Salesforce Agent & Lego Rover Integration Workshop

Unlocking the Power of Data + AI with Salesforce Agents

Workshop Objectives

The core objectives are:

- **Demonstrate Physical Robotic Actions:** Show attendees how a Salesforce Einstein Agent in the cloud can make decisions that trigger physical actions in the real world.
- **Practical Application:** Participants will build and program a LEGO SPIKE Prime robot to perform various actions based on how an incoming "inspection report" (represented as a case) is classified by the Salesforce agent.

This workshop showcases a practical application of Data + AI in Salesforce, proving that AI-driven decisions can seamlessly translate into immediate, real-world, robotic actions, thereby automating triage and physical logistics for issues like contamination or inspection requirements.

You would get hands-on experience in building Salesforce agents, setting up automations, and implementing event-driven integration, including connectivity with a LEGO SPIKE Prime robot.

Salesforce Admin Configuration

1. Ensure Agents are Enabled

1. In Salesforce Setup, use the Quick Find box to search for **Einstein Setup**.
2. Toggle the switch to **Turn on Einstein** (if not already enabled).
3. Use the Quick Find box to search for and select **Agentforce Agents**.
4. Toggle the switch to **Turn on Agentforce**.
5. Ensure the **Agentforce (Default) Agent** is enabled.

2. Create Brand New Prompt Template

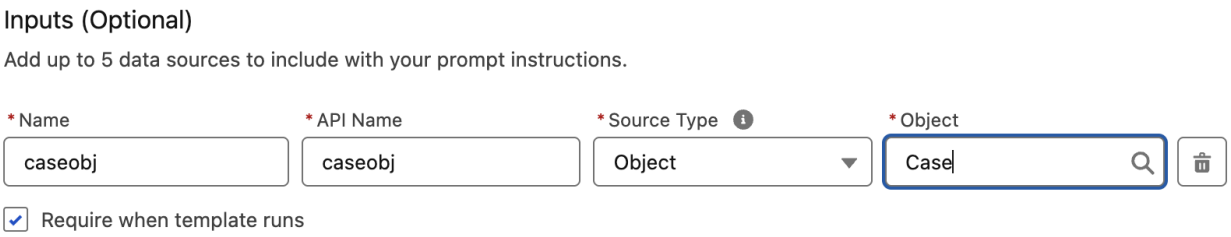
This template is of type Flex and will be used later in the Agent.

1. In Salesforce Setup, search for **Prompt Builder**.
2. Click **New Prompt Template**.
3. Set the **Name**: Support Classify the case.
4. Set the **API Name**: Support_Classify_the_case.
5. Select **Prompt Template Type**: Flex.

6. Template Description

   “Analyze the provided Case Description and any associated notes, then classify the required support as one of the following three types: "Inspection," "Normal," or "Contaminated”

7. Set the Input as below for the flex prompt template



8. Update content of it

   Analyze the provided Case Description and any associated notes, then classify the required support as one of the following three types: "Inspection," "Normal," or "Contaminated."

   Classification Definitions:

   Inspection:

   Required when the reported issue is ambiguous, relates to site access/layout, involves a persistent service failure without clear cause, or requires a safety assessment before normal service can proceed.

   Examples: Blocked access, container missing (but not confirmed stolen), unusual odor source, potential structural damage to a collection area.

   Normal:

   Required when the issue is a standard service request or failure that can be resolved by a routine crew/driver/dispatcher action, and does not involve potentially hazardous or prohibited waste.

   Examples: Missed scheduled pickup, container replacement due to normal wear and tear, requesting a new service/bin delivery, billing inquiry.

   Contaminated:

   Required when the case explicitly mentions or strongly suggests the presence of prohibited, hazardous, or non-compliant material in a bin/load that requires special handling, disposal, or regulatory reporting.

   Examples: Report of chemicals/batteries/paint in a recycling bin, medical waste found, asbestos suspicion, fire residue in the container.

Input Case Description: {!$Input:caseobj.Description}

Addition Case information: {!$RecordSnapshot:caseobj.Snapshot}

Output: Classification: [Inspection | Normal | Contaminated]

9. Click **Save**.

3. Create New Field "Support Classification"

Create a custom picklist field on the Case object for classification.

1. In Salesforce Setup, navigate to **Object Manager** > **Case** > **Fields & Relationships**.
2. Click **New** to create a custom field.
3. Select **Picklist** for the Data Type.
4. **Field Label**: Support Classification.
5. **Field Name**: Support_Classification.
6. Enter the following values, each on a new line:
    - RECYCLING_OK
    - CONTAMINATED
    - INSPECTION
7. Set field-level security and add the field to the necessary page layouts.
8. Click **Save**.

4. Creation of a new platform Events

Search for Platform Events in the Admin setup

LEGO_Command__e

Field	Purpose
`Command__c`	`RECYCLING_OK`, `CONTAMINATED`, `INSPECTION`
`Case_Id__c`	originating Case ID

LEGO_Robot_Status__e

Field	Purpose
`Case_Id__c`	echoed ID
`Message__c`	e.g., “Target Zone Reached”

5. Create a record triggered flow to Publish Platform Event

Flow Logic:

