A CASE STUDY ON ADOBE ANALYTICS

TITLE: - ADOBE ANALYTICS IMPLEMENTATION FRAMEWORK: A THEORETICAL CASE STUDY

NAME: - MEDA SAI VRISHYANG KUMAR

DATE: - 26/08/2025

TOOLS: - ADOBE ANAYTICS WORKSPACE, ADOBE EXPERIENCE PLATFORM TAGS (ADOBE LAUNCH), ADOBE EXPERIENCE CLOUD DEBUGGER

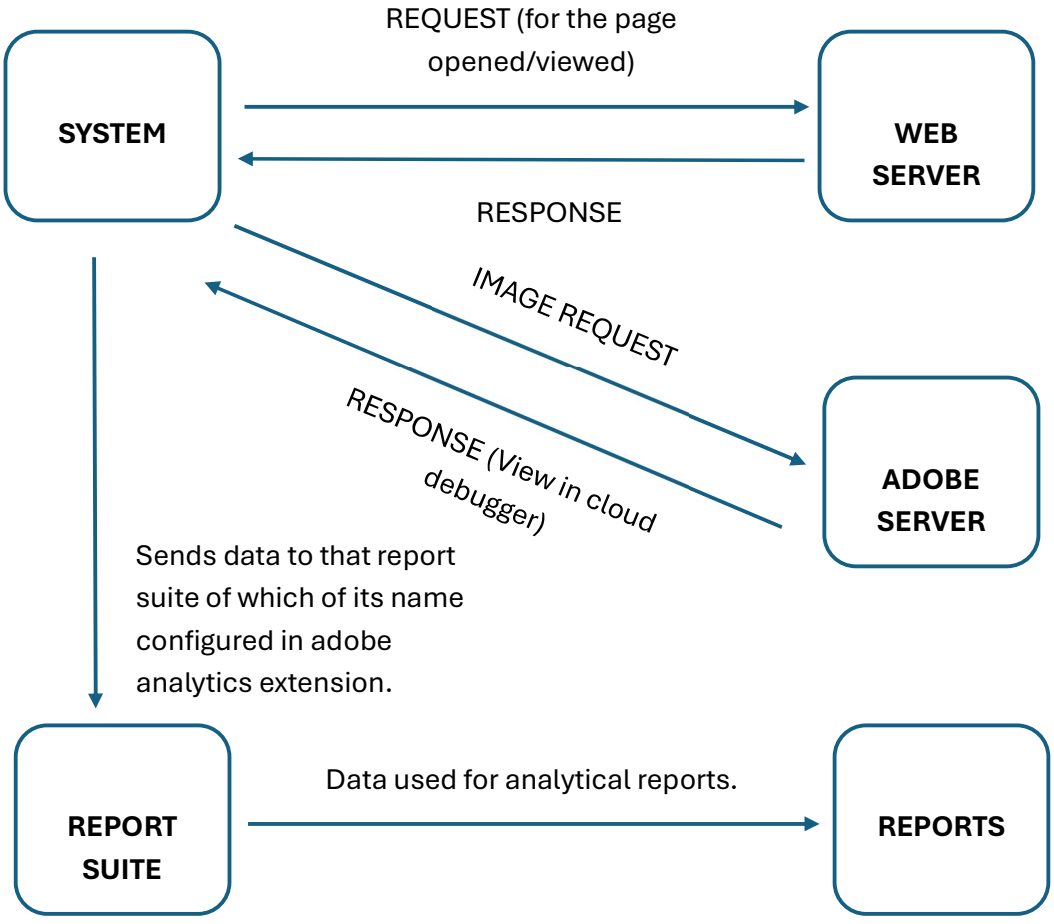
SUMMARY

Adobe Analytics is a digital analytics platform that helps organizations measure, analyse, and optimize user interactions across websites and applications. It provides businesses with actionable insights into customer behaviour, enabling data-driven decisions and improved digital experiences.

The process begins with the **implementation stage**, where Adobe Launch (Adobe's tag management system) is installed on a website. Within Launch, rules and data elements are configured and then published to determine how user activity is captured. Once a page is configured in this way, every time it loads or a tracked interaction occurs, an **image request is sent to the Adobe server**. This request carries the information set in the configured variables and events.

To ensure accuracy, this data transfer is validated through tools such as the **Adobe Experience Cloud Debugger** and the browser's Inspect console. These allow practitioners to check whether the correct variables and events are being recorded as intended. After validation, the captured data becomes available in **Analysis Workspace**, where analysts can create dashboards, visualizations, and segments to derive insights.

In summary, Adobe Analytics works through a cycle of **implementation (Launch setup) → validation (Debugger/Inspect) → analysis (Workspace)**. This flow ensures that user interactions on a website are properly tracked, verified, and transformed into meaningful insights for decision-making.



ADOBE ANALYTICS WORKSPACE

Adobe Analytics Workspace (Analysis Workspace) is the central interface where data collected through Adobe Analytics is explored, analysed, and visualized. It is designed as a flexible, drag-and-drop canvas that enables analysts and marketers to build customized dashboards and reports without the need for coding.

How to Access Workspace

Workspace can be accessed by logging into the **Adobe Experience Cloud** and selecting **Analytics** from the available applications. Within Analytics, the **Workspace** tab provides access to existing projects or the option to create a new one.

Key Components of Workspace

1. **Panels**

- Panels act as containers for analysis.
- Each panel can represent a different analysis, dataset, or visualization.
- Examples: Freeform Table Panel, Blank Panel, Segment Comparison Panel, Cohort Panel, etc.

2. **Visualizations**

Workspace provides a wide range of visualization options to help interpret data effectively. Common visualization types include:

- **Freeform Tables** (the foundation of Workspace; allow drag-and-drop of metrics, dimensions, and segments).
- **Line, Bar, and Donut Charts** for trend and distribution analysis.
- **Maps** for geographic analysis.
- **Flow Visualizations** to understand paths users take through a site.
- **Fallout Reports** to analyse conversion funnels.
- **Summary Number & Text Visualizations** for KPIs or quick highlights.

