Workshop guide

Creado por Pablo Moreno, modificado por última vez Hace 31 minutos

This guide contains a few blocks that the attendees can follow to learn how to build an App Builder extension.

Access & Credentials

- ***Adobe Experience Cloud IMS Organization:*** *AEP Partner Shared Training Sandbox*
- Access to ***Adobe Developer Console*** & ***App Builder:*** *https://developer.adobe.com*
- ***Adobe Commerce admin:*** https://staging-5em2ouy-efo2rc3kezxte.eu-4.magentosite.cloud/admin
- **Base source code**: https://github.com/adobe/app-builder-workshop/tree/skeleton

Technical prerequisites

- node 14...18 LTS (Recommended v18)
- npm 9 or higher
  - @adobe/aio-cli >= 9.3.0

If you plan to develop App Builder locally using *aio app run --local, then the following additional requirements are needed:*

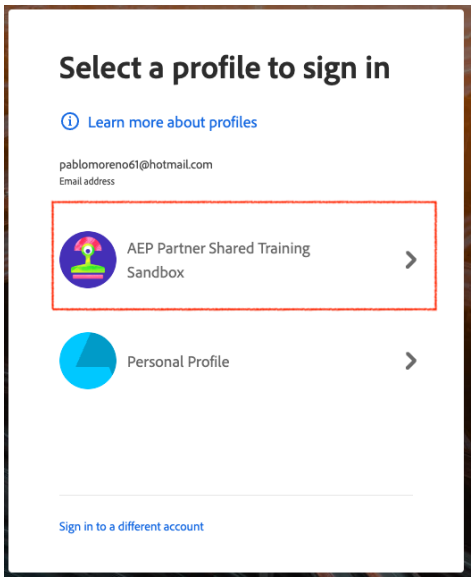
- Java 11 or Higher
- Docker 24 or higher

Official reference: https://developer.adobe.com/app-builder/docs/getting_started/

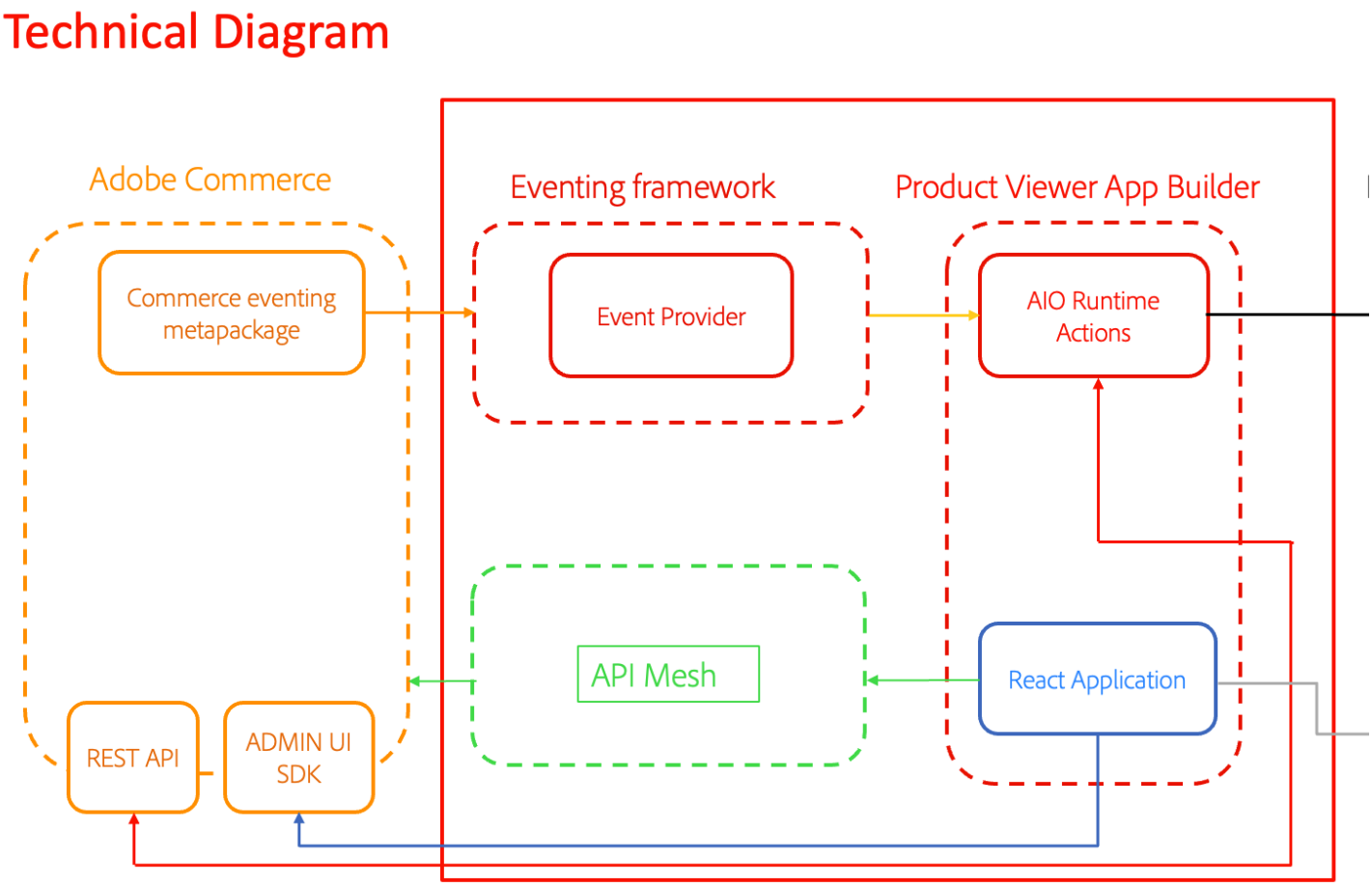
Add yourself to the Developer Console

Accessing the developer console for the very first time will require to set up your account:

1. Using the guest e-mail you provided to the organizers, navigate to developer.adobe.com and click on the *Sign-in* link.
2. Fill out the form with your e-mail and click on Continue.
3. Click on the *Reset your password* link.
4. You will receive a verification code in your inbox, copy it onto the Verification code page and introduce your new password.
5. After setting your new password, you will receive a confirmation e-mail. you will be able to sign up using your guest e-mail.
    a. If you have multiple profiles and the sign-up was properly done, you should see a profile selection window when accessing the Sign-in link from the developer.adobe.com page.

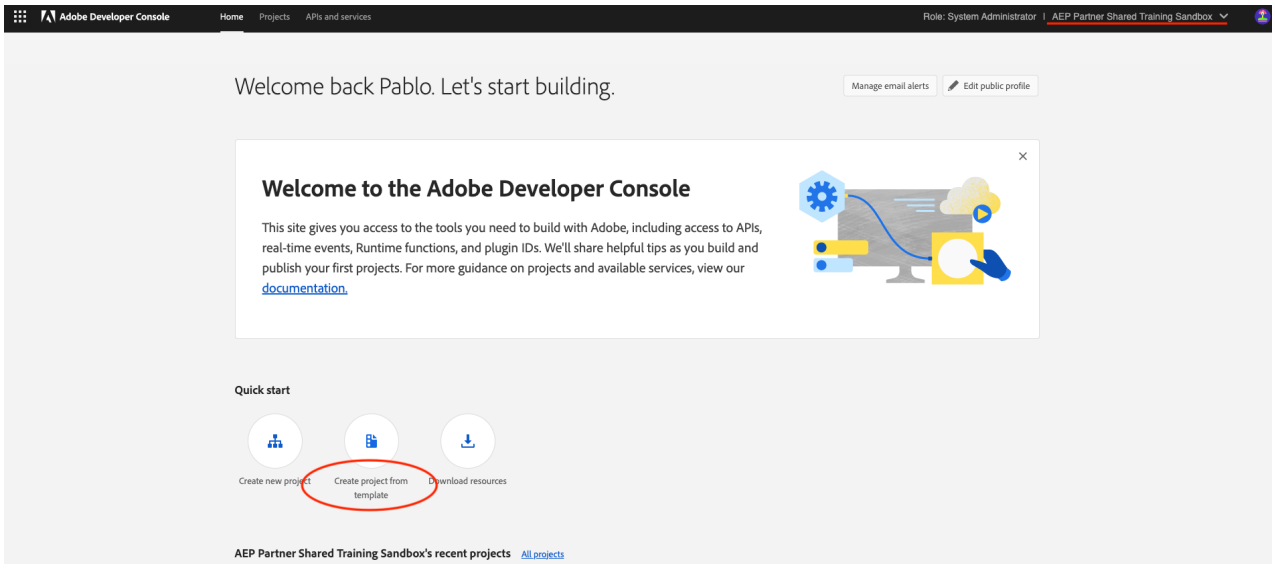


Project diagram

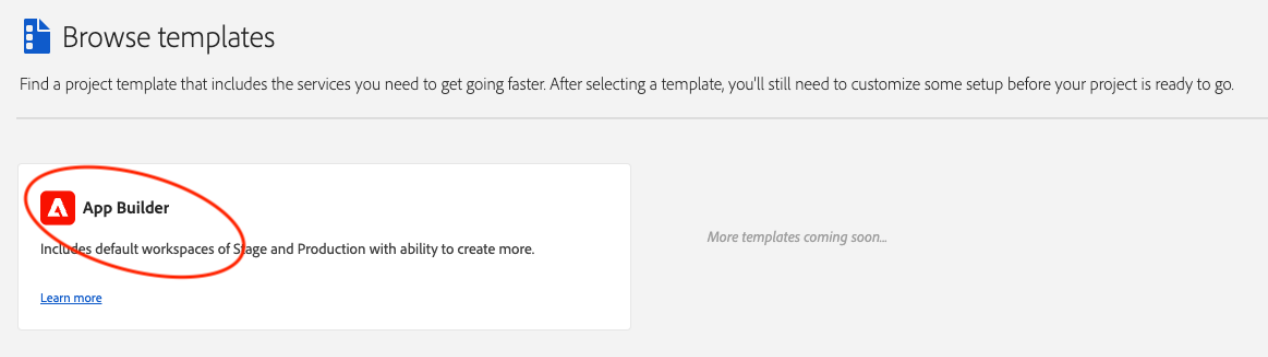


Setup your App Builder project

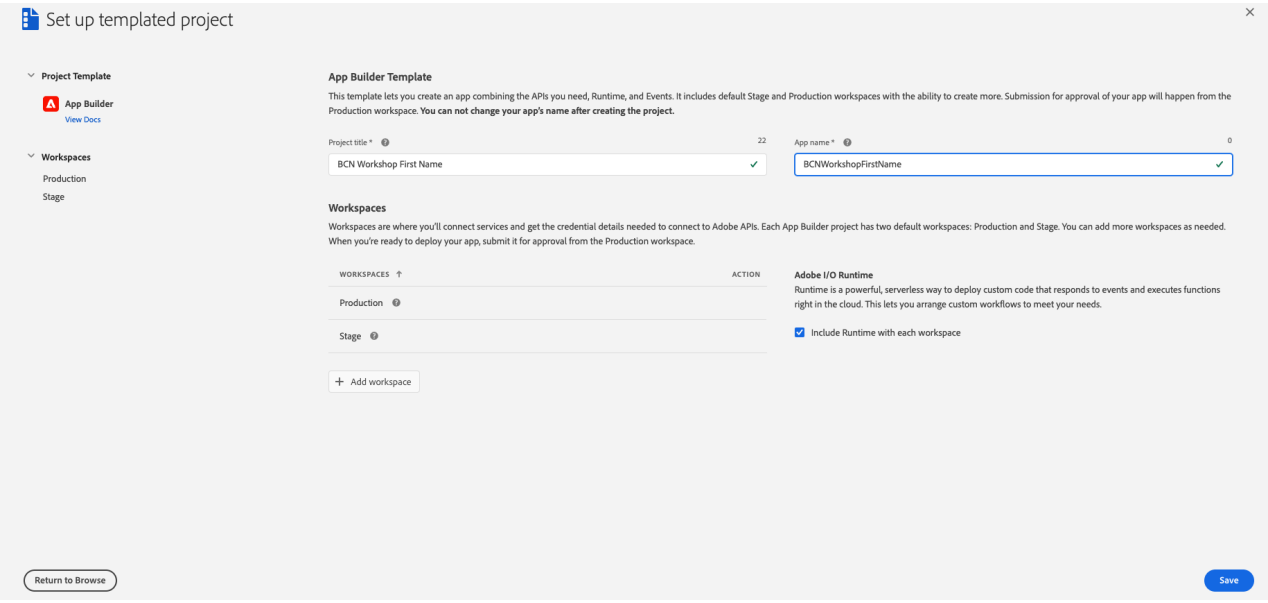
1. Log in to the Adobe Developer Console and select the ***"AEP Partner Shared Training Sandbox"*** organization from the dropdown menu in the top-right corner.



