

Adobe Summit

LAB WORKBOOK

**L537 – Implement Adobe
Target for Optimal
Performance**

Adobe Summit 2025

Lab Overview.....	3
Key Takeaways.....	3
Prerequisites	3
Part 1: Create On-Device Decisioning (ODD) activities in Target and assign it to CDN experimentation property	4
Exercise 1: Create a CDN Experimentation Target property for optimizing the ODD rules package size (preconfigured).....	4
Exercise 3: Create the test A/B activity in the CDN Experimentation workspace	8
Part 2: CDN edge-worker CLI and CDN Property configuration.....	13
Intro: CDN overview	13
Exercise 1: CDN CLI and CDN credentials.....	13
Step 1 [preconfigured on the lab machines]: Install Akamai CLI	13
Step 2: Verify Akamai CLI Installation - To ensure Akamai CLI is installed properly, run:	14
Step 3 [preconfigured on the lab machines]: Install Akamai CLI for EdgeWorkers Assuming Akamai CLI is installed, run the following command to install EdgeWorkers:.....	14
Step 4: Verify EdgeWorkers Installation - To ensure EdgeWorkers is installed properly, run:.....	14
Step 5 [preconfigured on the lab machines]: Create an API Client	15
Step 6: Check Credentials.....	18
Exercise 2: CDN Property configuration (preconfigured for the lab).....	19
Part 3: Coding with CDN Experimentation SDK	24
Exercise 1: Open the Lab Project in the IDE	25
Exercise 2: Creating the CDN Experimentation SDK Client Instance	27
Exercise 3: Retrieving the consequences	30
Exercise 4: Sending the Display event	31
Exercise 5: Working with the consequences	31
Appendix.....	37
Edge worker runtime environment limitations	37

Lab Overview

In this lab you will learn a new implementation method for Adobe Target that will allow you to perform CDN experimentation. This new method will allow you to create A/B and XT Target activities and perform the decisioning work on your CDN for the fastest possible results.

Key Takeaways

- Understand the different Target implementation methods and how using edge workers can lead to better performance
- Using the Target Properties learn how to be selective about which activities are included in your rules bundle
- Get familiar with setting up the local environment for development
- Understand how to configure an Akamai edge worker for Target experimentation
- Learn the basics of how to code in the CDN environment

Prerequisites

- Adobe Target with product admin rights for editing workspaces and properties
- Enable the on-device decisioning capability in Adobe Target
- Access to your preferred CDN edge worker environment with the appropriate rights to configure settings
- Visual Studio Code software or another development environment of your choice

Part 1: Create On-Device Decisioning (ODD) activities in Target and assign it to CDN experimentation property

Section overview: In this section, we'll start by reviewing the On-device decisioning features available in Adobe Target. Next, we'll create a CDN Experimentation Target property (optional step for premium users). Then, using the provided lab user credentials, we will create an A/B Target test that will be applied on the lab test website.

Intro: ODD in Adobe Target

- Enables near-zero latency content delivery by performing in-memory decisioning on the device it operates on.
- Caches A/B Test and Experience Targeting (XT) activity data
- Requires a rules artifact

Exercise 1: Create a CDN Experimentation Target property for optimizing the ODD rules package size (preconfigured)

Step 1 [preconfigured on the lab machines]: Confirm the user Product Role for the workspace is set to Approver and Save

Add users to this product profile

Enter users' email addresses to add them to this product profile.

User 1

Name, user group, or email address	vivas@adobe.com	ID type	Adobe ID
Name	VIVAS	Email	vivas@adobe.com
PRODUCT ROLE	Approver		

User 2

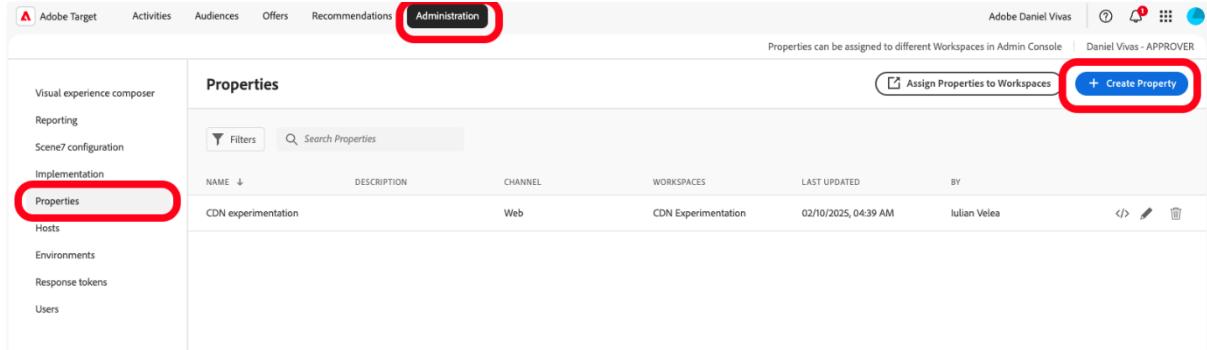
Name, user group, or email address

User 3

Name, user group, or email address

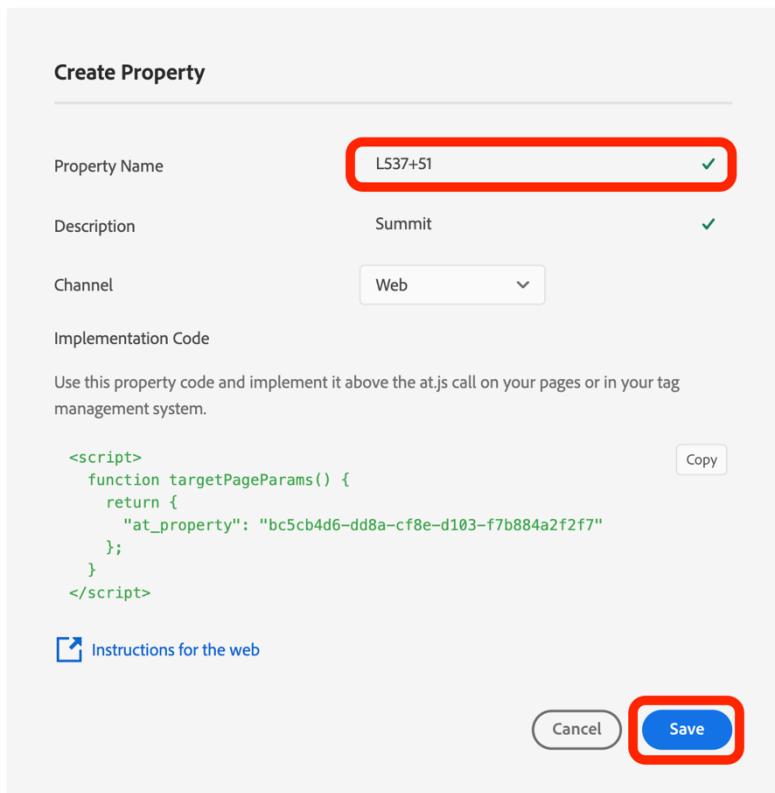
Cancel **Save**

Step 2 [preconfigured on the lab machines]: In the Target UI navigate to the **Administration/Properties** section and click on Create Property button



The screenshot shows the Adobe Target Admin Console interface. At the top, there are navigation tabs: Adobe Target, Activities, Audiences, Offers, Recommendations, and Administration (which is highlighted with a red box). Below the tabs, there's a sidebar with links: Visual experience composer, Reporting, Scene7 configuration, Implementation, Properties (highlighted with a red box), Hosts, Environments, Response tokens, and Users. The main area is titled 'Properties' and contains a table with columns: NAME, DESCRIPTION, CHANNEL, WORKSPACES, LAST UPDATED, and BY. A single row is visible: 'CDN experimentation' (DESCRIPTION), 'Web' (CHANNEL), 'CDN Experimentation' (WORKSPACES), '02/10/2025, 04:39 AM' (LAST UPDATED), and 'Iulian Velea' (BY). At the top right of the properties table, there are buttons for 'Assign Properties to Workspaces' and '+ Create Property' (highlighted with a red box).

Step 3 [preconfigured on the lab machines]: Edit the properties details (your property will have the lab and seat number in the name e.g: L537+##) and click Save



The screenshot shows the 'Create Property' dialog box. It has fields for 'Property Name' (L537+51), 'Description' (Summit), and 'Channel' (Web). Below these is an 'Implementation Code' section containing a script block:

```
<script>
  function targetPageParams() {
    return {
      "at_property": "bc5cb4d6-dd8a-cf8e-d103-f7b884a2f2f7"
    };
  }
</script>
```

At the bottom, there are 'Cancel' and 'Save' buttons, with 'Save' highlighted with a red box.

Step 4 [preconfigured on the lab machines]: Confirm the on-device decisioning setting is toggled on; this allows us to use the activities for On-Device Decisioning

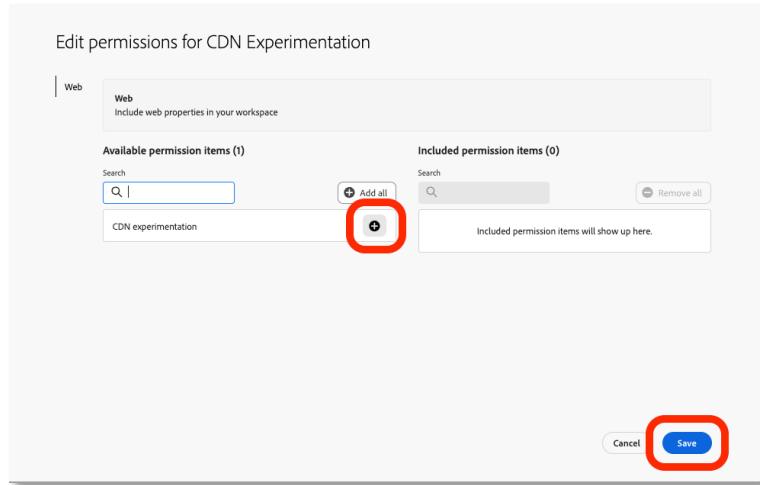
The screenshot shows the Adobe Target Administration interface. The top navigation bar includes links for Adobe Target, Activities, Audiences, Offers, Recommendations, and Administration. The Administration link is highlighted with a red box. The main content area is titled 'Implementation'. On the left, there's a sidebar with links for Visual experience composer, Reporting, Scene7 configuration, Implementation (which is also highlighted with a red box), Properties, Hosts, Environments, Response tokens, and Users. The right side displays 'Account details' with fields for Client Code (adobedanielvivas) and IMS Organization Id (B2C94E025B2385B40A495E2C@AdobeOrg). A prominent red box highlights the 'On-Device Decisioning' toggle switch, which is turned on (blue). Below it is a checkbox for 'Include all existing on-device decisioning qualified activities in the artifact'.