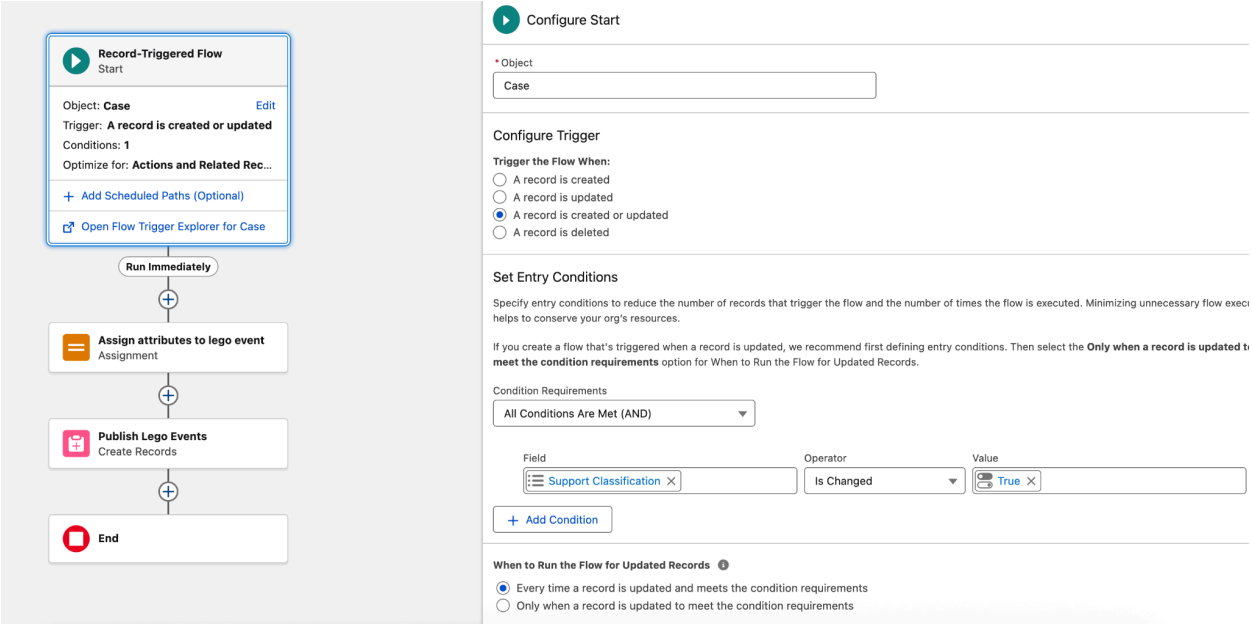
1. **Check for Change:** The flow first checks if the `Support Classification` field has been modified on the Case record.
2. **Assignment:** If the `Support Classification` field was modified, the flow assigns the following values to a new `LEGO Command` object:
    - **Case ID:** The `Case` record's `Id` is assigned to the `LEGO Command` object's `Case_Id__c` field.
    - **Support Classification Value:** The new value of the `Case`'s `Support_Classification__c` field is assigned to the `LEGO Command`

object's `Command__c` field.

3. **Creation/Update:** The flow then creates or updates the `LEGO Command` object (without using an upsert operation).

Objects and Fields Involved (Format: Object - Field: Interaction)

- `Case` - `Id`: The Case record's Id is assigned to the `LEGO Command` object's `Case_Id__c` field.
- `Case` - `Support_Classification__c`: The new value of the `Support Classification` field is assigned to the `LEGO Command` object's `Command__c` field.



5. Create a New Employee Agent (Topic & Action Creation)

Create a dedicated agent to handle employee inquiries about the Rover.

1. In Salesforce Setup, search for **Agentforce Agent**.
2. Click **Open in Builder** for **Employee Agent**
3. Create a **New Topic**.
    - **Name**: Support Classification Analysis
    - **Classification Description**: Analyze the provided case description and associated notes to classify the required support type. Trigger further actions based on the classification.

- **Scope**: My job is to classify the required support as 'Inspection,' 'Normal,' or 'Contaminated' based on the provided case information, update the case's support classification field with the determined value
- **Instructions**: Review the case description and any associated notes to determine the appropriate support classification.
- **Instructions**: Classify the required support as 'Inspection,' 'Normal,' or 'Contaminated' based on the analysis.
- **Instructions**: Update the case's support classification field with the determined value after classification.

4. Add the **Action**:
   - Click **New Action** and select **action type** as “Prompt”
   - **Reference Action**:Support Classify the case
   - Agent Action Instruction: “Analyze the provided Case Description and any associated notes, then classify the required support as one of the following three types: "Inspection," "Normal," or "Contaminated.”
5. Add **other actions** to this topic

   Select and add other out of the box actions to this topic by clicking on the **“Add from Asset library”** button

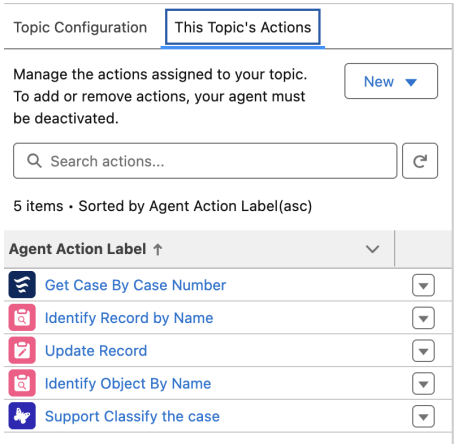
   Get Case By Case Number

   Identify Record by Name

   Update Record

   Identify Object By Name

After doing this step, it show as per below screen shot.



6. Test Employee Agent

Before testing create one case which you would like to use for your testing. After come back to the agent builder screen.

1. Click on the **Refresh button** in the conversation plane
2. Type “Give me support classification for the case "**Your Case Number**"”
3. Provide name of the contact associated with that case
4. Update the "Support Classification" with this value
5. Check the response from the agent

7. Click **Save** and **Activate** the agent.

8. Connected App (External Client App)

Navigate to External Client App Manager in the Setup

External Client App Name: Lego Agent Bridge

Contact Email

Distribution State: Local

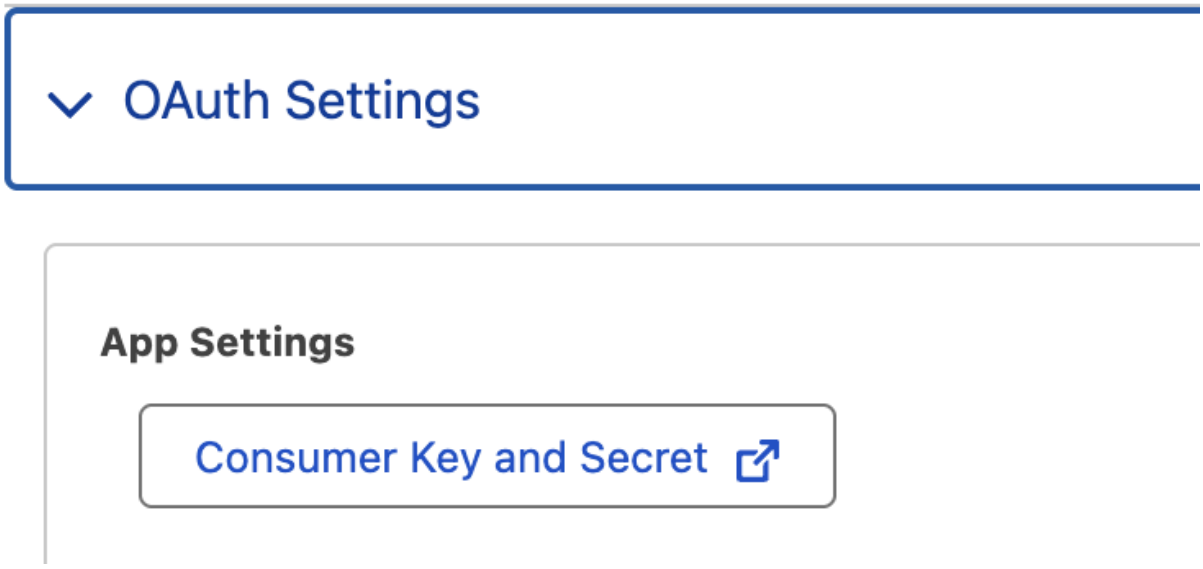
Enable oAuth Setting:
Callback URL: http://localhost:8080/callback

