Below is a **completely new real- world Zapier Webhooks implementation** (different from all prior prompts). It uses **Salesforce Platform Events** as the **Trigger** and **SendGrid's REST API** as the **Action**—a common enterprise pattern for real- time customer engagement.

✅Real- World Zapier Webhook Implementation

Trigger + Action (Enterprise CRM → Email Infrastructure)

🧩 Scenario

- **Trigger (Inbound): Salesforce Platform Event** fires when a customer's **account status changes to "At- Risk."** Salesforce sends the event to Zapier via an HTTPS webhook.
- **Action (Outbound):** Zapier uses **Webhooks by Zapier (POST)** to call the **SendGrid API** to send a **priority alert email** to the Customer Success team.

This pattern is used by SaaS and enterprise teams for **real- time alerts** without polling APIs or building custom middleware.

1️⃣ Webhook as Trigger

Salesforce Platform Events → Zapier

Why Platform Events?

- Event- driven (push- based)
- Decoupled from UI changes
- Guarantees delivery and retries on failure
- Ideal for status changes and alerts

A. Create the Zap Trigger

1. Zapier → **Create Zap**
2. Trigger App: **Webhooks by Zapier**
3. Trigger Event: **Catch Hook**
4. Zapier generates a webhook URL:

B. Salesforce: Configure the Platform Event

Create a **Platform Event** in Salesforce:

- Event Name: Account_Risk_Change__e
- Fields:
    - Account_Id__c
    - Account_Name__c
    - Risk_Level__c
    - Owner_Email__c

C. Salesforce Flow → Call Zapier Webhook

Create a **Record- Triggered Flow**:

- Object: Account
- Condition: Risk_Level__c = "At-Risk"
- Action: **HTTP Callout**

Endpoint (Zapier Webhook URL):

```
POST https://hooks.zapier.com/hooks/catch/987654/xyzabc
```

Payload Sent from Salesforce

```
{
  "event_type": "ACCOUNT_RISK_UPDATE",
  "account_id": "{!Account.Id}",
  "account_name": "{!Account.Name}",
  "risk_level": "{!Account.Risk_Level__c}",
  "owner_email": "{!Account.Owner.Email}",
  "timestamp": "{!$Flow.CurrentDateTime}"
}
```

✅Salesforce pushes the event → Zapier instantly triggers.

D. Zapier Receives & Parses Data

Zapier auto- parses:

- account_name
- risk_level
- owner_email
- timestamp

(Optional) Add a **Filter**:

```
Only continue if risk_level = "At-Risk"
```

✅Webhook Trigger complete.

2□□ Webhook as Action

Zapier → SendGrid API (Send Alert Email)

Now Zapier will **POST** to SendGrid's REST API to notify Customer Success.

A. Add Action Step

- Action App: **Webhooks by Zapier**
- Event: **POST**

B. SendGrid Endpoint

```
POST https://api.sendgrid.com/v3/mail/send
```

C. Headers Configuration

```
Authorization: Bearer {{SENDGRID_API_KEY}}
Content-Type: application/json
```

D. JSON Payload (Mapped from Salesforce Webhook)

```
{
  "personalizations": [
    {
      "to": [
        {
          "email": "cs-alerts@company.com"
        }
      ],
      "subject": "□ Account At Risk: {{account_name}}"
    }
  ],
  "from": {
    "email": "alerts@company.com",
    "name": "CRM Alert System"
  },
  "content": [
    {
```

```
        "type": "text/plain",
        "value": "Account {{account_name}} has changed status to
AT-RISK.\n\nOwner: {{owner_email}}\nTime:
{{timestamp}}\n\nImmediate follow-up required."
      }
    ]
  }
```

Zapier sends this **server-to-server** with no plugins or UI actions.

✅ Email delivered instantly.

🔁 End-to-End Flow

```
Salesforce Account Updated
          ↓
Platform Event Fired
          ↓
Salesforce Flow HTTP Callout
          ↓
Zapier Catch Hook (Trigger)
          ↓
Zapier Filter / Formatter
          ↓
 Zapier POST → SendGrid API
          ↓
 Customer Success Alert Email
```

✅ Why This Is a True Real-World Use Case

- Used by **enterprise Salesforce customers**
- No polling, no schedulers
- Works even when SendGrid has no direct Zap step
- SOC-friendly (API keys, HTTPS only)
- Extensible to Slack, Teams, PagerDuty later

🧠 Best Practices

- Store SENDGRID_API_KEY in **Zapier Environment Variables**
- Make Salesforce Flow **fault-tolerant**

- Ensure SendGrid API returns **202 Accepted**
- Add Zap history monitoring for retries

⁉ Variations You Can Apply

- Replace SendGrid with **AWS SES**
- Fan- out to **PagerDuty API**
- Write to **Data Warehouse API**
- Add **ML risk scoring service** in the middle
