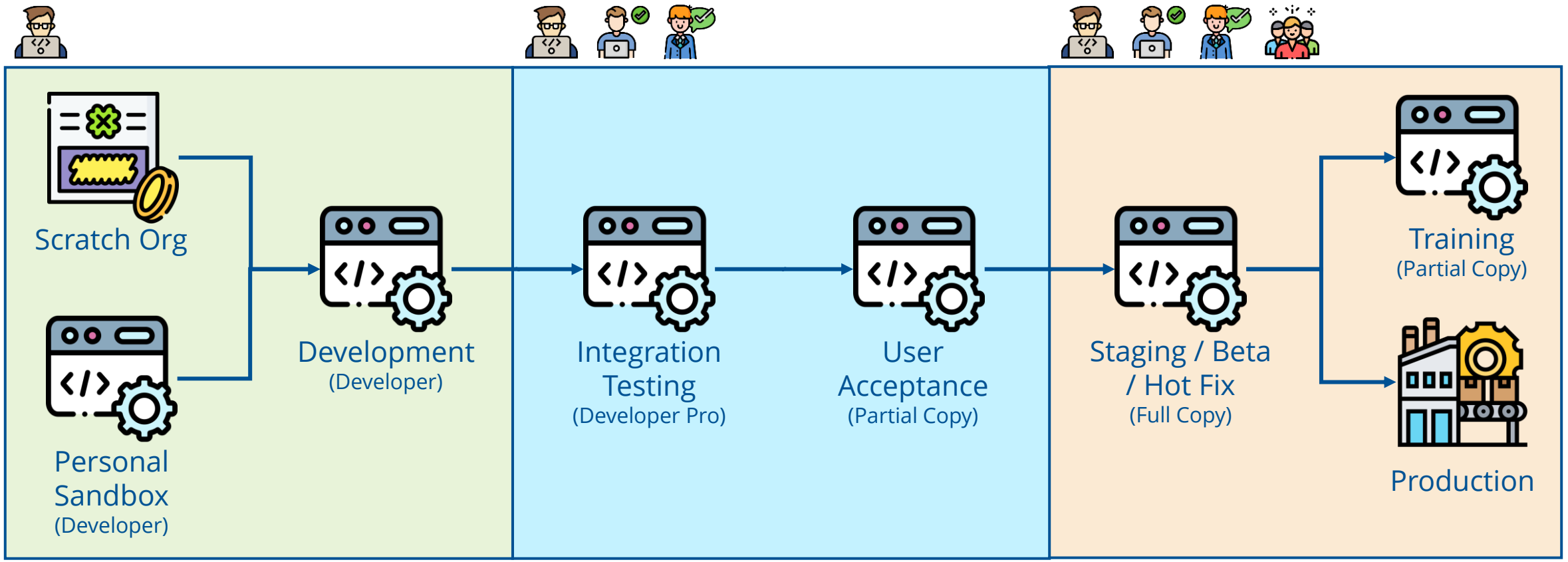
How Does Apex Work?

- Runs entirely on-demand on the Lightning Platform.

- Developer
  - Writes and saves Apex code to the platform, the platform application server first compiles the code into an abstract set of instructions that can be understood by the Apex runtime interpreter, and then saves those instructions as metadata.

- End User
  - Triggers the execution of Apex, the platform application server retrieves the compiled instructions from the metadata and sends them through the runtime interpreter before returning the result.
  - The end user observes no differences in execution time from standard platform requests.

How to get Apex code into Salesforce



Apex Class Definition

- The keyword **class** followed by the name of the class

- Access Modifiers
  - **Private**
    - Declares that this class is only known locally, that is, only by this section of code.
    - This is the default access for inner classes
    - This keyword can only be used with inner classes (or with top-level test classes marked with the @IsTest annotation).
  - **Public**
    - Declares that this class is visible in your application or namespace.
  - **Global**
    - Declares that this class is known by all Apex code everywhere.
    - All classes containing methods defined with the webservice keyword must be declared as global.
    - If a method or inner class is declared as global, the outer, top-level class must also be defined as global.

private | public | global [virtual | abstract | with sharing | without sharing] class *ClassName* [implements *InterfaceNameList*] [extends *ClassName*] { // The body of the class }

Apex Class Definition

- Definition Modifiers (Optional)
  - **With Sharing** and **Without Sharing** keywords specify the sharing mode for this class.
  - **Virtual** definition modifier declares that this class allows extension and overrides.
    - You can't override a method with the override keyword unless the class has been defined as virtual.
  - **Abstract** definition modifier declares that this class contains abstract methods, that is, methods that only have their signature declared and no body defined.

- Optional extensions or implementations or both

private | public | global [virtual | abstract | with sharing | without sharing] class *ClassName* [implements *InterfaceNameList*] [extends *ClassName*] { // The body of the class }

Invoking Apex

- **Anonymous Blocks**
  - Apex code that doesn't get stored in the metadata, but that can be compiled and executed.
- **Triggers**
  - Apex triggers enable you to perform custom actions before or after changes to Salesforce records, such as insertions, updates, or deletions.
  - Custom Code to Flows
- **Asynchronous Apex**
  - Apex offers multiple ways for running your Apex code asynchronously. Choose the asynchronous Apex feature that best suits your needs.
- **Exposing Apex Methods as SOAP Web Services**
  - You can expose your Apex methods as SOAP web services so that external applications can access your code and your application.
- **Exposing Apex Classes as REST Web Services**
  - You can expose your Apex classes and methods so that external applications can access your code and your application through the REST architecture.

Invoking Apex

- **Apex Email Service**
  - You can use email services to process the contents, headers, and attachments of inbound email. For example, you can create an email service that automatically creates contact records based on contact information in messages.

- **Using the InboundEmail Object**
  - For every email the Apex email service domain receives, Salesforce creates a separate InboundEmail object that contains the contents and attachments of that email.

- **Visualforce Classes**
  - In addition to giving developers the ability to add business logic to Salesforce system events such as button clicks and related record updates, Apex can also be used to provide custom logic for Visualforce pages through custom Visualforce controllers and controller extensions.

- **JavaScript Remoting**
  - Use JavaScript remoting in Visualforce to call methods in Apex controllers from JavaScript. Create pages with complex, dynamic behavior that isn't possible with the standard Visualforce AJAX components.

- **Lightning Components/Lightning Web Components**
  - Apex can also be used to provide custom logic for lightning web components through custom controllers and controller extensions.

Execution Governors and Limits

Per-Transaction Apex Limits

Description	Synchronous Limit	Asynchronous Limit
Total # of SOQL queries issued	100	200
Total # of records retrieved by SOQL queries	50,000	50,000
Total # of records retrieved by Database.getQueryLocator	10,000	10,000
Total # of SOSL queries issued	20	20
Total # of records retrieved by a single SOSL query	2,000	2,000
Total # of DML statements issued	150	150
Total # of records processed because of DML statements	10,000	10,000
Total stack depth for any Apex invocation that recursively fires triggers due to insert, update, or delete statements	16	16
Total # of callouts in a transaction	100	100
Maximum cumulative timeout for all callouts in a transaction	120 Seconds	120 Seconds

Execution Governors and Limits

Description	Synchronous Limit	Asynchronous Limit
Maximum number of methods with the future annotation allowed per Apex invocation	50	0 in batch and future contexts; 50 in queueable context
Maximum number of Apex jobs added to the queue with System.enqueueJob	50	1
Total # of sendEmail methods allowed	10	10
Total heap size	6 MB	12 MB
Maximum CPU time on the Salesforce servers	10,000 milliseconds	60,000 milliseconds
Maximum execution time for each Apex transaction	10 minutes	10 minutes
Maximum number of push notification method calls allowed per Apex transaction	10	10
Maximum number of push notifications that can be sent in each push notification method call	2,000	2,000
Maximum number of EventBus.publish calls for platform events configured to publish immediately	150	150