Scopes:

- Manage user data via APIs (`api`)
- Full access (`full`)
- Refresh tokens (`refresh_token`, `offline_access`)
- Access the Salesforce API Platform (`sfap_api`)

Uncheck all the checkboxes in **Flow Enablement and Security** section and click save.

- In the **External App** in the salesforce admin console,click on the Consumer key and Secret button there after you will receive the verification code in the email address used in the external app, provide that code then after it would reveal the Consumer key and Secret.



Create sf_config.json in the root project directory as instructed by the instructor and insert them into file content as below:

```
{
 "CLIENT_ID" : "Your Client ID",
 "CLIENT_SECRET" : "Your Client Secret"
}
```

Then save the file

Rover Configuration

1. Explain Parts of Rover

The Rover is comprised of several key components:

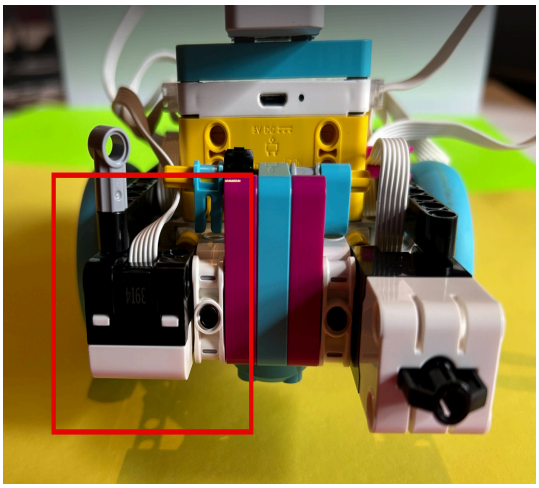
- **Main Processor/Controller**: The brain of the Rover, running the Python scripts.

- **Motor/Actuators**: Components responsible for movement and sorting arm operation. **(Already Connected)**
- **Distance Sensor**: Used to detect the presence of an object in the sorting area.
- **Color Sensor**: Used to identify the type of material (e.g., plastic, glass, metal).

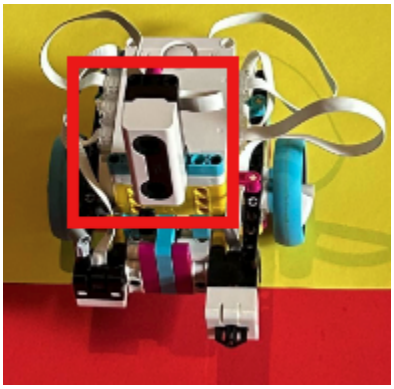
2. Add Distance Sensor and Color Sensor

Physically connect the following sensors to the main controller:

Sensor Type	Port	Notes
Color Sensor	Port B	Mounted downward on the black grabber arm to detect colored zones (see photo).
Distance Sensor	Port F	Detects obstacles and allows stopping ~2 inches before the barrier.