Step 5 [preconfigured on the lab machines]: Navigate back to the Admin Console/Adobe Target/Default Workspace profile settings and edit the Profile Permissions by clicking on the pencil icon next to “Web” under “Permissions Access”

The screenshot shows the Admin Console interface. The top navigation bar includes Overview, Products, Users, Account, Insights, Settings, and Support. The current page is 'All products and services > Adobe Target > Default Workspace'. The workspace name 'Default Workspace' is shown with a blue gear icon. Below it, the 'Auto assignment rule' is listed as 'None'. The 'Permissions' tab is selected in the navigation bar. A search bar for 'Search permissions by name' is present. A red box highlights the 'Edit permission Web' button, which is located above a table. The table has columns for 'Name', 'Edit permission Web' (with a red box), and 'Description'. It lists three items: 'Web' (59 of 59 included), 'Mobile App' (3 of 3 included), and 'Email' (10 of 10 included).

Name	Edit permission Web	Description
Web	59 of 59 included	Include web properties in your workspace
Mobile App	3 of 3 included	Include mobile apps in your workspace
Email	10 of 10 included	Include email campaigns in your workspace

Step 6 [preconfigured on the lab machines]: Include the properties you will be using, in the permissions items by clicking on the “+” button and then Save



Exercise 2: Log in to Adobe Target UI

Step 1: Navigate to experience.adobe.com

Step 2: Enter the email address: L537+##@adobeeventlab.com

Step 3: Select “Company or School Account”

Step 4: Enter the password: Adobe4Summit!

Step 5: Select Adobe Target from the list of available solutions

The screenshot shows the Adobe Experience Cloud homepage. At the top, there is a banner for 'Adobe Summit' with the text 'Prepare to set the standard in digital experience.' and a 'Register now' button. Below the banner, there is a 'Quick access' section with several icons: Admin Console, Advertising Search, Social, & Commerce (Beta), Analytics, Audience Manager, Data Collection, Experience Platform, Privacy UI, Software Distribution, System Status, and Target. The 'Target' icon is highlighted with a red box.

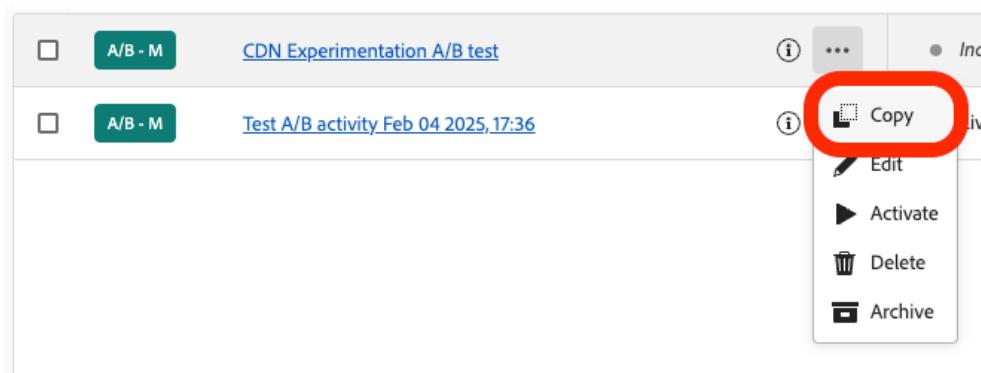
Exercise 3: Create the test A/B activity in the CDN Experimentation workspace

Lab website: <https://main--aem-cdn-experimentation-demo--aenascut.aem.live/>

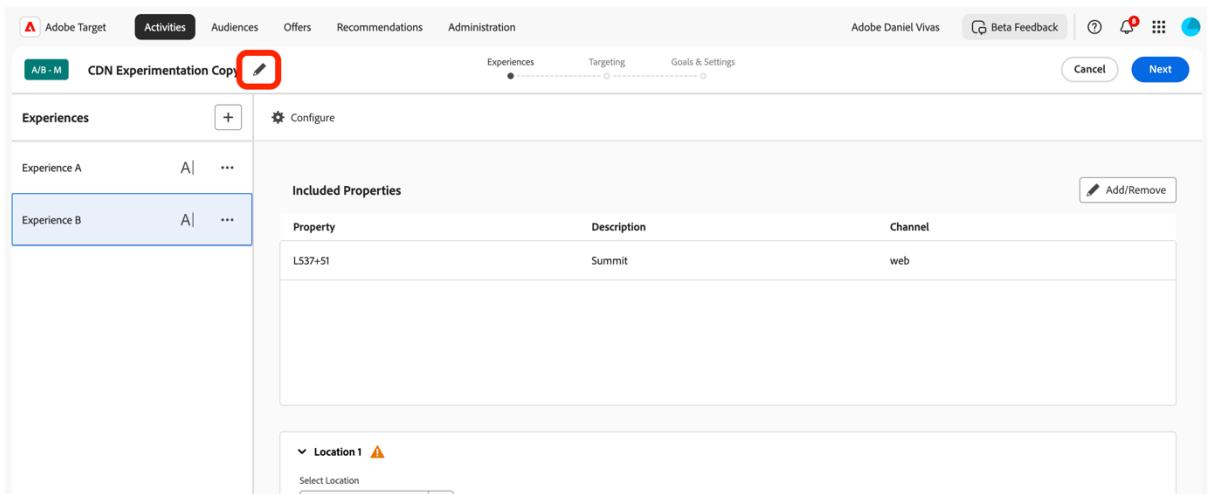
Step 1: Click on the Activities tab in the top horizontal navigation and identify the 'CDN Experimentation' activity that we have created in advance

The screenshot shows the 'Activities' page in Adobe Target. The top navigation bar includes 'Activities' (which is highlighted with a red box), 'Audiences', 'Offers', 'Recommendations', and 'Administration'. The workspace is set to 'Default Workspace'. The main area displays a table titled 'All Activities' with columns for 'Type', 'Name', 'Status', 'Last Updated', 'Priority', and 'Property'. Two activities are listed: 'CDN Experimentation A/B test' (Inactive) and 'Test A/B activity Feb 04 2025, 17:36' (Live). The 'CDN Experimentation A/B test' row is also highlighted with a red box.

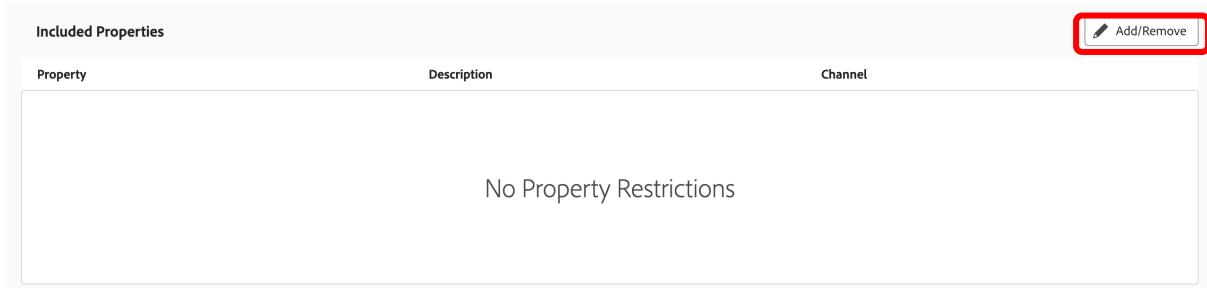
Step 2: Click on the ellipsis menu icon and select 'Copy' to duplicate the test activity

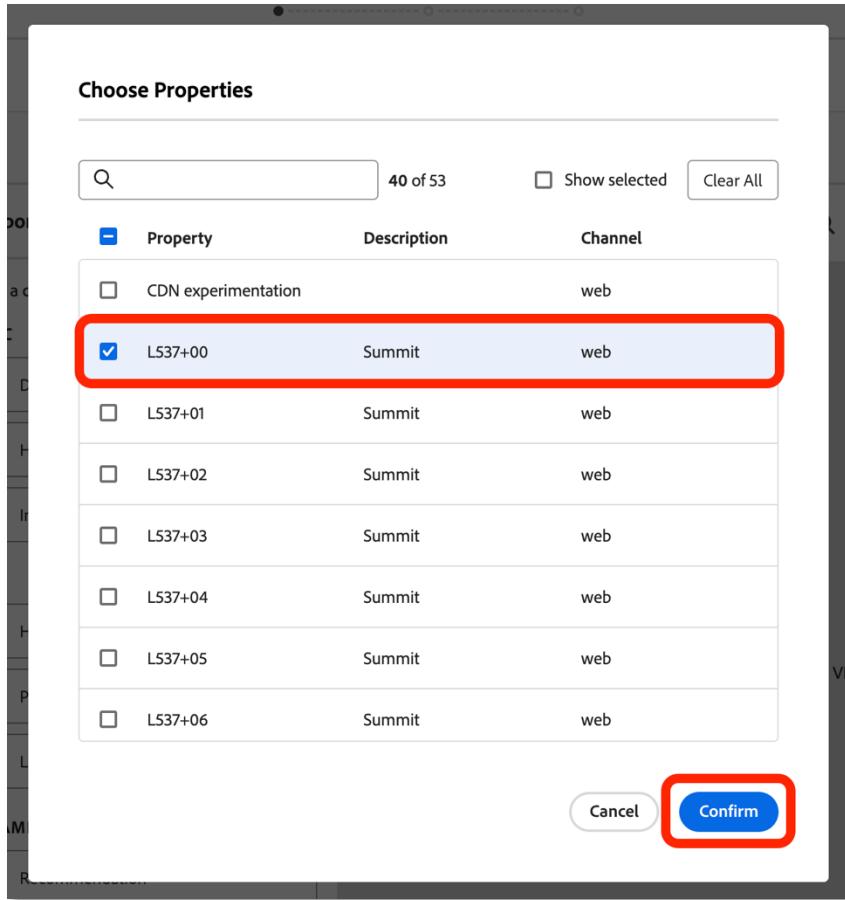


Step 3: Edit the activity name to include your seat number (ex: CDN Experimentation A/B test seat ##) and click save



Step 4: From the activity editing page, in the Included Properties section click “Add/Remove” to add the corresponding property





Step 5: Examine the changes being made to **Experience B** and click **Next**.