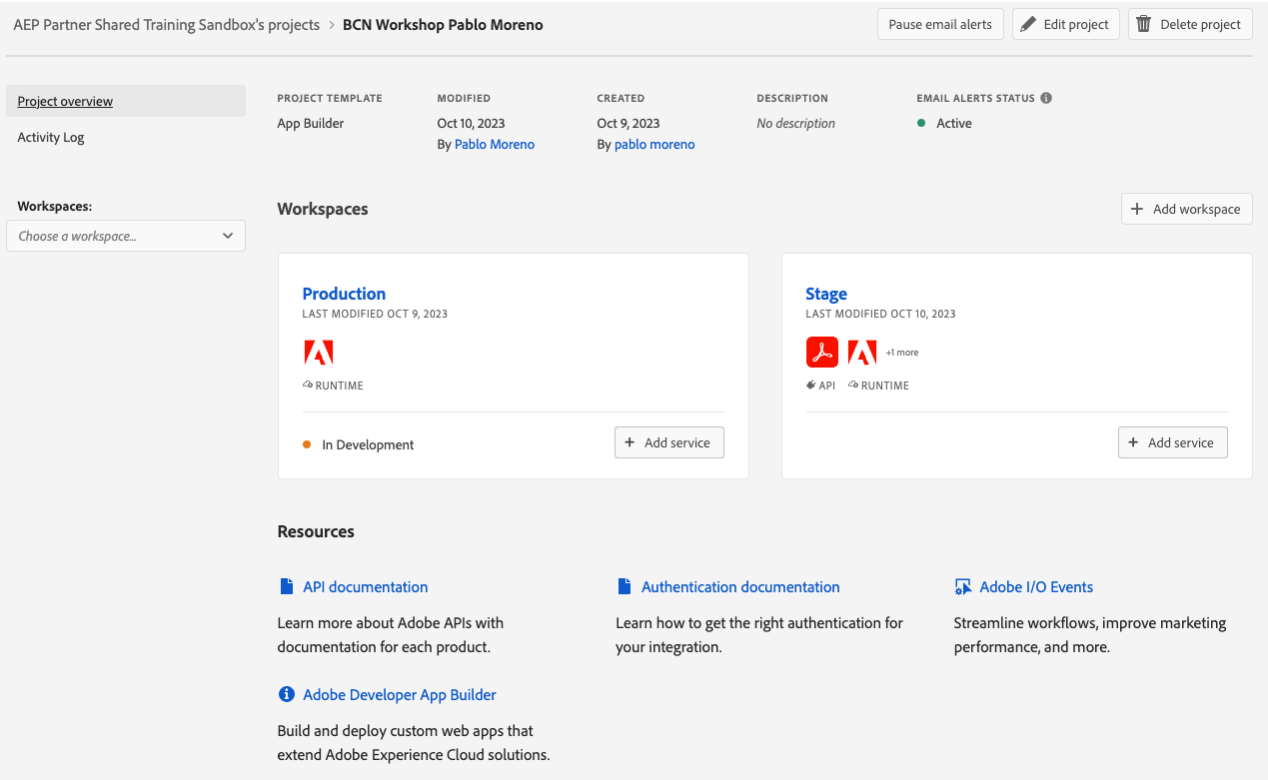
2. Click **Create project from template** and select *App Builder*.



3. Specify a project title **BCN Workshop <Your name>** and an app name. Make sure to check the option **Include Runtime with each workspace**. Click **Save**.



4. The Console will create 2 workspaces for you, so you should see Production and Stage. **This session is going to be focused on Production workspace only**.



Prepare your code

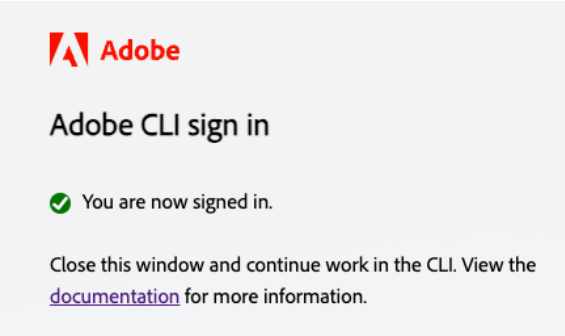
1. Go to the path where you want to store the source code and **clone the base repo**.

```
git clone -b skeleton https://github.com/adobe/app-builder-workshop.git
```

2. Log in to *aio cli* tool using your guest account.
   a. Run the following command

```
➜  <you-project-path> ✗ aio login
```

   b. A new page should be opened in your browser, asking for sign-in. Use the credentials you provided during the "Add yourself to the Developer Console". Once logged in, you should see a screen as follows:



   c. Back to the CLI, you should verify that "Login successful" appears in the *aio login* response.

3. Select the active organization:

```
➜  <you-project-path> ✗ aio console org select
? Select Org: AEP Partner Shared Training Sandbox
Org selected AEP Partner Shared Training Sandbox
You are currently in:
1. Org: AEP Partner Shared Training Sandbox
2. Project: <no project selected>
3. Workspace: <no workspace selected>
```

4. Then, select the active project:

```
➜  <you-project-path> ✗ aio console project select
? Select Project: BCN Workshop Pablo Moreno
Project selected BCNWorkshopPabloMor
You are currently in:
1. Org: AEP Partner Shared Training Sandbox
2. Project: BCNWorkshopPabloMor
3. Workspace: <no workspace selected>
```

5. Select the active workspace:

```
➜  <you-project-path> ✗ aio console workspace select
? Select Workspace: Production
Workspace selected Production
You are currently in:
1. Org: AEP Partner Shared Training Sandbox
2. Project: BCNWorkshopPabloMor
3. Workspace: Production
```

6. Finally, **import the App Builder project configuration** for the active namespace selected using the aio cli tool:
   a. *This command will automatically generate 2 new files: .aio and .env.*

```
➜  <you-project-path> ✗ aio app use
You are currently in:
1. Org: <no org selected>
2. Project: <no project selected>
3. Workspace: <no workspace selected>

? Switch to a new Adobe Developer Console configuration: A. Use the global Org / Project / Workspace configuration:
    1. Org: AEP Partner Shared Training Sandbox
    2. Project: BCNWorkshopPabloMor
    3. Workspace: Production

✔ Successfully imported configuration for:
1. Org: AEP Partner Shared Training Sandbox
2. Project: BCNWorkshopPabloMor
3. Workspace: Production.
```

7. **Within the root path of your project, find the *.env.dist* file, copy its content and append it to the *.env* file**. Don't worry about the empty variables, you will be asked to fill them out later on.
   Here you have an example of how the .env.dist looks like.

```
# To fill during API Mesh implementation
API_MESH_URL=

# To fill after configuring PDF Embed API
SERVICE_PDF_EMBED_API_CLIENT_ID=

# Commerce auth. To fill before integrating Commerce Events
COMMERCE_BASE_URL=
COMMERCE_CONSUMER_KEY=
COMMERCE_CONSUMER_SECRET=
COMMERCE_ACCESS_TOKEN=
COMMERCE_ACCESS_TOKEN_SECRET=

# Randomizer. To fill before integrating Commerce Events
ATTENDEE_FULLNAME=

RANDOMIZER_URL=
RANDOMIZER_FIRSTNAME=
RANDOMIZER_LASTNAME=
RANDOMIZER_COMPANY_NAME=
RANDOMIZER_API_KEY=
```

8. Install NPM dependencies found in your *package.json* file:

```
npm install
```

Populate React UI component

Goals

- Render a basic React UI using static content.

Add a Products grid React component

1. **Create a new React component** in */web-src/src/components/Products.js*.

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/
import {
    Content,
    Heading,
    IllustratedMessage,
    TableView,
    TableHeader,
    TableBody,
    Column,
    Row,
    Cell,
    View,
    Flex,
    ProgressCircle
} from '@adobe/react-spectrum'
import React from "react";

export const Products = props => {
    const isLoadingCommerceProducts = false
    const commerceProducts = [{name:'Sample Product', id:1}]
    const productsColumns = [
        {name: 'Name', uid: 'name'},
        {name: 'SKU', uid: 'sku'},
        {name: 'Id', uid: 'id'},
        {name: 'Price', uid: 'price'},
```

```
        {name: 'Type ID', uid: 'type_id'},
        {name: 'Created At', uid: 'created_at'},
        {name: 'Updated At', uid: 'updated_at'},
    ]

    function renderEmptyState() {
        return (
            <IllustratedMessage>
                <Content>No data available</Content>
            </IllustratedMessage>
        )
    }

    return (
        <View>
            {isLoadingCommerceProducts ? (
                <Flex alignItems="center" justifyContent="center" height="100vh">
                    <ProgressCircle size="L" aria-label="Loading…" isIndeterminate/>
                </Flex>
            ) : (
                <View margin={10}>
                    <Heading level={1}>Fetched products from Adobe Commerce</Heading>
                    <TableView
                        overflowMode="wrap"
                        aria-label="products table"
                        flex
                        renderEmptyState={renderEmptyState}
                        minHeight="static-size-1000"
                    >
                        <TableHeader columns={productsColumns}>
                            {column => <Column key={column.uid}>{column.name}</Column>}
                        </TableHeader>
                        <TableBody items={commerceProducts}>
                            {product => (
                                <Row key={product['sku']}>{columnKey => {
                                    let content = product[columnKey];
                                    return <Cell key={`${product['sku']}-${columnKey}`}>{content}</Cell>
                                }}
                                </Row>
                            )}
                        </TableBody>
                    </TableView>
                </View>
            )}
        </View>
    )
}
```

2. Import and add the newly created component to the Browser route of your App React component:

```
<Route index element={<Products/>} />
```

Validation

Display React UI using static content

Fetched products from Adobe Commerce

Name	SKU	Id	Price	Type ID	Created At	Updated At
Sample Product		1				

Query products from Commerce API using API Mesh

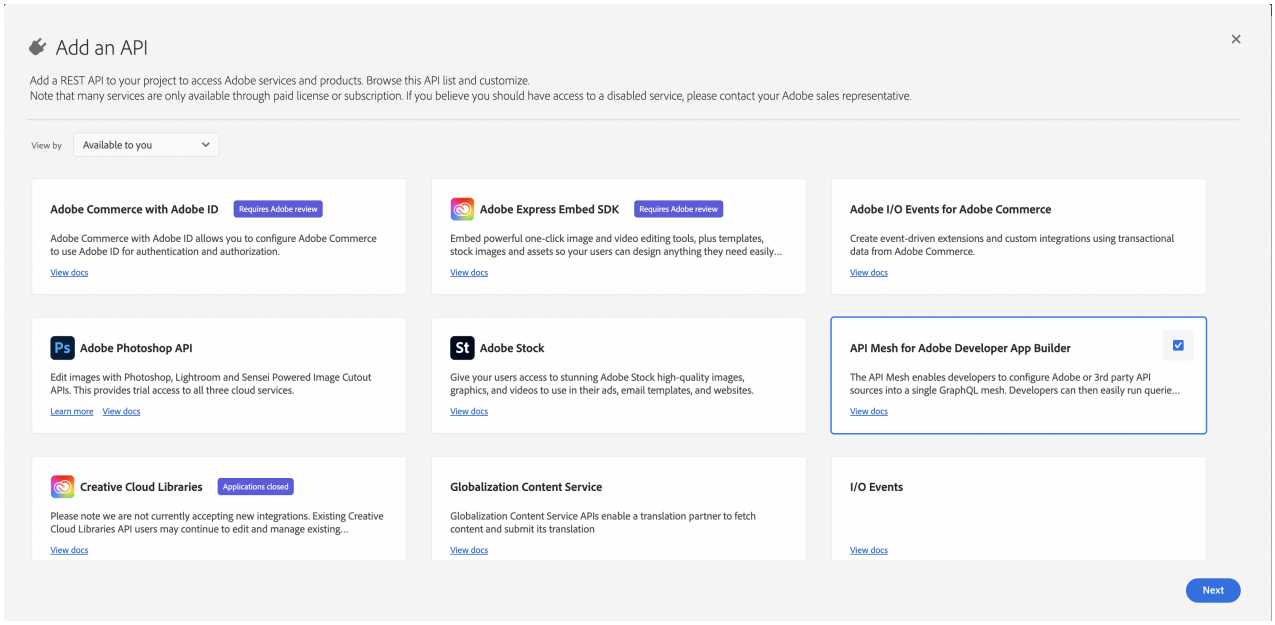
Goals

- Use Commerce graphQL API call to query products from the Commerce catalog using API Mesh.
- Replace static content with API Mesh response.

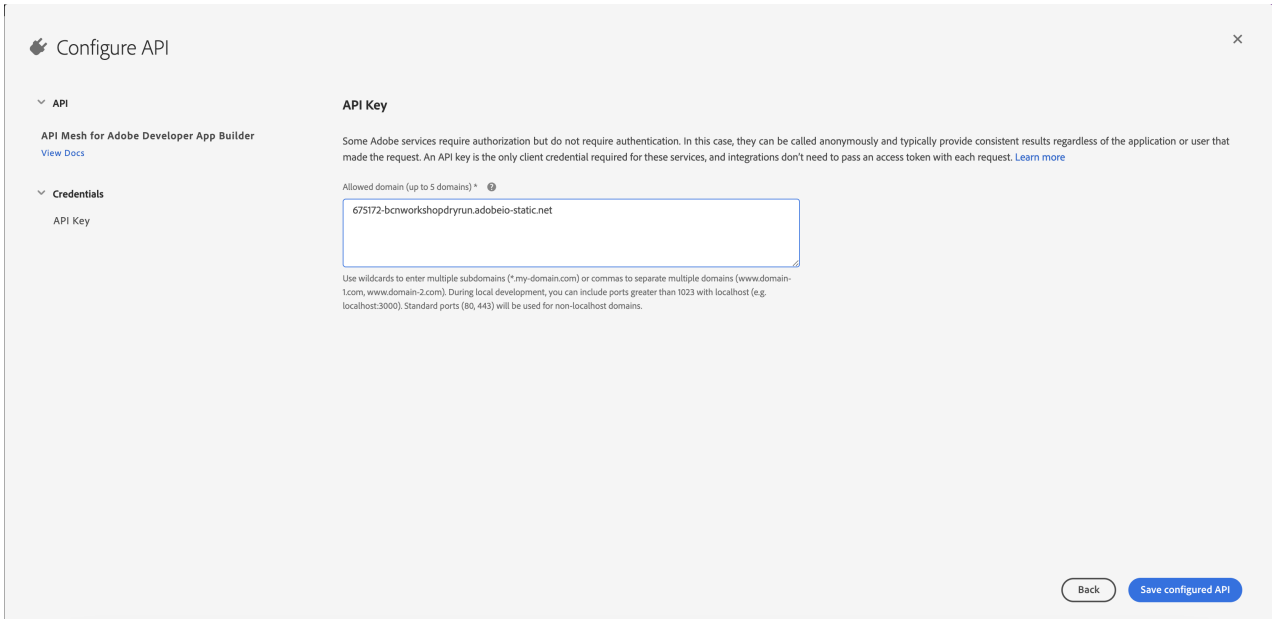
Add API Mesh support

Configure API Mesh for Production environment

1. Go to your workspace and click on Add Service → API. On that page, select the **API Mesh for Adobe Developer App Builder**.



2. **Enter the allowed domain base URL**, it should be your App Builder URL, following this example:
   To fill out the form, get the *AIO_runtime_namespace* environment value from your .env file and append ".adobeio-static.net" to it.



3. Click on **Save configured API.**

4. Create ***/mesh.config.json*** with the following content:

```
{
  "meshConfig": {
    "sources": [
      {
        "name": "CommerceGRAPHQL",
        "handler": {
          "graphql": {
            "endpoint": "https://staging-5em2ouy-efo2rc3kezxte.eu-4.magentosite.cloud/graphql/"
          }
        }
      }
    ],
    "responseConfig": {
      "CORS": {
        "methods": [
          "POST",
          "GET",
          "PUT",
          "HEAD",
          "OPTIONS"
        ],
        "origin": "*"
      }
```

```
    }
  }
}
```

5. Install AIO API Mesh plugin.

```
aio plugins:install @adobe/aio-cli-plugin-api-mesh
```

6. Create a Mesh endpoint using the previous configuration file to get a **Mesh endpoint** for the current workspace.

```
aio api-mesh create mesh.config.json
```

7. In your *.env* file, add the following property ***API_MESH_URL***:

```
API_MESH_URL=<YOUR_API_MESH_URL>
```

Code your API Mesh integration

1. Add a new **function named *callMesh*** inside the */web-src/src/utils.js*:

```
export async function callMesh(url, pageSize, currentPage) {
  const res = await fetch(url, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(
      {
        query: `query {
          products(
            search: ""
            pageSize: ${pageSize}
            currentPage: ${currentPage}
            sort: {name: ASC}
          ) {
            items {
              id,
              sku,
              name,
              type_id,
              created_at,
              updated_at,
              price {
                regularPrice {
                  amount {
                    value
                  }
                }
              }
              pdf_file
            }
          }
        }`
      }
    )
  })
  return await res.json();
}
```

2. **Create a hook** in */web-src/src/hooks/useCommerceProducts.js* to call API Mesh:

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/
import { useEffect, useState } from 'react'
```

```
import { callMesh } from '../utils'

export const useCommerceProducts = props => {
    const [isLoadingCommerceProducts, setIsLoadingCommerceProducts] = useState(true)
    const [commerceProducts, setCommerceProducts] = useState([])

    const fetchCommerceProducts = async () => {

        const commerceProductsResponse = await callMesh(process.env.API_MESH_URL, props.pageSize, props.currentPage)
        console.log("%s products returned by Commerce REST API", commerceProductsResponse.data.products.items.length);


        commerceProductsResponse.data.products.items.forEach((item) => {
            item.price = item?.price?.regularPrice?.amount?.value;
        });

        setCommerceProducts(commerceProductsResponse.error ? [] : commerceProductsResponse.data.products.items)
    }

    useEffect(() => {
        fetchCommerceProducts().then(() => setIsLoadingCommerceProducts(false))
    }, [])

    return { isLoadingCommerceProducts, commerceProducts }
}
```

3. Finally, in your Products component, replace the static calls with **useCommerceProducts** call to load the content from API Mesh:

```
const {isLoadingCommerceProducts, commerceProducts} = useCommerceProducts({...props, pageSize: 20, currentPage: 1})
```

Validation

Fetch products using API Mesh

Fetched products from Adobe Commerce

Name	SKU	Id	Price	Type ID	Created At	Updated At
Abominable Hoodie	MH09	196	69	configurable	2023-10-10 09:46:34	2023-10-10 09:46:34
Adrienne Trek Jacket	WJ08	1322	57	configurable	2023-10-10 09:47:09	2023-10-10 09:48:17
Advanced Pilates & Yoga (Strength)	240-LV08	49	18	downloadable	2023-10-10 09:46:02	2023-10-10 09:46:02
Aeon Capri	WP07	1867	48	configurable	2023-10-10 09:47:27	2023-10-10 09:47:27
Aero Daily Fitness Tee	MS01	564	24	configurable	2023-10-10 09:46:48	2023-10-10 09:46:48
Aether Gym Pant	MP11	873	74	configurable	2023-10-10 09:46:59	2023-10-10 09:46:59
Affirm Water Bottle	24-UG06	15	7	simple	2023-10-10 09:45:44	2023-10-10 09:45:44
Aim Analog Watch	24-MG04	36	45	simple	2023-10-10 09:45:53	2023-10-10 09:45:53
Ajax Full-Zip Sweatshirt	MH12	244	69	configurable	2023-10-10 09:46:35	2023-10-10 09:46:35
Ana Running Short	WSH10	2023	40	configurable	2023-10-10 09:47:30	2023-10-10 09:47:30
Angel Light Running Short	WSH06	1996	42	configurable	2023-10-10 09:47:29	2023-10-10 09:47:29
Antonia Racer Tank	WT08	1802	34	configurable	2023-10-10 09:47:23	2023-10-10 09:47:23
Apollo Running Short	MSH02	904	32.5	configurable	2023-10-10 09:46:59	2023-10-10 09:46:59
Arcadio Gym Short	MSH11	1021	20	configurable	2023-10-10 09:47:01	2023-10-10 09:47:01
Argus All-Weather Tank	MT07	700	22	configurable	2023-10-10 09:46:52	2023-10-10 09:46:52
Ariel Roll Sleeve Sweatshirt	WH09	1168	39	configurable	2023-10-10 09:47:05	2023-10-10 09:47:05
Artemis Running Short	WSH04	1973	45	configurable	2023-10-10 09:47:29	2023-10-10 09:47:29

------ Coffee break ------

Integrate PDF Embed API service

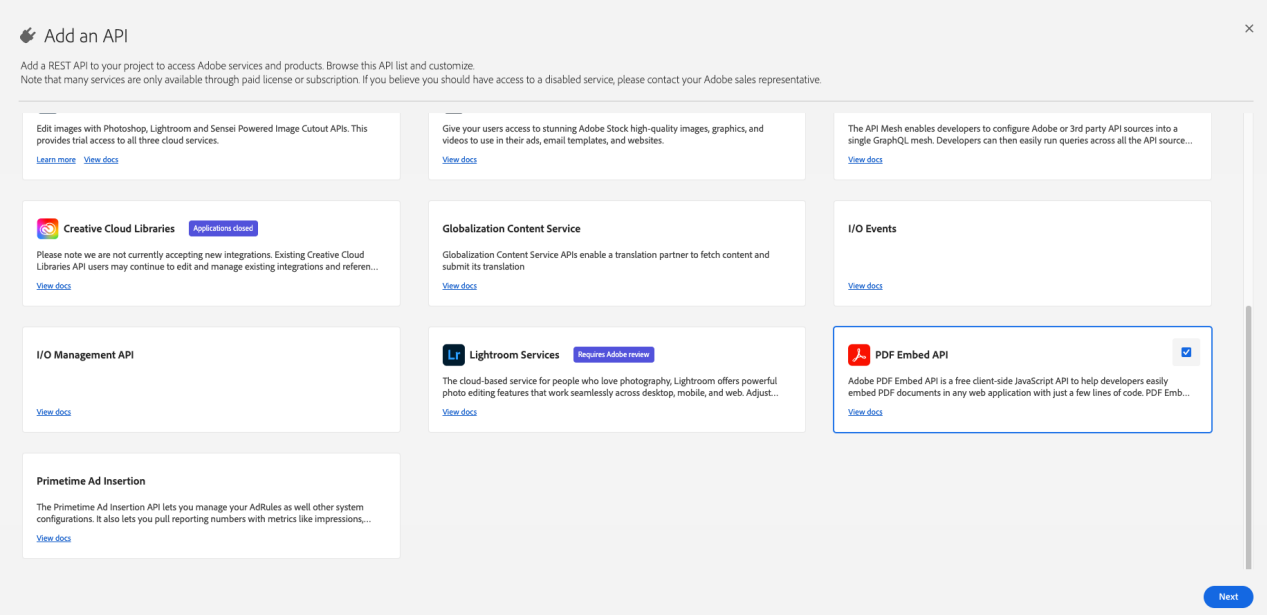
Goals

- Do a third-party API call against the PDF embed API to display a PDF file.
- Display a React Button to view the content of the PDF file.

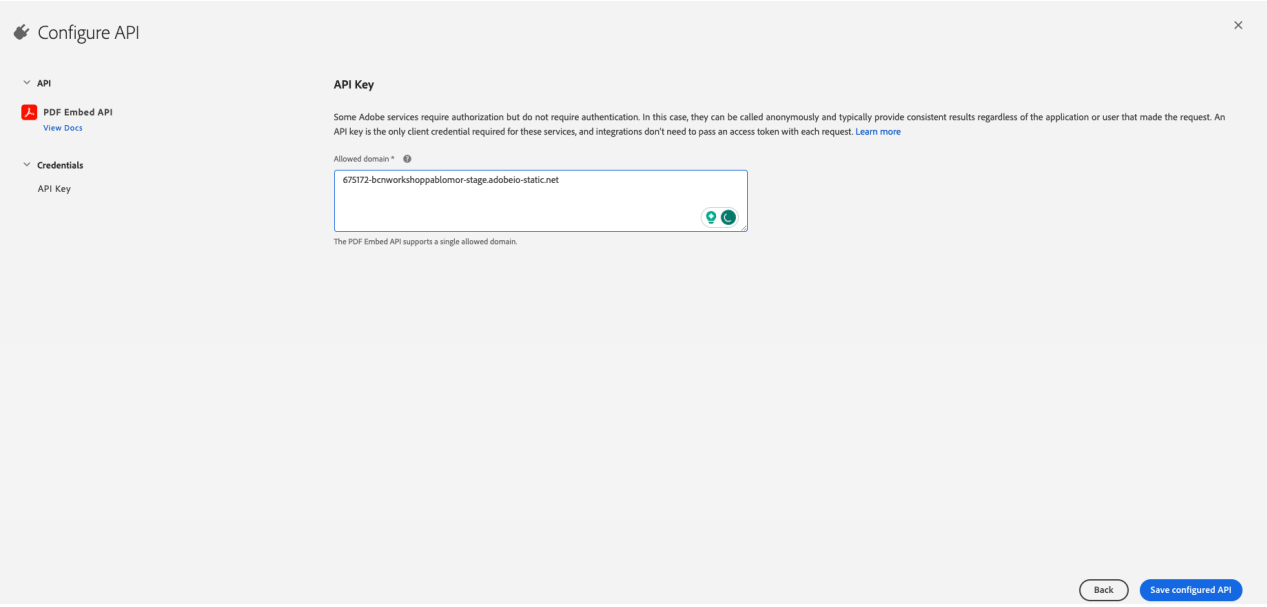
Add PDF Embed API support

Configure PDF Embed API for Production environment

1. Select the Production environment.
2. Click on Add Service → API and **select PDF Embed API** from the available options.