Color Sensor Placement



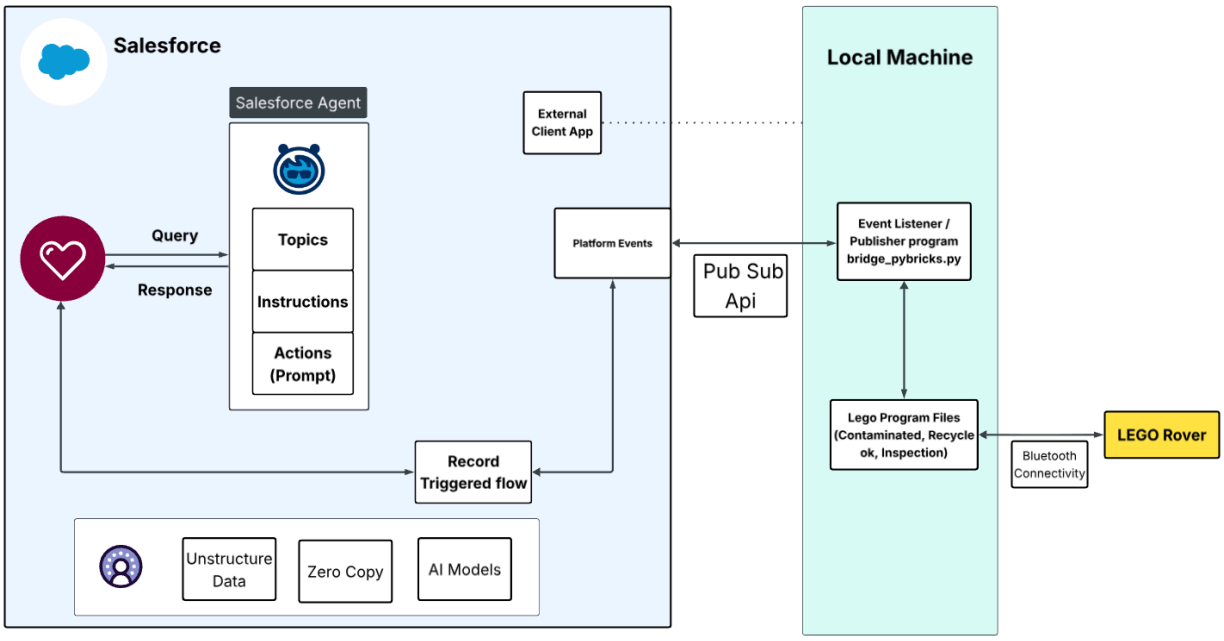
Ultrasonic Sensor

3. How It Works – Architecture Overview

1. Salesforce
    - Agent or automation publishes a *LEGO_Command__e* event:
        - `Command__c` = `RECYCLING_OK`, `CONTAMINATED`, or `INSPECTION`
        - `Case_Id__c` = the originating Case Id
    - The robot returns a *LEGO_Robot_Status__e* event:
        - `Case_Id__c`
        - `Message__c` (e.g., “Target Zone Reached”)
2. Python Bridge (`bridge_pybricks.py`)
    - Subscribes to Salesforce Pub/Sub
    - When a command arrives:
        - Runs `pybricksdev run ble contamination_sorter.py`
        - Parses `STATUS:` output lines
        - Publishes a Robot Status PE back to Salesforce
3. Pybricks Program (`contamination_sorter.py`)
    - Uses reflection values, not color names
    - Detects RED / YELLOW / GREEN zones
    - Drives to the appropriate zone
    - Prints `STATUS:` lines that the bridge forwards to Salesforce
    -

Public Github Repo: https://github.com/mikeletulle/LegoWorkshop/

Architecture diagram



4. Run Color Calibration

The color sensor requires calibration for accurate color identification as it depends on the intensity of the light in the room. Use color_calibrater.py to measure reflection values on each zone.

```
pybricksdev run ble color_calibrater.py
```

Notice most frequent REF values on each color

```
REF = 4    AMB = 4.0    COLOR = Color.NONE
REF = 4    AMB = 4.0    COLOR = Color.NONE
REF = 4    AMB = 4.0    COLOR = Color.NONE
REF = 4    AMB = 4.0    COLOR = Color.NONE
REF = 4    AMB = 4.0    COLOR = Color.NONE
REF = 4    AMB = 4.0    COLOR = Color.NONE
REF = 4    AMB = 4.0    COLOR = Color.NONE
REF = 5    AMB = 4.0    COLOR = Color.NONE
REF = 4    AMB = 3.9    COLOR = Color.NONE
REF = 4    AMB = 4.0    COLOR = Color.NONE
REF = 6    AMB = 3.9    COLOR = Color.NONE
REF = 2    AMB = 4.8    COLOR = Color.NONE
REF = 2    AMB = 6.5    COLOR = Color.NONE
REF = 8    AMB = 6.2    COLOR = Color.YELLOW
REF = 13    AMB = 6.3    COLOR = Color.YELLOW
REF = 13    AMB = 6.3    COLOR = Color.YELLOW
REF = 13    AMB = 6.2    COLOR = Color.YELLOW
REF = 13    AMB = 6.2    COLOR = Color.NONE
REF = 13    AMB = 6.2    COLOR = Color.YELLOW
REF = 13    AMB = 6.2    COLOR = Color.YELLOW
REF = 13    AMB = 6.2    COLOR = Color.YELLOW
REF = 13    AMB = 6.2    COLOR = Color.YELLOW
REF = 13    AMB = 6.2    COLOR = Color.YELLOW
```

There after replace values in contamination_sorter.py as shown below:

```
# ========================================

# --- COLOR CALIBRATION VALUES (EDIT HERE) ---

# ========================================

CAL_RED    = 5

CAL_GREEN  = 11

CAL_YELLOW = 13.5  # Range 13-15
```

5. Establishing Connectivity between Salesforce & Machine

This creates connectivity between salesforce and your machine

- Run Salesforce Login Script (sf_login.py)

  This script:

  Opens Salesforce OAuth login UI in a browser

  Stores access token in sf_token_info.json

  Required before Pub/Sub event subscription works

```
None
    python sf_login.py
```

  After successful login, you should see token info to **sf_token_info.json**

Running the System

Start bridge:

```
python bridge_pybricks.py
```

Wait until you see the message in the output window:

[INFO] salesforce_pubsub: Subscribing to /event/LEGO_Command__e on api.pubsub.salesforce.com:7443

Test with Anonymous Apex

Send command via Salesforce Debug Anonymous (Apex) just to test the connectivity.

```
LEGO_Command__e eventRecord = new LEGO_Command__e(
    Command__c = '', (Chose one from INSPECTION,RECYCLING_OK,CONTAMINATED)
    Case_Id__c = 'Your SF CASE ID'
);
Database.SaveResult sr = EventBus.publish(eventRecord);
System.debug('SR: ' + sr);
```

Testing End to End:

Create the following cases in the Salesforce Service cloud to test the end-to-end agent and classification flow.

Scenario	Outcome	Case Support Classification (Expected)
Recycling_OK	Rover moves to Green	RECYCLING_OK
Contamination	Rover moves to Red	CONTAMINATED
Inspection	Rover moves to Yellow	INSPECTION

1. **Create 3 different service cases**
2. Open the **Service cloud** application.
    - Create 3 new cases with following fields:
    - ----------
    - Subject: Prohibited Waste / Hazardous Material Alert
    - Contact: Lauren Bailey
    - With Description like "Driver alerted the dispatcher that the contents of the last commercial pickup (Site ID: 900 Industrial Park, Asset ID 55112 - 8-yard compactor) appears to be highly contaminated. The load contains multiple sealed 5-gallon buckets marked with chemical warning labels (possibly paint thinner or solvent) and several discarded fluorescent light tubes. The material has been flagged and isolated at the yard. Requires hazmat team notification and segregation for special disposal."
    - ----------
    - Subject: Missed Collection (Residential)

- Contact: Lauren Bailey
- With Description like “"Customer called stating their gray residential waste cart (Asset ID 10045) was missed on the scheduled route yesterday morning. They confirmed the bin was placed out by 7:00 AM and accessible at the curb. We verified that no exception tag was left. The bin is full and needs to be emptied ASAP. Issue is a standard missed pickup requiring a make-up collection."
- "
- ---
- Subject: Access Problem / Unidentified Debris
- Contact: Lauren Bailey
- With Description like “Driver reported being unable to service the commercial property at 450 King Street for the past three weeks. The assigned bin is behind a newly installed chain-link fence. We have no key or access code on file. Driver noted that the area is very congested and suggests a smaller bin size might be required if the current location is permanent. Needs a supervisor site inspection to assess access requirements and potentially relocate the asset."
-