Bonus step: Add additional changes to the experience that you would want to see in your final result (e.g. change the **h1** element text and/or text color)

Experiences

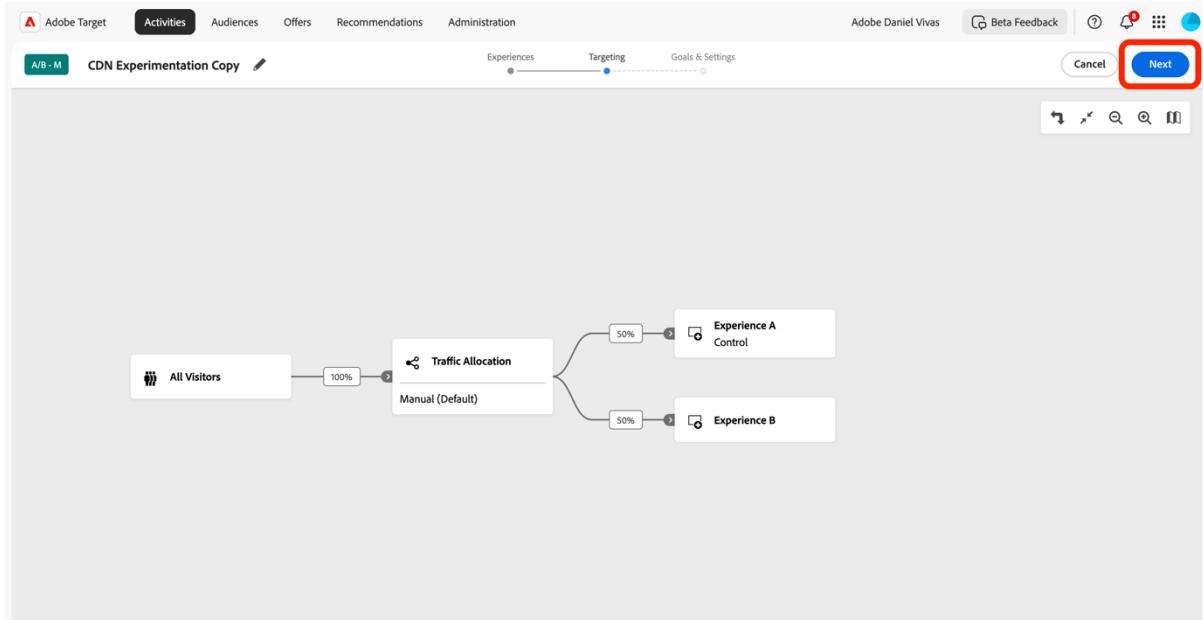
Experiences Targeting Goals & Settings

Content

```
{
  "payload": [
    {
      "selector": "#experience-the-magic-unforgettable-carousel-rides-for-the-whole-family",
      "payload": "<h1 id='experience-the-magic-unforgettable-carousel-rides-for-the-whole-family'>Experience the Magic: Unforgettable Carousel Rides for the Whole Family!</h1>"
    },
    {
      "selector": "main > div > p > picture",
      "payload": "<picture style='min-height:420px;'><source srcset='https://luma.enabledelements.adobe.com/content/luma/us/en/men/_jcr_content/root/hero_image.coreimg.jpeg' /><img alt='Unforgettable Carousel Rides for the Whole Family!'></picture>"
    }
  ]
}
```

Next

Step 6: Click Next on the “Targeting page”



Step 7: Click Save & Close to finish the editing

The screenshot shows the 'Activity Settings' page. The 'Save & Close' button in the top right corner is highlighted with a red box. The page includes sections for 'Objective', 'Priority' (with a priority scale from 0 to 10), 'Duration' (start: When Activated, end: When Deactivated), and 'Reporting Settings' (reporting source: Adobe Target).

Step 8: Activate your Target activity

Screenshot of the Adobe Target interface showing the configuration of an A/B test activity.

The top navigation bar includes: Adobe Target, Activities, Audiences, Offers, Recommendations, Administration, Adobe Daniel Vivas, Beta Feedback, and various icons.

The main content area shows the activity details:

- Activity Status:** Inactive (button circled in red)
- Activity Name:** CDN Experimentation A/B test DV
- On-Device Decisioning Eligible:** Enabled
- Workspace:** Default Workspace
- Activity Location:** target-global-mbox
- Goal:** I want to... increase engagement
Measured by... Page Views
- Info:** Priority: 0, Activity ID: 272170

Target: All Visitors → Traffic Allocation (Manual (Default)) → Experience A (Control) and Experience B (50% each).

```
graph LR; AllVisitors[All Visitors] -- 100% --> TA[Traffic Allocation  
Manual (Default)]; TA --> EA[Experience A Control]; TA --> EB[Experience B]; EA -- 50% --> EA; EB -- 50% --> EB;
```

Part 2: CDN edge-worker CLI and CDN Property configuration

Intro: CDN overview

Content Delivery Networks benefits:

- Distributed servers delivering web content based on user location.
- Reduces load times and improves performance by caching content globally.

Edge Computing aspects:

- Brings computation and data storage closer to data sources and users.
- Reduces latency, conserves bandwidth, and enhances user experience.

CDNs + Edge Computing key benefits:

- Real-time data processing and lower latency.
- Improved performance for dynamic content.

Exercise 1: CDN CLI and CDN credentials

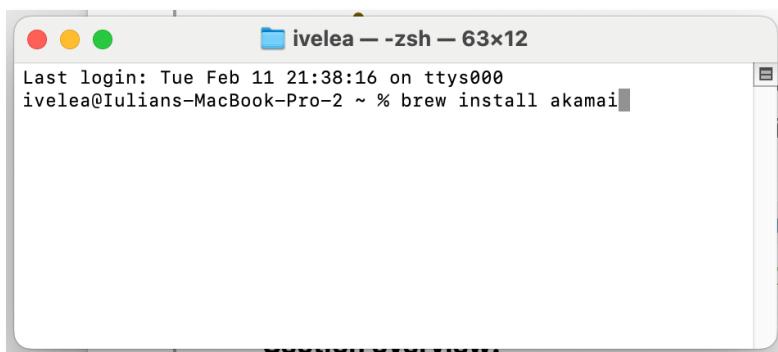
This section contains steps related to configurations already available on the lab machines.

Section overview: To be able to work with Akamai EdgeWorkers there are a few things that are required:

- Access to Akamai Control Center: <https://control.akamai.com/>
- Access to your organization group in Akamai
- A configured Property in Akamai
- Akamai CLI installed on development machine
- Akamai CLI for EdgeWorkers installed on development machine
- API Client credentials

Step 1 [preconfigured on the lab machines]: Install Akamai CLI

1. Open your terminal.



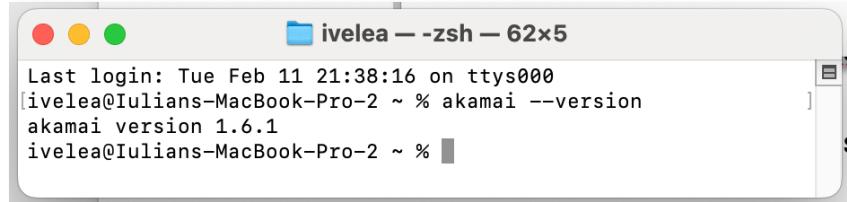
- Run the following command to install Akamai CLI using Homebrew:

```
brew install akamai
```

Step 2: Verify Akamai CLI Installation - To ensure Akamai CLI is installed properly, run:

```
akamai --version
```

This should print the Akamai CLI version, confirming the installation.



A screenshot of a macOS terminal window titled "ivelea — zsh — 62x5". The window shows the following text:
Last login: Tue Feb 11 21:38:16 on ttys000
ivelea@Iulians-MacBook-Pro-2 ~ % akamai --version
akamai version 1.6.1
ivelea@Iulians-MacBook-Pro-2 ~ %

Step 3 [preconfigured on the lab machines]: Install Akamai CLI for EdgeWorkers

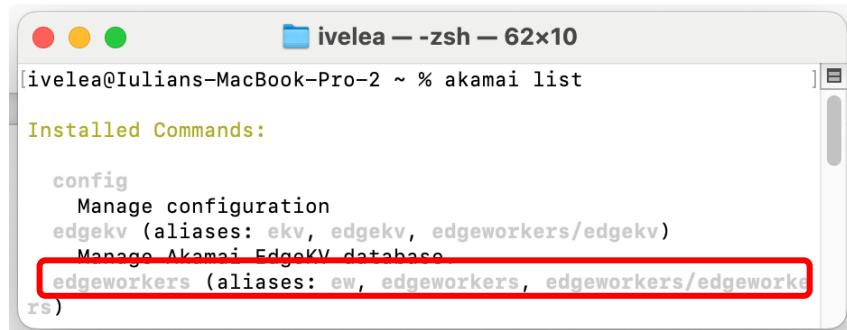
Assuming Akamai CLI is installed, run the following command to install EdgeWorkers:

```
akamai install edgeworkers
```

Step 4: Verify EdgeWorkers Installation - To ensure EdgeWorkers is installed properly, run:

```
akamai list
```

This should print a list of available Akamai CLI commands, including "**edgeworkers**".