These visualizations can be combined in a single project to create a dashboard-style view tailored to business needs.

3. **Segments**

- Segments allow filtering and breaking down data into smaller groups (e.g., mobile visitors, first-time users, or purchasers).
- Segments can be dragged directly onto a table or visualization to apply them instantly.
- They can also be created or modified directly within Workspace using the segment builder.

4. **Date Ranges**

- Workspace offers flexible date range controls.
- Analysts can use pre-built ranges like Today, Last 7 Days, Last Month, or create custom rolling/fixed date ranges (e.g., Last 30 Days, This Quarter).

- Multiple date ranges can be applied within the same project for comparison purposes.

5. Dimensions and Metrics

- **Dimensions** represent qualitative attributes of data, such as page name, traffic source, device type, or campaign ID.
- **Metrics** represent quantitative values, such as page views, visits, revenue, or conversion rate.
- Both dimensions and metrics are accessible from the left-hand rail in Workspace and can be dragged into tables and visualizations for analysis.

In addition to its core features, Adobe Analytics Workspace also offers powerful options such as **calculated metrics** and **breakdowns**. Calculated metrics allow analysts to build custom formulas, like conversion rates or engagement ratios, tailored to specific business needs. Breakdowns, on the other hand, make it possible to drill deeper into data by stacking dimensions (for example, analysing Page Views → broken down by Device Type → further broken down by Geography). These features make Workspace not just a reporting tool, but also an environment for exploration and deeper analysis.

Workspace also enhances collaboration and usability through **project sharing, export options, and interactivity**. Dashboards can be shared with stakeholders or exported in PDF/CSV formats for reporting. Interactive elements, such as drag-and-drop filters, anomaly detection, and contribution analysis, help uncover insights quickly and explain unusual patterns in data. Together, these capabilities make Workspace a flexible and comprehensive platform for turning raw tracking data into meaningful business insights

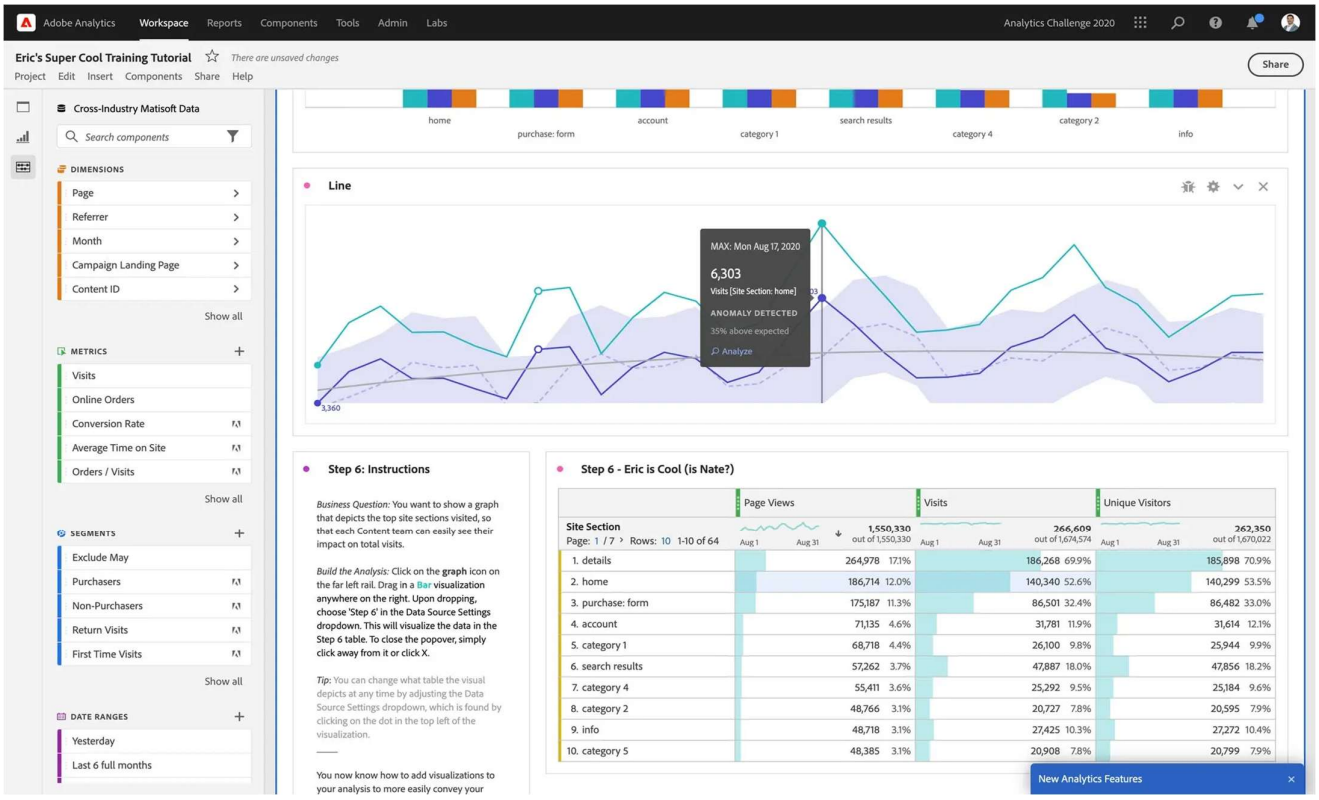


Fig: Adobe Analytics Workspace

VARIABLES

In Adobe Analytics, **variables** are the fundamental components used to capture and store data about user interactions on a website or application. They define what information is collected (such as page name, campaign ID, or revenue) and how it should be processed for reporting. Variables are configured in the **Report Suite Manager** and include different types like **eVars (conversion variables), props (traffic variables), and events (success metrics)**. Together, these variables form the foundation for accurate tracking and meaningful analysis. In Adobe Launch, this function retrieves the current value of a defined **Data Element** at runtime. It helps reference stored values like page name, user ID, or campaign code inside custom scripts or rules.