Running the System

Start bridge:

```
python bridge_pybricks.py
```

Open the Agent

Give me support classification for the case "Your Case Number"

Provide the contact associated with the case if asked

Update the "Support Classification" with this value

Case would be updated

Platform event would be published

Rover should move

Project Files

- `bridge_pybricks.py`
- `salesforce_pubsub.py`
- `contamination_sorter.py`
- `color_calibrater.py`
- `sf_login.py`
- `sf_config.json` (local only)
- `robot_board.png`
- `Sensor_placement.png`
- `requirements.txt`

Unlocking the Power of Data + AI with Salesforce Agents

This workshop is a hands-on guide for setting up a Salesforce Einstein Agent to interact with a physical device, the "Rover," which simulates a smart recycling sorting process.Workshop Objectives

- **Implement Integration:** Configure the end-to-end event-driven integration between the Salesforce platform and the physical LEGO SPIKE Prime robot (Rover).
- **Configure Salesforce Agent:** Set up the necessary Salesforce components, including a new Prompt Template, Case Classification, and a dedicated Employee Agent.
- **Rover Setup:** Understand and connect the Rover's physical components (sensors, motors) and prepare the local Python environment.
- **Validate Flow:** Test the entire system by sending commands from Salesforce to trigger physical actions on the Rover.

-----1. Initial Setup: Local Environment1.1. Pybricks Firmware Installation

1. Go to code.pybricks.com.
2. Click the **Bluetooth icon**, select your hub, and install the firmware.
3. **If Bluetooth scanning doesn't work:**

- Navigate to `chrome://flags` in your browser.
- Enable **Experimental Web Platform Features**.

4. **If Bluetooth still fails:**
   - Hold the hub's Bluetooth button.
   - Plug the USB cable into your laptop (the LED will cycle: pink → green → blue).
5. *Windows 11 Note:* You may need the WinUSB driver, which is available via the Pybricks troubleshooting wizard.

1.2. Laptop Setup: Virtual Environment and Dependencies

1. **Create and Activate Virtual Environment:**
   - Create the environment:

```
None
```

   - `python3 -m venv venv`

   - **Activate on macOS / Linux:**

```
None
```

   - `source venv/bin/activate`

   - **Activate on Windows:**

```
None
```

   - `venv\Scripts\activate`

2. *Your terminal should now show the `(venv)` prefix.*
3. **Install Dependencies:** Install all required packages:

```
None
4. pip install -r requirements.txt
```

5. **Required Packages:** `pybricksdev`, `grpcio`, `fastavro`, `aiohttp`, `requests`.

Refer to the public GitHub repository for project files: LegoWorkshop-----2. Salesforce Configuration2.1. Enable Agent Features

1. In Salesforce Setup, Quick Find **Einstein Setup**.
2. Toggle the switch to **Turn on Einstein** (if not already enabled).
3. Quick Find **Agentforce Agents**.
4. Toggle the switch to **Turn on Agentforce**.
5. Ensure the **Agentforce (Default) Agent** is enabled.

2.2. Create Prompt Template for Classification

Create a **Flex** type prompt template that the agent will use to classify incoming service cases.

1. In Salesforce Setup, search for **Prompt Builder**.
2. Click **New Prompt Template**.
3. Set the **Name**: `Support Classify the case`.
4. Set the **API Name**: `Support_Classify_the_case`.
5. Select **Prompt Template Type**: `Flex`.
6. Paste the following content into the template:

```
None
```

7. `Analyze the provided Case Description and any associated notes, then classify the required support as one of the following three types: "Inspection," "Normal," or "Contaminated."`
8.
9. `Classification Definitions:`
10.
11. `Inspection:`
12. `Required when the reported issue is ambiguous, relates to site access/layout, involves a persistent service failure without clear cause, or requires a safety assessment before normal service can proceed.`
13. `Examples: Blocked access, container missing (but not confirmed stolen), unusual odor source, potential structural`

```
   damage to a collection area.
14.
15. Normal:
16. Required when the issue is a standard service request or
    failure that can be resolved by a routine
    crew/driver/dispatcher action, and does not involve
    potentially hazardous or prohibited waste.
17. Examples: Missed scheduled pickup, container replacement due
    to normal wear and tear, requesting a new service/bin
    delivery, billing inquiry.
18.
19. Contaminated:
20. Required when the case explicitly mentions or strongly
    suggests the presence of prohibited, hazardous, or
    non-compliant material in a bin/load that requires special
    handling, disposal, or regulatory reporting.
21. Examples: Report of chemicals/batteries/paint in a recycling
    bin, medical waste found, asbestos suspicion, fire residue
    in the container.
22.
23. Input Case Description: {\\!$Input:caseobj.Description}
24.
25. Addition Case information:
    {\\!$RecordSnapshot:caseobj.Snapshot}
26.
27. Output: Classification: [Inspection | Normal | Contaminated]
```

28. Click **Save**.

2.3. Create Custom Case Field

Create a custom picklist field on the Case object to store the classification result.

1. Navigate to **Object Manager** > **Case** > **Fields & Relationships**.
2. Click **New** and select **Picklist** for the Data Type.
3. **Field Label**: `Support Classification`.
4. **Field Name**: `Support_Classification`.
5. Enter the following values (each on a new line):
    - `RECYCLING_OK`
    - `CONTAMINATED`
    - `INSPECTION`
6. Set field-level security and add the field to the necessary page layouts.
7. Click **Save**.

2.4. Define Platform Events

Set up the events used for communication between Salesforce and the Rover.

Event Name	Field	Purpose
`LEGO_Command__e`	`Command__c`	`RECYCLING_OK`, `CONTAMINATED`, or `INSPECTION` command for the Rover.
	`Case_Id__c`	The originating Salesforce Case ID.
`LEGO_Robot_Status__e`	`Case_Id__c`	Echoed ID from the command.
	`Message__c`	Status update message (e.g., “Target Zone

		Reached”).

2.5. Create Flow: Publish Platform Event

This flow is a simple automation that triggers an event when a case is classified as contaminated.