3. **Enter the allowed domain base URL**, it should be your App Builder URL, following this example:
   To fill out the form, get the *AIO_runtime_namespace* environment value from your .env file and append *".adobeio-static.net*" to it.

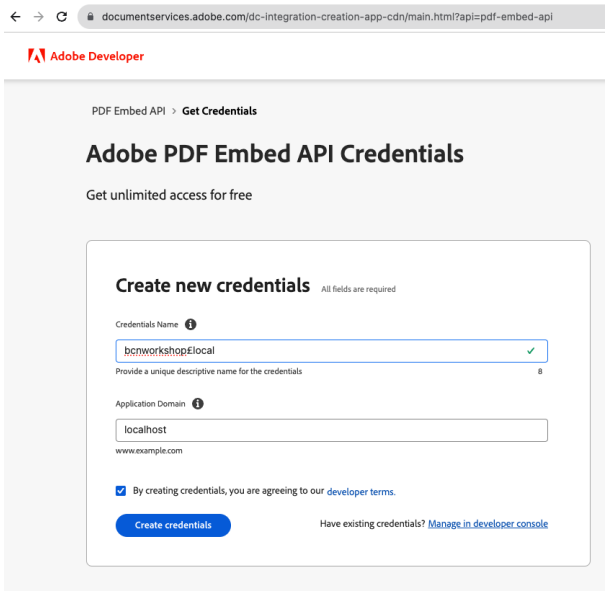


4. After **saving the configuration API**, you will be redirected to the PDF Embed API page.
5. From the PDF Embed API page, get the API Key and **replace** the variable **SERVICE_PDF_EMBED_API_CLIENT_ID** value from your .env file with the API key (*client id)*.

Configure PDF Embed API for the Local environment (Optional)

Only to be done if you plan to run your application locally (*aio app run*)

1. Go to https://documentservices.adobe.com/dc-integration-creation-app-cdn/main.html?api=pdf-embed-api
2. Fill the form with **"BCNWorkshop-Local-<YourName>"** name and **localhost** as a domain.



3. Finally, click on **Create credentials** and add the client ID generated to your .env file as SERVICE_PDF_EMBED_API_CLIENT_ID.
    a. *Note: **Do not replace the previous PDF Embed API key generated** for the Production workspace, instead keep both keys in your .env file and activate only the key needed on each case. Keep in mind this note before deploying the code.*

Code your PDF Embed API integration

1. Import PDF viewer into your */web-src/index.html*:

```
<script type="text/javascript" src="https://acrobatservices.adobe.com/view-sdk/viewer.js"></script>
```

2. Create a new file ***/web-src/src/libraries/ViewSDKClient.js*** to interact with the PDF Embed API service:

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/
class ViewSDKClient {
    constructor() {
        this.readyPromise = new Promise((resolve) => {
            if (window.AdobeDC) {
                resolve();
            } else {
                /* Wait for Adobe Acrobat Services PDF Embed API to be ready */
                document.addEventListener("adobe_dc_view_sdk.ready", () => {
                    resolve();
                });
            }
        });
        this.adobeDCView = undefined;
    }

    ready() {
        return this.readyPromise;
    }

    previewFile(divId, filename, viewerConfig) {
        const config = {
            clientId: process.env.SERVICE_PDF_EMBED_API_CLIENT_ID,
        };
        if (divId) { /* Optional only for Light Box embed mode */
            /* Pass the div id in which PDF should be rendered */
            config.divId = divId;
        }
        /* Initialize the AdobeDC View object */
        this.adobeDCView = new window.AdobeDC.View(config);
```

```
        const url = new URL(filename);

        /* Invoke the file preview API on Adobe DC View object */
        return this.adobeDCView.previewFile({
            content: {
                location: {
                    url: url.toString(),
                },
            },
            metaData: {
                fileName: filename,
                id: crypto.randomUUID().toString(),
            }
        }, viewerConfig);
    }
}

export default ViewSDKClient;
```

3. Import the ViewSDKClient library into your Products React component

```
import ViewSDKClient from "../libraries/ViewSDKClient";
```

4. Then, inside your Products React component, add a button handler to **handle the click on the *"View PDF"* button** at the beginning of the component.

```
const handleOnPress = (pdfFilename) => {
    if (pdfFilename === '') {
        return;
    }

    const viewSDKClient = new ViewSDKClient();
    viewSDKClient.ready().then(() => {
        viewSDKClient.previewFile("", pdfFilename, {
            embedMode: "LIGHT_BOX"
        });
    });
};
```

5. Add a new column to the table to display the PDF button:

```
{name: 'Attachments', uid: 'pdf_file'},
```

6. Within the same React component, add a button to trigger the PDF Embed API call inside the <TableBody> loop.

```
if (columnKey === 'pdf_file' && content) {
    content = (
        <Button type="button" variant={'primary'} onPress={() => handleOnPress(product[columnKey])}>View PDF</Button>
    );
}
```

7. Finally, add a layer to embed the PDF content. The content of the PDF file will be opened inside this layer