Example: var variable_name/s.variable_name (pre-defined variable) = _satellite.getVar("Data Element");

1. Pre-Defined Variables

Adobe provides several built-in variables for consistent data capture. Below are the key ones with examples:

- **s.pageName** → Stores the page title or identifier.
  Example: s.pageName = "Home:Electronics:Mobiles"
- **s.channel** → Groups pages into site sections for aggregated reporting.
  Example: s.channel = "Electronics"
- **s.server** → Captures server/domain of the page.
  Example: s.server = "www.mystore.com"
- **s.pageType** → Classifies page type (used mainly for error pages).
  Example: s.pageType = "errorPage"
- **s.campaign** → Tracks marketing campaigns through tracking codes.
  Example: s.campaign = "SUMMER2025_SALE_EMAIL"
- **s.purchaseID** → Unique order ID to avoid duplicate transactions.
  Example: s.purchaseID = "ORD12345"
- **s.transactionID** → Secondary transaction reference for reconciliation.
  Example: s.transactionID = "TXN98765"
- **s.currencyCode** → Captures currency for accurate revenue reporting.
  Example: s.currencyCode = "INR"
- **s.products** → Stores product details in a structured string.
  Format:
  Category;Product;Quantity;Price;Events (Increementors);eVars (Merchandising)
  Example:
  s.products =
  "Electronics;Mobile_XYZ;2;15000;event1=2|event2=30000;eVar1=Samsung|eVar2=Merchandise"
  - **Incrementors**: events like event1 (units) automatically add counts, can also be numeric or currency metrics.
  - **Merchandise eVars**: product-level breakdown variables (e.g., brand).
- **s.state** → Captures visitor's state/province.
  Example: s.state = "Telangana"

- **s.zip** → Stores visitor postal/ZIP code.
  Example: s.zip = "500081"
- **s.events** → Holds success events triggered during the visit.
  Example: s.events = "event1, event2"

2. Success / Standard Events

Events measure **user actions** and business outcomes. They populate into reports as metrics.

- **prodView** → Product detail viewed.
  Example: s.events = "prodView"
- **scAdd** → Product added to cart.
  Example: s.events = "scAdd"
- **scOpen** → Shopping cart opened.
  Example: s.events = "scOpen"
- **scCheckout** → Checkout process initiated.
  Example: s.events = "scCheckout"
- **purchase** → Successful order completed.
  Example: s.events = "purchase"
- **scRemove** → Product removed from cart.
  Example: s.events = "scRemove"

3. eVars (Conversion Variables)

Purpose:
eVars are **conversion variables (persistent)** used to track and attribute values that contribute to success events (like purchases, sign-ups, or downloads). Unlike props (traffic variables), eVars persist beyond a single page view depending on their expiry settings, making them essential for tracking attribution over a session or across visits.

Configuration (Report Suite Manager → Edit Settings → Conversion → Conversion Variables):

- **Name & Friendly Name:** Assign a business-relevant name (e.g., Campaign ID, Product Category).
- **Enabled:** Activate the eVar for use.

Key Settings:

- **Allocation:**
    - Most Recent (last value gets credit for the conversion).
    - Original Value (first captured value holds credit).
    - Linear (credit split equally among all values encountered).
- **Expiration:**
    - Defines how long the eVar holds its value.
    - Options include Page View, Visit, Purchase, or a Time Period (e.g., 30 days).
- **Type:**
    - Text String/Type String (most common).

    - Counter (increments a count for each hit).
    - Numeric (captures numbers).
    - Currency (captures monetary values).
- **Merchandising eVars:**
    - Special type tied to product tracking.
    - Can be set to "Product Syntax" or "Conversion Syntax," allowing credit to be tied directly to individual products.
- **List Variables (if enabled for eVars):**
    - Allow multiple values to be stored in one variable (e.g., multiple search keywords).

4. Props (Traffic Variables)

Purpose:
Props (short for traffic variables) are used for **page-level or hit-level tracking (non-persistent)**. They capture values that do not persist beyond the page view unless explicitly repeated. Props are ideal for analysing traffic paths, counts, and correlations between variables.

Configuration (Report Suite Manager → Edit Settings → Traffic → Traffic Variables):

- **Name & Friendly Name:** Assign to reflect use cases (e.g., Page Name, Author Name, Device Type).
- **Enabled:** Activate the prop.

Key Settings:

- **List Props:**
    - Allow multiple values in a single prop separated by a delimiter (e.g., categories tagged on an article).
- **Pathing (Path Reports):**
    - Can be enabled for props to analyze user navigation paths (Next Page, Previous Page reports).
    - Commonly used for "Page Name" or "Site Section."
- **Character Limit:**
    - Props generally have shorter character limits than eVars (100 characters typically).

5. Events (Success Events / Custom Events)

Purpose:
Events are the **metrics** that measure key interactions or outcomes on a site, such as purchases, clicks, video views, or downloads. They represent the "what happened" part of the tracking.

Configuration (Report Suite Manager → Edit Settings → Conversion → Success Events):

- **Name & Friendly Name:** Define the event clearly (e.g., Purchase, Form Submission, Video Start).
- **Enabled:** Activate the event.

Event Types:

- **Counter Events:** Increment a count (e.g., number of downloads).
- **Numeric Events:** Record a number (e.g., time spent on video, score).
- **Currency Events:** Record monetary value (e.g., revenue).

Key Settings:

- **Event Serialization:** Prevents duplicate event counting if the same event fires multiple times (e.g., refreshing an order confirmation page).
- **Success Event Allocation:** Determines how eVars (conversion variables) get credit when this event fires.
- **Merchandising Events:** Used when tied with merchandising eVars for product-level tracking.

6. Variable Scopes

- **Hit Scope** → Value applies only to the current server call (e.g., s.pageName).
- **Visit Scope** → Value persists for the entire visit/session (e.g., campaign eVar).
- **Visitor Scope** → Value persists across visits until expiry (e.g., customer ID).