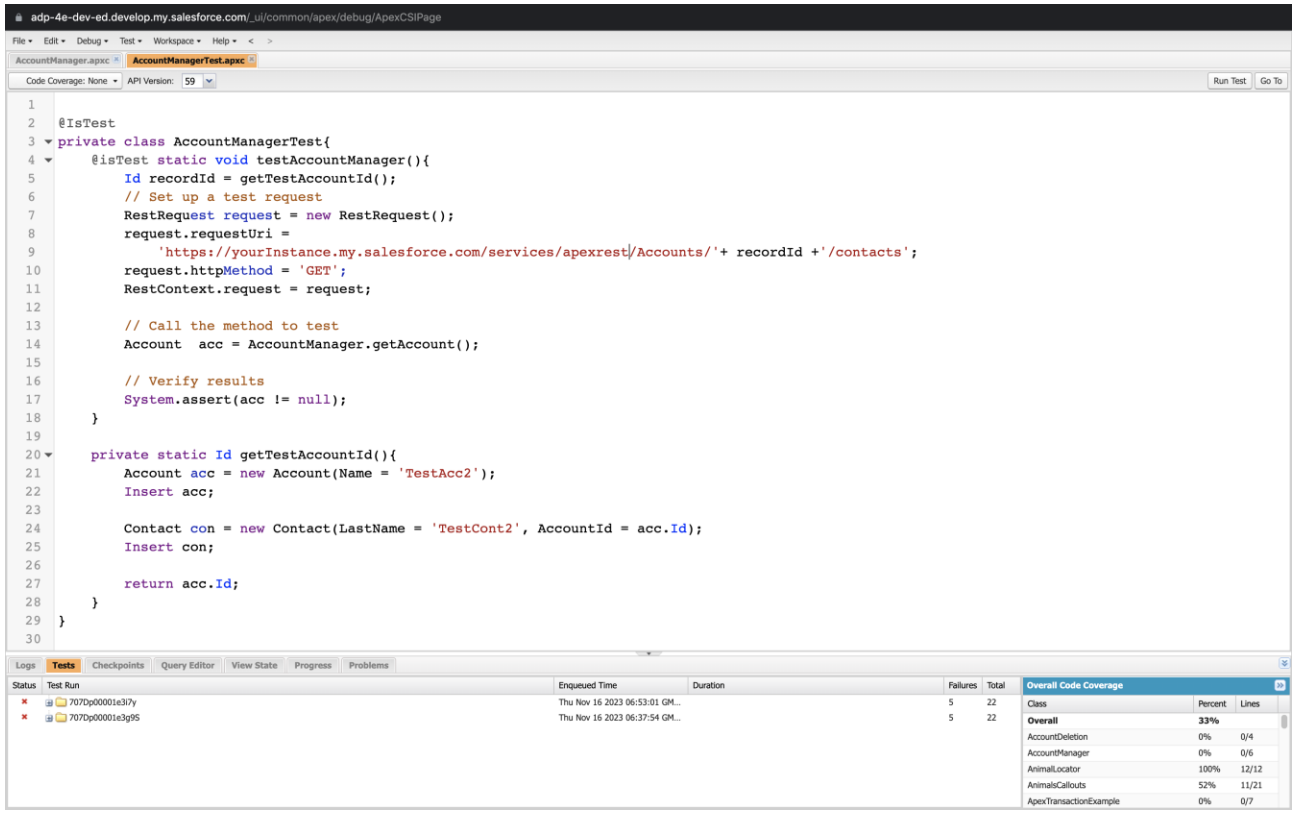
Unit Tests

- Salesforce Requires 75% of your Apex code must be covered by tests before deploying to production

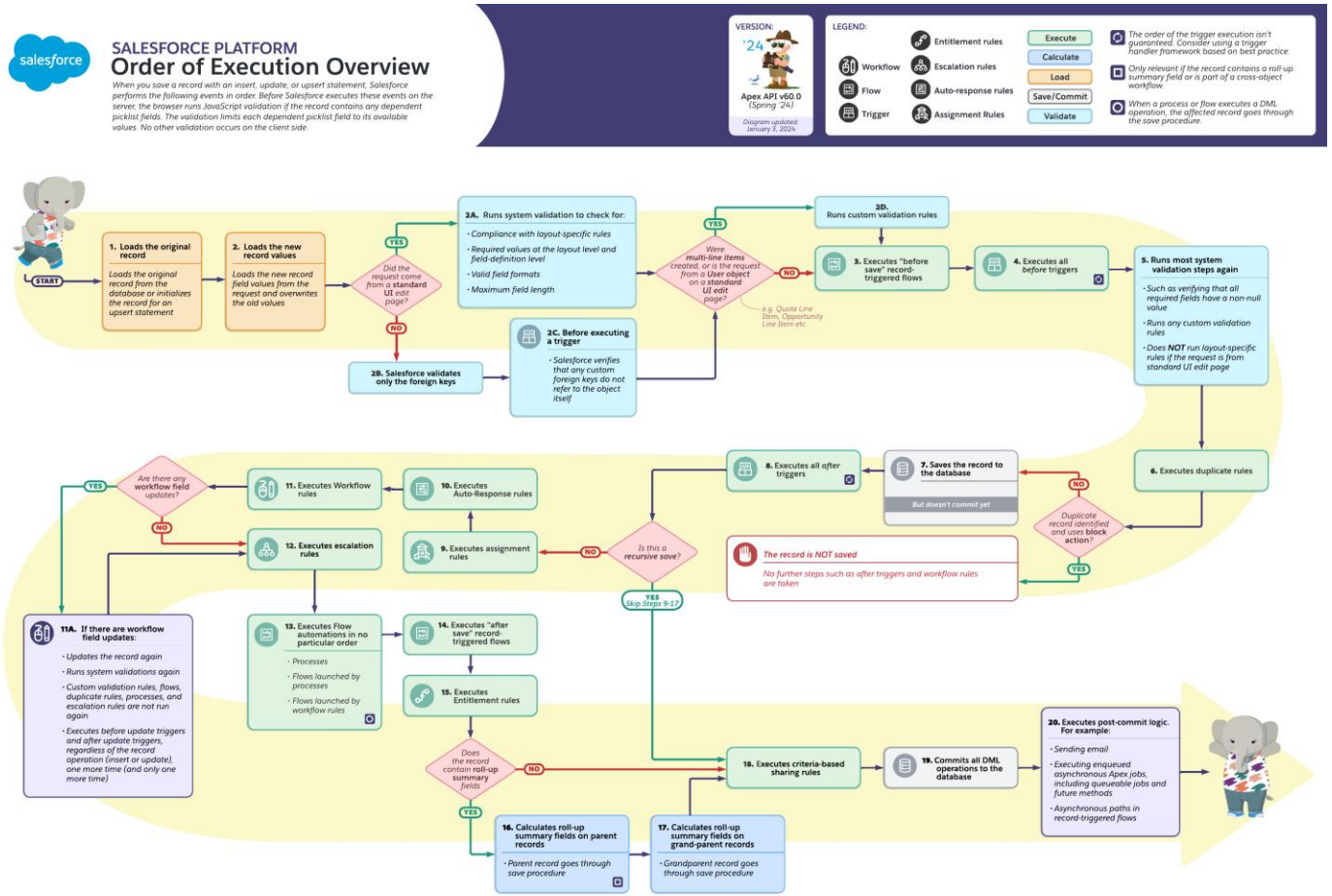
- @IsTest Annotation
  - Define classes and methods that only contain code used for testing your application.
  - Can take multiple modifiers within parentheses and separated by blanks.