```
<div id="container" className="light-box-container"></div>
```

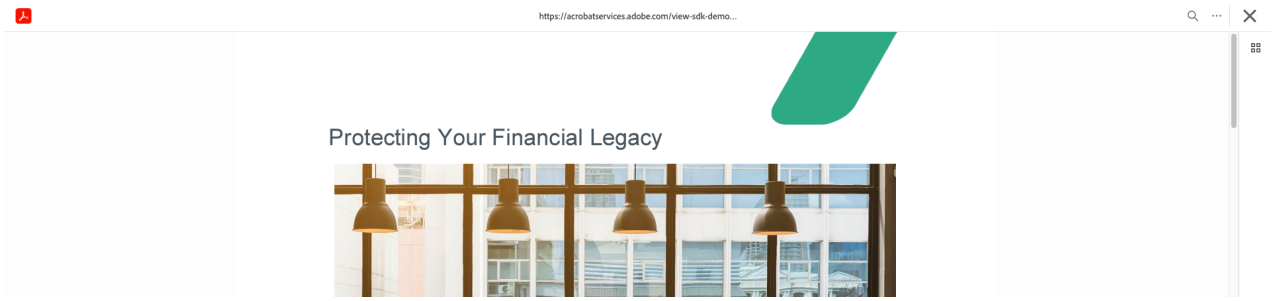
Validation

List of products with PDF button

Fetched products from Adobe Commerce

SKU	Name	Status	Price	Type ID	Created At	Updated At	Attachments
24-MB01	Joust Duffle Bag	1	34.3	simple	2023-10-10 09:45:34	2023-10-16 14:44:27	View PDF
24-MB04	Strive Shoulder Pack	1	32.01	simple	2023-10-10 09:45:35	2023-10-13 14:49:11	View PDF
24-MB03	Crown Summit Backpack	1	38	simple	2023-10-10 09:45:36	2023-10-10 09:45:36	
24-MB05	Wayfarer Messenger Bag	1	45	simple	2023-10-10 09:45:37	2023-10-10 09:45:37	

After clicking on the View PDF button



References

- https://developer.adobe.com/document-services/docs/overview/pdf-embed-api/

Deal with Commerce Events using the Eventing Framework

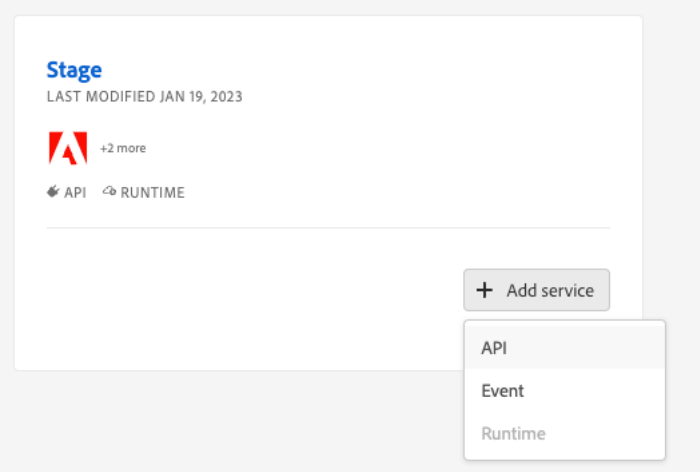
Goals

- Handle Commerce events.
- Create a runtime action that gets triggered by a product updated Commerce event:
  - Calls Randomizer API to generate a random number.
  - Call back to Commerce API to store the generated number as a catalog attribute for the product updated.

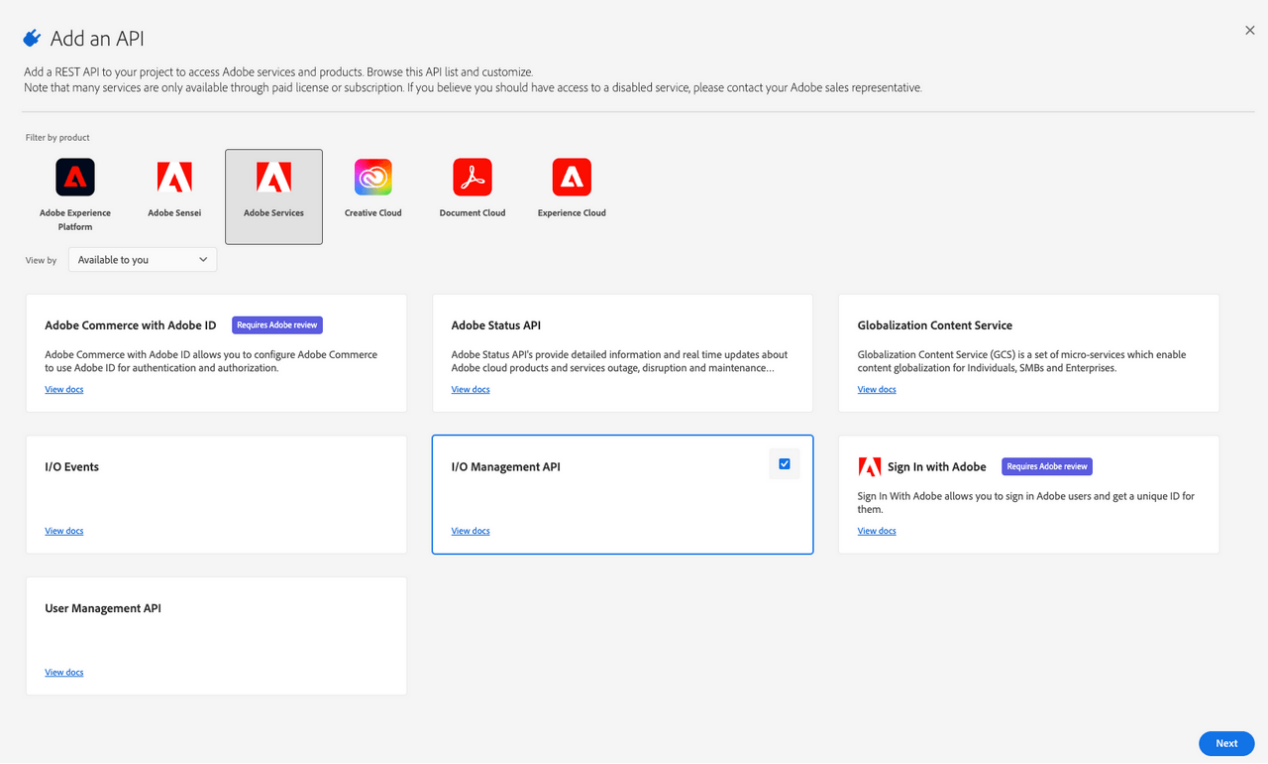
Add I/O Events support

Configure the required I/O Services in Production

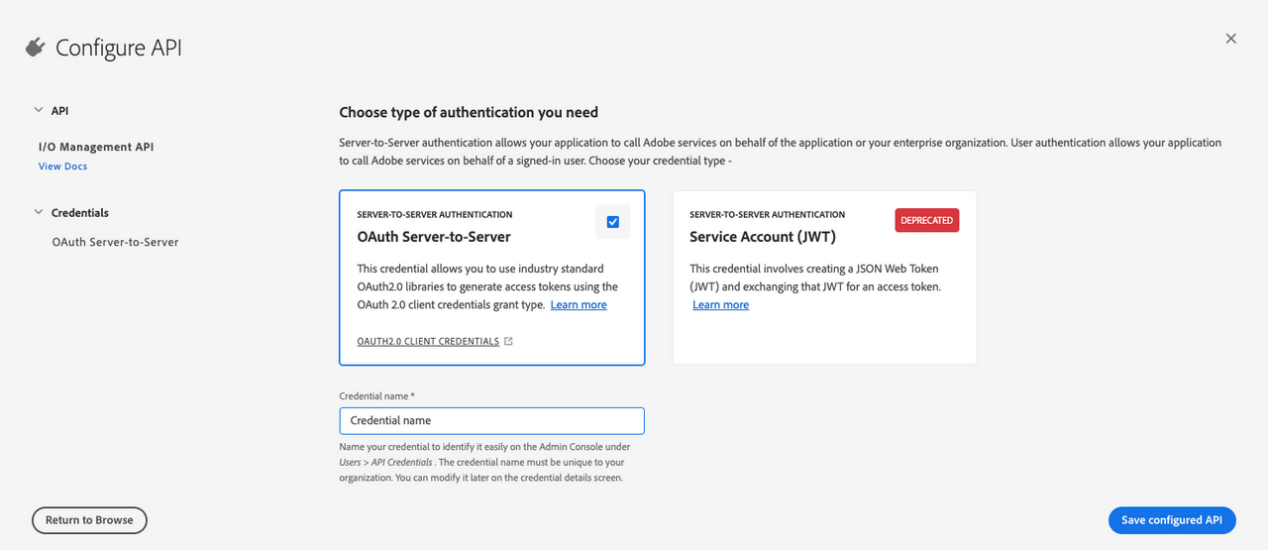
1. In your workspace, click the **Add service** pop-up menu and select **API**.



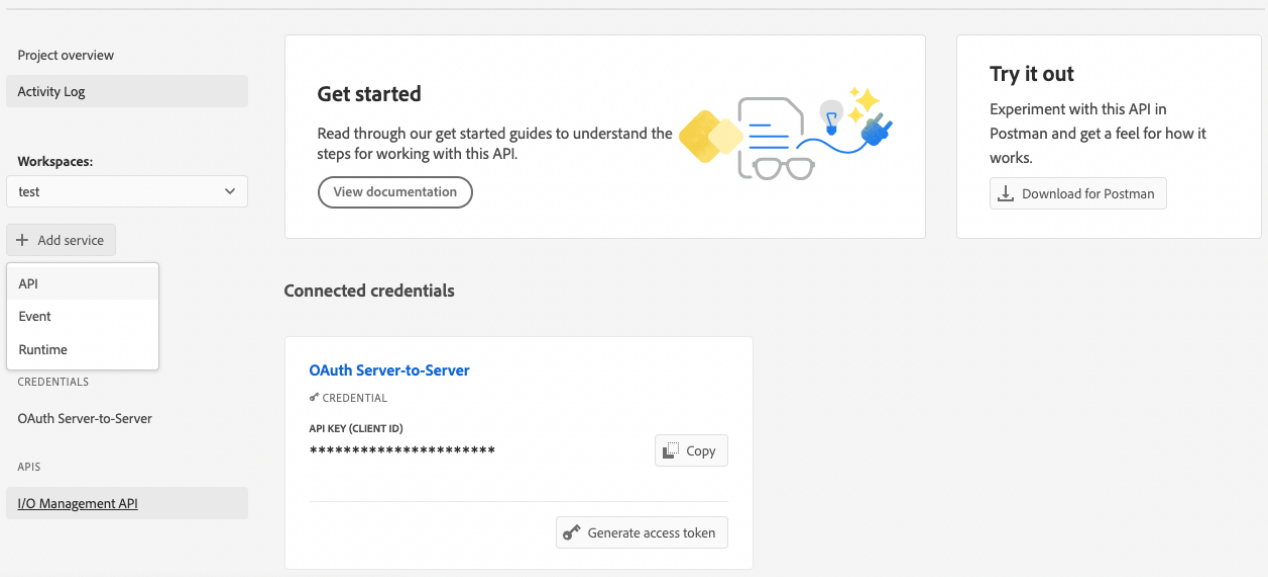
2. On the **Add an API** page, filter on **Adobe Services** and select **I/O Management API**. Then click **Next**.
   Note: This service is required to add the appropriate authentication scopes used by App Builder.



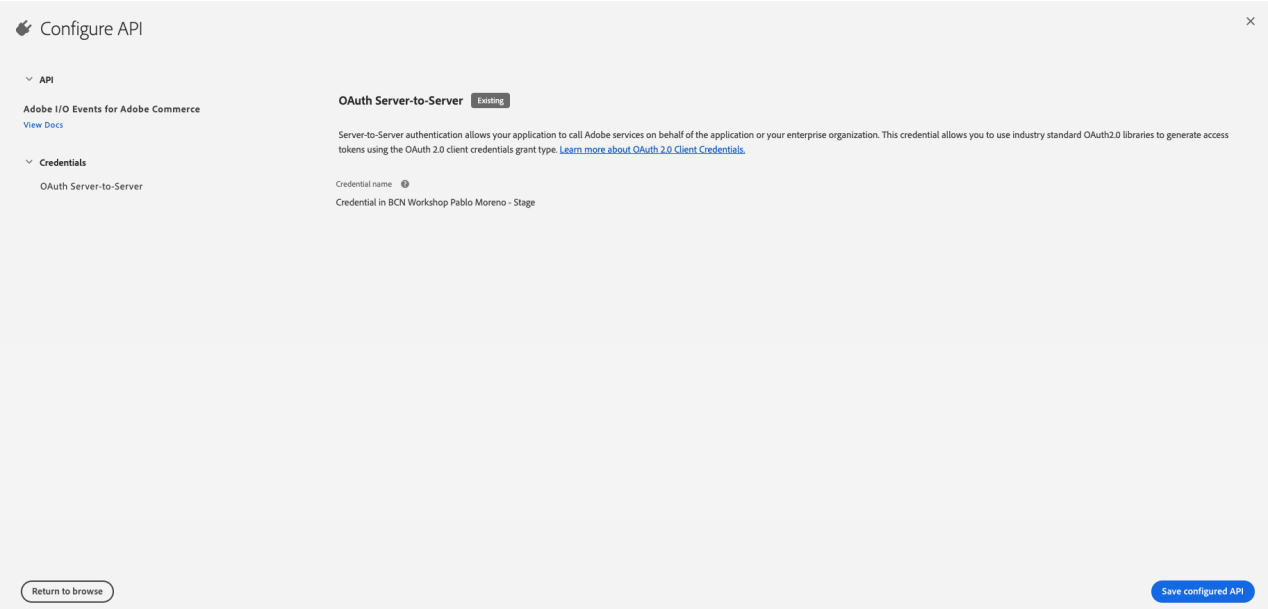
3. On the **Configure API** page, select the **OAuth Server-to-Server** option, fill the credential name and click **Save configured API**.



4. On the front page of your workspace, click the **Add service** pop-up menu and select **API**.

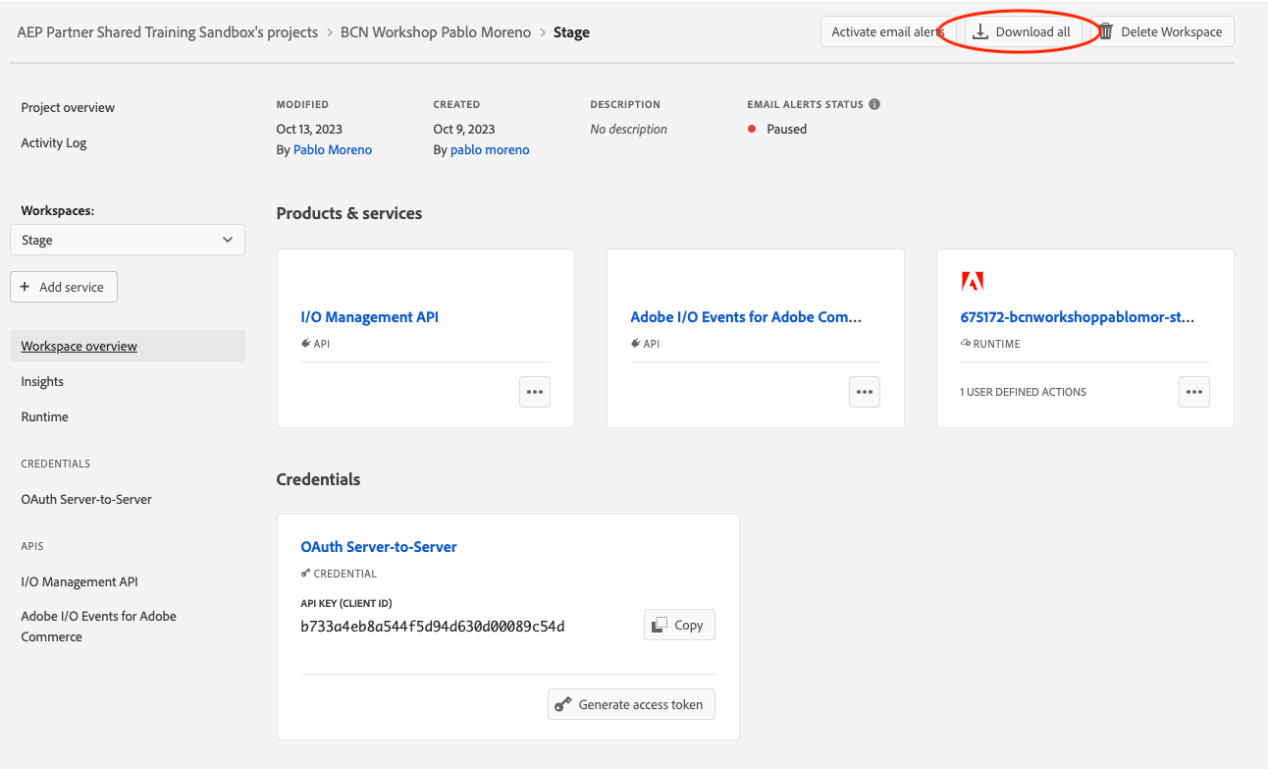


5. On the **Add an API** page, filter by **Experience Cloud** and select **Adobe I/O Events for Adobe Commerce**. OAuth Server-to-Server should be determined by default.



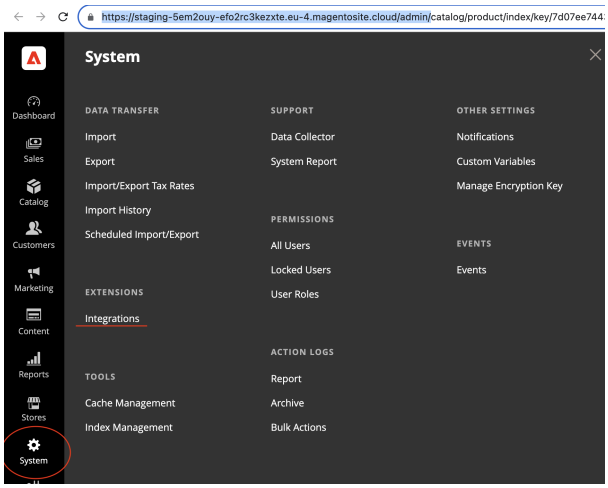
Then click Next and **Save configured API.**

6. Finally, you will need to **download the workspace configuration file**. Go to the overview page of your workspace and click the Download All button in the top-right corner.



Populate .env configuration

1. Go to the commerce instance (https://staging-5em2ouy-efo2rc3kezxte.eu-4.magentosite.cloud/admin/).
2. Navigate to System → Integrations



3. Click on Edit button to the ***BCN Workshop Oct*** **integration**

4. You will **find the Commerce credentials** under the Integration Details section. Use them to populate the **.env** file variables prefixed with ***COMMERCE_**.
    a. Add the Commerce URL https://staging-5em2ouy-efo2rc3kezxte.eu-4.magentosite.cloud/ to the **COMMERCE_BASE_URL** as well.
5. Populate the .env variables prefixed with ***RANDOMIZER_**** to allow calls to the 3rd party API that is going to be used from the event handler.