7. Real-World Usage Tips

- Use hierarchical naming for **pageName** and **channel** to simplify pathing.
- Use hierarchical naming for **js variables** (objectname.section/fn name.variable name) while creating a data element for these variables.
- Combine **campaign** and **eVar** tracking for deeper attribution.
- Validate **purchaseID** uniqueness — duplicate IDs inflate revenue.
- Always map **zip/state** data with caution (privacy regulations may apply).

8. Variable Processing Order

- Data goes **from Launch → Analytics call (s object) → Processing Rules → Reporting**.
- Knowing this helps debug when values don't appear as expected.

9. Variable Character Limits

- **Page Name / Props / eVars** → Typically 255 characters.
- **Products String** → 64k characters (but formatting errors often cause truncation).
- Keeping values short & clean prevents processing issues.

10. Case Sensitivity

- Variables like eVars and props are **case-sensitive** ("MOBILE" ≠ "mobile").
- Standardize casing at implementation level to avoid duplicate reporting.

ADOBE EXPERIENCE PLATFORM TAGS

Adobe Experience Platform Tags, formerly known as Adobe Launch, is the **tag management system (TMS)** that controls how analytics and marketing tags are deployed on websites or apps. It ensures that data is collected accurately and delivered to Adobe Analytics (or other tools) without manual hard-coding each time.

1. Account

- The **account level** is the top hierarchy tied to your Adobe Experience Cloud organization.
- Access is managed through **Adobe Admin Console**, where users are assigned product profiles and permissions.
- Within an account, multiple **Properties** can be created, each representing a website, app, or digital property.

2. Property

- A **Property** is the main container for managing tags for a website or app.
- Inside a property, you define **Data Elements, Rules, Extensions, Libraries, and Environments**.
- **Fields in a Property setup:**
    - **Name:** Friendly name for the property (e.g., Corporate Website).
    - **Domains:** The domains or subdomains where this property will run.
    - **Platform:** Web or Mobile (depending on implementation).
    - **Published Libraries:** Displays the libraries currently published to each environment.

3. Environments

Environments define where the tag library is deployed. Each property has three main environments: **Development** (for testing changes), **Staging** (for QA verification), and **Production** (for live site).

Each environment provides:

- **Name & Type** (Dev, QA, Prod)
- **Embed Code (Script URL):** Unique JavaScript snippet to place on the site
- **Status & Last Published Library** to track deployment

This setup ensures safe testing before publishing tags to the live site. After this, one needs to **install** the launch in the website by adding the **installation code** in the website (mostly in <body> tag) following the instructions and only then the **rules/events data** can be recorded.

4. Libraries

A **Library** is a package of changes (rules, data elements, extensions) that you prepare and publish to an environment.

Fields inside a Library:

- **Name:** A meaningful label (e.g., Login Tracking Update).
- **Environment:** Select which environment (Dev, Staging, Production) this library will be built for.

- **Resources Included:** List of Rules, Data Elements, and Extensions added to the library.
- **Status:** Shows progress in publishing workflow (Not Submitted, Submitted, Approved, Published).

Libraries allow you to bundle multiple changes together and control when they go live.

5. Extensions

Extensions are pre-built code packages that add functionality to your property. They can be thought of as "plug-ins."

By default, every property contains:

Core Extension

- This comes pre-installed.
- Includes essential actions, events, and conditions like Page Load, Clicks, Custom Code, Set Variables, Send Beacon, Cookie handling.

Adobe Analytics Extension

- Used to connect your website implementation with **Adobe Analytics Workspace and specific report suites**.
- **Fields inside Analytics Extension:**
  - **Report Suites:** Select report suite(s) where data will be sent (dev, staging, prod report suites).
  - **Tracking Server / Secure Tracking Server:** Defines Adobe servers where data is sent.
  - **Global Variables:** Set global values for eVars, props, events, and other variables.
  - **Link Tracking:** Enable automatic link tracking for downloads, exits, or custom link clicks.
  - **Config Options:** Choose how/when Analytics beacons are fired.

(Other extensions like Target, AEP Web SDK, or third-party tools can be added, but I'll keep the focus on **Core + Analytics** for the case study.)

6. Publishing Workflow

Publishing is the process of **moving changes live**. After every update (rule creation, data element addition, or extension changes), you must add those changes to a library and publish.

Steps in Publishing Workflow:

1. **Create or Update a Library**
   - Add your changes (rules, data elements, extensions).
   - Assign the library to an environment (usually Development first).
2. **Build for Development**
   - Launch generates the tag library for Dev environment.
   - Test changes on the site using the Dev embed code.
3. **Submit for Approval**
   - After testing, submit the library for review.

4. **Build for Staging**
    - Push the same library to the QA environment for broader testing.
5. **Publish to Production**
    - Once approved, publish the library to the Production environment, making changes live.

Important Note: Every time a new change is made (new data element, updated rule, or added extension), the publishing workflow must be **repeated** to ensure the changes are deployed correctly.

7. Rules

Rules define **when and how actions are executed** on your site. Each Rule consists of **Events, Conditions, and Actions**.

Components of a Rule:

1. **Name** – Unique identifier for the rule.
2. **Events** – What triggers the rule.
    - **Event Type**: Page Load, Click, Form Submit, Custom Event, etc.
    - **Selector:** CSS selector to target specific elements (optional).
    - **Delay / Throttle:** Delay before execution or limit repeated firing.
    - **Scope:** Page, Library Loaded, or Custom (defines how broadly the event listens).
3. **Conditions** – Optional checks to determine if Actions should run.
    - **Type**: Data Element, JavaScript, Cookie, Local/Session Storage, URL Parameter, Environment/Library.
    - **Operator:** Equals, Not Equals, Contains, Regex, Greater/Less Than.
    - **Value:** Value to compare against (e.g., "beginner").
    - **Scope:** Page (per page view), Event (specific trigger), Library (on library load), Global/Persistent (across sessions).
    - **Logical Combinations:** AND, OR, nested logic allowed.
    - **Optional:** Negate (NOT), case sensitivity, evaluation delay.
4. **Actions** – What happens when the Rule fires and conditions pass.
    - **Extension**: Core/Custom Code, Adobe Analytics, or other installed extensions.
    - **Action Type:**
        - **Adobe Analytics:** Set Variables, Send Beacon, Clear Variables.
        - **Custom Code:** Execute JS.
        - **Other Extensions:** Trigger pixels, third-party scripts, A/B tests, etc.
    - **Configuration / Parameters:** Variables or fields specific to the action (e.g., eVars, props, events for Analytics).
    - **Order / Sequencing:** Rules run in library order; priority can be defined.
    - **Environment Assignment:** Decide which environment(s) the rule is published to (Dev, Staging, Prod).