A screenshot of a macOS terminal window titled "ivelea — zsh — 62x10". The window shows the following text:
ivelea@Iulians-MacBook-Pro-2 ~ % akamai list

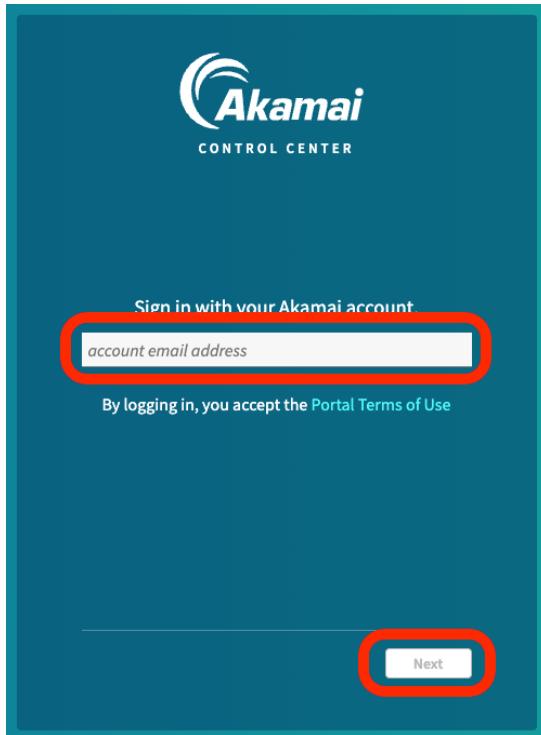
Installed Commands:

config
 Manage configuration
edgekv (aliases: ekv, edgekv, edgeworkers/edgekv)
 Manage Akamai EdgeKV database
edgeworkers (aliases: ew, edgeworkers, edgeworkers/edgeworkers)
rs)

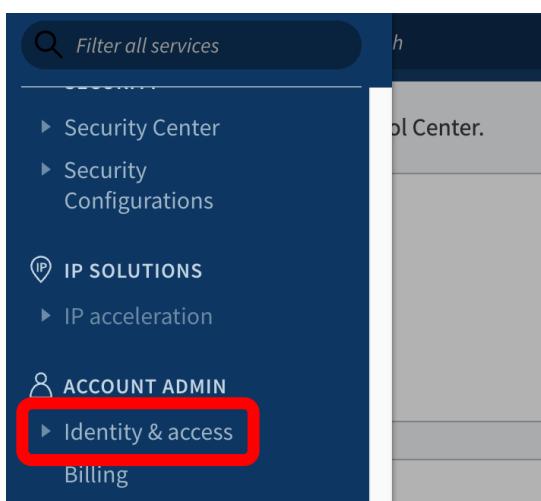
The "edgeworkers" command is highlighted with a red rectangle.

Step 5 [preconfigured on the lab machines]: Create an API Client

1. Go to **control.akamai.com** and use the registered email address to sign in by clicking on the “**Next**” button.



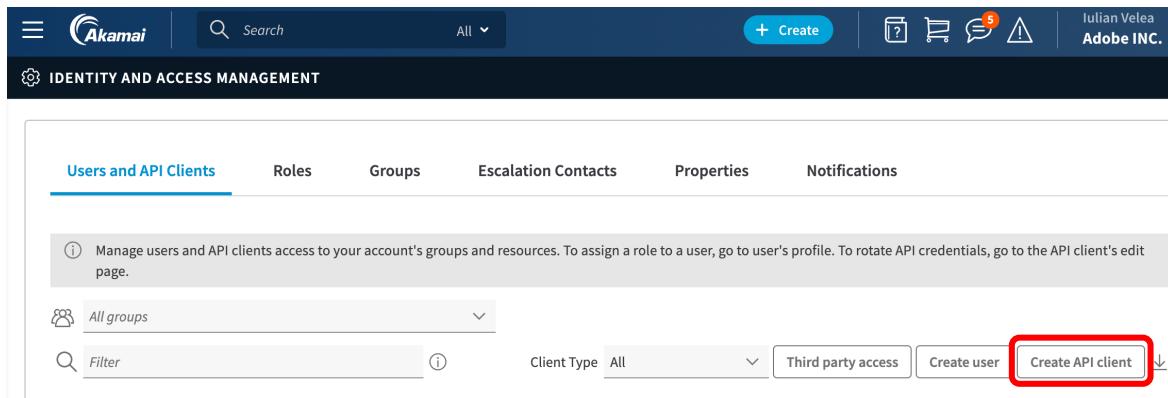
2. From the left navigation bar, go to "**Account Admin**". Select "**Identity and Access**".
- 3.



You should be redirected to the "**Users and API Clients**" tab.

4. On the "Users and API Clients" page, click "**Create API Client**".

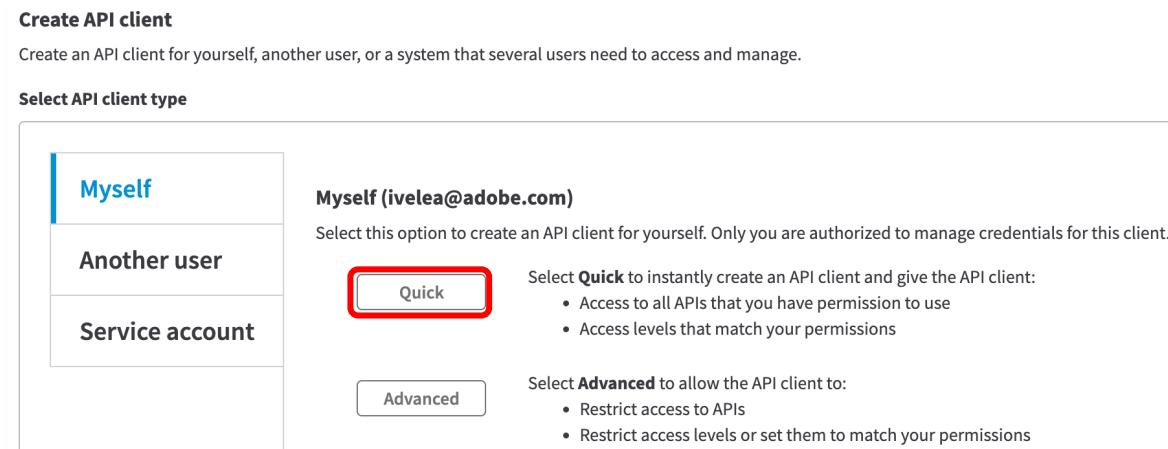
5.



The screenshot shows the Akamai Identity and Access Management dashboard. At the top, there's a navigation bar with the Akamai logo, a search bar, and various icons. Below it, a secondary header bar says "IDENTITY AND ACCESS MANAGEMENT". The main content area has a tab bar with "Users and API Clients" (which is active and underlined), "Roles", "Groups", "Escalation Contacts", "Properties", and "Notifications". Underneath this, there's a help message about managing users and API clients. Below that is a search/filter section with dropdowns for "All groups" and "Filter", and buttons for "Client Type All", "Third party access", "Create user", and a redboxed "Create API client".

6. On the "**Create API Client**" page, select "**Myself**" and then click the "**Quick**" button. This will create an API Client with all your credentials. For a more restricted version, select "**Advanced**" and grant only the necessary permissions.

7.



The screenshot shows the "Create API client" page. It starts with a heading "Create API client" and a sub-instruction: "Create an API client for yourself, another user, or a system that several users need to access and manage." Below this is a section titled "Select API client type" with three options: "Myself", "Another user", and "Service account". The "Myself" option is highlighted with a blue background. To its right, there's a sub-section for "Myself (ivelea@adobe.com)". It says: "Select this option to create an API client for yourself. Only you are authorized to manage credentials for this client." Below this are two buttons: "Quick" (which is redboxed) and "Advanced". The "Quick" button has a description: "Select Quick to instantly create an API client and give the API client:" followed by a bulleted list: "• Access to all APIs that you have permission to use" and "• Access levels that match your permissions". The "Advanced" button has a description: "Select Advanced to allow the API client to:" followed by a bulleted list: "• Restrict access to APIs" and "• Restrict access levels or set them to match your permissions".

A new page named "**API Client for you**" will open.

8. On the "**API Client for you**" page, go to the bottom table and copy your credentials or create a new one.

API client for you

You have created an API client with access levels that match your permissions and access to all APIs that you have permission to use. To learn more, see [Get started with APIs](#).

Details

Name: ivelea@adobe.com_2 [Edit](#)

Description: ivelea@adobe.com_2 [Edit](#)

Notification list: ivelea@adobe.com

IP Allowlist: Enable

IP/Mask: No IP/Mask entries

[Show additional details](#)

Credentials

Host: akab-kgyfqt5fbfig2y5h-uoc7xogffjerja5b.luna.akamaiapis.net	
Access Token: akab-agwvx4xgudlg5hec-tc42hrkpphm6lsdc	
Download Copy credential <input checked="" type="checkbox"/> Show credential Create credential	
Copy and paste this information into a text file for all automated programs. The client secret is unique to the request and available once.	
<pre>client_secret=w6Q0vop+u1cUAIIIBI2fYDcCi1B/tHuPjrqKp1PoGc8= host=akab-kgyfqt5fbfig2y5h-uoc7xogffjerja5b.luna.akamaiapis.net access_token=akab-agwvx4xgudlg5hec-tc42hrkpphm6lsdc client_token=akab-nsvq7hajmqu4dhdf-hn3666vhzkapl46u</pre>	

Make sure to copy the credentials and store them.

Step 6: Check Credentials

Akamai CLI needs credentials to execute operations that might change the Akamai account state. These credentials should be stored in a special file named `.edgerc`.