1. In Salesforce Setup, search for **Flow**.
2. Click **New Flow** and select **Record-Triggered Flow**.
3. **Object**: `Case`.
4. **Trigger the Flow When**: A record is created or updated.
5. **Entry Conditions**: `Support_Classification__c` equals `'CONTAMINATED'`.
6. **Optimize the Flow For**: Actions and Related Records (After Save).
7. Add an **Action** element.
8. **Action Type**: Platform Event > `Rover_Action__e` (assumes this event exists).
9. **Set Field Values**:
    - `Action_Type__c = 'Contamination_Detected'`
    - `Case_ID__c = Case.Id`
10. **Label**: Publish Contamination Event.
11. Click **Save** and **Activate**.

2.6. Create Employee Agent

Create a dedicated agent to handle employee inquiries about the Rover's status and usage.

1. In Salesforce Setup, search for **Agentforce Agent**.
2. Click **Open in Builder** for **Einstein Copilot**.
3. Create a **New Topic**:
    - **Name**: `Employee Rover Guide`.
    - **Classification Description**: Provides guidance to employees about the Rover, including status checks and how to report issues.
    - **Scope**: Only answer questions about the Rover's operational status and guide

employees on case creation.

    - **Instructions**: Use the `Support_Classify_the_case` template for all status inquiries. Always ask the employee for the device ID.

4. Add the **Action**:
    - Click **New Action**.
    - **Action Name**: `Check_Rover_Health`.
5. Link the `Support_Classify_the_case` template to this Topic.
6. Click **Save** and **Activate** the agent.

2.7. Configure Connected App (External Client App)

1. Navigate to **External Client App Manager** in the Setup.
2. **External Client App Name**: `Lego Agent Bridge`.
3. Add a **Contact Email**.
4. **Distribution State**: `Local`.
5. Enable OAuth Settings:
    - **Callback URL**: `http://localhost:8080/callback`
    - **Scopes**: `api`, `full`, `refresh_token`, `offline_access`, `sfap_api`.
6. **Uncheck** all checkboxes in the **Flow Enablement and Security** section.
7. Retrieve the **Consumer Key** and **Secret** from the External App console (requires email verification).

-----3. Rover Configuration and Calibration3.1. Rover Hardware

The Rover is composed of:

- **Main Processor/Controller**: The hub running the Python scripts.
- **Motor/Actuators**: Responsible for movement and the sorting arm.
- **Distance Sensor (Port F)**: Detects objects, allowing the Rover to stop before a barrier.
- **Color Sensor (Port B)**: Mounted on the grabber arm to detect colored sorting zones.

3.2. Run Color Calibration

Calibrate the color sensor to your environment's lighting conditions.

1. Run the calibration script:

```
None
2. pybricksdev run ble color_calibrater.py
```

3. Note the most frequent Reflection (`REF`) values for the RED, GREEN, and YELLOW zones.
4. Update the calibration values in the `contamination_sorter.py` script:

```
Python
5. # --- COLOR CALIBRATION VALUES (EDIT HERE) ---
6. CAL_RED    = 5    # Replace with your measured REF value
7. CAL_GREEN  = 11   # Replace with your measured REF value
8. CAL_YELLOW = 13.5 # Replace with your measured REF value
```

-----4. System Operation and Testing4.1. Establish Connectivity

1. **Configure Client Credentials:** Create an `sf_config.json` file in the root directory and paste your credentials:

```
None
2. {
3.   "CLIENT_ID" : "**Your Client ID**",
4.   "CLIENT_SECRET" : "**Your Client Secret**"
5. }
```

6. **Run Salesforce Login Script:** This script handles OAuth login and stores the access token in `sf_token_info.json`.

```
None
7. python sf_login.py
```

4.2. Run the System

1. **Start the Bridge:** Start the Python Bridge to begin listening for Salesforce commands.

```
None
```

2. `python bridge_pybricks.py`

3. **Test with Anonymous Apex (Manual Command):** Use the Salesforce Debug Anonymous (Apex) window to send a test event. Replace `**Your SF CASE ID**` and choose a valid command.

```
None
```

4. `LEGO_Command__e eventRecord = new LEGO_Command__e(`
5. `    Command__c = 'INSPECTION', // Choose one: INSPECTION, RECYCLING_OK, CONTAMINATED`
6. `    Case_Id__c = '**Your SF CASE ID**'`
7. `);`
8.
9. `Database.SaveResult sr = EventBus.publish(eventRecord);`
10. `System.debug('SR: ' + sr);`

4.3. End-to-End Testing Scenarios

Create the following service cases in the Service Cloud to test the entire flow: Agent classification → Flow trigger → Platform Event → Python Bridge → Rover action.

Scenario	Case Subject	Case Description Focus	Expected Rover Action	Classification
Contamination	Prohibited Waste / Hazardous Material Alert	Mentions "chemical warning labels" and "hazmat team notification."	Moves to Red zone	`CONTAMINATED`

Recycling_OK	Missed Collection (Residential)	"Issue is a standard missed pickup requiring a make-up collection." (Maps to Normal in classification definitions)	Moves to Green zone	`RECYCLING_OK`
Inspection	Access Problem / Unidentified Debris	Mentions "newly installed chain-link fence" and "Needs a supervisor site inspection."	Moves to Yellow zone	`INSPECTION`

Test Employee Agent:

1. Open the **Service Console**.
2. Start a new conversation with the Employee Agent.
3. **Test Utterance**: "What is the status of Rover ID Person?"
4. Verify the agent uses the custom prompt template and references the (simulated) status.

-----Project Files

- `bridge_pybricks.py`
- `salesforce_pubsub.py`
- `contamination_sorter.py`
- `color_calibrater.py`

- `sf_login.py`
- `sf_config.json` (local only)
- `robot_board.png`
- `sensor_placement.pngI have thoroughly reviewed and refined the workshop guide to enhance clarity, correct grammar, and improve the overall flow, making the instructions easier for the audience to follow.Salesforce Agent & Rover Integration Workshop`

Unlocking the Power of Data + AI with Salesforce Agents

This workshop is a hands-on guide for setting up a Salesforce Einstein Agent to interact with a physical device, the "Rover," which simulates a smart recycling sorting process.Workshop Objectives

- **Implement Integration:** Configure the end-to-end event-driven integration between the Salesforce platform and the physical LEGO SPIKE Prime robot (Rover).
- **Configure Salesforce Agent:** Set up the necessary Salesforce components, including a new Prompt Template, Case Classification, and a dedicated Employee Agent.
- **Rover Setup:** Understand and connect the Rover's physical components (sensors, motors) and prepare the local Python environment.
- **Validate Flow:** Test the entire system by sending commands from Salesforce to trigger physical actions on the Rover.