Key Notes:

- Rules are **modular**; can reference **Data Elements** in Events, Conditions, or Actions.

- Actions fire **only if all conditions are satisfied**.
- Reusable logic ensures consistency and maintainability across the site.

8. Data Elements

Data Elements are variables in Adobe Launch that **store values** used across Rules and Extensions. They are essentially placeholders for dynamic or static data.

Structure of a Data Element:

1. **Name** – The identifier used to reference this element in rules and extensions.
2. **Extension Type** – Most are standard (Core) but can also be provided by other extensions.
3. **Data Element Type / Source** – Determines how the value is retrieved:
    - **JavaScript Variable** – Pulls a value from a global JS variable.
    - **Cookie** – Reads a browser cookie.
    - **Local Storage / Session Storage** – Pulls from localStorage or sessionStorage.
    - **URL Parameter** – Extracts value from query string.
    - **DOM Element** – Gets value from a specific HTML element.
    - **Custom Script** – Executes custom JS to return a value.
4. **Path / Selector / Key** – Depending on type:
    - JS Variable → e.g., window.userType
    - Cookie → e.g., user_session
    - DOM Element → CSS selector to target the element
    - URL Parameter → parameter name, e.g., utm_source
5. **Default Value** – Optional fallback value if nothing is found.
6. **Persistence / Lifetime** – Defines how long the value should be kept:
    - Page View (reset every page)
    - Session
    - Visitor (across sessions via cookie)
7. **Scope / Environment** – Can be limited to certain library environments.

Key Notes:

- **Data Elements are reusable** across multiple Rules.
- They are **read-only during rule execution**; to set a new value, you typically use Actions in a Rule.
- Common use cases:
    - Capturing user type, product ID, form values and tracking URL parameters for campaigns.
    - Reading login state from cookies.

ADOBE CLOUD DEBUGGER

A real-time tool to **inspect and troubleshoot Adobe Launch implementations**, Analytics, and other Adobe Experience Cloud tags.

Key Components:

1. **Library / Environment** – Choose Dev, Staging, or Prod library to debug.
2. **Event Stream** – Shows all page events (Page Load, Click, Custom), timestamped, with triggering rules.
3. **Rule Execution** – Lists rules that fired or were evaluated:
    - Name, triggering events, conditions (pass/fail), executed actions.
4. **Data Elements** – Displays current values for all Data Elements: name, type, source, value.
5. **Network / Beacons** – Shows outgoing requests (Analytics, Launch, etc.), variables, and status.
6. **Errors & Warnings** – Real-time logs for misconfigured rules, missing Data Elements, or JS errors.
7. **Search & Filter** – Filter events by type, rule, or Data Element for easy troubleshooting.
8. **Real-Time Updates** – View changes immediately without multiple pages reloads.

Notes:

- Observational only; does not modify the site.
- Works with Adobe Analytics, Target, Audience Manager, and Launch extensions.