On Mac OS X, the `.edgerc` file should be located at:

`/Users/{username}/.edgerc`

Check your local file by running the following command in your terminal:

```
cat /Users/ivelea/.edgerc
```

```
[ivelea@Iulians-MacBook-Pro-2 lab % cat /Users/ivelea/.edgerc
[default]
client_secret = LnI0YiShXuCGMGcHChD1NGu7iUcqXdx1t6CbhqTphS0=
host = akab-afbg4lk2cjuecl5e-ptzfemwj4obaz4lq.luna.akamaiapis.net
access_token = akab-urflycxooypjqqh-jptb3t37ijxbmtsx
client_token = akab-2k7zvjeuagquzpk-wkmifnjh6q4zecj1
ivelea@Iulians-MacBook-Pro-2 lab %]
```

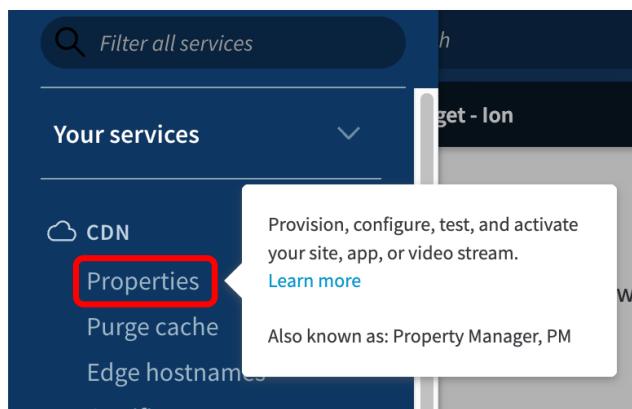
Exercise 2: CDN Property configuration (preconfigured for the lab)

Step 1: Log in to Akamai Control Center

Open your web browser and go to the Akamai Control Center: control.akamai.com
Enter your credentials to log in.

Step 2: Navigate to Property Manager

From the main dashboard, go to the left menu section. Click on "**Properties**" under the "**CDN**" section.



Step 3: Create a New Property

- In the Property Manager, click on the "+New Property" button.
- Select "**Ion Standard**" from the product list and click "**Create Property**".
- The "**Set up Ion Standard**" page should be displayed

Set up Ion Standard

Create a new property in Property Manager to control how the Akamai Edge Platform manages traffic to your site.

Property name

The contract you choose determines how your traffic is billed. Once you've chosen a contract, select the group your new configuration will save in and click **Next**.

Contract (i)

Group (i)

Rule Format

[Previous](#)

[Next](#)

- In this page set a “**Property name**”, select the “**Contract**” for your organization and click on the “**Next**” button

Step 4: Configure Property Settings

- In the **Edit** property configuration page, you will see various settings and options.

The screenshot shows the 'Edit' property configuration page with several sections:

- Property Version Information:** Production Status: INACTIVE; Staging Status: INACTIVE; Security Options: Standard TLS ready (selected); Enhanced TLS (radio button); Version Notes: Add notes for this property version...; Product: Ion Standard; Rule Format: latest; Contract: M-283VSBD; Modules: Access Control, Adaptive Acceleration, Advanced Offload, Brotli Support, Client Hints, Content Targeting / EdgeScape, Custom Behaviors, DataStream Logs, Device Characterization, EdgeWorkers, Enhanced Akamai Protocol, Enhanced Secure Delivery - Customer Cert, GraphQL Caching, HTTP/2, Mobile Detection & Redirect, Origin Services (Beta), Real User Monitoring, Request Debugging, Resource Optimizer Extended, Compatibility, Script Management, Site Failover, SiteShield, Standard Secure Delivery - Customer Cert, Variable Support, Web Application Firewall.
- Property Hostnames:** A red error message says "At least one property hostname is required." Below it is a table with columns: Status, Property Hostname, Certificate, Edge Hostname, Actions. A note says "Add a property hostname by clicking Add".
- Property Variables:** A table with a single row: + Variables.
- Property Configuration Settings:** Rules tab selected. It shows a Default Rule template with sections: Augment insights, Traffic reporting. A note says "NOTE: Default behaviors are applied to all requests for the property hostname(s)." Another note says "The Default Rule template contains all the necessary and recommended behaviors. Rules are evaluated from top to bottom and the last matching rule wins."

Step 5: Set up Hostnames

The screenshot shows the 'Property Hostnames' management page with a table:

Status	Property Hostname	Certificate	Edge Hostname	Actions
abc.adobelab2025.com	*.adobelab2025.edgekey.net	Unknown certificate Manage in CPS	*.adobelab2025.edgekey.net	...
abc.adobelab2025.com	*.adobelab2025.test.edgekey-staging.net	Unknown certificate Manage in CPS	*.adobelab2025.test.edgekey-stagi...	...
abc.adobelab2025.com	abc.adobelab2025.com	Unknown certificate Manage in CPS	scene7.com.edgekey.net	...

A red box highlights the '+ Hostnames' button at the top right of the table header.

Hostnames section define the origin **Hostnames** URLs on which the Akamai Property applies, the associated **Certificates** and the **Edge Hostnames** corresponding to the origin hostnames. When defining hostnames wildcards are accepted.

You can add new Hostnames by clicking on the “**+Hostnames**” button.

Step 6: Set Up Rules and Behaviors

The screenshot shows the configuration interface for a 'Default Rule'. On the left, a sidebar lists various properties like 'edge.adobedc.net', 'Allowed methods' (POST, OPTIONS, PUT, DELETE, PATCH), and 'Minimize payload' (Compressible objects, Enable HTTP2). The main area is divided into 'Criteria' and 'Behaviors' sections.

Criteria Section:

- If:** Hostname is one of student1.adobelab2025.test.edgekey-staging.net
- And:** Path matches one of /

Behaviors Section:

- EdgeWorkers:** Identifier 94606 (Tier 200) student1
- Enable:** On
- Identifier Information:** If you haven't created an Identifier yet, use the EdgeWorkers Management application to create one. When you're done, reload this page to view and select your new Identifier from the list.
- Resource Tier:** The resource tier determines the billing rate for an EdgeWorker ID. To learn more, open the EdgeWorkers Management application. Then select your EdgeWorker ID and open the Resource Tiers tab.

- In the Property configuration page, go to the "**Property Configuration**" section. In here you can define the Behavior of the property by adding **Rules**, configuring **Criteria** and **Behaviors** for each rule.
- When a request is received by the CDN Edge nodes, the rules are evaluated sequentially, and the most specific rule will be the one applied.
- If you want to add a new **Rule** you need to click on the "**+Rule**" button from the left panel.
- In the **Criteria** section you can define the conditions which are evaluated for each **Rule**. The **Condition** is a Boolean expression.
- In the **Behaviors** section you can define the behavior when a Rule has been selected to be applied.
- In the above example when the requested URL hostname matches the value student1.adobelab2025.test.edgekey-staging.net and the path is "/" then the "**student1**" rule will be selected. In this case the Behavior is to route the request to the EdgeWorker having the ID equal with **94606**.
- Click "**Save**" to persist the configuration of the **Property**.

Step 7: Activate the Property

- Once you have configured all the necessary settings, when you get back to the **Property Details** page you will see a new version being available. Click on the "Activate" button from the contextual menu.

The screenshot shows the 'Property Details' page for 'cdnexp.adobelab2025'. At the top, there are buttons for 'Back to Group', 'Open Version 14', 'Related apps', and three dots. Below this is a 'Property Description' field with placeholder text. The main area is divided into two sections: 'Active Staging Version' and 'Active Production Version'. The 'Active Staging Version' section contains configuration details like Version (Version 14), Status (Active), Product (Ion Standard), Rule format (latest), Hostnames (*.adobelab2025.edgekey.net, *.adobelab2025.test.edgekey-staging.net, abc.adobelab2025.com), and CP Code(s) (1781592 — cdnexp.adobelab2025). The 'Active Production Version' section notes that no versions are active. Below these sections is a 'Version History' table and an 'Activation History' table. The 'Activation History' table shows a single row for Version 14, which is based on Version 13, was last edited on Mar 9, 2025, by author aenascut@adobe.com, and is currently in the 'Active' state. To the right of the tables is a context menu with options: View, Review, Test This Staging Version, Edit New Version, **Activate** (which is highlighted with a red box), Deactivate Staging, Compare, View XML, and Download XML. There is also a three-dot ellipsis at the bottom right of the menu.

- Select the environment (e.g., **Staging** or **Production**) where you want to activate the property.
- Review the configuration and click "**Activate vxx on Production**" to deploy the new Property version.

Step 8: Test the Configuration

- After activation, test the configuration by accessing the external URL through your Akamai property.

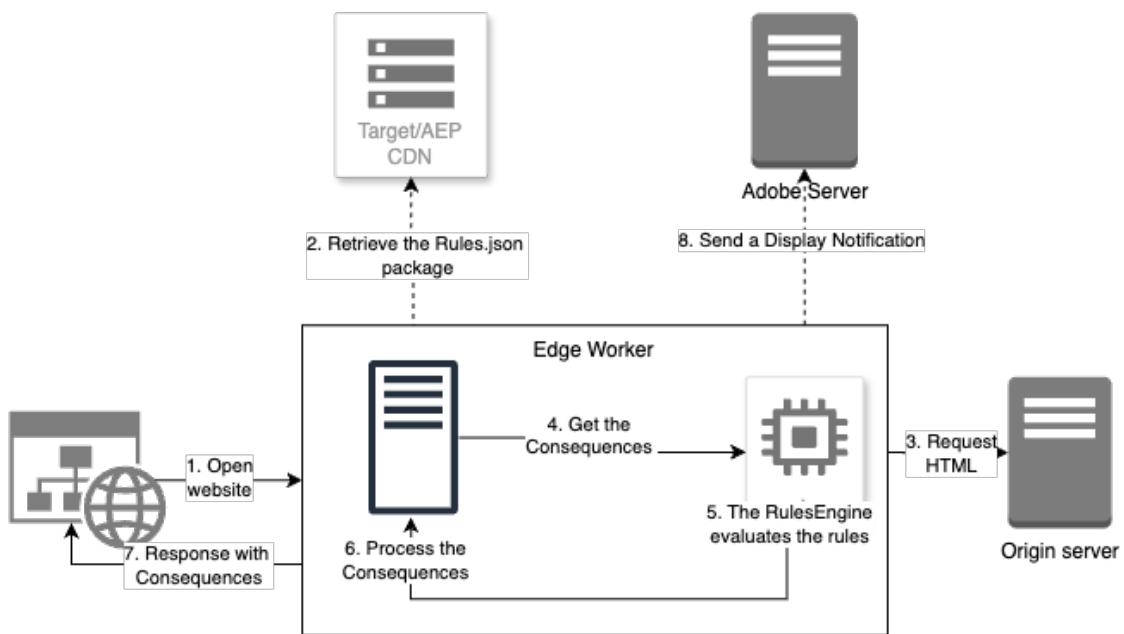


- Ensure that the requests are properly routed, and that the external URL is accessible.