- @TestSetup Annotation
  - Used for creating common test records that are available for all test methods in the class.
  - Records that are created in a test setup method are available to all test methods in the test class and are rolled back at the end of test class execution.

- SeeAllData Annotation
  - @isTest(SeeAllData=true) allow test methods to access all org records.
  - The best practice, however, is to run Apex tests with data silo using @isTest(SeeAllData=false).



Order of Execution



[Link to Diagram]

Order of Execution

Link to Diagram

- Loads from the database/initializes record for an upsert.
- Loads the new record field values with system validation steps
- Executes record-triggered flows that are configured to run before the record is saved.
- Executes all before triggers.
- Runs most system validation steps again, such as verifying that all required fields have a non-null value and runs any custom validation rules.
- Executes duplicate rules.
- Saves the record to the database but doesn't commit yet.
- Executes all after triggers.
- Executes assignment rules.
- Executes auto-response rules.
- Executes workflow rules. If there are workflow field updates:
    - Updates the record again.
    - Runs system validations again. Custom validation rules, flows, duplicate rules, processes, and escalation rules aren’t run again.
    - Executes before update triggers and after update triggers, regardless of the record operation (insert or update), one more time (and only one more time)

Order of Execution Con.

Link to Diagram

- Executes escalation rules.
- Executes these Salesforce Flow automations, **but not in a guaranteed order**.
  - Processes
  - Flows launched by processes
  - Flows launched by workflow rules (flow trigger workflow actions pilot)
    - Flow that executes a DML operation, the affected record goes through the save procedure.
- Executes After Save record-triggered flows
- Executes entitlement rules.
- Roll-up summary field or cross-object workflow, performs calculations and updates the roll-up summary field in the parent record. Parent record goes through save procedure.
- If the parent record is updated, and a grandparent record contains a roll-up summary field or is part of a cross-object workflow, performs calculations and updates the roll-up summary field in the grandparent record. Grandparent record goes through save procedure.
- Executes Criteria Based Sharing evaluation.
- Commits all DML operations to the database.
- After the changes are committed to the database, executes post-commit logic are executed. E.g.
  - Sending email
  - Enqueued asynchronous Apex jobs, including queueable jobs and future methods
  - Asynchronous paths in record-triggered flows

Lightning Web Components

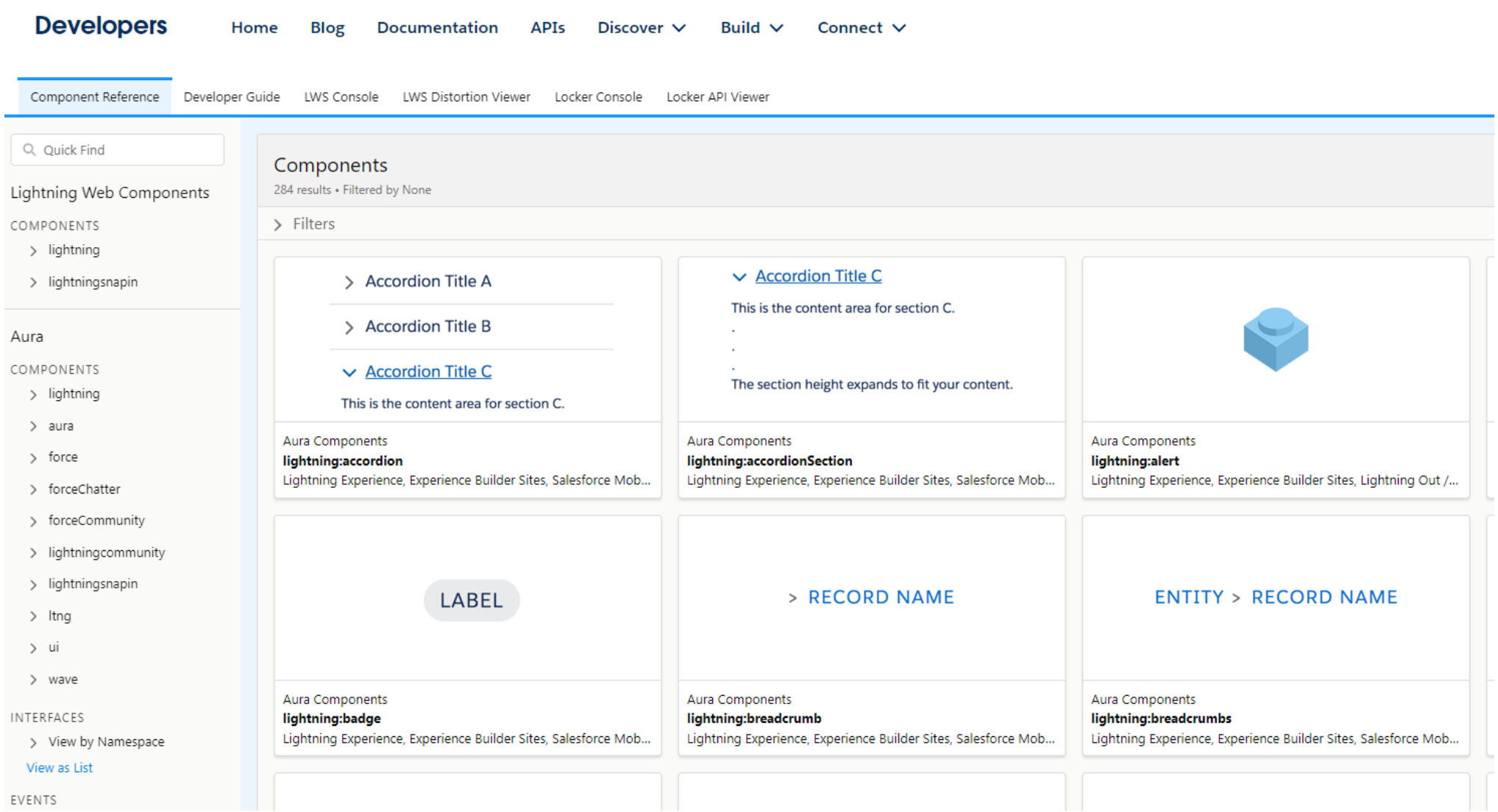
Lightning Web Components

- What are Lightning Web Components?
- Lightning Web Component Structure
- Component Files
- Decorators
- Lightning Data Service
- Apex & Lightning Web Components

What are Lightning Web Components?

- Javascript ECMAScript 6 and above – supports all modern browsers
- Lightning Web Components (LWC) are a user interface (UI) framework that Salesforce Developers use to create customized pages and functions on the Salesforce platform.
- Custom Web Elements using HTML & Javascript
- Salesforce provides base Lightning web components that can be used as building blocks for our own components.
- CAREFUL: Salesforce has old “Aura” components which are similar, but don’t follow modern web standards. We always want to use the Lightning Web Component – not Aura Component!

Lightning web component reference



Lightning Web Component Structure

- UI Components
  - HTML (Required)
  - JavaScript (Required)
  - Metadata (Required)
  - CSS (Optional)
  - SVG Icon (Optional)
- Service Components
  - JavaScript (Required)
  - Metadata (Required)
- CSS Module
  - CSS (Required)
  - Metadata (Required)

Client and Server

Component Files

- Java Script File
  - Every UI component must include a JavaScript file with at least the default code.
  - Defines the HTML element

```
import { LightningElement } from "lwc";
export default class MyComponent extends LightningElement {}
```

  - To import a class, function, or variable declared in a module, use the import statement.
  - To allow other code to use a class, function, or variable declared in a module, use the export statement.
  - The import statement imports LightningElement from the lwc module.