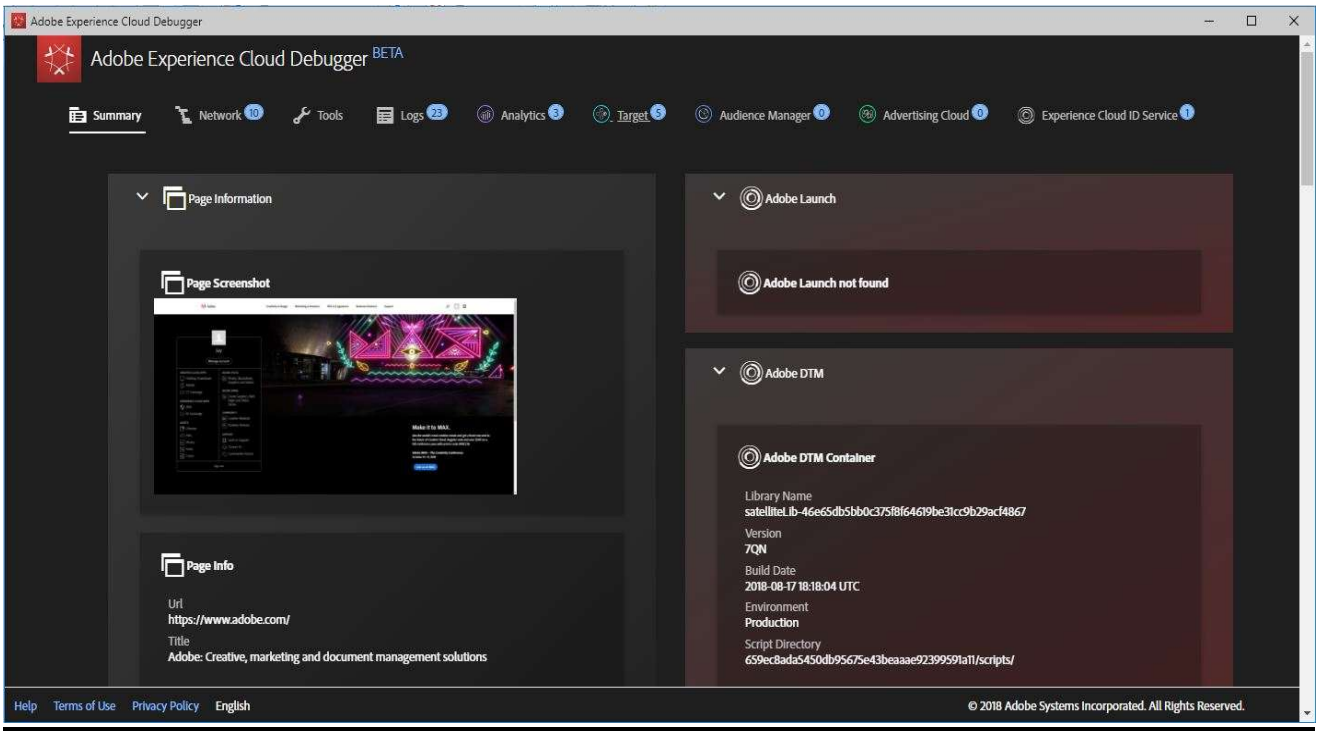


Fig: Adobe Experience Cloud Debugger

SIMILARITIES WITH GOOGLE ANALYTICS

1. Adobe Analytics Workspace vs Google Analytics (GA4)

Feature / Aspect	Adobe Analytics Workspace	Google Analytics (GA4)	Similarity
Data Collection	Uses report suites and variables (eVars, props, events)	Uses data streams and parameters	Both collect structured user interaction data and events.
Event-Based Tracking	All interactions can be events; can map to custom variables	All interactions are events (enhanced measurement + custom events)	Event-driven model in both; flexible custom tracking.
Segmentation / Filters	Segments can be applied to analysis	Audiences or segments in reports	Both allow slicing data based on user behavior or attributes.
Visualization / Analysis	Drag-and-drop Workspace panels, tables, visualizations	Exploration in GA4: tables, charts, funnels	Both provide flexible drag-drop analysis and visualizations.
Real-Time Reporting	Limited real-time reports; mostly near real-time	Real-time reports available	Both provide some level of immediate insight.
User / Session Tracking	Visitor ID, custom variables for user/session	User ID, session ID	Both track individual users and sessions for behaviour analysis.
Custom Metrics / Calculated Metrics	Yes, via Calculated Metrics	Yes, via Custom Metrics & Formulas	Both allow creating derived metrics from raw data.

Takeaway: Both are **event-driven analytics platforms**, allow **custom metrics, segmentation, and visualizations**, but Adobe is more enterprise-focused with more granular variables, while GA4 is simpler and more automated.

2. Adobe Launch vs Google Tag Manager (GTM)

Feature / Aspect	Adobe Launch	Google Tag Manager	Similarity
Tag Management	Manages Adobe and third-party tags	Manages GA, Adobe, and third-party tags	Both are tag management systems (TMS) for deploying scripts without touching code.
Rules / Triggers	Rules define Events → Conditions → Actions	Triggers define when Tags fire based on Events & Conditions	Both use event-condition-action logic.
Data Elements / Variables	Data Elements store values from JS, DOM, cookies, URL	Variables store values from JS, DOM, cookies, URL	Both use reusable variables for tag firing and data collection.
Actions / Tags	Actions execute Extensions or Custom Code	Tags execute scripts (GA, Ads, custom JS)	Both can send data to analytics or third-party tools when conditions meet.
Environments / Libraries	Dev, Staging, Prod libraries	Workspaces with Environments (Preview/Publish)	Both allow testing in non-prod environments before going live.
Extensions / Built-in Plugins	Adobe extensions + Custom	GTM templates + Custom HTML	Both provide pre-built integrations and ability to run custom code.
Versioning / Publishing	Library versions, rollback available	Versioned container, rollback available	Both maintain version control of deployments.

Takeaway: Adobe Launch and GTM are conceptually very similar **event-driven, variable-based tag management systems**, with environments, extensions/templates, and publishing controls.

3. Debug View / Preview vs Cloud Debugger

Feature / Aspect	Cloud Debugger (Adobe)	Debug View / Tag Assistant (GTM & GA4)	Similarity
Purpose	Real-time inspection of rules, data elements, and beacons	Real-time inspection of GTM triggers/tags and GA4 events	Both allow live debugging of data collection and tag firing.
Rule / Trigger Evaluation	Shows rules that fired or failed, with conditions & actions	Shows triggers that fired or didn't, tags executed	Both show event-condition-action evaluation.
Variable / Data Elements	Displays all Data Element values on page	Displays all Variables accessible in GTM / GA4	Both allow checking dynamic data before it reaches analytics.
Network Requests / Beacons	Shows Analytics beacons, request parameters, status	Shows GA4 events and tag network requests	Both allow verifying outgoing requests and payloads.
Errors & Warnings	Shows JS errors, misconfigured rules, missing elements	Shows tag errors, misfires, or console warnings	Both help identify implementation issues.
Filtering / Searching	Filter by event, rule, or Data Element	Filter by event, tag, or variable	Both provide tools to quickly locate specific events or variables.
Real-Time Updates	Observe changes immediately without reload	Observe GA4 events and GTM tags in real-time	Both provide instant feedback during testing.

Takeaway: Cloud Debugger and GTM/GA4 Debug View serve **the same purpose**: live troubleshooting, inspecting variables/data, and validating tag or rule firing.

IMPLEMENTATION SCENARIOS

A. Purchase Event

The **Purchase event** is one of the most important tracking points in Adobe Analytics because it captures the final transaction details, enabling revenue, product performance, and customer behaviour analysis.

In this implementation scenario, I configure the **purchase tracking** on the order confirmation page. The website dynamically pushes order, customer, and cart details into a digitalData object, which Adobe Launch then consumes using **data elements** and **rules**. These values are mapped to Adobe Analytics variables (s.products, eVars, props, and events) to record product-level and transaction-level metrics.

This scenario covers the **full workflow**:

1. **Website Code** → pushes dynamic purchase details.
2. **Adobe Launch** → creates data elements, defines rules, applies conditions, and sets variables (including incrementor and merchandising eVars).
3. **Adobe Analytics Workspace** → configures events, eVars, and props for reporting.