Part 3: Coding with CDN Experimentation SDK

Overview of CDN Experimentation Solution:

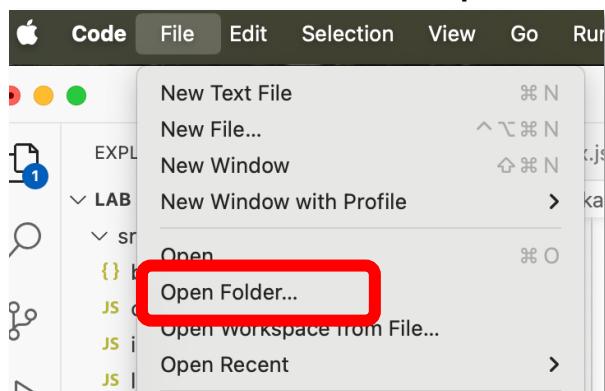
1. The visitor accesses the website, initiating a request to the CDN edge.
2. The rules package is either fetched or supplied.
3. The HTML page is retrieved from the origin.
4. The rules are assessed to determine the outcomes.
5. Consequences are applied.
6. Personalized page is displayed
7. Event reports are dispatched.



Exercise 1: Open the Lab Project in the IDE

Step 1: Open the **Visual Studio Code** IDE 

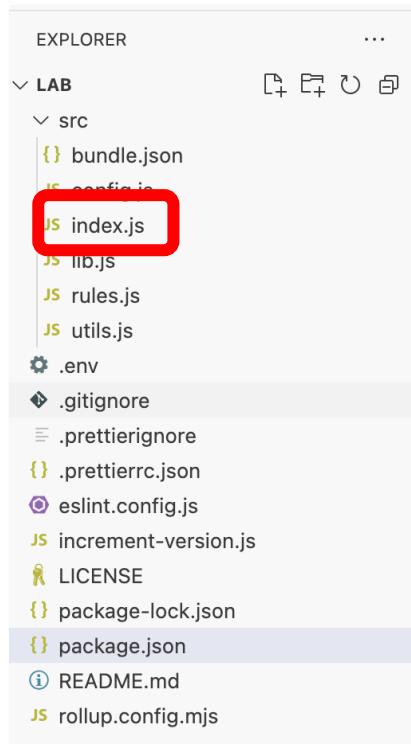
Step 2: From the Menu bar select **File/Open Folder** action



Step 3: From the **Desktop** folder select the **lab** subfolder containing the lab code



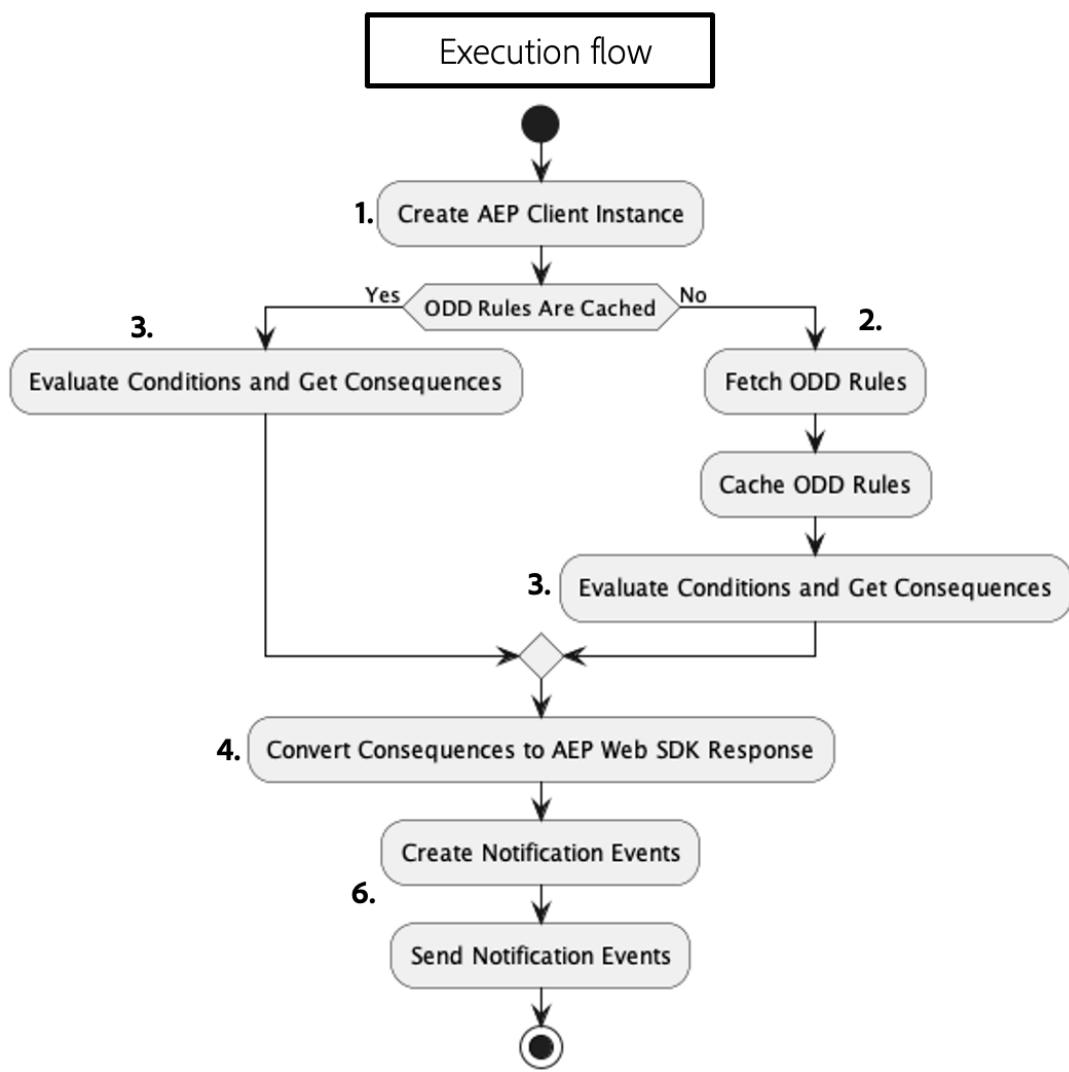
You should see the following project structure in VS Code **Explorer** window:



Step 4: Click on the **index.js** file to open it in the Editor pane.

EdgeWorker execution process:

1. The SDK Client is set up and configured (for example, with the rules bundle CDN address).
2. It fetches and stores the rules package.
3. The **rules** are then assessed, and the applicable **consequences** are determined.
4. These consequences are transformed into a response compatible with AEP Web SDK.
5. Based on this response, the SDK user can decide whether to apply it (using *html-rewriter*), integrate it into the HTML page, or utilize it for other personalization needs.
6. The SDK user has the option to notify the Adobe Edge Network once the response has been "consumed" (to ensure accurate metric tracking).



Exercise 2: Creating the CDN Experimentation SDK Client Instance

The following snippet demonstrates the instantiation of the CDN Experimentation client with the specified arguments:

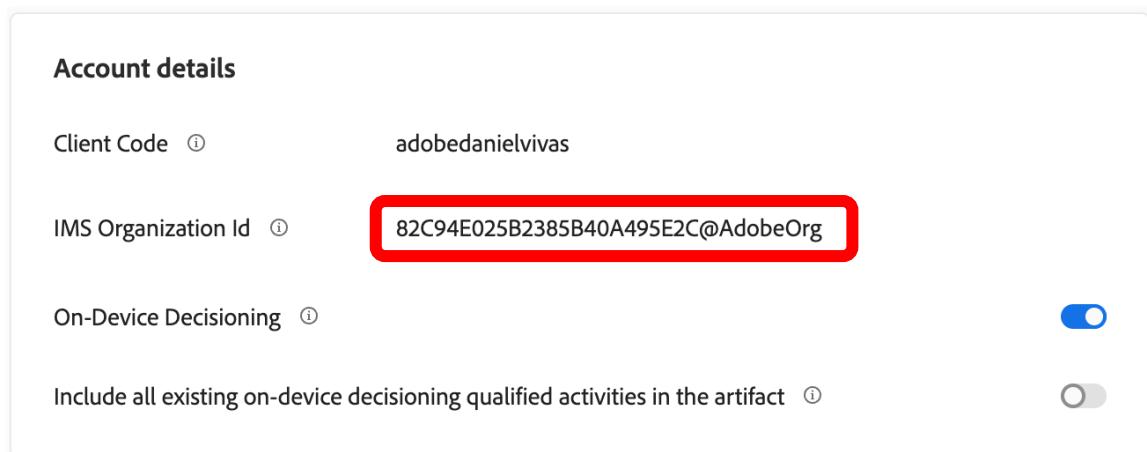
```
1 import { Client } from "@adobe/cdn-experimentation-akamai-sdk";
2 import { rules } from "./rules.js";
3 import { PROPERTY_TOKEN } from "./config.js";
4
5 export async function responseProvider(request) {
6   try {
7     const client = await Client({
8       datastreamId: DATASTREAM_ID,
9       orgId: ORG_ID,
10      propertyToken: PROPERTY_TOKEN,
11      oddEnabled: true,
12      rules: rules,
13    });
14  }
```

Step 1: Into the **index.js** file add the following lines of code, inside the **responseProvider** function, to create and configure the SDK client:

```
const client = await Client({
  orgId: "82C94E025B2385B40A495E2C@AdobeOrg",
  datastreamId: "bdb5cb8a-4496-4abd-8afc-e9396c1b1c27",
  propertyToken: "",
  oddEnabled: true,
  edgeDomain: "abc.adobelab2025.com",
});
```

Step 2: orgId: it represents the IMS Organization Id.