Component Files

- HTML File
  - Every UI component must have an HTML file with the root tag <template>
  - Create the HTML for a Lightning web component declaratively, within the <template> tag. The HTML template element contains your component's HTML.

```
<!-- myComponent.html -->
<template>
  <!-- Replace comment with component HTML -->
</template>
```

Component Files

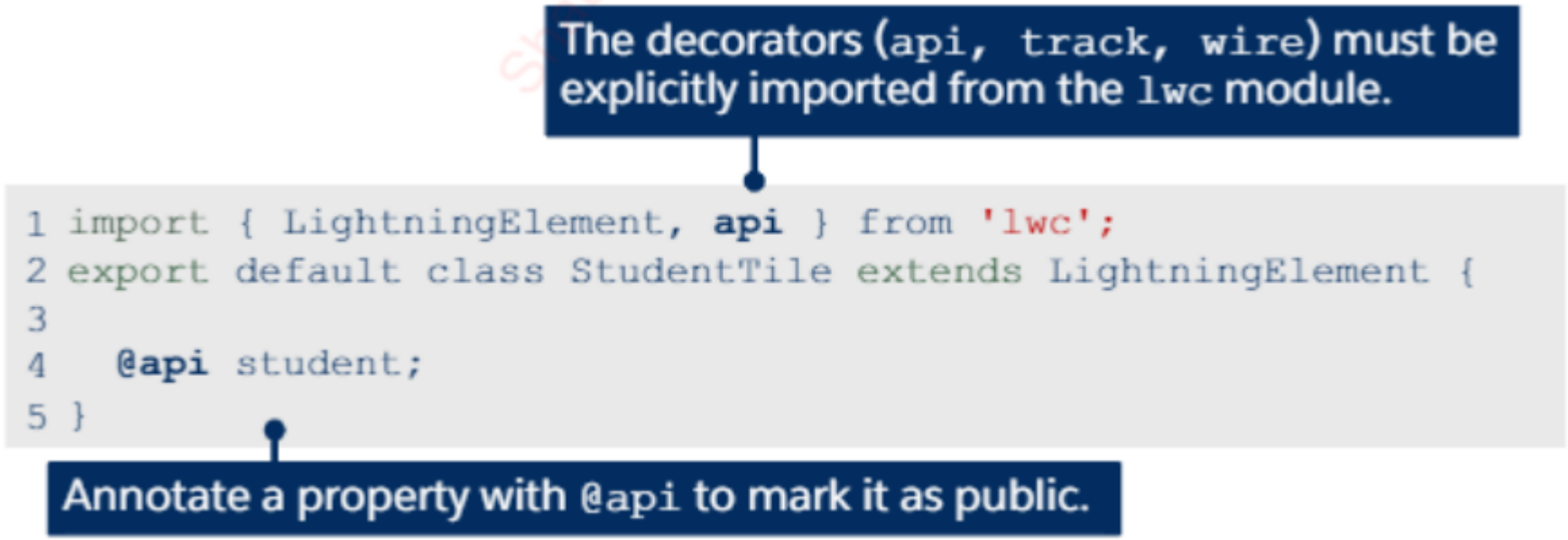
- Configuration File
  - Defines the metadata values for the component, including targets and the design configuration for the Lightning App Builder and Experience Builder.

lightningWebComponent.js-meta.xml

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <LightningComponentBundle xmlns="urn:metadata.tooling.soap.sforce.com" fqn="LightningWebComponent">
3       <apiVersion>45.0</apiVersion>
4       <isExposed>true</isExposed>
5       <targets>
6           <target>lightning__AppPage</target>
7           <target>lightning__RecordPage</target>
8           <target>lightning__HomePage</target>
9       </targets>
10  </LightningComponentBundle>
```

Decorators

- @api
  - Use to expose a public reactive property
- @track
  - Observe object properties or array elements
- @wire
  - Used to read Salesforce data or invoke Apex

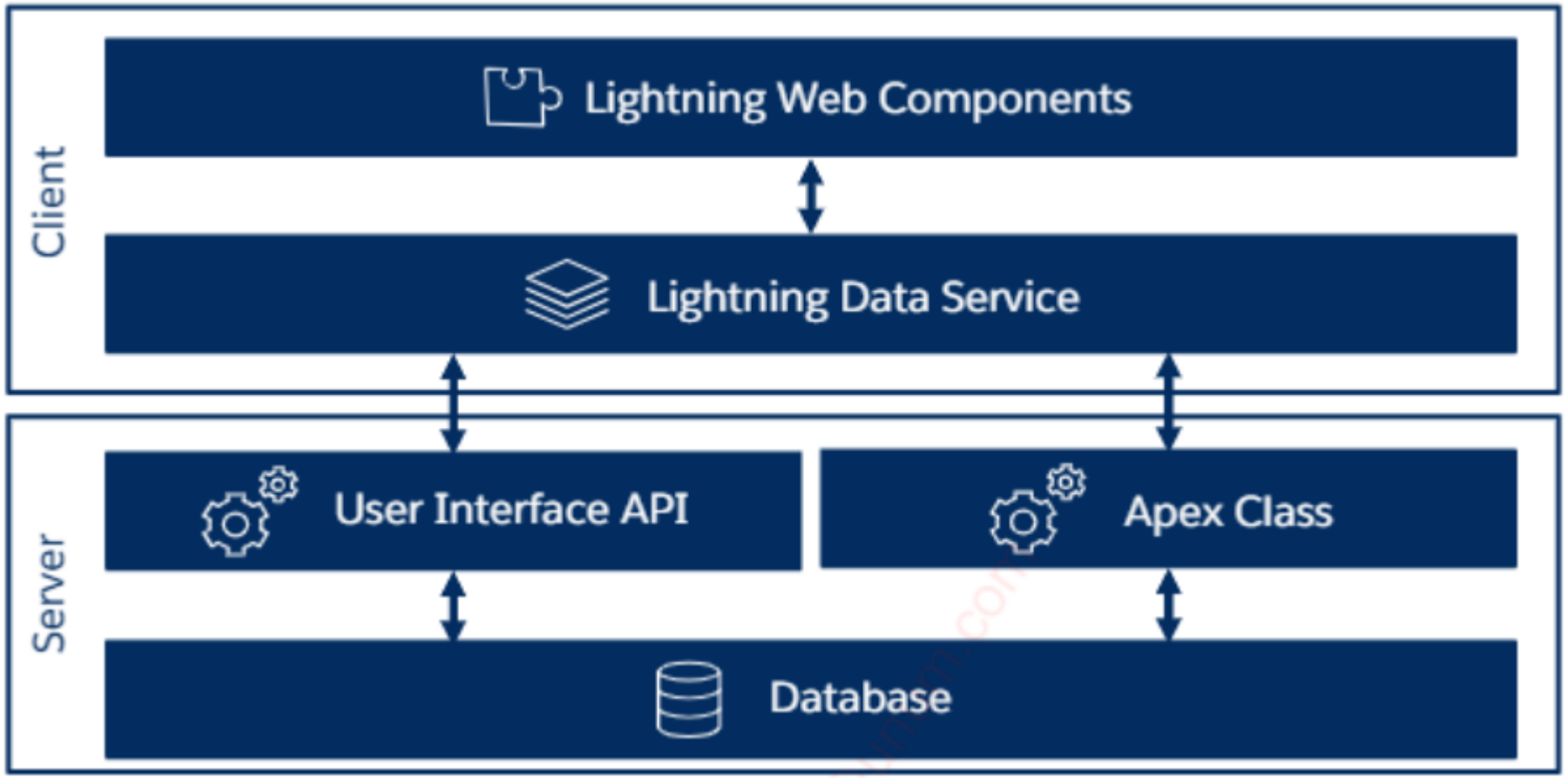


Lightning Data Service

- Ability to load, create, edit, or delete a record in your component without requiring Apex code.
- Handles sharing rules and field-level security
- Examples
  - lightning:recordForm
- Note: LDS only allows you to **work with a single record**. (this is where Apex comes in)

```
<aura:component implements="flexipage:availableForRecordHome,force:hasRecordId">
    <aura:attribute name="recordId" type="String" />
    <aura:attribute name="fields" type="String[]" default="['Name','Industry']" />
    <lightning:recordForm recordId="{!v.recordId}"
                          objectApiName="Account"
                          mode="readonly"
                          fields="{!v.fields}" />
```

Lightning Data Service



Exposing Apex Method to Lightning Web Components

- Method must be
  - **@AuraEnabled** annotated
  - Static
  - Global or Public

- Add **cacheable=true** to cache the method results on the client to improve runtime performance.
  - Method must only **get data**. It can't mutate data.