```
RANDOMIZER_URL=https://675172-bcnworkshoprandom-stage.adobeio-static.net/api/v1/web/randomizer-app-builder/eventRegister
RANDOMIZER_FIRSTNAME=<Your name>
RANDOMIZER_LASTNAME=<Your last name>
RANDOMIZER_COMPANY_NAME=<Your company name>
RANDOMIZER_API_KEY=1WantT0Random!ze
```

Handling the event in code

1. Go to the source code of your project.
2. Create an *index.js* javascript file under the *actions/randomizer* directory.

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/
import {getCommerceOauthClient} from "../oauth1a";

const { Core } = require('@adobe/aio-sdk')
const { errorResponse, checkMissingRequestInputs} = require('../utils')
import axios from 'axios';

export async function main(params) {
    const logger = Core.Logger('main', { level: params.LOG_LEVEL || 'info' });

    try {
        const requiredParams = ['COMMERCE_BASE_URL', 'RANDOMIZER_URL', 'RANDOMIZER_API_KEY', 'RANDOMIZER_FIRSTNAME', 'RANI
        const requiredHeaders = [];
        const errorMessage = checkMissingRequestInputs(params, requiredParams, requiredHeaders);

        if (errorMessage) {
            return errorResponse(400, errorMessage, logger);
        }

        const res = await axios.post(params.RANDOMIZER_URL, {
            "firstname": params.RANDOMIZER_FIRSTNAME,
            "lastname": params.RANDOMIZER_LASTNAME,
            "company_name": params.RANDOMIZER_COMPANY_NAME,
            "api_key": params.RANDOMIZER_API_KEY
        });

        const randomizedContent = await res.data;

        if (!randomizedContent?.random) {
```

```
            throw new Error("Random number not returned");
        }

        logger.info("Random number returned: %s", randomizedContent.random);

        const oauth = getCommerceOauthClient(
            {
                url: params.COMMERCE_BASE_URL,
                consumerKey: params.COMMERCE_CONSUMER_KEY,
                consumerSecret: params.COMMERCE_CONSUMER_SECRET,
                accessToken: params.COMMERCE_ACCESS_TOKEN,
                accessTokenSecret: params.COMMERCE_ACCESS_TOKEN_SECRET
            },
            logger
        )

        const attributeOptions = await oauth.get('products/attributes/random_numbers/options');

        logger.info("Got custom attribute options: %s", JSON.stringify(attributeOptions).toString());

        // Add random number to custom attributes if not already there
        const option = attributeOptions.find(o => o.label === randomizedContent.random.toString());

        if (!option) {
            await oauth.post('products/attributes/random_numbers/options', {
                option: {
                    "value": randomizedContent.random.toString(),
                    "label": randomizedContent.random.toString()
                }
            });
            logger.info("Random number added to custom attributes: %s", randomizedContent.random);
        }

        return {
            statusCode: 200,
            body: {
                "random": randomizedContent.random
            }
        }
    } catch (error) {
        logger.error("Cannot process catalog updated event: ", error);
        return errorResponse(500, error, logger);
    }
}
```

3. Add the following config **as a child of *runtimeManifest* property** in your *app.config.yaml* file:

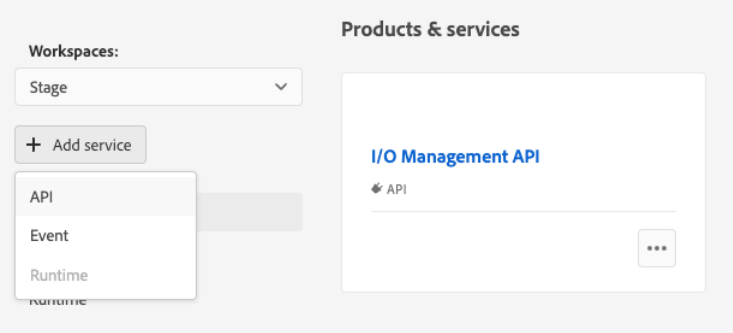
```
packages:
  BcnAppBuilderWorkshop:
    license: Apache-2.0
    actions:
      randomizer:
        function: actions/randomizer/index.js
        web: 'yes'
        runtime: 'nodejs:16'
        inputs:
          COMMERCE_BASE_URL: $COMMERCE_BASE_URL
          COMMERCE_CONSUMER_KEY: $COMMERCE_CONSUMER_KEY
          COMMERCE_CONSUMER_SECRET: $COMMERCE_CONSUMER_SECRET
          COMMERCE_ACCESS_TOKEN: $COMMERCE_ACCESS_TOKEN
          COMMERCE_ACCESS_TOKEN_SECRET: $COMMERCE_ACCESS_TOKEN_SECRET
          LOG_LEVEL: debug
          RANDOMIZER_FIRSTNAME: $RANDOMIZER_FIRSTNAME
          RANDOMIZER_LASTNAME: $RANDOMIZER_LASTNAME
          RANDOMIZER_COMPANY_NAME: $RANDOMIZER_COMPANY_NAME
          RANDOMIZER_URL: $RANDOMIZER_URL
          RANDOMIZER_API_KEY: $RANDOMIZER_API_KEY
        annotations:
          require-adobe-auth: true
          final: true
```

Configure the required event provider in Production

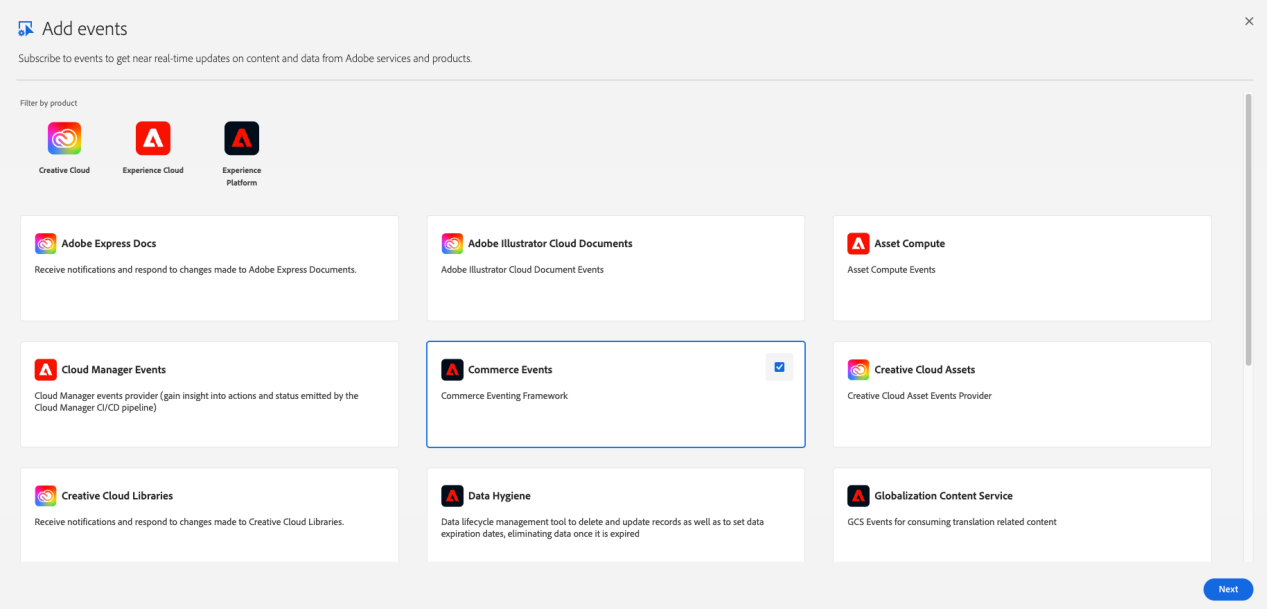
1. Deploy your code to the Production environment, so it's required in order to associate the event received to a handler.

```
aio app deploy
```

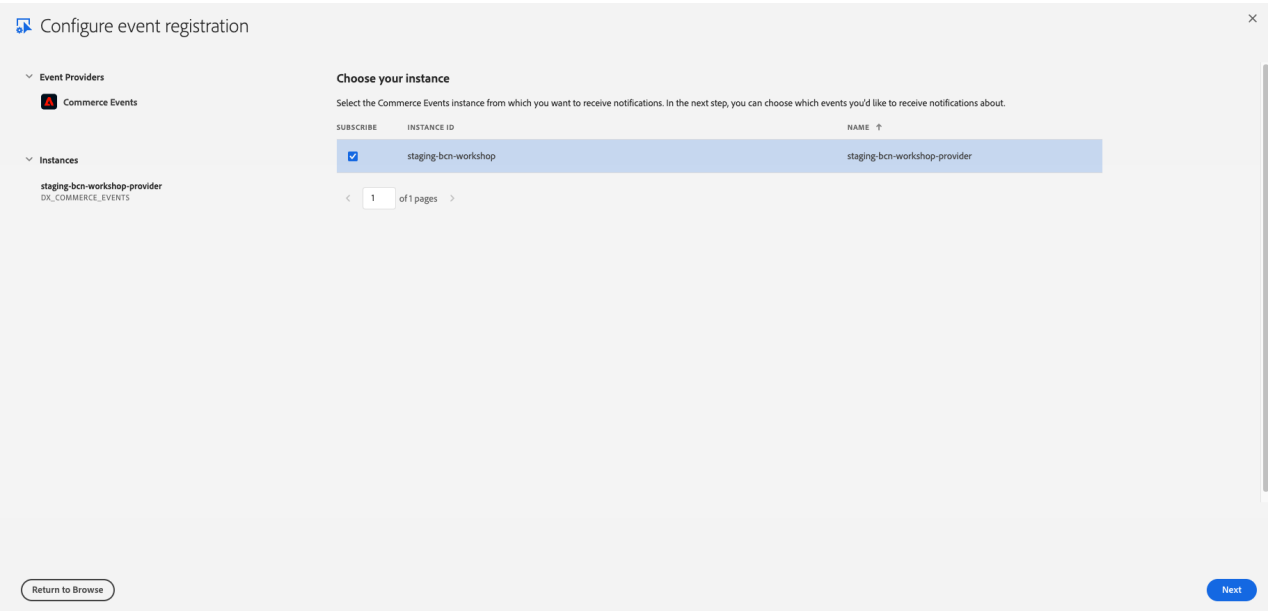
2. From your workspace. Click the **Add service** pop-up menu and select **Event**.



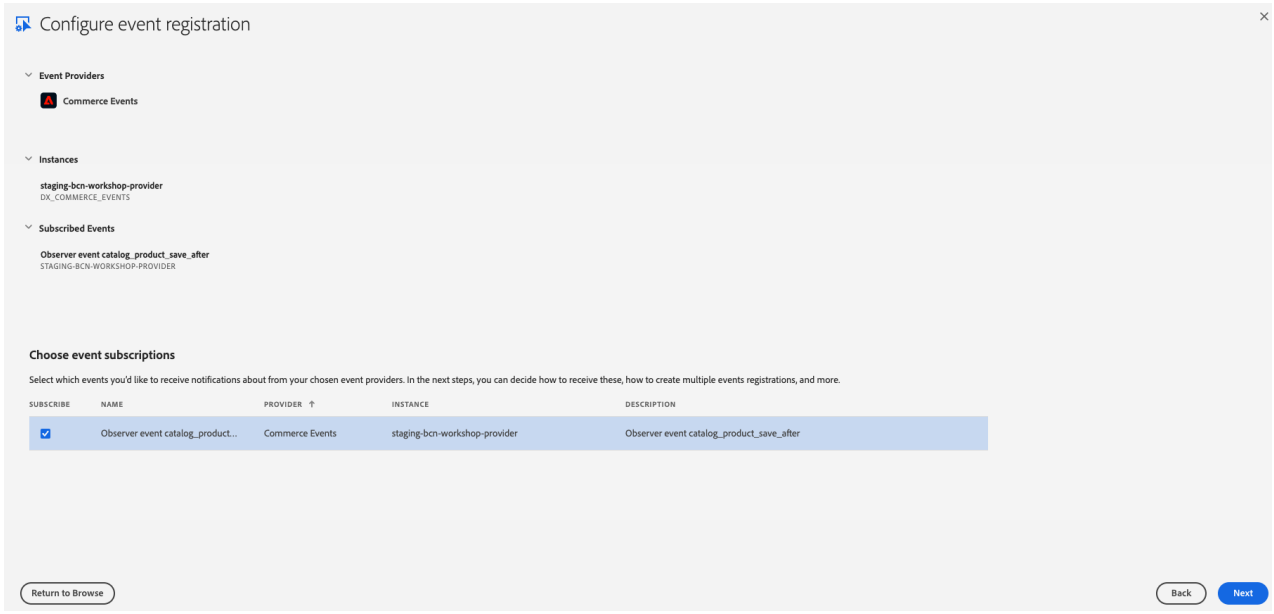
3. On the **Add events** page, select the event provider *"Commerce Events"* to handle events triggered by Adobe Commerce. Then click **Next**.