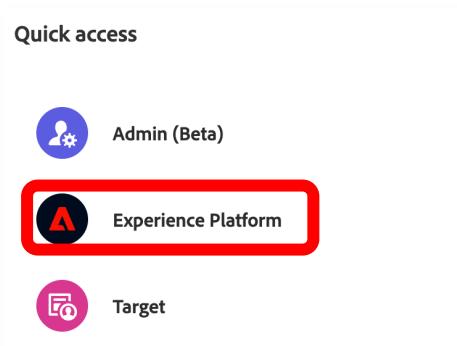
To retrieve the IMS Organisation Id you need to navigate to the Adobe Target – **Administration / Implementation** section, **Account details**, select and copy the value for the org Id:



The copied value should be set as the value for the orgId parameter in the previous code snippet. In the case of the lab exercise the orgId is: 82C94E025B2385B40A495E2C@AdobeOrg

Step 3: datastreamId: the datastreamId from the Adobe Experience Platform – [Datastream section](#), connected to Adobe Target (e.g. bdb5cb8a-4496-4abd-8afc-e9396c1b1c27).

Step 3.1: Navigate back to the **Adobe Experience Cloud Home** page and select **Experience Platform** solution from the **Quick Access** panel:



Step 3.2: Select the **Datastreams** section from the left Panel and then, in the right **Datastreams** list click on the “**Copy to clipboard**” button associated with the datastream we want to use, to copy its **datastreamId**:

A screenshot of the Adobe Experience Platform Datastreams page. The left sidebar shows navigation options like Home, Workflows, Use Case Playbooks, Customer, Privacy, Data Management, Administration, and Data Collection. The 'Datastreams' section is selected. The main content area shows a table with columns 'Friendly Name' and 'Datastream ID'. A row for 'Demo Lab Datastream' has a 'bdb5cb8a-4496-4abd-8afc-e9396c1b1c27' in the 'Datastream ID' column and a 'Copy to clipboard' button to its right, which is highlighted with a red box. At the bottom of the table, there are 'Next >' and '< Previous' buttons. The 'Datastreams' section in the left sidebar is also highlighted with a red box.

The copied value should be set as the value for the datastreamId parameter in the previous code snippet.

Step 4: propertyToken: [Adobe Target property token](#) associated with the activity used to reduce the bundle size by selecting only the activities associated with it.

Step 4.1: To retrieve the **propertyToken** you need to navigate to the **Adobe Target Administration** page, **Properties** section.

Locate your property associated with the seat number (e.g. L537-seat01) and click on the **Code button** (</> icon):

The screenshot shows the Adobe Target Administration interface. On the left, there's a sidebar with links: Visual experience composer, Reporting, Scene7 configuration, Implementation, Properties (which is highlighted with a red box), Hosts, and Environments. The main area is titled 'Properties' and contains a table with columns: NAME, DESCRIPTION, CHANNEL, WORKSPACES, LAST UPDATED, and BY. One row is visible, showing 'CDN ex...', 'Web', '--', '02/10/2025, 01:39 PM', 'Iulian Ve...', and a red box around the '</>' icon. At the top right, there are buttons for 'Assign Properties to Workspaces' and '+ Create Property'. The top bar also shows 'Adobe Daniel Vivas' and 'Iulian Velea - APPROVER'.

Step 4.2: From the **Code** dialog select and copy the property token value: **Todo**

The screenshot shows a 'CDN experimentation' dialog. It has a 'Web' tab at the top right. Below it, there's a 'Implementation Code' section containing a script tag. Inside the script, there's a function 'targetPageParams()' that returns an object with a key 'at_property' set to the value '11e71d11-fc01-4d1e-f157-2783bc9c0e77'. This specific line is highlighted with a red box. At the bottom, there's a 'Copy' button and a 'Instructions for the web' section.

The copied value should be set as the value for the **propertyToken** parameter in the previous code snippet.

Step 5: oddEnabled: true/false;

- **true** – will do the decisioning inside the edge worker.
- **false** – the library will act as a proxy to the Adobe API and call the `/interact` endpoints for each decision request. No local decisioning is made.

For the lab exercise **oddEnabled** parameter value needs to be set to **true**.

Step 5 [not required]: rules: object containing the rules retrieved from the Adobe servers in AEP format. The rules can be retrieved by the developer and passed to the SDK client

as a configuration parameter. If the rules are not provided, the SDK Client will retrieve them from the Adobe servers.

For the lab exercise the rules bundle will be automatically fetched.

Evaluation and preparation of the rules

The **SDK Client** will return an object that contains the rules ready for evaluation and execution. If there is any problem the **Client** will throw an error.

The types of errors that can appear can be related to

- the library not being able to fetch the rules (might be a configuration problem in Akamai)
- the rules are not in the correct format
- the configuration Id's are missing or incorrect
- ...other problems

Exercise 3: Retrieving the consequences

Upon receiving a client request, the SDK will assess the request and determine the implications. The **Event** must be built on the EdgeWorker [in the XDM format](#).

In the code sequence below, we are constructing an event that enables decision-making based on the visitor allocation to the available Experiences, using the ECID cookie value, and the URL, by passing in the address.

```
12  const address = `${request.scheme}://${request.host}${request.url}`;
13  const event = {
14    "type": "decisioning.propositionFetch",
15    "personalization": {
16      "sendDisplayEvent": false
17    },
18    "xdm": {
19      "web": [
20        {
21          "webPageDetails": {
22            "URL": address
23          }
24        }
25      ]
26    }
27  }
28
const consequences = await client.sendEvent(event);
```

Step 1: The **client.sendEvent** function will receive the **event** and respond with the consequences.

In the **index.js** file go to the section where the event is created and add the following line of code to retrieve the consequences:

```
const consequences = await client.sendEvent(event);
```

Exercise 4: Sending the Display event

Step 1: To track displayed events, we need to send a notification to Adobe servers. Typically, this can be done in a fire-and-forget manner, as the result is not needed. Set “**sendDisplayEvent: true**” in the **event** passed to the **sendEvent** function to let the library automatically send the event.

Exercise 5: Working with the consequences

The consequences provided by the **client.sendEvent** function can be applied in the Edgeworkers code for A/B testing the origin website.

In the following code example, we are using the **HtmlRewritingStream**, extracting the consequence items, and applying the content of the consequence on the Origin response stream.

```
37  const consequenceItems = consequences.handle
38  .filter((payload) => payload.type === "personalization:decisions")
39  .map((consequence) => consequence.payload.map((payload) => payload.items))
40  .flat(2);
41
42
43  consequenceItems.forEach((item) => {
44    switch (item.schema) {
45      case "https://ns.adobe.com/personalization/json-content-item":
46        const { content } = item.data;
47        // match the DOM element based on the CSS selector
48        streamRewriter.onElement(content.selector, (el) => {
49          // example of working with the consequence
50          el.replaceWith(content.payload);
51        });
52        break;
53    }
54  });
55
```

Step1: Go to the line `// Exercise 5. Apply the consequences to the origin stream` and add the following lines of code to instantiate the **HTMLRewritingStream**

```
const streamRewriter = new HtmlRewritingStream();
```

Step 2: Go to the **forEach** loop in the **index.js** file and, after the line containing the comment `// match the DOM element based on the CSS selector`, add the following code sequence:

```
streamRewriter.onElement(proposition.selector, (el) => {
  el.replaceWith(proposition.payload);
});
```

Exercise 6: Returning the processed response

After creating the HTML rewriter processing stream, we need to return the Response to the client and the EdgeWorker code will apply the changes on the returned page.

```
61
62     return createResponse(
63         200,
64         {
65             "Powered-By": ["Akamai EdgeWorkers" + Date.now()],
66         },
67         originResponse.body.pipeThrough(streamRewriter),
68     );
69
70
```

Step 1: In the **index.js** file go before the line marking the end of the **catch** block and add the following code snippet:

```
return createResponse(
  200,
  {
    "Powered-By": ["Akamai EdgeWorkers " + new Date().toString()],
  },
  originResponse.body.pipeThrough(streamRewriter),
);
```

Exercise 7: Deploy your code to the EdgeWorker

Step 1: Set the EdgeWorker ID

Open the **.env** file in the IDE and add the **EdgeWorker ID** provided for your seating number (e.g. 94606) as the value for the **EDGEWORKER_ID** variable:

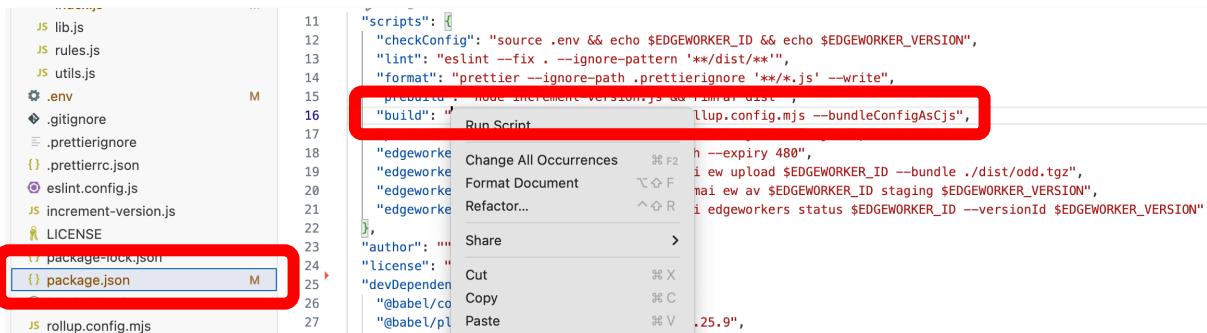


```
JS index.js M .env M package.json M
1 # used to identify the EdgeWorker to be uploaded
2 EDGEWORKER_ID=94606
3 # used to identify the EdgeWorker version to be uploaded and activated by the NPM commands
4 EDGEWORKER_VERSION=0.8
```

Leave the **EDGEWORKER_VERSION** unchanged as this will automatically be update during the **prebuild** step.