```
import { LightningElement, wire } from 'lwc';
import getContactList from '@salesforce/apex/ContactController.getContactList';

export default class ApexWireMethodToFunction extends LightningElement {
    contacts;
    error;

    @wire(getContactList)
    wiredContacts({ error, data }) {
        if (data) {
            this.contacts = data;
            this.error = undefined;
```

```
public with sharing class ContactController {
    @AuraEnabled(cacheable=true)
    public static List<Contact> getContactList() {
        return [SELECT Id,Name,Title,Phone,Email FROM Contact LIMIT 10];
    }
}
```

LWC and Apex

Lightning Web Component:

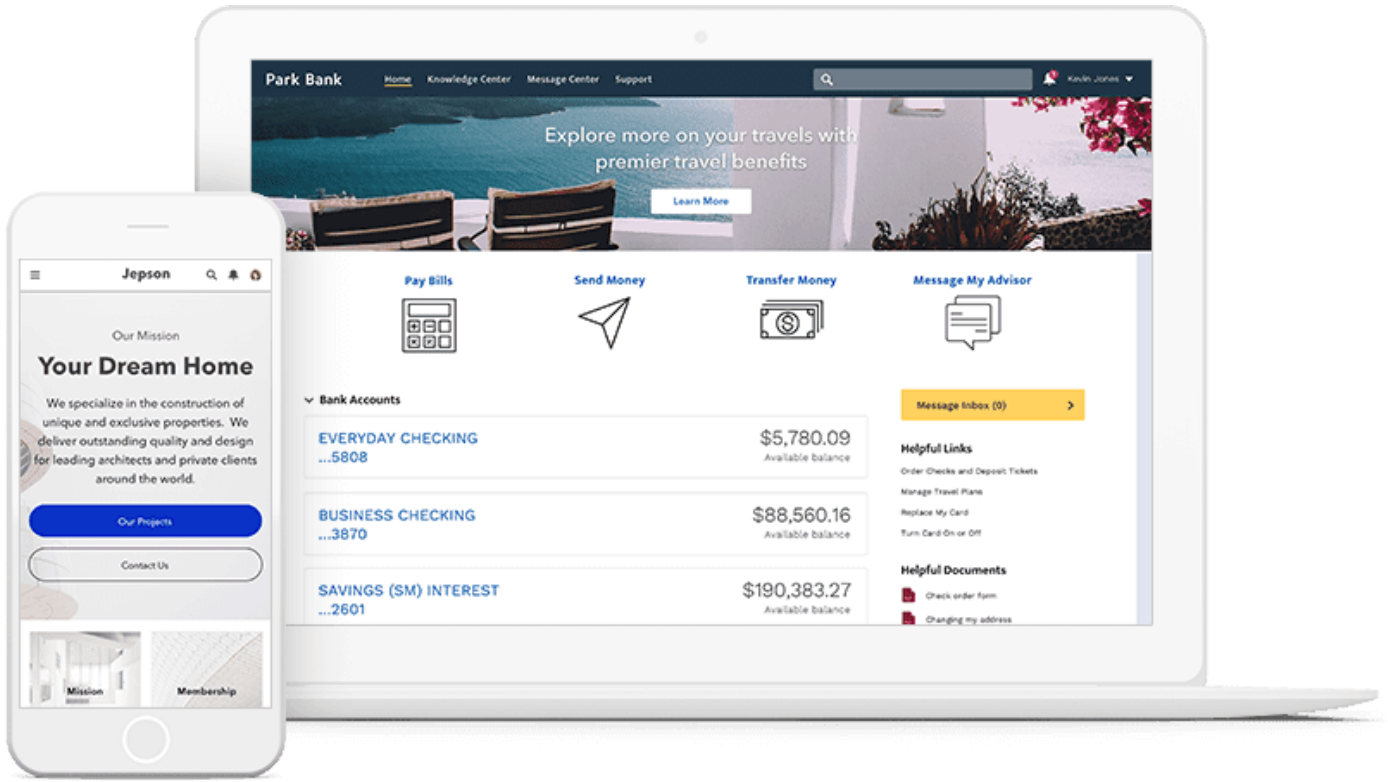
```
import { LightningElement, wire } from 'lwc';
import getContactList from '@salesforce/apex/ContactController.getContactList';

export default class ApexWireMethodToFunction extends LightningElement {
    contacts;
    error;

    @wire(getContactList)
    wiredContacts({ error, data }) {
        if (data) {
            this.contacts = data;
            this.error = undefined;
        } else if (error) {
            this.error = error;
            this.contacts = undefined;
        }
    }
}
```

Apex code executed via @wire:

```
public with sharing class ContactController {
    @AuraEnabled(cacheable=true)
    public static List<Contact> getContactList() {
        return [SELECT Id,Name,Title,Phone,Email FROM Contact LIMIT 10];
    }
}
```

Digital Experience Cloud

Digital Experience Cloud



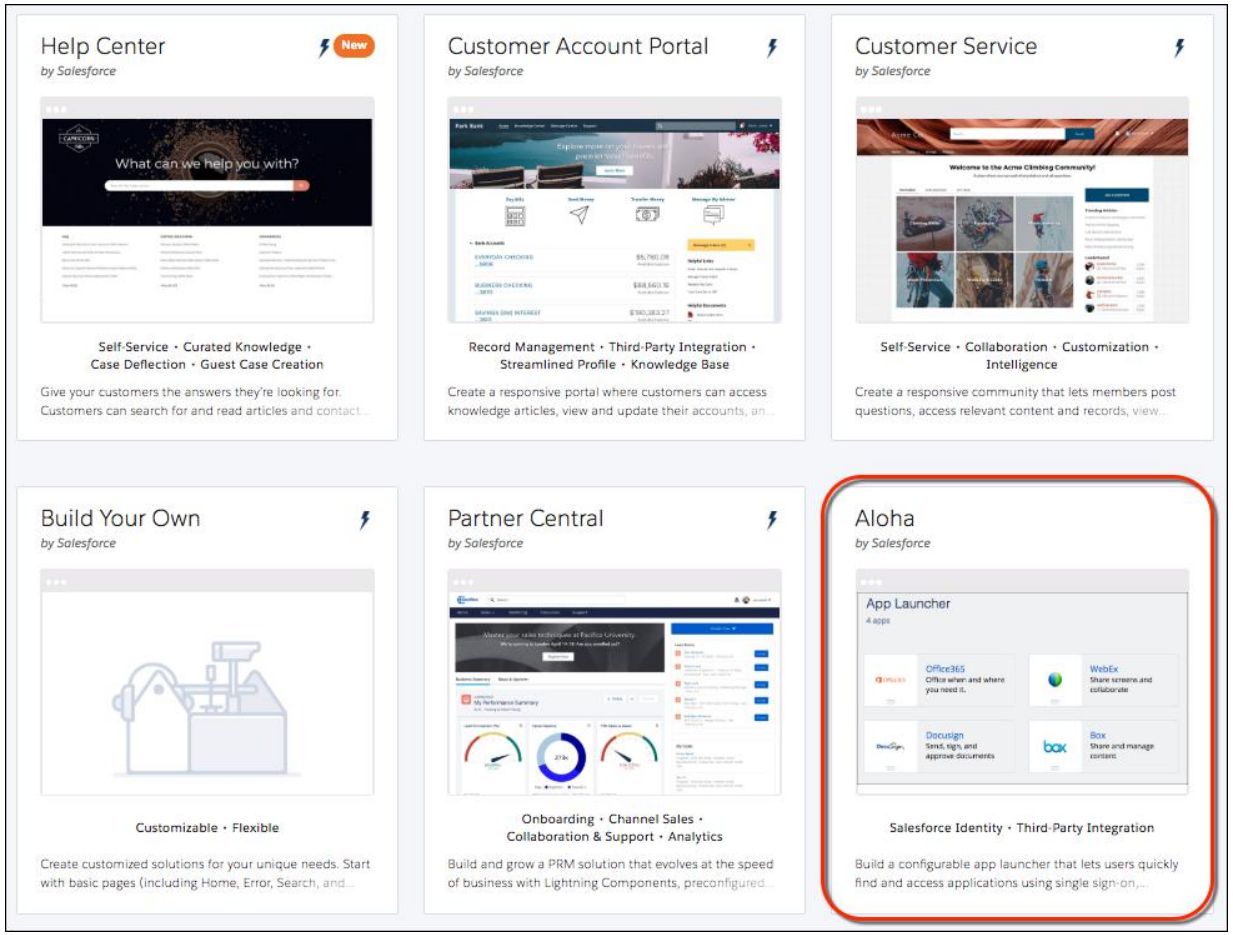
- What is Digital Experience Cloud
- Experience Builder
- Experience Workplace
- Cloud Site Templates

Digital Experience Cloud

- Allows the Creation of a branded digital experiences to share information and collaborate with people who are key to your business processes, such as customers, partners, or employees.
- Create Multiple Sites for different purposes or Persona’s

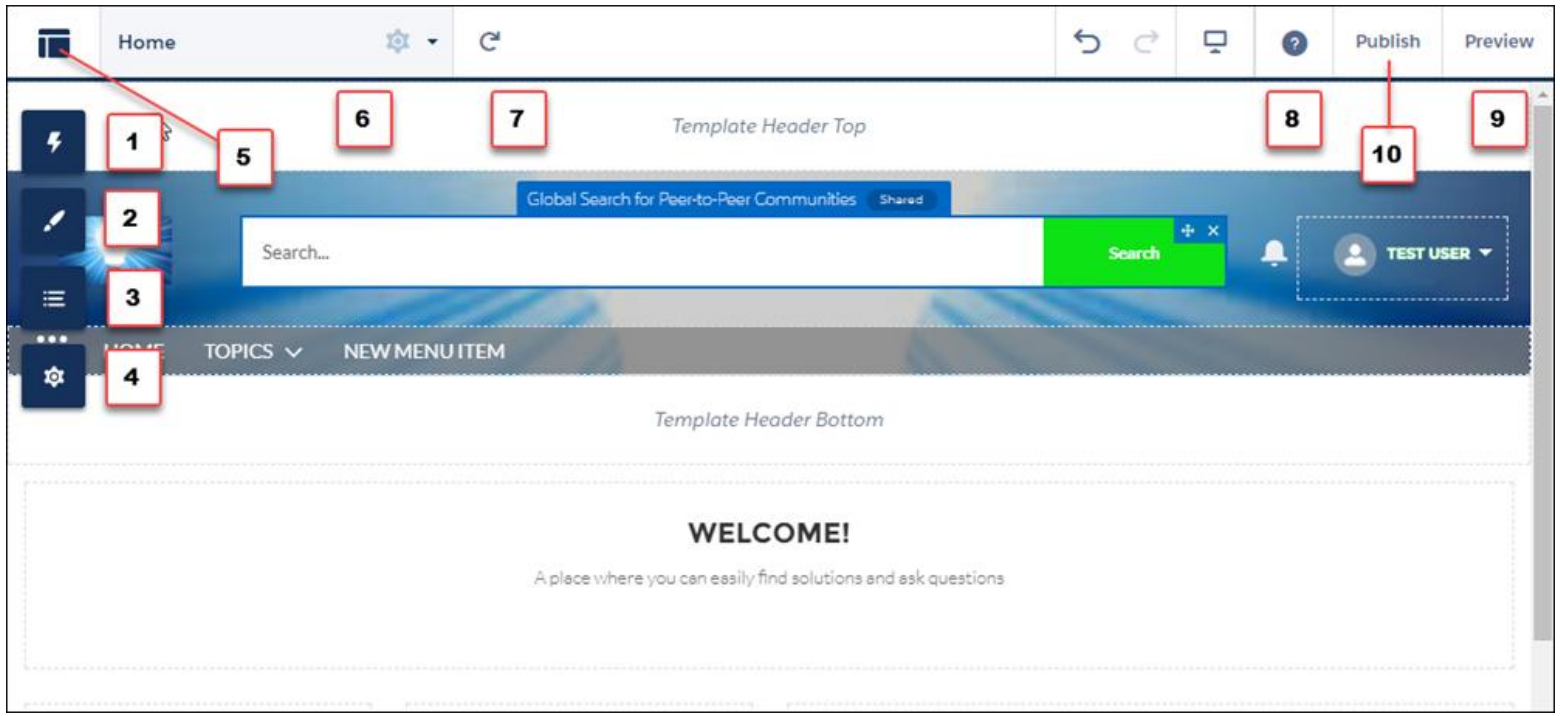
Cloud Site Templates

- Salesforce Created default responsive theme
- Includes pages such as handling the login experience, password resets and identity verification



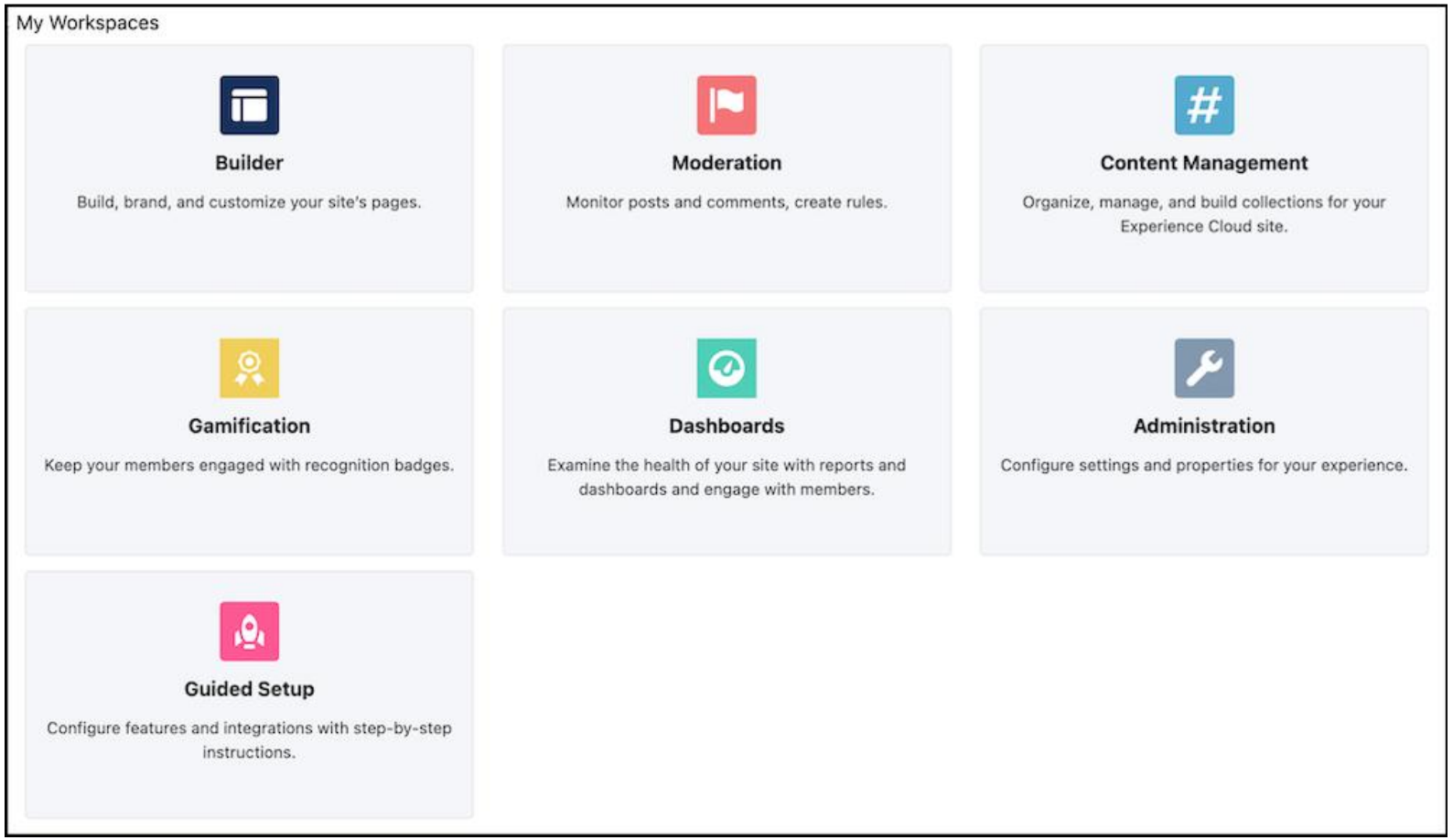
Experience Builder

- Allow you to style, customize and deliver personalized experiences for different audiences
- Allow you to Preview & Publish the Site
- View the site on different Devices



Experience Workspace

- One-stop shop for building, setting up, and monitoring your Experience Cloud site
- Manage the Sites Setup and Branding
- View Dashboards for members, feeds and Groups usage

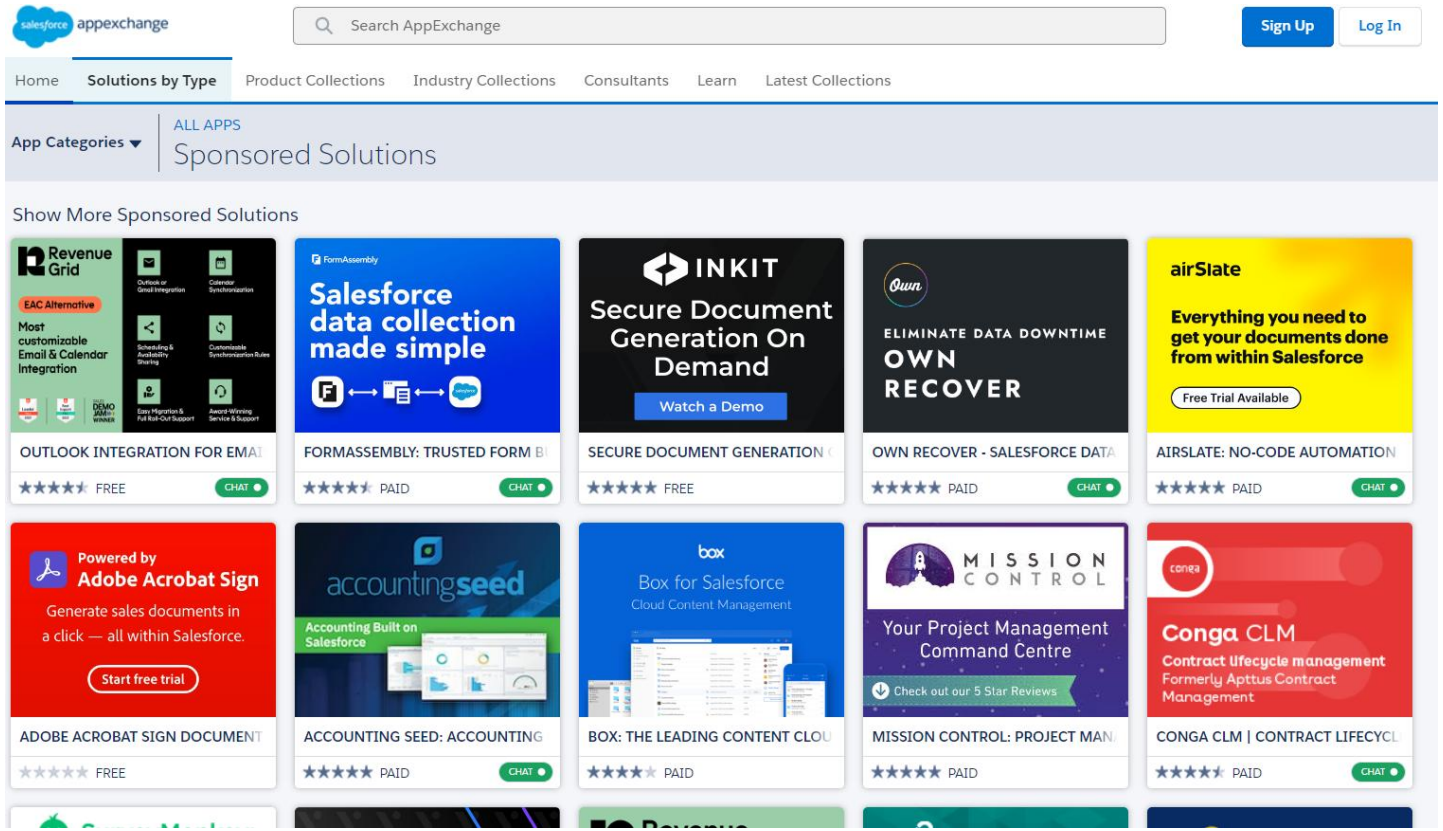


AppExchange

AppExchange

- What is AppExchange?
- What can you do on AppExchange?

AppExchange