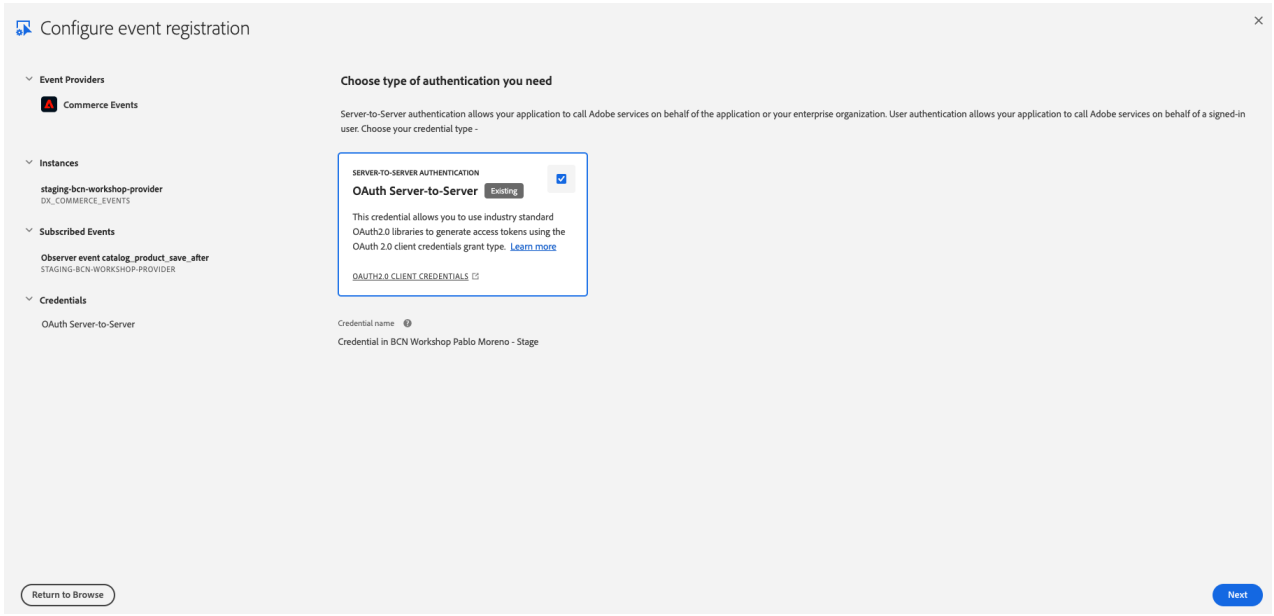
4. Select the commerce instance from which you want to receive events.



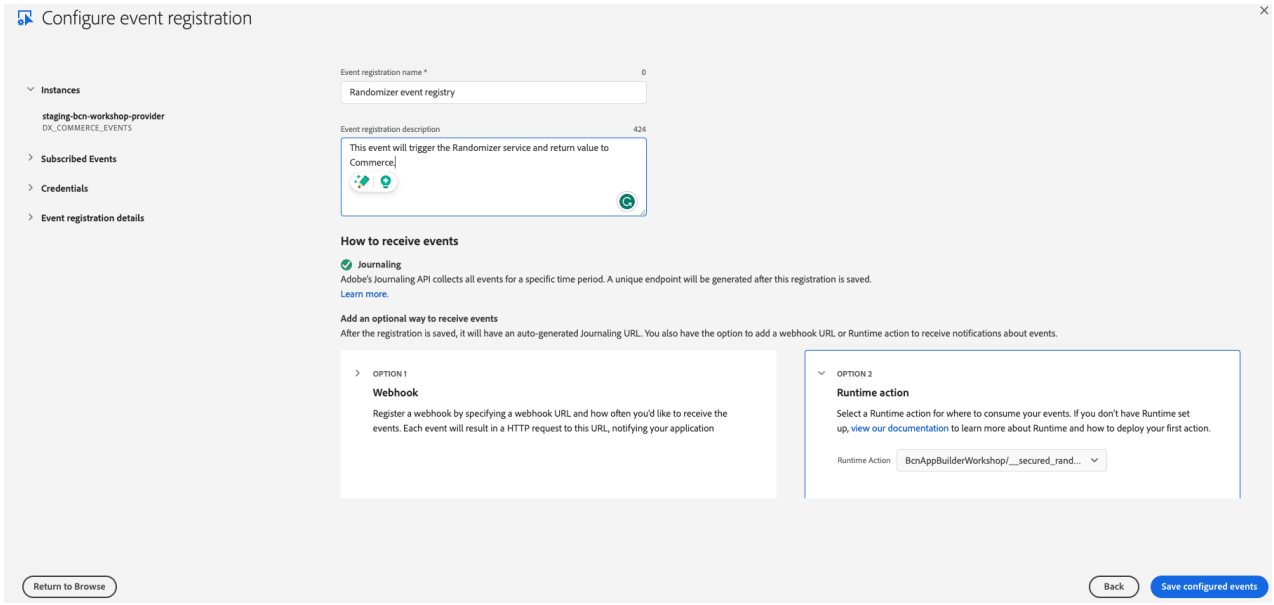
5. Select the events to subscribe to. In this case, we are interested in the catalog updated event. Then click **Next**.



6. Select *Oauth Server-to-Server* as the authentication method, selected by default.



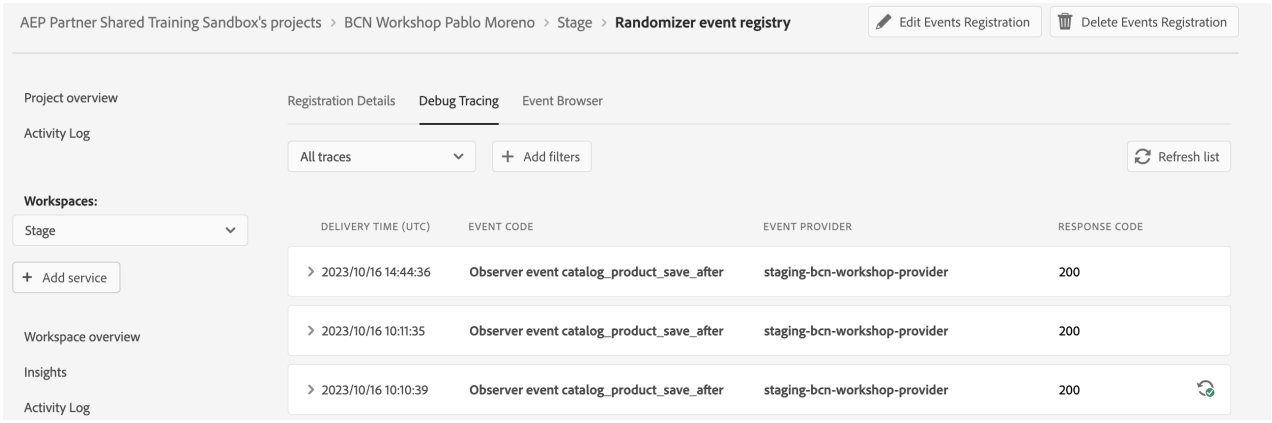
7. Update the **Event registration name** and **Event registration description** fields. In the **How to receive events** section, under **Option 2**, select the runtime action you created in the previous section and click **Save configured events**.
Note. If there are no actions in the Runtime action dropdown, review the step where you deployed the application



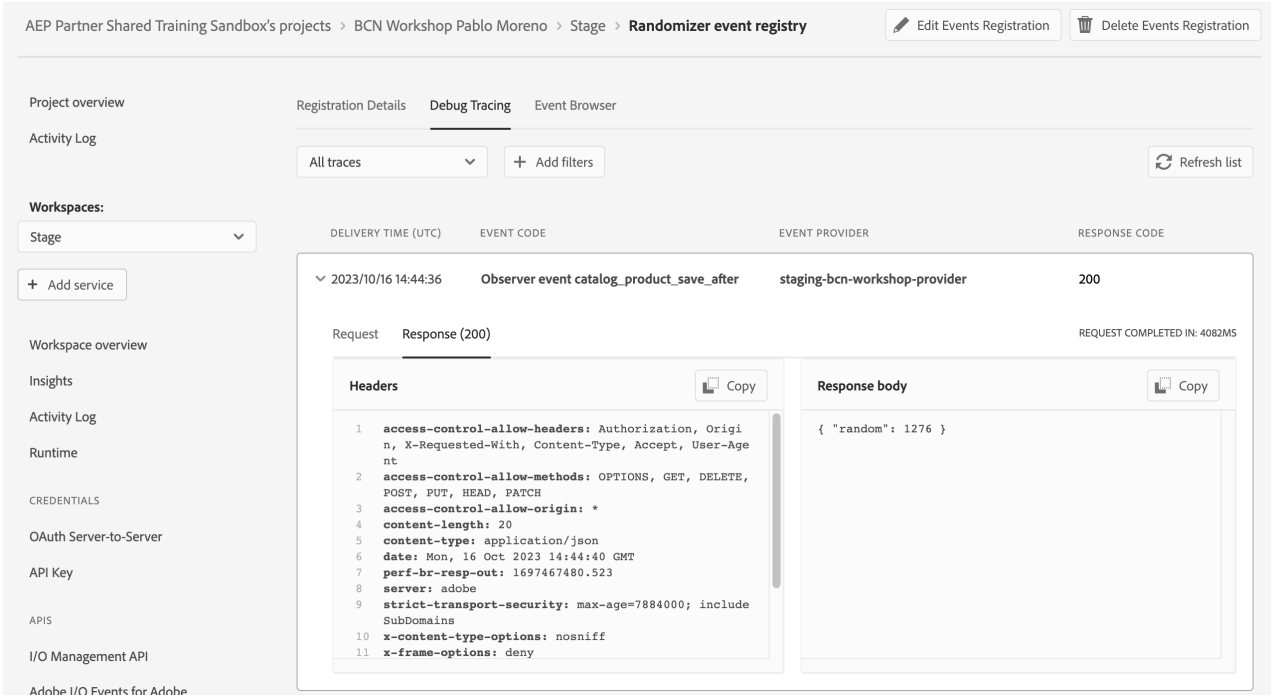
At this point, you are ready to visit the Commerce admin (https://staging-5em2ouy-efo2rc3kezxte.eu-4.magentosite.cloud/admin) and **update a product to trigger an event**.

Validation

Event reaching your project

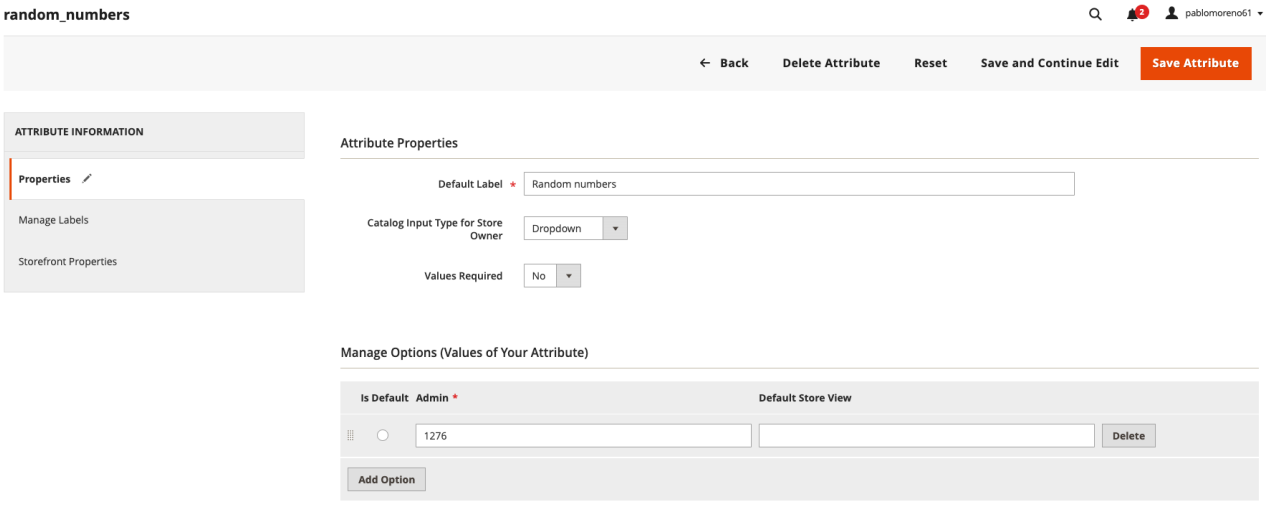


Getting a successful response



Product attribute created

Verify that the received number in the previous step is created in the Commerce Attribute page for the **random_numbers** attribute.