-----1. Initial Setup: Local Environment1.1. Pybricks Firmware Installation

1. Go to code.pybricks.com.
2. Click the **Bluetooth icon**, select your hub, and install the firmware.
3. **If Bluetooth scanning doesn't work:**
    - Navigate to `chrome://flags` in your browser.
    - Enable **Experimental Web Platform Features**.
4. **If Bluetooth still fails:**
    - Hold the hub's Bluetooth button.
    - Plug the USB cable into your laptop (the LED will cycle: pink → green → blue).
5. *Windows 11 Note:* You may need the WinUSB driver, which is available via the Pybricks troubleshooting wizard.

1.2. Laptop Setup: Virtual Environment and Dependencies

1. **Create and Activate Virtual Environment:**
    - Create the environment:

```
None
```

        - `python3 -m venv venv`

    - **Activate on macOS / Linux:**

```
None
```

        - `source venv/bin/activate`

    - **Activate on Windows:**

```
None
```

        - `venv\Scripts\activate`

2. *Your terminal should now show the `(venv)` prefix.*
3. **Install Dependencies:** Install all required packages:

```
None
   4. pip install -r requirements.txt
```

5. **Required Packages:** `pybricksdev`, `grpcio`, `fastavro`, `aiohttp`, `requests`.

Refer to the public GitHub repository for project files: LegoWorkshop-----2. Salesforce Configuration2.1. Enable Agent Features

1. In Salesforce Setup, Quick Find **Einstein Setup**.
2. Toggle the switch to **Turn on Einstein** (if not already enabled).
3. Quick Find **Agentforce Agents**.
4. Toggle the switch to **Turn on Agentforce**.
5. Ensure the **Agentforce (Default) Agent** is enabled.

2.2. Create Prompt Template for Classification

Create a **Flex** type prompt template that the agent will use to classify incoming service cases.

1. In Salesforce Setup, search for **Prompt Builder**.
2. Click **New Prompt Template**.
3. Set the **Name**: `Support Classify the case`.
4. Set the **API Name**: `Support_Classify_the_case`.
5. Select **Prompt Template Type**: `Flex`.
6. Paste the following content into the template:

```
None
Analyze the provided Case Description and any associated
   notes, then classify the required support as one of the
   following three types: "Inspection," "Normal," or
   "Contaminated."


Classification Definitions:


Inspection:

Required when the reported issue is ambiguous, relates to site
   access/layout, involves a persistent service failure without
   clear cause, or requires a safety assessment before normal
   service can proceed.

Examples: Blocked access, container missing (but not confirmed
   stolen), unusual odor source, potential structural damage to
   a collection area.


Normal:

Required when the issue is a standard service request or
   failure that can be resolved by a routine
   crew/driver/dispatcher action, and does not involve
   potentially hazardous or prohibited waste.

Examples: Missed scheduled pickup, container replacement due
```

```
   to normal wear and tear, requesting a new service/bin
   delivery, billing inquiry.


Contaminated:

Required when the case explicitly mentions or strongly
   suggests the presence of prohibited, hazardous, or
   non-compliant material in a bin/load that requires special
   handling, disposal, or regulatory reporting.

Examples: Report of chemicals/batteries/paint in a recycling
   bin, medical waste found, asbestos suspicion, fire residue
   in the container.


Input Case Description: {\\!$Input:caseobj.Description}


Addition Case information:
   {\\!$RecordSnapshot:caseobj.Snapshot}


7. Output: Classification: [Inspection | Normal | Contaminated]
```

8. Click **Save**.

2.3. Create Custom Case Field

Create a custom picklist field on the Case object to store the classification result.

1. Navigate to **Object Manager** > **Case** > **Fields & Relationships**.
2. Click **New** and select **Picklist** for the Data Type.
3. **Field Label**: `Support Classification`.
4. **Field Name**: `Support_Classification`.
5. Enter the following values (each on a new line):
    - `RECYCLING_OK`

- `CONTAMINATED`
- `INSPECTION`

6. Set field-level security and add the field to the necessary page layouts.
7. Click **Save**.

2.4. Define Platform Events

Set up the events used for communication between Salesforce and the Rover.

Event Name	Field	Purpose
`LEGO_Command__e`	`Command__c`	`RECYCLING_OK`, `CONTAMINATED`, or `INSPECTION` command for the Rover.
	`Case_Id__c`	The originating Salesforce Case ID.
`LEGO_Robot_Status__e`	`Case_Id__c`	Echoed ID from the command.
	`Message__c`	Status update message (e.g., “Target Zone Reached”).

2.5. Create Flow: Publish Platform Event

This flow is a simple automation that triggers an event when a case is classified as contaminated.

1. In Salesforce Setup, search for **Flow**.
2. Click **New Flow** and select **Record-Triggered Flow**.
3. **Object**: `Case`.
4. **Trigger the Flow When**: A record is created or updated.

5. **Entry Conditions**: `Support_Classification__c` equals `'CONTAMINATED'`.
6. **Optimize the Flow For**: Actions and Related Records (After Save).
7. Add an **Action** element.
8. **Action Type**: Platform Event > `Rover_Action__e` (assumes this event exists).
9. **Set Field Values**:
    - `Action_Type__c = 'Contamination_Detected'`
    - `Case_ID__c = Case.Id`
10. **Label**: Publish Contamination Event.
11. Click **Save** and **Activate**.

2.6. Create Employee Agent

Create a dedicated agent to handle employee inquiries about the Rover's status and usage.

1. In Salesforce Setup, search for **Agentforce Agent**.
2. Click **Open in Builder** for **Einstein Copilot**.
3. Create a **New Topic**:
    - **Name**: `Employee Rover Guide`.
    - **Classification Description**: Provides guidance to employees about the Rover, including status checks and how to report issues.
    - **Scope**: Only answer questions about the Rover's operational status and guide employees on case creation.
    - **Instructions**: Use the `Support_Classify_the_case` template for all status inquiries. Always ask the employee for the device ID.
4. Add the **Action**:
    - Click **New Action**.
    - **Action Name**: `Check_Rover_Health`.
5. Link the `Support_Classify_the_case` template to this Topic.
6. Click **Save** and **Activate** the agent.