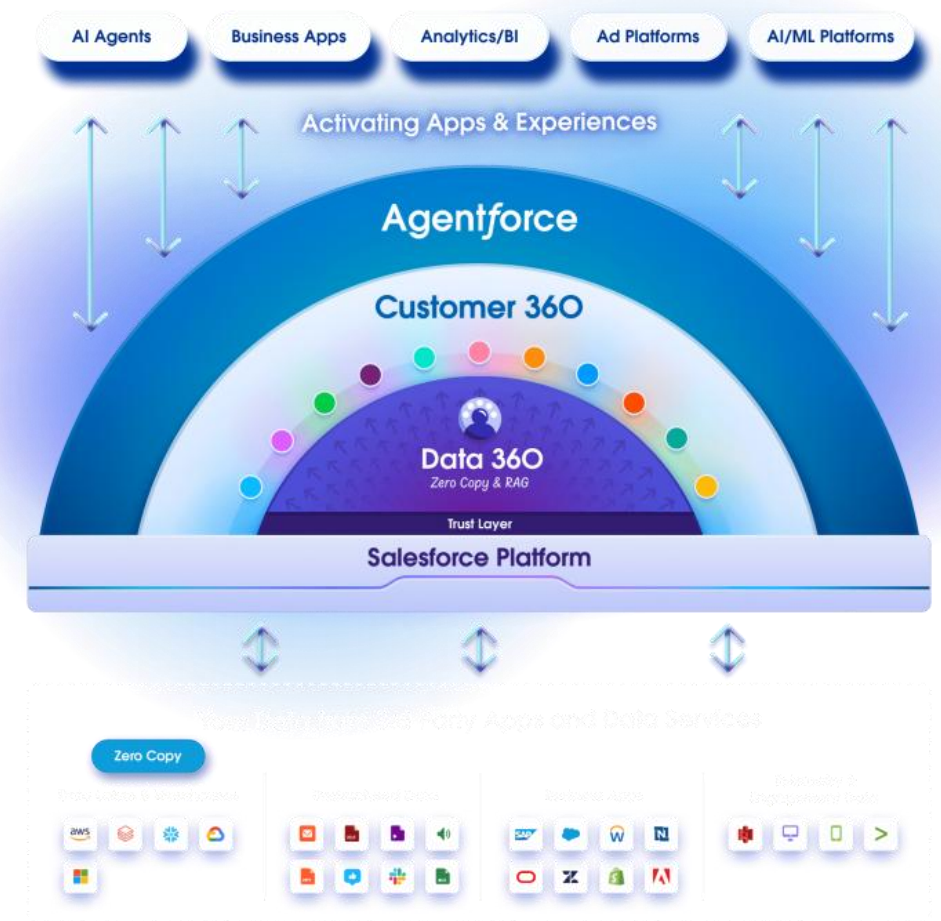
- AppExchange gives partners a place to create and publish extensions to Salesforce.
- Admins can then choose and install solutions to fit the needs of the org

What can you do on AppExchange?

- Browse
  - View the descriptions, reviews, and demos of solutions on AppExchange.
  - They choose the solution that fits their needs.
- Test Drive
  - Review a fully functional demo of the solution as a read-only user
  - Plan their implementation with their admins.
- Install
  - Add the solution and all its components to their Salesforce environment.
- Deploy
  - Immediately give their users access to the solution or customize it for a select group of users.

Data 360

What Is Salesforce Data 360?



- Real-time customer data platform for unifying data
- Connects data from CRM, websites, mobile apps, marketing, and external systems
- Creates a single, accurate, continuously updated customer profile
- Enables analytics, AI, and automation across Salesforce

How Data 360 Works

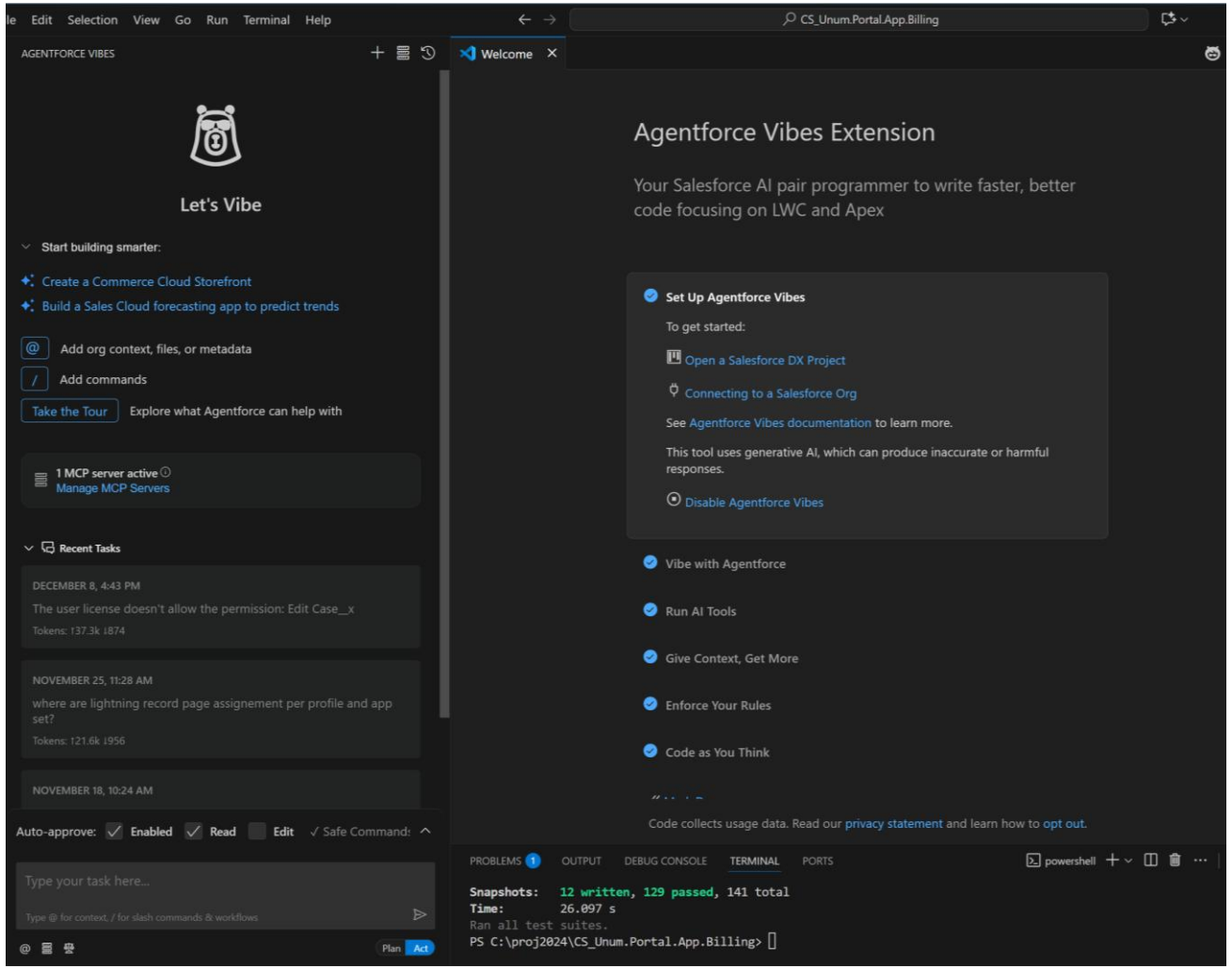
- Connect: Ingest batch and streaming data from many sources
- Harmonize: Standardize data using models to align fields and formats
- Unify: Identity resolution creates a unified customer profile
- Analyze & Predict: Use unified data for BI dashboards and AI models
- Act: Use insights in Salesforce apps, automations, and activations

Why Data 360 Matters

- Real-time insights enable more personalised experiences
- AI becomes more accurate with unified, high-quality data