1. Website Code (PHP or Node.js Snippet)

Here's an example in **Node.js (Express)**. Same logic applies PHP is preferred. This snippet sits on the **order confirmation page** and outputs purchase data dynamically from backend variables:

```
// Example: Node.js (Express + any DB, e.g., MySQL/Postgres/Mongo)

app.get("/order/confirmation/:orderId", async (req, res) => {

 const orderId = req.params.orderId;



 try {

  // Fetch order from DB

  const order = await db.query(`

   SELECT o.id, o.total, o.currency, o.shipping, o.tax, c.id as customer_id, c.email

   FROM orders o

   JOIN customers c ON o.customer_id = c.id

   WHERE o.id = ?

  `, [orderId]);



  // Fetch cart items
```

```
const cart = await db.query(`
  SELECT p.id, p.name, p.category, oi.price, oi.quantity
  FROM order_items oi
  JOIN products p ON oi.product_id = p.id
  WHERE oi.order_id = ?
`, [orderId]);


// Build the digitalData object dynamically
const purchaseData = {
  purchaseEvent: "purchase",
  order: {
    id: order[0].id,
    total: order[0].total,
    currency: order[0].currency,
    shipping: order[0].shipping,
    tax: order[0].tax
  },
  customer: {
    id: order[0].customer_id,
    email: order[0].email
  },
  cart: cart.map(item => ({
    id: item.id,
    name: item.name,
    category: item.category,
    price: item.price,
    quantity: item.quantity
  }))
};
```

```
  res.send(`
    <html>
      <head><title>Order Confirmation</title></head>
      <body>
        <h1>Thank you for your purchase!</h1>
        <script>
          window.digitalData = ${JSON.stringify(purchaseData)};
        </script>
      </body>
    </html>
  `);
 } catch (err) {
  console.error("Error fetching order:", err);
  res.status(500).send("Something went wrong");
 }
});
```

2. Adobe Launch Setup

Step 1: Data Elements (all JavaScript Variables)

Naming convention: objectName.sec1.sec2.varName **(Persistence/Lifetime: Page View)**

Data Element Name	Type	Path (JS Variable)
purchase.event	JS Variable	digitalData.purchaseEvent
purchase.order.id	JS Variable	digitalData.order.id
purchase.order.total	JS Variable	digitalData.order.total
purchase.order.currency	JS Variable	digitalData.order.currency
purchase.order.shipping	JS Variable	digitalData.order.shipping
purchase.order.tax	JS Variable	digitalData.order.tax
purchase.customer.id	JS Variable	digitalData.customer.id
purchase.customer.email	JS Variable	digitalData.customer.email
purchase.cart	JS Variable	digitalData.cart (returns full array)

Step 2: Rule Setup

- **Rule Name:** Purchase Confirmation
- **Event:** Page Load (Bottom of Page)
- **Condition:**
    - Type: Value Comparison
    - Data Element: purchase.event
    - Operator: Equals
    - Value: purchase (enable Regex)
- **Action:** Adobe Analytics → Set Variables

Step 3: Setting Analytics Variables in Rule

Inside **Set Variables Action**:

1. **Events**
    - purchase event mapped to event1 (or your reserved event ID).
2. **eVars**
    - purchase.order.id → eVar1 (Order ID)
    - purchase.customer.id → eVar2 (Customer ID)
    - purchase.customer.email → eVar3 (Customer Email)
    - purchase.order.currency → eVar4 (Currency)
3. **Props** (if you need quick reporting props)
    - purchase.order.id → prop1
    - purchase.customer.email → prop2
4. **Merchandising Variables (s.products)**

    Format:

    s.products="category;productID;quantity;price;eventX=incrementorValue|eventY=incrementorValue"

    In Launch, configure purchase.cart Data Element in a **Custom Code Action** to loop and build s.products.

    Example:

```
var cartItems = _satellite.getVar("purchase.cart");
var productsStr = "";
cartItems.forEach(function(item, index){
  if(index > 0) productsStr += ",";
```

```
  productsStr += item.category + ";" + item.id + ";" + item.quantity + ";" + item.price + ";event2=" +
_satellite.getVar("purchase.order.shipping") + "|event3=" + _satellite.getVar("purchase.order.tax");

});

s.products = productsStr;
```

3. Adobe Analytics Workspace Setup

Since the Launch Extension already has the **Report Suite configured**, here's what you map inside Analytics Admin:

1. **Conversion Events**
    - event1 = Purchase (Counter)
    - event2 = Shipping Amount (Numeric Incrementor)
    - event3 = Tax Amount (Numeric Incrementor)
2. **Merchandising Variables**
    - Enable **product-level merchandising** for eVars if needed (e.g., Product Category, Product ID).
3. **eVars**
    - eVar1 = Order ID
    - eVar2 = Customer ID
    - eVar3 = Customer Email
    - eVar4 = Currency
4. **Props (optional)**
    - prop1 = Order ID
    - prop2 = Customer Email
5. **Validation in Workspace**
    - Create a Freeform Table:
        - Rows → eVar1 (Order ID)
        - Columns → Event1 (Purchases)
        - Metrics → Revenue, Shipping (event2), Tax (event3)

End Result:

- When confirmation page loads, Launch checks if purchase.event = purchase.
- Rule fires, sets variables & s.products dynamically.
- Adobe Analytics receives: event1, order ID, revenue, shipping, tax, and full cart details.
- Workspace shows purchases, order details, shipping/tax breakdowns, and product merchandising.

B. Campaign Tracking

Campaign tracking is crucial in Adobe Analytics to identify how users arrive at the site through marketing campaigns (email, paid ads, social, etc.). The common industry practice is to capture campaign identifiers (e.g., utm_campaign) from the URL and map them to an **eVar** or Campaign variable. This allows attribution reporting and feeds into **Marketing Channels**.

In this implementation scenario, campaign values are captured dynamically from the **URL query string** and passed into Adobe Analytics via Adobe Launch.