References

- https://developer.adobe.com/commerce/extensibility/events/project-setup/

Render App Builder extension in Commerce using Admin UI SDK

Goals

- Display a new option in the Adobe Commerce menu linking to our App Builder extension.
- Display our App Builder extension inside Adobe Commerce layout.

Publish the App Builder project

- *Note: **If you were using the PDF Embed API key generated for the local environment**, switch back to your production API key or you won't be able to the PDFs in the commerce UI.*

1. Add Nodejs library ***uix-guest*** to the package.json file to allow communication between Adobe Commerce and our App Builder extension.

```
"@adobe/uix-guest": "^0.8.2"
```

2. Run this command to install the new package

```
npm install
```

3. **Fill in the ATTENDEE_FULLNAME environment** variable:
Note: This value will be used as part of the new menu option that will be added to the Adobe Commerce menu.

```
ATTENDEE_FULLNAME=<Your name>
```

4. **Create *install.yaml* file** in the root of your project. This file is used to create and configure remote Developer Console resources.

```
$schema: http://json-schema.org/draft-07/schema
$id: https://adobe.io/schemas/app-builder-templates/1

categories:
  - action
  - ui

extensions:
  - extensionPointId: commerce/backend-ui/1
```

5. **Create *extension-manifest.yml*** file at the root of your project. This file contains data about your extension that allows Adobe Experience Platform to properly consume it.

```
{
  "name": "bcn-app-builder-workshop",
  "displayName": "BCN App Builder Workshop",
  "description": "Dive into the extensibility capabilities of Adobe Commerce",
  "platform": "web",
  "id": "bcn-app-builder-workshop-app",
  "version": "1.0.0"
}
```

6. In the ***/web-src/src/components*** directory, you should create the ***Constants.js*** file

```
/*
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/

module.exports = {
  extensionId: 'bcn_app_builder_workshop_app_' + process.env.ATTENDEE_FULLNAME.replace(' ', '_').toLowerCase(),
  rootElementId: 'backend-ui',
};
```

7. In the ***/web-src/src/components*** directory, you should create the ***ExtensionRegistration.js*** file

```
/*
```

```
Copyright 2023 Adobe. All rights reserved.
This file is licensed to you under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License. You may obtain a copy
of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under
the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS
OF ANY KIND, either express or implied. See the License for the specific language
governing permissions and limitations under the License.
*/
import { register } from '@adobe/uix-guest'
import { extensionId } from './Constants'

export default function ExtensionRegistration() {
  init().catch(console.error);
  return <></>;
}

const init = async () => {
  await register({
    id: extensionId,
    methods: {
      menu: {
        getItems() {
          return [
            {
              id: `${extensionId}`,
              title: 'App Builder Workshop ' + process.env.ATTENDEE_FULLNAME,
              parent: 'Magento_Backend::marketing'
            }
          ]
        }
      },
      page: {
        getTitle() {
          return 'App Builder Workshop ' +  process.env.ATTENDEE_FULLNAME
        }
      }
    }
  })
}
```

8. Add ExtensionRegistration to the **Routes** in the App.js file

```
<Route path={'index.html'} element={<ExtensionRegistration />} />
```

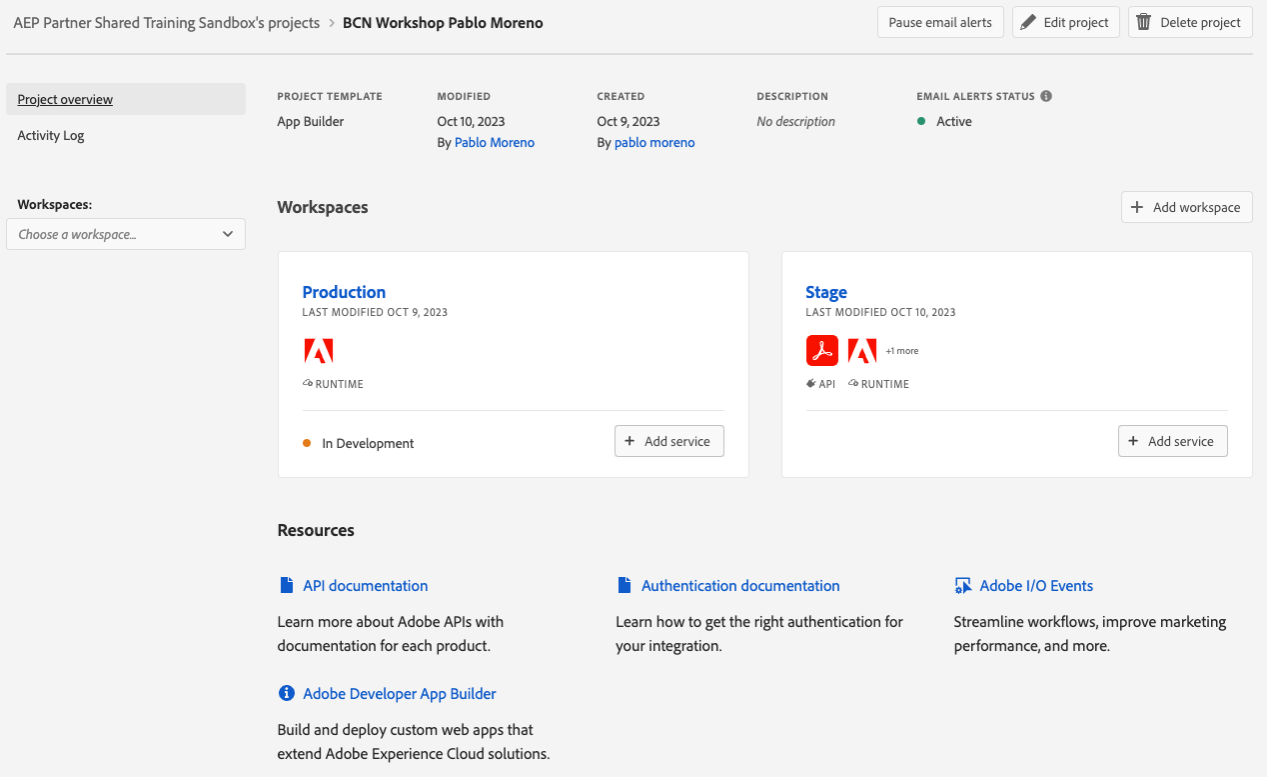
9. **Deploy the App** into the Production workspace:

   a. Deploy your code to Prod:

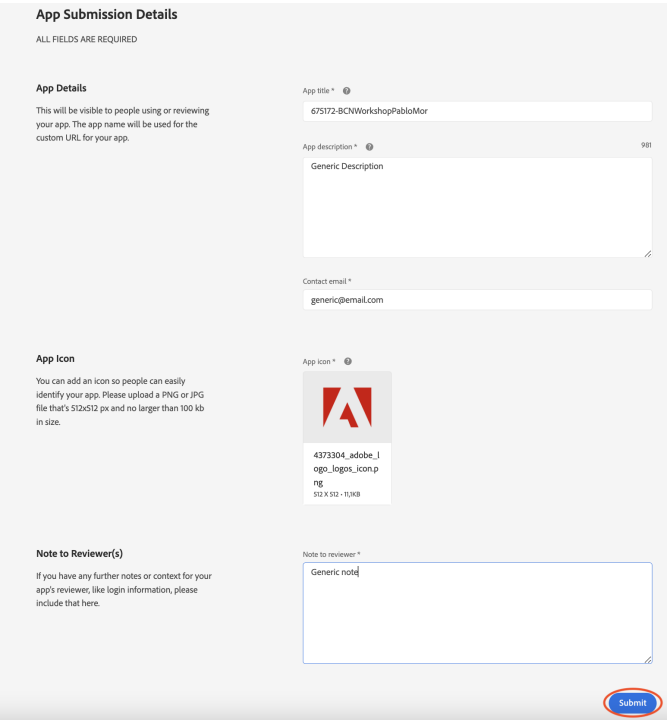
   ```
   ➜  <you-project-path> ✗ aio app deploy
   ```

   b. Check if your deployed application is working properly. Retrieve the *AIO_runtime_namespace* environment value from your .env file and append ".adobeio-static.net" to it,
   Ex: 675172-bcnworkshopjoao.adobeio-static.net

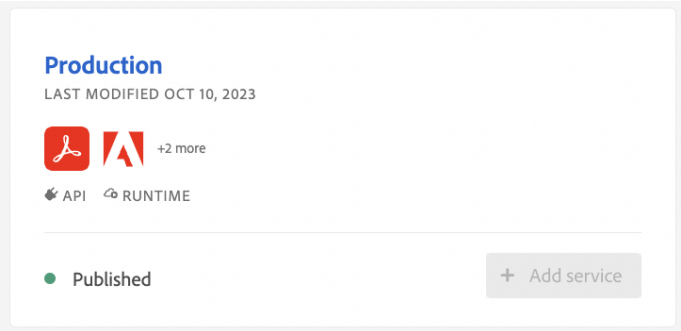
10. **Publish the App** in the Developer Console
    a. Go into the Production workspace and select **Workspace Overview** on the left menu.



b. Click on **Submit for Approval.**
c. Scroll down to fill out the App Submission form and **press Submit.**
You can get the App icon image from the /resources/ directory in your source code.

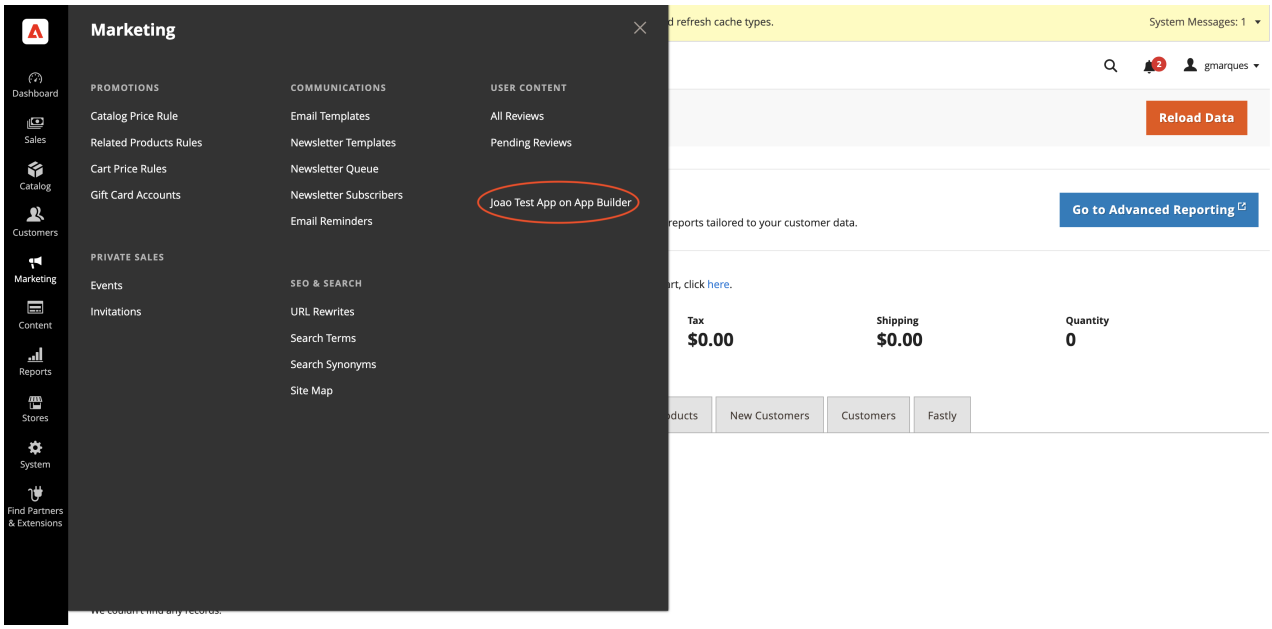


d. Your app will be submitted for approval. At this point, **ask any of the organizers for approval**.



e. Once the app has been approved, you will receive an e-mail "Adobe Developer App Builder App Approved" meaning that your application has been published. Also, in your Workspace Overview, you should notice that the **status has changed to Published**.

11. Finally, go to the Commerce admin instance to verify that a **new menu option is available under the Marketing tab**.



12. Clicking on the new option, the one with the name of the App you defined, should **display your App Builder extension embedded in Commerce**.

References

https://developer.adobe.com/commerce/extensibility/admin-ui-sdk/troubleshooting/

Sin etiquetas