- Automations can act on clean and complete customer information
- Supports cross-cloud journeys across Sales, Service, Marketing, and Commerce

Agentforce Vibes

- AI-powered developer tool
- Visual Studio Code extension
- Code completion
- MCP tools integration
- Connected to Salesforce Environment
- Launches a browser to verify local apps
- Build and edit flows



Exercises

Go to https://trailhead.salesforce.com/ and login to your account.

- Complete the following Modules that are part of the Build Flows with Flow Builder Trail
  - Flow Builder Basics
  - Build a Simple Flow (Optional)
- Continue the Developer Beginner Trail and complete the following
  - Apex Basics & Database
  - Apex Triggers (Optional)
  - Apex Testing (Optional)
  - Lightning Web Components Basics
- Project: Build a Bear-Tracking App with Lightning Web Components

References

- https://help.salesforce.com/s/articleView?id=sf.process_which_tool.htm&type=5
- https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_how_does_apex_work.htm
- https://developer.salesforce.com/blogs/2018/12/introducing-lightning-web-components
- https://architect.salesforce.com/1/asset/immutable/s/e6cf2ac/assets/images/Salesforce-Order-Of-Execution-Diagram.png
- https://developer.salesforce.com/docs/component-library/overview/components
- https://help.salesforce.com/s/articleView?id=data.c360_a_data_cloud.htm&type=5
- https://developer.salesforce.com/docs/ai/agentforce/guide/get-started.html