1. Data Element Setup (Persistence/Lifetime: Page View)

- **Type**: Query String Parameter
- **Name**: campaign.utm.campaign (follows convention objectname.sec1.sec2.varname)
- **Parameter to capture**: utm_campaign
- This data element will return the campaign value whenever it is present in the page URL.

2. Rule Setup (All Page Rule)

- **Rule Name**: All Pages – Campaign Tracking
- **Event**: Page Load → Bottom of Page
- **Condition**:
    - Type: Exception
    - Path **does not contain** → /error-page/ (assuming this is the error page)
    - This ensures campaign values are not tracked on error pages.
- **Action**:
    - Set Variables → Map campaign.utm.campaign to Campaign variable (or eVar configured for campaign).
    - Send Beacon → Standard beacon fired to Adobe Analytics.

3. Analytics Variable Configuration

- **Campaign Variable**:
    - In **Report Suite Manager → Conversion Variables (eVars)**, configure:
        - **eVar Name**: Campaign
        - **Expiration**: Visit (or custom, depending on attribution needs)
        - **Allocation**: Most Recent (commonly used for campaign tracking)

  This eVar will now hold the campaign value captured from utm_campaign.

4. Marketing Channel Setup (Glimpse)

- Navigate to **Admin → Report Suites → Marketing Channel Processing Rules**.
- Define a rule for Campaign tracking:

- **Condition**: If Campaign (or utm_campaign) is set → Attribute to channel **"Campaigns"**.
- You can set other rules for Paid Search, Display, Email, Social, etc. based on different query string parameters (utm_source, utm_medium). Enable all which is by default.

- This enables attribution reporting in the **Marketing Channels report**.

End Result:

With this, the **Campaign implementation scenario is complete** — capturing the campaign parameter, excluding error pages, mapping it to an eVar, and ensuring attribution flows into Marketing Channels.

C. User Properties

The **User Properties event** tracks login-based user segmentation (e.g., Beginner vs. Professional users). This helps you break down behaviors, conversions, and content usage by **user type** in Adobe Analytics.

In this scenario, I have two distinct login flows:

- **Beginner User** → Redirected to a beginner landing page
- **Professional User** → Redirected to a professional landing page

Each path loads a **JavaScript snippet** that pushes dynamic variables for the logged-in user type. Adobe Launch then picks these up through **data elements** and **rules**, setting the correct variables into Analytics.

1. Website Code (placed in respective login landing pages)

These snippets are dynamic (values come from backend/session), but the naming convention ensures separation of Beginner and Professional.

Beginner Landing Page Snippet

```
<script>
window.digitalData = window.digitalData || {};
digitalData.user = {
  beginner: {
    type: "beginner",
    event: "beginner_event"
  }
};
</script>
```

Professional Landing Page Snippet

```
<script>
window.digitalData = window.digitalData || {};
digitalData.user = {
  professional: {
    type: "professional",
```

```
  event: "professional_event"
 }
};
</script>
```

Each page only contains the relevant snippet, so **Beginner users only get beginner variables**, and **Professional users only get professional variables**.

2. Adobe Launch Setup

Data Elements (all type = JavaScript Variable) (Persistent/Lifetime: Visitor)

- user.beginner.type → captures "beginner"
- user.beginner.event → captures "beginner_event"
- user.professional.type → captures "professional"
- user.professional.event → captures "professional_event"

Rules

I'll create **two separate rules** for Beginner and Professional.

Rule 1 – Beginner Login Tracking

- **Event**: Page Load → Bottom of Page
- **Condition**: Data Element → user.beginner.event equals "beginner_event"
- **Actions**:
    - Set Variables → Map user.beginner.type → eVar (or prop, but better as **eVar** since it persists across visit)
    - Set Variables → Map user.beginner.event → Custom Event (e.g., "Beginner Login Event")
    - Send Beacon

Rule 2 – Professional Login Tracking

- **Event**: Page Load → Bottom of Page
- **Condition**: Data Element → user.professional.event equals "professional_event" (Scope = Visitor)
- **Actions**:
    - Set Variables → Map user.professional.type → eVar (or prop, same logic)
    - Set Variables → Map user.professional.event → Custom Event (e.g., "Professional Login Event")
    - Send Beacon

3. Analytics Configuration

- **eVars (User Type Tracking)**
    - eVarX → Stores **Beginner** or **Professional** (captured via user.type)

- Expiration = Visit
- Allocation = Most Recent

- **Events (Login Events)**
  - EventA = Beginner Login Event (counter)
  - EventB = Professional Login Event (counter)

End-to-End Flow:

1. **Beginner User logs in** → lands on Beginner page → snippet sets beginner_event and beginner → Launch rule fires → Analytics records "Beginner Login Event" + eVar(User Type=Beginner).
2. **Professional User logs in** → lands on Professional page → snippet sets professional_event and professional → Launch rule fires → Analytics records "Professional Login Event" + eVar(User Type=Professional).
3. In **Workspace**, analysts can break down all KPIs (purchases, content usage, campaign performance) **by User Type** and measure **unique Beginner vs. Professional logins**.

LEARNINGS &REFERENCES

- Gained hands-on knowledge of Adobe Analytics implementation covering **purchase, campaign, and user property tracking** using data elements, rules, events, eVars, props, and beacons.
- Learned how data flows from **website (dynamic backend values & snippets) → Launch (data elements, rules, conditions, triggers) → Analytics (variable mapping, incrementors, merchandise, campaign configs) → Workspace (reporting & segmentation)**.
- Understood **use cases and configurations**: purchase conversions, campaign attribution via query parameters, and user segmentation (beginner vs. professional) with persistent variables.
- Built awareness of **best practices** such as avoiding hardcoding, using eVars for persistence, props for pathing, and applying conditions/exceptions for accurate real-world tracking.

Reference (Udemy Course Link): - https://www.udemy.com/course/adobe-analytics-implementation-guide/

- Additional topics learned from YouTube tutorials + AI tools like ChatGPT (mainly for js codes – DataLayer, custom js code for setting variables, etc) + Browser/Search Engine.