Step 2: Build your project

In the Visual Studio Code left navigation panel select the **package.json** file, go to the **scripts** section, and run the “**build**” script:



```
11 "scripts": [
12   "checkConfig": "source .env && echo $EDGEWORKER_ID && echo $EDGEWORKER_VERSION",
13   "lint": "eslint --fix . --ignore-pattern '**/dist/**/*',
14   "format": "prettier --ignore-path .prettierignore '**/*.js' --write",
15   "prebuild": "node increment-version.js && rimraf dist",
16   "build": "rollup.config.mjs --bundleConfigAsCjs",
17 ],
18   "edgeworke Change All Occurrences ⌘ F2
19   "edgeworke Format Document ⌘ ⌘ F
20   "edgeworke Refactor... ⌘ ⌘ R
21   "edgeworke Share ⌘ ⌘ ⌘
22   "author": "...",
23   "license": "...",
24   "devDependen Cut ⌘ X
25   "@babel/co Copy ⌘ C
26   "@babel/pl Paste ⌘ V
27   ".25.9",
```

There should be no errors reported in the terminal:

```
> lab-2025@0.0.1 build
> NODE_ENV=production rollup -c rollup.config.mjs --bundleConfigAsCjs

./src/index.js -> dist/main.js...
created dist/main.js in 61ms

> lab-2025@0.0.1 postbuild
> cd dist/ && tar -czvf odd.tgz main.js bundle.json

a main.js
a bundle.json
* Terminal will be reused by tasks, press any key to close it.
```

Step 3: Deploy your code

From the **package.json** file go to the line with the “**edgeworker:deploy**” script and run it. If the deploy is succesful you should see the following message in the terminal window:

```
> lab-2025@0.0.1 edgeworker:deploy
> source .env && akamai ew upload $EDGEWORKER_ID --bundle ./dist/odd.tgz
```

```
--- New version uploaded for EdgeWorker Id: 94606 ---
```

(index)	edgeWorkerId	version	checksum	createdBy	createdTime
0	94606	'0.36'	'a876ce92536ea1110ccab8c1fb24083f46106637f6511c6610a17b974f9988a9'	'ivelea@adobe.com'	'2025-03-10T12:22:17Z'

Step 4: Activate your deployment

From the **package.json** file go to the line with the “**edgeworker:activate**” script and run it in order to activate the new package deployed on the EdgeWorker.

For checking the status of the deployment you can run the “**edgeworker:status**” script. After the activation has been completed, this command should display in the terminal panel the **Complete** value in the **status** column. This usually takes a couple of minutes depending on the size of the new build package:

```
> lab-2025@0.0.1 edgeworker:status
> source .env && akamai edgeworkers status $EDGEWORKER_ID --versionId $EDGEWORKER_VERSION
```

```
--- The following EdgeWorker Activations currently exist for account: 148-NP2D, EdgeWorker Id: 94606, version: 0.14, activationId: any, network: any ---
```

(index)	edgeWorkerId	version	activationId	status	network	createdBy	createdTime
0	94606	'0.14'	35	'COMPLETE'	'STAGING'	'ivelea@adobe.com'	'2025-03-09T19:28:24Z'

Exercise 8: Test your new deployment

Step 1: Test the newly activated code package

Open a new Browser Tab and enter the `seatXX.adobelab2025.test.edgekey-staging.net` value in the address bar, depending on your seat number.

The page should be loaded and the result of the EdgeWorker processing should be embedded in the displayed version. Depending on the performed user allocation, you should be served with the result of applying Experience A or Experience B defined in the Target Activity.

To get the result of the other Experience you can open a new Incognito browser window and access the same URL.

Additional steps:

Exercise 9: Persisting the ECID cookie value

Adobe Target identifies unique visitors based on the ECID value that it's generated in the server. The ECID value provides a stable Experience selection and reporting.

The ECID value need to be persisted in the browser for each visitor in Cookies.

When the requests arrive from the browser we will see if the ECID cookie is present and add it to the Event object used for deciding the Experience.

```
// Exercise 10. Extract the ECID from the request cookie and if it's present add it to the request
const cookies = new Cookies(request.getHeader("Cookie"));
const ecid = cookies.get("X-ADOBEC-ECID");
if (ecid) {
  event.xdm.identityMap = [
    {
      ECID: [
        {
          id: ecid,
          authenticatedState: "ambiguous",
          primary: true,
        },
      ],
    },
  ];
}
```

When returning the response, we make sure that the ECID cookie is set so that we persist it across the sessions.

```
25 // Exercise 10. Persist the ECID in a cookie in the browser
26 const ecidNamespace: T = consequences.handle.find(
27   (payload: T) =>
28     payload.type === "identity:result" &&
29     payload.payload[0].namespace.code === "ECID",
30 );
31 const ecidValue = ecidNamespace.payload[0].id;
32 streamRewriter.onElement("body", (el): void => {
33   el.append(
34     `<script> document.cookie = "X-ADOBEC-ECID=${ecidValue};SameSite=None; Secure"; </script>`,
35   );
36 });
37 }
```

Additional steps:

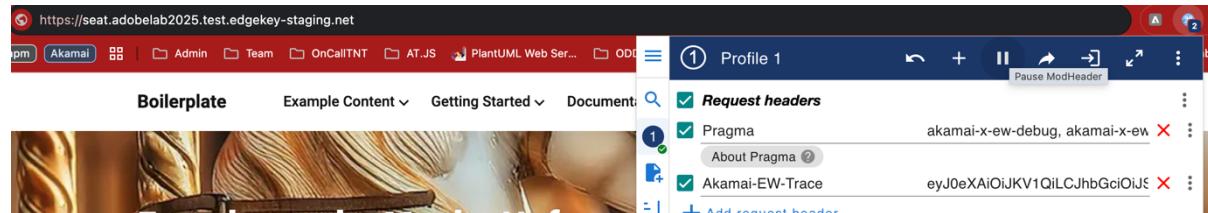
Exercise 10: Debugging your new deployment

Akamai Edgeworkers communicate the log messages through the response headers attached to the response. To activate the debug functionality, we need to set two headers to be sent along with the webpage request inside the ModHeader chrome extension.

Step 1) Inside Google Chrome find the ModHeader extension

Step 2) The **Pragma** header specifies the type of logs and debug information that will be returned. For the Lab set the header name Pragma and the value the following list:
akamai-x-ew-debug, akamai-x-ew-debug-rp, akamai-x-ew-debug-subs, akamai-x-ew-subworkers

For a reference of values please see [the Akamai documentation](#)



Step 3) Go to the project and run the command “**edgeworker:auth**” from the **package.json** file. This will provide you with an authentication token. Copy paste just the token value highlighted in the screenshot below.

Step 4) Go to Google Chrome and open the ModHeader extension. Set the Header name **Akamai-EW-Trace** and the value copied from the console in step 3. This will allow the Edgeworker to output the log in the response headers.

Copy paste the token the red border inside the ModHeader chrome extension.

```

increment-version.js
LICENSE
package.json
package-lock.json
README.md
rollup.config.mjs
External Libraries
Scratches and Consoles
scripts > edgeworkerauth
/usr/aenascut/.nvm/versions/node/v22.13.0/bin/npm run edgeworker:auth

> lab-2025@0.0.1 edgeworker:auth
> akamai edgeworkers auth --expiry 480

----- Add the following request header to your requests to get additional trace information. ----

Akamai-EW-Trace: eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ2Q10IiINoJiwiia2lKIjo2lCJhY2wiOlsIlyoiXSwizXheTjoNxQxOTA4NTkwLCJpYXQiOjE3NDE4NzkzOTAsImp0aSI6IJFkZTQwZDMyLThzNjMtNGY2My05OTAwLWR1ODg4N2E3MWR1OSJ9.JQKzK0gPMLMzaei93CmkkobEyiFC4Uf8fq4vRQ-xqtqELApccympGJBCAo0i1YKsfaoXbdGVqlRMU52UYv0e0ER6jG3d4pC50UloP29C_qJm0lmZv5-Qv7npizw_WQlfwFaH1Y4rtRsIoFMCpmSc-xpJD0EvVkfelra10v6ADYgJpaJPR21EnQ5SvrijeaAm
yBnLnII2amyA8NDTSkL_rJE-FWFHra2UEhnt7LMboByuqfnCGVkr77Ov2a9u1w81duyCine_3YxQHRyR0wEN7Dcj7tHuPS8ywJ_SHKHBEAFXL5wduWtkhJpozFFX8X1RcbcK1W-g

```

Appendix

Edge worker runtime environment limitations

- <https://techdocs.akamai.com/edgeworkers/docs/resource-tier-limitations>

responseProvider	Maximum memory usage for responseProvider	2 MB	4 MB	6 MB
	Maximum CPU time for responseProvider	100 milliseconds	200 milliseconds	300 milliseconds
	Maximum wall time for responseProvider	4 seconds	8 seconds	10 seconds
	Maximum number of HTTP sub-requests allowed for responseProvider	50	50	50
	Maximum number of HTTP sub-requests allowed in parallel for responseProvider	5	5	10
	Maximum wall time per HTTP sub-request during the execution of the responseProvider event handler	4 seconds	8 seconds	10 seconds
	Maximum response size per HTTP sub-request during the execution of the responseProvider event handler	5 MB	5 MB	8 MB

- <https://techdocs.akamai.com/edgeworkers/docs/limitations>

Compressed size for a code bundle	512 KB
Uncompressed size for a code bundle	1 MB
Maximum response header size for HTTP sub-requests	8192 bytes
Maximum body size for responses created using the <code>respondWith()</code> method	2048 characters
Maximum body size for responses from origin to an EdgeWorkers function, using <code>httpRequest()</code>	128 KB when using <code>json()</code> or <code>text()</code> to buffer the entire response. 5 MB when using <code>body</code> to read the response as a stream.
Maximum body size for responses from an EdgeWorkers function to a browser when the response is passed through as a string	100,000 bytes if you pass a string to <code>createResponse</code> in <code>responseProvider</code> No direct limit if you pass a <code>stream</code> to <code>createResponse</code> in <code>responseProvider</code> .
Maximum body size for responses from an EdgeWorkers function to a client browser	2048 characters if you use <code>request.respondWith()</code> in the <code>onClientRequest</code> or <code>onClientResponse</code> event handler.
Maximum body size when <code>request.text()</code> or <code>request.json()</code> is called in the <code>responseProvider</code> event handler	16 KB
Maximum streaming request body size when <code>request.body</code> is called in the <code>responseProvider</code> event handler	1 MB
Maximum size of the response status and header for responses generated using <code>createResponse</code>	8 KB

Workflow for the rules package:

1. A user of Adobe Target creates **Activities**.
2. The rules package is uploaded to an AWS S3 bucket.
3. Content from the S3 bucket is delivered through a CDN.
4. The rules package is accessible for download via the [adobe.io API](#).

