Phase 2: Org Setup & Configuration

Introduction

A well-structured and properly configured development environment is the cornerstone of any Salesforce project. In the context of the **ApexHub** initiative, Phase 2 focuses on establishing a scalable Salesforce development setup that supports both declarative (Admin) and programmatic (Developer) customizations. This phase ensures that the development team has all the necessary tools, configurations, and structures in place to build, test, and deploy features efficiently.

Before any real development can begin, it is critical to set up the Salesforce org with the right features enabled, user access policies defined, and repository structure aligned with **Salesforce DX best practices**. A misconfigured environment could lead to deployment failures, access issues, and wasted effort. Hence, this phase is dedicated to laying the foundation for all future development activities.

Objectives

The objectives of this phase are as follows:

1. Set up a **Salesforce DX (SFDX)** environment and configure the CLI for smooth project deployment.
2. Establish a **Scratch Org** to serve as a sandboxed environment for isolated development and testing.
3. Configure the Salesforce org with **profiles, roles, and permission sets** aligned with the ApexHub project's needs.
4. Enable core features like **Platform Events, Flows, and Lightning Web Components (LWCs)**.
5. Design and validate a **repository structure** that follows Salesforce DX standards for maintainability and scalability.
6. Perform initial validation to confirm that the environment is correctly configured and supports custom development.

Activities

1. Salesforce DX Setup

The development team began by installing the **Salesforce CLI (SFDX CLI)**, which acts as the primary tool for managing metadata, creating scratch orgs, and interacting with Salesforce environments from the command line.

Key steps included:

- Authorizing the **Dev Hub Org** using `sfdx auth:web:login`.
- Creating the **Scratch Org** using project-specific configuration files.
- Setting scratch org duration policies (typically 7–30 days) to ensure developers work in clean, temporary environments.
- Establishing best practices for push (`sfdx force:source:push`) and pull (`sfdx force:source:pull`) commands to keep local code and org metadata in sync.

This step ensured that all developers could create, test, and refresh their environments independently without conflicts.

---

2. Org Configuration

Once the scratch org was created, it was customized to support ApexHub's requirements.

- **Permission Sets:**
  A dedicated permission set called `ApexHub_Admin` was created. This gave admin-level permissions to developers and testers while maintaining principle-of-least-privilege practices for other roles.
- **Profiles and Roles:**
  Default profiles were adjusted to restrict unnecessary access. Roles were created to test data access scenarios for different user types.
- **Feature Enablement:**
  - **Platform Events** were enabled for real-time data communication.
  - **Flows** were enabled to allow declarative automation.
  - **Lightning Web Components (LWC)** support was turned on for front-end development.
- **Security Configuration:**
  Field-level security and object-level permissions were configured to ensure that custom objects such as `Recipe__c` and `RecipeLog__c` were accessible only to authorized users.

This step ensured the org could support both business logic and UI customization while respecting governance limits.

---

3. Repository & Folder Structure

The repository was structured based on Salesforce DX conventions. This ensures that the project remains modular, easy to maintain, and scalable for future development.

```
force-app/ main/ default/ classes/ -> Apex Classes for backend logic triggers/ -> Apex Triggers for automation objects/ -> Custom Objects (Recipe__c, RecipeLog__c) lwc/ -> Lightning Web Components for UI flows/ -> Declarative Flows and automation permissionsets/ -> Permission sets for access management
```

This structure provides:

- **Separation of Concerns:** Code, UI, and automation are clearly organized.
- **Version Control Compatibility:** Developers can collaborate using GitHub, track changes, and conduct code reviews.
- **CI/CD Readiness:** The project structure is aligned with Salesforce DevOps best practices, enabling future pipeline automation.

---

4. Initial Validation

After setup, the environment was validated through smoke testing:

- Tested deployment of sample Apex classes and triggers.

- Validated access to **standard objects** (Account, Contact, Opportunity).
- Created and tested **custom objects** (`Recipe__c` and `RecipeLog__c`) to simulate core project requirements.
- Confirmed that **Flows and LWCs** could be deployed and executed successfully.
- Verified that assigned permission sets provided the correct level of access without overexposing sensitive functionality.

Deliverables

At the end of Phase 2, the following deliverables were achieved:

1. Salesforce CLI and Dev Hub authorized successfully.
2. Scratch Org created and validated for development.
3. Profiles, roles, and permission sets (`ApexHub_Admin`) configured.
4. Org features enabled: Platform Events, Flows, LWCs.
5. Repository structure designed and aligned with Salesforce DX.
6. Initial validation tests completed successfully.

Expected Outcomes

The completion of this phase ensures that:

- The project is aligned with **modern Salesforce DevOps practices**.
- Developers have isolated environments to experiment without risk.
- The repository structure supports long-term scalability.
- The groundwork is ready for **Process Automation (Phase 4)**, **Apex Programming (Phase 5)**, and **User Interface Development (Phase 6)**.

This structured setup significantly reduces the chances of deployment errors, access conflicts, and integration failures in the later stages of the project.