2.7. Configure Connected App (External Client App)

1. Navigate to **External Client App Manager** in the Setup.

2. **External Client App Name**: `Lego Agent Bridge`.
3. Add a **Contact Email**.
4. **Distribution State**: `Local`.
5. Enable OAuth Settings:
    - **Callback URL**: `http://localhost:8080/callback`
    - **Scopes**: `api`, `full`, `refresh_token`, `offline_access`, `sfap_api`.
6. **Uncheck** all checkboxes in the **Flow Enablement and Security** section.
7. Retrieve the **Consumer Key** and **Secret** from the External App console (requires email verification).

-----3. Rover Configuration and Calibration3.1. Rover Hardware

The Rover is composed of:

- **Main Processor/Controller**: The hub running the Python scripts.
- **Motor/Actuators**: Responsible for movement and the sorting arm.
- **Distance Sensor (Port F)**: Detects objects, allowing the Rover to stop before a barrier.
- **Color Sensor (Port B)**: Mounted on the grabber arm to detect colored sorting zones.

3.2. Run Color Calibration

Calibrate the color sensor to your environment's lighting conditions.

1. Run the calibration script:

```
None
   2. pybricksdev run ble color_calibrater.py
```

3. Note the most frequent Reflection (`REF`) values for the RED, GREEN, and YELLOW zones.
4. Update the calibration values in the `contamination_sorter.py` script:

```
Python
   # --- COLOR CALIBRATION VALUES (EDIT HERE) ---
```

```
CAL_RED    = 5    # Replace with your measured REF value
CAL_GREEN  = 11   # Replace with your measured REF value
5. CAL_YELLOW = 13.5 # Replace with your measured REF value
```

-----4. System Operation and Testing4.1. Establish Connectivity

1. **Configure Client Credentials:** Create an `sf_config.json` file in the root directory and paste your credentials:

```
None
{
  "CLIENT_ID" : "**Your Client ID**",
  "CLIENT_SECRET" : "**Your Client Secret**"
2. }
```

3. **Run Salesforce Login Script:** This script handles OAuth login and stores the access token in `sf_token_info.json`.

```
None
4. python sf_login.py
```

4.2. Run the System

1. **Start the Bridge:** Start the Python Bridge to begin listening for Salesforce commands.

```
None
2. python bridge_pybricks.py
```

3. **Test with Anonymous Apex (Manual Command):** Use the Salesforce Debug Anonymous (Apex) window to send a test event. Replace `**Your SF CASE ID**` and choose a valid command.

```
None
LEGO_Command__e eventRecord = new LEGO_Command__e(
```

```
    Command__c = 'INSPECTION', // Choose one: INSPECTION,
  RECYCLING_OK, CONTAMINATED
    Case_Id__c = '**Your SF CASE ID**'
);

Database.SaveResult sr = EventBus.publish(eventRecord);
4. System.debug('SR: ' + sr);
```

4.3. End-to-End Testing Scenarios

Create the following service cases in the Service Cloud to test the entire flow: Agent classification → Flow trigger → Platform Event → Python Bridge → Rover action.

Scenario	Case Subject	Case Description Focus	Expected Rover Action	Classification
Contamination	Prohibited Waste / Hazardous Material Alert	Mentions "chemical warning labels" and "hazmat team notification."	Moves to Red zone	`CONTAMINATED`
Recycling_OK	Missed Collection (Residential)	"Issue is a standard missed pickup requiring a make-up collection." (Maps to Normal in classification definitions)	Moves to Green zone	`RECYCLING_OK`
Inspection	Access Problem / Unidentified	Mentions "newly installed	Moves to Yellow	`INSPECTION`

	Debris	chain-link fence" and "Needs a supervisor site inspection."	zone	

Test Employee Agent:

1. Open the **Service Console**.
2. Start a new conversation with the Employee Agent.
3. **Test Utterance**: "What is the status of Rover ID Person?"
4. Verify the agent uses the custom prompt template and references the (simulated) status.

-----Project Files

- `bridge_pybricks.py`
- `salesforce_pubsub.py`
- `contamination_sorter.py`
- `color_calibrater.py`
- `sf_login.py`
- `sf_config.json` (local only)
- `robot_board.png`
- `sensor_placement.png`

Initial Setup:

Pybricks Firmware Installation (already done)

Steps:Go to https://code.pybricks.com

- Click the Bluetooth icon, select your hub, and install
- If Bluetooth scanning doesn't work:
    - Go to `chrome://flags`
    - Enable Experimental Web Platform Features
- If Bluetooth still fails:
    - Hold hub Bluetooth button
    - Plug USB cable into laptop (flashing pink → green → blue)
- On Windows 11, you might need the WinUSB driver via Pybricks troubleshooting wizard

Laptop Setup

Create & Activate Virtual Environment

```
python3 -m venv venv
```

macOS / Linux:

```
source venv/bin/activate
```

Windows:

```
venv\Scripts\activate
```

```
Your terminal should now show (venv) prefix
```

```
After that execute the below command:
```

```
pip install -r requirements.txt
```

Install dependencies (Required Packages)

- pybricksdev
- grpcio
- fastavro
- aiohttp
- requests

```
pip install grpcio grpcio-tools fastavro requests pybricksdev bleak
```

Install Salesforce Pub/Sub API Python Stubs

This step is REQUIRED—without it, import pubsub_api_pb2 will fail.

Step A — Clone Salesforce's repo

git clone https://github.com/salesforce/pub-sub-api.gitcd pub-sub-api

Step B — Generate Python gRPC stubs

Inside the cloned repo:

```
python -m grpc_tools.protoc \
    -I proto/v1 \
    --python_out=../LegoWorkshop \
    --grpc_python_out=../LegoWorkshop \
    proto/v1/pubsub_api.proto
```

This generates:

* pubsub_api_pb2.py

* pubsub_api_pb2_grpc.pydirectly inside your LegoWorkshop folder.

These files are imported by your working code:

import pubsub_api_pb2 as pubsubimport pubsub_api_pb2_grpc as pubsub_grpc

Streaming monitor installation:
packagingSetupUI/ipLanding.app?apvId=04tJ5000000gQFx